A Mathematical Formalism for Linguistic Theories with an Application in Head-Driven Phrase Structure Grammar

Frank Richter

Philosophische Dissertation
angenommen von der Neuphilologischen Fakultät
der Universität Tübingen
am 23. Juni 2000

Tübingen
2000

Gedruckt mit Genehmigung der Neuphilologischen Fakultät
der Universität Tübingen

Hauptberichterstatter:	Prof. Dr. Erhard W. Hinrichs
1. Mitberichterstatter:	Prof. Dr. Uwe Mönnich
2. Mitberichterstatter:	Priv. Doz. Dr. Fritz Hamm
3. Mitberichterstatter:	Prof. Carl Pollard, Ph.D. (Columbus, Ohio, USA)
Dekan	Prof. Dr. Bernd Engler

Contents

Acknowledgments

It was in spring 1992 when I first came into contact with HPSG, and from the very beginning, my thinking about HPSG was associated with the four people who had the most profound influence on this thesis. Tilman Höhle, who taught me almost everything I know about syntax, had decided to give his first seminar on HPSG. The idea was that we would read the manuscript of what was to become Pollard and Sag 1994 in class, and try to understand this relatively new linguistic framework. For me, this became an obsession for years. Paul King, who can turn a lecture on logic into one of the most exciting events on earth, had just come from Stanford to Tübingen and taught the first of many classes to come on the formalization of HPSG. Despite his enormous efforts, I did not understand much in that first seminar, and it most certainly was not his fault. In the same semester, Carl Pollard visited Tübingen and gave a few lectures in Tilman Höhle's seminar. I still vividly remember the moment when they first met. And, as always, Manfred Sailer was in the same classes as I was, and we spent hours together laughing and wondering about the strange ways of linguistic theories. From our first years at university, when we did our computer science assignments together, right through to our latest joint papers, we have been working side-by-side, and we hope the tradition continues. Manfred is a wonderful person, a great friend, and I cannot imagine what life would be like without him. Thank you for your friendship, Manfred!

Without Tilman Höhle, Manfred, Paul, and Carl, this thesis would never have been written. They were always a part of the project, they never stopped asking questions, and they never let me get away with cheap answers. I hope that they look kindly on what I did with their ideas and suggestions.

In spring 1995, I joined the HPSG project(s) of the SFB 340, which at the time included Thilo Götz, Detmar Meurers, Guido Minnen, and Manfred. Detmar's enthusiasm for everything having to do with linguistics and his readiness to help with any kind of problem in any way he possibly can made even the most stressful times bearable. Thilo tried very patiently to explain to me over and over again what a relational extension of a constraint language is and what a feature constraint grammar means. I hope I understood at least some of what he was trying to explain.

In 1996, Gerald Penn came to join our HPSG group. It is impossible to express how much I learned from him, and the ideas that Manfred and I discussed with him about constructing a comprehensive formalism for HPSG were strengthened immeasurably thanks to his rigorous and uncompromising standards. No single little detail remained unquestioned. Thank you, Gerald, for becoming such a good friend, for those long evenings of stimulating conversations about linguistics, mathematical formalisms, philosophy, and everything else! I miss you a lot now you've left Tübingen, and I hope that we can stay in touch despite the ocean that lies between us now.

Sometime in 1998, I had the pleasure of reading the first linguistics paper that was formalized in RSRL. Adam Przepiórkowski became the first linguistic user, he kept pressing me to make RSRL as user-friendly as I could, and to make an extra effort to explain mathematical concepts in linguistic terms. As an invaluable proofreader of many papers, he spotted more mistakes than ever should have existed. But I also fondly remember our long non-linguistic conversations, especially that night in Edinburgh when we were standing outside in the cold summer air of Scotland.

Uwe Mönnich gave me the opportunity to present an early version of RSRL in his reading group that included Tom Cornell, Hans-Peter Kolb, and Frank Morawietz, all of whom gave valuable comments and made me aware of many problems that I had missed. I have greatly benefited from comments from and discussions with Mike Calcagno, Bob Carpenter, Thilo Götz, Erhard Hinrichs, Bob Kasper, Stephan Kepser, Valia Kordoni, Kai-Uwe Kühnberger, Detmar Meurers, Guido Minnen, Uwe Mönnich, Sascha Rosen, Kiril Simov, Jesse Tseng, and Shuly Wintner. I also learned a lot from the students in my seminars, from the audiences at various conferences Manfred and I went to in the past few years, and from many friends and colleagues at the Seminar für Sprachwissenschaft in Tübingen.

Even in the final stages of writing a dissertation, there is some life outside of work. The daily lunch at the Mensa was one of the things that helped me to get through the hard times at the end, and I would like to thank Frederik Fouvry, Valia Kordoni, Ania Kupść, Gergana Popova, and Jesse Tseng for all the fun we had together.

I am especially grateful to the current members of the HPSG project, Frederik Fouvry, Kordula De Kuthy, Beata Trawinski, and Jesse Tseng for putting up with somebody who in the last few months was hardly contributing anything. I am also indebted to the members of the Tübingen/Sofia CLaRK program, Gergana Popova, Atanas Kiryakov, Ania Kupść, and Kiril Simov, for their patience with me.

My sincere gratitude goes to the research assistants of the project, Michael Auth, Doris Penka, Christian Richard, and Holger Wunsch, who helped me in many ways and made staying focused on work so much easier. Our system administrator, Jochen Saile, provided much-needed technical support. Our secretaries, Rosemary Drescher and Stephanie Schwarz, kept the paperwork under control and were always ready to help with the mysteries of university administration and budgets.

I would not have been able to write this dissertation without the constant support of my family, Gertrud, Horst, and my grandmother. Finally, and most importantly, I thank Janina for her patience, love, and understanding.

Chapter 1

Introduction

Some modern linguistic theories in the tradition of early generative grammar aspire to share a high degree of formalization with those sciences that are traditionally known as the "exact" sciences, such as chemistry and physics. As the linguistic descriptions in grammars of natural languages become more and more elaborate, as the grammars themselves become more and more comprehensive, and as efforts to make use of these grammars in computational applications of language processing persist, the techniques of mathematics turn out to be an indispensable means of ensuring the internal coherence of large grammars and of expressing them with unambiguous clarity. Without the helping hand of mathematics, it would be impossible to pin down exactly what the empirical predictions of even a moderately comprehensive grammar are, but without knowing the exact predictions of a grammar, it is hard to see how a grammar could be falsified according to criteria that go beyond aesthetic preferences; and any attempt at a computational modeling of the grammar would need to cope with not having a clearly defined target.

It is no surprise, then, that the need for grammatical frameworks that possess a well-defined semantics was recognized a long time ago even outside theoretical linguistics when alternatives to context free grammars and augmented transition networks became widely used in computational linguistics. Whereas the traditional formalisms had a semantics, the new frameworks that were designed to be better suited for developing large grammars did not. Pereira and Shieber 1984 summarizes the potential problems with an informal grammatical framework concisely:

> In the absence of a rigorous semantics for a given grammar for-

> malism, the user, critic, or implementer of the formalism risks misunderstanding the intended interpretation of a construct, and is in a poor position to compare it to alternatives. Likewise, the inventor of a new formalism can never be sure how it compares with existing ones. ... Rigorous definitions of grammar formalisms can and should be made available. (Pereira and Shieber, 1984, p. 123)

The call for formal grammars is, of course, only one facet of a much more diverse picture of modern linguistic theorizing. The extent to which mathematical formalisms make sense in linguistics depends on which aspects of language are investigated, on the philosophical perspective that a particular framework takes, and also on the purpose of a research program. In important branches of modern linguistic theory such as the prolific framework of Functional Grammar, or in sub-disciplines of linguistics like pragmatics and socio-linguistics, formalization plays only a minor or no role; and even proponents of some formalist frameworks like the Minimalist Program that succeeded the original generative grammars of the late 1950s sometimes regard attempts at a mathematical reconstruction of their frameworks as premature. When I adopt the formalist perspective expressed in the quote of Pereira and Shieber, this does not imply a dismissal of all of these alternative approaches to language. It rather signals that the present work is only concerned with those areas of linguistics in which formalization does make sense.

In this thesis, I develop a rigorous definition of a grammar formalism for the Head-driven Phrase Structure Grammar (HPSG) of Carl Pollard and Ivan Sag, one of the most prominent grammatical frameworks of formal linguistics, and a framework that is also popular among many computational linguists. HPSG has always had the reputation of having fairly explicit formal foundations, and various computational platforms can be regarded as interpretations (by the implementers of these platforms) of the formal basis of HPSG, or at least as practical approximations of it. However, there was arguably never a truly comprehensive formalism that allowed for the semantic interpretation of HPSG grammars as they are typically written by theoretical linguists without the explicit purpose of the interpretation by a specific computer program in mind. For determining the meaning of these theoretical grammars, some non-trivial transformation steps that must be performed by a human interpreter of the grammar are invariably necessary to bring the grammar into a format that conforms to any existing computational formalism. Such a transformation of a grammar is error-prone, it can be very difficult, and usually it will not preserve the exact meaning of the original grammar. While grammar transformations that change the meaning of grammars might to some extent be necessary for efficient language processing, this situation is thoroughly unsatisfactory from the point of view of theoretical linguistics. The formalism that I will present eliminates the transformation and interpretation step for the principles of the grammar of English that Pollard and Sag list in the appendix to Pollard and Sag 1994, where the (latest) framework of HPSG was first published, and for all HPSG grammars that do not stray too far from the outline of a formalism of that book.

Although the formalism that I will introduce is deliberately constructed as a formalism that can express HPSG grammars, it is by no means a formalism whose application is restricted to HPSG. Since it was developed with HPSG in mind, it has the advantage of incorporating the experiences and demands of a large number of linguists working within that framework; and existing grammars illustrate the usefulness of the formalism and how grammars can be written in it. However, as a mathematical formalism it is not committed to the theories of one linguistic framework but fit to express arbitrary linguistic theories, provided they accept certain theoretical premises. At a very abstract level, the formalism comprises a small number of ontological assumptions about natural languages, and any linguist who accepts these assumptions and finds the descriptive means that the formalism provides appropriate to his or her purposes can express their grammars in the formalism, independent of whether they bear any resemblance to HPSG grammars or not. The relationship between a grammar and a natural language can be expressed as the conditions under which a grammar is true of a natural language.

Interestingly enough, the fact that the formalism is not committed to HPSG but rather to certain assumptions about the meaning of grammars can be—and will be—illustrated within HPSG itself. The HPSG framework of Pollard and Sag 1994 has a predecessor in the unification-based HPSG framework of Pollard and Sag 1987. While the actual grammars written in unification-based HPSG look very similar to grammars written in the succeeding framework, the assumptions of the two HPSG frameworks about the meaning of grammars and the nature of natural languages are radically different. For example, whereas unification-based HPSG grammars crucially rely on a mathematical construct called *feature structures*, the later framework is not committed to feature structures, and the meaning of its grammars can be explained without any reference to feature structures. As a consequence of their superficial similarities, it is not difficult to bring a unification-based HPSG grammar into the syntactic form of the succeeding kind of HPSG grammars; but it will then mean something completely different. Conversely, a grammar that is structured and notated in a way that has little or nothing to do with the structure and notation of the grammar of Pollard and Sag 1994 can be expressed in the formalism, if its intended interpretation agrees with the premises about the meaning of a grammar of the formalism of this thesis, and if the grammar uses no constructions that are beyond the expressive power of the formal languages of the formalism.

In this very precise sense, the formalism that I will discuss is not only the formalism of (the current version of) HPSG, but a mathematical formalism for a certain class of linguistic theories to which the theory of English of Pollard and Sag 1994 and many other grammars belong. The formalization of HPSG grammars is one conceivable application of the formalism.

1.1 A Formal Framework for Linguistic Theories

A formal framework for linguistic theories of the kind that I envisage here must consist of several components. First of all, it must provide a definition of what exactly a grammar is. From a formal framework, we would expect that it defines an infinite number of possible grammars. Each of these grammars must comprise expressions of some formal language that serve to notate generalizations about the linguistic observations about the natural language under investigation. A denotational semantics assigns each expression of each formal language a meaning. Based on the meaning of all of the expressions in a grammar—which are usually called the *principles* of the grammar—we must be able to determine the meaning of the grammar. Formulating a model theory that tells us what the intended models of a grammar are is one way of determining the meaning of grammars. It means that there is a clearly defined collection of structures that interpret the possible grammars. Finally, a formal framework should provide an explication of the relationship between the intended model(s) of each grammar and the empirical domain that the grammar considers. If the framework did not explain this relationship, the grammars would be of no scientific value, because we would not know how to falsify the predictions of a grammar, since we would not know which empirical observations could possibly falsify them.

Of course, an infinite number of linguistic formalisms could be designed that meet these requirements, and almost all of them would almost certainly be useless, no matter what their mathematical virtues are. Linguists are unlikely to adopt a formal framework that does not prove its linguistic viability. A new formalism needs a good *raison d'être* to be accepted by the linguistic community. A demonstration that there already exist grammars that a new formalism subsumes can be a good justification. If this justification is not available, a new linguistic formalism should at least be accompanied by theoretically interesting grammars that it can express, and it should show how it expresses them.

These practical considerations make HPSG a good target for a new mathematical formalism, because it will obviously provide a significant number of fairly diverse grammars that can exemplify its application. Not only is HPSG a thriving linguistic framework, it also has some other features that make it an attractive candidate for a comprehensive formalization: Firstly, HPSG analyses cover several areas of linguistics. Although most of the research in HPSG focuses on syntax, it has also been applied to semantics, morphology, phonology, and to pragmatics. Secondly, most of the HPSG literature does not depart too far from the fundamental ideas of the framework as it was originally presented, which makes it reasonable to expect that a formalization of Pollard and Sag 1994 extends to most of it. Last but not least, while Pollard and Sag sketch the mathematical foundations of HPSG informally, their description is precise enough to make feasible a serious attempt at a formalization of the basic intuitions that underly their ideas. In fact, there exist already a number of clearly related formalisms and of partial formalizations of HPSG that can serve as promising sources for such an enterprise. I will choose one of those formalisms as my starting point, and we will see that it already contains most of the ingredients for whose necessity in a formalization of HPSG I will argue.

At the beginning of this enterprise one might ask if it is really a good idea to design a new formalism for expressing HPSG grammars instead of taking an existing standard logic such as first-order predicate logic and trying to show how particular HPSG grammars can be expressed with it. This method would certainly have the advantage that we would know the formal, model-theoretic, and computational properties of the selected formalism from the very beginning. For example, Johnson 1995 shows how many feature structure-based systems can be axiomatized in the Schönfinkel-Bernays class of formulae of standard first-order logic, which implies that the satisfiability problem of these systems is decidable. If (a fragment of) first-order logic were not sufficient for HPSG, we could still choose a more expressive standard logic. The reason I do not pursue this route is that linguistic theories such as HPSG envisage customized notations and customized model theories that support certain intuitions that are very important to the linguist. Only if a formalism is designed to capture these very intuitions directly, can we be sure that the grammars expressed in that formalism really do express what the linguists want to say. This way of proceeding keeps the notation and the model theory of the formalism simple and transparent for the linguist. Relating the resulting formalism to standard logics may then be an important second step in a mathematical investigation of its properties. Before this investigation is worthwhile, however, we must make sure that the formalism accomplishes what the linguists want to do with it in their own terms.

The working strategy of providing a formalism that takes its justification from specifically linguistic needs determines the general perspective that I will adopt in this thesis. I am interested in the question of what kind of formalism is appropriate to express the grammars that linguists actually write in the HPSG framework. How can the meaning that they ascribe implicitly and intuitively to their grammars be captured formally? What are the linguistic concepts that need to be expressed by mathematical structures? And which mathematical structures correspond to which linguistic intuitions? How should the formalism be designed such that each of its mathematical constructs can be justified as precisely and adequately capturing the linguistic concepts that the linguists have in mind when they write their grammars?

This perspective also implies that, in contrast to many other investigations in formal linguistics, particularly in the context of HPSG and of so-called feature structure-based grammar formalisms, I am not at all concerned with issues of computation. As I will argue below, I see questions of computation as being independent of the initial design of a formalism that supports the precise articulation of grammars as they are conceived and written by theoretical linguists. The initial goal must be to find out what the meaning and the predictions of the theoretical grammars are. Just as determining the relationship of a linguistic formalism to standard logics follows after the design of an adequate formalism, I regard determining the relationship of a theory of language to a theory of language processing as a separate research project that may follow the design and application to empirical linguistic research of a linguistically adequate formalism. HPSG grammars did and do exist independent of whether there is a computational theory for them. If we manage to establish a mathematical interpretation of these existing grammars, we may hope that the task of developing a theory of language processing for them will become, if not easy, then at least much more manageable.

The primary goal of this thesis is thus to design a customized formalism that makes it simple for a linguist who shares the philosophical assumptions of HPSG about the relationship between a grammar and a natural language to express exactly what he wants to say in a straightforward, perspicuous, and very natural way. The customized model theory that belongs to the formalism will tell us which structures the resulting grammars specify. This will allow for a rigorous mathematical investigation and comparison of the interpreting structures of different grammars. From this we may learn more about which properties grammars that superficially look quite different share. As soon as unambiguously specified mathematical structures of a number of grammars become available, their properties can be studied in the search for linguistic universals. The models of the grammars of natural languages that linguists write may be found to exhibit common properties that help to develop theories of efficient language processing that can supplement the declarative specification of these structures. Moreover, the natural but rigorous formalization of HPSG grammars makes their empirical predictions mathematically precise, and they become as open to empirical falsification as possible and desirable for a scientific theory, since counterexamples can no longer be explained away as unsolved problems of the underlying formalism.

1.2 Linguistic Analyses

This thesis does not contain any genuinely new empirical linguistic analyses. Its first and foremost objective is to provide a formalism in which previously informally presented HPSG grammars can easily, faithfully, and rigorously be formalized. However, in order to show that the formalism meets this goal, I must show how it expresses all aspects of theoretically interesting HPSG grammars. I will demonstrate the adequacy of the formalism by a complete formalization of all of the principles of the grammar of English contained in the appendix to Pollard and Sag 1994. This formalization will provide ample opportunity to explore the use of the various syntactic constructs that the syntax of the formalism defines, and I will argue that all of them are needed in a faithful and precise rendering of the principles of the grammar of Pollard and Sag. We will also see that their informally given principles can directly be expressed in the formal language without rephrasing them. We can thus be sure that the formalization of the grammar really expresses what the authors wanted to say. Since the empirical coverage of Pollard and Sag's grammar of English is broad, the formalization gives us a first idea of how a broad range of data and phenomena of linguistic interest can be captured in the new formalism. Moreover, the concrete application of the constructs to familiar principles from the linguistic literature gives an extensive illustration of the technical tools, and it is an opportunity to study their potential and their adequacy to the task at hand.

It can be argued that the formalism that I present is, in a sense, not a new formalism, since it is the formalism that Pollard and Sag and other HPSG linguists have been using implicitly for a long time. It can make a big difference, however, if linguists use a formalism only on the basis of an intuitive understanding of the underlying foundations (as Pollard and Sag do), or if they are in a position to exploit the expressive means of explicitly defined formal constructs with well-known properties and a clear semantics. To illustrate this point, I will summarize recent work in HPSG that has been carried out on the basis of the formalism of this thesis. The three areas of research that I will consider are a non-configurational theory of case assignment, a new approach to linearization grammars, and work on the specification of standard semantic representation languages as object languages of HPSG grammars. The complexity of the principles that these lines of research require demonstrates that in all three cases it is necessary to have a clear semantics of the principles of the grammars and to know the properties of the structures that the grammars specify. It is hard to imagine how grammatical principles of the necessary degree of complexity could be made precise without a mathematical formalism for linguistic theories. In the case of semantic representation languages, we must even be able to prove that the specified structures can be viewed as representations of the intended languages. Without exact mathematical definitions, it would be impossible to show whether or not the envisaged languages of semantic representations can be defined as object languages of grammars in our grammatical framework.

1.3 Contribution of this Research

I begin with an introduction to the two ways of characterizing natural languages with grammars that were consecutively considered in the evolution of HPSG. In Chapter 2, a contrastive overview of the two types of HPSG formalisms that Pollard and Sag put forth in their two books on HPSG will exemplify how very similar mathematical constructs can be used for significantly different purposes. This observation will later help to make a clear distinction between mathematical constructs and the linguistic concepts that they are meant to express. The older formalism is unification-based and views language in terms of algebraic operations on pieces of partial information. The succeeding formalism is constraint-based. Its main characteristic is the use of a formal description language for the specification of natural languages as collections of total objects. In a unified terminology and notation, I will explain how similar mathematical structures play an important role in spelling out the technical details of the basic ideas underlying the two distinct formalisms, and why it is important for the linguist to distinguish between the two approaches. Since the focus of the present thesis is not on unification-based grammar formalisms, no attempt will be made to present unification-based HPSG in a version that reflects the current state of the art. Instead, I will restrict myself to an exemplary mathematical reconstruction that stays close to the original (informal) presentation of Pollard and Sag 1987. In the course of the discussion of unification-based and constraint-based HPSG, all mathematical constructs that will be needed in the succeeding chapters are introduced, and their roles in the two different formalisms are investigated.

The remainder of this thesis is concerned with the development and the linguistic application of an extension of the basic formalization of Chapter 2 of the constraint-based approach. In Chapter 3, I will augment the formal languages of the basic formalism with bounded existential and universal quantification and with relations. The resulting new formalism is designed as a comprehensive formalism for expressing the complete grammar of English of Pollard and Sag 1994 and most grammars of the constraint-based HPSG literature. I will argue that bounded quantification and relations that are defined on the basis of bounded quantification are necessary to express HPSG grammars as they are formulated by theoretical linguists. Earlier formalizations of HPSG that omit bounded quantification and relations cannot be complete.

In previous work by King 1999 and Pollard 1999, it was shown that the basic feature logic formalism that I introduce in Chapter 2 supports different plausible and formally rigorous interpretations of the meaning of grammars. I will show that, despite the substantive additions to the syntax and semantics of the feature logic formalism that King and Pollard use, the new formalism of Chapter 3 preserves their respective explanations of the meaning of grammars. The extended formalism for constraint-based HPSG provides mathematical structures that express the philosophical notions of the tokens of a language and the object types of a language, and I will explain how the two kinds of structures are formally related. The investigation of the mathematical structures that interpret the grammars of the extended formalism confirms the insight, previously expressed by other researchers, that the interpretation of constraint-based HPSG grammars is not tied to feature structures, and a formalism of constraint-based HPSG does not necessarily comprise a feature logic in the classical sense of the word.

For practical purposes, the syntax of the languages of a formalism is almost as important as the semantics. To facilitate the application of the extended formalism to existing HPSG grammars, I will furnish it with a syntax of a class of languages of attribute-value matrices of the kind that are employed in the linguistic literature. These languages obviate the need to translate the grammars that linguists write into an unfamiliar notation. Conversely, they make grammars written in the new formalism immediately accessible to linguists that are versed in HPSG-style grammars.

Chapter 4 substantiates the claim that the extended formalism is appropriate to the formalization of HPSG grammars by a formalization of the grammar of English presented in Pollard and Sag 1994. The formalization of the grammar of Pollard and Sag will give rise to a thorough discussion and justification of all syntactic and semantic aspects of the logical languages of the new formalism, and to a close investigation of what they accomplish in the formalization of a grammar of a natural language. In order to prove that the formalism truly captures the entire grammar of Pollard and Sag, all principles of their grammar that are not discussed in Chapter 4 are formalized in Appendix C.

In the last chapter, I will finally turn to research in HPSG that has already been carried out on the basis of the formalism of Chapter 3. Summarizing some selected results of that work, we will see how linguistic analyses can benefit from a totally explicit formulation of linguistic principles in a mathematically rigorous framework for linguistic theories. In its ultimate consequence, the application of the present formalism to concrete analytical problems beyond classical HPSG may be seen as a testing ground for the question of what can be gained from a totally formalized linguistic theory as compared to a framework that leaves its foundations on a more intuitive level.

Pereira and Shieber (1984, p. 128) expressed high expectations in mathematically rigorous approaches to linguistic theories when they wrote that "the *linguistic discipline* enforced by a rigorous approach to the design and analysis of grammar formalisms may make possible a hitherto unachievable standard of research in this area." In this spirit, the goal of the present thesis is to contribute to the enterprise of a better understanding of the mathematical foundations of modern formal linguistic theories, and to help find out whether rigorous formal foundations can improve the standards of research in HPSG.

Chapter 2

Formal Foundations I

In this chapter, I discuss two formalizations of two fundamentally different HPSG formalisms. I will call the two formalisms HPSG 87 and HPSG 94, respectively. Their names derive from the books in which they were first presented, Pollard and Sag 1987 and Pollard and Sag 1994. When I say that these two books present two grammar formalisms, HPSG 87 and HPSG 94, that statement is in fact not entirely accurate. Both Pollard and Sag 1987 and Pollard and Sag 1994 are at pains to characterize their envisaged formalisms up to a high degree of precision, but neither of them is mathematically rigorous. As usual with ultimately informal characterizations, this leaves some room for interpretation, and many major and minor details are left to be worked out by the reader. In both books, however, the description of the intended formalism and of the underlying intuitions as well as their illustration with substantive fragments of English are precise enough to decide whether a given mathematically rigorous formalism is a contender at being an appropriate formalization of HPSG 87 or of HPSG 94.

The discussion of HPSG 87 serves several purposes: Firstly, it represents a conception of language that is different from the one that underlies HPSG 94. This will give me a convenient point of comparison in the following discussion of HPSG 94. Contrasting the two perspectives will shed more light on both of them. Secondly, HPSG 87 serves to introduce a number of mathematical constructs that play an important role in the later discussion of the HPSG 94 formalism. Having them already presented in the context of HPSG 87 will make it easier to focus on the issues that are important for HPSG 94 itself instead of dividing the attention between the introduction of new mathematical constructs and the explication of their linguistic significance. Thirdly, HPSG 94 is a descendant of HPSG 87, and, especially in the initial phase after the shift from the old formalism to the new one, HPSG 87 influenced the perception of linguists of HPSG 94, its terminology, and the thinking about the models of HPSG 94 grammars. Finally, the two formalisms will illustrate how closely related mathematical structures can serve to spell out very different underlying intuitions about the purpose of grammar and the nature of language. We will, thus, see that it is very important even for a linguist who only takes a casual interest in formalisms to keep HPSG 87 and HPSG 94 and their respective terminologies apart. They conceive of language and the task of grammar in very different and logically incompatible ways. By choosing one of the two formalisms, linguists make a substantial decision about their view of the nature of language and about the meaning of their grammars.

HPSG 87 sees the task of a grammar in specifying the information that a mature speaker of a language has about his or her language. By making explicit the information that the members of a language community share about their language, a grammar indicates which resources are available to speakers when they speak or process their language and how communication is possible through the use of that information. The means of specifying information are mathematical entities called *feature structures*, and at the heart of an HPSG 87 grammar is a set of feature structures that represent information about the natural language under consideration. Crucially, feature structures are representations of *partial* information. Some feature structures contain more information about language than others, and the combination of two feature structures—which is called their *unification*—results in a feature structure that systematically represents at least as much information about language tokens as either one of the two alone does. A grammar of a particular language, then, specifies a set of feature structures that represent the partial information available to speakers about their language. Some of that information is universal across all languages and thus held in common by all speakers of any natural language, some of it is idiosyncratic to a given language. In language processing, all of that information can, incrementally or in parallel, be brought to bear, possibly together with other, non-linguistic information. Notice that an HPSG 87 grammar deliberately does not make any commitments about the nature of the entities that constitute a natural language. They might be total objects in the brain (or somewhere else), they might be partial objects in the brain (or somewhere else), or they might be something else entirely and not be amenable to a characterization in terms of any kind of objects. No commitment is necessary, because all an HPSG 87 grammar directly talks of is information about language, not the entities of the language themselves.

The picture of HPSG 94 that we find in Pollard and Sag 1994 is entirely different. According to Pollard and Sag 1994, the task of an HPSG 94 grammar is the specification of the collection of object types of a natural language.[1] The object types are modeled by feature structures. Feature structures are regarded as idealizations of equivalence classes of possible well-formed linguistic entities. Due to their different purpose, the feature structures of HPSG 94 have properties that distinguish them from the feature structures of HPSG 87. The HPSG 94 feature structures are not partial. They are complete representations of idealized linguistic entities. Subsumption and unification are no longer needed because the feature structures do not represent partial information that can be made more specific by adding more information. Instead, feature structures are now the possible object types of the natural language that the grammar captures. While a definite commitment as to the exact ontological status of these object types is avoided in Pollard and Sag 1994, it is clear that feature structures as object types play a new role in the grammar. When the grammar talks about feature structures, it talks about idealizations of equivalence classes of possible entities of the language and not merely about information about the entities of the language. A grammar specifies the intended collection of object types constituting the language by using descriptions of a logical description language that consists of so-called attribute-value matrix (AVM) diagrams. Partiality in grammar only occurs as partial descriptions of complete feature structures. That means that if a need for a notion of information ordering arises at all, it can only be defined among descriptions. With a set of descriptions, conventionally called the principles of the grammar, a grammar of a natural language characterizes precisely and unambiguously the collection of feature structures which the grammar admits.

In subsequent work, King 1999 and Pollard 1999 proposed different accounts of the meaning of HPSG 94 grammars that no longer use the feature structures of Pollard and Sag 1994 as modeling structures. While they differ

[1]In fact, Pollard and Sag 1994 is not always as clear about the precise nature of the entities in the denotation of a grammar as my summary might suggest. In order to avoid a tedious and ultimately unfruitful analysis of its occasionally inconsistent terminology, I adopt the uncontroversial interpretation of the text that informs the discussion of the original HPSG 94 formalism in King 1999 and in Pollard 1999.

from the account of Pollard and Sag 1994 in philosophically relevant ways, they retain the idea that an HPSG 94 grammar specifies a collection of total objects. The difference resides in the mathematical structure and in the ontological status that they ascribe to these objects. We will see later that all three accounts are mathematically closely related.

The brief summaries of the most prominent features of HPSG 87 and HPSG 94 indicate in which respect the two formalisms differ. HPSG 87 belongs to a family of formalisms that construe a natural language as a system of partial information about linguistic objects, and delimit such a system algebraically. It is a typical example of a *unification-based* or *information-based* grammar formalism. HPSG 94, on the other hand, construes a natural language as a system of possible linguistic objects, and delimits such a system model theoretically. Accordingly, it could be called an *object-based* grammar formalism. More commonly, it is nowadays called a *constraint-based* grammar formalism. As pure formalisms, the two do not have much in common. However, there are several reasons why, to linguists, they have sometimes looked deceptively similar on the surface. The first is their application. Both of them were originally applied to the formulation of a theory of English, and the theory formulated in the first formalism is clearly a predecessor of the theory formulated in the second, while both theories of English share many assumptions about the structure of English signs. Secondly, the AVM notation in which the grammars in Pollard and Sag 1987 and Pollard and Sag 1994 are presented is essentially identical. This tends to obscure the fact that the AVMs have different meanings in the two formalisms. Another source of mixed terminology in the literature is the extensive application of HPSG in the computational implementation of grammars. Implementations already play a role at the earliest stages of HPSG, and they have their roots in unification-based grammar formalisms.[2] ALE (Carpenter and Penn, 1996), which is a particularly influential computational system for HPSG grammar implementation, follows the information-based paradigm of Carpenter 1992. This has led to a situation where the terminology of HPSG 87 continues to be used in the context of grammar implementations, while linguistic analyses often refer to the HPSG 94 formalism of Pollard and Sag 1994. Finally, both formalisms employ feature structures to define the meaning of grammars, although they use different kinds of feature structures for different purposes. I will define and investigate the respective kinds of feature structures in Sec-

[2]See Pollard and Sag 1987, pp. 49–50.

tions 2.1 and 2.2. In Section 2.2 we will also see that feature structures are not a necessary part of a formalism of HPSG 94. The different functions of feature structures in unification-based HPSG and in constraint-based HPSG stress the importance of separating the mathematical constructs from what they are supposed to signify in the scientific description of language.

The formalisms of HPSG must be distinguished from the theories of language that are expressed in them. In his class lectures on HPSG, Carl Pollard coined the terms HPSG-1, HPSG-2, and HPSG-3 to single out different versions of HPSG. HPSG-1 means HPSG in the style presented in Pollard and Sag 1987. HPSG-2 is HPSG as presented in Chapters 1–8 of Pollard and Sag 1994, and HPSG-3 is HPSG as presented in Chapter 9 of Pollard and Sag 1994. HPSG-1 assumes the unification-based formalism, and HPSG-2 and HPSG-3 assume the constraint-based formalism. To obtain a similar but more fine-grained terminology that separates the theories of English of the two books by Pollard and Sag from the formalism in which they are expressed, one could use the terms HPSG-A, HPSG-B, and HPSG-C instead, where HPSG-A designates the theory of English of Pollard and Sag 1987, HPSG-B is the theory of English of Chapters 1 to 8 of Pollard and Sag 1994, and HPSG-C refers to the revised theory of Chapter 9. Whereas the differences between the formalisms concern the meaning of the grammars of English, the difference in the theory of English consists in different assumptions about the structure of English phrases and about the principles that predict which (empirically observable) expressions are well-formed expressions of English and which ones are not. Assuming for a moment that HPSG 87 and HPSG 94 provided the means to formalize all relevant linguistic principles equally faithfully, it would be possible to express all three different theories of English, HPSG-A, HPSG-B, and HPSG-C, in either one of the two formalisms. Each theory of English should then still predict the same expressions as the expressions of English; but the two interpretations of the same grammars of English would disagree about what an expression of English is and how it is characterized by the grammars.[3]

A number of formalisms from the two families that I have distinguished

[3]The assumption that the principles of the three grammars of English could equally well be expressed in the two formalisms is, of course, an idealization. As we will see, the concrete formalism of HPSG 87 discussed in Section 2.1 does not formalize all concepts whose integration into the HPSG 94 formalism I will investigate in Section 3.1. Whenever relations or quantification are needed in HPSG 87 to express a principle, they could be approximated at best.

have served as a source of inspiration for HPSG 87 and HPSG 94, and others have formally reconstructed variants of them. The unification-based approach of HPSG 87 belongs to the tradition of the feature structure formalisms of Rounds and Kasper 1986, Kasper and Rounds 1986, Moshier 1988, Pollard 1989, and Carpenter 1992. Since in this thesis, I am not interested in the information-based approach to grammar *per se*, I will not review the extensive literature devoted to it.[4] My exposition of HPSG 87 will be a generalization of the formalization that it receives in Chapter 2 of King 1989, because it is the only one that is directly concerned with reconstructing the informal presentation of Pollard and Sag 1987, whereas the other formalisms are more loosely related to HPSG 87. The object-based approach of HPSG 94 is close to the view of grammars that is taken in Johnson 1988 for Lexical Functional Grammar and in Smolka 1992 in the context of an investigation of constraint solving methods for the processing of constraint grammars. HPSG 94 owes many details to the suggestions developed in Chapter 3 of King 1989 in response to what King perceives to be problematic about the unification-based approach of HPSG 87. King 1999 and Pollard 1999 take the logical language of King 1989 as their starting point of an investigation of the model theory of HPSG 94. My discussion of HPSG 94 will initially follow the formalism that King and Pollard adopt, because that formalism, unlike other logics of descriptions, was expressly designed for HPSG and, therefore, implicitly incorporates several HPSG-specific mechanisms that others do not. Moreover, it arguably represents the current standard view of HPSG 94.

In an interesting line of research that is more loosely related to HPSG, Marcus Kracht, Patrick Blackburn, and others, investigated the relationship between feature logic and modal logic (e.g., Blackburn, 1993, 1994; Kracht, 1995). They showed that feature logics can be understood as modal logics. This insight makes a wealth of mathematical results about modal logics applicable to the investigation of the properties of different kinds of feature logics. Blackburn and Spaan 1993 uses the connection between feature logics and modal logics in order to prove a number of computational complexity results about feature logics with description languages of differing expressive power. Reape (1991, 1994b) added n-ary functions and relations to a feature value logic defined in a modal framework in an attempt to formalize HPSG 87, including the relational extension of the feature structure formalism to which Pollard and Sag appeal. Unfortunately, the stage that this work reached

[4] For a historical overview, see Smolka 1992, pp. 82–84.

represents work in progress. By the author's own assessment (Mike Reape, personal communication, 1999), it was an attempt to provide a formalism for HPSG 87 that was designed to be equally comprehensive as the formalism that I propose for HPSG 94 in Chapter 3. I will have nothing to say about the relationship between the HPSG formalisms that I will discuss, and modal logic.

The present chapter is organized as follows. Section 2.1 introduces the information-based formalism of HPSG 87 in more detail. I will define three kinds of feature structures and explain how the meaning of a grammar is related to the specification of a set of one of the three kinds of feature structures. I will then briefly discuss the consequences of the intuitionistic approach of HPSG 87 for the linguist and point to the literature that subsequently improved and extended the sketch of a formalism of Pollard and Sag 1987. Section 2.2 introduces the mathematics of the object-based formalism. Close relatives to two of the previously discussed three kinds of feature structures return in three different ways of defining the meaning of HPSG 94 grammars. One of them will add one more kind of feature structures to the picture. I will discuss the relationship between the three ways of defining the meaning of HPSG 94 grammars and show how they fit into a bigger, unified picture. We will see that, mathematically, they are really only three sides of one single formalism, although they express different philosophical standpoints. I will then proceed to criticize that formalism as being insufficient for a straightforward logical specification of some important aspects of HPSG 94 grammars. Although it comes close in many respects, it is not fit for an entirely accurate and faithful formalization of the grammar of Pollard and Sag 1994 and other HPSG literature that adopts HPSG 94, because it lacks two important concepts that are needed in a declarative and formally rigorous specification of existing grammars, namely bounded quantification and relations. I will therefore claim that it can only be seen as a first approximation to the formalism that the authors of HPSG 94 and other linguists working in their framework use implicitly. This claim will lead to a considerable extension of the formalism in Chapter 3 that results in a new formalism that remedies the observed discrepancies. The application of the extended formalism to the grammar of Pollard and Sag 1994 in Chapter 4 yields a first justification of the development of that formalism: It is one way of spelling out the formal foundations of HPSG 94. A second justification will be added in Chapter 5: It also provides a highly flexible logical language that is tailored to the needs of linguists and linguistic description. This claim will be substantiated by discussing some recent linguistic analyses beyond classical HPSG that have been developed and formalized in the extended formalism.

2.1 Language as Partial Information

The discussion and formal reconstruction of HPSG 87 serves to set the stage for my central topic, the investigation of the formal foundations of HPSG 94. I will focus on the intuitive ideas that underly HPSG 87 and how they are captured by mathematical entities called *feature structures*. In the context of HPSG 87, I will introduce three different kinds of feature structures. They are called *concrete feature structures*, *abstract feature structures*, and *disjunctive feature structures*. I will explain how the three kinds of feature structures are related to each other, and I will motivate why they occur in HPSG 87 by the linguistic intuitions that feature structures are supposed to express. We will see that many variants of each kind of feature structure are conceivable. The comparison of HPSG 87 with HPSG 94 will show how these differences matter for the linguistic formalisms in which the different variants of feature structures are used.

The leading questions of the present section are: What are the feature structures of HPSG 87? Which role do the feature structures play in the theory of language? How do they fulfill their task? The answers to these questions and familiarity with the different kinds of feature structures of HPSG 87 prepare the later investigation of the possible functions of feature structures in HPSG 94, where they serve purposes that have little or nothing to do with their role in HPSG 87. The fact that HPSG 94 is historically a descendant of HPSG 87 is an important factor for the occurrence of a particular kind of abstract feature structures in one of the explanations of the meaning of HPSG 94 grammars that I will discuss in Section 2.2. It is significant that this explanation of the meaning of HPSG 94 grammars was the first to be given. The subsequent explanations move further away from the feature structures of HPSG 87 without giving up any substantive assumptions that are inherent to HPSG 94. This development illustrates that some aspects of early expositions of HPSG 94 are easier to appreciate with some background knowledge about the technicalities and the motivation of its predecessor, which sometimes strongly influenced the perception of the object-based formalism among linguists.

To forestall possible misunderstandings, it is important to be entirely clear about the purpose of my formalization of HPSG 87, what it is supposed to be, and what it is *not* supposed to accomplish. Pollard and Sag (1987, p. 27) say explicitly that they do not intend to give a detailed technical characterization of their envisaged formalism. In fact, the literature that clarified the formal foundations of HPSG 87 succeeded the publication of Pollard and Sag's book (Moshier, 1988; Pollard, 1989; King, 1989; Pollard and Moshier, 1990; Carpenter, 1992). As is natural in the course of the development of a formalism that starts from a largely informal presentation of ideas, many improvements and generalizations of the initial proposals of Pollard and Sag were discovered that led to mathematically more elegant formalisms for information-based grammar than what they had described. Overall, the technical part of Pollard and Sag 1987 was not influential for the mathematical development of the information-based approach to grammar. When I return to a formalism that is much closer to the sketch of an information-based formalism of Pollard and Sag 1987 than the subsequent mathematically oriented literature, I do so deliberately and for several reasons: First, linguists are likely to be more familiar with Pollard and Sag 1987 than with the more remote literature (for linguists) on the formal foundations of information-based formalisms. It is easier to explain the genuinely *linguistic* motivation of the information-based approach with reference to the more familiar original ideas and examples of Pollard and Sag than with reference to comparatively unfamiliar mathematical constructs that do not exactly match the original, linguistic exposition. Second, the return to an older, or even outdated, formalism is not in conflict with the goal of my thesis, since I will not pursue the information-based approach after Section 2.1. Moreover, the subsequent mathematical literature had little effect on the perception of linguists of HPSG 87. Third, despite the lack of possible mathematical generality, a formalization of Pollard and Sag 1987 is very straightforward and simple and does not require mathematical machinery that I will not need in the investigation of the formal foundations of HPSG 94. This would not be the case for more recent information-based formalisms, whose generality requires additional, domain-theoretic mathematical concepts. Sticking closer to Pollard and Sag's original work thus avoids the introduction of technical material that is not necessary to understand all aspects of HPSG 94 and its historical relationship to HPSG 87. Finally, the concepts of HPSG 87 can easily and without any loss of clarity be illustrated with a formalization that is close to Pollard and Sag 1987, and the exposition of the linguistically relevant aspects might even gain from its greater mathematical simplicity. Where appropriate, I will refer to the literature that clarified issues that were still unclear in Pollard and Sag 1987, and the reader interested in formal issues of HPSG 87 is invited to follow the references.

In summation, I will not present an information-based formalism for HPSG that takes advantage of the readily available insights into possible generalizations of the formalism that is sketched in Pollard and Sag 1987. My presentation of HPSG 87 is motivated by the need to provide a simple but mathematically precise reconstruction of the predecessor of HPSG 94 that illuminates the internal structure and development of HPSG 94. I maintain that it is impossible to understand considerable parts of the early literature of HPSG 94—and even some of the current terminology in the HPSG literature—without knowing the mathematical concepts of HPSG 87 as they were presented in Pollard and Sag's first book.

There is one important aspect of feature structures that is essential for their use in grammar formalisms that I will deliberately ignore throughout. Their success in linguistics owes much to the development of efficient computational techniques for the treatment of feature structures in parsing and in constraint solving systems.[5] Questions of parsing and questions of the computational properties of HPSG formalisms are beyond the scope of my thesis, and I will only briefly turn to some very general results about computational properties of HPSG 94 in Section 3.3. My investigation of linguistic formalisms is strictly limited to their contribution to theories of language rather than to theories of parsing.

2.1.1 The Idea

Under the perspective of HPSG 87, a grammar specifies a system of partial information about linguistic objects. One piece of information can be more informative than some other piece of information. Alternatively, two pieces of information might be incomparable as to their degree of informativeness, if they provide information about unrelated entities, states or events. To capture the relative degree of informativeness of pieces of information, HPSG 87 assumes that pieces of partial information are ordered in a *subsumption hierarchy.* The position of a piece of information in the subsumption hierarchy indicates its informativeness relative to the other pieces of information in

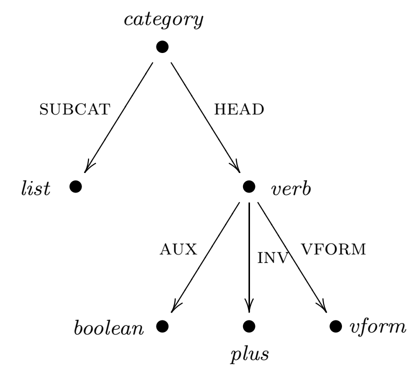
[5]For a broad overview of unification-based grammars in the context of computer implementations at the time when HPSG 87 was conceived, see Shieber 1986.

the hierarchy. The pieces of information that carry less information stand higher in the information ordering, and the ones with more information, i.e., the more specific ones, are lower in the ordering. A bottom element indicates too much or inconsistent information. The set of all possible pieces of information together with the subsumption ordering that is defined on them constitute a certain kind of algebra, namely a *Heyting algebra*. The position of each possible piece of partial information relative to all other possible pieces of partial information is given by the structure of the Heyting algebra. A grammar specifies a set of pieces of partial information in the Heyting algebra and uses the standard operations of a Heyting algebra—*join*, *meet* and *relative pseudocomplement*—as a recipe to construct from the specified pieces of partial information one huge piece of partial information. The resulting piece of partial information constitutes the information that a language community shares about their language.

The idea of language as partial information very naturally leads to an interesting account of language processing. In that account, language processing makes use of the subsumption ordering of pieces of information and of the algebraic operations on the pieces of information. The algebraic operations provide a theoretical foundation for an approach to language processing that assumes that the auditory information of a speech event is processed together with other information that is available to the hearer about the situation in which he or she is. Information-based approaches to grammar thus comprise basic mechanisms that determine how an auditory piece of information can be combined with pertinent information from other modes of perception. The relevant pieces of information that hearers acquire are *unified* with the pieces of information about their language that they have by virtue of their grammar. In other words, hearers accumulate information by unifying pieces of partial information that become available to them from various sources. The algebraic operations lead the hearer from the partial information that he or she gathers in an auditory event and from other internal or external partial information pertaining to their current situation to information about the meaning of the auditory event. Since nothing in this account of language processing hinges on a particular procedural direction of the combination of pieces of information, and since the specification of the grammar is entirely declarative, it can be applied in a theory of language generation as well as in a theory of parsing.

The mathematical entities that HPSG 87 chooses in order to represent pieces of partial information are feature structures. The basic idea is that feature structures are recursively structured, complex entities. Informally, they consist of a set of nodes that are connected by attributes. There is a distinguished root node from which all other nodes of a feature structure can be reached via the attributes. In HPSG, feature structures are *sorted*: The root node of every feature structure has a sort label, as do all other nodes of a feature structure. The sort of the root node indicates about which kind of entity the feature structure provides information. Typical sort labels are *sign*, *category*, or *content*. The nodes of feature structures bear attributes such as CASE, HEAD or DAUGHTERS, which in turn yield (possibly complex) sorted values, e.g., *nominative, verb* or *constituent-structure*. Both the attributes and their values are understood as specifying properties of the entities that the feature structures partially describe. Figure 2.1 illustrates the basic idea of feature structures. The root node of the informal example is labeled *category*, and it has two attributes, SUBCAT and HEAD, with values of sort *list* and *verb*, respectively.

Figure 2.1: Illustration of the idea of feature structures



The recursive embedding of attribute bearing values makes feature structures a flexible tool as arbitrarily complex information-bearing entities. They can serve as very sophisticated partial descriptions of the entities in the empirical domain. The most complex type of feature structure of HPSG 87, disjunctive feature structures, consists of sets of the simpler kind of feature structure just described. Disjunctive feature structures allow for the representation of disjunctive information. They are the feature structures that are actually employed in the specification of principles of grammar in HPSG 87.

We have already noted that the feature structures stand in a subsumption hierarchy. A second subsumption hierarchy of the sort labels of feature structures, as well as conditions on the appropriateness of attributes to certain sorts increase the capability of the formalism to capture the structure of a complex empirical domain. One of the main concerns of HPSG 87 that results naturally from this architecture is the question of how information is distributed or packaged together in feature structures.

In summary, an HPSG 87 grammar construes a natural language as partial information that is represented by feature structures. Every piece of partial information that is true of the language is said to model or to represent the language. Feature structures are located in a Heyting algebra according to their degree of informativeness. The operations of the algebra function as a recipe for constructing the total information that speakers have about their language from the set of feature structures specified by the grammar.

2.1.2 Formal Reconstruction

My formal reconstruction of HPSG 87 is a generalization of King 1989, Chapter 2.[6] Occasionally, I adopt a reading of Pollard and Sag 1987 that is slightly closer to HPSG 94 than the more literal reading of King, and I change some minor technical details of the sketch of a formalism of Pollard and Sag where subsequent research showed that the technical choices of the authors had been erroneous or misled. These differences do not affect the overall architecture of the formalism, and they will prove advantageous for the discussion of the role of feature structures in HPSG 94 below. The classic sources for the mathematics of feature structures construed as partial information are Kasper and Rounds 1986, Moshier 1988, and, for HPSG, Pollard 1989.

In this section, we will successively consider three kinds of feature structures, concrete feature structures, abstract feature structures, and disjunctive feature structures. Abstract feature structures build on concrete feature

[6]My presentation profits substantially from later, much more detailed discussions of the same and of related mathematical material in unpublished manuscripts and lecture notes of Paul King, in particular from King 1995 and King 1996. King 1995 comprises 100 pages and contains a wealth of propositions and detailed proofs of many properties of the formalism, and King 1996 contains an excellent overview of the pertinent aspects of King 1989, Pollard 1989, and Carpenter 1992. I am indebted to Carl Pollard for substantial suggestions concerning the technical execution of the formalization and the presentation of HPSG 87 of this section.

structures, and disjunctive feature structures build on abstract feature structures. While concrete feature structures realize the basic intuitions behind the use of feature structures in HPSG 87, some properties of them are irrelevant for the purposes of HPSG 87, and we can abstract away from them. The abstraction step leads to abstract feature structures. Abstract feature structures stand in an important, well-known relationship to concrete feature structures, and we will make that relationship explicit. The relationship between concrete and abstract feature structures will become very important in the discussion of the distinction between types and tokens of HPSG 94. Yet, abstract feature structures do not have all of the properties that HPSG 87 requires of its feature structures. In particular, they cannot express disjunctive information, and they do not constitute a Heyting algebra under the subsumption relation. Therefore, we will proceed from abstract feature structures to disjunctive feature structures. Disjunctive feature structures are sets of abstract feature structures with certain conditions imposed on the members of the set. They are the only type of feature structure discussed here that does not have a counterpart in HPSG 94.

Before I can define the most basic type of feature structures, I need to provide the set(s) of symbols that are necessary to build them. In HPSG 87, there are two alphabets, a set of *types* and a set of *attributes*, which are sometimes called *features*. Since the types are called *sorts* in most of the HPSG literature, and since the sorts of HPSG 94 replace the types of HPSG 87, I will henceforth call the types of HPSG 87 sorts for ease of exposition.[7] The sorts are thought of as standing in a subsumption ordering which is conventionally called the *sort hierarchy*. The idea of a hierarchy of sort labels goes back to Ait-Kaci 1984 and was first formally explained in HPSG by Pollard (1989), who introduced the term sort instead of type in HPSG. Technically, a sort hierarchy of HPSG 87 is a finite *partial order*. Recall that a partial order, $\langle S, \leq \rangle$, is a set, S, with a reflexive, transitive, and antisymmetric relation, $\leq$, in S. A partial order, $\langle S, \leq \rangle$, is called *finite* iff S is finite. The sort hierarchy should not be confused with the subsumption ordering of feature structures. The subsumption relations on feature structures and the subsumption relation on sorts are different relations on different domains, although the subsumption relation on sorts is important for the definition of the subsumption relation on feature structures.

[7]Another reason for choosing this terminology is that the term *type* is used for a very different concept in HPSG 94, and I want to reserve the term for that concept.

The sort symbols label all nodes of feature structures. If the root node of a feature structure has sort σ, I say that the feature structure is of sort σ. The sort of a feature structure indicates about what kind of object a feature structure provides information. Usually a top sort, sometimes written as '$\top$', stands as the unique maximal element at the top of the sort hierarchy and subsumes all other sort symbols.[8] Sometimes a corresponding bottom element is assumed. However, neither a top element nor a bottom element are obligatory in sort hierarchies. If a top element exists, feature structures of sort '$\top$' carry the least information. They are information about all objects in the domain. A feature structure labeled by the bottom element—if it exists—indicates inconsistent information. Since the sorts that stand lower in the sort hierarchy are labels for more specific information, I will call all sort symbols that are properly subsumed by some sort σ *more specific than* σ.

A collection of feature structures is a collection of feature structures relative to a signature, where a signature comprises a sort hierarchy and a set of attributes. That means that the feature structures may only employ sorts and attributes from the signature. However, our signatures will contain more than just a sort hierarchy and a set of attributes. The set of sorts and the set of attributes of feature structures can be used to restrict the permitted shape of feature structures. One might want to say that, loosely speaking, only certain attributes are appropriate to nodes of a given sort, or the sort of the attribute value of an attribute that "originates" from a node of sort σ must be subsumed by some other sort, σ'. For example, in a theory of English, one might want to say that a feature structure with a root node of sort *phrase* can have a DAUGHTERS attribute, whereas a feature structure of sort *word* or *case* can never have a DAUGHTERS attribute. If an appropriate attribute is present, it leads to another node of the feature structure, and that node is again of some sort. Similarly, appropriateness conditions can be used to restrict the admissible sorts of the attribute values. One could demand that if the originating node is of sort *phrase* and the attribute that leads to the attribute value is DAUGHTERS then the attribute value can only have certain sort labels. For example, it could be allowed to be of sort *constituent-structure* but not of sort *index*. One possibility of capturing appropriateness conditions is to encode them as a function in the signature. In

[8] With the convention of putting '$\top$' at the top of the sort hierarchy, I follow Pollard and Sag 1987 and an intuitive understanding of a 'top' element as the highest element in a hierarchy. It is the dual of the ordering in Pollard 1989, where higher in the order means 'more information'.

my definition of 87 signatures, DEFINITION 1, appropriateness is realized as the function $\mathcal{F}$:

Definition 1 Σ *is an* ***87 signature*** *iff*

> Σ *is a quadruple* $\langle \mathcal{S}, \sqsubseteq, \mathcal{A}, \mathcal{F} \rangle$,
>
> $\langle \mathcal{S}, \sqsubseteq \rangle$ *is a finite partial order,*
>
> $\mathcal{A}$ *is a set,*
>
> $\mathcal{F}$ *is a partial function from the Cartesian product of* $\mathcal{S}$ *and* $\mathcal{A}$ *to* $\mathcal{S}$,
>
> *for each* $\sigma_1 \in \mathcal{S}$, *for each* $\sigma_2 \in \mathcal{S}$ *and for each* $\alpha \in \mathcal{A}$,
>
> > *if* $\mathcal{F}\langle \sigma_1, \alpha \rangle$ *is defined and* $\sigma_2 \sqsubseteq \sigma_1$
> > *then* $\mathcal{F}\langle \sigma_2, \alpha \rangle$ *is defined and* $\mathcal{F}\langle \sigma_2, \alpha \rangle \sqsubseteq \mathcal{F}\langle \sigma_1, \alpha \rangle$.

I call each element of $\mathcal{S}$ a *sort*, $\langle \mathcal{S}, \sqsubseteq \rangle$ the *sort hierarchy*, each element of $\mathcal{A}$ an *attribute*, and $\mathcal{F}$ the *appropriateness function*. If for two sorts σ_1 and σ_2, $\sigma_2 \sqsubseteq \sigma_1$, then I say that σ_2 is *at least as specific as* σ_1. Alternatively, I say that σ_1 *subsumes* σ_2, or σ_2 is a *subsort* of σ_1. The condition on $\mathcal{F}$ enforces attribute inheritance: If an attribute α is appropriate to some sort σ_1, then it is also appropriate to all subsorts of σ_1, and the value of $\mathcal{F}$ at the subsorts of σ_1 and α is at least as specific as $\mathcal{F}\langle \sigma_1, \alpha \rangle$. Note that the definition of an 87 signature does not enforce the presence of a top or bottom sort in the sort hierarchy, and leaves the decision to include them or to omit them to the grammar writer.

An 87 signature determines which symbols are available to indicate the informational content of the feature structures under the respective signature, and the interpretation of an 87 signature ultimately defines an epistemological system of the empirical domain. In Figure 2.2, I specify an 87 signature of a very restricted toy world. With the 87 signature in Figure 2.2, I want to talk about cars, their owners and their drivers, and what people like best. The sort *top* subsumes all other sorts of the sort hierarchy, and it immediately subsumes *car* and *person*. Each one of these two sorts has two immediate subsorts: *vw* and *bmw* are more specific than *car*, and *woman* and *man* are more specific than *person*. Three attributes exist for specifying properties of entities: DRIVER, OWNER, and LIKES-BEST. The idea behind the specification of the appropriateness function, $\mathcal{F}$, is to determine which property is appropriate to what kind of entity in my little toy world, and what the

Figure 2.2: An example of an 87 signature

$\mathcal{S} = \{top, car, vw, bmw, person, man, woman\}$,

$$\sqsubseteq = \left\{ \begin{array}{l} \langle top, top\rangle, \langle car, car\rangle, \langle vw, vw\rangle, \langle bmw, bmw\rangle \langle person, person\rangle, \\ \langle man, man\rangle, \langle woman, woman\rangle, \langle car, top\rangle, \langle vw, top\rangle, \langle bmw, top\rangle, \\ \langle person, top\rangle, \langle man, top\rangle, \langle woman, top\rangle, \langle vw, car\rangle, \langle bmw, car\rangle, \\ \langle man, person\rangle, \langle woman, person\rangle \end{array} \right\},$$

$\mathcal{A} = \{\text{DRIVER}, \text{OWNER}, \text{LIKES-BEST}\}$, and

$$\mathcal{F} = \left\{ \begin{array}{l} \langle\langle car, \text{OWNER}\rangle, person\rangle, \langle\langle car, \text{DRIVER}\rangle, person\rangle, \\ \langle\langle vw, \text{OWNER}\rangle, person\rangle, \langle\langle vw, \text{DRIVER}\rangle, person\rangle, \\ \langle\langle bmw, \text{OWNER}\rangle, person\rangle, \langle\langle bmw, \text{DRIVER}\rangle, person\rangle, \\ \langle\langle person, \text{LIKES-BEST}\rangle, top\rangle, \langle\langle man, \text{LIKES-BEST}\rangle, top\rangle, \\ \langle\langle woman, \text{LIKES-BEST}\rangle, top\rangle \end{array} \right\}$$

possible attribute values are at entities of a certain type. Cars have owners and drivers, and the owners and drivers are persons. Given this specification, the conditions on $\mathcal{F}$ in the last three lines of DEFINITION 1 enforce that Volkswagens and BMWs also must have drivers and owners; and the drivers and owners are at least as specific as *person*, but they might be of sort *man* or *woman*. A Volkswagen, however, cannot be the driver of a BMW. Persons like something or someone best, which implies that men and women like something or someone best because *man* and *woman* are subsorts of *person*. Since the value of $\mathcal{F}\langle person, \text{LIKES-BEST}\rangle$ is *top*, the entity that is liked best can be labeled by any sort of the hierarchy. Note that, so far, I have not said anything about how the intended effects of $\mathcal{F}$ in the domain under consideration will actually be realized in interpretations of 87 signatures, nor have I said how $\mathcal{F}$ will affect the information that feature structures under my 87 signature will be allowed to carry. For example, it is conceivable that $\mathcal{F}$ will only structure the interpretations of 87 signatures but will not impose any conditions on the well-formedness of feature structures. On the other extreme, only feature structures which convey information that is consistent with the intended structure of my toy world might be well-formed feature structures under my 87 signature. To decide which path to pursue, we will have to investigate what Pollard and Sag (1987) tell us about conditions on well-formed feature structures.

The problem with notating an 87 signature as explicitly as in Figure 2.2 is that it is cumbersome to write and very hard to read. Henceforth, I will therefore adopt the notational conventions exemplified in Figure 2.3, which is supposed to depict the same 87 signature as Figure 2.2. Indentation indicates the subsumption relation among sorts. Following the sort symbols, an attribute symbol and another sort symbol specify the appropriateness function in a perspicuous way.

Figure 2.3: A more compact notation for 87 signatures

top				
	car	OWNER	*person*	
		DRIVER	*person*	
		vw		
		bmw		
	person	LIKES-BEST	*top*	
		man		
		woman		

The depiction of $\mathcal{F}$ in Figure 2.3 takes attribute inheritance for granted and does not repeat appropriate attributes and attribute values at proper subsorts of sorts for which the attribute is already declared. For example, we already know that LIKES-BEST is appropriate to *woman* and has *top* as its value at *woman*, because it is declared on *person*, and an 87 signature obeys DEFINITION 1. 'LIKES-BEST *top*' is thus not repeated after *woman*. LIKES-BEST would only have to be repeated if the value of LIKES-BEST were more specific at *woman* than it is at *person*. This would be the case if we would want to declare that women always like BMWs best. Then the last line of Figure 2.3 would be '*woman* LIKES-BEST *bmw*'.

At this point, standard definitions of logical languages would proceed to define how signatures are interpreted in a domain of objects, and they would define the syntax and semantics of a class of formal languages under the possible signatures. For example, an interpretation of an 87 signature could interpret sorts as unary relations over a domain of objects, and attributes as partial functions over the same domain, with the appropriateness function restricting admissible attribute interpretations in correlation to the interpretation of the sort symbols in the domain of objects. As we have seen, however, HPSG 87 does not directly focus on the entities in the domain. Instead, it is mainly interested in information and in the structuring of information about the empirical domain; and pieces of information are expressed as feature structures. As a consequence, HPSG 87 turns to the definition of the representations of linguistic information first. I will not return to the question of interpretations of 87 signatures before the general discussion of the HPSG 87 formalism in Section 2.1.3, where I will say more about the relationship between HPSG 87 grammars and the empirical domain of natural languages.

The most basic definition of feature structures is based on the finite state machines of automata theory.[9] Moshier (1988, pp. 31ff.) calls these feature structures *concrete feature structures*, and I follow his terminology, although my definition differs slightly from his.[10] Concrete feature structures can be considered a natural starting point for any attempt to formalize HPSG 87, since Pollard and Sag 1987, p. 27, indicates that it wants to think of feature structures as "a certain kind of connected, acyclic, finite state machine." As their name suggests, concrete feature structures are easily pictured as real, complex objects. A concrete feature structure simply consists of a set of nodes, one of which is a distinguished root node. From the root node, every other node can be reached by the (possibly repeated) application of a transition function. Every node of a concrete feature structure is labeled by a sort symbol:

Definition 2 *For each 87 signature* $\Sigma = \langle \mathcal{S}, \sqsubseteq, \mathcal{A}, \mathcal{F} \rangle$, $\mathbb{C}$ *is a* ***concrete feature structure under*** Σ *iff*

> $\mathbb{C}$ *is a quadruple* $\langle Q, \hat{q}, \Delta, \Lambda \rangle$,
>
> Q *is a set,*
>
> $\hat{q} \in Q$,
>
> Δ *is a partial function from the Cartesian product of* Q *and* $\mathcal{A}$ *to* Q,
>
> *for each* $q \in Q$, *for some* $i \in \mathbb{N}$, *for some* $q_0 \in Q$, $\ldots$, *for some* $q_i \in Q$,

[9]See Rounds and Kasper 1986 and Kasper and Rounds 1986 for early definitions of feature structures in linguistics.

[10]In Pollard 1989 and Pollard and Moshier 1990, they are dubbed *ordinary feature structures*. King 1989 introduces (a variant of) them as *finite state machines*. Other terminologies exist in the literature.

$q_0 = \hat{q}$,
for each $j \in \mathbb{N}$ *such that* $j < i$, *for some* $\alpha \in \mathcal{A}$,
$\Delta \langle q_j, \alpha \rangle$ *is defined and* $\Delta \langle q_j, \alpha \rangle = q_{j+1}$, *and*
$q_i = q$, *and*

Λ *is a total function from* Q *to* $\mathcal{S}$.

If Σ is an 87 signature, I write $\mathbb{CFS}^{\Sigma}$ for the *class of concrete feature structures under* Σ. Note that the definition of concrete feature structures under Σ is independent of many details of Σ. For example, it is not affected by the presence or absence of an appropriateness function or a subsumption ordering on the set of sorts, $\mathcal{S}$. The class of concrete feature structures under Σ could thus be defined relative to other signatures as well, as long as they provide two disjoint sets of symbols, $\mathcal{A}$ and $\mathcal{S}$. In order to avoid awkward formulations, I will henceforth usually omit the explicit reference to Σ when talking about (concrete) feature structures.

In automata theory, Q is called the *set of states* of the automaton, and Q is assumed to be a finite set. $\hat{q}$ is the *initial state* of the automaton. In the concrete feature structures of linguistics, Q is called the *set of nodes*, and it is not necessarily finite. $\hat{q}$ is the distinguished *root node* of a concrete feature structure. In the terminology of finite state machines, Δ is the *transition function* that transits from one node to another node consuming one symbol of the *input alphabet*, $\mathcal{A}$. I call the elements of the set of finite strings over $\mathcal{A}$ *paths*. I write ε for the empty path consisting of the string of zero attributes, and $\mathcal{A}^*$ for the set of finite strings (or paths) of attributes. The *connectedness conditions* in lines 7 to 11 ensure that every node of a concrete feature structure can be reached from the root node in finitely many transitions consuming a finite string of elements of $\mathcal{A}$. Another way to express this fact is to say that every node is reachable from the root node by interpreting a path of attributes on the root node. For simplicity, I say that an attribute, α, is defined on a node, q, if $\Delta \langle q, \alpha \rangle$ is defined, or, in other words, if we can get to some node q' by interpreting α on q. Λ assigns each node of a feature structure a sort symbol, σ. I say that a node is labeled by the sort σ, or it has sort σ. In automata theory, $\mathcal{S}$ is called the *output alphabet*, because the idea is that when we transit from the initial state to some state on an input string from $\mathcal{A}^*$, the automaton returns the output symbols from $\mathcal{S}$ that label each state that we pass.

In (1), I give three examples of concrete feature structures under the signature of Figure 2.3:

(1) a. $\mathbb{C}_1 = \langle Q_1, \hat{q}, \Delta_1, \Lambda_1 \rangle$ with

$Q_1 = \{q_1, q_2, q_3\}$,

$\hat{q} = q_1$,

$\Delta_1 = \left\{ \begin{array}{l} \langle\langle q_1, \text{OWNER}\rangle, q_2\rangle, \langle\langle q_1, \text{DRIVER}\rangle, q_3\rangle, \\ \langle\langle q_2, \text{LIKES-BEST}\rangle, q_3\rangle, \langle\langle q_3, \text{LIKES-BEST}\rangle, q_2\rangle \end{array} \right\}$, and

$\Lambda_1 = \{\langle q_1, vw\rangle, \langle q_2, woman\rangle, \langle q_3, man\rangle\}$.

b. $\mathbb{C}_2 = \langle Q_2, \hat{q}, \Delta_2, \Lambda_2 \rangle$ with

$Q_2 = \{q'_1, q'_2\}$,

$\hat{q} = q'_1$,

$\Delta_2 = \{\langle\langle q'_1, \text{LIKES-BEST}\rangle, q'_2\rangle\}$, and

$\Lambda_2 = \{\langle q'_1, person\rangle, \langle q'_2, bmw\rangle\}$.

c. $\mathbb{C}_3 = \langle Q_3, \hat{q}, \Delta_3, \Lambda_3 \rangle$ with

$Q_3 = \left\{ q''_n \,\middle|\, n \in \mathbb{N} \text{ and } n > 0 \right\}$,

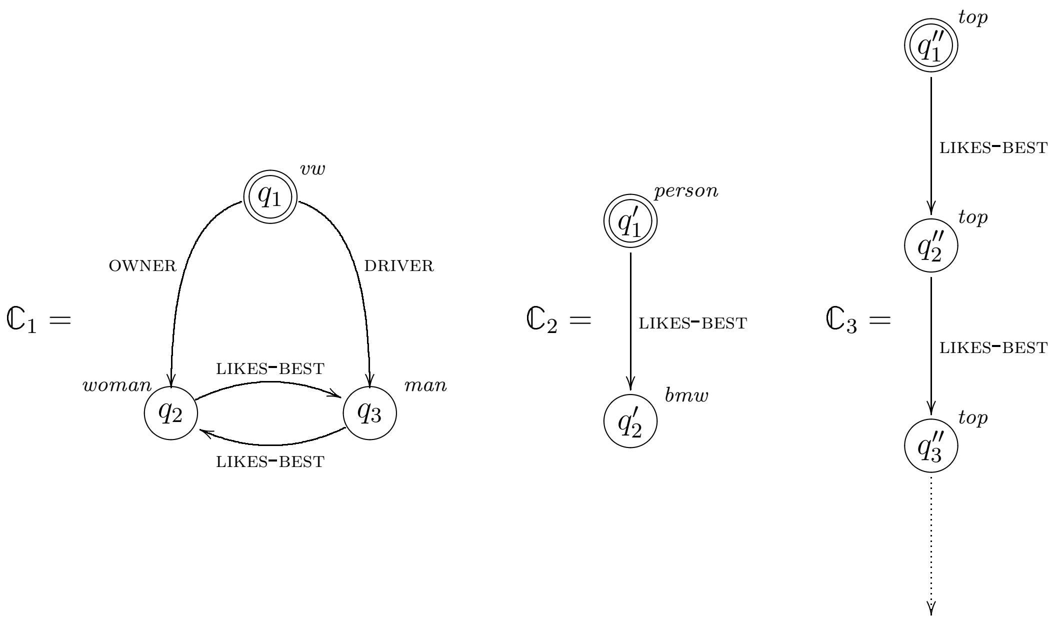
$\hat{q} = q''_1$,

$\Delta_3 = \left\{ \left\langle \langle q''_n, \text{LIKES-BEST}\rangle, q''_{n+1} \right\rangle \,\middle|\, n \in \mathbb{N} \text{ and } n > 0 \right\}$, and

$\Lambda_3 = \left\{ \langle q''_n, top\rangle \,\middle|\, n \in \mathbb{N} \text{ and } n > 0 \right\}$.

$\mathbb{C}_1$ represents information about Volkswagens whose female owner and male driver like each other best. $\mathbb{C}_2$ represents information about persons who like a BMW best. And $\mathbb{C}_3$ is about entities in the domain that like another entity best that likes another entity best and so on forever. Note that $\mathbb{C}_3$ is a legitimate concrete feature structure under Σ, although LIKES-BEST is not appropriate to *top* according to Σ. For reasons of readability, I will henceforth depict concrete feature structures in the conventional graph notation illustrated in (2) with the concrete feature structures $\mathbb{C}_1$, $\mathbb{C}_2$, and $\mathbb{C}_3$ of (1):

(2) *The concrete feature structures of (1) in graph notation*



It is convenient to generalize the transition function, Δ, that specifies the transitions with one attribute from a node to another node to the transition function Δ^* that specifies transitions with an arbitrary path of attributes from one node to another:

Definition 3 *For each 87 signature $\Sigma = \langle \mathcal{S}, \sqsubseteq, \mathcal{A}, \mathcal{F} \rangle$, for each $\mathbb{C} = \langle Q, \hat{q}, \Delta, \Lambda \rangle \in \mathbb{CFS}^{\Sigma}$, Δ^* is the partial function from the Cartesian product of Q and $\mathcal{A}^*$ to Q such that,*

for each $q \in Q$,

$\Delta^* \langle q, \varepsilon \rangle$ *is defined and* $\Delta^* \langle q, \varepsilon \rangle = q$, *and*

for each $q \in Q$, for each $\pi \in \mathcal{A}^*$, for each $\alpha \in \mathcal{A}$,

$\Delta^* \langle q, \pi\alpha \rangle$ *is defined if* $\Delta \langle \Delta^* \langle q, \pi \rangle, \alpha \rangle$ *is defined, and*
if $\Delta^* \langle q, \pi\alpha \rangle$ *is defined then* $\Delta^* \langle q, \pi\alpha \rangle = \Delta \langle \Delta^* \langle q, \pi \rangle, \alpha \rangle$.

Interpreting the empty path, ε, on a node, q, Δ^* transits to q itself. For each nonempty path, π, Δ^* is the iteration of the application of Δ on the attributes in π and the node that has been reached by the (iterative) application of Δ to the preceding attributes in π. Extending my previous simplifying terminology, I say that a path, π, is defined on a node, q, if $\Delta^* \langle q, \pi \rangle$ is defined. Alternatively, I sometimes say that a concrete feature structure, $\mathbb{C} = \langle Q, \hat{q}, \Delta, \Lambda \rangle$, has a path π, meaning that $\Delta^* \langle \hat{q}, \pi \rangle$ is defined.

The concrete feature structures of HPSG 87 are more restricted than the concrete feature structures of Definition 2. The 87 concrete feature structures are finite, which means that they only have finitely many nodes. Moreover, they were originally envisaged to be acyclic: With the exception of the empty path, the interpretation of no path on a node leads back to the node (Pollard and Sag, 1987, p. 37). However, the acyclicity condition was motivated by restrictions that were built into parsers using feature structures at the time when Pollard and Sag's book was written, and by the erroneous assumption that cyclic structures would pose computational problems; there was no genuinely linguistic motivation for it. Therefore, I drop it and permit cycles in 87 concrete feature structures. Finally, 87 concrete feature structures respect appropriateness to some degree. According to Pollard and Sag 1987, p. 39, "[...] feature structures come in different *types* [sorts] depending on the kind of object they describe, and a different set of attributes is appropriate for each type of feature structure." That means that if a node is labeled by sort σ then it may only have attributes that are appropriate to σ. Pollard and Sag are not as clear about whether they want to enforce the appropriateness of attribute values in feature structures. On the one hand, they say that "[i]t should also be clear that for different attributes, different types of values are appropriate" (ibid, p. 39). On the other hand, in a comment to an AVM depiction of a feature structure,[11] they say that "[b]y virtue of not being mentioned, the PHONOLOGY attribute is *implicitly* specified as $\top$" (ibid, p. 39), which is presumably only possible if appropriateness is not required of attribute values. Since formally nothing in HPSG 87 depends on appropriateness in feature structures, the choice is arbitrary. Based on what Pollard and Sag say, I choose to enforce the appropriateness of attributes but not the appropriateness of attribute values in 87 concrete feature structures. This decision will carry over to abstract and disjunctive feature structures below.

[11] Example (72), Pollard and Sag 1987, p. 38.

Definition 4 *For each 87 signature* $\Sigma = \langle \mathcal{S}, \sqsubseteq, \mathcal{A}, \mathcal{F} \rangle$, $\mathbb{C}$ *is an* ***87 concrete feature structure under*** Σ *iff*

$\mathbb{C} = \langle Q, \hat{q}, \Delta, \Lambda \rangle \in \mathbb{CFS}^{\Sigma}$,

Δ *is finite, and*

for each $q_1 \in Q$, *for each* $\alpha \in \mathcal{A}$,

if $\Delta \langle q_1, \alpha \rangle$ *is defined then* $\mathcal{F} \langle \Lambda(q_1), \alpha \rangle$ *is defined.*

If Σ is an 87 signature, I write $\mathbb{CFS}^{\Sigma}_{87}$ for the *class of 87 concrete feature structures under* Σ. Requiring that the transition function, Δ, be finite,[12] entails that the set of nodes, Q, is finite, because the connectedness condition of DEFINITION 2 makes sure that each node is accessible from the root node by some path. To enforce the original acyclicity condition of Pollard and Sag, one could add the condition that for each $q \in Q$, for each $\pi \in \mathcal{A}^*$, if $\Delta^* \langle q, \pi \rangle$ is defined and $\Delta^* \langle q, \pi \rangle = q$ then $\pi = \varepsilon$.

Since I did not include the acyclicity condition in DEFINITION 4, the concrete feature structure $\mathbb{C}_1$ in (1a) is also an 87 concrete feature structure under the signature of Figure 2.3, although it is cyclic. For example, interpreting the nonempty path 'LIKES-BEST LIKES-BEST' on the node q_2 leads back to the node q_2. $\mathbb{C}_1$ has a finite set of nodes, and the attributes obey the appropriateness conditions of the signature. The concrete feature structure $\mathbb{C}_3$ of (1c) violates two conditions on 87 concrete feature structures. Firstly, its set of nodes, Q_3, is infinite, and secondly, nodes of sort *top* are not allowed to have the attribute LIKES-BEST because LIKES-BEST is not appropriate to *top*. $\mathbb{C}_3$ is thus not an 87 concrete feature structure. The second concrete feature structure in (1) that satisfies all conditions on 87 concrete feature structures is $\mathbb{C}_2$.

The definition of 87 concrete feature structures does not allow for set-valued feature structures. Pollard and Sag 1987 argues that for linguistic reasons, the feature structures of HPSG 87 should have the capability of expressing information about sets, and Pollard and Moshier 1990 and Moshier and Pollard 1994 define an appropriate extension of concrete (and abstract) feature structures. Since the inclusion of set values in HPSG 87 complicates the mathematical technicalities of feature structures but does not affect the overall architecture of HPSG 87 nor the underlying intuitions, and since the

[12] Throughout this thesis, I do not distinguish between functions and their graphs.

techniques that Pollard and Moshier introduce will not become immediately relevant to my approach to HPSG 94, I ignore set-valued feature structures here. In Section 2.1.3, I will briefly characterize the formalisms of Pollard and Moshier.

I am now ready to define the subsumption ordering of concrete feature structures. In a general approach, it could be defined for concrete feature structures as given in DEFINITION 2. From the general notion of concrete feature structure subsumption I could derive a notion of subsumption for 87 concrete feature structures. However, since below I will only need the subsumption relation for 87 concrete feature structures, I skip the general definition and define the subsumption relation directly on 87 concrete feature structures. For that definition, I first need the notion of a *morphism* between 87 concrete feature structures; the subsumption ordering will then be defined in terms of of morphisms.

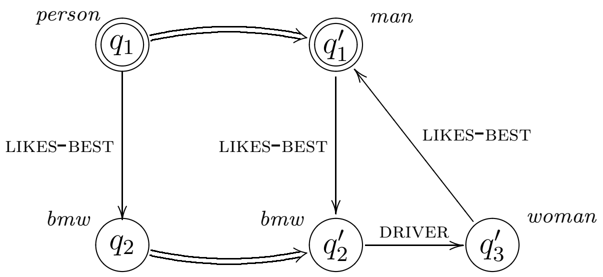
Definition 5 *For each 87 signature* $\Sigma = \langle \mathcal{S}, \sqsubseteq, \mathcal{A}, \mathcal{F} \rangle$, *for each* $\mathbb{C}_1 = \langle Q_1, \hat{q}_1, \Delta_1, \Lambda_1 \rangle \in \mathbb{CFS}^{\Sigma}_{87}$, *for each* $\mathbb{C}_2 = \langle Q_2, \hat{q}_2, \Delta_2, \Lambda_2 \rangle \in \mathbb{CFS}^{\Sigma}_{87}$, μ *is a* ***morphism from*** $\mathbb{C}_1$ ***to*** $\mathbb{C}_2$ ***under*** Σ *iff*

> μ *is a total function from* Q_1 *to* Q_2,
>
> $\mu(\hat{q}_1) = \hat{q}_2$,
>
> *for each* $q \in Q_1$, *for each* $\alpha \in \mathcal{A}$,
>
> > *if* $\Delta_1 \langle q, \alpha \rangle$ *is defined,*
> > *then* $\Delta_2 \langle \mu(q), \alpha \rangle$ *is defined and* $\Delta_2 \langle \mu(q), \alpha \rangle = \mu(\Delta_1 \langle q, \alpha \rangle)$, *and*
>
> *for each* $q \in Q_1$, $\Lambda_2(\mu(q)) \sqsubseteq \Lambda_1(q)$.

The existence of a morphism from $\mathbb{C}_1$ to $\mathbb{C}_2$ means that $\mathbb{C}_1$ is in some sense embedded in $\mathbb{C}_2$, or $\mathbb{C}_2$ is an extension of $\mathbb{C}_1$. The existence of a morphism from $\mathbb{C}_1$ to $\mathbb{C}_2$ ensures that every node that is present in $\mathbb{C}_1$ has a counterpart in $\mathbb{C}_2$, and that counterpart bears a sort label that is at least as specific as the sort label of the corresponding node of $\mathbb{C}_1$. The counterpart of the root node of $\mathbb{C}_1$ is the root node of $\mathbb{C}_2$. If a node in $\mathbb{C}_1$ can be reached by a path π then a counterpart that can be reached by π exists in $\mathbb{C}_2$; but in $\mathbb{C}_2$, there might be more paths defined on the root node than in $\mathbb{C}_1$. A standard result tells us that if there is a morphism from $\mathbb{C}_1$ to $\mathbb{C}_2$ then it is unique.

The example in (3) illustrates the definition of morphisms. The double arrows indicate the mapping of the nodes of the 87 concrete feature structure on the left to the 87 concrete feature structure on the right:

(3)



Subsumption is defined in terms of morphisms as follows:

Definition 6 *For each 87 signature* Σ, *for each* $\mathbb{C}_1 \in \mathbb{CFS}^{\Sigma}_{87}$, *for each* $\mathbb{C}_2 \in \mathbb{CFS}^{\Sigma}_{87}$, $\mathbb{C}_1$ ***subsumes*** $\mathbb{C}_2$ ***under*** Σ *iff*

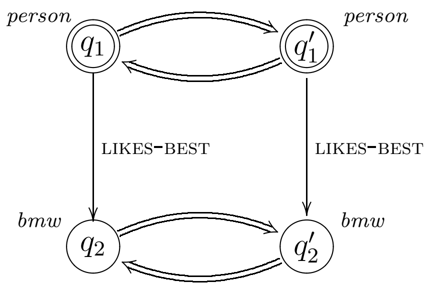
for some μ, μ *is a morphism from* $\mathbb{C}_1$ *to* $\mathbb{C}_2$ *under* Σ.

If Σ is an 87 signature, I write $\mathbb{C}_2 \trianglelefteq \mathbb{C}_1$ for 87 concrete feature structure under Σ, $\mathbb{C}_1$, subsumes 87 concrete feature structure under Σ, $\mathbb{C}_2$. The underlying idea of DEFINITION 6 is that if one 87 concrete feature structure, $\mathbb{C}_1$, subsumes a second, $\mathbb{C}_2$, then $\mathbb{C}_2$ provides more information or is more specific than $\mathbb{C}_1$. Conversely, $\mathbb{C}_1$ is more general than $\mathbb{C}_2$, because it conveys information about a bigger variety of objects. In (3), the 87 concrete feature structure on the left subsumes the 87 concrete feature structure on the right, because there is a morphism from the nodes of the first to the nodes of the second. Notice that this ordering of 87 concrete feature structures is the dual of the ordering that is usually assumed in computer science, and also in Moshier 1988, Pollard 1989 and Pollard and Moshier 1990. The metaphor that is adopted there is that higher up in the hierarchy means more information, whereas, in Pollard and Sag 1987, higher up in the subsumption hierarchy means more general information.

Recall that the purpose of our definitions is to obtain representations of partial information which, together with a subsumption relation on these representations, constitute a Heyting algebra, because HPSG 87 wants to model the grammatical principles with the operations of a Heyting algebra on pieces of partial information. What are the relevant properties of

$\left\langle \mathbb{CFS}^{\Sigma}_{87}, \trianglelefteq \right\rangle$? 87 concrete feature structures under subsumption almost constitute a *preorder*:[13] 87 concrete feature structure subsumption is transitive and reflexive. Notice, however, that with DEFINITION 4, $\mathbb{CFS}^{\Sigma}_{87}$ may constitute a proper class. In the general case, $\left\langle \mathbb{CFS}^{\Sigma}_{87}, \trianglelefteq \right\rangle$ can thus only be a preorder if one chooses a definition of preorder that admits a class-sized domain for the subsumption relation. This technical concern can be allayed by choosing an appropriately restricted domain of entities from which to draw the nodes of 87 concrete feature structures. Then $\left\langle \mathbb{CFS}^{\Sigma}_{87}, \trianglelefteq \right\rangle$ is a preorder under a set-theoretic definition of preorder. A second observation is more important to us. Even if we guarantee that $\mathbb{CFS}^{\Sigma}_{87}$ is a set, $\left\langle \mathbb{CFS}^{\Sigma}_{87}, \trianglelefteq \right\rangle$ cannot constitute a partial order. Consider the two 87 concrete feature structures depicted in (4). I refer to the feature structure on the left as $\mathbb{C}_1$ and to the one on the right as $\mathbb{C}_2$.

(4)



Obviously, $\mathbb{C}_1$ and $\mathbb{C}_2$ mutually subsume each other. Nevertheless, they are not identical. They are distinct 87 concrete feature structures with different nodes. That means that the subsumption relation is not antisymmetric, and $\left\langle \mathbb{CFS}^{\Sigma}_{87}, \trianglelefteq \right\rangle$ is, therefore, not a partial order. But if $\left\langle \mathbb{CFS}^{\Sigma}_{87}, \trianglelefteq \right\rangle$ cannot constitute a partial order, it can never constitute a Heyting algebra either, which is what HPSG 87 is after, because it determines the meaning of a grammar on the basis of the operations of Heyting algebras.[14] We must conclude that in order to obtain a Heyting algebra of feature structures under subsumption, a different kind of feature structure is needed.

[13] A preorder is a set S together with a reflexive and transitive relation in S.

[14] Of course, the fact that the subsumption relation is not antisymmetric is not the only reason for why $\langle \mathbb{CFS}^{\Sigma}_{87}, \trianglelefteq \rangle$ is not the right choice. It also does not provide the relative pseudocomplement operation of Heyting algebras that HPSG 87 needs to represent conditional information, and it cannot represent the right kind of disjunction with the join operation. See pp. 51ff. below for an explanation of the relevant algebraic notions.

The usual method for obtaining a Heyting algebra of feature structures from simple feature structures like our concrete feature structures is to proceed from the simple feature structures to (appropriately restricted) sets of feature structures, and to define appropriate subsumption relations on these sets of feature structures. In other words, the new feature structures are sets of the more primitive feature structures whose algebra does not provide all of the operations that we want to use. While, in principle, such a construction could start from 87 concrete feature structures, it is usually not done that way. It is technically more attractive to move from the potentially large, non-antisymmetric preorders of concrete feature structures to partial orders of a new type of feature structure first. The largely aesthetical reason for that is that partial orders of a set-sized collection of feature structures are mathematically more familiar and tractable.

The technical step from the preorders of concrete feature structures to partial orders of feature structures can also be motivated informally with intuitions about which properties of pieces of partial information are really important to us. Given what we want to do with them, making a distinction between two concrete feature structures that are identical up to their nodes is superfluous. It is not necessary to distinguish between the two 87 concrete feature structures of (4) as information-bearing entities. Two information-bearing entities could be the same if they bear the same information in the same way, but that is not the case for 87 concrete feature structures. The distinctions that are made between them are more fine-grained than what we really need. Therefore, we can characterize a new type of feature structure in terms of isomorphism classes of concrete feature structures without loosing anything. In fact, this is the type of feature structure that Pollard and Sag (1987) envisage as their basic feature structures.

Several techniques can be applied to get from concrete feature structures to a form of representation that contains a unique complex entity for each collection of concrete feature structures that mutually subsume each other, and yields a partial order of the set of complex entities under a new subsumption relation. One possibility is to pick a canonical representative of each isomorphism class of concrete feature structures. Another possibility is to use the fact that concrete feature structures that mutually subsume each other can be characterized by the set of paths that are defined on their root node, by the equivalence classes of paths that lead to the same node, and by a function that assigns each path the sort of the node to which it leads. Informally, this allows to abstract away from individual nodes and to replace them by equivalence classes of paths. Whichever representation we choose, we obtain the desired property of identity under mutual subsumption of the new, less concrete kind of feature structure, and the collection of feature structures becomes set-sized.

Instead of defining the new feature structures as equivalence classes of concrete feature structures, I adopt the equivalent but much more elegant representation of Moshier 1988.[15] Moshier 1988 calls the new feature structures *abstract feature structures*. Parallel to concrete feature structures, I present a general definition first before I introduce the more restricted abstract feature structures that HPSG 87 demands. I postpone the explication of the relationship between concrete feature structures and the Moshier representation of abstract feature structures until after I have defined and discussed 87 abstract feature structures. The abstract feature structures of DEFINITION 7 correspond to the *abstract ordinary feature structures* of Pollard 1989 and Pollard and Moshier 1990:

Definition 7 *For each 87 signature* $\Sigma = \langle \mathcal{S}, \sqsubseteq, \mathcal{A}, \mathcal{F} \rangle$, $\mathbb{A}$ *is an* ***abstract feature structure under*** Σ *iff*

$\mathbb{A}$ *is a triple* $\langle \beta, \varrho, \lambda \rangle$,

$\beta \subseteq \mathcal{A}^*$,

$\varepsilon \in \beta$,

for each $\pi \in \mathcal{A}^*$, *for each* $\alpha \in \mathcal{A}$,

if $\pi\alpha \in \beta$ *then* $\pi \in \beta$,

ϱ *is an equivalence relation over* β,

for each $\pi_1 \in \mathcal{A}^*$, *for each* $\pi_2 \in \mathcal{A}^*$, *for each* $\alpha \in \mathcal{A}$,

if $\pi_1\alpha \in \beta$ *and* $\langle \pi_1, \pi_2 \rangle \in \varrho$ *then* $\langle \pi_1\alpha, \pi_2\alpha \rangle \in \varrho$,

λ *is a total function from* β *to* $\mathcal{S}$, *and*

for each $\pi_1 \in \mathcal{A}^*$, *for each* $\pi_2 \in \mathcal{A}^*$,

if $\langle \pi_1, \pi_2 \rangle \in \varrho$ *then* $\lambda(\pi_1) = \lambda(\pi_2)$.

[15] The idea that underlies Moshier's representation goes back to the representations of automata of Nerode 1958.

If Σ is an 87 signature, I write $\mathbb{AFS}^{\Sigma}$ for the *set of abstract feature structures under* Σ. If $\mathbb{A} = \langle \beta, \varrho, \lambda \rangle$, I call β the *basis set in* $\mathbb{A}$, ϱ the *re-entrancy relation in* $\mathbb{A}$, and λ the *label function in* $\mathbb{A}$. The basis set in $\mathbb{A}$ is a prefix closed set of paths, the re-entrancy relation in $\mathbb{A}$ is a right invariant equivalence relation on the basis set in $\mathbb{A}$, and the label function in $\mathbb{A}$ is a total function from the basis set in $\mathbb{A}$ to the set of sorts, $\mathcal{S}$, which respects the re-entrancy relation in $\mathbb{A}$. In other words, if two paths in β are re-entrant then they are assigned the same sort.[16]

Whereas the nodes of concrete feature structures are real objects that are accessible from the root node by the interpretation of paths, the paths themselves take over the role of representing the former nodes in abstract feature structures. The basis set in $\mathbb{A}$ specifies the shape of $\mathbb{A}$, because it specifies which paths belong to $\mathbb{A}$. The re-entrancy relation in $\mathbb{A}$ specifies the set of abstract nodes of $\mathbb{A}$ as the set of equivalence classes, $Quo(\langle \beta, \varrho \rangle)$, of the equivalence relation $\langle \beta, \varrho \rangle$.[17] Each element of the quotient of $\langle \beta, \varrho \rangle$ represents an abstract node. Since the label function in $\mathbb{A}$ assigns each path in β that belongs to the same equivalence class in $\langle \beta, \varrho \rangle$ the same sort, it effectively assigns each abstract node a sort label.

(5) shows the abstract feature structures under the signature of Figure 2.3 that correspond to the concrete feature structures of example (1).

(5) a. Let

$$n_2 = \left\{ \text{OWNER } \underbrace{\text{LIKES-BEST} \ldots \text{LIKES-BEST}}_{2*n \ \ times} \,\middle|\, n \in \mathbb{N} \right\}$$
$$\cup \left\{ \text{DRIVER } \underbrace{\text{LIKES-BEST} \ldots \text{LIKES-BEST}}_{n \ \ times} \,\middle|\, \begin{array}{l} n \text{ is an odd} \\ \text{natural number} \end{array} \right\} \text{ and}$$
$$n_3 = \left\{ \text{DRIVER } \underbrace{\text{LIKES-BEST} \ldots \text{LIKES-BEST}}_{2*n \ \ times} \,\middle|\, n \in \mathbb{N} \right\}$$
$$\cup \left\{ \text{OWNER } \underbrace{\text{LIKES-BEST} \ldots \text{LIKES-BEST}}_{n \ \ times} \,\middle|\, \begin{array}{l} n \text{ is an odd} \\ \text{natural number} \end{array} \right\}.$$

With the sets n_2 and n_3, I define $\mathbb{A}_1$ as follows:

[16] It might be interesting to notice that the *feature tree structures* of Smolka and Treinen 1994, p. 234, are weaker than abstract feature structures, since they lack a re-entrancy relation.

[17] Recall that an equivalence relation is a preorder, $\langle S, \circ \rangle$, where S is a set and '$\circ$' is a reflexive, transitive and symmetric relation in S. If $\langle S, \circ \rangle$ is an equivalence relation, I write $Quo(\langle S, \circ \rangle)$ for the set of all equivalence classes of $\langle S, \circ \rangle$. $Quo(\langle S, \circ \rangle)$ is called the *quotient* of $\langle S, \circ \rangle$.

$$\mathbb{A}_1 = \langle \beta_1, \varrho_1, \lambda_1 \rangle, \text{ where}$$

$$\beta_1 = \{\varepsilon\} \cup \left\{ \text{OWNER } \underbrace{\text{LIKES-BEST} \ldots \text{LIKES-BEST}}_{n \ \textit{times}} \middle| n \in \mathbb{N} \right\}$$
$$\cup \left\{ \text{DRIVER } \underbrace{\text{LIKES-BEST} \ldots \text{LIKES-BEST}}_{n \ \textit{times}} \middle| n \in \mathbb{N} \right\},$$

$$\varrho_1 = \{\langle \varepsilon, \varepsilon \rangle\} \cup \left\{ \langle \pi_1, \pi_2 \rangle \middle| \pi_1 \in n_3, \text{ and } \pi_2 \in n_3 \right\}$$
$$\cup \left\{ \langle \pi_1, \pi_2 \rangle \middle| \pi_1 \in n_2, \text{ and } \pi_2 \in n_2 \right\}, \text{ and}$$

$$\lambda_1 = \{\langle \varepsilon, \textit{vw} \rangle\} \cup \left\{ \langle \pi, \textit{woman} \rangle \middle| \pi \in n_2 \right\} \cup \left\{ \langle \pi, \textit{man} \rangle \middle| \pi \in n_3 \right\}.$$

b. $\mathbb{A}_2 = \langle \beta_2, \varrho_2, \lambda_2 \rangle$, where

$$\beta_2 = \{\varepsilon, \text{LIKES-BEST}\},$$

$$\varrho_2 = \{\langle \varepsilon, \varepsilon \rangle, \langle \text{LIKES-BEST}, \text{LIKES-BEST} \rangle\}, \text{ and}$$

$$\lambda_2 = \{\langle \varepsilon, \textit{person} \rangle, \langle \text{LIKES-BEST}, \textit{bmw} \rangle\}.$$

c. $\mathbb{A}_3 = \langle \beta_3, \varrho_3, \lambda_3 \rangle$, where

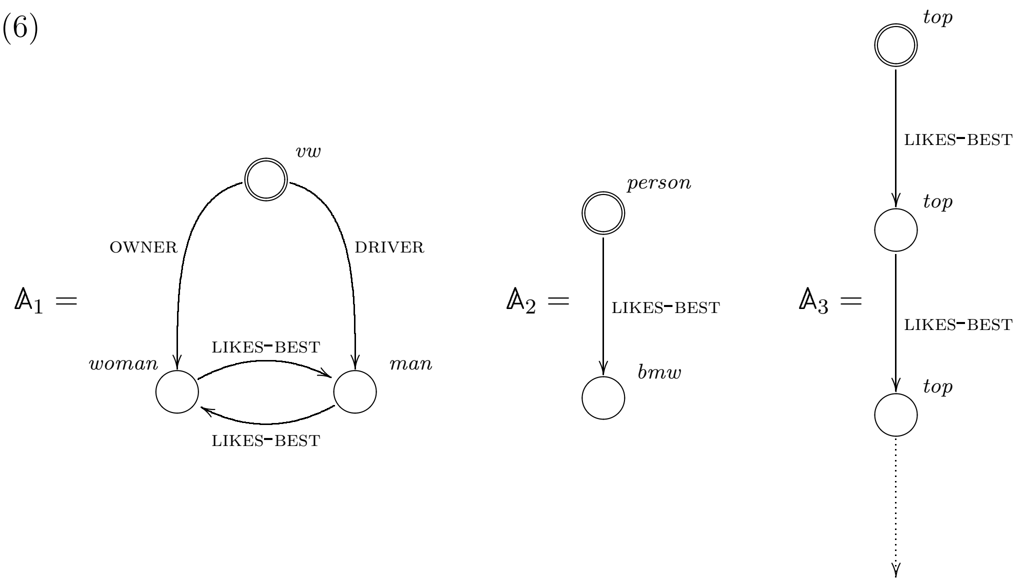
$$\beta_3 = \left\{ \underbrace{\text{LIKES-BEST} \ldots \text{LIKES-BEST}}_{n \ \textit{times}} \middle| n \in \mathbb{N} \right\},$$

$$\varrho_3 = \left\{ \langle \pi, \pi \rangle \middle| \pi \in \beta_3 \right\}, \text{ and}$$

$$\lambda_3 = \left\{ \langle \pi, \textit{top} \rangle \middle| \pi \in \beta_3 \right\}.$$

As in the case of concrete feature structures, abstract feature structures are much more readable when they are depicted in a graph notation. In the notational conventions that I adopt, they are depicted almost exactly as concrete feature structures. The only difference is that their nodes do not have names (such as q_1), which indicates that they are abstract nodes. The notational distinction becomes clear by comparing the pictures of abstract feature structures in (6) to the pictures of their corresponding concrete feature structures in (2), where every node has a unique name.

(6)



As we have seen above, HPSG 87 imposes a number of extra conditions on its feature structures that must now be applied to abstract feature structures in order to obtain 87 abstract feature structures. The conditions are that feature structures be finite and obey the appropriateness conditions of attributes, i.e., only appropriate attributes may be present.

Definition 8 *For each 87 signature* $\Sigma = \langle \mathcal{S}, \sqsubseteq, \mathcal{A}, \mathcal{F} \rangle$, $\mathbb{A}$ *is an* ***87 abstract feature structure under*** Σ *iff*

$\mathbb{A} = \langle \beta, \varrho, \lambda \rangle \in \mathbb{AFS}^{\Sigma}$,

$Quo(\langle \beta, \varrho \rangle)$ *is finite, and*

for each $\pi \in \mathcal{A}^*$, *for each* $\alpha \in \mathcal{A}$,

if $\pi\alpha \in \beta$ *then* $\mathcal{F}\langle \lambda(\pi), \alpha \rangle$ *is defined.*

If Σ is an 87 signature, I write $\mathbb{AFS}^{\Sigma}_{87}$ for the *set of 87 abstract feature structures under* Σ. The requirement that the set of equivalence classes of $\langle \beta, \varrho \rangle$ be finite, entails finiteness. If an abstract feature structure has infinitely many abstract nodes then the quotient of $\langle \beta, \varrho \rangle$ must be infinite, because the equivalence classes represent the abstract nodes. If I wanted to enforce the additional restriction of Pollard and Sag that feature structures be acyclic, I could strengthen the conditions in Definition 8 by replacing the condition that the quotient of $\langle\beta, \varrho\rangle$ be finite with the condition that β be finite. If an abstract feature structure is cyclic, its basis set must be infinite, because the cycle entails that infinitely many paths in β lead to each abstract node in the cycle, as (5a) illustrates. If β is finite then the abstract feature structure cannot have infinitely many abstract nodes either, because the quotient of $\langle\beta, \varrho\rangle$ cannot be infinite if β is finite. The requirement that β always be finite would thus entail both acyclicity and finiteness. Since I allow cyclic structures in 87 abstract feature structures, the two abstract feature structures of the examples in (5) that are also 87 abstract feature structures are the cyclic (5a), and (5b).

As in the case of the subsumption relation over concrete feature structures, I omit a general definition of abstract feature structure subsumption in favor of defining 87 abstract feature structure subsumption immediately. The reason is the same as above: Subsumption will not play any role in HPSG 94, whereas we will encounter a subset of abstract feature structures that differs from the 87 abstract feature structures. The different properties of HPSG 94's abstract feature structures will illuminate the different roles of abstract feature structures in the two HPSG formalisms.

The definition of 87 abstract feature structure subsumption is straightforward and wears the underlying idea on the sleeve that more information corresponds to bigger feature structures, which are therefore subsumed by those smaller feature structures that only contain parts of the bigger feature structures.

Definition 9 *For each 87 signature* $\Sigma = \langle\mathcal{S}, \sqsubseteq, \mathcal{A}, \mathcal{F}\rangle$, *for each* $\mathbb{A}_1 = \langle\beta_1, \varrho_1, \lambda_1\rangle \in \mathbb{AFS}^{\Sigma}_{87}$, *for each* $\mathbb{A}_2 = \langle\beta_2, \varrho_2, \lambda_2\rangle \in \mathbb{AFS}^{\Sigma}_{87}$, $\mathbb{A}_1$ ***subsumes*** $\mathbb{A}_2$ ***under*** Σ *iff*

$\beta_1 \subseteq \beta_2$,

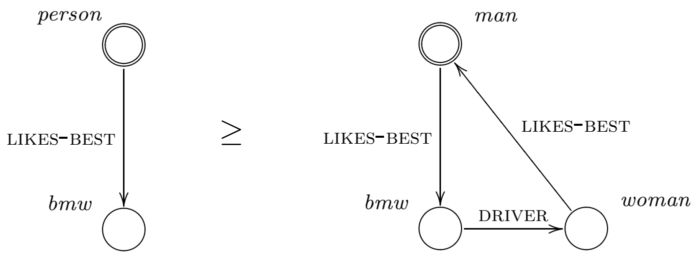
$\varrho_1 \subseteq \varrho_2$, *and*

for each $\pi \in \beta_1$, $\lambda_2(\pi) \sqsubseteq \lambda_1(\pi)$.

If Σ is an 87 signature, I write $\mathbb{A}_2 \leq \mathbb{A}_1$ for 87 abstract feature structure under Σ, $\mathbb{A}_1$, subsumes 87 abstract feature structure under Σ, $\mathbb{A}_2$. With this definition of 87 abstract feature structure subsumption, we obtain the desired partial order property of $\left\langle\mathbb{AFS}^{\Sigma}, \leq\right\rangle$. The ordering of abstract feature structures here and in Pollard and Sag 1987 is the dual of the ordering of the abstract ordinary feature structures of Pollard 1989 and Pollard and Moshier 1990.

As one would intuitively expect, the 87 abstract feature structures that correspond to the two 87 concrete feature structures that illustrate morphisms in (3) stand in the 87 abstract feature structure subsumption relation. I will presently summarize the mathematical results that confirm the intuition that this correspondence between concrete feature structure subsumption and abstract feature subsumption always holds.

(7)



There are a number of important, well-known results about the relationship between concrete feature structures and abstract feature structures, which directly transfer to their 87 analogues. An abstraction relation, Abst_Σ, captures the transition from concrete feature structures to abstract feature structures concisely. Since abstraction will reappear in HPSG 94, I give the general definition:

Definition 10 *For each 87 signature* $\Sigma = \langle \mathcal{S}, \sqsubseteq, \mathcal{A}, \mathcal{F} \rangle$,

> Abst_Σ *is the binary relation on* $\mathbb{CFS}^\Sigma \times \mathbb{AFS}^\Sigma$ *such that, for each* $\mathbb{C} = \langle Q, \hat{q}, \Delta, \Lambda \rangle \in \mathbb{CFS}^\Sigma$, *for each* $\mathbb{A} = \langle \beta, \varrho, \lambda \rangle \in \mathbb{AFS}^\Sigma$, $\langle \mathbb{C}, \mathbb{A} \rangle \in \mathsf{Abst}_\Sigma$ *iff*
>
> $$\beta = \left\{ \pi \in \mathcal{A}^* \,\middle|\, \Delta^* \langle \hat{q}, \pi \rangle \textit{ is defined} \right\},$$
>
> $$\varrho = \left\{ \langle \pi_1, \pi_2 \rangle \in \mathcal{A}^* \times \mathcal{A}^* \,\middle|\, \begin{array}{l} \Delta^* \langle \hat{q}, \pi_1 \rangle \textit{ is defined,} \\ \Delta^* \langle \hat{q}, \pi_2 \rangle \textit{ is defined, and} \\ \Delta^* \langle \hat{q}, \pi_1 \rangle = \Delta^* \langle \hat{q}, \pi_2 \rangle \end{array} \right\}, \textit{ and}$$
>
> $$\lambda = \left\{ \langle \pi, \sigma \rangle \in \mathcal{A}^* \times \mathcal{S} \,\middle|\, \begin{array}{l} \Delta^* \langle \hat{q}, \pi \rangle \textit{ is defined, and} \\ \Lambda(\Delta^* \langle \hat{q}, \pi \rangle) = \sigma \end{array} \right\}.$$

If Σ is an 87 signature, I call Abst_Σ the *abstraction relation under* Σ. A concrete feature structure under Σ, $\mathbb{C}$, and an abstract feature structure under Σ, $\mathbb{A}$, stand in the abstraction relation under Σ exactly if, (a), every path that is defined on the root node of $\mathbb{C}$ is in the basis set of $\mathbb{A}$, (b), every pair of paths whose interpretation on the root node of $\mathbb{C}$ lead to the same node of $\mathbb{C}$ are in the re-entrancy relation of $\mathbb{A}$, and, (c), the sort of the node that is reached by the interpretation of a path π on the root node of $\mathbb{C}$ is the same as the sort that the label function of $\mathbb{A}$ assigns to π.

It is not hard to see that Abst_Σ is a total function from $\mathbb{CFS}^\Sigma$ onto $\mathbb{AFS}^\Sigma$. For every concrete feature structure under Σ, there is exactly one abstract feature structure under Σ to which it abstracts. Moreover, for each abstract feature structure under Σ we can find a concrete feature structure under Σ that abstracts to it. In fact, isomorphic concrete feature structures abstract to the same abstract feature structure. If $\mathbb{C}$ is a concrete feature structure then I call $\mathsf{Abst}_\Sigma(\mathbb{C})$ its abstraction. Note that if $\mathbb{C}_1$ subsumes $\mathbb{C}_2$ (under concrete feature structure subsumption) then the abstraction of $\mathbb{C}_1$ also subsumes the abstraction of $\mathbb{C}_2$ (under abstract feature structure subsumption).[18] The abstractions of two concrete feature structures that mutually subsume each other are identical. Regarding my previous examples of concrete and abstract feature structures, the abstraction relation, Abst_Σ, makes precise the notion of correspondence between the concrete feature structures in (1a)–(1c) and the abstract feature structures in (5a)–(5c) to which I have appealed above.

All of the cited results also hold of the 87 abstraction relation under Σ that we obtain by taking the intersection of Abst_Σ with the Cartesian product of $\mathbb{CFS}^\Sigma_{87}$ and $\mathbb{AFS}^\Sigma_{87}$: 87 concrete feature structures can be abstracted to 87 abstract feature structures in the same way. As a consequence, the two 87 concrete feature structures of example (4) that mutually subsume each other stand in the 87 abstraction relation under my given signature to one and the same 87 abstract feature structure, namely the one given in (5b), and every other concrete feature structure that is isomorphic to them does so, too.

As it turns out, 87 abstract feature structures are still not exactly the kind of feature structure that HPSG 87 needs because they do not permit the representation of either disjunctive or conditional information. A single feature structure must be capable of conveying that at least one out of possibly infinitely many pieces of information holds. For example, a single feature structure can bear the information that it is about a person who likes a woman best or about a car whose owner is its driver. With two feature struc-

[18]The definition of general abstract feature structure subsumption simply replaces the two occurrences of $\mathbb{AFS}^\Sigma_{87}$ in Definition 9 by $\mathbb{AFS}^\Sigma$.

tures, it should be possible to express conditional information. To achieve this, Pollard and Sag 1987, p. 41, generalizes basic feature structures—which correspond to my abstract feature structures—to a new kind of feature structure. Together with their subsumption relation, they constitute a Heyting algebra. Each of Pollard and Sag's new feature structures is a set of pairwise subsumption-incomparable basic feature structures. Since I extend Pollard and Sag's basic feature structures from acyclic feature structures to possibly cyclic feature structures, I also need a slightly different definition here, and I replace their construction with sets of *downward-closed subsets* of the partial order of 87 abstract feature structures under subsumption. Downward-closed subsets are defined as follows:

Definition 11 *For each preorder, $\langle S, \circ \rangle$, D is a* ***downward-closed subset*** *of $\langle S, \circ \rangle$ iff*

$D \subseteq S$, and

for each $x \in S$, for each $y \in D$, if $x \circ y$ then $x \in D$.

If a downward-closed subset of a preorder, $\langle S, \circ \rangle$, contains an element, x, of S, then it also contains all other elements of S that are besides or below x in the preorder, $\langle S, \circ \rangle$. If $\langle S, \circ \rangle$ is a preorder then I write $\delta(\langle S, \circ \rangle)$ for the set of downward-closed subsets of $\langle S, \circ \rangle$.

Since every partial order is a preorder, the set of downward-closed subsets of the partial order of 87 abstract feature structures under Σ and 87 abstract feature structure subsumption is defined for each signature Σ. I use this fact to define 87 disjunctive feature structures:

Definition 12 *For each 87 signature Σ, $\mathbb{D}$ is an* ***87 disjunctive feature structure under*** *Σ iff*

$$\mathbb{D} \in \delta\left(\langle \mathbb{AFS}^{\Sigma}_{87}, \leq \rangle\right).$$

If Σ is an 87 signature, I write $\mathbb{DFS}^{\Sigma}_{87}$ for the *set of 87 disjunctive feature structures under* Σ. An 87 disjunctive feature structure is a downward-closed subset of 87 abstract feature structures. If an 87 disjunctive feature structure contains an 87 abstract feature structure, $\mathbb{A}$, it also contains all 87 abstract feature structures that are below $\mathbb{A}$ in the partial order of 87 abstract feature structures.

Finally, we need a notion of 87 disjunctive feature structure subsumption. It is defined by standard set inclusion:

Definition 13 *For each 87 signature* Σ*, for each* $\mathbb{D}_1 \in \mathbb{DFS}^{\Sigma}_{87}$*, for each* $\mathbb{D}_2 \in \mathbb{DFS}^{\Sigma}_{87}$*,* $\mathbb{D}_1$ ***subsumes*** $\mathbb{D}_2$ ***under*** Σ *iff*

$$\mathbb{D}_2 \subseteq \mathbb{D}_1.$$

An 87 disjunctive feature structure $\mathbb{D}_1$ subsumes an 87 disjunctive feature structure $\mathbb{D}_2$ exactly if $\mathbb{D}_1$ is a superset of $\mathbb{D}_2$.

For each 87 signature Σ, $\left\langle \mathbb{DFS}^{\Sigma}_{87}, \subseteq \right\rangle$ is a complete Heyting algebra, i.e., a bounded, distributive lattice in which each pair of elements of $\mathbb{DFS}^{\Sigma}_{87}$ has a relative pseudocomplement, and finite meets distribute over arbitrary joins. This follows immediately from the definitions by standard results: By construction, $\left\langle \mathbb{DFS}^{\Sigma}_{87}, \subseteq \right\rangle = \left\langle \delta\left(\langle \mathbb{AFS}^{\Sigma}_{87}, \leq \rangle\right), \subseteq \right\rangle$, and the downward-closed subset of any preorder is a complete Heyting algebra under set inclusion.[19] Since $\left\langle \mathbb{DFS}^{\Sigma}_{87}, \subseteq \right\rangle$ is a Heyting algebra, it has a unique top element ($\mathbb{AFS}^{\Sigma}_{87}$) and a unique bottom element ($\emptyset$), and supplies the standard algebraic operations that HPSG 87 uses to determine the meaning of a grammar. They are *join*, which HPSG 87 calls *disjunction*, *meet*, which HPSG 87 calls *unification*, *relative pseudocomplement*, and the unary *pseudocomplement* operation, which HPSG 87 calls *negation*. As we will see, HPSG 87 employs these operations for the unification of pieces of partial information, and to express disjunctive, (non-classical) negative and (non-classical) conditional information. Recall that in a lattice, the join of each pair of elements, x and y, of the lattice is the unique least upper bound in the lattice of the two elements, x and y; and the meet of each pair of elements, x and y, of the lattice is the unique greatest lower bound in the lattice of the two elements, x and y. In our Heyting algebras of 87 disjunctive feature structures, join corresponds to set

[19] $\langle \mathbb{DFS}^{\Sigma}_{87}, \subseteq \rangle$ is the *Alexandrov topology* of the dual preorder of $\langle \mathbb{AFS}^{\Sigma}_{87}, \leq \rangle$, and any topology is a complete Heyting algebra. If $\langle \mathbb{AFS}^{\Sigma}_{87}, \leq \rangle$ has no infinite ascending chains, i.e., infinitely many distinct elements a_n of $\mathbb{AFS}^{\Sigma}_{87}$ such that $a_1 \leq a_2, a_2 \leq a_3, a_3 \leq a_4, \ldots$, then $\langle \mathbb{DFS}^{\Sigma}_{87}, \subseteq \rangle$ is isomorphic to the lattice of the set of subsets of $\mathbb{AFS}^{\Sigma}_{87}$ whose elements are pairwise subsumption-incomparable, ordered in such a way that a set, $\mathbb{D}_1$, subsumes a set, $\mathbb{D}_2$, iff each element of $\mathbb{D}_1$ subsumes some element of $\mathbb{D}_2$, which is Pollard and Sag's original characterization of the lattice of disjunctive feature structures (Pollard and Sag, 1987, p. 41). However, infinite ascending chains of elements of $\mathbb{AFS}^{\Sigma}_{87}$ exist since cyclic feature structures are permitted. They would not exist if 87 abstract feature structures were required to be acyclic.

Thanks are due to Carl Pollard for suggesting to use the Alexandrov topology, pointing out that it is a complete Heyting algebra, and that it is isomorphic to the construction in Pollard and Sag 1987 in the absence of infinite ascending chains.

union, and meet corresponds to set intersection, i.e., for each signature Σ, for each $\mathbb{D}_1 \in \mathbb{DFS}^{\Sigma}_{87}$, for each $\mathbb{D}_2 \in \mathbb{DFS}^{\Sigma}_{87}$, the join of $\mathbb{D}_1$ and $\mathbb{D}_2$ is $\mathbb{D}_1 \cup \mathbb{D}_2$, and the meet of $\mathbb{D}_1$ and $\mathbb{D}_2$ is $\mathbb{D}_1 \cap \mathbb{D}_2$.

In contrast to the situation in the more familiar Boolean algebras, an element, x, of a Heyting algebra does not necessarily have a complement, i.e., there might not be an element, y, such that the join of x and y is the top element of the algebra, and their meet is the bottom element. However, each element, x, has a pseudocomplement. Informally, the pseudocomplement of x is defined as that element y whose meet with x is the bottom element such that all other elements whose meet with x is the bottom element are below y. It is the greatest element whose meet with x is bottom. We can say that y is the closest approximation to the complement of x that we can get in a Heyting algebra. Technically, the pseudocomplement of x in a Heyting algebra is given as the (unique) *relative pseudocomplement* of x and the bottom element. The relative pseudocomplement operation is defined as follows:

Definition 14 *For each lower semilattice,*[20] $\langle S, \leq \rangle$, *with meet* $\wedge$, z *is a* ***relative pseudocomplement*** *of* x *and* y *in* $\langle S, \leq \rangle$ *iff*

$$x \in S,\ y \in S,\ z \in S,\ x \wedge z \leq y, \text{ and,}$$
$$\text{for each } w \in S, \text{ if } x \wedge w \leq y \text{ then } w \leq z.$$

The relative pseudocomplement of x and y is the largest z whose meet with x is still below y. For our Heyting algebras of 87 disjunctive feature structures, $\left\langle \mathbb{DFS}^{\Sigma}_{87}, \subseteq \right\rangle$, this means that the relative pseudocomplement of two 87 disjunctive feature structures, $\mathbb{D}_1 \in \mathbb{DFS}^{\Sigma}_{87}$ and $\mathbb{D}_2 \in \mathbb{DFS}^{\Sigma}_{87}$, is the union of $\mathbb{D}_2$ with the largest downward-closed subset of $\mathbb{AFS}^{\Sigma}_{87}$ whose intersection with $\mathbb{D}_1$ is the empty set.

It is not hard to see that the relative pseudocomplement operation of a Heyting algebra can serve as an approximation to the representation of classical implication in a Boolean algebra, where the element that represents x implies y can be seen as the join of the complement of x, and y. The approximation in a Heyting algebra is the join of the pseudocomplement of x, and y, which is the relative pseudocomplement of x and y. The representation of

[20] A *lower semilattice* is a partial order in which each pair of elements has a greatest lower bound. Therefore, each Heyting algebra is is a lower semilattice.

implicative information that we obtain through the relative pseudocomplement operation in a Heyting algebra is known as *intuitionistic* implication.[21] We will presently see that the relative pseudocomplement operation is of crucial importance to HPSG 87 grammars, and I will illustrate its effect with a simple example of a principle taken from the grammar of Pollard and Sag 1987. The properties of Heyting algebras do not need to concern us here beyond an intuitive understanding of their significance for HPSG 87.

The definition of 87 disjunctive feature structures and the subsumption relation between 87 disjunctive feature structures, and the proposition that for each 87 signature, Σ, $\left\langle \mathbb{DFS}^{\Sigma}_{87}, \subseteq \right\rangle$ constitutes a complete Heyting algebra, concludes the formal reconstruction of HPSG 87. With all definitions in hand, it is now time to discuss how an HPSG 87 grammar uses 87 disjunctive feature structures and the join, meet, relative pseudocomplement and pseudocomplement operations to characterize a natural language.

The predictive power of an HPSG 87 grammar derives from two sources. First, an 87 signature structures the domain about which the grammar talks. Second, there are a set of grammatical principles, a set of *lexical entries*, and a set of *grammar rules*. For the moment, I postpone a discussion of lexical entries and grammar rules. The grammatical principles of Pollard and Sag 1987 are essentially statements of the form "if A then B", where A and B can be understood as propositions about the structure of linguistic signs. These propositions are expressed by pieces of partial information, i.e., by 87 disjunctive feature structures. As discussed above, the operations of a Heyting algebra cannot express classical implication in the general case, because a complement does not necessarily exist for each element (of the carrier) of a Heyting algebra. If it did, each statement of the form "if A then B" could be expressed classically as "not A or B" with the corresponding operations of the algebra, complement and join. To guarantee the existence of a complement for any arbitrary element in the algebra, we would need a Boolean algebra, which $\left\langle \mathbb{DFS}^{\Sigma}_{87}, \subseteq \right\rangle$ is generally not. HPSG 87 chooses the intuitionistic negation of Heyting algebras, the pseudocomplement operation, to express conditional information. Recall that by definition, for each pair of elements, x and y, of a Heyting algebra, there exists a unique relative pseudocomplement of x and y, and each element of a Heyting algebra has a unique pseudocomplement. From the more familiar classical point of view, the relative pseudocomplement operation of the Heyting algebra of 87

[21] For a discussion in the context of HPSG, see Moshier and Rounds 1987.

disjunctive feature structures is getting as close to classical implication as the structure of a Heyting algebra permits. Performing the relative pseudocomplement operation on two elements of a Heyting algebra yields that element of the Heyting algebra that is the closest one available under the circumstances to representing classical implication. Of course, Pollard and Sag 1987 deliberately prefers the intuitionistic approach over the classical approach, because it wants to avoid the well-known problems that usually arise in computing with classical negation. As a result, taking the relative pseudocomplement of two 87 disjunctive feature structures is HPSG87's method of stating conditional information.

An example from the grammar of English of Pollard and Sag 1987 illustrates best what that means and how the principles of HPSG 87 must be read. I tacitly assume a suitable, fixed signature, Σ, for Pollard and Sag's grammar that induces the Heyting algebra, $\left\langle \mathbb{DFS}^{\Sigma}_{87}, \subseteq \right\rangle$, of their grammar. Henceforth, I write $\mathbb{HA}_{\mathrm{PS}}$ for the Heyting algebra of the grammar of Pollard and Sag 1987. Pollard and Sag 1987, p. 58, notates the Head Feature Principle as follows:

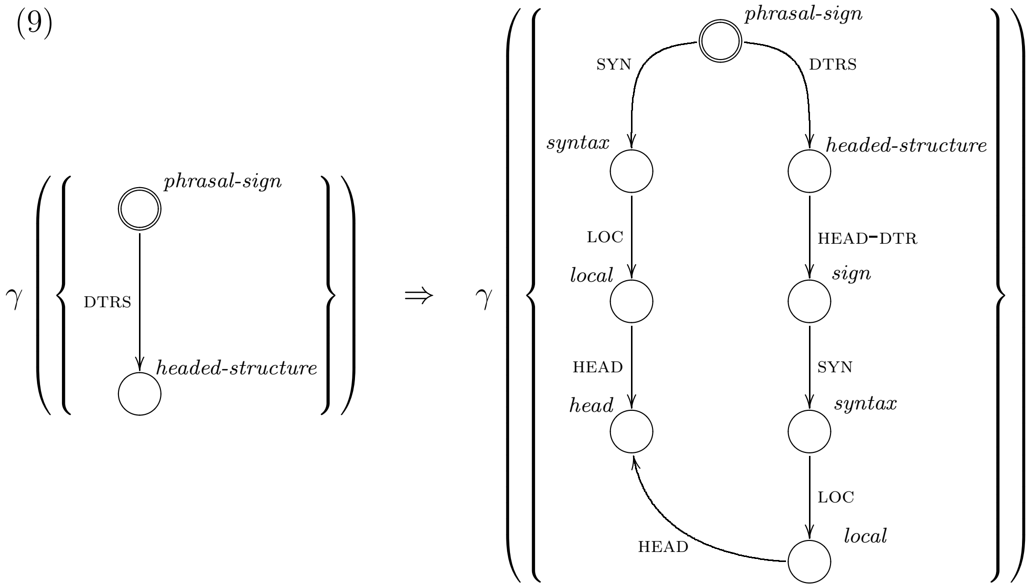
(8) $\left[\textsc{dtrs} \quad {}_{\textit{headed-structure}}\,[\;]\right] \;\Rightarrow\; \begin{bmatrix} \textsc{syn}|\textsc{loc}|\textsc{head}\ \boxed{1} \\ \textsc{dtrs}|\textsc{head-dtr}|\textsc{syn}|\textsc{loc}|\textsc{head}\ \boxed{1} \end{bmatrix}$

The symbol '$\Rightarrow$' stands for the relative pseudocomplement operation of $\mathbb{HA}_{\mathrm{PS}}$. The AVMs to the left and to the right of the relative pseudocomplement symbol are a convenient notation to depict or visualize 87 disjunctive feature structures. At first, it might not be obvious why the two AVMs depict disjunctive feature structures and not abstract feature structures. However, recall that it is the disjunctive feature structures that form a Heyting algebra under subsumption, whereas the abstract feature structures do not. The use of the relative pseudocomplement operation thus forces us to assume that the depicted feature structures are 87 disjunctive feature structures.

It should already be noted here that the role of AVMs in HPSG 87 that (8) exemplifies is very different from the role that AVMs play in HPSG 94. In HPSG 94 they are necessarily understood as expressions of a logical description language, not as shorthand pictures of disjunctive feature structures (which do not exist in HPSG 94). The AVM notation of Pollard and Sag 1987 can be understood as an alternative, informal notation to the one derived from depicting finite state automata that I have adopted above, which depicts 87 disjunctive feature structures as sets of 87 abstract feature structures that consist of (sort-) labeled circles and (attribute-) labeled arcs. Reading

(8), it is also important to keep in mind that Pollard and Sag adopt the abbreviatory convention of omitting all sort labels in the AVM depictions of feature structures that can be inferred by the reader from the attribute names and from the context (Pollard and Sag, 1987, p. 39). Taking this convention into account, (9) depicts the Head Feature Principle, (8), with the notational conventions that I have been using in this section, and says exactly the same as (8), only in a different notation:

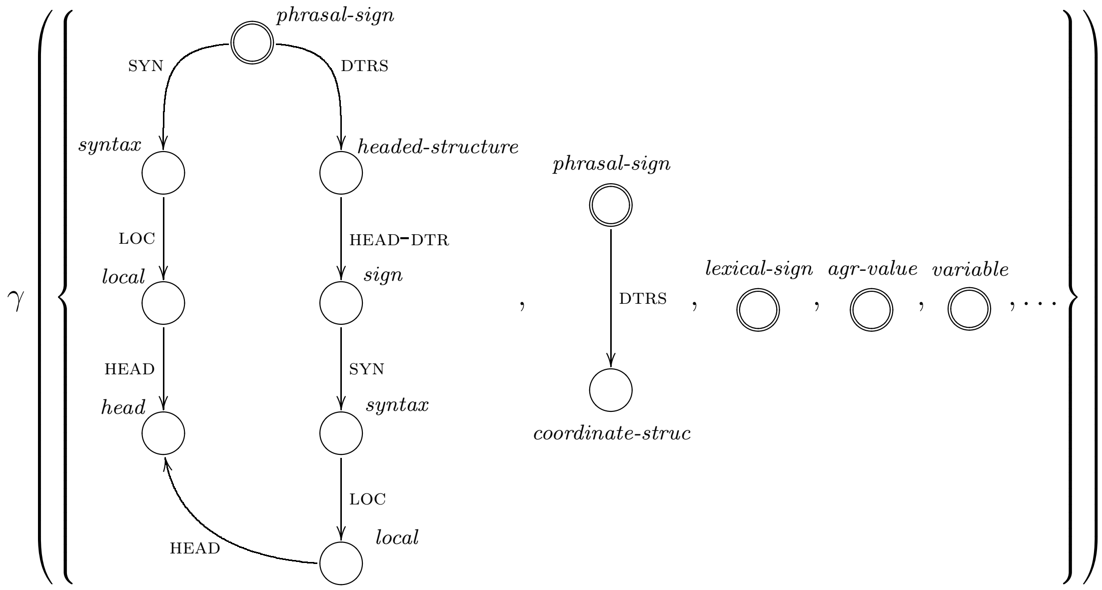
(9)



In (9), γ is the function from the power set of 87 abstract feature structures under Pollard and Sag's signature to the set of 87 disjunctive feature structures under Pollard and Sag's signature that assigns each set of 87 abstract feature structures, A, the smallest downward-closed subset of 87 abstract feature structures that contains the abstract feature structures in A. Informally, γ returns the set of all 87 abstract feature structures that are subsumed by those that are given in the input set. For each set of 87 abstract feature structures, A, I call $\gamma(A)$ the 87 disjunctive feature structure *determined by A*. I need the function γ for a compact, explicit notation of the Head Feature Principle, because my disjunctive feature structures are sets of downward-closed subsets of abstract feature structures. The arguments of γ in (9) are clearly not 87 disjunctive feature structures under Pollard and Sag's signature.

According to the definition of the relative pseudocomplement operation, (8) is a way of pinning down the least specific 87 disjunctive feature structure whose unification with the 87 disjunctive feature structure to the left of '⇒' is still subsumed by the 87 disjunctive feature structure to the right of '⇒'. Equivalently, it is the join of the pseudocomplement of the 87 disjunctive feature structure to the left and the 87 disjunctive feature structure to the right of '⇒'. The resulting 87 disjunctive feature structure is the piece of information in the Heyting algebra $\mathbb{HA}_{\mathrm{PS}}$ that can be understood as expressing intuitionistically that if an object's DTRS value is of sort *headed-structure*, then its SYN LOC HEAD value equals its DTRS HEAD-DTR SYN LOC HEAD value. In (10), I sketch an 87 disjunctive feature structure that is intended to give a first intuition about the resulting 87 disjunctive feature structure. Since Pollard and Sag (1987) do not state the signature of their grammar of English explicitly, we can only approximate the result based on what they tell us about the signature at various places in the book. The dots in (10) stand for the 87 abstract feature structures under Σ that consist of abstract nodes labeled by the immediate subsorts of the top sort of the sort hierarchy, with the exception of *sign*.

(10)



The linguistic significance of the piece of information embodied in the 87 disjunctive feature structure that expresses the HEAD FEATURE PRINCI-

PLE, (8), is that it belongs to a set of pieces of information that Pollard and Sag deem to be true of all natural languages. These pieces of information provide information about all natural languages. In more traditional terms, they express the universal principles of grammar; all speakers of any natural language share that information. Other universal principles of Pollard and Sag 1987 are the SUBCATEGORIZATION PRINCIPLE and the SEMANTICS PRINCIPLE. Assume that there are n universal principles, and let P_1 to P_n be the 87 disjunctive feature structures that express the information of the universal principles.

In addition to the universal principles, Pollard and Sag assume that there is a finite number of language-specific principles for each natural language. They provide conditional, language-specific information about the signs of a given natural language. Again, the relevant pieces of information can be represented as 87 disjunctive feature structures that are specified in the same format as the HEAD FEATURE PRINCIPLE above, i.e., by performing the relative pseudocomplement operation on two 87 disjunctive feature structures that represent information about signs. Assume that there are m language-specific principles of English and call the 87 disjunctive feature structures that express them P_{n+1} to P_{m+n}.

For its theory of English, Pollard and Sag 1987, p. 44 and p. 147, postulates two other types of pieces of information that have a different conceptual status from the pieces of information that express universal and the language-specific principles. Pollard and Sag state them in a format that differs from the format of the principles. Firstly, they set apart information that concerns lexical signs. Pollard and Sag assume that they can exhaustively enumerate the lexical signs of English in a list and express them directly as a finite number of 87 disjunctive feature structures that represent information about words. Let L_1 to L_k be these feature structures. The second kind of information whose status differs from the status of the information that expresses principles is the kind of information that is contained in a small number of so-called grammar rules. The grammar rules are the predecessors of the ID schemata of HPSG 94 grammars. They are expressed as 87 disjunctive feature structures that are determined by singleton sets whose only members are 87 abstract feature structures of sort *phrasal-sign*. Each of them provides information about "[one option] offered by the language in question for making big signs from little ones" (Pollard and Sag, 1987, p. 147). Call the 87 disjunctive feature structures that express the combinatory options for constructing English phrases R_1 to R_i.

The difference between expressing the principles and expressing the grammar rules and the information about lexical signs consists in the way in which the pertinent information is specified. For the principles, it is specified with the relative pseudocomplement operation on two 87 disjunctive feature structures. For the lexical signs and the grammar rules, it is specified as 87 disjunctive feature structures without applying the relative pseudocomplement operation.

We now have four sets with pieces of information about (the English) language:

1. a set of universal, conditional information about signs,
2. a set of language-specific, conditional information about the signs of English,
3. a set of language-specific information about the lexical signs of English, and
4. a set of information about the admissible ways of constructing phrases from smaller signs.

The remaining question is how the four sets of information can be combined to construct a theory of English. What all the information has in common is that it is represented by 87 disjunctive feature structures that are elements of $\mathbb{HA}_{\mathrm{PS}}$. In HPSG 87's view, a theory of a natural language is a theory of (partial) information about that language. The obvious move of Pollard and Sag is, then, to employ the operations of $\mathbb{HA}_{\mathrm{PS}}$ to construct one appropriate big piece of information about English from the information in the four sets enumerated above. The two operations that they use for that purpose are unification, written as '$\wedge$', and join, written as '$\vee$':

(11) $\textsc{English}_{\mathrm{S}} = \mathrm{P}_1 \wedge \ldots \wedge \mathrm{P}_{m+n} \wedge (\mathrm{L}_1 \vee \ldots \vee \mathrm{L}_k \vee \mathrm{R}_1 \vee \ldots \vee \mathrm{R}_i)$

Pollard and Sag (1987, p. 147) call (11) their theory of English. $\textsc{English}_{\mathrm{S}}$ is a piece of information about English signs that is constructed with the unification and the join operations of $\mathbb{HA}_{\mathrm{PS}}$. Pollard and Sag say that an entity is an English sign if and only if $\textsc{English}_{\mathrm{S}}$ provides information about the entity. This is only the case if firstly, the entity obeys all language-specific and all universal principles, and secondly, either it is a word and at least one of L_1 to L_k represents valid lexical information about it, or it is a phrase and at least one of the feature structures, R_1 to R_i, expressing the grammar rules represents valid information about it. If we regard the conjuncts and disjuncts of (11) as the feature structure representations of descriptions of a logic of descriptions then this can be reformulated in more familiar terminology. An entity is an English sign if and only if it satisfies the universal and language-specific principles, and it is either described by at least one feature structure out of L_1 to L_k (i.e., it is a well-formed word of English), or it is described by at least one feature structure out of R_1 to R_i (i.e., it obeys one of the grammar rules).

Notice, however, that according to the 87 signature sketched in Pollard and Sag 1987, the English language does not only consist of entities that are words or phrases. The book contains pictures of feature structures that provide information about many other types of entities, such as *agr-value* objects, *category* objects and *quantified-circumstance* objects. Even more importantly, entities that are not signs are necessarily components of signs. For example, in a sentence like *Kim has seen Sandy*, there is a sign for *Kim* that has a component that is an entity of type *category* that plays a role in specifying the syntactic category of *Kim*; and it also contains an entity of type *agr-value* that is crucial for the treatment of subject-verb agreement in the sentence. The existence in the language of entities that are not signs raises the question if it is fully adequate to think that the task of a theory of English is to provide information only about signs. Alternatively, one could try to use the information inherent in the four sets of 87 disjunctive feature structures above to construct a piece of information that must be true of all entities in the English language rather than just of the English signs.[22]

When considering this question, it is also worth noticing that the characterization of a theory of English as information about the signs of English only makes sense as long as none of the principles introduces restrictive information about entities that are not signs. This is the case in the theory of Pollard and Sag 1987: All principles are expressed as relative pseudocomplement operations on pairs of 87 disjunctive feature structures that represent information about signs. As we will see, the grammar of English in Pollard and Sag 1994 contains principles that are implicative statements whose antecedents do not describe signs, and HPSG 94 grammars in general con-

[22]In fact, the information-based formalism for HPSG of Pollard 1989 already takes the alternative view, but it constructs the information about entities of each sort in a way that is very different from the sketch of a formalism of Pollard and Sag 1987. See Section 2.1.3 for a brief summary.

tain implicative statements with arbitrary descriptions in the antecedent. Nothing forbids corresponding principles of grammar in HPSG 87, but their restrictive content would not have an effect on $\text{E}\textsc{nglish}_{\text{S}}$.

There is a straightforward way to turn Pollard and Sag's theory of English signs, $\text{E}\textsc{nglish}_{\text{S}}$, into a theory about all entities in the English language. I call that theory $\text{E}\textsc{nglish}_{\text{NL}}$. The simple, basic idea is to apply the relative pseudocomplement operation two more times, and to express a word principle and a principle of grammar rules. The feature structures expressing the two new principles are then simply unified with the feature structures that express the universal and language-specific principles. In (12), I adopt Pollard and Sag's AVM notation to depict two 87 disjunctive feature structures that represent information about all lexical signs and about all phrasal signs:

(12) $\text{E}\textsc{nglish}_{\text{NL}} =$

$$\begin{array}{l} \text{P}_1 \wedge \ldots \wedge \text{P}_{m+n} \wedge \\ \left({}_{\textit{lexical-sign}}\, [\;] \Rightarrow (\text{L}_1 \vee \ldots \vee \text{L}_k)\right) \wedge \left({}_{\textit{phrasal-sign}}\, [\;] \Rightarrow (\text{R}_1 \vee \ldots \vee \text{R}_i)\right) \end{array}$$

The theory of all entities of English, $\text{E}\textsc{nglish}_{\text{NL}}$, is a piece of information that must hold of all entities of English. Each universal and each language-specific principle provides valid information about each entity of English, and so do the pieces of information in $\mathbb{HA}_{\text{PS}}$ that result from applying the relative pseudocomplement operation to, (a), the 87 disjunctive feature structure determined by the set that contains exactly the 87 abstract feature structure with one abstract node of sort *lexical-sign*, and the (successive) join of L_1 to L_k, and, (b), the 87 disjunctive feature structure determined by the set that contains exactly the 87 abstract feature structure with one abstract node of sort *phrasal-sign*, and the (successive) join of R_1 to R_i. The new way of constructing pieces of information about lexical and phrasal signs is, of course, the intuitionistic way of expressing the conditional information in $\mathbb{HA}_{\text{PS}}$ that, (a), if an entity is a lexical sign then at least one of L_1 to L_k represents valid information about it, and, (b), if an entity is a phrasal sign, then at least one of the 87 disjunctive feature structures expressing the grammar rules represents valid information about it.

2.1.3 Discussion

In this section, I discuss some issues that were left open in the formalization of Pollard and Sag 1987 of the previous section, and I put Pollard and Sag

1987 into the broader context of other information-based formalisms that have been applied to HPSG. However, all details of the information-based approach that will play a role in the discussion of HPSG 94 are already contained in Section 2.1.2. I begin with open issues of the feature structure formalism of the previous section. First, I discuss the possible meanings that can be ascribed to an HPSG 87 grammar. Then I turn to the question of adding a description language to HPSG 87 instead of picturing disjunctive feature structures informally in the principles of grammar. This leads to a brief historical overview over the information-based grammar formalisms that were put forth in the context of HPSG 87.

In Section 2.1.2, I defined complex entities whose purpose it is to serve as representations of partial information. This is of course only one side of the coin. Information-based grammar looks at information from two perspectives. HPSG 87 in particular sees the task of a grammar in specifying the partial information about the language that a mature, normal speaker of a natural language shares with his or her language community. In this formulation, information appears in two relevant ways: First, linguists study and specify (representations of) partial information, because partial information is what the speakers of a language share. Second, the partial information that a grammar specifies is information about a natural language. The partial information that the speakers of a language share specifies the natural language.

The dual role of partial information in HPSG 87 determines the task of linguists. When they write grammars, they must correctly specify the partial information that speakers have. To do that, linguists must pay attention to the properties of pieces of partial information as given by the subsumption relation and the operations of the Heyting algebra to which the pieces of information belong. The linguist studies the shape of the pieces of information and uses the operations of the Heyting algebra of disjunctive feature structures to construct new pieces of information. The principles of grammar and the theory of English, $\text{E}\textsc{nglish}_{\text{S}}$ (11), of Pollard and Sag 1987 are constructed with algebraic operations that depend on the particular properties of the representations of information. For example, 87 abstract feature structures are not adequate pieces of information in the principles of grammar of HPSG 87 because they do not constitute a Heyting algebra under subsumption. But the principles of grammar are expressed as the relative pseudocomplement operation on two pieces of partial information.

When linguists specify the partial information that native speakers have, they usually do that with an eye on the second role of information. Information is not an end in itself, it is also information *about* something. From that perspective, the important question is in which relationship information stands to the objects about which it is information. This is crucial, because the empirically observable data consists of tokens of the language. The correctness of the specification of partial information in a grammar can thus only be tested via a conventional correspondence between the abstract entities that the partial information specifies and tokens of the language. Given the currently available scientific methods, pieces of partial information are not empirical entities themselves, because we cannot observe them in the heads of language users.

Since HPSG 87 employs feature structures as information-bearing entities, they inherit the dual role of information. They are representations of the linguistic knowledge of the speakers and they are descriptions of language. Some potential practical difficulties with HPSG 87 come from the consequences that the algebraic operations on feature structures have for a possible semantics of feature structures as descriptions of linguistic entities.

Partial feature structures are first and foremost designed to be a set of complex entities with a subsumption relation that indicates their relative informativeness. In the previous section we studied partial feature structures as entities that model or represent information,[23] and we ultimately defined a Heyting algebra of 87 disjunctive feature structures, $\left\langle \mathbb{DFS}_{87}^{\Sigma}, \subseteq \right\rangle$, in which we can represent the theory of English of Pollard and Sag. However, we did not say anything about the second task of feature structures, how 87 disjunctive feature structures describe linguistic entities. Intuitively we might want to say that they describe an entity exactly if the information that they represent is partial information about the entity. Under that view, 87 disjunctive feature structures function as descriptions that are true or false when interpreted on objects in a domain, and the denotation of an 87 disjunctive feature structure is the set of linguistic objects that it describes. In fact, when Pollard and Sag 1987 explains subsumption, unification, join, and relative pseudocomplement, it often does so in semantic terms rather than in terms of algebraic operations.[24] For subsumption, it says that if a feature structure A subsumes a feature structure B that also means that whatever is described by B is also described by A. The result of unifying feature struc-

[23]Pollard and Sag use the terms *to represent* and *to model* interchangeably.

[24]See Pollard and Sag 1987, p. 30, pp. 40–41, and p. 43.

ture A and feature structure B is a feature structure that describes an object exactly if both A and B describe it; the disjunction of two feature structures, A and B, only describes an object, if A describes it or B describes it; and about the relative pseudocomplement operation Pollard and Sag (1987, p. 43) say that the most important consequence of its definition is that "if X is a linguistic object which is appropriately described by both C and A $\Rightarrow$ B, and C is subsumed by A, then C′ = C $\wedge$ B is also an appropriate description of X."

As to technical remarks concerning the formalization of HPSG 87, Pollard and Sag 1987 focuses on feature structures as representations of linguistic information and says little about feature structure denotation. However, from the fact that 87 disjunctive feature structures under subsumption constitute a Heyting algebra and not a Boolean algebra, it follows immediately that a classical interpretation of 87 disjunctive feature structures and the algebraic operations on them is not possible.[25] Most importantly, standard results tell us that the pseudocomplement operation cannot be interpreted as classical negation, and the relative pseudocomplement operation cannot be interpreted as classical implication. More generally, the classical assumptions about the denotation of the expressions of logical languages and the standard logical connectives cannot hold of 87 disjunctive feature structures and the operations of their Heyting algebra:

(13) In general, we cannot define a classical semantics such that:

a. For each $\mathbb{D}_1 \in \mathbb{DFS}^{\Sigma}_{87}$, for each $\mathbb{D}_2 \in \mathbb{DFS}^{\Sigma}_{87}$,
$\mathbb{D}_2 \subseteq \mathbb{D}_1$ iff the denotation of $\mathbb{D}_2$ is a subset of or equals the denotation of $\mathbb{D}_1$,

b. for each $\mathbb{D}_1 \in \mathbb{DFS}^{\Sigma}_{87}$, for each $\mathbb{D}_2 \in \mathbb{DFS}^{\Sigma}_{87}$,
the denotation of the unification of $\mathbb{D}_1$ and $\mathbb{D}_2$ = the intersection of the denotation of $\mathbb{D}_1$ with the denotation of $\mathbb{D}_2$,

c. for each $\mathbb{D}_1 \in \mathbb{DFS}^{\Sigma}_{87}$, for each $\mathbb{D}_2 \in \mathbb{DFS}^{\Sigma}_{87}$,
the denotation of the disjunction of $\mathbb{D}_1$ and $\mathbb{D}_2$ = the union of the denotation of $\mathbb{D}_1$ with the denotation of $\mathbb{D}_2$,

[25] King (1989, pp. 61–78) proves this result for his formalization of Pollard and Sag 1987.

d. for each $\mathbb{D}_1 \in \mathbb{DFS}^{\Sigma}_{87}$, for each $\mathbb{D}_2 \in \mathbb{DFS}^{\Sigma}_{87}$,

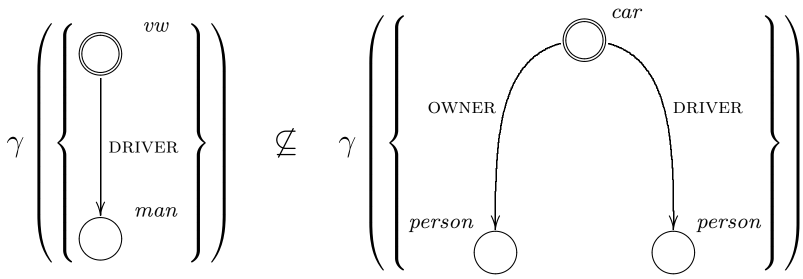
the denotation of the relative pseudocomplement of $\mathbb{D}_1$ and $\mathbb{D}_2$ = the union of the set complement of the denotation of $\mathbb{D}_1$ with the denotation of $\mathbb{D}_2$,

e. for each $\mathbb{D} \in \mathbb{DFS}^{\Sigma}_{87}$,

the denotation of the negation of $\mathbb{D}$ = the set complement of the denotation of $\mathbb{D}$.

The consequences for the linguist of the non-classical approach of HPSG 87 can best be illustrated with a small example. (14) shows a fact about 87 disjunctive feature structure subsumption under the car scenario signature of Figure 2.3:

(14)



Given the definitions of 87 disjunctive feature structure subsumption, the right disjunctive feature structure does not subsume the left disjunctive feature structure. Reading the information specified by these feature structures and considering the underlying 87 signature, this can be surprising. Informally, (14) tells us that information about a car whose owner and driver are persons is not necessarily information about a VW with a male driver, although we know from the signature that *vw* is subsumed by *car*, DRIVER and OWNER are appropriate for *vw*, and *person* subsumes *man*. In a classical setting, we would certainly expect that if the feature structure to the left is true of an object in a domain, then the feature structure to the right is also true of the object in the domain. In short, classical intuitions about the meaning of descriptions can be misleading in HPSG 87. Similar effects as in (14) can be observed for the relationship between the behavior of the relative pseudocomplement operation and intuitions about the meaning of implication.

Of course, these observations do not mean that the approach of HPSG 87 is wrong. They merely emphasize the point that linguists who work in this framework should be familiar with the kind of non-classical interpretation that is needed here, lest their intuitions about the meanings of their grammars lead them astray. It also shows that, for a complete formalism of HPSG 87, we should add an intuitionistic semantics to the algebra of 87 disjunctive feature structures. Only if we have a well-defined semantic interpretation of the pieces of partial information do we know what the principles of grammar exactly mean. Moshier and Rounds 1987 outline an intuitionistic semantics for feature structures which we could use to provide the necessary definitions. However, giving a semantics for HPSG 87 is beyond the scope of my thesis.

Alternatively, one might argue that representations of information suffice for the purposes of grammar writers, and a semantics for the representations of linguistic knowledge is not necessary. While this view is possible in principle, it is quite unsatisfactory for the theoretical linguist, because it does not even address the relationship between grammars and the empirical domain of observable tokens of expressions of natural languages. It remains unclear what a natural language is, and how the predictions of a grammar can be falsified by data.

Another issue besides the semantics of the pieces of information that Pollard and Sag 1987 does not address directly is the question of the formal status of the AVM diagrams that serve as illustrations of feature structures and grammatical principles. Recall that the AVMs in Pollard and Sag 1987 are taken to be a convenient and informal way of depicting feature structures that serves the same purpose as the graph notation that I have adopted. These pictures must be strictly distinguished from a formal description language for which a denotation function is defined.[26] Starting with Pollard 1989, succeeding information-based formalisms for HPSG included logical description languages for partial feature structures. These description languages for feature structures build on early work by Kasper and Rounds 1986 and are characterized by the fact that they do not provide negation or implication. For example, the language SRK (sorted Rounds-Kasper logic) of Pollard 1989 provides sort descriptions, conjunction, disjunction, path equations (for describing the fact that two paths lead to the same node), and formulae for

[26]The fact that it is possible to identify certain classes of informationally equivalent descriptions with the feature structures that they denote (as summarized in Pollard 1999, p. 283) and to treat these representations of the feature structures as the feature structures they represent is independent of the necessity of this distinction.

describing attribute values. If desired, it would be straightforward to augment the formalism of the previous section with such a description language for feature structures. The description language could serve to describe the 87 disjunctive feature structures of the principles of grammar, but it would not change anything that is essential to the formalism. In HPSG 87, the operations of the Heyting algebra of feature structures determine the meaning of the grammar regardless of how exactly the feature structures in the theory of a language are specified. In the theory of English, $\text{E}\textsc{nglish}_{\text{S}}$, of Pollard and Sag, the 87 disjunctive feature structures that must be specified are the two arguments of the relative pseudocomplement operation of each grammatical principle, the feature structures that represent information about the lexical signs of English, and the feature structures that express the grammar rules. It does not make a difference if these 87 disjunctive feature structures are given directly (as I did it), in an abbreviatory fashion as an element of the equivalence class of descriptions that represents the feature structure, or using formulae of a description language of feature structures and a satisfaction relation between formulae and disjunctive feature structures.

The formalization of Pollard and Sag 1987 that I presented in Section 2.1.2 can be understood as a generalization of the formalization of King (1989), who followed the original informal characterization of the formalism more closely. Therefore, the basic feature structures of King, which correspond to our 87 abstract feature structures, are required to be acyclic, and King's feature structures, which correspond to my 87 disjunctive feature structures, are defined as sets of pairwise subsumption-incomparable basic feature structures, i.e., no basic feature structure in a King feature structure subsumes a distinct second basic feature structure in the King feature structure. To guarantee that his feature structures constitute a Heyting algebra under subsumption, King must introduce a restriction on basic feature structures that we did not need. The crucial point is that feature structures that are defined as sets of pairwise subsumption-incomparable basic feature structures only form a Heyting algebra under (feature structure) subsumption[27] if there are no infinite ascending chains of basic feature structures in the partial order of basic feature structures under (basic feature structure) subsumption. This is only the case if basic feature structures are acyclic and if the sort hierarchy

[27]A King feature structure, $\mathbb{D}_1$, subsumes a King feature structure, $\mathbb{D}_2$, iff each basic feature structure in $\mathbb{D}_1$ subsumes some basic feature structure in $\mathbb{D}_2$. See also footnote 19 for discussion.

does not contain an infinite ascending chain of sorts. These restrictions of King are compatible with Pollard and Sag 1987, since it says that basic feature structures are acyclic, and the sort hierarchy of the grammar is finite. A finite sort hierarchy cannot comprise an infinite ascending chain of sorts.

The construction using the Alexandrov topology of the dual preorder of $\langle \mathbb{AFS}^{\Sigma}_{87}, \leq \rangle$ is more general, since it allows for divergent definitions of abstract feature structures while it preserves a Heyting algebra of the disjunctive feature structures that are constructed from these abstract feature structures. For example, we can impose the acyclicity condition on abstract feature structures, in which case the resulting Heyting algebra of disjunctive feature structures is isomorphic to King's Heyting algebra of feature structures; or we can make the conditions on them more liberal and admit infinite abstract feature structures, meaning that an abstract feature structure may have infinitely many (abstract) nodes, and we still obtain a Heyting algebra of disjunctive feature structures.

Pollard 1989 proposes a domain-theoretic formalism for information-based HPSG that addresses and answers the open questions about the sketch of a formalism of Pollard and Sag 1987. Pollard defines a logical description language for feature structures (SRK), together with an equational calculus. The Pollard formalism comprises variants of concrete and abstract feature structures, which he calls concrete ordinary feature structures and abstract ordinary feature structures, respectively, but he constructs the algebra of *disjunctive abstract ordinary feature structures* differently and for a different purpose compared to our 87 disjunctive feature structures. Pollard uses a Smyth power domain construction to define his algebra of disjunctive feature structures, which also yields a complete Heyting algebra, although it is not a Heyting algebra in which the relative pseudocomplement of each pair of elements represents conditional information in the sense of HPSG 87. Pollard rather uses this complete Heyting algebra to prove a fixed point theorem about the denotation of his grammars.[28] The basic idea underlying this approach is that each sort in the sort hierarchy of the signature of a grammar is assigned a disjunctive feature structure which represents all the information

[28]I am grateful to Carl Pollard for clarifying discussions on these issues. Carl Pollard pointed out to me that with a domain-theoretic approach, an alternative complete Heyting algebra for conditional information becomes available, namely the Scott topology of the ideal-completion of the dual preorder of $\langle \mathbb{AFS}^{\Sigma}_{87}, \leq \rangle$. However, Section 2.1.2 deliberately avoids domain-theoretic notions, since they have nothing to do with HPSG 94, and they do not illuminate the relationship between HPSG 87 and HPSG 94.

about possible entities of that sort. The denotation of a grammar is then obtained via a fixed point construction that construes a language as the partial information that the grammar makes available for each sort of object. Concluding this line of research, Pollard and Moshier 1990 augments the feature structures of the formalism of Pollard 1989 with set-valued nodes, and Moshier and Pollard 1994 investigates the domain-theoretic properties of the augmented formalism. Notice that this approach moves away from trying to express the principles of grammar in the formalism exactly as they are written down in the grammar of English of Pollard and Sag 1987. The principles of English in that grammar are meant to be conditional statements whose antecedents are typically expressed as complex feature structures. At some level of abstraction, the idea of associating each sort with the information about objects of that sort can be understood as allowing only sorts as the antecedents of implicative statements in the grammar. Some transformation of grammars such as the grammar of English in Pollard and Sag 1987 is therefore necessary in order to bring the principles into a format that the formalism can express.

Carpenter 1992 is interested in systems that allow for efficient processing with feature structures. For that purpose, Carpenter develops a logical description language together with a sound and complete logic.[29] An augmented version of disjunctive feature structures that includes an inequation relation serves as canonical models of so-called constraint-systems that employ expressions of the description language. Carpenter's constraint systems correspond to the grammars of linguistics. As in the formalisms above, the disjunctive feature structures represent partial information about the entities in the empirical domain. Disjunctive feature structures constitute a lattice under subsumption. Carpenter's formalism is designed in such a way that for each description of the logical description language, there is a disjunctive feature structure that is its most general satisfier, i.e., it subsumes all other disjunctive feature structures that satisfy the description. In order to guarantee this result, certain restrictions must be imposed on the description language. There is no general negation of feature structure descriptions, and there is no general implication. As in the formalism of Pollard 1989, a Carpenter grammar associates with each sort the partial information about objects of that sort that is provided by a disjunctive feature structure. In Carpenter's formalism, each one of these disjunctive feature structures is the

[29] The description language of Carpenter is based on Pollard's (1989) SRK.

most general satisfier of the descriptions associated with that sort in a given grammar. The restrictions on the description language are unproblematic for Carpenter, since his starting point is the broader question of how to compute with feature structures rather than the more specific question of designing a description language that is as close as possible to existing grammars written in some particular grammatical framework. The potential application of his formalism to HPSG is only one among many factors that influence his design decisions. As with the formalism of Pollard 1989, expressing the grammar of English of Pollard and Sag 1987 in Carpenter's formalism requires grammar transformations in order to bring the principles into a format that the formalism can express.

The logical description language of Carpenter's formalism can be regarded as a link between the information-based approach of HPSG 87 and the object-based approach of HPSG 94, to which I will turn in the next section. However, there are significant differences between Carpenter's formalism and formalisms for HPSG 94 regarding both their description languages and the feature structure interpretations of description language expressions. Referring to the object-based formalisms of Johnson 1988, Smolka 1988, and King 1989, Carpenter (1992, p. 51) observes that "[o]ther researchers [...] use similar, but much more powerful logical description languages and take a totally different view of the nature of interpretations or models for these languages. The main difference in their descriptions is the presence of general description negation, implication, and variables."[30] Especially with regard to HPSG, general description language negation and implication plays an important role in the design of a formalism that strives to permit a direct formal interpretation of the complex implicational principles of grammar as they are stated by linguists. Carpenter stresses the crucial difference between feature structures whose purpose it is to represent partial information—as they do in HPSG 87 and in his formalism—and feature structures that are total objects in the models of a grammar: "We should point out that this notion of feature structures as representing partial information is very different from the notion of feature structure developed by Johnson (1988), Smolka (1988) and King (1989). They interpret feature structures as total objects and introduce a logical language in which to express partial information about these total objects" (Carpenter, 1992, p. 35). These conceptual differences will be-

[30]As we will see below, true variables are not present in the object-based formalism of King 1989, but they will be in my extension of that formalism.

come evident in the investigation in the following section of the formalism for HPSG 94 that started with the development of a formal language for HPSG in King 1989.

Despite the conceptual differences, there is a special case of Carpenter's system that can be seen as a point of contact between the partial-information view of HPSG 87 and the total-object view of HPSG 94. Carpenter studies the consequences of putting various restrictions on feature structures. One special case is to require that, given a signature, each (abstract) node of a (possibly infinite) abstract feature structure be of a sort that does not subsume any other sort besides itself; for each node of each feature structure, exactly each attribute that is appropriate to the sort of the node be present, and the attribute values obey appropriateness; and for each pair of nodes of a feature structure, it be represented whether or not they are identical. As other researchers noticed before,[31] in that case the subsumption relation between feature structures becomes fully degenerate, and each feature structure only subsumes itself. Moreover, the resulting feature structures can no longer reasonably be seen as representations of *partial* information. Since every feature structure is as complete and as specific as it can possibly be, it is rather a representation of *total* information. While these total representations go against the spirit of Carpenter's system, which is designed for the partial-information view, they correspond to the total 94 abstract feature structures of HPSG 94 (DEFINITION 32, page 113) that we will discuss in Section 2.2. As we will see, the set of 94 abstract feature structures admitted by an HPSG 94 grammar can be regarded as the set of mathematical representations of the object types of a natural language. In effect, the total feature structures of Carpenter occupy an intermediary position between the representations of partial information of unification-based formalisms and the total objects in models of constraint-based grammars. Under the partial-information view of unification-based formalisms, they represent the special case of total information; under the total-object view of constraint-based formalisms, they represent equivalence classes of total objects.[32]

[31] For example, see King 1999, pp. 347–348, fn. 11.

[32] See Section 2.2.2.3 for a detailed discussion of the status of abstract feature structures in HPSG 94.

2.2 Language as a Collection of Total Objects

Characterizing the formal foundations of HPSG 94 is in some respects easier and in others harder than characterizing the formal foundations of HPSG 87. It is easier because much fewer mathematical constructs are needed in order to say what an HPSG 94 grammar is and what it means. The main reason for the simplicity of HPSG 94 in comparison to HPSG 87 is that HPSG 94 no longer sees the task of grammar in specifying representations of partial information about linguistic entities. The subsumption relations between feature structures that HPSG 87 puts in place to capture the relative informativeness of pieces of information disappear completely, and with them all algebraic operations such us unification and relative pseudocomplement. All that is left in HPSG 94 of the formal devices of HPSG 87 of the previous section are the signatures. But where HPSG 87 used the alphabets of the signatures to build feature structures, HPSG 94 uses signatures to define a formal language and a class of interpretations for each signature. As we will see, the interpreting structures of HPSG 94 are not necessarily feature structures. A grammar is a signature together with a set of expressions of the formal language. A model theory determines what a grammar means. If HPSG 94 refers to a notion of relative informativeness at all, it is given independently by a logical entailment relation between descriptions of the description languages.

Although the outline of the new architecture of grammar seems to indicate a straightforward account, it is not easy to pinpoint the formal foundations of HPSG 94, because Pollard and Sag 1994 is much less precise about many substantive details of the envisioned formalism than Pollard and Sag 1987. For example, it is not even clear from the outset what a formal language for HPSG 94 should be able to express in order to be considered adequate. Pollard and Sag give very concise descriptions in natural language of all important principles of their grammar of English in the appendix of the book, but they do not define a syntax and a semantics of a formal language in which they can be expressed. They consider a formalization of their theory an important goal, but at the same time, they want to avoid the technical design decisions that a formalization necessitates, and they fear that the use of a rigorously defined description language would impede understanding of the principles of their grammar by the intended audience, *viz.* (not necessarily formally inclined) linguists (Pollard and Sag, 1994, p. 9). An informally introduced language of AVM diagrams and various references to the literature on logical languages for reasoning about feature structures indicate the minimal requirements that must be met. My main criticism of previous attempts at a mathematically rigorous rendering of HPSG 94 will be that these attempts fail to include quantificational and relational expressions, which I will show to play an important role in an accurate and direct formalization of the grammar of English of Pollard and Sag 1994, although this might not be immediately obvious. Especially the necessity of quantificational expressions is far from evident from the informal AVM expressions, and can only be inferred from the intended meaning of the given principles. Regarding the meaning of HPSG 94 grammars, the situation is even more difficult. Pollard and Sag 1994 speaks of the entities in the denotation of the expressions of the formal language as totally well-typed and sort resolved feature structures, and it contains examples that depict them in graph notation, but it never says if the feature structures are concrete or abstract, and it is not explicitly said if any other conditions on feature structures beyond total well-typedness and sort resolvedness should be imposed. For example, it can make a difference for the properties of the formalism if feature structures with infinitely many nodes are permitted or forbidden, and different feature structure formalisms have chosen different answers to the question of infinite sets of nodes. Pollard and Sag 1994 also does not address the additional question which collection of feature structures a grammar determines. *Prima facie*, it even remains an open question why Pollard and Sag populate the models of grammar with feature structures instead of other, simpler objects. The traditional idea of treating linguistic entities as bundles of features does not necessarily need to be expressed by feature structures. However, the initial thinking about the meaning of HPSG 94 grammars in terms of feature structures bridges the gap between HPSG 94 and the preceding, unification-based HPSG formalism(s), the semantics of the related logical language of typed feature logics of Carpenter 1992, and computational implementation efforts of HPSG grammars in unification-based frameworks.

Different possible formalizations of the basic ideas of the grammar formalism outlined in Pollard and Sag 1994 have been explored in King 1994, 1999, and in Pollard 1999. The purpose of the current section is to summarize these formalizations of HPSG 94 and to discuss what they have in common and where they differ. Where relevant, I will discuss what the mathematical constructs that occur in these formalizations have in common with similar notions in HPSG 87. I will begin with a broad characterization of the architecture of HPSG 94 as the formalism is sketched in Pollard and Sag 1994. This informal characterization will later serve as a point of reference when we ask how the concrete mathematical formalisms that we will see are related to the original conception.

2.2.1 The Idea

Under the perspective of Pollard and Sag 1994, an HPSG 94 grammar determines a collection of object types. Object types are idealizations of isomorphism classes of possible well-formed linguistic entities. Since Pollard and Sag 1994 does not commit itself to a specific view of what object types are and what their precise ontological status is, it is not necessary to explicate the notion here beyond an intuitive understanding. For the moment it suffices to assume that object types are some kind of abstraction of possible linguistic tokens or events that occur in the real world and can be observed when a speaker makes an utterance. We will get a clearer idea of the nature of object types when we see their mathematical reconstructions in terms of the HPSG 94 formalisms of King and Pollard below. The collection of object types is specified by means of a highly expressive formal language with classical Boolean connectives and relational expressions of the kind familiar from feature logics. A signature provides the non-logical symbols such as attribute symbols, sort symbols and relation symbols of the formal language, and the universal and language-particular principles of grammar are stated as formulae of the formal language.

The object types of a given natural language are modeled by feature structures that must satisfy the set of formulae in the grammar. Most importantly, feature structures that are models of object types are total. Pollard and Sag 1994 does not say much about what kind of feature structures it intends to use as models of object types, but it insists on two important properties of the feature structures that crucially distinguishes them from any of the feature structures of HPSG 87: The feature structures must be *totally well-typed* and *sort resolved*. These two properties are tied to the fact that the feature structures of HPSG 94 are total models of object types and not representations of partial information as they are in HPSG 87. The notions of well-typed and totally well-typed feature structures are defined in Carpenter 1992, p. 88 and p. 95, respectively. A feature structure is *well-typed* if and only if each of its (possibly abstract) nodes only has attributes (and attribute values) that are appropriate for the sort of the node according to the signature. For example, under the signature of Figure 2.2, a node of sort *woman* may only have the attribute LIKES-BEST, but not the attributes OWNER or DRIVER, because

LIKES-BEST is appropriate to *woman* but OWNER and DRIVER are not. A feature structure is totally well-typed if and only if it is well-typed, and each node has in fact every attribute that is appropriate to it. For feature structures under the signature of Figure 2.2 that means that every node of sort *vw* or *bmw* must have exactly the attributes OWNER and DRIVER, and every node of sort *man* or *woman* must have exactly the attribute LIKES-BEST (with values as permitted by the signature): All and only the appropriate attributes must be present. A feature structure is sort resolved if and only if each of its nodes is of a maximally specific sort.[33] The sort resolved feature structures under our example signature might only have nodes of the sorts *vw*, *bmw*, *man* and *woman*, since they are the species of the signature. A sort resolved feature structure may not have a node of sort *person*, *car* or *top*, because they are not maximally specific.

The idea behind modeling object types with totally well-typed and sort resolved feature structures is that the feature structures are mathematical idealizations of isomorphism classes of linguistic entities. The ontological category of the entities (as expressed by the sort labels of the nodes of feature structures) determines their properties (as expressed by the attributes that a node has), and each linguistic entity is of exactly one ontological category with a fixed inventory of properties. Therefore, the feature structures of HPSG 94 cannot be partial. Since it is the partiality of feature structures that informs definitions of feature structure subsumption such as the one provided in DEFINITION 6 above, feature structure subsumption disappears together with the partiality of feature structures; and there is no interesting way in which feature structures can be said to represent information about linguistic objects. They are themselves total models of object types.

In summary, Pollard and Sag 1994 construes a natural language as a collection of object types that are abstractions of possible linguistic tokens. Object types are modeled by totally well-typed and sort resolved feature structures. The set of feature structures that model the object types of a natural language is specified by a set of descriptions of a formal language of a feature logic. Each of the feature structures that models an object type satisfies all descriptions in the grammar. As the means of specifying natural languages the formal languages of HPSG 94 replace the disjunctive feature structures of HPSG 87. Since the expressions of the formal lan-

[33] A sort in a sort hierarchy is maximally specific if it does not have any subsorts except itself. The maximally specific sorts are sometimes called *species*.

guage are the means by which a linguist characterizes a natural language in HPSG 94, the description language is of particular interest in a formalization of object-based HPSG. To make the intuitions that underly the architecture of HPSG 94 mathematically precise, it is thus necessary to define a class of formal languages in which the principles of Pollard and Sag 1994 can be faithfully expressed, and to state a model theory for these languages that makes explicit which collection of feature structures is licensed by a set of descriptions of each formal language.

2.2.2 Formal Reconstruction and Different Perspectives

I begin my formal reconstruction of HPSG 94 with the definition of the formal languages of King's SRL (Speciate re-entrant logic) in Section 2.2.2.1. SRL has been used as the common basis of three different ways of characterizing the meaning of an HPSG 94 grammar. The first one that I will present (Section 2.2.2.2) is the account of King 1999, which holds that the task of a grammar of a natural language is to determine a certain subclass of models of the entire class of models of the grammar. King calls the class of models that the grammar determines the class of *exhaustive models* of the grammar. According to King, the class of exhaustive models comprises one particular exhaustive model, $\mathcal{L}$, that contains exactly the system of all possible entities in the natural language. The actual tokens of the natural language that we observe in utterances in the real world are in $\mathcal{L}$, as are the non-actual tokens that are licensed by the grammatical system but never realized as actual utterance events in our world. The other models in the class of exhaustive models are indistinguishable from $\mathcal{L}$. In Section 2.2.2.3, I show how King's exhaustive models are related to the object types of Pollard and Sag 1994. The object types of Pollard and Sag will be formalized as totally well-typed, sort resolved and potentially infinite abstract feature structures that stand in an abstraction relation with the entities in any given exhaustive model of a grammar. Both the abstraction relation and the abstract feature structures are closely related to the constructs of the same name that we have seen in HPSG 87. In Section 2.2.2.4, I will turn to Pollard's (1999) notion of the *strong generative capacity* of an HPSG 94 grammar. The strong generative capacity of a grammar is a set of pairwise non-isomorphic mathematical representations of the possible linguistic tokens of the language. Each representation is isomorphic to each possible token in an equivalence class of indistinguishable possible tokens of the language, and for each equivalence class of indistinguishable possible tokens, the strong generative capacity of the grammar contains a representation of the tokens in the class. One of the three definitions of the strong generative capacity of a grammar of Pollard 1999 will establish a direct connection to the exhaustive models of a grammar, thus completing the picture of how the three explanations of the meaning of a grammar are related.

2.2.2.1 SRL

My presentation of SRL follows the definitions of King 1999,[34] which generalizes some of the original definitions of King 1989. For ease of comparison with corresponding notions in HPSG 87 and with the extended formalism of HPSG 94 of Chapter 3, I adopt the notational conventions that I use there.

The fundamental intuition about formal languages that underlies SRL is that each expression of its formal languages is true or false *of an entity* in an interpretation. Given an interpretation, an expression thus denotes a set of entities. It is because of this property that logics like SRL are sometimes referred to as logics of descriptions. The idea is that the languages are used to describe entities. As formal languages of HPSG, they are used to describe the entities that constitute a natural language. SRL provides a class of formal languages, each of which consists of a signature and a class of interpretations of the signature. Each signature provides the non-logical symbols from which the formal language is constructed.

Definition 15 Σ *is an **SRL signature** iff*

> Σ *is a triple* $\langle \mathcal{S}, \mathcal{A}, \mathcal{F} \rangle$,
>
> $\mathcal{S}$ *is a set,*
>
> $\mathcal{A}$ *is a set, and*
>
> $\mathcal{F}$ *is a total function from the Cartesian product of* $\mathcal{S}$ *and* $\mathcal{A}$ *to the power set of* $\mathcal{S}$.

I call each element of $\mathcal{S}$ a *species*, each element of $\mathcal{A}$ an *attribute*, and $\mathcal{F}$ the *appropriateness function*. The symbols $:$, $\sim$, $\approx$, $\neg$, $[$, $]$, $\wedge$, $\vee$, and $\rightarrow$

[34] The definitions of King 1999 are consistent with the ones presented in Pollard 1999.

are reserved symbols, and I will henceforth assume that none of them are a species or an attribute symbol. In contrast to King 1989, infinite sets of species are admitted. The original definition was more restrictive, because King 1989 was interested in logical aspects of SRL that will not play a role below.

It is useful to have a compact terminology to talk about the appropriateness of attributes and species. I say that an attribute α is appropriate to a species σ if $\mathcal{F}\langle\sigma,\alpha\rangle$ is a nonempty set of species. A species σ' is appropriate for an attribute α at some species σ if σ' is an element of $\mathcal{F}\langle\sigma,\alpha\rangle$.

A striking difference between an SRL signature and an 87 signature is that an SRL signature does not include a sort hierarchy. Instead, it simply provides a set of species. This simplification also means that the appropriateness function is much simpler, because it no longer has to enforce the inheritance of appropriate attributes and attribute values in accordance with the ordering of the sorts in the sort hierarchy. In Pollard and Sag 1994 the appropriateness function is given in so-called "feature declarations" that obey the 87 appropriateness conditions that we have seen in 87 signatures: If an attribute α is 87 appropriate to some sort σ, then it is also 87 appropriate to all of its subsorts; and the sort that is the value of the 87 appropriateness function at the subsorts of σ and α is at least as specific as the value of $\mathcal{F}$ at σ and α. At first blush, the lack of an explicit sort hierarchy seems to be a problem for using SRL in formalizing HPSG 94 grammars, because, just as HPSG 87 grammars, they make heavy use of sort hierarchies and of attribute inheritance. However, King 1999, pp. 329–331, shows that the inclusion of a sort hierarchy in the signature of the formal languages for HPSG 94 is mathematically superfluous and that the essential information that is encoded in sort hierarchies can be expressed without them. Before I can explain why this is the case, I need to introduce interpretations of signatures:

Definition 16 *For each SRL signature* $\Sigma = \langle\mathcal{S},\mathcal{A},\mathcal{F}\rangle$, I *is a* Σ ***interpretation*** *iff*

> I *is a triple* $\langle\mathsf{U},\mathsf{S},\mathsf{A}\rangle$,
>
> U *is a set,*
>
> S *is a total function from* U *to* $\mathcal{S}$,
>
> A *is a total function from* $\mathcal{A}$ *to the set of partial functions from* U *to* U,

for each $\alpha \in \mathcal{A}$ *and each* $u \in \mathsf{U}$,

$$\mathsf{A}(\alpha)(u) \textit{ is defined iff } \mathcal{F}\langle \mathsf{S}(u), \alpha\rangle \neq \emptyset, \textit{ and}$$
$$\textit{if } \mathsf{A}(\alpha)(u) \textit{ is defined then } \mathsf{S}(\mathsf{A}(\alpha)(u)) \in \mathcal{F}\langle \mathsf{S}(u), \alpha\rangle.$$

U is the *set of entities in the universe*, S is the *species assignment function*, and A is the *attribute interpretation function*.

S partitions the universe of entities by assigning each entity exactly one species. I sometimes simply say that an entity has a species. Informally, I also say that a species denotes a set of entities, which is the set of entities that have that species. The attribute interpretation function provides a denotation for the attribute symbols. Each attribute denotes a partial function from entities to entities. The attribute interpretation function must obey appropriateness. That means that an attribute is defined on an entity exactly if the attribute is appropriate to the species of the entity; and if an attribute is defined on an entity, u, then the species of the entity u' which is the result of interpreting α on u, is appropriate for the species of u and α.[35]

With the definition of Σ interpretations in hand, I can now return to the question of how SRL can express HPSG 94 sort hierarchies without representing them in its signatures. The reason that an SRL signature suffices to express the sort hierarchies of HPSG 94 grammars as characterized in Pollard and Sag 1994, pp. 395–396, is the intended interpretation of sort hierarchies. Algebraically, the sort hierarchies of HPSG 94 are finite partial orders.[36] The terminology that Pollard and Sag 1994 uses to describe the direction of the ordering relation is not entirely consistent. Pollard and Sag assume a sort, *object*, which subsumes all other sorts and denotes all entities, and they call all sorts subsorts of *object*. Sorts that do not have a proper subsort are called maximal or maximally specific. This terminology agrees with the sort hierarchy of 87 signatures (DEFINITION 1, page 30). On the other hand, Pollard and Sag 1994, pp. 17–18, assumes a reversed ordering that agrees with the

[35]The reader might notice that the behavior of attribute interpretation with respect to appropriateness corresponds to the total well-typedness of concrete feature structures.

[36]Pollard and Sag 1994, p. 395, also postulates that, for each sort σ_1, for each sort σ_2, for each sort σ_3, if σ_1 is a subsort of σ_2 and of σ_3, then either σ_2 is a subsort of σ_3 or *vice versa*. In other words, no sort is an immediate proper subsort of two distinct sorts. While that condition holds for the grammar of Pollard and Sag 1994, it is not met by grammars with so-called multiple inheritance hierarchies (see Kathol 1995; Sag 1997; Przepiórkowski 1999a, among many others). Since I am interested in a formalism that captures as much of the literature of HPSG 94 as possible, I will ignore Pollard and Sag's additional restriction.

one in Carpenter 1992, where the more inclusive sorts are lower in the ordering. For example, according to that convention *sign* is immediately below the sorts *word* and *phrase*. Since the direction of the ordering is merely a matter of convention, and no mathematical properties depend on which one we choose, I choose to retain the direction of ordering of 87 signatures to keep my terminology coherent across the different formalisms, and because that ordering is consistent with the intuitive understanding of the terms subsort, supersort, and maximally specific sort.

With respect to the denotation of sorts, Pollard and Sag assume that if σ' is a subsort of σ, then σ' denotes a subset of σ. The root sort, *object*, denotes all entities in the interpretation. Each entity is in the denotation of exactly one maximally specific sort. Given the assumptions about the sort hierarchy, the denotation of each of its sorts is thus fully determined by the denotation of the maximally specific sorts that it subsumes. Given the assumptions about attribute inheritance of Pollard and Sag 1994, the domain and range of the function in the denotation of each attribute is also fully determined by the appropriateness of the attribute to maximally specific sorts. For each signature with a sort hierarchy of the kind described above and with the respective attribute inheritance, we can thus give a function that maps it to an SRL signature such that each interpretation of the original signature is also an interpretation of the derived SRL signature. In addition, as King 1999, p. 330, shows, the use of nonmaximal sorts in descriptions of the formal language can easily be understood as metasyntactic notation for bigger, disjunctive descriptions that only use maximal sorts. It follows immediately that the formal languages of SRL are capable of expressing all aspects of HPSG 94 that are related to sort hierarchies.

The SRL signature in Figure 2.4 illustrates how a sort hierarchy is expressed in an SRL signature without having it explicitly included. It corresponds directly to the 87 signature with sort hierarchy in Figure 2.2. $\mathcal{S}$ is the set of maximally specific sorts of the 87 signature. The set of attributes, $\mathcal{A}$, does not change. The appropriateness function, $\mathcal{F}$, is now a total function, and it states for every species and every attribute which species are appropriate for them. If no species is appropriate for a given pair then the value of $\mathcal{F}$ at that pair is the empty set. For example, DRIVER is not appropriate to *man*, thus $\mathcal{F}\langle man, \text{DRIVER}\rangle = \emptyset$. For every species, σ, in the 87 signature for which an attribute, α, is 87 appropriate, the value of the corresponding appropriateness function in our SRL signature without sort hierarchy, $\mathcal{F}\langle\sigma,\alpha\rangle$, is the set of the maximally specific sorts subsumed by its sort value in the 87 signature. For example, since *person* is appropriate to the pair '$\langle vw, \text{OWNER}\rangle$' in the 87 signature of Figure 2.2, the value of $\mathcal{F}\,\langle vw, \text{OWNER}\rangle$ is now the set $\{man, woman\}$.

Figure 2.4: An example of an SRL signature

$\mathcal{S} = \{vw, bmw, man, woman\}$,

$\mathcal{A} = \{\text{DRIVER}, \text{OWNER}, \text{LIKES-BEST}\}$, and

$$\mathcal{F} = \left\{ \begin{array}{l} \langle\langle vw, \text{OWNER}\rangle, \{man, woman\}\rangle, \langle\langle vw, \text{DRIVER}\rangle, \{man, woman\}\rangle, \\ \langle\langle vw, \text{LIKES-BEST}\rangle, \{\}\rangle, \langle\langle bmw, \text{OWNER}\rangle, \{man, woman\}\rangle, \\ \langle\langle bmw, \text{DRIVER}\rangle, \{man, woman\}\rangle, \langle\langle bmw, \text{LIKES-BEST}\rangle, \{\}\rangle, \\ \langle\langle man, \text{LIKES-BEST}\rangle, \{vw, bmw, man, woman\}\rangle, \\ \langle\langle man, \text{DRIVER}\rangle, \{\}\rangle, \langle\langle man, \text{OWNER}\rangle, \{\}\rangle, \\ \langle\langle woman, \text{LIKES-BEST}\rangle, \{vw, bmw, man, woman\}\rangle, \\ \langle\langle woman, \text{DRIVER}\rangle, \{\}\rangle, \langle\langle woman, \text{OWNER}\rangle, \{\}\rangle \end{array} \right\}$$

From a more general perspective, the fact that an explicit sort hierarchy is mathematically superfluous in HPSG 94 but not in HPSG 87 is a consequence of the fact that signatures no longer form the basis of a subsumption ordering of feature structures that is meant to capture their relative degree of informativeness. The lack of a subsumption ordering on the level of entities is directly reflected by the fact that there are no entities of nonmaximal sorts in the interpretations of SRL signatures. The entities are no longer thought of as information bearing objects, but as models of linguistic entities of some kind.

The fact that each 87 signature with a sort hierarchy can be translated into a corresponding SRL signature without a sort hierarchy that has the same interpretations does not mean that sort hierarchies are not linguistically relevant. In principle, it is conceivable that psycholinguistic evidence which supports the representational reality of a sort hierarchy can be found. One research area where this question could be investigated is language acquisition.[37] Green 2000 programmatically explores the idea that first language acquisition proceeds along incremental and largely monotonic extensions of a

[37]To the best of my knowledge, ideas about the psychological reality of sort hierarchies, although alluded to occasionally, are mere speculations that have not been tested in scientific experiments yet. Moreover, the computational results about the description languages of HPSG 94 of Section 3.3 caution against far-reaching assumptions about their psychological reality.

sort hierarchy that is extended as more and more types of linguistic entities are distinguished by the child.[38] From the practical perspective of a grammar writer, sort hierarchies provide the means for a more compact notation of principles and for a useful structuring of the empirical domain in the eyes of the linguist. Even if finite sort hierarchies are superfluous from a mathematical perspective, they can still be an expedient syntactic construct for writing grammars.

The logical language of SRL is designed to describe (sets of) entities. Given an interpretation, essentially, we can say that entities have a certain sort ("sort assignment"); or that a certain attribute is defined on an entity, whose interpretation then leads to another entity; or that one entity equals another entity. To achieve this, SRL provides a set of terms, which are composed of the reserved symbol, ':', and the attributes of a given SRL signature, Σ, and a set of descriptions, which are built from the terms, the species of Σ, and reserved symbols for expressing sort assignment and equality, '$\sim$' and '$\approx$'. The descriptions are closed under negation, conjunction, disjunction, and implication. Other logical connectives can be obtained by combinations of these. I define Σ terms and Σ descriptions simultaneously:

Definition 17 *For each SRL signature $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{F} \rangle$, $\mathcal{T}^\Sigma$ and $\mathcal{D}^\Sigma$ are the smallest sets such that*

$: \in \mathcal{T}^\Sigma$,

for each $\alpha \in \mathcal{A}$ *and each* $\tau \in \mathcal{T}^\Sigma$, $\tau\alpha \in \mathcal{T}^\Sigma$,

for each $\sigma \in \mathcal{S}$, *for each* $\tau \in \mathcal{T}^\Sigma$, $\tau \sim \sigma \in \mathcal{D}^\Sigma$,

for each $\tau_1 \in \mathcal{T}^\Sigma$, *for each* $\tau_2 \in \mathcal{T}^\Sigma$, $\tau_1 \approx \tau_2 \in \mathcal{D}^\Sigma$,

for each $\delta \in \mathcal{D}^\Sigma$, $\neg\delta \in \mathcal{D}^\Sigma$,

for each $\delta_1 \in \mathcal{D}^\Sigma$, *for each* $\delta_2 \in \mathcal{D}^\Sigma$, $[\delta_1 \wedge \delta_2] \in \mathcal{D}^\Sigma$,

for each $\delta_1 \in \mathcal{D}^\Sigma$, *for each* $\delta_2 \in \mathcal{D}^\Sigma$, $[\delta_1 \vee \delta_2] \in \mathcal{D}^\Sigma$, *and*

for each $\delta_1 \in \mathcal{D}^\Sigma$, *for each* $\delta_2 \in \mathcal{D}^\Sigma$, $[\delta_1 \rightarrow \delta_2] \in \mathcal{D}^\Sigma$.

[38]The formal foundations underlying Green's account are, however, not entirely clear to me, as her terminology fluctuates between HPSG 87, HPSG 94 and terms that apparently stem from other sources not directly related to HPSG formalisms. It remains to be seen if her research program can be rigorously rendered in a formalism for HPSG.

For each SRL signature Σ, I call each element of the set $\mathcal{T}^{\Sigma}$ a Σ *term*, and each element of $\mathcal{D}^{\Sigma}$ a Σ *description*.

A Σ term consists of the reserved symbol, ':', followed by a (possibly empty) finite string of Σ attributes. Just as in Chapter 2, I will use the term "paths" to refer to finite strings of attributes. The symbol ':' can be viewed as the single variable of the languages of SRL. When we define denotations for our expressions, it will denote the entities being described. I call Σ descriptions of the form $\tau \sim \sigma$ *sort assignments*; Σ descriptions of the form $\tau_1 \approx \tau_2$ are *path equations*.

Superficially, the Σ descriptions of SRL do not bear any apparent resemblance to the conventional, loosely specified syntax of AVM diagrams that linguists use. King 1989, pp. 106–135, addresses this issue, and provides a translation from the expressions of formally defined languages of attribute value matrices to SRL's descriptions that, under certain conditions, preserves the intuitively intended meaning of the attribute value matrices. For my discussion of the SRL-based formalisms of HPSG 94, it suffices to know that a translation is possible in principle, and to appeal to an informal understanding of how AVM diagrams and Σ descriptions correspond. Once I have presented my extension of SRL in Section 3.1, I will return to the issue of linguistic notation in Section 3.2 and define an AVM syntax for the extended formalism that is modeled after the notational conventions of the HPSG literature.

Before discussing a few examples of descriptions, I want to define the denotation of descriptions in interpretations. To understand the denotation of terms, it might initially be helpful to compare the term interpretation function, T_{I}, of a given Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$, to the iterated transition function, Δ^*, of a given concrete feature structure under Σ with transition function Δ. The denotation of the term consisting of the colon corresponds to the transition from one node to another by interpreting the empty path, and the denotation of a term of the form $:\pi$, where π is a nonempty path, corresponds to the transition from one node to another by interpreting π. The elements of U correspond to the nodes of the concrete feature structure.[39] That means that by interpreting a term at an element, u, of U, we transit

[39] There is, of course, no connectivity condition on interpretations that stipulates that each element of U be reachable from some root element in U. The Σ interpretations of SRL are more general structures than (totally well-typed and sort resolved) concrete feature structures. I will discuss the relationship between concrete feature structures and interpretations of SRL signatures in more detail in Sections 2.2.2.3 and 2.2.2.4.

from u to another element of U. In the special cases of a cyclic transition or a term consisting only of the colon, the element that we reach is identical to u. Since descriptions are built from terms, their interpretation depends directly on the interpretation of terms, and I define the two interpretation functions simultaneously:

Definition 18 *For each SRL signature $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{F} \rangle$, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$, T_I is the total function from $\mathcal{T}^\Sigma$ to the set of partial functions from U to U, and D_I is the total function from $\mathcal{D}^\Sigma$ to the power set of U such that for each $u \in \mathsf{U}$,*

$T_\mathsf{I}(:)(u)$ is defined and $T_\mathsf{I}(:)(u) = u$,

for each $\tau \in \mathcal{T}^\Sigma$, for each $\alpha \in \mathcal{A}$,

$T_\mathsf{I}(\tau\alpha)(u)$ is defined

iff $T_\mathsf{I}(\tau)(u)$ is defined and $\mathsf{A}(\alpha)(T_\mathsf{I}(\tau)(u))$ is defined, and

if $T_\mathsf{I}(\tau\alpha)(u)$ is defined

then $T_\mathsf{I}(\tau\alpha)(u) = \mathsf{A}(\alpha)(T_\mathsf{I}(\tau)(u))$,

for each $\tau \in \mathcal{T}^\Sigma$, for each $\sigma \in \mathcal{S}$,

$$D_\mathsf{I}(\tau \sim \sigma) = \left\{ u \in \mathsf{U} \,\middle|\, \begin{array}{l} T_\mathsf{I}(\tau)(u) \textit{ is defined, and} \\ \mathsf{S}(T_\mathsf{I}(\tau)(u)) = \sigma \end{array} \right\},$$

for each $\tau_1 \in \mathcal{T}^\Sigma$, for each $\tau_2 \in \mathcal{T}^\Sigma$,

$$D_\mathsf{I}(\tau_1 \approx \tau_2) = \left\{ u \in \mathsf{U} \,\middle|\, \begin{array}{l} T_\mathsf{I}(\tau_1)(u) \textit{ is defined,} \\ T_\mathsf{I}(\tau_2)(u) \textit{ is defined, and} \\ T_\mathsf{I}(\tau_1)(u) = T_\mathsf{I}(\tau_2)(u) \end{array} \right\},$$

for each $\delta \in \mathcal{D}^\Sigma$, $D_\mathsf{I}(\neg\delta) = \mathsf{U} \backslash D_\mathsf{I}(\delta)$,

for each $\delta_1 \in \mathcal{D}^\Sigma$, for each $\delta_2 \in \mathcal{D}^\Sigma$, $D_\mathsf{I}([\delta_1 \wedge \delta_2]) = D_\mathsf{I}(\delta_1) \cap D_\mathsf{I}(\delta_2)$,

for each $\delta_1 \in \mathcal{D}^\Sigma$, for each $\delta_2 \in \mathcal{D}^\Sigma$, $D_\mathsf{I}([\delta_1 \vee \delta_2]) = D_\mathsf{I}(\delta_1) \cup D_\mathsf{I}(\delta_2)$, and

for each $\delta_1 \in \mathcal{D}^\Sigma$, for each $\delta_2 \in \mathcal{D}^\Sigma$, $D_\mathsf{I}([\delta_1 \rightarrow \delta_2]) = (\mathsf{U} \backslash D_\mathsf{I}(\delta)) \cup D_\mathsf{I}(\delta_2)$.

For each SRL signature Σ, I call T_{I} the Σ *term interpretation function with respect to* I, and D_{I} the Σ *description interpretation function with respect to* I. As discussed already, a Σ term interpretation function with respect to $\langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$ is a partial function from U to U. More specifically, the colon denotes the identity function, and we obtain the denotation of each term in which a nonempty string of attributes succeeds the colon by functional composition of the denotation of the colon and the attribute interpretation of each of the attribute symbols in the term, with the interpretation of the last attribute coming first.

The denotation of Σ descriptions follows naturally from the denotation of Σ terms: Each Σ description $\tau \sim \sigma$ denotes the subset of entities in U on whose members the Σ term interpretation function (with respect to $\langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$) of τ is defined and yields an entity of species σ. Each Σ description of the form $\tau_1 \approx \tau_2$ denotes the set of those entities in U on which the Σ term interpretation function (with respect to $\langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$) of both τ_1 and τ_2 is defined and yields the same entity in both cases. Negation, conjunction, disjunction and implication of Σ descriptions are interpreted classically as set complement of the denotation of the description, as the intersection and union of the denotations of the two descriptions of the complex description, and as the union of the set complement of the denotation of the first description in the implication with the denotation of the second description, respectively.

A signature expresses how we conceive of the world that we want to talk about as being structured. Assume the particular SRL signature given in Figure 2.4. For the purposes of the following examples, I will take it as fixed and leave it implicit when talking about terms and descriptions. According to the signature, the entities in our universe are of sort *man*, *woman*, *bmw* and *vw*; and there are no other entities. A *bmw* and a *vw* have a DRIVER and an OWNER, who are either a *woman* or a *man*.[40] But neither a *vw* nor a *bmw* have the property of liking somebody or something best. Each *man* and each *woman*, on the other hand, likes some entity of the universe best, and the entity that they like best can be of any sort. Neither a *woman* nor a *man* has a DRIVER or an OWNER. Let us now construct one of the many possible interpretations of this SRL signature.

Suppose the following scenario involving two cars on the parking lot in front of the Seminar für Sprachwissenschaft in Tübingen: The first car, which

[40]Note that these appropriateness conditions exclude, among other conceivable scenarios, multiple owners of a single car.

belongs to Anke, is a Volkswagen. Anke is in the driver's seat. We know that Anke is married, and she likes her husband best. Her husband, of course, likes her best. The second car is a Volkswagen, too. It is owned by Kordula, but since she is on vacation, Detmar is driving it. Kordula and Detmar like each other best. We can capture some facts of this little scenario in an interpretation of our signature. The two cars, Anke, her husband, Detmar and Kordula are the entities in the interpretation. They are assigned the obvious species given the intuitive meaning of the species symbols. The attributes receive a denotation that obeys appropriateness and gives the attributes their natural denotations with respect to the above scenario. Figure 2.5 shows this interpretation, which I will call $\mathsf{I}_{2.5}$.

Figure 2.5: An interpretation of the SRL signature of Figure 2.4

Let $\mathsf{I}_{2.5} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$, with:

U = {Anke, Anke's husband, Detmar, Kordula, first car, second car}

S(Anke) = *woman*,
S(Kordula) = *woman*,
S(Anke's husband) = *man*,
S(Detmar) = *man*,
S(first car) = *vw*, and
S(second car) = *vw*

A(LIKES-BEST)(Anke) = Anke's husband,
A(LIKES-BEST)(Anke's husband) = Anke,
A(LIKES-BEST)(Detmar) = Kordula,
A(LIKES-BEST)(Kordula) = Detmar,
A(OWNER)(first car) = Kordula,
A(OWNER)(second car) = Anke,
A(DRIVER)(first car) = Detmar,
A(DRIVER)(second car) = Anke,
A(LIKES-BEST) is undefined on the first car and on the second car,
A(OWNER) is undefined on Anke, Anke's husband, Detmar and Kordula, and
A(DRIVER) is likewise undefined on Anke, Anke's husband, Detmar and Kordula

We can now inspect the denotation of descriptions in the interpretation $\mathsf{I}_{2.5}$. In (15a) we see that the set of women in $\mathsf{I}_{2.5}$ consists of Anke and Kordula. There is no BMW in $\mathsf{I}_{2.5}$ (15b). Finally, only the second car is a Volkswagen whose owner is also its driver (15c), because Anke drives her own car, whereas Detmar drives Kordula's car; and there is no other car in $\mathsf{I}_{2.5}$.

(15) a. $D_{\mathsf{I}_{2.5}}(: \sim woman) = \{\text{Anke, Kordula}\}$

b. $D_{\mathsf{I}_{2.5}}(: \sim bmw) = \emptyset$

c. $D_{\mathsf{I}_{2.5}}([: \sim vw \wedge :\text{OWNER} \approx :\text{DRIVER}]) = \{\text{second car}\}$

Note that in a different interpretation of the same signature, the descriptions in (15) might denote different sets of entities. As soon as there are BMWs in the interpretation, '$: \sim bmw$' denotes that set of cars. Moreover, it is easy to see how the symbols of the sort hierarchy of the 87 signature of Figure 2.2 that are not present in our corresponding SRL signature can be understood as an abbreviation of disjunctive descriptions. For example, '$: \sim person$' can be defined as a metasyntactic notation for '$[: \sim man \vee : \sim woman]$', which denotes the set comprising Anke, Anke's husband, Detmar and Kordula. In other words, a sort assignment with a sort σ that is not a species is interpreted as the disjunction of the sort assignments with the species that σ subsumes in the envisaged sort hierarchy of an 87 signature.

An HPSG 94 grammar is a pair consisting of a signature and a set of expressions of the description language, the principles of grammar. Definition 19 captures this conception of a grammar with the notion of an SRL grammar:

Definition 19 Γ *is an **SRL grammar** iff*

Γ *is a pair* $\langle \Sigma, \theta \rangle$,

Σ *is an SRL signature, and*

$\theta \subseteq \mathcal{D}^{\Sigma}$.

Since an SRL grammar always includes a fixed signature, Σ, I will simply talk about descriptions of SRL grammars below instead of Σ descriptions of SRL grammars.

The purpose of a grammar is to describe a natural language. In HPSG 94, the most important means to delineate a natural language is clearly the set of principles. It is, therefore, a prerequisite for determining what an SRL grammar means to define what a set of descriptions means. The denotation of a set of descriptions, θ, in a given interpretation, I, is the set of all entities in I of which each description in θ is true:

Definition 20 *For each SRL signature* Σ*, for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$*,* Θ_{I} *is the total function from the power set of* $\mathcal{D}^{\Sigma}$ *to the power set of* U *such that for each* $\theta \subseteq \mathcal{D}^{\Sigma}$*,*

$$\Theta_{\mathsf{I}}(\theta) = \left\{ u \in \mathsf{U} \,\middle|\, \begin{array}{l} \textit{for each } \delta \in \theta, \\ u \in D_{\mathsf{I}}(\delta) \end{array} \right\}.$$

I call Θ_{I} the *theory denotation function with respect to* I. For example, the theory $\theta_1 = \{[:\sim bmw \rightarrow \neg :\approx:],\ [:\sim vw \rightarrow :\text{OWNER} \sim woman]\}$ describes every entity in the interpretation $\mathsf{I}_{2.5}$, i.e., $\Theta_{\mathsf{I}_{2.5}}(\theta_1) = \{$Anke, Anke's husband, Detmar, Kordula, first car, second car$\}$. This is so, because each of the descriptions of θ_1 describes every entity in $\mathsf{I}_{2.5}$: The first, '$[:\sim bmw \rightarrow \neg :\approx:]$,' describes all entities that are not BMWs or not identical with themselves. This is true of all entities that are not BMWs and false of all BMWs. Since $\mathsf{I}_{2.5}$ does not contain any entity of sort *bmw*, the description is true of all entities in $\mathsf{I}_{2.5}$. The second description in θ_1 describes entities that are not VWs or whose owner is a woman. That is true of all entities that are not VWs and of all VWs whose owner is a woman. In other words, it is true of all entities in $\mathsf{I}_{2.5}$, because Anke, Anke's husband, Detmar and Kordula are men and women, and the two VWs are owned by Anke and Kordula, respectively.

As the example shows, the theory denotation of a finite set of descriptions equals the description denotation of the conjunction of the descriptions that the set contains. A finite theory can thus be expressed by the conjunction of the descriptions in it, and for the finite case, theories and theory denotation functions are not strictly necessary. However, the inductive definition of the languages of SRL does not permit infinite conjunctions. The theory denotation functions with respect to an interpretation thus add the possibility of interpreting infinite theories.

Denotations of theories in interpretations do not yet determine the meaning of a grammar by themselves. The theory denotation function only tells us of which entities in an interpretation a set of descriptions is true. This property can, however, be exploited in a first approximation of the intended interpretations of a grammar. Clearly, the linguist is only interested in those interpretations of a grammar in which every description is true of every entity: Every description of the grammar describes every entity in the interpretations, or, equivalently, no description is false of any entity in the intended interpretations. Interpretations that have that property with respect to a given grammar are called *models* of the grammar:

Definition 21 *For each SRL grammar* $\Gamma = \langle \Sigma, \theta \rangle$, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$,

I *is a* Γ ***model*** *iff* $\Theta_{\mathsf{I}}(\theta) = \mathsf{U}$.

The theory, θ, of an SRL grammar, Γ, denotes the entire universe of entities, U, in each Γ model, $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$. Since every entity in the interpretation $\mathsf{I}_{2.5}$ is in the denotation of each element of θ_1, $\mathsf{I}_{2.5}$ is a Γ model of the grammar Γ that consists of the signature of Figure 2.4 and θ_1. Intuitively speaking, no entity in $\mathsf{I}_{2.5}$ violates the conditions of not being a BMW and of being owned by a woman if it is a Volkswagen. Not every theory has a nonempty model. Consider the theory, θ_2, that contains the single description '$: \sim vw$.' A Γ model with the theory θ_2 may only contain entities of sort *vw*, because '$: \sim vw$' is only true of VWs and false of all other entities. But by virtue of the signature, each Volkswagen must have an owner who is either a woman or a man. Therefore, no Γ model with the theory θ_2 can contain a Volkswagen; each Γ model with the theory θ_2 must have an empty universe.

Considering the notion of a model of a grammar from a linguistic perspective, it restricts the candidates for the meaning of a grammar to those interpretations whose entities do not violate any principles. For example, assuming that a grammar contains a correct specification of subject verb agreement in English, the sentence *John love Mary* cannot be in any model of the grammar, because it violates subject verb agreement. The sentence contains at least one entity that is not described by at least one description in the theory of the grammar. The models of SRL grammars will be the starting point for all three explications of the meaning of HPSG 94 grammars in the following three sections.

My introduction to SRL deliberately omitted the logic of SRL and results pertaining to it, because the logic of SRL is of no immediate relevance for the characterization of natural languages in the SRL formalism. For completeness, I mention some of the logical aspects of SRL here.[41] For each SRL signature Σ with a finite set of species, King 1989 provides a Hilbert and Ackermann style calculus, and shows that the inference relation that the calculus determines is sound and complete with respect to entailment, i.e., for each set of Σ descriptions θ and for each Σ description δ, θ infers δ if and only if θ entails δ. Kepser 1994 then proceeds to show that for each SRL signature

[41]See King 1999, pp. 328–329, and the literature cited there for a more comprehensive overview over the logical results about SRL.

with a finite set of species and a recursive set of attributes, the satisfiability problem is decidable: There is an effective algorithm that decides for each SRL signature of the kind indicated, and for each description δ generated by that signature, whether there is an interpretation of the signature in which δ has a nonempty denotation. In Section 3.3, I will return to the question of the significance of these and related results to a theoretical linguist who uses formal languages of the kind that HPSG 94 envisages in order to describe natural languages.

2.2.2.2 Exhaustive Models

In a substantial reformulation of an earlier attempt to characterize the meaning of HPSG 94 grammars in King 1994, King 1999 investigates the question of when an SRL grammar is true of a natural language. King (1999) formulates three necessary conditions for an SRL grammar to be true of a natural language. These conditions are met if a natural language belongs to a certain class of models of a given SRL grammar. King calls that class of models the class of *exhaustive models* of a grammar. The meaning of an SRL grammar is thus determined as delineating the class of its exhaustive models, and the grammar is true of a language, $\mathcal{L}$, if $\mathcal{L}$ is an exhaustive model of the grammar.

In this section, I present the motivation behind King's notion of exhaustive models, and their definition. As the introductory, short description in the preceding paragraph of King's approach reveals, the perspective of King 1999 is realistic in the sense that the task of SRL grammars is to directly characterize natural languages without the intervention of some modeling mathematical structure.[42] The natural languages themselves are the intended models of grammars. The realistic approach to the meaning of scientific theories is the most important feature that distinguishes King's explanation of the meaning of HPSG 94 grammars from the explanations of Pollard and Sag 1994 and of Pollard 1999, which choose a representational approach. Despite their fundamentally different assumptions about the ontological status of the structures that grammars characterize, the three formalisms are very similar and mathematically closely related. Understanding the technical side to the

[42]The term *realistic* is meant in a pre-theoretical sense here; it is not meant to be a technical characterization of King's philosophy of science. In particular, King does not explicitly call himself a realist. I use the term to refer to his view because it emphasizes his predominant interest in describing linguistic behavior directly, and stresses the contrast to the representational view of Pollard and Sag.

idea of exhaustive models will thus also illuminate many crucial aspects of the other two approaches, and exhaustive models will become the mathematical link between them. For expository reasons, I adopt the realistic terminology of King for the rest of the present section.

At first it might be surprising that the notion of the models of an SRL grammar defined in Definition 21 of the previous section is insufficient to determine the meaning of a grammar. The basic problem is, however, very easy to see: A non-contradictory grammar, i.e., a grammar that has a nonempty model, does not have a unique model; and the models of an interesting grammar differ in linguistically significant ways. A trivial example of an obviously unintended model is the model with the empty universe. A less trivial example of an unintended model is a model of Pollard and Sag's (1994) grammar which contains exactly one token of the sentence *Kim likes bagels* and nothing else. Presupposing that the grammar contains the necessary lexical entries and that there are no hidden mistakes in the grammar of Pollard and Sag, every entity in the sentence *Kim likes bagels* is described by every principle in the theory of their grammar. But the grammar is supposed to license many more sentences that do not occur in this particular small model. For example, Pollard and Sag's fragment of English also contains the sentence *We believe there to have been many solutions to that problem*, and this sentence must therefore also be in the intended model of their grammar. Intuitively, every model that is missing an expression of English that is licensed by a given grammar of English is in a relevant sense too small to be the intended model of that grammar. The intended model should contain instances of all expressions of English that are permitted by the theory.

What is missing so far is a model theory that tells us unambiguously which models are the intended models of an SRL grammar. If we knew which models are the "right" models of a grammar, then these models would also provide the intended meaning of the grammar. In the terminology of generative grammar, the question of the right models of a given grammar is connected to the question which linguistic structures are *generated* by the grammar. Adapting the traditional terminology of generative grammar to the constraint-based framework of HPSG 94, we can slightly rephrase the question: In a constraint-based framework, we ask which linguistic structures are *licensed* by a grammar. This question is of utmost importance to generative grammarians. Given a natural language or a certain fragment thereof, linguists want to check whether their grammar generates exactly the sentences of the language or of the language fragment.[43] If the grammar generates sentences that are not in the language, generative grammarians say that the grammar *overgenerates*. If the grammar does not generate sentences that are in the language, generative grammarians say that the grammar *undergenerates*. Correspondingly, we can say that an HPSG 94 grammar *overlicenses* if it licenses linguistic structures that are not in the language; and it *underlicenses* if it does not license linguistic structures that are in the language. With an arbitrary model of a grammar, we cannot reliably determine whether the grammar overlicenses or underlicenses the target language. If the chosen model contains all the linguistic structures that the grammar is supposed to license, there might still be some "bigger" model of the grammar that contains structures that are judged ungrammatical, and we would then want to say that the grammar overlicenses. On the other hand, the existence of a model that is "too small" does not mean that a grammar underlicenses. This is so because there might be another, "bigger" model of the grammar that contains the structures that are missing in the smaller model. The theory of the intended models of a grammar is supposed to give us models that tell us if a grammar overlicenses or underlicenses. The exhaustive models of a grammar are these intended models. If an exhaustive model of a grammar contains structures that are not in the language, the grammar overlicenses. If an exhaustive model of a grammar does not contain structures that are in the language, the grammar underlicenses.[44] Of course, it is conceivable that a grammar simultaneously overlicenses and underlicenses a language.

The situation can be illustrated with our little car scenario of interpretation $\mathsf{I}_{2.5}$ from the previous section. A linguist who wants to write a grammar of (a fragment of) a natural language faces a given intended model, the natural language, and the task of the linguist is to write a grammar that neither overlicenses nor underlicenses the language. As a simple, non-linguistic illustration of the problem of the linguist, consider the little scenario $\mathsf{I}_{2.5}$. In that scenario, we find the following observationally given facts: Kordula is the owner of the first car, Detmar drives the first car, and Kordula and Detmar

[43] In this context, a *sentence* means a phonological string together with (at least) a syntactic and a semantic structure. Phonological strings together with unintended syntactic or semantic structures are not sentences of the language in that sense, even if the phonological string in question belongs to a sentence of the language.

[44] For the sake of giving the reader a first, simple intuition about the idea of exhaustive models, this characterization of overlicensing and underlicensing of a grammar relative to an exhaustive model is oversimplifying a bit. It will be refined presently.

like each other best. Anke is the owner and the driver of the second car, and she and her husband like each other best. Kordula and Anke are women, and Detmar and Anke's husband are men. Recall that $\mathsf{I}_{2.5}$ is an interpretation of the SRL signature of Figure 2.4. I call that signature $\Sigma_{2.4}$. The sort symbols of $\Sigma_{2.4}$ allow us to talk about entities of sort *man, woman, vw* and *bmw*, and their attributes are DRIVER, OWNER and LIKES-BEST. The appropriateness function restricts the attributes to entities where they intuitively make sense. Note that with this signature, we cannot talk about particular men and women such as Anke and Detmar, and their cars. We can only talk about men, women, Volkswagens and BMWs, and three kinds of relationships in which they stand to each other. With our grammar, we thus necessarily abstract away from specific entities, and focus on the configurations in which entities of a certain kind stand. Informally, the interpretation $\mathsf{I}_{2.5}$ contains two kinds of configurations. In the first configuration, a Volkswagen is owned by a woman who likes a man best that drives her car and likes the woman best. In the second configuration, a Volkswagen is owned and driven by a woman who likes a man best who likes the woman best. A grammar whose intended model is $\mathsf{I}_{2.5}$ in the sense that the grammar neither overlicenses nor underlicenses $\mathsf{I}_{2.5}$ should thus license models containing configurations of cars, women and men that are isomorphic to the two configurations in $\mathsf{I}_{2.5}$.

As a first attempt at a grammar of $\mathsf{I}_{2.5}$, consider the $\Sigma_{2.4}$ theory $\theta_1 = \{[:\sim \mathit{bmw} \rightarrow \neg :\approx:], [:\sim \mathit{vw} \rightarrow :\textsc{owner} \sim \mathit{woman}]\}$ of Section 2.2.2.1. Given an interpretation, θ_1 is true of all entities that are not a BMW, of all Volkswagens that belong to a woman, and of all entities that are neither a Volkswagen nor a BMW. In Section 2.2.2.1, we already saw that $\mathsf{I}_{2.5}$ is a model of the grammar $\langle \Sigma_{2.4}, \theta_1 \rangle$, because $\mathsf{I}_{2.5}$ does not contain a BMW, the first Volkswagen belongs to Kordula, the second Volkswagen belongs to Anke, and there are no other cars. However, there are configurations in other models of $\langle \Sigma_{2.4}, \theta_1 \rangle$ that are not isomorphic to the ones in $\mathsf{I}_{2.5}$. For example, $\mathsf{I}_{2.5}$ does not contain the legitimate configuration in which a Volkswagen is owned by a woman who likes a man best who likes her best, but the driver of the Volkswagen is a second man who likes a second woman best who likes the second man best. Figure 2.6 shows a model of $\langle \Sigma_{2.4}, \theta_1 \rangle$ that contains exactly such a scenario: The third car is a Volkswagen that is owned by Anne, while Mike drives it. Anne and Manfred and Mike and Maria like each other best.

I conclude that $\langle \Sigma_{2.4}, \theta_1 \rangle$ overlicenses $\mathsf{I}_{2.5}$, because there are models of $\langle \Sigma_{2.4}, \theta_1 \rangle$ that contain configurations of entities that are not isomorphic to any configurations in the target model, $\mathsf{I}_{2.5}$. In fact, it is easy to see that $\langle \Sigma_{2.4}, \theta_1 \rangle$

Figure 2.6: A second model of $\langle \Sigma_{2.4}, \theta_1 \rangle$

Let $\mathsf{I}_{2.6} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$, with:

U = {Anne, Manfred, Maria, Mike, third car}

S(Anne) = *woman*,
S(Maria) = *woman*,
S(Manfred) = *man*,
S(Mike) = *man*,
S(third car) = *vw*, and

A(LIKES-BEST)(Anne) = Manfred,
A(LIKES-BEST)(Manfred) = Anne,
A(LIKES-BEST)(Maria) = Mike,
A(LIKES-BEST)(Mike) = Maria,
A(OWNER)(third car) = Anne,
A(DRIVER)(third car) = Mike,
A(LIKES-BEST) is undefined on the third car,
A(OWNER) is undefined on Anne, Manfred, Maria and Mike, and
A(DRIVER) is likewise undefined on Anne, Manfred, Maria and Mike

licenses infinitely many pairwise non-isomorphic configurations of entities that are not isomorphic to a configuration of entities in $\mathsf{I}_{2.5}$. The exhaustive models of $\langle \Sigma_{2.4}, \theta_1 \rangle$ must contain infinitely many different configurations, because the theory θ_1 is very unrestrictive. For example, imagine a scenario that only contains a man who likes another man best who likes himself best. But there is also a scenario with three men in which the first likes the second and the second likes the third best; and the third likes himself best. This series of scenarios can be continued with four men, with five men, and so on *ad infinitum*, and they are all models of $\langle \Sigma_{2.4}, \theta_1 \rangle$. Therefore, they must all be in a model of $\langle \Sigma_{2.4}, \theta_1 \rangle$ that is exhaustive. Of course, there are infinitely many other possible scenarios involving men, women, and Volkswagens that are possible models of $\langle \Sigma_{2.4}, \theta_1 \rangle$, which I will not consider here.

Clearly, we need a stronger, more restrictive theory about the configurations of entities in $\mathsf{I}_{2.5}$ in order to obtain a grammar of which $\mathsf{I}_{2.5}$ is an exhaustive model. The theory θ_2 is a new attempt at constructing a grammar of $\mathsf{I}_{2.5}$:

(16) $\theta_2 =$

$$\left\{\begin{array}{l}[:\sim \textit{woman} \leftrightarrow [:\text{LIKES-BEST} \sim \textit{man} \wedge :\text{LIKES-BEST LIKES-BEST} \approx:]]\,,\\ [:\sim \textit{man} \leftrightarrow [:\text{LIKES-BEST} \sim \textit{woman} \wedge :\text{LIKES-BEST LIKES-BEST} \approx:]]\,,\\ [:\sim \textit{bmw} \rightarrow \neg :\approx:]\,,\\ [:\sim \textit{vw} \rightarrow [:\text{OWNER} \sim \textit{woman} \wedge :\text{OWNER} \approx: \text{DRIVER}]]\end{array}\right\}$$

The theory θ_2 enforces the relationship between men and women that we observe in $\mathsf{I}_{2.5}$ in all models of the grammar $\langle \Sigma_{2.4}, \theta_2 \rangle$. Every woman likes a man best who likes her best, and every man likes a woman best who likes him best. No BMWs are admitted. The last description in θ_2 excludes the $\Sigma_{2.4}$ interpretation of Figure 2.6 from being a $\langle \Sigma_{2.4}, \theta_2 \rangle$ model, because it is only true of those Volkswagens whose owner is also the driver, and the owner of the third car is Anne, whereas Mike is the driver. So far, it looks as if $\langle \Sigma_{2.4}, \theta_2 \rangle$ were the grammar that we are after, because it does not overlicense $\mathsf{I}_{2.5}$. Unfortunately, the new grammar, $\langle \Sigma_{2.4}, \theta_2 \rangle$, underlicenses $\mathsf{I}_{2.5}$. It requires that the owner of each Volkswagen be also its driver, but in $\mathsf{I}_{2.5}$, Detmar is the driver of Kordula's car. In other words, $\mathsf{I}_{2.5}$ is not even a model of the grammar $\langle \Sigma_{2.4}, \theta_2 \rangle$, because the last description of θ_2 does not describe Kordula's car.

As a grammar of $\mathsf{I}_{2.5}$, $\langle \Sigma_{2.4}, \theta_1 \rangle$ is too unrestrictive, whereas $\langle \Sigma_{2.4}, \theta_2 \rangle$ is too restrictive. However, based on the experience with these two grammars, it is not difficult to find a grammar that imposes the right conditions. We must be careful not to strengthen θ_1 too much, so $\mathsf{I}_{2.5}$ remains a model of the grammar. At the other end of the spectrum, we must relax θ_2 sufficiently, so every description in the new grammar is still true of every entity in $\mathsf{I}_{2.5}$. Consider the theory θ_3 in (17):

(17) $\theta_3 =$

$$\left\{\begin{array}{l}[:\sim \textit{woman} \leftrightarrow [:\text{LIKES-BEST} \sim \textit{man} \wedge :\text{LIKES-BEST LIKES-BEST} \approx:]]\,,\\ [:\sim \textit{man} \leftrightarrow [:\text{LIKES-BEST} \sim \textit{woman} \wedge :\text{LIKES-BEST LIKES-BEST} \approx:]]\,,\\ [:\sim \textit{bmw} \rightarrow \neg :\approx:]\,,\\ \left[:\sim \textit{vw} \rightarrow \left[\begin{array}{l}:\text{OWNER} \sim \textit{woman} \wedge\\ [:\text{OWNER} \approx: \text{DRIVER} \vee :\text{DRIVER} \approx: \text{OWNER LIKES-BEST}]\end{array}\right]\right]\end{array}\right\}$$

Let Γ_3 be $\langle \Sigma_{2.4}, \theta_3 \rangle$. In addition to the descriptions of θ_1, θ_3 contains descriptions that are true of all entities in a $\Sigma_{2.4}$ interpretation if every woman and every man likes a person of the opposite sex best, and the other person returns their feelings. Moreover, every Volkswagen in a model of Γ_3 is owned by a woman who is either also the driver, or the man she likes best is the driver. This is where θ_3 is not as restrictive as θ_2. BMWs remain excluded from the models. It is straightforward to check that $\mathsf{I}_{2.5}$ is an exhaustive model of Γ_3, because it contains all kinds of configurations of entities that can occur in models of Γ_3. In particular, the configuration under the third car in the model $\mathsf{I}_{2.6}$ has no isomorphically configured counterpart in an exhaustive model of Γ_3, because the last description in θ_3 does not describe the third car: Neither is its driver the woman who owns it (Anne), nor the man she likes best (Manfred).

It is now time to define a terminology that allows me to be more precise about what I mean by configurations of entities and isomorphically configured entities in an interpretation. To simplify some formulations, I will from now on say that an entity, u, is an element of a Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$ if $u \in \mathsf{U}$, and I write $u \in \mathsf{I}$ to indicate that u is a member of the universe of I. The first step towards defining a notion of configurations of entities in an interpretation is the definition of componenthood. An entity, u', of a Σ interpretation I is a *component* of an entity, u, of I, if there is a Σ term τ such that the interpretation of τ at u is defined and yields u'. Informally, there is a path (preceded by the colon) whose interpretation leads from u to u', i.e., u' is in the subalgebra of I generated by u. DEFINITION 22 captures the notion of componenthood:

Definition 22 *For each SRL signature* $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{F} \rangle$, *and for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$, Co_I *is the total function from* U *to the power set of* U *such that for each* $u \in \mathsf{U}$,

$$\mathsf{Co}_\mathsf{I}(u) = \left\{ u' \in \mathsf{U} \,\middle|\, \begin{array}{l} \textit{for some } \tau \in \mathcal{T}^\Sigma \\ T_\mathsf{I}(\tau)(u) \textit{ is defined, and } T_\mathsf{I}(\tau)(u) = u' \end{array} \right\}.$$

$\mathsf{Co}_\mathsf{I}(u)$ is the *set of components of* u *in* I. Conversely, I say that an entity, u, of a Σ interpretation I is a *common ancestor* of two entities u_1 and u_2 in I if u_1 and u_2 are components of u in I.

As configurations of entities I want to single out parts of Σ interpretations that comprise exactly one freely chosen entity with the set of its components and the relationships that exist between all these entities by virtue of the attribute interpretation function, but that do not contain any other, unrelated entities. For example, I want to have a simple terminology that allows me to talk about the second car and everything that we can see about the second car in the interpretation $\mathsf{I}_{2.5}$ of Figure 2.5: that Anke owns it, that Anke drives it, and that Anke likes her husband best, who in turn likes her best. What I am after is clearly a subinterpretation of the interpretation $\mathsf{I}_{2.5}$. The subinterpretation comprises an entity, the second car, of $\mathsf{I}_{2.5}$, all its components in $\mathsf{I}_{2.5}$, and the sort assignment and the attribute interpretation function restricted to these entities. DEFINITION 23 formalizes subinterpretations of that kind of a given interpretation. I_u is the subalgebra generated by u in I:

Definition 23 *For each SRL signature $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{F} \rangle$, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$, for each $u \in \mathsf{I}$,*

$\mathsf{I}_u = \langle \mathsf{U}_u, \mathsf{S}_u, \mathsf{A}_u \rangle$ *is the Σ* ***interpretation under*** *u* ***in*** *I iff*

$$\begin{aligned}
&\mathsf{U}_u = \mathsf{Co}_{\mathsf{I}}(u),\\
&\mathsf{S}_u = \mathsf{S} \cap (\mathsf{U}_u \times \mathcal{S}),\\
&\mathsf{A}_u = \mathsf{A} \cap (\mathcal{A} \times \{\mathsf{U}_u \times \mathsf{U}_u\}), \textit{ and}\\
&\mathsf{I}_u = \langle \mathsf{U}_u, \mathsf{S}_u, \mathsf{A}_u \rangle.
\end{aligned}$$

By DEFINITION 23, the interpretation under Kordula in $\mathsf{I}_{2.5}$ is the interpretation $\mathsf{I}_{\text{Kordula}} = \langle \mathsf{U}_{\text{Kordula}}, \mathsf{S}_{\text{Kordula}}, \mathsf{A}_{\text{Kordula}} \rangle$, with
$\mathsf{U}_{\text{Kordula}} = \{\text{Kordula, Detmar}\}$,
$\mathsf{S}_{\text{Kordula}} = \{\langle \text{Kordula}, \textit{woman} \rangle, \langle \text{Detmar}, \textit{man} \rangle\}$, and
$\mathsf{A}_{\text{Kordula}} = \{\langle \text{LIKES-BEST}, \{\langle \text{Kordula}, \text{Detmar} \rangle, \langle \text{Detmar}, \text{Kordula} \rangle\}\}$,
because the only component of Kordula (besides Kordula herself) is Detmar, Kordula is a woman, Detmar is a man, and they like each other best. It is convenient to have a name to refer to an arbitrary entity in an interpretation and the subinterpretation under that entity:

Definition 24 *For each SRL signature $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{F} \rangle$, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$, for each $u \in \mathsf{I}$,*

$\langle u, \mathsf{I}_u \rangle$ *is the* ***configuration of entities under*** *u* ***in*** *I iff*

I_u *is the Σ interpretation under u in I.*

I will omit the reference to I if an interpretation is fixed. In that case I will simply talk about the configuration (of entities) under some entity, u. Assume the interpretation $\mathsf{I}_{2.5}$. Then $\langle \text{Kordula}, \mathsf{I}_{\text{Kordula}} \rangle$ is the configuration (of entities) under Kordula.

Crucially, we want to be able to compare configurations under different entities in possibly different interpretations of the same SRL signature. This is necessary in order to determine if one interpretation contains a configuration under an entity that is in fact isomorphic to a configuration under a different entity in another interpretation. Alternatively, we might also want to say that, in one and the same interpretation, all components of two distinct entities are equally configured. For that purpose, I define what it means for a pair of an entity and an interpretation of which the entity is a member to be congruent to a second pair of an entity and an interpretation. Two configurations under different entities in $\mathsf{I}_{2.5}$ that are congruent are $\langle \text{Kordula}, \mathsf{I}_{\text{Kordula}} \rangle$ and $\langle \text{Anke}, \mathsf{I}_{\text{Anke}} \rangle$. But $\langle \text{Anke}, \mathsf{I}_{\text{Anke}} \rangle$ and $\langle \text{Detmar}, \mathsf{I}_{\text{Detmar}} \rangle$ are not congruent, although the configuration under Anke and the configuration under Detmar each have two components on which LIKES-BEST is defined; and in both cases LIKES-BEST, when interpreted on one component, denotes the other component. The reason that the two configurations are nevertheless not congruent is that Anke is a woman and Detmar is a man. DEFINITION 25 makes the notion of an SRL congruence that I have used in my examples explicit:

Definition 25 *For each SRL signature* Σ*, for each* Σ *interpretation* $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1 \rangle$*, for each* $u_1 \in \mathsf{I}_1$*, for each* Σ *interpretation* $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2 \rangle$*, for each* $u_2 \in \mathsf{I}_2$*,*

f is an ***SRL congruence from*** $\langle u_1, \mathsf{I}_1 \rangle$ ***to*** $\langle u_2, \mathsf{I}_2 \rangle$ ***in*** Σ

iff f is a bijection from $\mathsf{Co}_{\mathsf{I}_1}(u_1)$ *to* $\mathsf{Co}_{\mathsf{I}_2}(u_2)$*,*

for each $u \in \mathsf{Co}_{\mathsf{I}_1}(u_1)$*,* $\mathsf{S}_1(u) = \mathsf{S}_2(f(u))$*,*

for each $\alpha \in \mathcal{A}$*, for each* $u \in \mathsf{Co}_{\mathsf{I}_1}(u_1)$*,*

$\mathsf{A}_1(\alpha)(u)$ *is defined iff* $\mathsf{A}_2(\alpha)(f(u))$ *is defined, and*
if $\mathsf{A}_1(\alpha)(u)$ *is defined then* $f(\mathsf{A}_1(\alpha)(u)) = \mathsf{A}_2(\alpha)(f(u))$*.*

In the definition of SRL congruences, the interpretations in the two pairs of an entity and an interpretation need not be interpretations under the respective entities in the pair: u_1 is an element of I_1, but not all elements of I_1 are necessarily components of u_1. However, since I_1 by definition obeys the appropriateness conditions imposed by its SRL signature, it is ensured that the interpretation under u_1 is a subinterpretation of I_1. Each interpretation, I, contains the interpretation under each u that is an element of I as a subinterpretation. In the special case in which I_1 only contains entities that are components of u_1 and I_2 only contains entities that are components of u_2, the existence of an SRL congruence from $\langle u_1, \mathsf{I}_1\rangle$ to $\langle u_2, \mathsf{I}_2\rangle$ implies that I_1 and I_2 are isomorphic. Informally, I say that the configurations under u_1 and u_2 are isomorphic, or the components of u_1 and u_2 are isomorphically configured.

A first entity in a first Σ interpretation and a second entity in a second Σ interpretation are *SRL congruent in* Σ exactly if there is a species and attribute preserving bijection from the components of the first entity in the first interpretation to the components of the second entity in the second interpretation, and the bijection maps the first entity to the second entity:

Definition 26 *For each SRL signature* Σ*, for each* Σ *interpretation* $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1\rangle$*, for each* $u_1 \in \mathsf{I}_1$*, for each* Σ *interpretation* $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2\rangle$*, for each* $u_2 \in \mathsf{I}_2$*,*

$\langle u_1, \mathsf{I}_1\rangle$ *and* $\langle u_2, \mathsf{I}_2\rangle$ *are* ***SRL congruent in*** Σ
iff for some f*,* f *is an SRL congruence from* $\langle u_1, \mathsf{I}_1\rangle$ *to* $\langle u_2, \mathsf{I}_2\rangle$ *in* Σ*.*

The terminology of Definition 22–26 allows us to be precise about an important aspect of King's idea of using exhaustive models to explicate the meaning of HPSG 94 grammars that I have already mentioned in the discussion of the car scenario of Figure 2.5. This aspect is intimately tied to King's realistic approach to scientific theories, which takes the natural language itself to be the target model of the grammar instead of some mathematical representation of the language. When I formulated the grammar of $\mathsf{I}_{2.5}$, I did not say that the envisioned grammar should have $\mathsf{I}_{2.5}$ as its unique exhaustive model. What I said was that the grammar should license configurations of entities that were configured isomorphically to the configurations involving Anke, Kordula, Detmar, Anke's husband, the first car, and the second car. To reformulate this in our new terminology, it is not required that every configuration under some entity in every model of the grammar be in the intended model of the grammar; it is only required that the intended model contain at least one configuration under an entity *that is SRL congruent* to any such configuration. That means that not every possible model is part of the intended model. In the example above, the fact that it is only required that a configuration under an entity that is SRL congruent to the configuration under the third car of Figure 2.6 must be part of an exhaustive model of the SRL grammar $\langle \Sigma_{2.4}, \theta_1\rangle$ means that it is not necessary that the concrete scenario involving the third car, Anne, Manfred, Maria, and Mike is part of the intended model. What must be part of an exhaustive model of $\langle \Sigma_{2.4}, \theta_1 \rangle$ is at least one scenario that looks just like the one in Figure 2.6: There must be a Volkswagen that is owned by a woman and driven by a man, and both people like different persons of the opposite sex best. Crucially, the scenario might involve different entities from the ones in $\mathsf{I}_{2.6}$, namely a different Volkswagen and different people.

Similarly, linguistic grammars have a class of exhaustive models which differ from each other with respect to the number of SRL congruent configurations under entities in them, and with respect to the nature of the objects in the universe of the models. The significance of this will become clear when looking at the characterization of King, 1999, p. 343, of the content of an SRL grammar of a natural language. The idea is to formulate "a strong and well-defined necessary condition for an HPSG grammar to be true of a natural language" (King, 1999, p 336): An SRL grammar $\langle \Sigma, \theta \rangle$ is true of a natural language only if

1. the natural language can be construed as an interpretation, I, of the SRL signature Σ, where I is a system of linguistic tokens,

2. each description in θ is true of each entity in the natural language, i.e., the natural language, I, is a $\langle \Sigma, \theta \rangle$ model, and

3. some description in θ is false of some component of each entity, u', in any Σ interpretation, I', for which no entity u in the natural language, I, has isomorphically configured components.

I will discuss the implications of King's three conditions for a mathematical characterization of the meaning of SRL grammars in turn:

A crucial aspect of condition 1 is that natural languages are understood as systems of linguistic tokens. King takes tokens to be objects, events, states, locations, or complexes (King, 1999, p. 316). Most importantly, linguistic tokens are at least partly empirical entities that are accessible to observation and testing. Typical manifestations of linguistic tokens are the utterances of a speaker. In order to account for the fact that almost all of the infinitely many grammatical sentences of a natural language never occur as actual utterances, King concedes that he must subsume non-actual tokens under his notion of linguistic tokens. While the actual linguistic tokens are those that did occur or will occur in our world, non-actual tokens happen not to be manifest in our universe. In the context of an architecture for phonology expressed in HPSG, Höhle 1999, pp. 74–77, elaborates this concept of tokens and refers to the utterance tokens in the denotation of a grammar as *possible utterance events*.[45] Adopting Höhle's terminology, I will usually call King's tokens the *possible tokens* of a language.[46]

The linguistic knowledge of native speakers determines what a legitimate token of their language is, independent of whether a token is actual or non-actual. In order to account for the familiar differences in the grammaticality judgments of native speakers that might depend on the particular situation in which a native speaker is tested, King appeals to Chomsky's notion of an ideal speaker/hearer of a language. The ideal speaker/hearer of a natural language is a fictional and flawless representative of the community of speakers of the natural language. According to King, 1999, p. 317, a linguistic token belongs to a natural language exactly if the linguistic knowledge of the ideal speaker/hearer judges it grammatical.

King holds that we do not know much about the exact nature of linguistic tokens yet. But we know enough about them to propose empirically testable hypotheses about configurations of linguistic tokens. A formalism in which these hypotheses can be meaningfully expressed must be compatible with the assumption that the entities that are subject to description are empirical entities and as such accessible to observation and testing. In a long discussion of the approach of Pollard and Sag 1994, which takes linguistic object types as the structures that are subject to grammatical description, King argues against the widespread representational interpretation of the meaning of grammars and in favor of a realistic interpretation that avoids the additional abstraction step from (at least partly) empirical entities to the theoretical construct of linguistic types. According to King, considerations of ontological parsimony suggest that linguistic types should be avoided in a linguistic formalism, unless to do so would compromise the simplicity of grammars. Condition 1 implies that the entities in the intended interpretation of the SRL grammar of a natural language are possible tokens of the

[45] Höhle sketches an interpretation function that maps phonological linguistic tokens in possible utterance events to physical utterance events. The theory of this function makes clear that King's token objects can be regarded as abstract entities that are not necessarily identical with physical events.

[46] Note that the endorsement of possible tokens calls into question an unqualified technical characterization of King's view as realistic. As mentioned before, I use the term merely as a convenient label that highlights some important aspects of King's theory. I have nothing to say about philosophy of science.

natural language. In Section 2.2.2.3, I will return to the issue of possible tokens as elements of the interpretation of grammars in a mathematical reconstruction of the distinction between linguistic types and linguistic tokens of Pollard and Sag 1994.

Condition 2 narrows down the intended interpretations of linguistic SRL grammars to models. For any linguistic SRL grammar $\langle \Sigma, \theta \rangle$, only those Σ interpretations are candidates for the intended interpretation whose entities are linguistic tokens, and each description in the set of principles, θ, describes every token in the interpretation. While condition 1 ties the intended interpretation to the signature of the grammar, condition 2 links it to the descriptions in θ.

Condition 3, finally, says that the natural language is an exhaustive model of the grammar. More technically, the condition is equivalent to saying that for every entity u' in every model I' of the grammar, there is an entity u in the natural language, I, such that $\langle u, \mathsf{I} \rangle$ and $\langle u', \mathsf{I}' \rangle$ are SRL congruent. Essentially, condition 3 affirms that the intended model of the grammar contains every possible token of the natural language, and nothing else. If there is a model of a hypothetic SRL grammar of a natural language that contains a configuration under an entity for which there is no isomorphically configured possible token of the language, then we must conclude that the grammar is flawed, because it permits possible tokens that cannot exist.

The problem with formulating the requirement that the intended model of a grammar is the model that contains exactly the possible tokens of the language is that there is no way in which the concept of a linguistic token—as opposed to a non-linguistic token—can be made mathematically precise. To get around that problem, the formal characterization of the intended linguistic model must be accomplished without an appeal to the unknown properties that distinguish linguistic entities from other entities in the world. To achieve this, King says that a grammar is only true of a natural language if the natural language is in a certain class of models, namely in the class of exhaustive models of the grammar. The class of exhaustive models of a grammar is determined independently of the question of whether the entities in a particular model in that class are all linguistic, partly linguistic, or not linguistic at all. The intended interpretation of the grammar remains the natural language, but all other models in the class of exhaustive models that the grammar determines are descriptively indistinguishable from the natural language. We then know that any model in the class of exhaustive models equals the natural language that we intend to describe in all descriptively relevant respects, and any model in that class can be used in model theoretic and computational investigations of the natural language. To put it in slightly different words, every model in the class of exhaustive models emulates the natural language model, and the existence of a configuration under an entity in any exhaustive model therefore implies the existence of an SRL congruent configuration under a possible token of the natural language.

King's conditions 1–3 characterize natural languages as being in a class of models of SRL grammars of these natural languages. If it turned out that we cannot conceive of a natural language as a model of an SRL grammar or not even as an interpretation of an SRL signature, this might easily jeopardize the scientific value of writing grammars of natural languages in the SRL formalism, unless there is a systematic relationship between the actual structures of natural languages and the interpretations of SRL signatures. Below, when I criticize the SRL formalism as being insufficient for capturing how HPSG linguists think of natural languages, I will claim that the interpretations of SRL are structures that are not rich enough to capture the structure of natural languages (as conceived by HPSG linguists). However, the richer interpretations with which I will replace the interpretations of SRL are a proper extension of SRL interpretations. If the richer formalism that I will propose is on the right track, the meaning of SRL grammars can still be regarded as a valuable approximation to the structure of natural languages.[47]

In the remainder of this section I will provide a precise definition of the class of exhaustive models of a grammar. There are different equivalent possibilities of defining the class of exhaustive models. I will present two of them. The more intuitive definition is algebraic and uses SRL congruence. It is phrased in parallel to King's condition 3. Alternatively, the class of exhaustive models can be defined on the basis of the denotation of descriptions alone.

I begin with an algebraic definition of exhaustive models. For that purpose, I first define what it means for one Σ interpretation to simulate another

[47] From a type-based, representational perspective, the same problem would look slightly different. In that case, the set of object types that an SRL grammar specifies is understood to stand in a systematic relationship to the phenomena of the empirical domain. If the SRL formalism proved to be inadequate for expressing HPSG 94 grammars, it would mean that the previously assumed systematic relationship between the set of object types and the empirical domain can no longer be maintained. SRL grammars would remain a valuable approximation, if a new and possibly more complicated systematic relationship could be found.

Σ interpretation. The idea is that a first Σ interpretation, I_1, simulates a second Σ interpretation, I_2, if for each configuration under an entity in I_2, there is an SRL congruent counterpart in I_1. As far as configurations under entities in I_2 are concerned, I_1 can therefore be regarded as a perfect substitute of I_2: It can differ from I_2 because it might contain configurations under entities for which there is no SRL congruent counterpart in I_2; but it contains a counterpart to every configuration under an entity that is in I_2.

Definition 27 *For each SRL signature Σ, for each Σ interpretation $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1 \rangle$, for each Σ interpretation $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2 \rangle$,*

I_1 ***simulates*** I_2 ***in*** Σ

iff for each $u_2 \in \mathsf{U}_2$, for some $u_1 \in \mathsf{U}_1$, $\langle u_1, \mathsf{I}_1 \rangle$ and $\langle u_2, \mathsf{I}_2 \rangle$ are SRL congruent in Σ.

The notion of one interpretation simulating another interpretation leads directly to the definition of exhaustive models:

Definition 28 *For each SRL signature Σ, for each $\theta \subseteq \mathcal{D}^{\Sigma}$, for each Σ interpretation I,*

I *is an* ***exhaustive*** $\langle \Sigma, \theta \rangle$ ***model*** *iff*

I *is a $\langle \Sigma, \theta \rangle$ model, and*
for each Σ interpretation I',
if I' is a $\langle \Sigma, \theta \rangle$ model then I simulates I' in Σ.

A model, I, of a grammar Γ is an exhaustive model of Γ exactly if for every configuration under an entity in any other model of Γ, we can find an SRL congruent counterpart in I. Loosely speaking, every configuration of entities in any model has a mirror image in an exhaustive model. In terms of linguistic grammars, the intended model in the class of exhaustive models of a grammar contains every possible grammatical token that is licensed by the grammar; and since it is a model, it does not contain any ungrammatical configurations under a token.

The relationship between an algebraic definition of exhaustive models as in Definition 28 and a characterization that is based on description denotations becomes clear when looking at the notion of two entities in two interpretations of the same SRL signature being indiscernible and comparing it to SRL congruence. Two entities in two Σ interpretations are indiscernible if it is impossible to tell the two entities in their interpretations apart by any Σ description that the formal language provides:

Definition 29 *For each SRL signature Σ, for each Σ interpretation $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1 \rangle$, for each $u_1 \in \mathsf{U}_1$, for each Σ interpretation $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2 \rangle$, for each $u_2 \in \mathsf{U}_2$,*

> *$\langle u_1, \mathsf{I}_1 \rangle$ and $\langle u_2, \mathsf{I}_2 \rangle$ are* ***indiscernible in*** *Σ*
> *iff for each $\delta \in \mathcal{D}^\Sigma$, $u_1 \in D_{\mathsf{I}_1}(\delta)$ iff $u_2 \in D_{\mathsf{I}_2}(\delta)$.*

Proposition 1 asserts that indiscernibility and congruence are in fact equivalent. If it is impossible to distinguish an entity, u_1, in a Σ interpretation, I_1, from another entity, u_2, in another Σ interpretation, I_2, by any description of the formal language, then $\langle u_1, \mathsf{I}_1 \rangle$ and $\langle u_2, \mathsf{I}_2 \rangle$ are congruent, and *vice versa*:[48]

Proposition 1 *For each SRL signature Σ, for each Σ interpretation $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1 \rangle$, for each $o_1 \in \mathsf{U}_1$, for each Σ interpretation $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2 \rangle$, for each $o_2 \in \mathsf{U}_2$,*

> *$\langle o_1, \mathsf{I}_1 \rangle$ and $\langle o_2, \mathsf{I}_2 \rangle$ are SRL congruent in Σ iff $\langle o_1, \mathsf{I}_1 \rangle$ and $\langle o_2, \mathsf{I}_2 \rangle$ are indiscernible in Σ.*

The equivalence of indiscernibility and congruence leads to the possibility of a characterization of the simulation of models in terms of the denotation of descriptions. For any given SRL signature Σ, for each set of Σ descriptions θ, and for each Σ interpretation I, to say that if a Σ interpretation I' is a $\langle \Sigma, \theta \rangle$ model then I simulates I' is equivalent to saying that if I' is a $\langle \Sigma, \theta \rangle$ model and an arbitrary theory θ' describes something in I' then θ' also describes something in I. In effect, as far as description denotation is concerned, I is a perfect substitute for the model I'. Proposition 2 summarizes this result:

Proposition 2 *For each SRL signature Σ, for each $\theta \subseteq \mathcal{D}^\Sigma$, for each Σ interpretation I,*

[48] I omit the proof of Proposition 1 and Proposition 2 at this point. Both propositions are special cases of the corresponding propositions, Proposition 10 and Proposition 11, of the extended formalism of Section 3.1, which I prove in Appendix A on page 376 and on page 380, respectively.

for each Σ *interpretation* I',

if I' *is a* $\langle\Sigma,\theta\rangle$ *model then* I *simulates* I' *in* Σ

iff for each $\theta' \subseteq \mathcal{D}^{\Sigma}$, *for each* Σ *interpretation* I',

if I' *is a* $\langle\Sigma,\theta\rangle$ *model and* $\Theta_{\mathsf{I}'}(\theta') \neq \emptyset$ *then* $\Theta_{\mathsf{I}}(\theta') \neq \emptyset$.

Since the simulation of models can alternatively be expressed in terms of the denotation of theories as made precise in the preceding proposition, we can replace the simulation condition in the definition of exhaustive models (DEFINITION 28) by the equivalent characterization of the relationship between two interpretations in terms of nonempty denotations of sets of descriptions in them, and we get an alternative way of describing the class of exhaustive models of a grammar:

Corollary 1 *For each SRL grammar* $\langle\Sigma,\theta\rangle$, *for each* Σ *interpretation* I,

I *is an exhaustive* $\langle\Sigma,\theta\rangle$ *model iff*

I *is a* $\langle\Sigma,\theta\rangle$ *model, and*
for each $\theta' \subseteq \mathcal{D}^{\Sigma}$, *for each* Σ *interpretation* I',
if I' *is a* $\langle\Sigma,\theta\rangle$ *model and* $\Theta_{\mathsf{I}'}(\theta') \neq \emptyset$ *then* $\Theta_{\mathsf{I}}(\theta') \neq \emptyset$.

Intuitively, COROLLARY 1 asserts that an interpretation, I, of an SRL grammar Γ is an exhaustive model exactly if any arbitrary theory that denotes something in some model of Γ also denotes something in I: Whatever we can describe in any Γ model thus finds a counterpart in I. This corresponds to the requirement of the algebraic definition that every configuration under an entity in any Γ model find an SRL congruent counterpart in I.

The advantage of the non-algebraic characterization of the exhaustive models of a grammar in COROLLARY 1 over the algebraic characterization of DEFINITION 28 is that it does not need any additional algebraic notions such as congruence and simulation. The definitions of the formal language of SRL and the definition of the model of a grammar are sufficient to say what the class of exhaustive models of a grammar is.[49]

[49] Richter et al. 1999, p. 293, and Richter 1999, p. 90, choose this characterization of exhaustive models exactly because it needs fewer preliminary definitions.

THEOREM 1 provides a final important mathematical result about exhaustive models. It assures that each SRL grammar has an exhaustive model. Since there are exhaustive models for each SRL grammar, it makes sense to explain the meaning of an arbitrary SRL grammar in terms of the class of its exhaustive models.

Theorem 1 (King 1999) *For each SRL signature Σ, for each Σ theory θ, there exists a Σ interpretation I such that I is an exhaustive $\langle \Sigma, \theta \rangle$ model.*

It follows immediately that each SRL grammar that has a nonempty model, i.e., a model that contains at least one entity, also has a nonempty exhaustive model.

2.2.2.3 Types and Tokens

At first blush, King's explanation of the meaning of SRL grammars of Section 2.2.2.2 appears to be so different from the view of Pollard and Sag 1994—summarized in Section 2.2.1 above—that it is hard to see how they are related. Although the philosophical differences in their understanding of the meaning of grammars are indeed considerable, it is nevertheless possible to make both approaches to language and grammar mathematically explicit within the single formal framework of SRL, and doing so provides interesting insights into the relationship between the two approaches. Pollard and Sag's account of the entities that are specified by a grammar will also allow us to see HPSG 94 under the perspective of some of the technical apparatus of HPSG 87, thus making some aspects of the relationship between the two types of formalisms clearer on an intuitive and on a technical level.

Firstly, Pollard and Sag (1994) hold that an HPSG 94 grammar is about the object types of a language and not about possible tokens. A grammar admits a collection of object types. This view is motivated by their assumption that the knowledge of the speakers of a natural language is not about individual tokens of sentences like *John loves Mary*, but rather about the unique object type *John loves Mary*. Pollard and Sag do not specify the relationship between the object type of a sentence and the observable tokens of it in the world, but it seems natural to assume that there must be some kind of abstraction function that maps the tokens of a sentence to their common, unique object type. Secondly, Pollard and Sag assume that object types are modeled by some kind of totally well-typed and sort resolved feature structures. By contrast, the entities in the exhaustive models of King are not necessarily feature structures. The entities in interpretations of SRL signatures are simply given as a set of entities in an interpretation, without any commitment to their internal structure.[50] Moreover, the intended linguistic model in the class of exhaustive models that is the natural language is thought of as a system of possible tokens. The possible tokens of a language are entities in an interpretation. Linguistic tokens in an interpretation are not mathematical entities like feature structures; they are the empirical entities that constitute a natural language.

There are numerous ways to reconstruct the relationship between the views of King on the one hand and Pollard and Sag on the other. The most comprehensive mathematical presentation of this issue is put forth by King (1996, pp. 33–36), who provides a translation from Pollard and Sag's signatures with sort hierarchies to SRL signatures with the same interpretations, and constructs Pollard and Sag's object types as equivalence classes of indiscernible pairs of an entity and an interpretation, $\langle u, \mathsf{I} \rangle$, where $u \in \mathsf{I}$. By choosing equivalence classes of entities in interpretations as the objects of interest, the distinction between indiscernible (or SRL congruent) linguistic entities in interpretations is lost. Loosing the distinction between indiscernible possible tokens in interpretations captures the central idea of Pollard and Sag's object types. King proceeds to define feature structures under an SRL signature. *King feature structures* are essentially potentially infinite, totally well-typed and sort resolved abstract feature structures.[51] A definition of *King feature structure admission* says what it means for a set of Σ descriptions to admit a set of King feature structures under Σ. The basic idea is that a King feature structure is admitted by a set of descriptions iff each (abstract) node of the King feature structure satisfies each description in the set. Feature structure admission can thus be seen as saying for each SRL grammar Γ which set of King feature structures is admitted by Γ.[52]

Abstraction functions finally establish the connection between interpre-

[50] As I will show later, for each entity in an interpretation, a uniquely determined concrete feature structure can be constructed from the subalgebra generated by the entity. In that sense, the subalgebra generated by an entity in an interpretation can be understood as indicating an internal structure of the entity. In principle, however, the internal structure of an entity in an interpretation is unrelated to the structure of the subalgebra that it generates.

[51] The definition of *94 abstract feature structures* in DEFINITION 32 below is very similar to the definition of King feature structures.

[52] The definition of *morph admission*, DEFINITION 77, on page 195 is essentially an extension of King's feature structure admission.

tations and King feature structures: The abstraction functions that King defines are total functions from the set of entities in a Σ interpretation to the set of King feature structures under Σ. King shows that for each signature Σ, and for each Σ interpretation I, I is an exhaustive Γ model if and only if the abstraction function from the set of entities in the Σ interpretation I to the set of King feature structures under Σ determines a bijection from the set of object types determined by I to the set of King feature structures under Σ admitted by Γ (King, 1996, p. 36). In other words, the object types of any exhaustive model of a grammar stand in a one-to-one correspondence to the King feature structures admitted by that grammar. It is therefore justified to regard the set of King feature structures admitted by the grammar as the set of object types that Pollard and Sag originally had in mind. From King's point of view, the set of King feature structures that are admitted by an HPSG 94 grammar are structured mathematical objects that represent the object types of a natural language; and the object types of a natural language that they represent are equivalence classes of indiscernible possible tokens in the natural language.

In the present section I will investigate the same general picture from a slightly different perspective. I am interested in clarifying how elements of the HPSG 87 formalism relate to the distinction between possible tokens and types in HPSG 94. By relating interpretations of SRL grammars to sets of concrete feature structures, I obtain an alternative way of deriving Pollard and Sag's object types from the exhaustive models of an SRL grammar. What is interesting about choosing this route is that it establishes the link between exhaustive models and feature structure representations of object types by using definitions and techniques that we know from HPSG 87. Studying the distinction of possible tokens and object types in terms of two kinds of feature structures of HPSG 87 and their relationship sheds light on the change of perspective from HPSG 87 to HPSG 94 regarding the task of a grammar.

Pollard and Sag 1994 envisions an architecture of grammar in which a grammar determines a set of abstract feature structures by an admission function that assigns each grammar a set of abstract feature structures. The summary of the mathematical reconstruction in the SRL formalism of Pollard and Sag's object types has revealed their relationship to the exhaustive models of a grammar. Since the members of the set, S, of abstract feature structures admitted by a grammar stand in a one-to-one correspondence to the indiscernibility classes of possible tokens in an exhaustive model of the grammar, the answer that S gives to the question which linguistic data a grammar predicts is empirically equivalent to the answer that the exhaustive models of the grammar give. The admitted abstract feature structures represent the object types of a natural language; and the set of admitted abstract feature structures is the content of an HPSG 94 grammar. According to Pollard (personal communication), this is the standard view of generative grammarians specialized to the case where the structural representations of linguistic objects are abstract feature structures. Given the properties of abstract feature structures that we have seen in Section 2.1.2, it is an obvious thought to assume that if abstract feature structures correspond to the object types of a natural language, then the system of possible tokens of a natural language corresponds to a collection of concrete feature structures. In order to spell out this idea, I first need a definition of the collection of *94 concrete feature structures*.

Just as 87 concrete feature structures, 94 concrete feature structures are concrete feature structures in the sense of the general DEFINITION 2 with a few additional restrictions. The restrictions of 87 concrete feature structures were finiteness, acyclicity and the appropriateness of their attributes. The restrictions of 94 concrete feature structures are total well-typedness and sort resolvedness. A minor complication with defining 94 concrete feature structures as a special case of concrete feature structures is that in the current SRL-based framework, signatures do not contain a sort hierarchy, whereas concrete feature structures were defined as concrete feature structures under an 87 signature with a sort hierarchy. In addition, the appropriateness function, $\mathcal{F}$, of 87 signatures is a partial function from the Cartesian product of the set of sorts and the set of attributes to the set of sorts, whereas in SRL signatures, $\mathcal{F}$ is a total function from the Cartesian product of the set of species and the set of attributes to the power set of the set of species. To accommodate these differences, I need to introduce an auxiliary 87 signature, Σ', in the definition of 94 concrete feature structures under the SRL signature Σ:

Definition 30 *For each SRL signature* $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{F} \rangle$, $\mathbb{C}$ *is a* ***94 concrete feature structure under*** Σ *iff*

$$\sqsubseteq = \left\{ \langle \sigma, \sigma \rangle \in \mathcal{S} \times \mathcal{S} \,\middle|\, \sigma \in \mathcal{S} \right\},\ \mathcal{F}' = \{\},\ \Sigma' = \langle \mathcal{S}, \sqsubseteq, \mathcal{A}, \mathcal{F}' \rangle,$$

$$\mathbb{C} = \langle Q, \hat{q}, \Delta, \Lambda \rangle \in \mathbb{CFS}^{\Sigma'},$$

for each $q_1 \in Q$, *for each* $q_2 \in Q$, *for each* $\alpha \in \mathcal{A}$,

$\Delta \langle q_1, \alpha \rangle$ *is defined iff* $\mathcal{F} \langle \Lambda(q_1), \alpha \rangle \neq \emptyset$, *and*
if $\Delta \langle q_1, \alpha \rangle$ *is defined then* $\Lambda(\Delta \langle q_1, \alpha \rangle) \in \mathcal{F} \langle \Lambda(q_1), \alpha \rangle$.

If Σ is an SRL signature, I write $\mathbb{CFS}^{\Sigma}_{94}$ for the class of 94 concrete feature structures under Σ. Note that $\langle \mathcal{S}, \sqsubseteq, \mathcal{A}, \mathcal{F}' \rangle$ is an 87 signature. Its sort hierarchy, $\langle \mathcal{S}, \sqsubseteq \rangle$, is the finite partial order in which each species only subsumes itself, and the partial appropriateness function, $\mathcal{F}'$, is undefined on each pair of a species and an attribute. The signature Σ' is needed, because I defined concrete feature structures with reference to 87 signatures instead of SRL signatures. However, neither the partial order of the sort hierarchy nor the appropriateness function are actually referred to in the general definition of concrete feature structures, DEFINITION 2. Therefore, I can fix them in any way that satisfies the requirements of the definition of 87 signatures without consequences. Note also that the sort resolvedness of 94 concrete feature structures follows immediately from the fact that they are defined under SRL signatures, which only contain species and leave the sort hierarchy implicit.

With a precise notion of 94 concrete feature structure in hand, I can now pursue the question of how 94 concrete feature structures are related to the entities in interpretations of SRL signatures. DEFINITION 31 implies a straightforward answer:

Definition 31 *For each SRL signature* $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{F} \rangle$, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$,

$$\mathbb{CFS}^{\mathsf{I}}_{94} = \left\{ \begin{array}{l} \langle Q, \hat{q}, \Delta, \Lambda \rangle \\ \in \mathbb{CFS}^{\Sigma} \end{array} \middle| \begin{array}{l} \textit{for some } u \in \mathsf{I}, \\ Q = \mathsf{Co}_{\mathsf{I}}(u), \\ \hat{q} = u, \\ \Delta = \left\{ \begin{array}{l} \langle \langle u_1, \alpha \rangle, u_2 \rangle \in \\ (Q \times \mathcal{A}) \times Q \end{array} \middle| \begin{array}{l} \textit{for some } \alpha \in \mathcal{A}, \\ \textit{for some } u_1 \in \mathsf{Co}_{\mathsf{I}}(u), \textit{ and} \\ \textit{for some } u_2 \in \mathsf{Co}_{\mathsf{I}}(u), \\ \langle \alpha, \langle u_1, u_2 \rangle \rangle \in \\ \mathsf{A} \cap (\mathcal{A} \times (\mathsf{Co}_{\mathsf{I}}(u) \times \mathsf{Co}_{\mathsf{I}}(u))) \end{array} \right\}, \\ \Lambda = \mathsf{S} \cap (\mathsf{Co}_{\mathsf{I}}(u) \times \mathcal{S}) \end{array} \right\}.$$

It can easily be checked that for each SRL signature Σ, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$, and for each $u \in \mathsf{I}$, there is a concrete feature structure in $\mathbb{CFS}^{\mathsf{I}}_{94}$ with root node u. For each $u \in \mathsf{I}$, Q is a set, because the collection of components of each entity in an interpretation is a set. Under each root node $\hat{q} = u$, the connectivity condition on concrete feature structures is satisfied, because the transition function Δ is derived from the attribute interpretation function restricted to the components of u: The transition from $u_1 \in Q$ to $u_2 \in Q$ with attribute α is defined iff the attribute interpretation of α at the component u_1 of u is defined and yields the component u_2 of u. But being a component of some u in an interpretation is defined as being reachable by finitely many interpretations of attributes, and connectivity in concrete feature structures means that every node in the concrete feature structure can be reached from the root node in finitely many transitions that consume finitely many attributes. Finally, for each $u \in \mathsf{I}$, Λ is a total function from Q to $\mathcal{S}$, because Q is the set of components of u, and Λ is the species interpretation function restricted to the components of u.

Since $\mathbb{CFS}^{\mathsf{I}}_{94}$ is defined for interpretations of SRL signatures, which must obey the appropriateness function, and since the requirements of SRL's appropriateness function correspond to the total well-typedness of feature structures, the elements of $\mathbb{CFS}^{\mathsf{I}}_{94}$ can be characterized even more specifically as 94 concrete feature structures:

Proposition 3 *For each SRL signature Σ, for each Σ interpretation I,*

$\mathbb{CFS}^{\mathsf{I}}_{94}$ *is a set of 94 concrete feature structures.*

For each SRL signature Σ and for each Σ interpretation I, I call $\mathbb{CFS}^{\mathsf{I}}_{94}$ the *set of 94 concrete feature structures determined by* I. For each entity $u \in \mathsf{I}$, there is exactly one 94 concrete feature structure with root node u in $\mathbb{CFS}^{\mathsf{I}}_{94}$. If we look at an arbitrary element $\mathbb{C} = \langle Q, \hat{q}, \Delta, \Lambda \rangle$ with root node $\hat{q} = u$ of $\mathbb{CFS}^{\mathsf{I}}_{94}$, we can see an interesting correspondence of $\mathbb{C}$ to the configuration of entities under u in I, $\langle u, \langle \mathsf{U}_u, \mathsf{S}_u, \mathsf{A}_u \rangle \rangle$: Q equals U_u, Λ equals S_u, and there is a bijection from A_u to Δ. It follows that for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$, there is a bijection from the set of configurations of entities under some u in I to $\mathbb{CFS}^{\mathsf{I}}_{94}$ such that for each $u \in \mathsf{I}$, $\langle u, \langle \mathsf{U}_u, \mathsf{S}_u, \mathsf{A}_u \rangle \rangle$ is mapped to the 94 concrete feature structure in $\mathbb{CFS}^{\mathsf{I}}_{94}$ with the root node u. A 94 concrete feature structure determined by I with root node u is essentially another way of looking at the configuration under u in I.

Exhaustive models of an SRL grammar Γ contain SRL congruent counterparts to every configuration under an entity that can be found in some Γ model. For an exhaustive Γ model I, $\mathbb{CFS}^{\mathsf{I}}_{94}$ thus contains at least one 94 concrete feature structure that corresponds to an SRL congruent counterpart of any configuration under an entity that can occur in some Γ model.

Intuitively speaking, we might say that it contains at least one 94 concrete feature structure of every shape that is licensed by Γ. But just as in the case of exhaustive models where multiple occurrences of isomorphic configurations under an entity in an exhaustive model are possible, $\mathbb{CFS}^{\mathsf{I}}_{94}$ might contain multiple 94 concrete feature structures of the same shape. To be more precise, whenever an exhaustive Γ model I contains n configurations under an entity that are SRL congruent to $\langle u, \mathsf{I}_u \rangle$ for some u in I, then $\mathbb{CFS}^{\mathsf{I}}_{94}$ contains n 94 concrete feature structures with different root nodes such that they are pairwise isomorphic with the 94 concrete feature structure with the root node u. To put it slightly differently, these 94 concrete feature structures are identical except for their nodes. Assuming King's interpretation of the meaning of SRL grammars, if I is the exhaustive Γ model that is the natural language, then isomorphic 94 concrete feature structures in $\mathbb{CFS}^{\mathsf{I}}_{94}$ are mathematical representations of different tokens in the natural language of the same kind of entity, for example of different tokens of the sentence *Kim likes bagels*.

In HPSG 87, we encountered a corresponding phenomenon with 87 concrete feature structures. The fact that two distinct 87 concrete feature structures could potentially subsume each other prevented them from forming a partial order under subsumption, and we introduced 87 abstract feature structures in order to abstract away from distinct entities that carried the same information in the same way. The abstraction function collapsed 87 concrete feature structures that mutually subsumed each other into a single 87 abstract feature structure. Apart from being slightly unintuitive under the view that feature structure subsumption between different feature structures corresponds to the relative informativeness of feature structures, the potential isomorphisms between distinct 87 concrete feature structures prevented them from forming a partial order under subsumption. This was one of the reasons for why they could not form a Heyting algebra under subsumption either. The Heyting algebra of feature structures was an important part of the formalization of the HPSG 87 formalism of Section 2.1, because it provided the algebraic operations on feature structures on which an HPSG 87 grammar relies.

The perspective that HPSG 94 takes regarding concrete and abstract feature structures is very different, but the same technique that is used in HPSG 87 to get from concrete to abstract feature structures can be used in HPSG 94 for a different purpose, namely for capturing the relationship between representations of tokens and representations of types. If I is the natural language, then the elements of $\mathbb{CFS}^{!}_{94}$ are complete mathematical representations of the possible tokens in the natural language, and it does not make sense to ask about the relative informativeness of the isomorphic mathematical representations of two tokens. However, we might want to collapse them into a single representation of the object type of the two tokens. From HPSG 87 we know that the abstraction relation, Abst_Σ, of DEFINITION 10 does exactly what we are looking for: It abstracts away from the distinction that is caused by isomorphic concrete feature structures having distinct nodes.[53] The domain of the *94 abstraction function under* Σ that can be derived as a special case from Abst_Σ is the collection of 94 concrete feature structures, $\mathbb{CFS}^{\Sigma}_{94}$, and its range the set of 94 abstract feature structures, $\mathbb{AFS}^{\Sigma}_{94}$, which I define as follows:

Definition 32 *For each SRL signature* $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{F} \rangle$*,* $\mathbb{A}$ *is a* ***94 abstract feature structure under*** Σ *iff*

$$\sqsubseteq = \left\{ \langle \sigma, \sigma \rangle \in \mathcal{S} \times \mathcal{S} \,\middle|\, \sigma \in \mathcal{S} \right\},\ \mathcal{F}' = \{\},\ \Sigma' = \langle \mathcal{S}, \sqsubseteq, \mathcal{A}, \mathcal{F}' \rangle,$$

$$\mathbb{A} = \langle \beta, \varrho, \lambda \rangle \in \mathbb{AFS}^{\Sigma'},$$

for each $\pi \in \mathcal{A}^*$*, for each* $\alpha \in \mathcal{A}$*,*

$$\textit{if } \pi\alpha \in \beta \textit{ then } \lambda(\pi\alpha) \in \mathcal{F}\langle \lambda(\pi), \alpha \rangle,$$

for each $\pi \in \mathcal{A}^*$*, for each* $\alpha \in \mathcal{A}$*,*

$$\textit{if } \pi \in \beta \textit{ and } \mathcal{F}\langle \lambda(\pi), \alpha \rangle \neq \emptyset \textit{ then } \pi\alpha \in \beta.$$

If Σ is an SRL signature, I write $\mathbb{AFS}^{\Sigma}_{94}$ for the set of 94 abstract feature structures under Σ. Due to the restrictions on 94 abstract feature structures under Σ, they are totally well-typed. The sort resolvedness follows immediately from the fact that they are defined under SRL signatures.

All relevant properties of Abst_Σ hold of the 94 abstraction relation between $\mathbb{CFS}^{\Sigma}_{94}$ and $\mathbb{AFS}^{\Sigma}_{94}$. In particular, it is a total function, and it maps isomorphic 94 concrete feature structures to the same 94 abstract feature structure. The nodes of the 94 concrete feature structures are turned into equivalence classes

[53] Again, we need to modify the definition of Abst_Σ minimally in order to account for the fact that Σ is now an SRL signature instead of an 87 signature. As with other definitions before, this modification is unproblematic.

of paths that lead from the root node to the same node. Since an exhaustive Γ model, I, of a grammar Γ always contains SRL congruent counterparts to all existing configurations under an entity in any Γ model, and since it may only differ from other exhaustive Γ models with regard to how many SRL congruent "copies" of configurations under an entity it contains, the set of 94 concrete feature structures determined by I, $\mathbb{CFS}^{\mathsf{I}}_{94}$, only differs from the sets of 94 concrete feature structures determined by other exhaustive Γ models in the number of isomorphic 94 concrete feature structures of a given shape. Suppose that for a given grammar Γ and an exhaustive Γ model I, $\mathbb{AFS}^{\mathsf{I}}_{94}$ is the set of 94 abstract feature structures that we get by collecting the results of applying the 94 abstraction function to the members of $\mathbb{CFS}^{\mathsf{I}}_{94}$. By the properties of the 94 abstraction function and by the properties of the sets of 94 concrete feature structures determined by exhaustive Γ models it follows that for each SRL signature Σ, for each SRL grammar $\Gamma = \langle \Sigma, \theta \rangle$, and for each pair of exhaustive Γ models I and I', $\mathbb{AFS}^{\mathsf{I}}_{94} = \mathbb{AFS}^{\mathsf{I}'}_{94}$. That means that the set of 94 abstract feature structures that we abstract from each $\mathbb{CFS}^{\mathsf{I}}_{94}$ is actually independent of the choice of the exhaustive Γ model, I. It is always the same set of 94 abstract feature structures. Following the terminology of Pollard and Sag, I call it the *set of 94 abstract feature structures admitted by* Γ, and write $\mathbb{AFS}^{\Gamma}_{94}$.

Constructing the set of the 94 abstract feature structures, $\mathbb{AFS}^{\Gamma}_{94}$, admitted by a grammar Γ from a set of 94 concrete feature structures determined by an exhaustive Γ model can be shown to be equivalent to the definition of King feature structure admission of King 1996.[54] The members of $\mathbb{AFS}^{\Gamma}_{94}$ are mathematical representations of the object types of the natural language. In fact, arriving at $\mathbb{AFS}^{\Gamma}_{94}$ via sets of 94 concrete feature structures can be regarded as a complication of King's simpler abstraction function that maps entities in an interpretation directly to abstract feature structures. If the interpretation is an exhaustive Γ model, King's abstraction function is a surjection from the set of entities in the exhaustive Γ model to the abstract feature structures admitted by Γ. However, the detour that I have taken illuminates a number of terminological and conceptual issues in HPSG 94.

Firstly, two approaches to linguistic tokens are technically conceivable. The collection of possible tokens of a language is the intended model in the

[54]In order to prove this, King's definition of King feature structures must be replaced by 94 abstract feature structures, which causes some further minor adaptations in some mathematical details of King 1996.

realistic approach to the meaning of HPSG 94 grammars of King. Grammars directly specify the system of actual and non-actual tokens of a natural language. The natural language is one particular model in the class of exhaustive models of the grammar. Feature structures do not occur in this explanation of the meaning of HPSG 94 grammars at all. As we have seen, it is nevertheless possible to represent the possible tokens of a language as a collection of concrete feature structures. Any collection of 94 concrete feature structures determined by an exhaustive model of the grammar in question is appropriate to that task. In the set $\mathbb{CFS}_{94}^{\mathcal{L}}$ of 94 concrete feature structures that is determined by the natural language model, $\mathcal{L}$, in the class of exhaustive models of the grammar, every 94 concrete feature structure is a mathematical representation of a possible token of the language. The theoretical problem with this construction is that, under the perspective of a realist, mathematical representations of tokens are nonsensical, because a realist wants to specify tokens without any intervening modeling structure. From the representational perspective, specifying possible tokens as a collection of concrete feature structures is not reasonable either, because proponents of the representational approach aim at mathematical representations of the object types of the language as modeling structures and deny that it makes sense to see the meaning of a grammar in the collection of the possible tokens of the language. This position stands in the tradition of generative grammar, which sees the specification of linguistic competence as one of its major goals. Linguistic competence is understood as a knowledge system that is shared by the speakers of a language. The knowledge represented in the minds of the speakers of a language is assumed to be knowledge of linguistic object types rather than knowledge of individual linguistic utterance tokens.

Secondly, there are two approaches to types. In King's realistic view of HPSG 94, the object types of a natural language are the equivalence classes of indiscernible possible tokens of the natural language. They are an additional construct that is of no immediate interest, because the meaning of a grammar is given by its exhaustive models. In the eyes of Pollard and Sag (1994), this is completely different, because they see the meaning of a grammar in the set of object types admitted by the grammar. The object types of a natural language are represented by a set of potentially infinite, totally well-typed and sort resolved abstract feature structures. The set of 94 abstract feature structures admitted by a given grammar $\Gamma = \langle \Sigma, \theta \rangle$, $\mathbb{AFS}_{94}^{\Gamma}$, is the set of 94 abstract feature structures under Σ such that each (abstract) node of each one of them satisfies all descriptions in θ. $\mathbb{AFS}_{94}^{\Gamma}$ is thus determined directly by a feature structure admission function. Since the tokens of a language are the empirical basis of grammars, they are deemed as important in this approach as under the realistic perspective. However, in contrast to the realistic approach, tokens are not part of the linguistic formalism. The connection between the formalism and the data is made by the linguistic theory, which postulates a correspondence between certain observable aspects of tokens and certain properties of the structure of the object types that the grammars specifies.

We have seen two ways of relating the realistic and the representational interpretation of the object types of a natural language to each other. Firstly, I have summarized work by King, who defines abstraction functions that map each entity, u, in an exhaustive model, I, of a grammar to the abstract feature structure admitted by the grammar that represents the object type to which $\langle u, \mathsf{I} \rangle$ belongs. Secondly, I have sketched a way of first deriving the set of concrete feature structures determined by an exhaustive model of a grammar, and then abstracting the set of abstract feature structures admitted by the grammar from that set. Again, for each entity, u, in an exhaustive model, I, of a grammar, we obtain the abstract feature structure admitted by the grammar that represents the object type to which $\langle u, \mathsf{I} \rangle$ belongs. The second method is less direct, but it illuminates some aspects of the relationship between HPSG 87 and HPSG 94.

Comparing HPSG 94 with HPSG 87, we see a stark contrast in the role that feature structures play in the two formalisms. In HPSG 94, they are optional constructs that can be used as mathematical representations of tokens and types. But tokens and types can be defined without recourse to feature structures. From a realist's perspective, they are merely a convenient representation that does not play a role in explaining the meaning of an HPSG 94 grammar. Of course, that standpoint does not deny the mathematical or computational importance of feature structures in the broader context of an HPSG 94 formalism.[55] From a representational perspective, abstract feature structures are the only type of feature structures that are needed in HPSG 94, since the set of abstract feature structures admitted by an SRL grammar can be determined directly via an admission function.

In HPSG 87, feature structures are an integral part of the grammatical formalism. They function as indispensable representations of partial infor-

[55]For example, as we will see below, the existence proof of exhaustive models employs a canonical feature structure model.

mation. The fact that disjunctive feature structures form a Heyting algebra under subsumption provides the algebraic operations with which the grammatical principles and the way in which they are combined are formulated. The main source of the different status of feature structures in the two formalisms lies in a seemingly innocuous technical detail and the intuition about the task of grammars that it is meant to express: The feature structures of HPSG 87 are partial, because they are representations of partial information, and the unification of feature structures expresses the accumulation of partial information. The feature structures of HPSG 94 are totally well-typed and sort resolved, because, according to Pollard and Sag 1994, they are total representations of linguistic object types.

Coming back to the ongoing debate between the proponents of a realistic and of a representational interpretation of HPSG 94 grammars, it can be argued that the representational view of the relationship between the structures specified by a grammar and the empirical domain is currently more widely accepted among generative grammarians than the view proposed by King. It is not obvious, however, if the differences between the two, although philosophically deep, are significant for current linguistic practice in the HPSG framework. Since for each SRL grammar, both approaches make the same predictions about the empirical domain, only strong assumptions about the relationships between the possible tokens or the object types of a natural language and observable data, and arguments from philosophy of science can distinguish between the two positions.[56] An investigation of these issues is, however, well beyond the scope of my thesis, and I restrict myself to a very short summary of the arguments that have been put forth by King and Pollard.

King 1999 argues against the view that linguistic object types should be specified by HPSG 94 grammars and against explaining the meaning of grammars by means of feature structures. Regarding the notion of the object types of a natural language, which is fundamental in the theory of Pollard and Sag, King doubts that linguists share a commonly accepted intuition about what exactly a linguistic type is: "[It] is an amorphous, artificial notion that linguists can and do construe in a variety of ways to suit their particular purposes. Witness the different ways members of the same col-

[56]For a very interesting discussion of the relationship between possible tokens and physical utterance events, and of the theoretical implications of different assumptions about this relationship, see Höhle 1999, pp. 74–77.

lection of speech events are divided into types as the concerns of a linguist are phonetic or phonological" (King, 1999, p. 313). Regarding the integration of object types into the HPSG 94 framework, King claims that object types are secondary mathematical entities in a theory of natural language, because they are not entities in the empirical domain of observable linguistic events. The object types of a grammar themselves are immune to empirical testing, and it is only their relation to tokens that makes them empirically accessible. Given this situation, King argues that the scientific principle of ontological parsimony suggests to avoid object types completely. The object types of a grammar would only become empirical objects if it could be shown that the mental representations of linguistic knowledge are indeed about object types, and if there were methods to observe and test these representations. From the realistic perspective of King, a similar argument of parsimony applies to the use of feature structures. Feature structures can be used as mathematical representations of tokens and types, but they are dispensable. In particular, nothing can be gained by representing language tokens as 94 concrete feature structures instead of assuming that the system of language tokens itself is the intended model of the grammar. In an early paper on feature logics, Gert Smolka already formulated this perception of the status of feature structures in grammar formalisms of the kind of HPSG 94. Contrasting the closed world approach, in which only feature structures are admitted in the interpretations, with an open world semantics, in which there is no commitment as to the nature of the entities in the interpretations of the expressions of the logical language, Smolka (1988, p. 38) observes: "Although the closed world approach is technically okay, it is unsatisfying philosophically [...]. The open world semantics presented in this paper renders the detour over feature structures superfluous: one can now view feature terms as directly denoting sets of linguistic objects and there is still no need for making precise what linguistic objects are."

Besides the philosophical argument, King cautions against using feature structures and feature structure admission as the basis of the content of a grammar for methodological reasons. Feature structure admission is defined with abstract feature structures. In the discussion of formalisms of HPSG 87 and HPSG 94, we have seen that many different definitions of abstract feature structures exist. When natural languages are modeled by feature structures, it is often not clear which linguistic intuitions correspond to particular restrictions on feature structures. Typical examples are the acyclicity condition or the finiteness condition on feature structures. But if the linguistic significance of that kind of technical decision is not transparent then a linguist who relies on a particular definition of feature structure admission cannot know which assumptions about language might be implicitly hidden in the formalism, and the linguist might or might not share the hidden underlying intuitions. In contrast to the purely technical definition of the meaning of a grammar by feature structure admission, the approach of King expresses three necessary conditions that say what it means for a grammar to be true of a natural language. A linguist who uses that formalism is thus informed about the assumptions about natural languages that are inherent in the formalism.

In accordance with the mainstream opinion about the meaning of grammars in generative linguistics, Pollard (personal communication) rejects the idea that a grammar specifies a collection of tokens. Most importantly, Pollard is deeply troubled by the necessity of postulating non-actual tokens, which is caused by the (intended) fact that a grammar of a natural language specifies infinitely many non-isomorphic phrases, whereas one can reasonably assume that only finitely many tokens of a language ever occur in our world. It follows immediately that almost all tokens specified by an SRL grammar are fictitious entities. To Pollard, the concept of non-actual tokens is contradictory and nonsensical, and a theory of language which must assume them is unacceptable. More technically, Pollard doubts that there are any reasons to assume that the collection of possible tokens of a natural language forms a set, as it is supposed by the models of SRL grammars. If one does not share this premise, the model theory of King implies an assumption that is not supported by intuitions about the concept of possible tokens. In contrast to King, Pollard agrees with the widespread assumption that a scientific theory of language should not describe individual linguistic tokens within the formal framework. Instead, mathematical representations of object types that abstract away from individual tokens are the right choice. Linguistic tokens, manifest in utterance events, are predicted by the theory via intervening modeling structures that stand in a conventional correspondence relation to utterance events.[57] For proponents of this architecture of linguistic theories, even the actual tokens among the possible tokens of a language are unfit to

[57]This view is strongly reminiscent of physical theory as it emerged at the turn of the century. It distinguishes an *experimental domain*, a *mathematical model*, and a *conventional interpretation*. See Abraham and Marsden 1967, pp. 1–4, for a brief overview. Note that this theory precedes the formulation by Tarski of model-theoretic semantics, which might introduce interesting new aspects that were not recognized at the time.

appear as modeling structures, because their fundamental nature cannot be precisely known. Given that King's account must concede the existence of non-actual tokens, it may even appear questionable if his account is more parsimonious than a representationalist view that eschews ontological assumptions and defines the meaning of a grammar as a certain mathematical entity.

In the next section, I present Pollard's definition of the strong generative capacity of a grammar. Like its intellectual ancestor, the account of Pollard and Sag 1994, Pollard's (1999) account is strictly representational, and similar arguments apply for and against Pollard 1999 as for and against the approach of Pollard and Sag 1994. However, there are some notable differences. First of all, Pollard 1999 avoids claims about the ontological status of object types and regards them as isomorphism classes of structures that include the structures that idealize tokens. Object types do not necessarily exist in the world. On the technical side, this slight change of emphasis finds its counterpart in new modeling structures. Although Pollard 1999 still refers to them as feature structures and abstract feature structures, they are no longer the automata theoretic concrete feature structures and abstract feature structures of the mathematical reconstruction of Pollard and Sag 1994 of the present section.

2.2.2.4 Strong Generative Capacity

The purpose of Pollard 1999 is to give a precise characterization of the meaning of HPSG 94 grammars that explains in which sense an HPSG 94 grammar is a generative grammar. In contrast to Pollard and Sag 1994, Pollard 1999 does not espouse the idea that an HPSG 94 grammar specifies the set of object types of a natural language; nor does it share the view that an HPSG 94 grammar specifies the possible tokens of a natural language. The goal of the paper is to formally define the strong generative capacity (SGC) of object-based HPSG. With the SGC of a grammar, Pollard intends to describe the set of structures that is "generated" by an HPSG 94 grammar in a similar sense in which a generative grammar generates a language. According to Pollard, the SGC of a grammar should capture two intuitions. Firstly, no two members of the SGC of a grammar are structurally isomorphic. Secondly, if a grammar is correct, exactly those utterance tokens that are structurally isomorphic to members of the SGC will be judged grammatical (Pollard, 1999, p. 282). Pollard's definition of the SGC of a grammar is related to representations of types and to representations of tokens. With types it shares the assumption that there is only one representative of a each collection of isomorphically configured linguistic entities. However, in contrast to the two characterizations of types in the previous section, where types have a different structure than tokens, it assumes that each token of the language is structurally isomorphic to a member of the SGC.

Pollard 1999 contains very precise descriptions of the mathematical constructs it envisages. In the present section, I spell out the technical details of the paper in the notation and terminology of the previous sections in order to fit its concepts seamlessly into the context of the other explications of HPSG 94. I proceed in three steps. Firstly, I will present Pollard's definition of a feature structure, which is a construct that we have already seen under a different name. Secondly, Pollard's abstract feature structures are introduced as a canonical form of Pollard's feature structures. Thirdly, I present Pollard's three ways of obtaining the SGC of a grammar as sets of abstract feature structures, and I discuss their relationship to the above formalizations of tokens and types as feature structures. Finally, I use the definitions of Pollard to formulate a definition of a canonical exhaustive model of a grammar that Pollard (1999, p. 295) indicates only briefly.

Pollard Feature Structures Pollard 1999 chooses to define feature structures in a non-standard fashion, exploiting the existence of a bijection between the set of 94 concrete feature structures determined by an interpretation, I, and the set of all pairs of an entity, u, of I and the subinterpretation of I in which all entities have u as their common ancestor. In order to signal the difference between Pollard's feature structures and the standard definitions of concrete feature structures, I will call the new feature structures *Pollard feature structures*. DEFINITION 33 introduces Pollard's terminology for what I have called the interpretation under an entity in an interpretation (see DEFINITION 23):

Definition 33 *For each SRL signature* $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{F} \rangle$*, for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$*, for each* $v \in \mathsf{U}$*,*

$$\mathsf{I}_v = \langle \mathsf{U}_v, \mathsf{S}_v, \mathsf{A}_v \rangle \textit{ is the } \Sigma \textbf{\textit{ interpretation generated by }} v \textbf{\textit{ in }} \mathsf{I} \textit{ iff}$$

$$\mathsf{U}_v = \left\{ u \in \mathsf{U} \,\middle|\, \begin{array}{l} \textit{for some } \tau \in \mathcal{T}^{\Sigma} \\ T_{\mathsf{I}}(\tau)(v) \textit{ is defined, and } T_{\mathsf{I}}(\tau)(v) = u \end{array} \right\},$$

$\mathsf{S}_v = \mathsf{S} \cap (\mathsf{U}_v \times \mathcal{S})$,
$\mathsf{A}_v = \mathsf{A} \cap (\mathcal{A} \times (\mathsf{U}_v \times \mathsf{U}_v))$, *and*
$\mathsf{I}_v = \langle \mathsf{U}_v, \mathsf{S}_v, \mathsf{A}_v \rangle$.

A Pollard feature structure is simply a pair of an entity and an interpretation such that the interpretation is generated by the entity:

Definition 34 *For each SRL signature* Σ, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$, *for each* $v \in \mathsf{U}$,

$\langle v, \mathsf{I} \rangle$ *is a* ***Pollard feature structure*** *iff*

I *is the* Σ *interpretation generated by* v *in* I.

A Pollard feature structure determined by v in I is what I have called a configuration of entities under v in I:

Definition 35 *For each SRL signature* Σ, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$, *for each* $v \in \mathsf{U}$,

$\langle v, \mathsf{I}_v \rangle$ *is the* ***Pollard feature structure determined by*** v ***in*** I *iff*

I_v *is the* Σ *interpretation generated by* v *in* I.

In Section 2.2.2.3 we have seen that for every SRL signature Σ, for every Σ interpretation I, and for each $u \in \mathsf{I}$, exactly one 94 concrete feature structure with root node u is in the set of 94 concrete feature structures determined by I, and it corresponds to the Pollard feature structure determined by u in I in a natural way. It is because of this correspondence that Pollard 1999 chooses Pollard feature structures as its representation of concrete feature structures or tokens of linguistic entities in a natural language instead of a standard definition.

Pollard Abstract Feature Structures Pollard abstract feature structures stand in a similar relation to Pollard feature structures as standard abstract feature structures do to standard concrete feature structures. That means that the main idea remains to switch from nodes as concrete objects to an abstract representation of nodes as equivalence classes. The difference to the standard definition is that now the nodes that must be represented as equivalence classes are actually entities in an interpretation of an SRL signature. What is therefore needed is a canonical representation of entities in an interpretation. However, for a canonical representation of the entities in Pollard abstract feature structures, the only interpretations of interest are those whose entities are reachable by the interpretation of some term on a common ancestor, because the entities in a Pollard feature structure, $\langle v, \mathsf{I} \rangle$, are all components of v. For that reason, a variant of the familiar construction of an equivalence relation over paths can be applied: The entities in the interpretation of a feature structure generated by its root entity are identified with equivalence classes of terms. In preparation of this, Definition 36 introduces an appropriate equivalence relation over terms for each SRL signature.

Definition 36 *For each SRL signature* $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{F} \rangle$, ν *is a* ***node abstraction in*** Σ *iff*

ν *is a pair* $\langle \beta, \varrho \rangle$,

$\beta \subseteq \mathcal{T}^{\Sigma}$,

$: \in \beta$,

for each $\tau \in \mathcal{T}^{\Sigma}$, *for each* $\varphi \in \mathcal{A}$, *if* $\tau\varphi \in \beta$ *then* $\tau \in \beta$,

ϱ *is an equivalence relation over* β,

for each $\tau_1 \in \mathcal{T}^{\Sigma}$, *for each* $\tau_2 \in \mathcal{T}^{\Sigma}$, *for each* $\varphi \in \mathcal{A}$,

if $\tau_1\varphi \in \beta$ *and* $\langle \tau_1, \tau_2 \rangle \in \varrho$ *then* $\langle \tau_1\varphi, \tau_2\varphi \rangle \in \varrho$.

I call ϱ the *re-entrancy relation in the node abstraction in* Σ. A node abstraction in Σ is a structure that can easily be used as a canonical representation of entities in an interpretation generated by an entity in the interpretation. The element that generates the interpretation will naturally be represented by the equivalence class $|:|_{\varrho}$. Therefore, ':' must always be in the carrier, β. Moreover, β is a prefix closed set of terms, because each prefix of a term with a nonempty denotation at an entity in an interpretation also has a nonempty denotation at that entity in the interpretation. The canonical representation of entities must therefore contain entities that are values of all prefixes of a term that is defined on the canonical root entity, $|:|_{\varrho}$, and they will be represented by the equivalence class of the node abstraction that contains that prefix. Finally, ϱ is a right invariant equivalence relation, because if two Σ terms τ_1 and τ_2 are defined at an entity u in an interpretation and yield the same value then for any Σ path π, $\tau_1\pi$ is defined on u exactly if $\tau_2\pi$ is defined on u, and if they are defined they yield the same value in the interpretation. The equivalence classes of the node abstraction must mirror this fact, i.e., $\tau_1\pi$ and $\tau_2\pi$ must be in the same equivalence class representing an entity if the interpretation of τ_1 and τ_2 at $|:|_\varrho$ yield the same value; right invariance guarantees that property.

A node abstraction in Σ is not related to the appropriateness function in Σ yet, and no species are assigned to the equivalence classes of terms in the node abstraction. The intended intuitive correlation between a node abstraction and appropriateness is achieved in the definition of Pollard abstract feature structures by tying the structure of the interpretation of Pollard abstract feature structures in a clever way to the node abstraction whose equivalence classes represent the entities in the interpretation in the abstract feature structure:

Definition 37 *For each SRL signature $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{F} \rangle$, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$, for each $v \in \mathsf{U}$,*

$\langle v, \mathsf{I} \rangle$ *is a* ***Pollard abstract feature structure under*** Σ *iff*

$\langle v, \mathsf{I} \rangle$ is a Pollard feature structure,
for some node abstraction in Σ, $\langle \beta, \varrho \rangle$,

$\mathsf{U} = Quo(\langle \beta, \varrho \rangle)$,
$v = |:|_\varrho$,
for each $v_1 \in Quo(\langle \beta, \varrho \rangle)$, for each $v_2 \in Quo(\langle \beta, \varrho \rangle)$, for each $\varphi \in \mathcal{A}$,

$\mathsf{A}(\varphi)(v_1) = v_2$ *iff*
for some $\tau \in \beta$, $v_1 = |\tau|_\varrho$, and $v_2 = |\tau\varphi|_\varrho$.

Note that by construction, the collection of all Pollard abstract feature structures under a fixed SRL signature Σ is a set. I write $\mathbb{AFS}_{\mathrm{P}}^{\Sigma}$ for the set of Pollard abstract feature structures under Σ. If $\langle v, \mathsf{I} \rangle$ is a Pollard abstract feature structure, I will call v its *root entity*.

Since each Pollard abstract feature structure, $\langle v, \mathsf{I} \rangle$, under an SRL signature Σ is also a Pollard feature structure, its interpretation, I, is generated by its root entity, v. The entities in the interpretation I are the equivalence classes of a node abstraction in Σ, and the root entity, v, is the equivalence class containing the colon. The crucial trick is how the attribute interpretation function, A, of the Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$ of a Pollard abstract feature structure under Σ is related to the equivalence classes of terms that are its entities: An attribute φ is defined on an entity v_1, yielding v_2 as value, exactly if the equivalence class v_1 contains a term τ_1 that is a prefix of a term τ_2 in v_2, and $\tau_1\varphi = \tau_2$. It follows that if an entity v_n can be reached by interpreting the Σ term τ on the root entity then v_n is the equivalence class $|\tau|_\varrho$. Since I is a Σ interpretation, appropriateness follows by definition. More specifically, for each $\varphi \in \mathcal{A}$, for each $|\tau|_\varrho \in \mathsf{U}$, $\mathsf{A}(\varphi)(|\tau|_\varrho)$ is defined if and only if $\mathcal{F}\langle \mathsf{S}(|\tau|_\varrho), \varphi \rangle \neq \emptyset$, and if $\mathsf{A}(\varphi)(|\tau|_\varrho)$ is defined then $\mathsf{S}(|\tau\varphi|_\varrho) \in \mathcal{F}\langle \mathsf{S}(|\tau|_\varrho), \varphi \rangle$.

Since Pollard abstract feature structures have a canonical representation of their entities as equivalence classes of the terms whose interpretation on the root entity yields the respective entities, each one of them is a canonical representation of the Pollard feature structures that are isomorphic to it. Proposition 4 confirms the existence of a unique isomorphic Pollard abstract feature structure for each Pollard feature structure.

Proposition 4 *For each SRL signature Σ, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$, for each $v \in \mathsf{U}$, if $\langle v, \mathsf{I} \rangle$ is a Pollard feature structure then there exists exactly one $\langle v', \mathsf{I}' \rangle \in \mathbb{AFS}^\Sigma_{\mathrm{P}}$, such that for each $\delta \in \mathcal{D}^\Sigma$,*

$$v \in D_{\mathsf{I}}(\delta) \text{ iff } v' \in D_{\mathsf{I}'}(\delta).$$

With previous results, the indiscernibility of the root entities of a Pollard feature structure and a Pollard abstract feature structure entails that the two feature structures mutually simulate each other. Since their interpretations are generated by their root entities, it follows that they are isomorphic.

Exploiting the canonical representations of entities, it is in fact not hard to define an abstraction function that maps each Pollard feature structure to the isomorphic Pollard abstract feature structure. This abstraction function is the counterpart of the abstraction functions from various kinds of concrete feature structures to their corresponding abstract feature structures in the previous sections. Since the three ways of obtaining the strong generative capacity of a grammar that Pollard 1999 presents do not involve Pollard feature structures that are not Pollard abstract feature structures, I omit it here.

Pollard abstract feature structures have the properties that express the intuitions of Pollard 1999 about the strong generative capacity of a grammar.

No two distinct Pollard abstract feature structures are isomorphic, but they can be isomorphic to language tokens, if the latter are construed as pairs of an entity, v, and the interpretation, I_v, generated by v in the natural language, I. The strong generative capacity of a grammar will be characterized as the set of Pollard abstract feature structures that the grammar generates, where the notion of generating a set of abstract feature structures corresponds to the notion of (abstract) feature structure admission implicit in Pollard and Sag 1994.

Strong Generative Capacity Pollard 1999 presents three equivalent definitions of the strong generative capacity of a grammar. All of them use the by now familiar relationships between the models of a grammar, the class of exhaustive models of a grammar, and the set of abstract feature structures that can be abstracted from them. Since Pollard uses canonical, non-standard concrete feature structures as abstract feature structures, there is no notion of the set of abstract feature structures admitted by a grammar that is defined independently of the models of the grammar. On the other hand, since an interpretation generated by the root entity is part of every Pollard abstract feature structure, the third definition of the strong generative capacity of a grammar will offer a new way of deriving the intended set of abstract feature structures. The first two definitions of the SGC of a grammar are similar to constructions that were discussed in Section 2.2.2.3.

For the first definition of the strong generative capacity of a grammar, Pollard defines an abstraction function, Abst_{I}. For each entity v in the interpretation I, it contains pairs of v and Pollard abstract feature structures whose root entities are indiscernible from v:

Definition 38 *For each SRL signature Σ, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$,*

> Abst_{I} *is the binary relation on $\mathsf{U} \times \mathbb{AFS}^{\Sigma}_{\text{P}}$ such that, for each $v \in \mathsf{U}$, for each $\langle v', \mathsf{I}' \rangle \in \mathbb{AFS}^{\Sigma}_{\text{P}}$, $\langle v, \langle v', \mathsf{I}' \rangle \rangle \in \text{Abst}_{\mathsf{I}}$ iff*
>
> > *for the Pollard feature structure $\langle v, \mathsf{I}_v \rangle$ determined by v in I, for each $\delta \in \mathcal{D}^{\Sigma}$, $v \in D_{\mathsf{I}_v}(\delta)$ iff $v' \in D_{\mathsf{I}'}(\delta)$.*

Since each entity, v, in an interpretation, I, determines exactly one Pollard feature structure, $\langle v, \mathsf{I}_v \rangle$, in the interpretation I, it follows with PROPOSITION 4 that each Abst_{I} is functional:

Proposition 5 *For each SRL signature Σ, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$,* $\mathrm{Abst}_{\mathsf{I}}$ *is a total function from* U *to* $\mathbb{AFS}^{\Sigma}_{\mathrm{P}}$.

$\mathrm{Abst}_{\mathsf{I}}$ directly corresponds to King's abstraction function from the set of entities in an interpretation to abstract feature structures, and the same properties hold. Entities that are indiscernible in I abstract to the same Pollard abstract feature structure that is isomorphic to them. If v is an entity in an interpretation I, then I call $\mathrm{Abst}_{\mathsf{I}}(v)$ its *abstraction*. The strong generative capacity of a grammar is supposed to contain one isomorphic member to each utterance token in the natural language that is judged grammatical. King chooses exhaustive models for the explanation of the meaning of a grammar, because an exhaustive model contains indiscernible counterparts to all entities in all models of the grammar. Correspondingly, the SGC of a grammar must contain each Pollard abstract feature structure that can be abstracted from some entity in some model of the grammar. This leads to Pollard's first definition:

Definition 39 (SGC 1) *For each SRL signature Σ,* SGC *is the total function from the set of grammars to Pow($\mathbb{AFS}^{\Sigma}_{\mathrm{P}}$) such that, for each $\theta \subseteq \mathcal{D}^{\Sigma}$,*

$$\mathsf{SGC}(\langle \Sigma, \theta \rangle) = \left\{ \alpha \in \mathbb{AFS}^{\Sigma}_{\mathrm{P}} \;\middle|\; \begin{array}{l} \textit{for some } \langle \Sigma, \theta \rangle \textit{ model } \mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle, \\ \textit{for some } v \in \mathsf{U}, \\ \alpha = \mathrm{Abst}_{\mathsf{I}}(v) \end{array} \right\}.$$

For each grammar Γ, Pollard calls the set $\mathsf{SGC}(\Gamma)$ the *strong generative capacity of* Γ.

Pollard's first definition of the SGC does not refer to exhaustive models, although looking at every possible model of the grammar is ultimately motivated by the concept of exhaustive models. However, exhaustive models provide an obvious alternative to obtaining the SGC of a grammar by taking the abstract feature structures that can be abstracted from the entities in all models. It is clearly sufficient to consider only one model if that model is big enough in the sense of containing SRL congruent copies of all possible configurations under entities in all models, and to take the set of abstract feature structures that can be abstracted from the entities in such a model. Choosing this way of defining the SGC presupposes a definition of the class of exhaustive models of a grammar. Pollard 1999 presents a definition that is different from the two previous characterizations of exhaustive models in

Definition 28 and Corollary 1, because it exploits the existence of a canonical Pollard feature structure for each entity v in an interpretation, namely the existence of $\mathrm{Abst}_{\mathsf{I}}(v)$:

Definition 40 *For each SRL signature Σ, for each $\theta \subseteq \mathcal{D}^{\Sigma}$,*

> $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$ *is an* ***exhaustive*** $\langle \Sigma, \theta \rangle$ ***model*** *iff*
>
> I *is a $\langle \Sigma, \theta \rangle$ model, and for each $\langle \Sigma, \theta \rangle$ model $\mathsf{I}' = \langle \mathsf{U}', \mathsf{S}', \mathsf{A}' \rangle$, for each $v' \in \mathsf{U}'$, there exists a $v \in \mathsf{U}$ such that* $\mathrm{Abst}_{\mathsf{I}}(v) = \mathrm{Abst}_{\mathsf{I}'}(v')$.

Pollard's definition of the exhaustive models of a grammar Γ is closely related to their definition in terms of simulation. In his definition, an exhaustive Γ model is required to contain an SRL congruent Pollard feature structure for all possible Pollard feature structures in all Γ models, because he demands that each abstraction of each entity in each model be also an abstraction of an entity in the exhaustive model. But that is equivalent to saying that a Γ model is an exhaustive Γ model if and only if it simulates all other Γ models.

The function that is defined next maps each Σ interpretation I to the smallest set of Pollard abstract feature structures under Σ that, for each Pollard feature structure determined by some v in I, contains an element that is indiscernible from $\langle v, \mathsf{I}_v \rangle$:

Definition 41 *For each SRL signature Σ, ABST is the large total function from the class of Σ interpretations to $Pow(\mathbb{AFS}^{\Sigma}_{\mathrm{P}})$ such that, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle$,*

$$\mathsf{ABST}(\mathsf{I}) = \left\{ \alpha \in \mathbb{AFS}^{\Sigma}_{\mathrm{P}} \;\middle|\; \begin{array}{l} \textit{for some } v \in \mathsf{U}, \\ \alpha = \mathrm{Abst}_{\mathsf{I}}(v) \end{array} \right\}.$$

For each Σ interpretation I, I call $\mathsf{ABST}(\mathsf{I})$ the *set of Pollard abstract feature structures determined by* I. The function ABST leads to the second definition of the strong generative capacity of a grammar:

Definition 42 (SGC 2) *For each SRL signature Σ, SGC is the total function from the set of grammars to $Pow(\mathbb{AFS}^{\Sigma}_{\mathrm{P}})$ such that, for each $\theta \subseteq \mathcal{D}^{\Sigma}$,*

$$\mathsf{SGC}(\langle \Sigma, \theta \rangle) = \left\{ \alpha \in \mathbb{AFS}^{\Sigma}_{\mathrm{P}} \;\middle|\; \begin{array}{l} \textit{for each (equivalently some)} \\ \textit{exhaustive } \langle \Sigma, \theta \rangle \textit{ model } \mathsf{I}, \\ \alpha \in \mathsf{ABST}(\mathsf{I}) \end{array} \right\}.$$

The third and final way of Pollard 1999 of defining the strong generative capacity of a grammar exploits the fact that a Pollard abstract feature structure, in contrast to the Moshier representation of abstract feature structures, contains an interpretation of a signature. In particular, it contains the interpretation with canonical representations of the elements of the universe that is generated by its root entity. Pollard says that a grammar Γ *generates* a Pollard abstract feature structure exactly if its interpretation is a Γ model:

Definition 43 *For each SRL signature Σ, for each $\theta \subseteq \mathcal{D}^{\Sigma}$, for each Σ interpretation I, for each $v \in \mathsf{U}$, $\langle \Sigma, \theta \rangle$* ***generates*** *$\langle v, \mathsf{I} \rangle$ iff*

$\langle v, \mathsf{I} \rangle \in \mathbb{AFS}_{\mathrm{P}}^{\Sigma}$, and

I is a $\langle \Sigma, \theta \rangle$ model.

This leads to the last characterization of the SGC of a grammar as the set of all Pollard abstract feature structures whose interpretation is a model of a grammar Γ:

Definition 44 (SGC 3) *For each SRL signature Σ, SGC is the total function from the set of grammars to $Pow(\mathbb{AFS}_{\mathrm{P}}^{\Sigma})$ such that, for each $\theta \subseteq \mathcal{D}^{\Sigma}$,*

$$\mathsf{SGC}(\langle \Sigma, \theta \rangle) = \left\{ \alpha \in \mathbb{AFS}_{\mathrm{P}}^{\Sigma} \,\middle|\, \langle \Sigma, \theta \rangle \text{ generates } \alpha \right\}.$$

Just as standard definitions of abstract feature structure admission, **SGC 3** strictly limits its perspective to mathematical representations of linguistic entities. The relationship between the mathematical representations and the empirical domain of observable phenomena is left outside the formal framework. This architecture is fully in line with Pollard's view of the domain of scientific theories in general and of HPSG 94 grammars in particular: They do not talk directly about empirical phenomena, but about mathematical structures that are conventionally understood to correspond to (certain aspects of) empirical phenomena. With this interpretation of scientific theories, Pollard is in deliberate opposition to the realistic interpretation of King, who rejects the representational view of scientific theories of Pollard and holds that "the [empirical] domain is itself a mathematical structure that directly interprets the theory" (King, 1999, p. 310). For King, the natural language, $\mathcal{L}$, construed as a system of tokens, is the intended model of a grammar. $\mathcal{L}$ is in the class of exhaustive models of the grammar.

The philosophical incompatibility of the two positions means that the central concept of each position—the exhaustive models of a grammar and the strong generative capacity of a grammar, respectively—is conceptually meaningless to a proponent of the opposite philosophical persuasion. Ignoring the fundamental incompatibility at the conceptual level of the two positions, we can nevertheless show how they are mathematically tied to each other and how we can technically switch from one perspective to the other. The three definitions of the strong generative capacity of grammars are a good example of how different approaches to the same construct provide different insights into its properties. **SGC 1** considers every possible model of a grammar. According to the realistic perspective, the class of models of the grammar includes the model, $\mathcal{L}$, that is the natural language as well as every other conceivable model, and $\mathcal{L}$ simulates all of them. The relationship of **SGC 2** to the realistic perspective is given by the observation that the SGC of a grammar can be determined by taking the particular exhaustive model of the grammar that is the natural language and abstracting the intended set of Pollard abstract feature structures from it. Since **SGC 1** and **SGC 2** can thus be reinterpreted as referring to the model that is the natural language, the perspectives offered by them are more obviously linked to King's view of the meaning of HPSG 94 grammars than the perspective of **SGC 3**.[58] Since all three definitions are equivalent, it is of course also possible to relate **SGC 3** to a realistic view of the meaning of HPSG 94 grammars, just as it is possible to link feature structure admission to exhaustive models, but the relation is less direct than in the case of **SGC 1** and **SGC 2**. PROPOSITION 6 establishes a connection between the Pollard abstract feature structures generated by a grammar and the set of Pollard abstract feature structures determined by a model of the grammar.

Proposition 6 *For each SRL signature Σ, for each grammar $\langle \Sigma, \theta \rangle$, for each $\langle v, \mathsf{I} \rangle \in \mathbb{AFS}^{\Sigma}_{\mathrm{P}}$,*

> $\langle v, \mathsf{I} \rangle$ *is generated by* $\langle \Sigma, \theta \rangle$ *iff*
>
> *there exists a* Σ *model* I' *such that* $\langle v, \mathsf{I} \rangle \in \mathsf{ABST}(\mathsf{I}')$.

This result allows a proponent of the realistic interpretation of grammars to perceive the set of Pollard abstract feature structures generated by a grammar

[58]The indicated reinterpretation is, of course, against the express intention of Pollard, just as for King it is completely superfluous to extract the strong generative capacity from a natural language.

as the set of Pollard abstract feature structures abstracted from the natural language model.

Pollard 1999 does not make any claims about the ontological status of the strong generative capacity of an HPSG 94 grammar. The more modest aim of the paper is to give the intuitions behind the idea of the strong generative capacity of a grammar mathematical substance in object-based HPSG in order to relate the HPSG 94 framework to other formalisms of generative grammar and to their ideas about the structures that a grammar generates. In doing so, Pollard 1999 adds another interesting piece to the understanding of the role that different mathematical constructs play in determining the meaning of HPSG 94 grammars.

Regarding the terminology of Pollard 1999, it is important to be careful not to confuse Pollard feature structures with the standard, automata theoretic definition of concrete feature structures. Although it is easy to switch from an entity in an interpretation to the concrete feature structure that represents the entity, this change of perspective might have philosophical repercussions: A concrete feature structure is a complex mathematical object, but the ontological status of entities in an interpretation is open; they can be conceived as entities in the empirical domain. Pollard feature structures are entities in an interpretation, which leaves the possibility of a realistic perspective. The difference between Pollard abstract feature structures and standard abstract feature structures is also of technical importance. Standard abstract feature structures represent equivalence classes of tokens. They are not tokens themselves, and they are not isomorphic with tokens. They cannot be construed as entities in an interpretation, because they cannot model the situation in which two entities in the interpretation have congruent component structures but are still distinct tokens in the interpretation. Pollard abstract feature structures, however, are just a special kind of Pollard feature structures, namely Pollard feature structures whose entities are particular equivalence classes of terms. They are isomorphic with possible tokens. Therefore, they can well be understood as mathematical representations of idealizations of possible linguistic tokens, but there is only one mathematical representation for each collection of isomorphic tokens. These subtle differences between Pollard's feature structures and standard feature structures as well as the differences between the various different kinds of abstract and concrete feature structures that occur in the context of HPSG should caution linguists against an unspecific use of the term "feature structure."

Pollard abstract feature structures can serve yet another purpose. THEOREM 1 asserts the existence of an exhaustive model for each grammar. A natural way to prove this theorem is to construct a model for each grammar, and to show that it is exhaustive. King's (unpublished) proof uses feature structure admission and 94 abstract feature structures to construct the exhaustive models.[59] Pollard 1999, p. 295, briefly indicates that Pollard abstract feature structures and the set of Pollard abstract feature structures generated by a grammar can be used to the same end. In DEFINITION 45, I spell out this idea.

The entities in the interpretation are constructed as complex objects. They are pairs of a Pollard abstract feature structure generated by the grammar and an entity in the feature structure. Intuitively, these complex objects are Pollard abstract feature structures with distinguished nodes. The set of entities in the universe of the exhaustive models can be understood as all Pollard abstract feature structures that the grammar generates with one distinguished node, where each feature structure occurs once with each node as distinguished node. The sort assignment is obtained by assigning each entity the sort that the Pollard abstract feature structure in it assigns to the distinguished node. Similarly, the attribute interpretation mimics the attribute interpretations in the abstract feature structures. An attribute φ is defined on an entity u exactly if it is defined on the distinguished node v in the interpretation of the Pollard abstract feature structure in the entity u, and if φ is defined on an entity u then the value is the entity u' that contains the same Pollard abstract feature structure as u and the distinguished node that is the value of φ on v in the interpretation of the Pollard abstract feature structure in question. In effect, the interpretation that is obtained by this construction contains isomorphic copies of all interpretations in the Pollard abstract feature structures generated by the grammar.

Definition 45 *For each SRL signature $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{F} \rangle$, for each $\theta \subseteq \mathcal{D}^{\Sigma}$,*

$$\mathbf{U}^{\theta}_{\Sigma} = \left\{ \begin{array}{l|l} \langle v', \langle v, \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle \rangle \rangle & \langle \Sigma, \theta \rangle \textit{ generates } \langle v, \langle \mathsf{U}, \mathsf{S}, \mathsf{A} \rangle \rangle, \textit{ and} \\ \in \mathit{Pow}(\mathcal{T}^{\Sigma}) \times \mathbb{AFS}^{\Sigma}_{\mathrm{P}} & v' \in \mathsf{U} \end{array} \right\},$$

[59]We will see the details of King's construction below as part of the existence proof of the extended formalism.

$$\mathbf{S}^{\theta}_{\Sigma} = \left\{ \langle v', \sigma \rangle \in \mathbf{U}^{\theta}_{\Sigma} \times \mathcal{S} \;\middle|\; \begin{array}{l} \textit{for some } \langle v, \langle Quo(\langle \beta, \rho \rangle), \mathsf{S}, \mathsf{A} \rangle \rangle \textit{ generated} \\ \textit{by } \langle \Sigma, \theta \rangle, \\ \textit{for some } \tau \in \beta, \\ v' = \langle |\tau|_{\rho}, \langle v, \langle Quo(\langle \beta, \rho \rangle), \mathsf{S}, \mathsf{A} \rangle \rangle \rangle, \textit{ and} \\ \sigma = \mathsf{S}(|\tau|_{\rho}) \end{array} \right\},$$

$\mathbf{A}^{\theta}_{\Sigma}$ *is the total function from* $\mathcal{A}$ *to the power set of* $\mathbf{U}^{\theta}_{\Sigma} \times \mathbf{U}^{\theta}_{\Sigma}$ *such that, for each* $\varphi \in \mathcal{A}$,

$$\mathbf{A}^{\theta}_{\Sigma}(\varphi) = \left\{ \begin{array}{l} \langle v', v'' \rangle \\ \in \mathbf{U}^{\theta}_{\Sigma} \times \mathbf{U}^{\theta}_{\Sigma} \end{array} \;\middle|\; \begin{array}{l} \textit{for some } \langle v, \langle Quo(\langle \beta, \rho \rangle), \mathsf{S}, \mathsf{A} \rangle \rangle \textit{ generated} \\ \textit{by } \langle \Sigma, \theta \rangle, \\ \textit{for some } \tau \in \beta, \\ v' = \langle |\tau|_{\rho}, \langle v, \langle Quo(\langle \beta, \rho \rangle), \mathsf{S}, \mathsf{A} \rangle \rangle \rangle, \textit{ and} \\ v'' = \langle |\tau\varphi|_{\rho}, \langle v, \langle Quo(\langle \beta, \rho \rangle), \mathsf{S}, \mathsf{A} \rangle \rangle \rangle \end{array} \right\},$$

$\mathbf{I}^{\theta}_{\Sigma} = \left\langle \mathbf{U}^{\theta}_{\Sigma}, \mathbf{S}^{\theta}_{\Sigma}, \mathbf{A}^{\theta}_{\Sigma} \right\rangle$.

A standard proof shows that $\mathbf{I}^{\theta}_{\Sigma}$ is a Σ interpretation as PROPOSITION 7 claims:

Proposition 7 *For each SRL signature* $\Sigma = \langle \mathcal{S}, \mathcal{A}, \mathcal{F} \rangle$, *for each* $\theta \subseteq \mathcal{D}^{\Sigma}$,

> $\mathbf{U}^{\theta}_{\Sigma}$ *is a set,*
>
> $\mathbf{S}^{\theta}_{\Sigma}$ *is a total function from* $\mathbf{U}^{\theta}_{\Sigma}$ *to* $\mathcal{S}$,
>
> $\mathbf{A}^{\theta}_{\Sigma}$ *is a total function from* $\mathcal{A}$ *to the set of partial functions from* $\mathbf{U}^{\theta}_{\Sigma}$ *to* $\mathbf{U}^{\theta}_{\Sigma}$, *and*
>
> $\mathbf{I}^{\theta}_{\Sigma}$ *is a* Σ *interpretation.*

Moreover, it can be shown that for each SRL signature Σ, for each set of Σ descriptions θ, for each $\mathbf{I}^{\theta}_{\Sigma}$, for each $\langle \Sigma, \theta \rangle$ model I', and for each $v' \in \mathsf{I}'$, there exists a $v \in \mathbf{I}^{\theta}_{\Sigma}$ such that $\mathrm{Abst}_{\mathbf{I}^{\theta}_{\Sigma}}(v) = \mathrm{Abst}_{\mathsf{I}'}(v')$. In other words, $\mathbf{I}^{\theta}_{\Sigma}$ is not only a Σ interpretation, it is an exhaustive $\langle \Sigma, \theta \rangle$ model:

Proposition 8 *For each SRL signature* Σ, *for each* $\theta \subseteq \mathcal{D}^{\Sigma}$,

> $\mathbf{I}^{\theta}_{\Sigma}$ *is an exhaustive* $\langle \Sigma, \theta \rangle$ *model.*

Intuitively, this is not hard to see: We know that the set of Pollard abstract feature structures generated by a grammar is the same as the set of Pollard abstract feature structures that we obtain by collecting all the abstractions of all entities in all models of the grammar, because the first is the strong generative capacity of the grammar according to **SGC 3**, and the second is the strong generative capacity of the grammar according to the equivalent definition **SGC 1**. But we have seen that $\mathbf{I}^{\theta}_{\Sigma}$ copies over all interpretations of all Pollard abstract feature structures in the entities in $\mathbf{U}^{\theta}_{\Sigma}$ that were obtained that way. It immediately follows that we get all possible abstractions of all entities in all possible models of the grammar back when we take the set of abstractions of all entities in $\mathbf{I}^{\theta}_{\Sigma}$. Therefore, $\mathbf{I}^{\theta}_{\Sigma}$ is an exhaustive model.

2.2.3 Problems: Missing Concepts[60]

In Section 2.2.2, I presented a mathematical formalism, SRL, and I have summarized how King 1999 and Pollard 1999 express grammars, the meaning of grammars, and the notions of linguistic tokens and linguistic types within SRL. The accounts of tokens and types presented in both papers differ deliberately from each other and from the original, informal account of Pollard and Sag 1994, which King 1996 formalized in SRL. We have seen that these differences are due to different views of what linguistic tokens and linguistic types are, and how they are mathematically reconstructed in the formalism. The main differences, however, concern the opinions about the ontological status of the mathematical structures that a grammar delineates, and the relationship between these structures and the natural languages that are the empirical domain of grammars. What King's account and Pollard's account hold in common is that they are expressed in the SRL formalism. The common underlying assumption is that SRL is an adequate formal foundation of HPSG 94, and that HPSG 94 grammars can be made precise and can be expressed as SRL grammars. In the words of King 1999, p. 335: "In summation, a HPSG grammar can be expressed in very large part, if not in entirety, as a SRL grammar, where a SRL **grammar** is a pair (Σ, θ) such that Σ is a [SRL] signature and $\theta \subseteq \mathcal{D}^{\Sigma}$."

In the present section I will challenge this view. While I accept that all parts of the SRL formalism are necessary in a formalization of HPSG 94, and while I think it is desirable to maintain the explanations of the meaning of

[60]This section is partially based on work previously published in Richter et al. 1999.

HPSG 94 grammars of King 1999 and Pollard 1999 in an adequate formalism for HPSG 94, I will claim that the expressive means of the formal language of SRL are not sufficient in order to express all aspects of typical principles in actual HPSG 94 grammars concisely and naturally. In many cases, a specification of the grammatical principles in SRL is extremely cumbersome, does not correspond to the way in which the linguists think about their principles, and can only deliver an approximation to the actually intended principles. While such approximations might be sufficient in computational implementations, they are unsatisfactory to the point of being unacceptable in a rigorous, declarative account of what the principles of HPSG 94 grammars mean in the eyes of the theoretical linguist.

Besides the attribute symbols and the sort symbols, Pollard and Sag 1994 presupposes a set of relation symbols with appropriate denotations in the description language: "AVM descriptions will also employ functional or relational symbols such as append, union ($\cup$), and $\neq$, all of which we consider to be necessary in a linguistically adequate description language" (Pollard and Sag, 1994, p. 21). In fact, relations are used much more prolifically in Pollard and Sag 1994 and elsewhere in the HPSG literature than the quotation may suggest. Some are explicitly referred to as such, e.g., the `o-command` and `local-o-command` relations of the BINDING THEORY, or the `append` and the `member` relations in a considerable number of principles and descriptions, while many others are implicitly required by the natural language descriptions of principles or by abbreviatory conventions in AVM diagrams. The SUBCATEGORIZATION PRINCIPLE (Pollard and Sag, 1994, p. 399), for example, says:

> "In a headed phrase, the list value of DAUGHTERS | HEAD-DAUGHTER | SYNSEM | LOCAL | CATEGORY | SUBCAT is the concatenation of the list value of SYNSEM | LOCAL | CATEGORY | SUBCAT with the list consisting of the SYNSEM values (in order) of the elements of the list value of DAUGHTERS | COMPLEMENT-DAUGHTERS."

The consequent of this implication requires the use of at least one relation that (i) determines the list of *synsem* entities that results from taking the SYNSEM values of the elements of the list of complement daughters and putting them in their original relative order on a new list, and (ii) appends that list to the SUBCAT list of the syntactic mother node to obtain the SUBCAT list of the head daughter of headed phrases.[61] The recursive nature of

[61]The procedural flavor of the description of the effect of the relation is meant metaphor-

lists of arbitrary length requires the use of the relation.

Since SRL does not contain relation symbols in its formal language, we face two options: Either SRL already includes some other devices that make it possible to express the necessary relations, and we need to explain how these devices can be systematically used to express the relations that occur in HPSG 94 grammars, or we need to give up on the idea that SRL is a fully adequate formalism to the task at hand, and, minimally, we must find an appropriate extension of SRL that keeps the desirable properties of SRL and adds relations. Of course, Pollard 1999 and King 1999 are aware of the problem of missing relations in SRL; and as a solution, they suggest the so-called junk slot encoding of relations, a technique that reifies relations as recursive embeddings of special entities under special attributes. The invention of the junk slot technique is usually attributed to Ait-Kaci 1984, whereas the term "junk slot" was coined by Carl Pollard. If the junk slot method, which I will investigate presently in more detail, were generally successful, it would solve the problem of relations with the conservative, restricted descriptive devices of SRL. Ideally, relation symbols in grammars would simply be a metasyntactic notation for junk slots. Relations would then be expressed in SRL by an appropriate translation of the metasyntax of relations into the formal language, but relations would not be explicitly included in the syntax nor in the semantics of the description language. Their treatment would thus resemble King's approach to sort hierarchies and nonmaximal sorts in grammatical descriptions, which King expresses without representing sort hierarchies in SRL signatures.

Junk slots are configurations of entities that are introduced by additional attributes and sorts. The functions and entities in the denotation of these attributes and sorts do not correspond to posited properties of the entities in the empirical domain. They are only needed for technical reasons. They specify the relationship between empirically motivated entities in models of grammars, such as SUBCAT lists and complement daughters. In order to express relations as junk slots in a grammar, the signature and the theory of the grammar must be extended. Species symbols and attributes for the junk slots must be added, and the theory must be augmented with a theory of the proper configurations of the entities in the junk slots that guarantees that in models of the grammar, the junk slots have the intended shape. Further additional attributes connect the linguistically meaningful configurations of

---

ically. The principle and the relation are purely declarative.

entities with the junk slot configurations in the models. The SUBCATEGORIZATION PRINCIPLE is an easy example to illustrate the junk slot method. It can be expressed with the standard `append` relation and an additional, very simple relation that relates any list of signs to the list of the SYNSEM values (in order) of its elements. Following the metaphor of extracting a list of *synsem* entities from a list of signs, I will call that relation `extract-s(yn)s(em)`. Before I specify a possible junk slot extension of the SRL signature that expresses the signature of Pollard and Sag 1994 and state the theory of *append* and *extract-ss* in (19), it might be useful to recall how lists are standardly encoded in HPSG.

Models of grammars contain individual objects and no structured entities such as lists. This means that there are no genuine list entities in the universes of interpretations. Instead, lists are represented as complex configurations of entities. The SRL signature of lists looks as follows:[62]

(18) *elist*

nelist	FIRST	$\top$
	REST	{*elist, nelist*}

The species *elist* denotes atomic entities, i.e., entities on which no attributes are defined. Entities of species *elist* are empty lists. Entities of species *nelist* (nonempty lists) bear two attributes, FIRST and REST, where the value of FIRST is the first element of the list, and the value of REST is its tail. If the tail is the empty list, the list ends. In (18) and in the remainder of this section, I use the symbol '$\top$' as an abbreviation for the set of all species in a given signature. If the list signature (18) is part of a signature $\langle \mathcal{S}, \mathcal{A}, \mathcal{F} \rangle$, then$\top = \mathcal{S}$. That means that lists may contain any kind of entity as their members.[63] In signatures with a sort hierarchy, *elist* and *nelist* usually have *list* as their immediate supersort. In the context of SRL, *list* can be understood as representing the set {*elist, nelist*} (King, 1999, pp. 330–331). Below I will use analogous metasyntactic expressions like *append* and *extract-ss* to denote the obvious sets of species.

[62]For depicting SRL signatures, I use the notational conventions introduced on page 32 for 87 signatures. The fact that there is no sort hierarchy entails that there is no indentation for a subsort relationship.

[63]For the time being, I ignore the issue of parametric lists, which are lists that are specified in the signature to contain only elements of a particular sort. See Section 4.5.2 for discussion.

An SRL signature and a theory of *append* and *extract-ss* are defined in (19a) and (19b), respectively.[64]

(19) a. *f-append* LEFT $\{elist\}$
RIGHT $\{elist,\ nelist\}$
RESULT $\{elist,\ nelist\}$

u-append LEFT $\{nelist\}$
RIGHT $\{elist,\ nelist\}$
RESULT $\{nelist\}$
REC $\{f\text{-}append,\ u\text{-}append\}$

The theory of *append*:

$$:\sim f\text{-}append \rightarrow :\text{RIGHT} \approx :\text{RESULT}$$

$$:\sim u\text{-}append \rightarrow \begin{bmatrix} :\text{LEFT FIRST} \approx :\text{RESULT FIRST} \land \\ :\text{LEFT REST} \approx :\text{REC LEFT} \land \\ :\text{RIGHT} \approx :\text{REC RIGHT} \land \\ :\text{RESULT REST} \approx :\text{REC RESULT} \end{bmatrix}$$

b. *f-extract-ss* SIGNS $\{elist\}$
SYNSEMS $\{elist\}$

u-extract-ss SIGNS $\{nelist\}$
SYNSEMS $\{nelist\}$
REC $\{f\text{-}extract\text{-}ss,\ u\text{-}extract\text{-}ss\}$

The theory of *extract-ss*:

$$:\sim u\text{-}extract\text{-}ss \rightarrow \begin{bmatrix} :\text{SIGNS FIRST SYNSEM} \approx :\text{SYNSEMS FIRST} \land \\ :\text{SIGNS REST} \approx :\text{REC SIGNS} \land \\ :\text{SYNSEMS REST} \approx :\text{REC SYNSEMS} \end{bmatrix}$$

Entities of sort *append* come in two species, as *f(inished)-append* entities with the list-valued attributes LEFT, RIGHT and RESULT, and as *u(nfinished)-append* entities with the same list-valued attributes and with the *append*-valued attribute REC(URSION). The idea behind that distinction becomes immediately clear when we look at the theory of *append* that determines the possible configurations under *append* entities in models of a grammar to which (19a) belongs. In the case of entities of species *f-append*, the value of LEFT is an empty list, and the list value of RIGHT equals the value of RESULT. In other words, if we append a nonempty list to an empty list, we

[64]The idea of the encoding of the append relation in (19a) is due to Ait-Kaci 1984, p. 118.

get the appended nonempty list as result. *u-append* entities contain recursive embeddings of *append* entities under a path of REC attributes whose length correlates with the length of the list value of the LEFT attribute. For each *u-append* entity in a model of the *append* junk slot, the first element of the list value of LEFT equals the first element of the RESULT list, and the tail of the LEFT list, the tail of the RESULT list, and the list value of RIGHT are passed on to the attribute values of the corresponding attributes of the embedded *append* entity. But that means that for the lists that are passed on the append conditions also hold. It is easy to see that, in any model of a grammar containing the specifications in (19a), the RESULT list of the *append* entity is the concatenation of the LEFT list with the RIGHT list if the LEFT list is finite and non-cyclic.

The technique of the *extract-ss* junk slot in (19b) is exactly the same. *f-extract-ss* entities are the base case: The values of the SIGNS attribute and of the SYNSEMS attribute are of species *elist*. An empty list of *synsem* entities is extracted from an empty list of signs. In the recursive case of *u-extract-ss*, the SYNSEM value of the first element of the SIGNS list is the first element of the SYNSEMS list, and the tail of both lists are passed on to the SIGNS and SYNSEMS lists of the *extract-ss* entity that is the value of the REC attribute. Therefore, they must also obey the conditions just described, and they stand in an "extract-ss" relationship.

In models, the attribute values of *append* and *extract-ss* entities stand in the desired relationships, namely in an `append` relationship and in an `extract-ss` relationship. We can exploit these properties in a formalization of the SUBCATEGORIZATION PRINCIPLE if we place *append* and *extract-ss* entities somewhere in configurations under phrases such that we can use one of each for enforcing the intended relationship among two SUBCAT lists and the list of complement daughters in each headed phrase. For example, we can declare two new attributes APPEND and EXTRACT-SS appropriate to *phrase* and let $\mathcal{F}\langle \textit{phrase}, \textsc{append}\rangle = \{\textit{f-append, u-append}\}$ and $\mathcal{F}\langle \textit{phrase}, \textsc{extract-ss}\rangle = \{\textit{f-extract-ss, u-extract-ss}\}$. Given this final extension of the signature, the SUBCATEGORIZATION PRINCIPLE can be formalized as follows:

(20) *The* SUBCATEGORIZATION PRINCIPLE *with junk slots in SRL*

$$\left[:\sim \textit{phrase} \;\wedge \left[\begin{array}{l} :\textsc{dtrs} \sim \textit{head-adj-struc} \vee \\ :\textsc{dtrs} \sim \textit{head-comp-struc} \vee \\ :\textsc{dtrs} \sim \textit{head-filler-struc} \vee \\ :\textsc{dtrs} \sim \textit{head-mark-struc} \end{array} \right] \right] \rightarrow$$

$$\left[\begin{array}{l} :\text{SYNSEM LOC CAT SUBCAT} \approx :\text{APPEND LEFT} \wedge \\ :\text{DTRS H-DTR SYNSEM LOC CAT SUBCAT} \approx :\text{APPEND RESULT} \wedge \\ :\text{DTRS C-DTRS} \approx :\text{EXTRACT-SS SIGNS} \wedge \\ :\text{EXTRACT-SS SYNSEMS} \approx :\text{APPEND RIGHT} \end{array}\right]$$

The integration of the junk slots with the linguistically motivated configurations of entities leads to the intended relationship between the three linguistically relevant lists. (20) shows that it is indeed possible to express the SUBCATEGORIZATION PRINCIPLE in SRL without genuine relations. The price to pay are an extension of the signature of the grammar, and the specification of a theory of the admissible configurations under entities in junk slots. It is clear that in models of linguistic grammars, this method will lead to a huge number of junk entities,[65] and linguists might be uncomfortable with the fact that there is no distinction between the entities and attributes which have linguistic significance and others that serve a purely technical purpose. The ontological status of entities such as *u-append* in models of linguistic grammars is certainly an open question. They are suspicious if one wants to assume that all entities and their configurational properties in the intended linguistic model of a grammar should be of immediate linguistic significance. Technically, the specification of junk slots can be very unwieldy, but in the two examples above it is not very different from what a specification of the corresponding true relations would look like.[66] At first blush, the junk slot method seems to be very flexible, and based on the inspection of a junk slot encoding of the SUBCATEGORIZATION PRINCIPLE and the fact that other principles can be encoded similarly, King 1999, p. 334, speculates that "[b]y means of junk slot encodings and other technical devices, a theory can express most, if not all, of the (Pollard and Sag, 1994) principles."

However, the situation is not as simple as the first experience with the SUBCATEGORIZATION PRINCIPLE might suggest, and specifying junk slots to express particular uses of relations in linguistic principles can become a daunting task. Consider, as another apparently very simple example, the following principle, taken from Abeillé and Godard 1996, that assists in predicting the distribution of *avoir* and *être* in French:

[65]For an example that gives a clear idea about the consequences of using junk slots explicitly in a grammar, see the theory of word order in the German *Mittelfeld* in Richter and Sailer 1995, which is expressed entirely with junk slots.

[66]Cf. the definition of `append` in Section 3.2.3, (38), and of `extract-ss` in Section 4.3.2, (72).

(21) $\begin{bmatrix} \text{VFORM} & \textit{part-passé} \\ \text{S-ARG} & \langle \ldots \textit{aff\_r} \ldots \rangle \end{bmatrix} \longrightarrow \begin{bmatrix} \text{V-AUX} & \textit{etre} \end{bmatrix}$

What Abeillé and Godard meant to say with this principle is that assuming a suitable signature, for each entity in a model of their grammar, if the VFORM value is of sort *part-passé*, and there exists a member of the S-ARG list that is of sort *aff_r*, i.e., which is a reflexive clitic, then the V-AUX value must be *etre*. The notation with dots in the description of the list in the antecedent of the implication is an example of a frequently adopted notational convention that leaves the `member` relation implicit. The crucial formal difference between the SUBCATEGORIZATION PRINCIPLE and Abeillé and Godard's principle is that in the latter, but not in the former, the relation occurs in the scope of negation, because in Abeillé and Godard's principle, the relation is in the antecedent of an implication. Nevertheless, the principle is very easy to understand intuitively, and it may be a surprise that it is far from straightforward to express it with a junk slot.

Formalizing a junk slot for the `member` relation follows the pattern of *append* and *extract-ss*. There are two kinds of *member* entities, *f-member* and *u-member*. *f-member* entities do not have an attribute for recursion, and the entity that is its ELE value is the first element on the list that is the LIST value. In configurations under *u-member* entities, the ELE value and the tail of the LIST list are simply passed on to the recursively embedded *member* entity. It is easy to show that for each *member* entity in a model of a grammar that comprises (22), the ELE value of the *member* entity occurs somewhere on the LIST list if the list is finite and non-cyclic.

(22) *f-member* ELE $\top$
LIST $\{\textit{nelist}\}$

u-member ELE $\top$
LIST $\{\textit{nelist}\}$
REC $\{\textit{f-member, u-member}\}$

The theory of *member*:

$: \sim \textit{f-member} \rightarrow : \text{ELE} \approx : \text{LIST FIRST}$

$: \sim \textit{u-member} \rightarrow : \text{ELE} \approx : \text{REC ELE} \wedge : \text{LIST REST} \approx : \text{REC LIST}$

Although the theory of *member* is clearly an acceptable approximation to the `member` relation, *member* entities cannot formalize the occurrence of a member relation in the scope of negation. In order to illustrate this with

Abeillé and Godard's principle, it is useful to bring the original implication of the form $A \rightarrow B$ into the equivalent disjunctive form $\neg A \vee B$, and to pull the negation into the conjunctive antecedent. This results in a formulation of the principle that is logically equivalent to its original formulation and can be described as follows: For each entity in a model of the grammar: (1) the VFORM attribute is not defined on it, or it is defined and the value is not of sort *part-passé*; or (2) the S-ARG attribute is not defined on the entity; or (3) no entity of sort *aff_r* is a member of the S-ARG list; or (4) the V-AUX attribute is defined on the entity, and its value is of sort *etre*.[67] To be able to use the *member* junk slot in an attempt to formalize the principle, *member* entities must be made available in the relevant configurations of entities. For that purpose, I let a new attribute, MEMBER, be appropriate to the species σ to which VFORM and S-ARG are appropriate in Abeillé and Godard's signature, and I let $\mathcal{F}\langle\sigma, \text{MEMBER}\rangle = \{\textit{f-member, u-member}\}$. A naive specification of principle (21) would then negate the junk slot encoding of the membership relation just as a relational expression would be negated in its place:

$$(23)\quad \begin{bmatrix} \begin{bmatrix} \neg : \text{VFORM} \sim \textit{part-passé} \vee \\ \neg : \text{S-ARG} \approx : \text{S-ARG} \end{bmatrix} \vee \\ \neg \begin{bmatrix} : \text{MEMBER LIST} \approx : \text{S-ARG} \wedge \\ : \text{MEMBER ELE} \sim \textit{aff\_r} \end{bmatrix} \end{bmatrix} \vee \ [: \text{V-AUX} \sim \textit{etre}]$$

The first line in the left column corresponds to condition (1), the second to condition (2), the next disjunct is trying to capture the membership condition (3), and the right column expresses condition (4). The relevant question to ask is if the disjunct of (23) that is supposed to express condition (3) really describes an entity u exactly if no entity of sort *aff_r* is a member of the S-ARG list defined on u. For ease of reference, I will henceforth call that disjunct δ. Firstly, δ describes each entity on which MEMBER or S-ARG are not defined. Notice that in the extension of the signature I have made sure that MEMBER is defined on an entity exactly if S-ARG is defined on it. Saying that an S-ARG list of an entity that does not have an S-ARG value does not have a particular member seems to be a correct statement about the entity. Secondly, δ describes each entity whose MEMBER LIST value does not equal

[67] I tacitly suppose an SRL signature that contains a species, σ, to which VFORM and S-ARG are appropriate. Furthermore, the set of appropriate values for $\langle\sigma, \text{VFORM}\rangle$ comprises *part-passé*, and $\{\textit{elist}, \textit{nelist}\}$ is appropriate for $\langle\sigma, \text{S-ARG}\rangle$.

the S-ARG list. In this case the junk slot says nothing at all about the members of the S-ARG list, and it might have any kind of members, contrary to what condition (3) demands. This is clearly wrong. Thirdly, even if the MEMBER LIST value equals the S-ARG value, an entity in a model can be in the denotation of δ if the second line of δ does not describe it. If MEMBER is defined on the entity, the negation of the second line of δ being true of it implies only that the MEMBER ELE value is not of sort *aff_r*, which means that there is at least one member of the S-ARG list that is not of sort *aff_r*. This effect is due to the fact that the *member* junk slot only asserts that one particular entity, the value of ELE, is a member of a finite (and non-cyclic) list, which is an existential assertion. It cannot express a universal condition on all members of the list. As the formalization stands, the MEMBER LIST value of an entity that δ describes might equal the S-ARG value, and the S-ARG list might yet contain an element of sort *aff_r*, because δ describes the entity if its MEMBER ELE value is not a member of the S-ARG list. In summary, (23) permits configurations under entities in models that (21) wants to exclude.

To put it more generally, negating the *member* junk slot cannot give us the set complement of the set of pairs of entities and lists that are in the (intuitively given) member relation. The assertion that an entity is not a *member* entity has nothing to do with the assertion that a pair consisting of an entity and a list are not an element of the member relation. Since a junk slot expresses a relation as configurations of entities in a model, it is inherently positive. Negating junk slots with the negation of the description language cannot have the intended meaning. The negation of junk slots must be constructed with a negative junk slot, in the present example a *non-member* junk slot:

(24) *f-non-member* ELE $\top$
LIST $\{elist\}$

u-non-member ELE $\top$
LIST $\{nelist\}$
REC $\{$*f-non-member, u-non-member*$\}$

The theory of *non-member*:

$$:\sim u\text{-}non\text{-}member \rightarrow \begin{bmatrix} \neg :\text{ELE} \approx :\text{LIST FIRST} \wedge :\text{ELE} \approx :\text{REC ELE} \wedge \\ :\text{LIST REST} \approx :\text{REC LIST} \end{bmatrix}$$

No element is a member of the empty list (*f-member*); and an element is not a member of a list, if it is not the first element of the list, and it is not a member of the tail of the list (*u-member*). Just as *member* is an acceptable approximation to the member relation, *non-member* is an acceptable approximation to its negation.

We can now try to solve the problem of expressing the principle (21) that we encountered with *member* in (23) by replacing the naive negation of the *member* junk slot with the less naive assertion that there is a relation of non-membership. For that purpose, I assume that a new attribute, NON-MEMBER, is appropriate to the species σ to which VFORM and S-ARG are appropriate in Abeillé and Godard's signature, and I let $\mathcal{F}\langle\sigma, \text{NON-MEMBER}\rangle = \{$*f-non-member, u-non-member*$\}$. The new formulation of principle (21) equals the formulation in (23) except for the disjunct that tries to capture the membership condition (3).

(25) $$\left[\begin{array}{l} \left[\begin{array}{l} \neg : \text{VFORM} \sim \textit{part-passé} \vee \\ \neg : \text{S-ARG} \approx : \text{S-ARG} \end{array}\right] \vee \\ \left[\begin{array}{l} : \text{NON-MEMBER LIST} \approx : \text{S-ARG} \wedge \\ : \text{NON-MEMBER ELE} \sim \textit{aff\_r} \end{array}\right] \end{array}\right] \vee \ [: \text{V-AUX} \sim \textit{etre}]$$

Again, the interesting question to ask is whether the disjunct in (25) that is supposed to express the membership condition (3) expresses the right condition. I call the disjunct that replaces the description δ of (23) δ'. Unfortunately, δ' also describes entities whose S-ARG list contains an entity of sort *aff_r*. δ' describes an entity on which NON-MEMBER is defined and whose NON-MEMBER LIST value equals the S-ARG value, if the particular entity of sort *aff_r* that is the NON-MEMBER ELE value is not a member of the S-ARG list. That does not exclude the situation in which an *aff_r* entity that is not identical to the NON-MEMBER ELE value is on the S-ARG LIST. Therefore, (25) is also not an adequate formalization of principle (21). Although the *non-member* junk slot is an improvement compared to the first attempt with the *member* junk slot, because it expresses a non-membership relation, it can only say that the very entity that is its ELE value is not a member of its LIST value. It is unable to express the universal quantification over all entities of sort *aff_r* that is needed to express the principle.

To express the membership condition (3) correctly with a junk slot, both the negation and the universal quantification over all members of the list must be built in. The highly specialized junk slot *non-aff_r* in (26) does this. In the base case, *f-non-aff_r*, an empty list does not contain an element of sort *aff_r*. In the recursive case, *u-non-aff_r*, a nonempty list contains no element of sort *aff_r* if its first element is not of that sort, and its tail is passed on to the LIST value of the recursively embedded *non-aff_r* entity, which enforces the same conditions on the tail of the list either because the entity is an *f-non-aff_r* entity or because it is a *u-non-aff_r* entity, depending on whether the tail of the original list is empty or not.

(26) *f-non-aff_r* LIST {*elist*}

u-non-aff_r LIST {*nelist*}
REC {*f-non-aff_r, u-non-aff_r* }

The theory of *non-aff_r*:

$: \sim \textit{u-non-aff\_r} \rightarrow \neg : \text{LIST FIRST} \sim \textit{aff\_r} \wedge : \text{LIST REST} \approx : \text{REC LIST}$

The *non-aff_r* junk slot must be integrated with the linguistic part of the signature of Abeillé and Godard's grammar. Analogous to the previous specifications with *member* and *non-member*, I introduce a new attribute, NON-AFF, assume that it is appropriate to exactly the same species σ to which VFORM and S-ARG are appropriate, and I let $\mathcal{F}\langle\sigma, \text{NON-AFF}\rangle = \{$*f-non-aff_r, u-non-aff_r*$\}$. With these extensions to the signature of Abeillé and Godard, I can express the principle (21):

$$(27)\quad \left[\begin{array}{l} \left[\begin{array}{l} \neg : \text{VFORM} \sim \textit{part-passé} \vee \\ \neg : \text{S-ARG} \approx : \text{S-ARG} \end{array}\right] \vee \\ \left[\ : \text{NON-AFF LIST} \approx : \text{S-ARG}\ \right] \end{array}\right] \vee \ [: \text{V-AUX} \sim \textit{etre}]$$

The disjunct of the formalization of (21) that is supposed to express condition (3) does finally express condition (3): For each entity u in a model of the grammar that satisfies it, no entity of sort *aff_r* is a member of the S-ARG list of u.

Even for the very simple principle (21), determining which kind of junk slot is needed is not trivial. Most importantly, the example illustrates that a junk slot encoding of a relational condition might have to look at the linguistic phenomenon from a completely different angle than the original formulation of the principle. Whereas the principle of Abeillé and Godard is formulated with the simple and ubiquitous member relation in mind, the technical constraints on formulating a relation with the desired effect in terms of configurations of junk entities in models of a grammar forced us to construct a highly specialized junk slot whose relationship to the member relation is remote. As soon as the principles become more complex, the task of analyzing the original formulation of a principle and of formulating an equivalent junk slot encoding becomes very hard. For example, the complex interaction of negation, quantification, and relations of the BINDING THEORY would pose significant problems to anybody who would try to express the BINDING THEORY in terms of junk slots.[68] The lack of a formal interpretation in the description language of the original relational phrasing of the principles makes this task even harder, since it means that there cannot be a precise notion of equivalence between the relational formulation of the principle and the effect that a junk slot encoding has. All that a linguist has to go by is an informal understanding of what the relations of HPSG should mean. This is anything but precise. An adequate formal language should be able to express linguistic principles directly and without these complications.

The example also shows that relation symbols cannot be taken as metasyntactic notation for "corresponding" junk slots. It is not necessarily the case that a use of the member relation in a principle is translated into the *member* junk slot or its negative *non-member* twin. It is necessary to understand what a relation in a principle is supposed to express in order to reformulate it with an appropriate junk slot.

Although the problems of junk slots discussed above are serious enough to motivate alternative solutions, they are, to a certain extent, problems of elegance and practical feasibility in grammar writing and in reasoning about the models of grammars. There are, however, even more serious shortcomings that concern the semantics of junk slots. We have seen that junk slots cannot be negated with the negation of the description language. Instead, it is necessary to construct a second, negative junk slot to serve as the negation of the positive relation, as exemplified with the *member* junk slot (22) and the *non-member* junk slot (24). If the LIST value of a *u-member* entity, u, contains an infinite list, any entity can occur as value of the ELE attribute of u in a model of the grammar, and it is not necessary that the ELE value stands somewhere on the list: If there is an infinite embedding of *u-member* entities, each one of them satisfies the second description of (22). If the ELE value of u does not occur anywhere on its LIST value, it can at the same time be the ELE value of a *u-non-member* entity whose LIST value equals the LIST value of u. The positive and negative junk slot encodings of a relation might, thus, have

[68]The (relational) formalization of the BINDING THEORY in Section 4.2, (48)–(55), gives an impression of the complexities that are involved.

overlapping denotations. Entities with infinitely many components can easily arise in the recursive configurations of linguistic grammars, and they cannot be excluded by the expressive means of SRL. Non-standard effects through the lack of classical negation in the encoding of relations are thus a reality in HPSG 94 grammars that employ junk slot encodings of relations and their negations. This is certainly against the intentions of linguists working in the HPSG 94 framework, who assume that negation is classical. An adequate treatment of relations in a formalization of HPSG 94 should therefore preserve a classical interpretation of the negation of relations.

An adequate formalism for HPSG 94 must have true relations with true negation. This rules out the possibility of a definite clause extension of SRL as a relational extension of the description language. Because of practical, processing-related considerations, definite clause extensions of constraint languages (Höhfeld and Smolka, 1988) do not permit relations to occur in the scope of negation. They distinguish between the descriptions of the constraint language and an extra stratum of descriptions in which relations can be used. No such distinction is made in grammatical principles, as (21) illustrates. In HPSG, relations are used on a par with the other expressions of the description language. In systems for processing with constraint languages such as ALE (Carpenter and Penn, 1996), it is common to choose negation by failure as an approximation to the denotation of the negation of relations. Substitutes of this kind will not provide the set-theoretic interpretation of negation that matches the apparent intentions behind the use of negation in linguistic grammars, which are meant to be precise specifications of the natural languages that are the models of the grammars.

The examples have already shown that the relations of HPSG 94 occur with quantification over their arguments. The argument of sort *aff_r* of the member relation in principle (21) is existentially bound. The quantification becomes universal in the equivalent disjunctive form in which I investigated the possibilities of a junk slot encoding. It is easy to see that a relational formulation of the SUBCATEGORIZATION PRINCIPLE that corresponds to its junk slot encoding in (20) also involves at least one existential quantification: In each headed phrase, there exists a list l of *synsem* entities that stands in the `extract-ss` relation with the list of complement daughters, and the SUBCAT list of the mother, l, and the SUBCAT list of the head daughter stand in the `append` relation. The quantification over the arguments of relations raises the question of what kind of quantification is needed in a relational extension of SRL. Are the implicit quantifiers of HPSG 94 the standard quantifiers of first-order logic?

The answer to this question can be deduced from a number of prominent principles in Pollard and Sag 1994. The TRACE PRINCIPLE is one of them: "The SYNSEM value of any trace must be a (non-initial) member of the SUBCAT list of a substantive word"[69] (Pollard and Sag, 1994, p. 400). By definition, traces are signs with an empty phonology whose LOCAL value equals the single element of their INHERITED SLASH set and whose other nonlocal attributes have empty set values. The substantive word on whose SUBCAT list the SYNSEM value of the trace must appear is certainly not a component of the trace; neither can it be an arbitrary substantive word somewhere in the universe of linguistic entities, which would be the first-order existential interpretation of the existence requirement. For the principle to make sense, the minimal assumption must be that the trace and the substantive word are both components of one linguistic token, or, in other words, they are entities in a configuration under an entity that is their common ancestor. What Pollard and Sag 1994 meant to say is that for every linguistic token of a certain kind, if there is a path from the root entity to the trace, then there must also be a path from the root entity to a substantive word on whose SUBCAT list the SYNSEM value of the trace appears in non-initial position. The TRACE PRINCIPLE is working, so to speak, not on traces, but on some suitably chosen linguistic tokens. In a first approximation, one could assume that they are unembedded signs, i.e., signs that are independent utterances with illocutionary force. The relevant linguistic token delimits the possible scope of the quantification. For every trace that is a component of a token of the relevant kind, there exists a substantive word that is a component of the same token such that Pollard and Sag's condition holds. I conclude that the quantification of HPSG 94 is quantification over the components of an entity. In the light of the discussion of different technical explications of the meaning of HPSG 94 grammars, the relevant domain of quantification could also be characterized as quantification over the nodes in a 94 concrete feature structure or as quantification over the entities in a Pollard feature structure. In summary, quantification in HPSG 94 is quantification with respect to linguistic tokens; it is not quantification across linguistic tokens. This means that the existential and universal quantifiers of HPSG 94 are not first-order quantifiers, because first-order quantifiers would quantify over all entities in

[69] A substantive word is a word whose SYNSEM LOCAL CATEGORY HEAD value is in the denotation of *substantive*.

the linguistic universe.

The perspective of component quantification can be saliently observed in other principles as well. The CONTROL THEORY (Pollard and Sag, 1994, p. 401) demands that each control *qfpsoa* and each *local* entity whose CONTENT value equals the SOA-ARG value of the control *qfpsoa* observe certain conditions without specifying a structural relationship between the occurrence of the control *qfpsoa* and the *local* entity. Clearly, they must at least occur in one and the same linguistic token. The QUANTIFIER BINDING CONDITION (Pollard and Sag, 1994, p. 402) excludes occurrences of a quantifier's index that are not captured by the quantifier, and it means occurrences of the index in the same token. The BINDING THEORY (Pollard and Sag, 1994, p. 401) requires that certain entities in the language must or must not stand in the local-o-bind relation to some other entities; and it is clear that the entities that participate in the relation are always components of one token. Here it can be observed that relations between entities are defined with respect to the domain of linguistic tokens: The meaning of the relations of HPSG 94 is determined inside the tokens of the language; relations never hold between the entities of different tokens.

In the next chapter, I will present an extension to the formal languages of SRL that provides relations and existential and universal quantification over components of entities. The meaning of the relation symbols will be fixed by descriptions in the theory that employ component quantification. The interaction of the semantics of quantification and the semantics of relations will lead to a meaning of relations as relations between the entities in (linguistic) tokens. This will, in turn, make it possible to determine the meaning of the negation of the relations classically and to obtain an intuitively correct denotation of relational expressions and their negation.

Chapter 3

Formal Foundations II: RSRL

The purpose of the present chapter is to introduce and explain RSRL (Relational Speciate Re-entrant Language), a logical formalism for linguistic theories. RSRL provides a class of formal languages that is an extension of the class of languages of SRL. It adds variables, relations, and, most importantly, a restricted form of quantification to the logical languages of SRL. With these additional expressive means, RSRL is ideally suited to express HPSG 94 grammars of the kind presented in Pollard and Sag 1994 directly, faithfully, and very naturally.

The chapter is divided into three sections. Section 3.1 presents the languages and the model theory of RSRL. I will define the syntax and semantics of RSRL, and I will say what an RSRL grammar is and what it means for an RSRL grammar to be true of a natural language. To be more precise, I will offer three accounts of the meaning of an RSRL grammar: Initially, I will follow the work of King in SRL, and I will say that each RSRL grammar delimits a class of exhaustive models that contains the natural language. Under this perspective, it is the goal of the grammar writer to construct a grammar in such a way that the object language of the grammar—for example, Hungarian—is an exhaustive model of the grammar. I will then prove that each RSRL grammar indeed has an exhaustive model. The proof serves several purposes, and it justifies the choice of King's explanation of the meaning of grammars as the first explanation to be discussed here. The proof constitutes an explanation of what exactly an exhaustive model is; it relates the entities in the models of RSRL grammars to a variant of abstract feature structures, and it explains their relationship in technical detail; it gives a comprehensive mathematical justification for the informal discussion of the model theory of SRL in the previous chapter, since every SRL grammar is also an RSRL grammar; and, finally, it provides all the necessary constructs that are needed to show how the notions of tokens and of types, and all three definitions of the strong generative capacity of a grammar can be mathematically reconstructed in RSRL. That means that after the discussion of the proof of the existence of exhaustive models of RSRL grammars, we do not only have one, but we have three explanations of the meaning of RSRL grammars. Besides the realistic explanation of King, we also obtain the explanation of Pollard and Sag 1994 in terms of a set of abstract feature structures admitted by the grammar, and the explanation of Pollard (1999), who sees the meaning of a grammar in its strong generative capacity. In summation, I obtain a complete mathematical formalism for HPSG 94 that is compatible with all explanations of the meaning of HPSG 94 grammars that I discussed in Chapter 2.

At that point, however, one practical drawback for the application of the RSRL formalism to HPSG grammar writing is that the formal languages of RSRL adopt and extend the syntax of SRL, which bears no apparent resemblance to the commonly used AVM diagrams of the linguistic literature. On the one hand, that means that the syntax of RSRL is hard to use for linguists. On the other hand, the familiar AVM diagrams must be translated into a different syntax first before they receive a meaning in RSRL. If the conflict between mathematical precision and notational conventions could be resolved, existing grammars could be more easily interpreted as RSRL grammars as they stand, and RSRL would become much more accessible to the linguist. Although the particular format of the syntax of a formal language is of course irrelevant for the purposes of mathematical precision, it is an important issue for linguistic applications.[1] To be of maximal use, a formalism for HPSG 94 should provide an exact definition and an interpretation of a language that comes as close to the actual practice in the literature as the pervasive informal conventions permit. A formalism whose syntax complies with this demand is not only intuitively readable for the linguist; at the same time, it is suitable for rigorous formal specifications of natural language fragments.

[1]In the same spirit, Ronald Kaplan has emphasized the importance of notation in the context of formalisms for computer implementations of grammars: “Notation is a very important issue in the design of a linguistic theory, particularly if what you’re trying to do is to get people to use it. If you just want to use it yourself then perhaps it’s not such a big deal because you know what you mean.” (Kaplan, 1995, p. 358)

The goal of Section 3.2 is to bridge the gap between the syntax of RSRL and the notational conventions of the current HPSG literature by establishing an AVM syntax for RSRL. The meaning of the expressions of the AVM language will be determined by a translation function from AVM formulae to RSRL formulae, and I will show that for every grammar written in RSRL syntax, there is a grammar written in AVM syntax that is true of the same natural language. In other words, the AVM syntax is, in a linguistically relevant sense, equivalent to the original syntax of RSRL.

RSRL was designed as a formal language for precise specifications of grammars of natural languages, particularly for the specification of grammars written in the HPSG 94 framework. It was not designed for the purpose of computing with these grammars. Since HPSG grammars are often used in computational implementations, it is natural, however, to ask about the computational properties of RSRL. Section 3.3 reports three crucial results about the satisfiability problem, the modelability problem, and the prediction problem in RSRL, and briefly discusses the implications of these results for HPSG 94.

3.1 The Formalism

This section introduces the mathematics of the RSRL formalism. In 3.1.1, I define the syntax and semantics of the logical languages of RSRL. Since RSRL is a proper extension of the previously discussed SRL formalism, I will presuppose some familiarity with those constructs of RSRL that are already contained in SRL, and I will focus on those aspects of the description languages that are new. Occasionally, I will also use terminology that has been defined for SRL, before it is formally introduced in RSRL. I trust that this appeal to the intuitions of the reader will not lead to any confusion, and that it is always possible to see what the terminology of SRL means in the new, extended formalism. The additional constructs of the description language are variables, relations, quantification, and a number of reserved symbols to talk about *chains*.

The most important innovation of RSRL is component quantification. It expresses the insight that the quantification that is either explicitly stated or implicitly understood in many grammatical principles of constraint-based grammars like HPSG 94 is not quantification over all entities in the entire linguistic universe; it is quantification over the components of linguistic entities, for example, over the components of signs. It is component quantification that gives RSRL the ability to formalize grammatical principles of HPSG 94 very naturally.[2] The addition of variables to the formalism is necessitated by quantification and relations. Under the perspective of component quantification, the relations of HPSG 94 are relations between the components of linguistic entities. This view of relations—which, on the technical side, originates with the definition of the meaning of relational symbols under component quantification—will lead to a simple, classical interpretation of the negation of relational expressions. Chains are structures that look essentially like lists. They will allow me to generalize component quantification to quantification over structured collections of components. Chains will prove to be useful to capture certain generalizations about the use of relations that we will find in HPSG 94 grammars upon closer inspection of grammatical principles. When I define RSRL signatures, RSRL interpretations, and RSRL grammars below, I will simply call them signatures, interpretations, grammars, etc., since, for the rest of this thesis, I will only work within the new formalism, and no confusion with the other formalisms of Chapter 2 should arise.

Despite the additional complexity that the new constructs of RSRL add to the formalism of SRL, the meaning of RSRL grammars can be characterized in exactly the same ways as the meaning of SRL grammars. As a mathematical link between the three different views of the meaning of grammars of Pollard and Sag 1994, King 1999, and Pollard 1999, I use exhaustive models. My introduction of the description language of RSRL thus ends with a theorem about RSRL grammars: Each grammar has a class of exhaustive models. In 3.1.2, I outline and explain a proof of this central theorem about the grammars of RSRL. In the course of the proof, which involves constructing a canonical exhaustive model for each grammar, it will become clear that appropriately modified versions of all important notions surrounding the model theory of SRL find their counterpart in the extended formalism. Most importantly, if two entities in two interpretations cannot be told apart by any description under the description denotation function of the respective interpretations, the two entities in their interpretations are

[2] Bob Kasper and Carl Pollard pointed out to me that the component quantification of RSRL can be seen as an instance of the general notion of *bounded quantification*. Bounded quantification is quantification over some restricted set of objects in an interpretation rather than quantification over the entire universe of objects in an interpretation as in standard first-order logic.

also congruent, and *vice versa.* Since indiscernibility and congruence remain equivalent, exhaustive models can be defined exactly as in SRL either purely in terms of description denotation as those models of a grammar in which any description that describes some entity in some model of the grammar also describes an entity, or algebraically as those models of a grammar that simulate any other model of the grammar. On the surface, these definitions look exactly as before. The only difference is that the description language to which they refer has additional kinds of expressions, and the structure of the interpretations is richer.

In Section 3.1.3, I will discuss how the technicalities of the existence proof of exhaustive models can be used to reconstruct the different perspectives of the meaning of HPSG 94 grammars of Chapter 2. Each perspective, of course, is based on a different philosophy of science. To decide which one to choose, one must decide which interpretation of formal scientific theories comes closest to one's own views.

Obviously, the proof shows that the view that the meaning of a grammar is the class of its exhaustive models is available. The exact correspondence of the definitions of exhaustive models in SRL and RSRL implies that King's view of a language as a system of tokens, i.e., as entities in the (exhaustive) model of a grammar that is the natural language, can be maintained as well as the idea that the object types of a language are equivalence classes of tokens in an exhaustive model of the grammar. Supposing an appropriate philosophy of science, we can then say that an RSRL grammar is true of a natural language under the same conditions under which an SRL grammar is true of a natural language.

But the view of Pollard and Sag 1994 is also available. The existence proof of exhaustive models of Section 3.1.2 follows the structure of King's original proof for SRL. Therefore, the construction of the canonical exhaustive models of grammars involves a variant of abstract feature structure admission. Since the interpretations of RSRL signatures contain relations, suitable abstract feature structures must contain appropriate representations of relations. The definition of extended abstract feature structures is again based on the Moshier representation of abstract feature structures. To distinguish the augmented structures from ordinary 94 abstract feature structures, I will call them *morphs.* Morph admission is defined in parallel to the simpler abstract feature structure admission in feature logics without a relational extension. The one-to-one correspondence between the morphs admitted by a grammar and the object types of the natural language that the grammar captures remains intact. The set of morphs admitted by a grammar can therefore serve as a set of mathematical representations of the object types of the language that the grammar describes. The meaning of a grammar is the set of morphs admitted by the grammar.

Finally, I will show that Pollard feature structures and Pollard abstract feature structures have very close analogues in RSRL. The definition of extended Pollard abstract feature structures leads directly to a notion of the strong generative capacity of an RSRL grammar. According to one of the three definitions that Pollard offers, the strong generative capacity is the set of all extended Pollard abstract feature structures that can be abstracted from the models of the grammar. It is then possible to see the meaning of a grammar in its strong generative capacity.

3.1.1 The Description Language

Following SRL, descriptions of RSRL are true or false of entities in an interpretation. RSRL provides a class of formal languages consisting of a signature and a class of interpretations. The signature provides the sets of symbols which generate the formal language. Each interpretation of a signature provides a set of entities and assigns each symbol of the signature a meaning in terms of those entities. Formulae are built from variables, the sets of symbols of the signature, and a small set of reserved symbols. Each formula in which no variable occurs free is called a description. A theory is a set of descriptions. Each variable that occurs in a description in a theory will thus be bound by component quantification. Interpretations induce a formula denotation function, a description denotation function and a theory denotation function. The description denotation function assigns each description the set of entities of which it is true, and the theory denotation function assigns each theory the set of entities of which each description in the theory is true. A grammar is a signature together with a theory; and a model of a grammar is an interpretation of its signature that only contains entities of which every description in the theory of the grammar is true. Following King 1999, the meaning of a grammar can then be determined by a class of models which meet certain conditions. Informally, a grammar delimits a class of interpretations, EM, of its signature such that its theory is true of every entity in each interpretation in EM, and for every entity u in any other model of the grammar, each interpretation in EM contains an entity that is indiscernible from u.

The syntactic symbols of each formal language are given by the set of variables and the signature:

Definition 46 $\mathcal{VAR}$ *is a countably infinite set of symbols.*

I call each element of $\mathcal{VAR}$ a *variable*.

In contrast to SRL signatures, the signatures of RSRL contain an explicit sort hierarchy. The motivation for the inclusion of sort hierarchies comes from the intended main application of the formal languages, the specification of HPSG 94 grammars, which make heavy use of sort hierarchies and attribute inheritance in sort hierarchies. Including sort hierarchies explicitly in the formal languages merely avoids the extra step of the definition of a syntactic translation from grammars with a sort hierarchy to corresponding grammars without. From a purely mathematical point of view, nothing is gained by an explicit sort hierarchy. For reasons of mathematical elegance, it could as well be left implicit, as argued by King 1999, pp. 229–331. Similar to SRL, which allows infinite sets of species in signatures, I allow infinite sort hierarchies. In actual linguistic grammars, the sort hierarchies are finite.

Definition 47 Σ *is a* ***signature*** *iff*

> Σ *is a septuple* $\langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$,
>
> $\langle \mathcal{G}, \sqsubseteq \rangle$ *is a partial order,*
>
> $\mathcal{S} = \left\{ \sigma \in \mathcal{G} \,\middle|\, \begin{array}{l} \textit{for each } \sigma' \in \mathcal{G}, \\ \textit{if } \sigma' \sqsubseteq \sigma \textit{ then } \sigma = \sigma' \end{array} \right\}$,
>
> $\mathcal{A}$ *is a set,*
>
> $\mathcal{F}$ *is a partial function from the Cartesian product of* $\mathcal{G}$ *and* $\mathcal{A}$ *to* $\mathcal{G}$,
>
> *for each* $\sigma_1 \in \mathcal{G}$, *for each* $\sigma_2 \in \mathcal{G}$ *and for each* $\alpha \in \mathcal{A}$,
>
> > *if* $\mathcal{F}\langle \sigma_1, \alpha \rangle$ *is defined and* $\sigma_2 \sqsubseteq \sigma_1$
> > *then* $\mathcal{F}\langle \sigma_2, \alpha \rangle$ *is defined and* $\mathcal{F}\langle \sigma_2, \alpha \rangle \sqsubseteq \mathcal{F}\langle \sigma_1, \alpha \rangle$,
>
> $\mathcal{R}$ *is a finite set, and*
>
> $\mathcal{AR}$ *is a total function from* $\mathcal{R}$ *to* $\mathbb{N}^+$.

I call each element of $\mathcal{G}$ a *sort*, $\langle \mathcal{G}, \sqsubseteq \rangle$ the *sort hierarchy*, each element of $\mathcal{S}$ a *maximally specific sort* or a *species*, each element of $\mathcal{A}$ an *attribute*, $\mathcal{F}$ the *appropriateness function*, each element of $\mathcal{R}$ a *relation symbol*, and $\mathcal{AR}$ the *arity function*. I say that sort σ_2 is *at least as specific as* sort σ_1 iff $\sigma_2 \sqsubseteq \sigma_1$; conversely, I call σ_1 a *supersort* of σ_2. The symbols $\forall$, $\exists$, :, $\sim$, $\approx$, $\neg$, [,], $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, †, $\triangleright$, *chain, echain, nechain* and *metatop* are reserved symbols, and I assume that none of them are a variable, a sort, an attribute or a relation symbol. For reasons of generality, the set of attributes in a signature is allowed to be infinite. In signatures of linguistic grammars, the set of attributes is finite. The appropriateness function, $\mathcal{F}$, respects the usual postulates about attribute inheritance in sort hierarchies. If an attribute, α, is appropriate to a sort σ_1 then it is appropriate to any sort σ_2 which is more specific than σ_1, and $\mathcal{F}\langle \sigma_2, \alpha \rangle$ is at least as specific as $\mathcal{F}\langle \sigma_1, \alpha \rangle$. The arity function, $\mathcal{AR}$, specifies the arity of each relation symbol as a positive integer. For example, the standard `member` relation has arity 2, and the standard `append` relation has arity 3.

Before I define the class of interpretations of each signature, I need to introduce a few notational conventions that will make the following definitions easier to read. Suppose S is a set. S^* is the set of finite strings over S, including the empty string. I will refer to it as the set of finite sequences of elements of S. Similarly, S^+ is the set of nonempty finite sequences of elements of S. For convenience, I will henceforth write $\overline{S}$ as an abbreviation for $S \cup S^*$.

RSRL signatures are interpreted as follows:

Definition 48 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, I *is a* Σ ***interpretation*** *iff*

> I *is a quadruple* $\langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$,
>
> U *is a set,*
>
> S *is a total function from* U *to* $\mathcal{S}$,
>
> A *is a total function from* $\mathcal{A}$ *to the set of partial functions from* U *to* U,
>
> *for each* $\alpha \in \mathcal{A}$ *and each* $u \in \mathsf{U}$,
>
> > *if* $\mathsf{A}(\alpha)(u)$ *is defined*
> > *then* $\mathcal{F}\langle \mathsf{S}(u), \alpha \rangle$ *is defined, and* $\mathsf{S}(\mathsf{A}(\alpha)(u)) \sqsubseteq \mathcal{F}\langle \mathsf{S}(u), \alpha \rangle$, *and*

for each $\alpha \in \mathcal{A}$ *and each* $u \in \mathsf{U}$,

if $\mathcal{F}\langle \mathsf{S}(u), \alpha\rangle$ *is defined then* $\mathsf{A}(\alpha)(u)$ *is defined,*

R *is a total function from* $\mathcal{R}$ *to the power set of* $\bigcup\limits_{n \in \mathbb{N}} \overline{\mathsf{U}}^n$, *and*

for each $\rho \in \mathcal{R}$, $\mathsf{R}(\rho) \subseteq \overline{\mathsf{U}}^{\mathcal{AR}(\rho)}$.

U is the *set of entities in the universe.* I call S the *species assignment function*, A the *attribute interpretation function*, and R the *relation interpretation function.* As in the previous chapter, I adopt the convention of saying that an entity, u, is an element of a Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R}\rangle$ if $u \in \mathsf{U}$, and I sometimes write $u \in \mathsf{I}$ to indicate that u is in U.

S partitions the entities of the universe by assigning a species to every entity. Informally, one might say that each species denotes a set of entities, and each entity is in the denotation of exactly one species. Extending this terminology to sorts in general, I say that each sort, σ, denotes the union of the sets of entities denoted by the species of which σ is a supersort. The function A provides a denotation for attributes. Each attribute denotes a partial function from entities to entities. For convenience, I sometimes refer to the function denoted by an attribute, α, as (the function) α. The denotation of attributes in a Σ interpretation is governed by the appropriateness function, $\mathcal{F}$. An attribute, α, is defined on all or no entities in the denotation of a species, σ, depending on whether $\mathcal{F}\langle \sigma, \alpha\rangle$ is defined or not, and the value of the partial function denoted by α on an entity of species σ is in the denotation of $\mathcal{F}\langle \sigma, \alpha\rangle$. R interprets each relation symbol as a set of tuples of the form $\langle u_1, \ldots, u_n\rangle$, where: (a) each u_i is an entity in U, or (b) each u_i is a sequence of entities in U, or (c) some u_i are entities and others are sequences of entities in U. The number n is determined by the arity of the relation as given by the arity function.

As witnessed by the cases (b) and (c), arguments of relations may be sequences of entities in U. I call these sequences of entities *chains* (of entities). Technically, chains are strings over U. The syntax of the formal languages of RSRL provides several reserved symbols for the description of chains of entities. For example, we might want to specify the number of entities on a chain, their sorts, or the configurations of entities under the entities on a chain.

The main purpose of chains is to treat certain uses of apparent lists in the HPSG literature where the lists occur in the arguments of relations without being a component of the entities that are described. Only the elements of these "virtual lists" are components of the described entities, but not the apparent lists themselves. A typical relation of HPSG 94 grammars that uses chains in its arguments is the `shuffle` relation of linearization grammars. The relation `shuffle`, a relation between three lists, shuffles the lists in its first two arguments into the list in its third argument, preserving the relative order of their members.[3] In applications of `shuffle` in linguistic principles in the literature, the situation arises that one of the lists in the first two arguments of `shuffle` is the "output" of another relation that specifies that list based on other properties of the described configuration of entities. Then that list may occur nowhere in the configuration of entities under the entities that we wish to describe; it is merely an intermediate result that is shared between the arguments of two relations. This kind of apparent lists in the arguments of relations will be treated as chains, thus obviating the need for otherwise unmotivated book-keeping attributes and attribute values that would reify those lists as components of the described entities, or for a reformulation of the relations that does not reflect the underlying intuitions about the relevant principles of grammar, because it would miss generalizations that are important to the linguist. The linguistic generalizations that motivate chains in grammars will be discussed extensively in Section 4.3 below.

For reasoning with chains, I need to supplement the sort hierarchy, $\langle \mathcal{G}, \sqsubseteq \rangle$, the set of species, $\mathcal{S}$, and the set of attributes, $\mathcal{A}$. The symbols *chain, echain* and *nechain* serve as quasi-sorts for the description of chains, empty chains and nonempty chains, and are ordered in an extension to the sort hierarchy in the intuitive way: The symbols *chain, echain* and *nechain* are the elements of a finite partial order, the *chain hierarchy*, in which *echain* and *nechain* are below *chain*. I call *echain* and *nechain* maximally specific with respect to the chain hierarchy. With the quasi-attributes $\dagger$ and $\triangleright$, chains can be navigated. For the time being, it might be helpful to think of chains as lists, and to see the chain symbols in analogy to the standard symbols to describe lists in HPSG grammars. The quasi sorts *chain, echain* and *nechain* correspond to the sorts *list, elist* and *nelist*, and the quasi-attributes $\dagger$ and $\triangleright$ correspond to the FIRST and REST attributes, respectively. In DEFINITION 49, signatures are augmented by the partial order of quasi-sorts just described, the two

[3]The `shuffle` relation is defined in (64) in Section 4.3.

quasi-attributes, † and ▷, and another reserved quasi-sort symbol, *metatop*, which is the top of the expanded sort hierarchy:

Definition 49 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$,

$$\widehat{\mathcal{G}} = \mathcal{G} \cup \{\textit{chain}, \textit{echain}, \textit{nechain}, \textit{metatop}\},$$

$$\widehat{\sqsubseteq} = \sqsubseteq \cup \ \{\langle \textit{echain}, \textit{chain}\rangle, \langle \textit{nechain}, \textit{chain}\rangle\} \ \cup \left\{ \langle \sigma, \sigma \rangle \ \middle| \sigma \in \widehat{\mathcal{G}} \backslash \mathcal{G} \right\} \cup \left\{ \langle \sigma, \textit{metatop} \rangle \ \middle| \sigma \in \widehat{\mathcal{G}} \right\},$$

$$\widehat{\mathcal{S}} = \mathcal{S} \cup \{\textit{echain}, \textit{nechain}\}, \textit{ and}$$

$$\widehat{\mathcal{A}} = \mathcal{A} \cup \{\dagger, \triangleright\}.$$

Note that if $\langle \mathcal{G}, \sqsubseteq \rangle$ is a finite partial order then $\left\langle \widehat{\mathcal{G}}, \widehat{\sqsubseteq} \right\rangle$ is also a finite partial order. I call $\widehat{\mathcal{G}}$ the *expanded sort set*, $\left\langle \widehat{\mathcal{G}}, \widehat{\sqsubseteq} \right\rangle$ the *expanded sort hierarchy*, $\widehat{\sqsubseteq}$ the *expanded sort hierarchy relation*, $\widehat{\mathcal{S}}$ the *expanded species set*, and $\widehat{\mathcal{A}}$ the *expanded attribute set*. Besides the sorts of the signature, $\widehat{\mathcal{G}}$ contains the chain symbols and *metatop*, which, according to the definition of $\widehat{\sqsubseteq}$, is the top element of the expanded sort hierarchy. Moreover, the expanded sort hierarchy relation properly contains the chain hierarchy and the ordering relation of the sort hierarchy. The expanded species set, $\widehat{\mathcal{S}}$, includes the two maximally specific elements in the chain hierarchy, *echain* and *nechain*. For each σ_1 and for each σ_2 in the expanded sort hierarchy, I say that σ_1 *subsumes* σ_2 iff $\sigma_2 \mathrel{\widehat{\sqsubseteq}} \sigma_1$. Finally, the expanded attribute set, $\widehat{\mathcal{A}}$, contains the two reserved quasi-attribute symbols, † and ▷, besides the elements of the set of attributes, $\mathcal{A}$.

Parallel to the extension of the signature, the species assignment, S, and the attribute interpretation function, A, must be expanded to give the new symbols *echain*, *nechain*, †, and ▷ a denotation in an interpretation. The analogy of their denotation to the denotation of the normal list symbols *elist, nelist*, FIRST, and REST is obvious:

Definition 50 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$,

$\widehat{\mathsf{S}}$ *is the total function from* $\overline{\mathsf{U}}$ *to* $\widehat{\mathcal{S}}$ *such that*

for each $u \in \mathsf{U}$, $\widehat{\mathsf{S}}(u) = \mathsf{S}(u)$,
for each $u_1 \in \mathsf{U}$, ..., *for each* $u_n \in \mathsf{U}$,

$$\widehat{\mathsf{S}}(\langle u_1, \ldots, u_n \rangle) = \begin{cases} \textit{echain} & \textit{if } n = 0, \\ \textit{nechain} & \textit{if } n > 0 \end{cases}, \textit{ and}$$

$\widehat{\mathsf{A}}$ *is the total function from* $\widehat{\mathcal{A}}$ *to the set of partial functions from* $\overline{\mathsf{U}}$ *to* $\overline{\mathsf{U}}$ *such that*

for each $\alpha \in \mathcal{A}$, $\widehat{\mathsf{A}}(\alpha) = \mathsf{A}(\alpha)$,
$\widehat{\mathsf{A}}(\dagger)$ *is the total function from* U^+ *to* U *such that for each* $\langle u_0, \ldots, u_n \rangle \in \mathsf{U}^+$,

$$\widehat{\mathsf{A}}(\dagger)(\langle u_0, \ldots, u_n \rangle) = u_0, \textit{ and}$$

$\widehat{\mathsf{A}}(\triangleright)$ *is the total function from* U^+ *to* U^* *such that for each* $\langle u_0, \ldots, u_n \rangle \in \mathsf{U}^+$,

$$\widehat{\mathsf{A}}(\triangleright)(\langle u_0, \ldots, u_n \rangle) = \langle u_1, \ldots, u_n \rangle.$$

I call $\widehat{\mathsf{S}}$ the *expanded species assignment function*, and $\widehat{\mathsf{A}}$ the *expanded attribute interpretation function*. For all elements of U in its domain, $\widehat{\mathsf{S}}$ equals the species assignment function, S. In addition, $\widehat{\mathsf{S}}$ assigns the quasi-sort *echain* to the empty chain, and *nechain* to each nonempty chain of entities. Informally, I say that *echain* denotes the singleton set containing the empty chain, and *nechain* denotes the set of nonempty chains. Generalizing the terminology introduced for sorts to the expanded sort set, I say that each symbol, σ, in the expanded sort set denotes the union of the sets of entities or chains denoted by the maximally specific symbols which σ subsumes in the expanded sort hierarchy relation. Therefore, *metatop* denotes all entities and chains of entities of the universe in each interpretation. $\widehat{\mathsf{A}}$ equals the attribute interpretation function, A, for all attributes in its domain; but its domain also includes $\dagger$ and $\triangleright$, and its range includes partial functions whose domains are chains. $\widehat{\mathsf{A}}$ interprets the quasi-attribute $\dagger$ as mapping each nonempty chain to its first element, and the quasi-attribute $\triangleright$ as mapping each nonempty chain, c, to the chain that is just like c but with the first element of c missing.

Since the formal language of RSRL contains variables, variable assignments will be needed to define denotations for its expressions. Variables will occur in the arguments of relational expressions, and the arguments of relations can be entities or chains of entities. Variable assignments must thus be able to assign elements of U and chains of elements of U to variables:

Definition 51 *For each signature* Σ, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$,

$Ass_{\mathsf{I}} = \overline{\mathsf{U}}^{\mathcal{VAR}}$ *is the **set of variable assignments in*** I.

I call each element of Ass_{I} a *variable assignment in* I. Elements of Ass_{I} are functions which assign entities or chains of entities of the universe to variables. Of course, variables will potentially occur in syntactic environments other than the arguments of relations as well, thus giving rise to the possibility of describing chains of entities independent from the description of the arguments of relations.

We are now ready to define the syntax and semantics of the formal languages of RSRL. First, terms and their denotations are defined. The only difference to the terms of SRL is that terms may start with a variable instead of the colon. Second, the terms are used as basic syntactic expressions for constructing complex formulae and their denotation. The syntax of complex formulae will contain the interesting additions to the syntax of SRL, quantification and relations.

Definition 52 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, $\mathcal{T}^{\Sigma}$ *is the smallest set such that*

$: \in \mathcal{T}^{\Sigma}$,

for each $v \in \mathcal{VAR}$, $v \in \mathcal{T}^{\Sigma}$, *and*

for each $\alpha \in \widehat{\mathcal{A}}$ *and each* $\tau \in \mathcal{T}^{\Sigma}$, $\tau\alpha \in \mathcal{T}^{\Sigma}$.

I call each element of $\mathcal{T}^{\Sigma}$ a Σ *term*. A Σ term consists of either the reserved symbol ':' or a variable followed by a (possibly empty) string of symbols of the expanded attribute set. Σ terms are interpreted as follows:

Definition 53 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, *for each* $ass \in Ass_{\mathsf{I}}$, T_{I}^{ass} *is the total function from* $\mathcal{T}^{\Sigma}$ *to the set of partial functions from* U *to* $\overline{\mathsf{U}}$ *such that for each* $u \in \mathsf{U}$,

$T_{\mathsf{I}}^{ass}(:)(u)$ *is defined and* $T_{\mathsf{I}}^{ass}(:)(u) = u$,

for each $v \in \mathcal{VAR}$, $T_{\mathsf{I}}^{ass}(v)(u)$ *is defined and* $T_{\mathsf{I}}^{ass}(v)(u) = ass(v)$,

for each $\tau \in \mathcal{T}^{\Sigma}$, *for each* $\alpha \in \widehat{\mathcal{A}}$,

$T_{\mathsf{I}}^{ass}(\tau\alpha)(u)$ *is defined*

iff $T_{\mathsf{I}}^{ass}(\tau)(u)$ *is defined and* $\widehat{\mathsf{A}}(\alpha)(T_{\mathsf{I}}^{ass}(\tau)(u))$ *is defined, and* *if* $T_{\mathsf{I}}^{ass}(\tau\alpha)(u)$ *is defined*

then $T_{\mathsf{I}}^{ass}(\tau\alpha)(u) = \widehat{\mathsf{A}}(\alpha)(T_{\mathsf{I}}^{ass}(\tau)(u))$.

I call T_{I}^{ass} the Σ *term interpretation function with respect to* I *under a variable assignment in* I, *ass*. Each Σ term is interpreted as a partial function from U to $\overline{\mathsf{U}}$. The colon denotes the identity function in U. A variable, v, denotes a constant total function from U to $\overline{\mathsf{U}}$, where each element of U is assigned the entity in U or the chain of entities in U that the variable assignment in I assigns to v. Finally, the meaning of complex Σ terms results from functional composition of the denotation of each of the symbols of the expanded attribute set succeeding the colon or variable in reverse order of their appearance in the Σ term, and the meaning of the colon or variable. Note that only Σ terms which begin with a variable which is followed by a (possibly empty) string of the reserved symbol $\triangleright$ may map elements of U to chains of entities of U. All other terms denote functions which, if they are defined on any entity of the universe at all, may only map them to entities of U. This is easy to see: A Σ term starting with the colon behaves like a term in SRL that only maps elements of U to elements of U. If a Σ term starts with a variable, it can only map an element of U to a chain of entities of U if the variable assignment in I assigns a chain of entities to the variable, because no attribute and no quasi-attribute maps an element of U to a chain of elements of U. Finally, $\triangleright$ is the only element of the expanded attribute set that maps chains of entities of U to chains of entities of U, while † maps chains of entities of U to entities of U, and attributes denote partial functions from U to U. I say that a Σ term, τ, is *defined on an entity*, u, iff τ begins with the colon and $T_{\mathsf{I}}^{ass}(\tau)(u)$ is defined.

Notice that, in case τ begins with a variable, v, the value of $T_{\mathsf{I}}^{ass}(\tau)(u)$, and the question of whether it is defined, is independent of u, because it depends entirely on $ass(v)$. A small example will make this clear. Assume $\mathcal{VAR} = \left\{ v_n \,\middle|\, n \in \mathbb{N} \right\}$, and consider the signature $\Sigma_{3.1}$ in Figure 3.1 (which is the analogue in RSRL to the 87 signature of Figure 2.3 and the SRL signature of Figure 2.4), the $\Sigma_{3.1}$ interpretation $\mathsf{I}_{3.2}$ in Figure 3.2 (which, in essence, repeats the interpretation of Figure 2.6), and the variable assignments in $\mathsf{I}_{3.2}$, ass_0 and ass_1 in (28).

(28) a. $ass_0 = \left\{ \langle v_n, \text{Anne} \rangle \,\middle|\, n \in \mathbb{N} \right\}$

b. $ass_1 = \left\{ \langle v_n, \text{third car} \rangle \,\middle|\, n \in \mathbb{N} \right\}$

Figure 3.1: The car scenario signature in RSRL

top

car OWNER *person*
DRIVER *person*

vw

bmw

person LIKES-BEST *top*

man

woman

and $\mathcal{R} = \emptyset$, $\mathcal{AR} = \emptyset$.

Figure 3.2: An interpretation of the car scenario signature

Let $\mathsf{I}_{3.2} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, with:

$\mathsf{U} = \{$Anne, Manfred, Maria, Mike, third car$\}$

S(Anne) = *woman*,
S(Maria) = *woman*,
S(Manfred) = *man*,
S(Mike) = *man*,
S(third car) = *vw*, and

A(LIKES-BEST)(Anne) = Manfred,
A(LIKES-BEST)(Manfred) = Anne,
A(LIKES-BEST)(Maria) = Mike,
A(LIKES-BEST)(Mike) = Maria,
A(OWNER)(third car) = Anne,
A(DRIVER)(third car) = Mike,
A(LIKES-BEST) is undefined on the third car,
A(OWNER) is undefined on Anne, Manfred, Maria and Mike,
A(DRIVER) is likewise undefined on Anne, Manfred, Maria and Mike, and

$\mathsf{R} = \emptyset$.

Then $T^{ass_0}_{\mathsf{I}_{3.2}}(v_0\text{OWNER})(u)$ is undefined for all $u \in \mathsf{I}_{3.2}$, i.e., for the third car, Anne, Manfred, Maria, and Mike, because $ass_0(v_0)$ = Anne, and the attribute OWNER is not defined on Anne. $T^{ass_1}_{\mathsf{I}_{3.2}}(v_0\text{OWNER})(u)$, on the other hand, is defined for all $u \in \mathsf{I}_{3.2}$, and $T^{ass_1}_{\mathsf{I}_{3.2}}(v_0\text{OWNER})(u) = \text{Anne}$, because $ass_1(v_0) = \text{third car}$, the attribute OWNER is defined on the third car, and $\mathsf{A}(\text{OWNER})(\text{third car}) = \text{Anne}$.

When a Σ term of the form $v\pi$, where $v \in \mathcal{VAR}$ and $\pi \in \mathcal{A}^*$, is part of a formula without free occurrences of variables, component quantification will make the value of applying the denotation of $v\pi$ to an entity u dependent on u. It will guarantee that the assignment is always minimally changed such that v is mapped to a component (or chain of components) of u.

The syntax of formulae is based on terms. The syntax of RSRL's formulae contains relational formulae and quantification in addition to all syntactic constructs of the descriptions of SRL. Unlike definite clause extensions of constraint languages, which add relations in a separate layer to the core constraint language, relational formulae belong to the core description language itself, because I want to use them in the scope of negation and quantification.

Definition 54 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, $\mathcal{D}^\Sigma$ *is the smallest set such that*

for each $\sigma \in \widehat{\mathcal{G}}$, *for each* $\tau \in \mathcal{T}^\Sigma$, $\tau \sim \sigma \in \mathcal{D}^\Sigma$,

for each $\tau_1 \in \mathcal{T}^\Sigma$, *for each* $\tau_2 \in \mathcal{T}^\Sigma$, $\tau_1 \approx \tau_2 \in \mathcal{D}^\Sigma$,

for each $\rho \in \mathcal{R}$, *for each* $x_1 \in \mathcal{VAR}$, ..., *for each* $x_{\mathcal{AR}(\rho)} \in \mathcal{VAR}$,

$\rho(x_1, \ldots, x_{\mathcal{AR}(\rho)}) \in \mathcal{D}^\Sigma$,

for each $x \in \mathcal{VAR}$, *for each* $\delta \in \mathcal{D}^\Sigma$, $\exists x\, \delta \in \mathcal{D}^\Sigma$,

for each $x \in \mathcal{VAR}$, *for each* $\delta \in \mathcal{D}^\Sigma$, $\forall x\, \delta \in \mathcal{D}^\Sigma$,

for each $\delta \in \mathcal{D}^\Sigma$, $\neg\delta \in \mathcal{D}^\Sigma$,

for each $\delta_1 \in \mathcal{D}^\Sigma$, *for each* $\delta_2 \in \mathcal{D}^\Sigma$, $[\delta_1 \wedge \delta_2] \in \mathcal{D}^\Sigma$,

for each $\delta_1 \in \mathcal{D}^\Sigma$, *for each* $\delta_2 \in \mathcal{D}^\Sigma$, $[\delta_1 \vee \delta_2] \in \mathcal{D}^\Sigma$,

for each $\delta_1 \in \mathcal{D}^\Sigma$, *for each* $\delta_2 \in \mathcal{D}^\Sigma$, $[\delta_1 \rightarrow \delta_2] \in \mathcal{D}^\Sigma$, *and*

for each $\delta_1 \in \mathcal{D}^\Sigma$, *for each* $\delta_2 \in \mathcal{D}^\Sigma$, $[\delta_1 \leftrightarrow \delta_2] \in \mathcal{D}^\Sigma$.

I call each element of $\mathcal{D}^\Sigma$ a Σ *formula*. Σ formulae of the form $\tau \sim \sigma$ are called *sort assignment formulae*, Σ formulae of the form $\tau_1 \approx \tau_2$ are called *path equations*, and Σ formulae of the form $\rho(x_1, \ldots, x_{\mathcal{AR}(\rho)})$ are called *relational* Σ *formulae*. They constitute the atomic Σ formulae. Complex

Σ formulae can be formed from atomic Σ formulae by means of existential and universal quantification and the standard logical connectives. Readers familiar with the conventional linguistic notation of relational expressions might be surprised that variables are the only Σ expressions that are allowed in the argument positions of relational Σ formulae. The only reason for the conservative definition here is the simplicity of the formal languages. The alternative notation of Section 3.2 that is customized for linguistic grammars will be more liberal.

The expressions defined in Definition 54 are deliberately not called descriptions, in contrast to the expressions in the corresponding part of Definition 17 of SRL. I reserve the term "description" for the subset of formulae that will be permitted in the theory of a grammar. The formulae in a theory of a grammar must not contain free occurrences of variables. In SRL grammars, that condition is trivially satisfied, since its expressions do not contain any variables at all. In the present formalism, the expressions given by Definition 54 might contain free variables. In order to be able to say later what a formula without free occurrences of variables is, I define for each Σ term and formula, the set of free occurrences of variables in that Σ term or formula:

Definition 55 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$,

$FV(:) = \emptyset$,

for each $v \in \mathcal{VAR}$, $FV(v) = \{v\}$,

for each $\tau \in \mathcal{T}^{\Sigma}$, *for each* $\alpha \in \widehat{\mathcal{A}}$, $FV(\tau\alpha) = FV(\tau)$,

for each $\tau \in \mathcal{T}^{\Sigma}$, *for each* $\sigma \in \widehat{\mathcal{G}}$, $FV(\tau \sim \sigma) = FV(\tau)$,

for each $\tau_1, \tau_2 \in \mathcal{T}^{\Sigma}$, $FV(\tau_1 \approx \tau_2) = FV(\tau_1) \cup FV(\tau_2)$,

for each $\rho \in \mathcal{R}$, *for each* $x_1, \ldots, x_{\mathcal{AR}(\rho)} \in \mathcal{VAR}$,

$$FV(\rho(x_1, \ldots, x_{\mathcal{AR}(\rho)})) = \{x_1, \ldots, x_{\mathcal{AR}(\rho)}\},$$

for each $\delta \in \mathcal{D}^{\Sigma}$, *for each* $v \in \mathcal{VAR}$, $FV(\exists v\, \delta) = FV(\delta)\backslash\{v\}$,

for each $\delta \in \mathcal{D}^{\Sigma}$, *for each* $v \in \mathcal{VAR}$, $FV(\forall v\, \delta) = FV(\delta)\backslash\{v\}$,

for each $\delta \in \mathcal{D}^{\Sigma}$, $FV(\neg\delta) = FV(\delta)$,

for each $\delta_1 \in \mathcal{D}^{\Sigma}$, *for each* $\delta_2 \in \mathcal{D}^{\Sigma}$,

$$FV([\delta_1 \wedge \delta_2]) = FV(\delta_1) \cup FV(\delta_2),$$

for each $\delta_1 \in \mathcal{D}^\Sigma$, *for each* $\delta_2 \in \mathcal{D}^\Sigma$,

$$FV([\delta_1 \vee \delta_2]) = FV(\delta_1) \cup FV(\delta_2),$$

for each $\delta_1 \in \mathcal{D}^\Sigma$, *for each* $\delta_2 \in \mathcal{D}^\Sigma$,

$$FV([\delta_1 \rightarrow \delta_2]) = FV(\delta_1) \cup FV(\delta_2), \textit{ and}$$

for each $\delta_1 \in \mathcal{D}^\Sigma$, *for each* $\delta_2 \in \mathcal{D}^\Sigma$,

$$FV([\delta_1 \leftrightarrow \delta_2]) = FV(\delta_1) \cup FV(\delta_2).$$

The definition of free occurrences of variables distinguishes between Σ terms that begin with the colon, which do not contain free variables, and Σ terms that begin with a variable, which is then free in that Σ term.

As motivated in Section 2.2.3, existential and universal quantification in RSRL is quantification over the components of entities. To be able to define quantification as component quantification, I first need to be able to refer to the set of the components of each entity in an interpretation. In other words, I need an analogue in RSRL to the set of components of an entity u in an SRL interpretation I, given as $\mathsf{Co}_{\mathsf{I}}(u)$ in DEFINITION 22. The SRL definition, however, will not do, since it defines $\mathsf{Co}_{\mathsf{I}}(u)$ as the set of entities in the Σ interpretation I that is obtained by collecting the results of applying the denotation of all Σ terms to u. This yields the intended result, because every Σ term of SRL starts with the colon. To achieve the same effect in RSRL, the right subset of all Σ terms must be chosen. The set of Σ terms with which the set of components of an entity is defined must be restricted to those Σ terms that start with the colon, because—as discussed above—the denotation on an entity, u, of Σ terms that start with a variable might yield any entity in the universe of which u is not an ancestor, depending on which entity or chain of entities in I is assigned to the variable by the variable assignment in I.

Definition 56 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, *and for each* $u \in \mathsf{U}$,

$$\mathsf{Co}_{\mathsf{I}}^{u} = \left\{ u' \in \mathsf{U} \;\middle|\; \begin{array}{l} \textit{for some ass} \in \mathit{Ass}_{\mathsf{I}}, \\ \textit{for some } \pi \in \mathcal{A}^{*}, \\ \quad T_{\mathsf{I}}^{ass}(:\pi)(u) \textit{ is defined, and} \\ \quad u' = T_{\mathsf{I}}^{ass}(:\pi)(u) \end{array} \right\}.$$

I call $\mathsf{Co}_{\mathsf{I}}^{u}$ the *set of components of* u *in* I. Informally, Definition 56 limits the set of components of an entity u in I to those entities in U that are accessible from u by Σ terms defined on u. Note that the denotation of $:\pi$ is independent of the variable assignment in I, since there is no variable in Σ terms of that form.

As a final step towards the definition of the meaning of Σ formulae, I introduce a simple notational convention which I will need to fix the meaning of quantifiers:

Definition 57 *For each signature* Σ*, for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$*, for each* $ass \in \mathit{Ass}_{\mathsf{I}}$*, for each* $v \in \mathcal{VAR}$*, for each* $w \in \mathcal{VAR}$*, for each* $u \in \overline{\mathsf{U}}$*,*

$$ass\tfrac{u}{v}(w) = \begin{cases} u & \textit{if } v = w \\ ass(w) & \textit{otherwise.} \end{cases}$$

For each variable assignment in I, ass, $ass\frac{u}{v}$ is just like the function ass, except that it assigns entity u to variable v.

Everything is now in place to define the denotation function for Σ formulae, D_{I}^{ass}:

Definition 58 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$*, for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$*, for each* $ass \in \mathit{Ass}_{\mathsf{I}}$*,* D_{I}^{ass} *is the total function from* $\mathcal{D}^{\Sigma}$ *to the power set of* U *such that*

for each $\tau \in \mathcal{T}^{\Sigma}$*, for each* $\sigma \in \widehat{\mathcal{G}}$*,*

$$D_{\mathsf{I}}^{ass}(\tau \sim \sigma) = \left\{ u \in \mathsf{U} \;\middle|\; \begin{array}{l} T_{\mathsf{I}}^{ass}(\tau)(u) \textit{ is defined, and} \\ \widehat{\mathsf{S}}(T_{\mathsf{I}}^{ass}(\tau)(u)) \mathrel{\widehat{\sqsubseteq}} \sigma \end{array} \right\},$$

for each $\tau_1 \in \mathcal{T}^{\Sigma}$*, for each* $\tau_2 \in \mathcal{T}^{\Sigma}$*,*

$$D_{\mathsf{I}}^{ass}(\tau_1 \approx \tau_2) = \left\{ u \in \mathsf{U} \;\middle|\; \begin{array}{l} T_{\mathsf{I}}^{ass}(\tau_1)(u) \textit{ is defined,} \\ T_{\mathsf{I}}^{ass}(\tau_2)(u) \textit{ is defined, and} \\ T_{\mathsf{I}}^{ass}(\tau_1)(u) = T_{\mathsf{I}}^{ass}(\tau_2)(u) \end{array} \right\},$$

for each $\rho \in \mathcal{R}$, *for each* $x_1 \in \mathcal{VAR}$, ..., *for each* $x_{\mathcal{AR}(\rho)} \in \mathcal{VAR}$,

$$D_{\mathsf{I}}^{ass}(\rho(x_1, \ldots, x_{\mathcal{AR}(\rho)})) \\ = \left\{ u \in \mathsf{U} \,\middle|\, \left\langle ass(x_1), \ldots, ass(x_{\mathcal{AR}(\rho)}) \right\rangle \in \mathsf{R}(\rho) \right\},$$

for each $v \in \mathcal{VAR}$, *for each* $\delta \in \mathcal{D}^{\Sigma}$,

$$D_{\mathsf{I}}^{ass}(\exists v\ \delta) = \left\{ u \in \mathsf{U} \,\middle|\, \begin{array}{l} \textit{for some } u' \in \overline{\mathsf{Co}_{\mathsf{I}}^{u}}, \\ u \in D_{\mathsf{I}}^{ass\frac{u'}{v}}(\delta) \end{array} \right\},$$

for each $v \in \mathcal{VAR}$, *for each* $\delta \in \mathcal{D}^{\Sigma}$,

$$D_{\mathsf{I}}^{ass}(\forall v\ \delta) = \left\{ u \in \mathsf{U} \,\middle|\, \begin{array}{l} \textit{for each } u' \in \overline{\mathsf{Co}_{\mathsf{I}}^{u}}, \\ u \in D_{\mathsf{I}}^{ass\frac{u'}{v}}(\delta) \end{array} \right\},$$

for each $\delta \in \mathcal{D}^{\Sigma}$,

$$D_{\mathsf{I}}^{ass}(\neg\delta) = \mathsf{U} \backslash D_{\mathsf{I}}^{ass}(\delta),$$

for each $\delta_1 \in \mathcal{D}^{\Sigma}$, *for each* $\delta_2 \in \mathcal{D}^{\Sigma}$,

$$D_{\mathsf{I}}^{ass}([\delta_1 \wedge \delta_2]) = D_{\mathsf{I}}^{ass}(\delta_1) \cap D_{\mathsf{I}}^{ass}(\delta_2),$$

for each $\delta_1 \in \mathcal{D}^{\Sigma}$, *for each* $\delta_2 \in \mathcal{D}^{\Sigma}$,

$$D_{\mathsf{I}}^{ass}([\delta_1 \vee \delta_2]) = D_{\mathsf{I}}^{ass}(\delta_1) \cup D_{\mathsf{I}}^{ass}(\delta_2),$$

for each $\delta_1 \in \mathcal{D}^{\Sigma}$, *for each* $\delta_2 \in \mathcal{D}^{\Sigma}$,

$$D_{\mathsf{I}}^{ass}([\delta_1 \rightarrow \delta_2]) = (\mathsf{U} \backslash D_{\mathsf{I}}^{ass}(\delta)) \cup D_{\mathsf{I}}^{ass}(\delta_2), \textit{ and}$$

for each $\delta_1 \in \mathcal{D}^{\Sigma}$, *for each* $\delta_2 \in \mathcal{D}^{\Sigma}$,

$$D_{\mathsf{I}}^{ass}([\delta_1 \leftrightarrow \delta_2]) \\ = ((\mathsf{U} \backslash D_{\mathsf{I}}^{ass}(\delta_1)) \cap (\mathsf{U} \backslash D_{\mathsf{I}}^{ass}(\delta_2))) \cup (D_{\mathsf{I}}^{ass}(\delta_1) \cap D_{\mathsf{I}}^{ass}(\delta_2)).$$

I call D_{I}^{ass} the Σ *formula interpretation function with respect to* I *under a variable assignment in* I, *ass*.

Notice that conjunction, disjunction, implication and equivalence are interpreted classically; and negation, of course, corresponds to set complement. Notice also the crucial reference to the set comprising components and chains of components of u in I, $\overline{\mathsf{Co}_{\mathsf{I}}^{u}}$, in the interpretation of existential and universal quantification. This is where the restriction to quantification over components and chains of components is made. A Σ formula δ under existential quantification over the variable v, $\exists v\, \delta$, describes an entity u in a Σ interpretation I under the variable assignment *ass* in I, if there is at least one component (or chain of components), u', of u such that u is in the denotation of δ under the variable assignment ass' that we obtain from *ass* by letting it assign the component (or chain of components) u' to the variable v; in the case of universal quantification over the variable v, δ must describe u for all possible minimal changes to *ass* in which a component (or chain of components) of u is assigned to v.

To understand the denotation of sort assignment formulae and path equations, it is useful to distinguish cases when their Σ terms start with the colon from cases when they start with variables. If the Σ terms start with the colon, the denotation of the Σ formulae is just as in SRL. Sort assignment formulae, $\tau \sim \sigma$, in which τ starts with the colon denote the set of entities on which the Σ term τ is defined and leads to an entity which is in the denotation of σ. Path equations, $\tau_1 \approx \tau_2$, in which τ_1 and τ_2 start with the colon denote the set of entities on which τ_1 and τ_2 are defined and lead to the same entity.

The denotation of sort assignment formulae and path equations whose Σ terms start with a variable crucially depends on the chosen variable assignment in I. They either denote the entire universe or the empty set. Because the value of $T_{\mathsf{I}}^{ass}(\tau)(u)$, and the question of whether it is defined, is independent of u if τ starts with a variable, $\tau \sim \sigma$ denotes the entire universe if the function denoted by the string of symbols of the expanded attribute set following the variable is defined on the entity or chain assigned to the variable and leads to an entity or chain in the denotation of the symbol σ; and the empty set otherwise. A Σ formula, $\tau_1 \approx \tau_2$, denotes the entire universe if the functions denoted by the string of symbols of the expanded attribute set following the two variables are defined on the entities or chains that are assigned to the respective variables and lead to the same entity or chain; and the empty set otherwise.

This is best illustrated with an example. Recall the signature $\Sigma_{3.1}$ of Figure 3.1 (page 164), the $\Sigma_{3.1}$ interpretation $\mathsf{I}_{3.2}$ of Figure 3.2 (page 164), and the two variable assignments in $\mathsf{I}_{3.2}$, ass_0 and ass_1, given in (28) on page 163. If we choose the Σ formula interpretation function with respect to $\mathsf{I}_{3.2}$ under the variable assignment ass_0 to interpret the $\Sigma_{3.1}$ formula 'v_0OWNER $\sim$ *woman*', then we get $D^{ass_0}_{\mathsf{I}_{3.2}}(v_0\text{OWNER} \sim \mathit{woman}) = \emptyset$, because $ass_0(v_0) =$ Anne, and OWNER is not defined on Anne. If we choose the very same formula interpretation function with respect to the same interpretation $\mathsf{I}_{3.2} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$ to interpret the same formula, but this time under the variable assignment ass_1, then we get $D^{ass_1}_{\mathsf{I}_{3.2}}(v_0\text{OWNER} \sim \mathit{woman}) = \mathsf{U}$, because ass_1 assigns the third car to the variable v_0, and the owner of the third car is a woman, namely Anne. Similarly for path equations: If we choose the Σ formula interpretation function with respect to $\mathsf{I}_{3.2}$ under the variable assignment ass_0 to interpret the $\Sigma_{3.1}$ formula 'v_0OWNER $\approx$ v_1OWNER LIKES-BEST LIKES-BEST', then we get $D^{ass_0}_{\mathsf{I}_{3.2}}(v_0\text{OWNER} \approx v_1\text{OWNER LIKES-BEST LIKES-BEST}) = \emptyset$, because $ass_0(v_0) =$ Anne and $ass_0(v_1) =$ Anne. For ass_1, we get $D^{ass_1}_{\mathsf{I}_{3.2}}(v_0\text{OWNER} \approx v_1\text{OWNER LIKES-BEST LIKES-BEST}) = \mathsf{U}$, because $ass_1(v_0) =$ third car and $ass_1(v_1) =$ third car, and the owner of the third car (Anne) is the same person as the person that the person liked best by the owner of the third car (Manfred) likes best.

An n-ary relational Σ formula, $\rho(x_1, \ldots, x_n)$, is interpreted as the set of entities such that an n-tuple of entities and chains which is in the relation interpretation of ρ is assigned to the n-tuple of variables, $(x_1, \ldots, x_n)$, describing its arguments. Similar to sort assignment formulae and path equations with Σ terms beginning with variables, a relational Σ formula either denotes the entire universe or the empty set, depending on the variable assignment in I, *ass*. It denotes U if an n-ary tuple of entities or chains (or an n-ary tuple of entities and chains) which is in the relation interpretation is assigned to the tuple of variables that describe its arguments; and the empty set otherwise. This depends, of course, entirely on the choice of the variable assignment function in I, and on nothing else. The reason for this definition of the denotation of relational Σ formulae becomes clear when we consider how the meaning of relations is determined in grammars, and how the definition interacts with component quantification in Σ formulae without free variables.

In grammars, we will restrict ourselves to Σ formulae without free variables (Σ descriptions). In that case, the denotation of relational Σ formulae, sort assignment formulae and path equations involving variables becomes more transparent. When looking at the denotation of Σ descriptions, it is important to remember that in RSRL, just as in SRL, expressions denote a subset of the entities of the universe rather than truth values. This formalizes the intuition underlying both formalisms that their expressions describe entities and are, thus, true *of entities in an interpretation.*

As we have seen in the examples above, the denotation of Σ terms and Σ formulae may depend on the choice of the variable assignment in I under which they are interpreted in I. The important property of Σ descriptions is that their denotation in an interpretation is independent of the choice of the variable assignment in I under which they are interpreted in I. For each $ass \in Ass_{\mathsf{I}}$, they denote the same set of entities. Before I explain the denotation of Σ descriptions, especially the denotation of relations in Σ descriptions, I want to establish this result and define formally what a Σ description is. This is done in PROPOSITION 9, DEFINITION 59, and COROLLARY 2. PROPOSITION 9, which can be verified by a simple inductive proof, states that a Σ formula, δ, denotes the same set of entities under two different variable assignments in I if the two variable assignments agree on the free variables in δ. This observation will ultimately allow us to abstract away from variable assignments when I move from arbitrary Σ formulae to Σ formulae without free variables.

Proposition 9 *For each signature Σ, for each Σ interpretation I, for each $ass_1 \in Ass_{\mathsf{I}}$, for each $ass_2 \in Ass_{\mathsf{I}}$,*

for each $\tau \in \mathcal{T}^{\Sigma}$,

if for each $v \in FV(\tau)$, $ass_1(v) = ass_2(v)$
then $T_{\mathsf{I}}^{ass_1}(\tau) = T_{\mathsf{I}}^{ass_2}(\tau)$, and

for each $\delta \in \mathcal{D}^{\Sigma}$,

if for each $v \in FV(\delta)$, $ass_1(v) = ass_2(v)$
then $D_{\mathsf{I}}^{ass_1}(\delta) = D_{\mathsf{I}}^{ass_2}(\delta)$.

If two variable assignments in I agree on the variables that occur free in a given Σ term, τ, then τ denotes the same partial function under the Σ term interpretation function with respect to I under both variable assignments in I. Similarly, if two variable assignments in I agree on the variables that occur free in a given Σ formula, δ, then δ denotes the same set of entities of the universe under the Σ formula interpretation function with respect to I under both variable assignments in I.

I distinguish the set of Σ formulae without free variables by a special symbol, $\mathcal{D}_0^\Sigma$:

Definition 59 *For each signature* Σ,

$$\mathcal{D}_0^\Sigma = \{\delta \in \mathcal{D}^\Sigma \mid FV(\delta) = \emptyset\}.$$

I call each element of $\mathcal{D}_0^\Sigma$ a Σ *description*, and each subset of $\mathcal{D}_0^\Sigma$ a Σ *theory*. When working with a fixed signature, I will henceforth refer to terms, formulae and descriptions. As a corollary of PROPOSITION 9, we get the following result for the meaning of Σ descriptions:

Corollary 2 *For each signature* Σ*, for each* $\delta \in \mathcal{D}_0^\Sigma$*, for each* Σ *interpretation* I*, for each* $ass_1 \in Ass_\mathsf{I}$*, for each* $ass_2 \in Ass_\mathsf{I}$,

$$D_\mathsf{I}^{ass_1}(\delta) = D_\mathsf{I}^{ass_2}(\delta).$$

According to COROLLARY 2 the meaning of Σ descriptions is independent from the initial choice of the variable assignment in I. If there are variables in a Σ description δ, they are bound by a quantifier. Therefore, when a Σ term in δ in which a variable v occurs is interpreted on an entity u in I, the value of the original variable assignment in I is changed on v, and a component of u is assigned to v; what the initial variable assignment in I assigns to v never plays any role. With COROLLARY 2 in hand, I return to the discussion of the denotation of Σ formulae, now focusing on Σ descriptions.

In a complex Σ description, an n-ary relational Σ formula, $\rho(x_1, \ldots, x_n)$, denotes the set of entities whose n-tuples of components or chains of components, as determined by the quantification, are in the denotation of the relation. If all variables in a relational Σ formula are existentially bound, it denotes the set of entities, u, which have an n-tuple of components or chains of components in I that are in the denotation of the relation; if they are all universally bound, it denotes the set of entities, u, all n-tuples of whose components and chains of components in I are in the denotation of the relation. These properties are exploited in defining the meaning of the relation symbols. Note that in a Σ interpretation, the denotation of a relation ρ, $\mathsf{R}(\rho)$, is not in any way predefined. Only the arity of each relation symbol is given by the signature. The meaning of the relation symbols is determined by members of the theory of the grammar, i.e., by appropriate Σ descriptions. The idea is that a relation holds of all appropriate components of all entities in a Σ interpretation according to descriptions in the theory of a grammar. Although there are other possibilities, for each relation symbol in the signature, the theory of a grammar usually contains one bi-implicational description that fixes the meaning of the relation symbol.

To illustrate this, I modify the car scenario signature: I eliminate the attribute LIKES-BEST, and let the new attribute PASSENGERS be appropriate to *car*. The sort *list* is appropriate for $\langle car, \text{PASSENGERS}\rangle$, and the usual sort hierarchy, attributes, and appropriateness conditions for lists are added. The relation that I am interested in is `member`, which is of arity 2. The new signature, $\Sigma_{3.3}$, is shown in Figure 3.3. With $\Sigma_{3.3}$, I can talk about the familiar two kinds of cars, their drivers, owners, and a list of passengers in those cars. With the relation `member`, I want to talk about the members of the passenger list. For that purpose, the symbol `member` must receive the appropriate interpretation, lest any pair of entities and chains can be in its denotation. Now consider the bi-implicational description of the `member` relation in (29):

(29) $$\forall v_1\, \forall v_2 \left[\mathtt{member}(v_1, v_2) \leftrightarrow \left[\begin{array}{l} v_1 \approx v_2\text{FIRST} \\ \vee\, \exists v_3\, [v_3 \approx v_2\text{REST} \wedge \mathtt{member}(v_1, v_3)] \end{array}\right]\right]$$

In any $\Sigma_{3.3}$ interpretation, the description (29) describes exactly each entity which can be characterized as follows: All pairs of components of the entity are in the `member` relation if and only if either the first component is the first element on the list that is the second component, or there is a third component of the entity that equals the tail of the list that is the second component, and the first component and the third component are in the `member` relation (or both defining conditions hold). But that means that if (29) describes all entities in an interpretation, then in that interpretation `member` means what we intuitively want to express with it. For example, consider the $\Sigma_{3.3}$ interpretation $\mathsf{I}_{3.4}$ in Figure 3.4. It is a small scenario in which Mike drives the third car, which is owned by Anne. The persons in the car (the passengers) are Anne, Manfred and Mike. Manfred and Mike are men, and Anne is a woman. (29) describes each entity in $\mathsf{I}_{3.4}$. It is trivially true of Anne, Mike, Manfred, and the empty list, L4, because they do not have proper components and they are not in the member relation

Figure 3.3: A car scenario signature with a relation

top

list

elist

nelist FIRST *top*
REST *list*

car OWNER *person*
DRIVER *person*
PASSENGERS *list*

vw

bmw

person

man

woman

and $\mathcal{R} = \{\texttt{member}\}$, $\mathcal{AR} = \{\langle \texttt{member}, 2\rangle\}$.

Figure 3.4: An interpretation of the car scenario signature with a relation
Let $\mathsf{I}_{3.4} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R}\rangle$, with:

$\mathsf{U} = \{\text{Anne}, \text{Manfred}, \text{Mike}, \text{third car}, \text{L1}, \text{L2}, \text{L3}, \text{L4}\}$,

$$\mathsf{S} = \left\{ \begin{array}{l} \langle \text{Anne}, \textit{woman}\rangle, \langle \text{Manfred}, \textit{man}\rangle, \langle \text{Mike}, \textit{man}\rangle, \langle \text{third car}, \textit{vw}\rangle, \\ \langle \text{L1}, \textit{nelist}\rangle, \langle \text{L2}, \textit{nelist}\rangle, \langle \text{L3}, \textit{nelist}\rangle, \langle \text{L4}, \textit{elist}\rangle \end{array} \right\},$$

$$\mathsf{A} = \left\{ \begin{array}{l} \langle \text{DRIVER}, \{\langle \text{third car}, \text{Mike}\rangle\}\rangle, \\ \langle \text{OWNER}, \{\langle \text{third car}, \text{Anne}\rangle\}\rangle, \\ \langle \text{PASSENGERS}, \{\langle \text{third car}, \text{L1}\rangle\}\rangle, \\ \langle \text{FIRST}, \{\langle \text{L1}, \text{Mike}\rangle, \langle \text{L2}, \text{Anne}\rangle, \langle \text{L3}, \text{Manfred}\rangle\}\rangle, \\ \langle \text{REST}, \{\langle \text{L1}, \text{L2}\rangle, \langle \text{L2}, \text{L3}\rangle, \langle \text{L3}, \text{L4}\rangle\}\rangle, \end{array} \right\}, \text{ and}$$

$$\mathsf{R}(\texttt{member}) = \left\{ \begin{array}{l} \langle \text{Mike}, \text{L1}\rangle, \langle \text{Anne}, \text{L1}\rangle, \langle \text{Anne}, \text{L2}\rangle, \\ \langle \text{Manfred}, \text{L1}\rangle, \langle \text{Manfred}, \text{L2}\rangle, \langle \text{Manfred}, \text{L3}\rangle \end{array} \right\}.$$

with themselves. It is true of the list L3, because the pair $\langle \text{Manfred}, \text{L3}\rangle$ is in $\mathsf{R}(\texttt{member})$. It is true of the list L2, because $\langle \text{Anne}, \text{L2}\rangle \in \mathsf{R}(\texttt{member})$,

$\langle \text{Manfred}, \text{L2}\rangle \in \mathsf{R}(\texttt{member})$, and $\langle \text{Manfred}, \text{L3}\rangle \in \mathsf{R}(\texttt{member})$. It is true of L1, because $\langle \text{Mike}, \text{L1}\rangle \in \mathsf{R}(\texttt{member})$, $\langle \text{Manfred}, \text{L1}\rangle \in \mathsf{R}(\texttt{member})$, $\langle \text{Anne}, \text{L1}\rangle \in \mathsf{R}(\texttt{member})$, $\langle \text{Anne}, \text{L2}\rangle \in \mathsf{R}(\texttt{member})$, $\langle \text{Manfred}, \text{L2}\rangle \in \mathsf{R}(\texttt{member})$, and $\langle \text{Manfred}, \text{L3}\rangle \in \mathsf{R}(\texttt{member})$. Finally, it is true of the third car for the same reason for which it is true of L1. In short, in $\mathsf{I}_{3.4}$ the `member` relation has the intuitively right meaning. Therefore, descriptions using `member` that are interpreted with respect to $\mathsf{I}_{3.4}$ under some variable assignment, *ass*, also have the intuitively right meaning relative to our little scenario. For example, the third car is in the set of entities $D^{ass}_{\mathsf{I}_{3.4}}([:\sim vw \wedge \exists v_1 \exists v_2 \, [v_2 \approx :\text{PASSENGERS} \wedge \texttt{member}(v_1, v_2)]])$, because there are indeed passengers in the third car, and the third car is also in $D^{ass}_{\mathsf{I}_{3.4}}([:\sim vw \wedge \exists v_1 \exists v_2 \, [:\approx v_1 \wedge v_2 \approx :\text{PASSENGERS} \wedge \neg \, \texttt{member}(v_1, v_2)]])$, because the third car is not a passenger of itself. '$[:\sim vw \wedge \exists v_1 \, [v_1 \approx :\text{PASSENGERS} \wedge \forall v_2 \, [\texttt{member}(v_2, v_1) \rightarrow v_2 \sim man]]]$', however, does not describe the third car in $\mathsf{I}_{3.4}$, because every passenger of the third car is not a man.

Now consider the $\Sigma_{3.3}$ interpretation that is just like $\mathsf{I}_{3.4}$ but with the relation interpretation R', which is exactly like R but with $\langle \text{Anne}, \text{L1}\rangle$ missing from the denotation of `member`. Call the resulting $\Sigma_{3.3}$ interpretation $\mathsf{I}'_{3.4}$. In $\mathsf{I}'_{3.4}$, (29) no longer describes every entity. It still describes Anne, Manfred, Mike, the empty list, L4, the nonempty list L3, and the nonempty list L2. Since $\langle \text{Anne}, \text{L1}\rangle$ is not in $\mathsf{R}'(\texttt{member})$, it no longer describes L1 and the third car. It follows that the denotation of some descriptions using the `member` relation receive an intuitively wrong denotation when interpreted in $\mathsf{I}'_{3.4}$. For example, for any variable assignment *ass* in $\mathsf{I}'_{3.4}$, $D^{ass}_{\mathsf{I}'_{3.4}}([:\sim vw \wedge \exists v_1 \, [v_1 \approx :\text{PASSENGERS} \wedge \forall v_2 \, [\texttt{member}(v_2, v_1) \rightarrow v_2 \sim man]]])$ describes the third car, although Anne is among the passengers, and Anne is a woman.

The denotation of relational formulae in an interpretation is the most complex aspect of the semantics of the formal languages of RSRL, because their denotation only makes intuitive sense relative to interpretations in which the descriptions that fix the meaning of the relations describe every entity. Moreover, the variables in the formula must be bound by quantification, because otherwise the choice of the variable assignment alone determines their denotation in the interpretation, just as in the examples of sort assignment formulae and path equations with free variables above.

As with the denotation of relational formulae, understanding the denotation in interpretations of sort assignment formulae and path equations becomes very easy when we limit our attention to formulae that are part of descriptions. If the variable in the Σ term, τ, of a sort assignment formula, $\tau \sim \sigma$ is existentially bound, the sort assignment formula denotes the set of entities which have at least one component or chain of components, c, such that the function denoted by the string of symbols from the expanded attribute set following the variable in τ is defined on c and yields an entity or chain in the denotation of σ. To come back to an earlier example from page 171, if the $\Sigma_{3.1}$ formula '$v_0\text{OWNER} \sim \mathit{woman}$' occurs as a subformula of the $\Sigma_{3.1}$ description '$\exists v_0\, v_0\text{OWNER} \sim \mathit{woman}$', it describes exactly the third car in the $\Sigma_{3.1}$ interpretation $\mathsf{I}_{3.2}$, irrespective of the variable assignment in $\mathsf{I}_{3.2}$ under which it is evaluated, i.e., $D^{ass_0}_{\mathsf{I}_{3.2}}(\exists v_0\, v_0\text{OWNER} \sim \mathit{woman}) = D^{ass_1}_{\mathsf{I}_{3.2}}(\exists v_0\, v_0\text{OWNER} \sim \mathit{woman}) = \{\text{third car}\}$, where ass_0 and ass_1 are as defined in (28). The third car is the only entity in $\mathsf{I}_{3.2}$ that has a component whose owner is a woman, and that component is the third car itself. If the variables in the Σ terms τ_1 and τ_2 of a path equation, $\tau_1 \approx \tau_2$, are existentially bound, that path equation denotes the set of entities such that there is at least one component or chain of components, c_1, on which the function denoted by the string of symbols from the expanded attribute set following the variable in τ_1 is defined, there is at least one component or chain of components, c_2, on which the function denoted by the string of symbols from the expanded attribute set following the variable in τ_2 is defined, and the two functions have the same entity or chain as value on c_1 and c_2, respectively. Corresponding observations hold for universal quantification.

I have shown that determining the denotation of Σ descriptions in an interpretation is independent of the variable assignments in the interpretation. Therefore, the Σ description denotation function in an interpretation does not need to refer to variable assignments in the interpretation:

Definition 60 *For each signature Σ, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, D_I is the total function from $\mathcal{D}^\Sigma_0$ to the power set of U such that for each $\delta \in \mathcal{D}^\Sigma_0$,*

$$D_\mathsf{I}(\delta) = \left\{ u \in \mathsf{U} \;\middle|\; \begin{array}{l} \textit{for each } ass \in Ass_\mathsf{I}, \\ u \in D^{ass}_\mathsf{I}(\delta) \end{array} \right\}.$$

I call D_I the Σ *description denotation function with respect to* I. D_I abstracts away from particular variable assignments in I. As a result, description denotation in RSRL is completely parallel to description denotation in SRL, and the following definitions of theory denotations, grammars, models of grammars and exhaustive models of grammars are also parallel to the corresponding definitions in SRL. With D_{I} in hand it is straightforward to define the denotation of Σ theories:

Definition 61 *For each signature* Σ*, for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$*,* Θ_{I} *is the total function from the power set of* $\mathcal{D}_0^{\Sigma}$ *to the power set of* U *such that for each* $\theta \subseteq \mathcal{D}_0^{\Sigma}$*,*

$$\Theta_{\mathsf{I}}(\theta) = \left\{ u \in \mathsf{U} \;\middle|\; \begin{array}{l} \textit{for each } \delta \in \theta, \\ u \in D_{\mathsf{I}}(\delta) \end{array} \right\}.$$

I call Θ_{I} the *theory denotation function with respect to* I. Under Θ_{I} a Σ theory, θ, denotes the set of entities of the universe which are in the denotation of all Σ descriptions in θ. Just as in SRL, grammars are pairs of a signature Σ and a set of Σ descriptions:

Definition 62 Γ *is a* ***grammar*** *iff*

> Γ *is a pair* $\langle \Sigma, \theta \rangle$*,*
>
> Σ *is a signature, and*
>
> $\theta \subseteq \mathcal{D}_0^{\Sigma}$.

A grammar is a signature, Σ, together with a set of Σ descriptions, θ. As we know from SRL, not every Σ interpretation is an interesting interpretation of a grammar, $\langle \Sigma, \theta \rangle$. For example, the ungrammatical sentence *Sandy walks Kim Fido* could very well be in an interpretation of the grammar of Pollard and Sag 1994, but that interpretation does certainly not reflect the predictions the authors make about English. In RSRL, we have seen an additional technical reason for why arbitrary interpretations cannot be what a grammarian has in mind. The relational symbols only obtain the denotation that is expressed in the descriptions in the theory of the grammar that are meant to fix their meaning if those descriptions are true of every entity in the interpretation. In interpretations in which that is not the case, descriptions that employ the relational symbols might not have the intended meaning. Consequently, the models of a grammar are a first approximation to the interpretations in which linguists are interested when they write grammars:

Definition 63 *For each grammar* $\Gamma = \langle \Sigma, \theta \rangle$*, for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$*,*

I *is a* Γ ***model*** *iff* $\Theta_{\mathsf{I}}(\theta) = \mathsf{U}$.

In a Γ model, each description in the theory is true of each entity in the universe of the interpretation. Informally, the grammar denotes the whole universe of the interpretation. As pointed out before, the notion of a model formalizes the idea that with a grammar linguists want to characterize natural language as a set of entities which, in all their components, obey every principle of the grammar. However, as discussed in the previous chapter, arbitrary models cannot characterize natural languages adequately. In a sense, a model of a grammar might not be big enough to be the intended model, since it might not contain a configuration under an entity to which there exists an indiscernible configuration under some other entity in a different model of the grammar: One model of a grammar might lack a sentence that is contained in a different model of the grammar. A trivial example for an uninteresting model of a grammar is the model where U is the empty set, if the grammar in question has models with a nonempty universe. A less trivial example of a model that is too small is a model of Pollard and Sag's (1994) grammar which contains exactly the sentence *Kim likes bagels* and nothing else.

In SRL, the class of exhaustive models delimits the models that are big enough in the sense that they contain indiscernible counterparts to all possible configurations under an entity that can occur in some model of the grammar. The goal is to transfer that idea to RSRL. If this can be done then King's three conditions that say what it means for a grammar to be true of a natural language apply to RSRL grammars. This would explicate the meaning of a grammar, and I could define the tokens and types of a natural language as they are defined in SRL. From there it is only a small step to reconstruct the other accounts of the meaning of HPSG grammars of Pollard and Sag 1994 and of Pollard 1999.

Since all relevant notions of SRL have a counterpart in RSRL, it is easy now to rephrase the characterization of the class of exhaustive models that is based on description denotation alone in RSRL (cf. COROLLARY 1):

Definition 64 *For each grammar* $\Gamma = \langle \Sigma, \theta \rangle$, *for each* Σ *interpretation* I,

I *is an* ***exhaustive*** Γ ***model*** *iff*

I *is a* Γ *model, and*
for each $\theta' \subseteq \mathcal{D}_0^{\Sigma}$, *for each* Σ *interpretation* I',

$$\textit{if } \mathsf{I}' \textit{ is a } \Gamma \textit{ model and } \Theta_{\mathsf{I}'}(\theta') \neq \emptyset \textit{ then } \Theta_{\mathsf{I}}(\theta') \neq \emptyset.$$

An exhaustive Γ model, I, is a Γ model such that any theory which has a nonempty denotation in any other Γ model also has a nonempty denotation in I. Recall that an exhaustive Γ model contains indiscernible counterparts to all configurations under an entity that occur in some model of Γ. Among those entities are, of course, all sentences that are licensed by Γ. In linguistic terms this means that a grammar is true of a natural language if the natural language is an exhaustive model of the grammar.

The relevant question to ask now is if all RSRL grammars have exhaustive models. King has shown that every SRL grammar has an exhaustive model; and in Section 2.2.2.4, I spelled out a proposal by Pollard 1999 of how to construct a canonical exhaustive model for each SRL grammar. Note that every SRL grammar can trivially be stated as an RSRL grammar; and that subset of RSRL grammars will thus have exhaustive models. However, the syntactic and semantic additions to SRL that are contained in RSRL might in principle cause RSRL grammars not to have exhaustive models in the general case. In Section 3.1.2, I will outline a proof of the following theorem that lays these apprehensions to rest:

Theorem 2 *For each signature Σ, for each Σ theory θ, there exists a Σ interpretation I such that I is an exhaustive $\langle\Sigma, \theta\rangle$ model.*

As a corollary of Theorem 2, we get immediately that if an RSRL grammar has a nonempty model, it also has a nonempty exhaustive model. In summation, a non-vacuous grammar has a linguistically sensible meaning in the class of its exhaustive models. King's explanation of the meaning of grammars, summarized in Section 2.2.2.2, *mutatis mutandis*, carries over to RSRL.

In Section 2.2.3, I have argued that it follows from the omission of true relations and of component quantification from the logical languages of SRL that SRL grammars are not capable of expressing many typical principles of HPSG 94 grammars directly and faithfully. The extension to SRL that I proposed in this section comprises a number of additional new devices in its formal apparatus besides relations and component quantification. Most notably, I introduced variables, variable assignments in interpretations, and sets of components of entities in interpretations, and I made a distinction between formulae with free occurrences of variables and descriptions, in which all variables are bound. Moreover, for independent reasons I generalized component quantification to quantification over components and chains of components.

It is the reference to components and chains of components of entities in the definition of the meaning of the two quantifiers of RSRL (DEFINITION 58) where our quantification differs from the quantification of first-order logic. As a consequence of that definition, the relations of RSRL grammars are relations between the components (and chains of components) of entities, because the meaning of the relation symbols of grammars is typically fixed by principles of grammar of a form that I have exemplified with the `member` relation in (29), and because the meaning of relation principles of that form crucially depends on universal component quantification. In an example, we have seen that this approach results in an intuitively correct meaning of relations and negated relational formulae in models of RSRL grammars. The relational extension of SRL that I propose is thus intimately linked to component quantification, and component quantification is the most important innovation that distinguishes RSRL from other extensions of feature logic.

3.1.2 Constructing a Canonical Exhaustive Model

One method to prove that every grammar has an exhaustive model is to construct a particular model of each grammar, and to show that this model is an exhaustive model. In this section, I will provide such a construction and outline the structure of a proof that shows that the models that I will construct have the desired properties.[4] I will strictly focus on those aspects of the construction and the proof that are relevant in order to understand how the different perspectives on the meaning of an HPSG 94 grammar can be captured in RSRL. I omit all mathematical details that are not essential for that purpose and might only distract from the general line of the argument. However, to satisfy the need for explicitness, the details of the proof are provided separately in Appendix A and can be checked there.

In DEFINITION 64, I defined the exhaustive models of a grammar using the properties of the denotation of the theory of the grammar with respect to an exhaustive model relative to the denotation of the theory with respect to other models of the grammar. My proof of THEOREM 2 about the existence of an exhaustive model of each grammar, however, exploits the algebraic

[4]The proof is due to Richter and King 1997.

property of two entities in two interpretations being congruent, and it ultimately proves the theorem by showing that for each entity in some model of the grammar, there is an entity in the constructed model that is congruent to it. Therefore, I need a notion of congruence for pairs of entities and interpretations, and I must show that the class of exhaustive models of a grammar can also be characterized algebraically. Section 3.1.2.1 will provide the necessary definitions and results. It will also show that the notion of token does not change compared to SRL.

Section 3.1.2.2 sets the stage for defining the canonical exhaustive models of grammars. First, I define a set of structures that bear an obvious resemblance to 94 abstract feature structures. The only difference is that they have some additional structure that represents relations in feature structures. To distinguish the new kind of structures terminologically from simple abstract feature structures, I will call them *morphs*. An inductive definition of morph satisfaction, defining which set of morphs satisfies a given formula, prepares for the definition of morph admission, which determines which set of morphs a given set of descriptions admits. Intuitively, a morph is admitted by a set of descriptions if each (abstract) node of the morph satisfies each description in the theory. Morph admission will later serve two purposes: In Section 3.1.2.3, the set of morphs admitted by a set of descriptions is crucial for defining the canonical exhaustive model of each grammar. In Section 3.1.3, we will see that it also provides an explication of the object types of Pollard and Sag 1994. The morphs admitted by the theory of a grammar can be understood as mathematical representations of the equivalence classes of the tokens of the natural language that the grammar describes.

Section 3.1.2.3 outlines the most important steps of the proof that shows that the canonical models of each grammar that I define on the basis of morph admission are indeed exhaustive models. An abstraction function maps congruent entities in interpretations to identical morphs, and I will show that for each entity in a model of a grammar, the canonical model of the grammar contains an entity that abstracts to the same morph. It follows immediately that the canonical model simulates each other model of the grammar.

3.1.2.1 Congruence and Indiscernibility

In RSRL, a congruence from one entity in one interpretation to another entity in another interpretation must not only preserve the attributes and the species of the components of the first entity; it must also preserve the relations that hold between the components and chains of components of the first entity. Except for this extension, DEFINITION 65 mirrors the DEFINITION 25 of SRL congruences.

Definition 65 *For each signature Σ, for each Σ interpretation $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1 \rangle$, for each $u_1 \in \mathsf{U}_1$, for each Σ interpretation $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2 \rangle$, for each $u_2 \in \mathsf{U}_2$,*

> *f is a* ***congruence*** *from $\langle u_1, \mathsf{I}_1 \rangle$ to $\langle u_2, \mathsf{I}_2 \rangle$ in Σ*
>
> *iff f is a bijection from $\overline{\mathsf{Co}_{\mathsf{I}_1}^{u_1}}$ to $\overline{\mathsf{Co}_{\mathsf{I}_2}^{u_2}}$,*
>
> *for each $u \in \overline{\mathsf{Co}_{\mathsf{I}_1}^{u_1}}$, $\widehat{\mathsf{S}_1}(u) = \widehat{\mathsf{S}_2}(f(u))$,*
>
> *for each $\alpha \in \widehat{\mathcal{A}}$, for each $u \in \overline{\mathsf{Co}_{\mathsf{I}_1}^{u_1}}$,*
>
> *$\widehat{\mathsf{A}_1}(\alpha)(u)$ is defined iff $\widehat{\mathsf{A}_2}(\alpha)(f(u))$ is defined, and*
> *if $\widehat{\mathsf{A}_1}(\alpha)(u)$ is defined then $f(\widehat{\mathsf{A}_1}(\alpha)(u)) = \widehat{\mathsf{A}_2}(\alpha)(f(u))$,*
>
> *for each $\rho \in \mathcal{R}$, for each $o_1 \in \overline{\mathsf{Co}_{\mathsf{I}_1}^{u_1}}, \ldots,$ for each $o_{\mathcal{AR}(\rho)} \in \overline{\mathsf{Co}_{\mathsf{I}_1}^{u_1}}$,*
>
> *$\langle o_1, \ldots, o_{\mathcal{AR}(\rho)} \rangle \in \mathsf{R}_1(\rho)$ iff $\langle f(o_1), \ldots, f(o_{\mathcal{AR}(\rho)}) \rangle \in \mathsf{R}_2(\rho)$, and*
>
> *$f(u_1) = u_2$.*

Note that if all entities in I_1 are components of u_1 and all entities in I_2 are components of u_2, then a congruence is an isomorphism.

A first entity in a first interpretation and a second entity in a second interpretation are *congruent* iff there is a (quasi-) species, (quasi-) attribute and relation preserving bijection from the components and chains of components of the first entity in the first interpretation to the components and chains of components of the second entity in the second interpretation such that the bijection maps the first entity to the second entity:

Definition 66 *For each signature Σ, for each Σ interpretation $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1 \rangle$, for each $u_1 \in \mathsf{U}_1$, for each Σ interpretation $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2 \rangle$, for each $u_2 \in \mathsf{U}_2$,*

> *$\langle u_1, \mathsf{I}_1 \rangle$ and $\langle u_2, \mathsf{I}_2 \rangle$ are* ***congruent in*** *Σ*
>
> *iff for some f, f is a congruence from $\langle u_1, \mathsf{I}_1 \rangle$ to $\langle u_2, \mathsf{I}_2 \rangle$ in Σ.*

Two entities in two interpretations of a signature Σ are called *indiscernible in* Σ if there is no Σ description that can distinguish between the two entities. There is no description that describes one of the entities but not the other:

Definition 67 *For each signature Σ, for each Σ interpretation $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1 \rangle$, for each $u_1 \in \mathsf{U}_1$, for each Σ interpretation $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2 \rangle$, for each $u_2 \in \mathsf{U}_2$,*

> *$\langle u_1, \mathsf{I}_1 \rangle$ and $\langle u_2, \mathsf{I}_2 \rangle$ are* ***indiscernible in*** *Σ*
> *iff for each $\delta \in \mathcal{D}_0^{\Sigma}$, $u_1 \in D_{\mathsf{I}_1}(\delta)$ iff $u_2 \in D_{\mathsf{I}_2}(\delta)$.*

Indiscernibility and congruence turn out to be equivalent:

Proposition 10 *For each signature Σ, for each Σ interpretation $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1 \rangle$, for each $o_1 \in \mathsf{U}_1$, for each Σ interpretation $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2 \rangle$, for each $o_2 \in \mathsf{U}_2$,*

> *$\langle o_1, \mathsf{I}_1 \rangle$ and $\langle o_2, \mathsf{I}_2 \rangle$ are congruent in Σ iff $\langle o_1, \mathsf{I}_1 \rangle$ and $\langle o_2, \mathsf{I}_2 \rangle$ are indiscernible in Σ.*

If there is a congruence in Σ from one entity in one Σ interpretation to another entity in another Σ interpretation, then there is no Σ description that describes one entity but not the other entity. And if two entities in two Σ interpretations are indistinguishable by Σ descriptions, then I can construct a congruence in Σ from one entity and its (chains of) components in its interpretation to the other entity and its (chains of) components in the other interpretation. The basic idea is to map each component of the first entity, u_1, that is the value of interpreting some Σ term τ that starts with the colon on u_1 to the component of the second entity, u_2, that is the value of interpreting the same term τ on u_2, and to generalize this definition to chains of components.

Since with respect to indiscernibility and congruence entities in interpretations behave just like entities in SRL interpretations, it is possible to perceive the configurations of entities under entities that exist in exhaustive models of a grammar as tokens of a natural language. To be more precise, according to the view of King the entities in the particular exhaustive model of a correct grammar that is the natural language constitute a system of the actual and non-actual tokens of the language.

From the notion of two entities in two interpretations being congruent, I obtain the definition of one interpretation *simulating* another interpretation:

Definition 68 *For each signature* Σ*, for each* Σ *interpretation* $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1 \rangle$*, for each* Σ *interpretation* $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2 \rangle$,

I_1 ***simulates*** I_2 ***in*** Σ

iff for each $u_2 \in \mathsf{U}_2$*, for some* $u_1 \in \mathsf{U}_1$*,* $\langle u_1, \mathsf{I}_1 \rangle$ *and* $\langle u_2, \mathsf{I}_2 \rangle$ *are congruent in* Σ*.*

A Σ interpretation I_1 simulates Σ interpretation I_2 in Σ just in case every entity in I_2 has a congruent counterpart in I_1.

Since indiscernibility and congruence are equivalent, the simulation of models of grammars can alternatively be expressed in terms of the denotation of sets of descriptions in interpretations of the same signature. One Σ model I of a grammar, $\langle \Sigma, \theta \rangle$, simulates another Σ model I' of the grammar in Σ only if each Σ theory that describes something in I' also describes something in I. Analogous to Proposition 2, I formulate this as follows:

Proposition 11 *For each signature* Σ*, for each* $\theta \subseteq \mathcal{D}_0^\Sigma$*, for each* Σ *interpretation* I*,*

for each Σ *interpretation* I'*,*

if I' *is a* $\langle \Sigma, \theta \rangle$ *model then* I *simulates* I' *in* Σ

iff for each $\theta' \subseteq \mathcal{D}_0^\Sigma$*, for each* Σ *interpretation* I'*,*

if I' *is a* $\langle \Sigma, \theta \rangle$ *model and* $\Theta_{\mathsf{I}'}(\theta') \neq \emptyset$ *then* $\Theta_{\mathsf{I}}(\theta') \neq \emptyset$*.*

With Proposition 11, the exhaustive models of a grammar can—alternatively to Definition 64—be characterized algebraically:

Theorem 3 *For each signature* Σ*, for each* $\theta \subseteq \mathcal{D}_0^\Sigma$*, for each* Σ *interpretation* I*,*

I *is an exhaustive* $\langle \Sigma, \theta \rangle$ *model iff*

I *is a* $\langle \Sigma, \theta \rangle$ *model, and*
for each Σ *interpretation* I'*,*
if I' *is a* $\langle \Sigma, \theta \rangle$ *model then* I *simulates* I' *in* Σ*.*

Thus, as far as the characterization of exhaustive models of a grammar is concerned, the analogy between SRL and RSRL is complete.

3.1.2.2 Morphs and Morph Admission

Before I can define a canonical model for each grammar, I need to provide the mathematical structures from which I can later construct the entities in the canonical model of a grammar in such a way that the canonical model contains congruent counterparts to all configurations under an entity that may occur in some model of the grammar. In Section 2.2.2.4, (45), I have already presented a similar construction based on a proposal by Pollard 1999 using Pollard abstract feature structures. There, the entities in the universe of the constructed model are Pollard abstract feature structures with distinguished nodes, and the species assignment function and the attribute interpretation function derive from the species of the distinguished nodes and the attributes that are defined on them. The construction in Section 3.1.2.3 will be similar in many respects, but it will be based on a variant of the more standard Moshier representation of abstract feature structures with a relational extension. To distinguish the new structures terminologically from abstract feature structures without relations, I call them *morphs*:

Definition 69 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$,

μ *is a* ***morph in*** Σ

iff μ *is a quadruple* $\langle \beta, \varrho, \lambda, \xi \rangle$,

$\beta \subseteq \mathcal{A}^*$,

$\varepsilon \in \beta$,

for each $\pi \in \mathcal{A}^*$, *for each* $\alpha \in \mathcal{A}$,

if $\pi\alpha \in \beta$ *then* $\pi \in \beta$,

ϱ *is an equivalence relation over* β,

for each $\pi_1 \in \mathcal{A}^*$, *for each* $\pi_2 \in \mathcal{A}^*$, *for each* $\alpha \in \mathcal{A}$,

if $\pi_1\alpha \in \beta$ *and* $\langle \pi_1, \pi_2 \rangle \in \varrho$ *then* $\langle \pi_1\alpha, \pi_2\alpha \rangle \in \varrho$,

λ *is a total function from* β *to* $\mathcal{S}$,

for each $\pi_1 \in \mathcal{A}^*$, *for each* $\pi_2 \in \mathcal{A}^*$,

if $\langle \pi_1, \pi_2 \rangle \in \varrho$ *then* $\lambda(\pi_1) = \lambda(\pi_2)$,

for each $\pi \in \mathcal{A}^*$, *for each* $\alpha \in \mathcal{A}$, *if* $\pi\alpha \in \beta$ *then* $\mathcal{F}\langle\lambda(\pi), \alpha\rangle$ *is defined and* $\lambda(\pi\alpha) \sqsubseteq \mathcal{F}\langle\lambda(\pi), \alpha\rangle$,

for each $\pi \in \mathcal{A}^*$, *for each* $\alpha \in \mathcal{A}$,

if $\pi \in \beta$ *and* $\mathcal{F}\langle\lambda(\pi), \alpha\rangle$ *is defined then* $\pi\alpha \in \beta$,

$\xi \subseteq \mathcal{R} \times \overline{\beta}^*$,

for each $\rho \in \mathcal{R}$, *for each* $\pi_1 \in \overline{\beta}, \ldots,$ *for each* $\pi_n \in \overline{\beta}$,

if $\langle\rho, \pi_1, \ldots, \pi_n\rangle \in \xi$ *then* $n = \mathcal{AR}(\rho)$, *and*

for each $\rho \in \mathcal{R}$, *for each* $\pi_1 \in \overline{\beta}$, …, *for each* $\pi_n \in \overline{\beta}$, *for each* $\pi'_1 \in \overline{\beta}$, …, *for each* $\pi'_n \in \overline{\beta}$,

if $\langle\rho, \pi_1, \ldots, \pi_n\rangle \in \xi$, *and*
for each $i \in \{1, \ldots, n\}$,
$\pi_i \in \beta$ *and* $\langle\pi_i, \pi'_i\rangle \in \varrho$, *or*
for some $m \in \mathbb{N}$,
$\pi_i \in \beta^*$,
$\pi_i = \langle\pi_{i_1}, \ldots, \pi_{i_m}\rangle$,
$\pi'_i = \langle\pi'_{i_1}, \ldots, \pi'_{i_m}\rangle$, *and*
$\langle\pi_{i_1}, \pi'_{i_1}\rangle \in \varrho, \ldots, \langle\pi_{i_m}, \pi'_{i_m}\rangle \in \varrho$,
then $\langle\rho, \pi'_1, \ldots, \pi'_n\rangle \in \xi$.

Suppose Σ is a signature and $\mu = \langle\beta, \varrho, \lambda, \xi\rangle$ is a Σ morph. I call β the *basis set in* μ, ϱ the *re-entrancy relation in* μ, λ the *label function in* μ, and ξ the *relation extension in* μ. I write $\mathcal{M}_\Sigma$ for the set of Σ morphs.

Up to the relation extension, ξ, morphs are 94 abstract feature structures, as defined in Definition 32 above:[5] The basis set in a morph μ is a prefix closed set of paths that respects appropriateness, the re-entrancy relation in μ is a right invariant equivalence relation on the basis set in μ, and the label function in μ is a total function that respects the re-entrancy relation and the appropriateness function. In short, morphs comprise totally well-typed, sort resolved and potentially infinite abstract feature structures.

The relation extension, ξ, is new. For each relation symbol ρ in the signature Σ that is assigned arity n in Σ, the relation extension in a Σ morph

[5]The fact that they are defined with an RSRL signature that includes a sort hierarchy does not make a difference, since the label function, λ, of each morph μ only assigns maximally specific sorts to the elements of the basis set, β, in μ.

comprises a subset of the set of $n+1$ tuples consisting of ρ and n paths (or tuples of paths), $\pi_1, \ldots, \pi_n$. Each path (or tuple of paths), π_i, represents an argument of the relation in the morph. Informally, the abstract node to which the path leads is in the respective argument of the relation. Those arguments that are represented as tuples of paths contain chains. The representations of the arguments of relations in morphs respect the re-entrancy relation, i.e., if a path representing an argument is re-entrant with other paths, then each one of the re-entrant paths must occur as the respective argument of the representation of the relation in an element in the set ξ. For example, if $\langle \texttt{member}, \text{DRIVER}, \text{PASSENGERS} \rangle \in \xi$ and $\langle \text{DRIVER}, \text{OWNER} \rangle \in \varrho$ then $\langle \texttt{member}, \text{OWNER}, \text{PASSENGERS} \rangle \in \xi$.

In the following definition, I introduce a simple notational convention that is very convenient for talking about tuples of paths that all begin with the same prefix.

Definition 70 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$*, for each* $\pi \in \mathcal{A}^*$*, for each* $\langle \pi_1, \ldots, \pi_n \rangle \in \mathcal{A}^{**}$*,*

$\pi\langle \pi_1, \ldots, \pi_n \rangle$ *is an abbreviatory notation for* $\langle \pi\pi_1, \ldots, \pi\pi_n \rangle$.

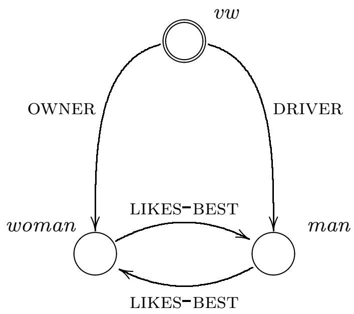
With the notational convention of Definition 70, I define reducts of morphs, which will later be needed in the definition of the set of morphs admitted by a theory. Informally speaking, morph admission refers to all the morphs that we can obtain following the paths in the basis set of morphs, because morph admission must ensure that all "embedded" morphs satisfy each description in the theory. Each abstract node to which a path leads is taken as the root node of a new morph that is "embedded" under the path in the bigger morph. All embedded morphs are reducts of the original morph.

Definition 71 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$*, for each* $\mu = \langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma$*, for each* $\pi \in \mathcal{A}^*$*,*

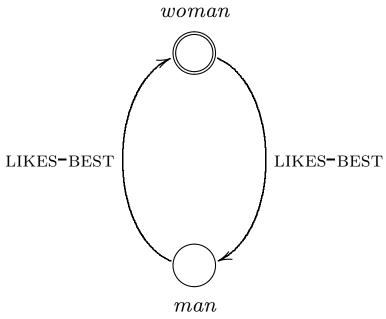
$$\beta/\pi = \{\pi' \in \mathcal{A}^* \mid \pi\pi' \in \beta\},$$
$$\varrho/\pi = \{\langle \pi_1, \pi_2 \rangle \in \mathcal{A}^* \times \mathcal{A}^* \mid \langle \pi\pi_1, \pi\pi_2 \rangle \in \varrho\},$$
$$\lambda/\pi = \{\langle \pi', \sigma \rangle \in \mathcal{A}^* \times \mathcal{S} \mid \langle \pi\pi', \sigma \rangle \in \lambda\},$$
$$\xi/\pi = \{\langle \rho, \pi_1, \ldots, \pi_n \rangle \in \mathcal{R} \times \overline{(\beta/\pi)}^* \mid \langle \rho, \pi\pi_1, \ldots, \pi\pi_n \rangle \in \xi\}, \textit{ and}$$
$$\mu/\pi = \langle \beta/\pi, \varrho/\pi, \lambda/\pi, \xi/\pi \rangle.$$

If Σ is a signature, μ is a Σ morph and π is a Σ path then I call μ/π the *π reduct of μ*. For example, assuming a signature Σ that corresponds to the 87 signature of Figure 2.3, which is the 87 signature for talking about car scenarios, the OWNER reduct of the Σ morph in (30a) is the Σ morph in (30b):

(30) a.



b.



If a Σ path π is in the basis set of the Σ morph μ then the π reduct of μ is also a Σ morph:

Proposition 12 *For each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, for each $\mu = \langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma$, for each $\pi \in \mathcal{A}^*$,*

if $\pi \in \beta$ then $\mu/\pi \in \mathcal{M}_\Sigma$.

With the definition of the π reduct of morphs, we have all constructs surrounding the notion of a morph in hand that are necessary to define morph admission. However, before I can define what it means for a Σ morph to satisfy a Σ formula, I must provide a number of technical auxiliary definitions, given in DEFINITION 72–75, that I will need in order to assign the set of morphs that satisfy a Σ formula to each Σ formula.

It is quite obvious under which circumstances a Σ morph should satisfy Σ formulae whose Σ terms start with the colon. For example, it is clear that a Σ morph that satisfies the sort assignment formula '$:\pi \sim \sigma$' (where π is a sequence of attributes) should have the pair $\langle \pi, \sigma \rangle$ in its label function. It assigns the sort σ to the path π in its basis set. We can say that we "insert" the path π for the Σ term '$:\pi$', and use π to determine which Σ morphs satisfy the Σ formula '$:\pi \sim \sigma$'. But under which conditions does a Σ morph satisfy Σ formulae that contain variables? Since the denotation of such formulae in interpretations depends on variable assignments in interpretations, a substitute for variable assignments must be defined for morphs. The answer to the question which conditions must hold for a morph to satisfy a formula must depend on the substitute for the variable assignment, on the sequence of attributes that follows a variable, and on their interaction. For terms with variables, appropriate paths or tuples of paths must be found whose shape and presence in or absence from the basis set, re-entrancy relation, label function, and relation extension of the morph determines whether or not the morph satisfies a formula. The DEFINITIONS 72–74 introduce the functions that are necessary to realize this idea.

DEFINITION 72 provides the substitute for the variable assignments in interpretations. Instead of entities or chains of entities, variables are assigned paths or tuples of paths:

Definition 72 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$,

I_Σ *is the set of total functions from* $\mathcal{VAR}$ *to* $\overline{\mathcal{A}^*}$.

Let Σ be a signature. Then I call each member of I_Σ a Σ *insertion*. Given a Σ insertion that assigns the Σ path π to the variable v, a Σ morph $\mu = \langle \beta, \varrho, \lambda, \xi \rangle$ satisfies the Σ formula '$v \sim \sigma$' only if $\langle \pi, \sigma \rangle \in \lambda$.

Usually, terms with variables do not only consist of variables; a sequence of attributes succeeds the variable. In that case, the path (or the tuple of paths) that I must insert for a given term in order to determine which morphs satisfy a formula depends not only on the Σ insertion but also on the sequence of attributes after the variable. The idea is that if the Σ insertion ι assigns the path π to the variable v, then a path insertion under ι in Σ appends the sequence of attributes following the variable in the term to π. If the Σ insertion ι assigns a tuple of paths $\langle \pi_1, \ldots, \pi_n \rangle$ to the variable v, then a path insertion under ι in Σ uses the sequence of (quasi-) attributes that may succeed the variable in the term in order to determine which path or tuple of paths it inserts (if any). It navigates tuples of paths that are introduced by ι with the quasi-attributes $\triangleright$ and $\dagger$, whereas other attributes cannot navigate tuples of paths. While the path insertion for terms that start with variables is quite complex, it is trivial for terms that start with the colon and do not contain quasi-attributes: The path insertion function will simply insert the sequence of attributes succeeding the colon, as we have already seen in the example above.

Before I define the path insertion functions for each signature, an auxiliary function, T_Σ, is defined that will be needed in the recursive case of the inductive definition of path insertions in Σ. T_Σ appends attributes (but not quasi-attributes) to paths and navigates tuples of paths with the quasi-attributes:

Definition 73 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$,

T_Σ *is the smallest partial function from* $\overline{\mathcal{A}^*} \times \widehat{\mathcal{A}}$ *to* $\overline{\mathcal{A}^*}$ *such that,*

for each $\pi \in \mathcal{A}^*$, *for each* $\alpha \in \mathcal{A}$,

$$\mathrm{T}_\Sigma(\pi, \alpha) = \pi\alpha,$$

for each $\langle \pi_0, \ldots, \pi_n \rangle \in \mathcal{A}^{**}$,

$$\mathrm{T}_\Sigma(\langle \pi_0, \ldots, \pi_n \rangle, \dagger) = \pi_0,$$
$$\mathrm{T}_\Sigma(\langle \pi_0, \ldots, \pi_n \rangle, \triangleright) = \langle \pi_1, \ldots, \pi_n \rangle.$$

For each signature Σ, T_Σ is undefined for each pair comprising a Σ path and a quasi-attribute, because paths cannot be navigated with quasi-attributes. On the other hand, T_Σ is undefined for each pair consisting of a tuple of paths and any ordinary attribute, because only the quasi-attributes can navigate tuples of paths.

Path insertions under a Σ insertion in Σ are partial functions from Σ terms to Σ paths and tuples of Σ paths. For the Σ term ':', each path insertion under a Σ insertion ι in Σ, Π^ι_Σ, inserts the empty Σ path, ε. For each variable v, it inserts the Σ path or tuple of Σ paths that ι assigns to v. Finally, for complex Σ terms of the form $\tau\alpha$ (where τ is a Σ term and α is a (quasi-) attribute), it determines the inserted Σ path with the partial function T_Σ as the result of applying T_Σ to the pair comprising the Σ path that Π^ι_Σ itself inserts for τ, and α.

Definition 74 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each* $\iota \in \mathrm{I}_\Sigma$,

Π^ι_Σ *is the smallest partial function from* $\mathcal{T}^\Sigma$ *to* $\overline{\mathcal{A}^*}$ *such that*

$\Pi^\iota_\Sigma(:) = \varepsilon$,
for each $v \in \mathcal{VAR}$, $\Pi^\iota_\Sigma(v) = \iota(v)$, *and*
for each $\tau \in \mathcal{T}^\Sigma$, *for each* $\alpha \in \widehat{\mathcal{A}}$, $\Pi^\iota_\Sigma(\tau\alpha) = \mathrm{T}_\Sigma(\Pi^\iota_\Sigma(\tau), \alpha)$.

Suppose Σ is a signature, and ι is a Σ insertion. Then I call each Π^ι_Σ the *path insertion function under* ι *in* Σ.

Definition 75 is the last step towards morph satisfaction. The re-entrancy relation, ϱ, and the label function, λ, in a morph, μ, do not suffice to determine whether the morph satisfies a formula that describes chains, because ϱ and λ are only defined for the members of β in μ, which are paths, and chains are represented by tuples of paths. For this reason, for every signature Σ, I define extensions of the re-entrancy relation and the label function of each Σ morph. I call these extensions the *expanded re-entrancy relation* and the *expanded label function*, and I write $\hat{\varrho}$ and $\hat{\lambda}$. Expanded re-entrancy relations allow tuples of paths in morphs to be re-entrant, and the expanded label function assigns the quasi-sort *echain* to the empty tuple of paths and the quasi-sort *nechain* to each nonempty tuple of paths in the morph:

Definition 75 *For each signature* Σ, *for each* $\langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma$,

$\hat{\varrho}$ *is the smallest subset of* $\overline{\beta} \times \overline{\beta}$ *such that*

$\varrho \subseteq \hat{\varrho}$, *and*
for each $\pi_1 \in \beta, \ldots, \pi_n \in \beta$, *for each* $\pi'_1 \in \beta, \ldots, \pi'_n \in \beta$,
if $\langle \pi_1, \pi'_1 \rangle \in \varrho$, $\ldots$, *and* $\langle \pi_n, \pi'_n \rangle \in \varrho$
then $\langle \langle \pi_1, \ldots, \pi_n \rangle, \langle \pi'_1, \ldots, \pi'_n \rangle \rangle \in \hat{\varrho}$,

$\hat{\lambda}$ *is the total function from* $\overline{\beta}$ *to* $\widehat{\mathcal{S}}$ *such that*

for each $\pi \in \beta$, $\hat{\lambda}(\pi) = \lambda(\pi)$,
for each $\pi_1 \in \beta$, $\ldots$, *for each* $\pi_n \in \beta$,

$$\hat{\lambda}(\langle \pi_1, \ldots, \pi_n \rangle) = \begin{cases} \textit{echain} & \textit{if } n = 0, \\ \textit{nechain} & \textit{if } n > 0. \end{cases}$$

For each signature Σ and for each Σ insertion $\iota \in \mathrm{I}_\Sigma$, the morph satisfaction function under ι in Σ, Δ^ι_Σ is defined inductively for all Σ formulae.

Definition 76 *For each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, for each $\iota \in \mathrm{I}_\Sigma$, Δ^ι_Σ is the total function from $\mathcal{D}^\Sigma$ to $Pow(\mathcal{M}_\Sigma)$ such that*

for each $\tau \in \mathcal{T}^\Sigma$, for each $\sigma \in \widehat{\mathcal{G}}$,

$$\Delta^\iota_\Sigma(\tau \sim \sigma) = \left\{ \langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma \,\middle|\, \begin{array}{l} \Pi^\iota_\Sigma(\tau) \textit{ is defined, and,} \\ \textit{for some } \sigma' \in \widehat{\mathcal{S}}, \\ \quad \langle \Pi^\iota_\Sigma(\tau), \sigma' \rangle \in \hat{\lambda} \textit{ and } \sigma' \mathrel{\widehat{\sqsubseteq}} \sigma \end{array} \right\},$$

for each $\tau_1 \in \mathcal{T}^\Sigma$, for each $\tau_2 \in \mathcal{T}^\Sigma$,

$$\Delta^\iota_\Sigma(\tau_1 \approx \tau_2) = \left\{ \langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma \,\middle|\, \begin{array}{l} \Pi^\iota_\Sigma(\tau_1) \textit{ is defined,} \\ \Pi^\iota_\Sigma(\tau_2) \textit{ is defined, and} \\ \langle \Pi^\iota_\Sigma(\tau_1), \Pi^\iota_\Sigma(\tau_2) \rangle \in \hat{\varrho} \end{array} \right\},$$

for each $\rho \in \mathcal{R}$, for each $v_1 \in \mathcal{VAR}$, ..., for each $v_n \in \mathcal{VAR}$,

$$\Delta^\iota_\Sigma(\rho(v_1, \ldots, v_n)) = \left\{ \langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma \,\middle|\, \langle \rho, \iota(v_1), \ldots, \iota(v_n) \rangle \in \xi \right\},$$

for each $v \in \mathcal{VAR}$, for each $\delta \in \mathcal{D}^\Sigma$,

$$\Delta^\iota_\Sigma(\exists v\, \delta) = \left\{ \langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma \,\middle|\, \begin{array}{l} \textit{for some } \pi \in \overline{\beta}, \\ \quad \langle \beta, \varrho, \lambda, \xi \rangle \in \Delta^{\iota[\frac{\pi}{v}]}_\Sigma(\delta) \end{array} \right\},$$

for each $v \in \mathcal{VAR}$, for each $\delta \in \mathcal{D}^\Sigma$,

$$\Delta^\iota_\Sigma(\forall v\, \delta) = \left\{ \langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma \,\middle|\, \begin{array}{l} \textit{for each } \pi \in \overline{\beta}, \\ \quad \langle \beta, \varrho, \lambda, \xi \rangle \in \Delta^{\iota[\frac{\pi}{v}]}_\Sigma(\delta) \end{array} \right\},$$

for each $\delta \in \mathcal{D}^\Sigma$,

$$\Delta^\iota_\Sigma(\neg\delta) = \mathcal{M}_\Sigma \backslash \Delta^\iota_\Sigma(\delta),$$

for each $\delta_1 \in \mathcal{D}^\Sigma$, for each $\delta_2 \in \mathcal{D}^\Sigma$,

$$\Delta^\iota_\Sigma([\delta_1 \wedge \delta_2]) = \Delta^\iota_\Sigma(\delta_1) \cap \Delta^\iota_\Sigma(\delta_2),$$

for each $\delta_1 \in \mathcal{D}^\Sigma$, *for each* $\delta_2 \in \mathcal{D}^\Sigma$,

$$\Delta^\iota_\Sigma([\delta_1 \vee \delta_2]) = \Delta^\iota_\Sigma(\delta_1) \cup \Delta^\iota_\Sigma(\delta_2),$$

for each $\delta_1 \in \mathcal{D}^\Sigma$, *for each* $\delta_2 \in \mathcal{D}^\Sigma$,

$$\Delta^\iota_\Sigma([\delta_1 \rightarrow \delta_2]) = (\mathcal{M}_\Sigma \backslash \Delta^\iota_\Sigma(\delta_1)) \cup \Delta^\iota_\Sigma(\delta_2), \textit{ and}$$

for each $\delta_1 \in \mathcal{D}^\Sigma$, *for each* $\delta_2 \in \mathcal{D}^\Sigma$,

$$\begin{aligned}\Delta^\iota_\Sigma([\delta_1 \leftrightarrow \delta_2]) = {} & ((\mathcal{M}_\Sigma \backslash \Delta^\iota_\Sigma(\delta_1)) \cap (\mathcal{M}_\Sigma \backslash \Delta^\iota_\Sigma(\delta_2))) \cup \\ & (\Delta^\iota_\Sigma(\delta_1) \cap \Delta^\iota_\Sigma(\delta_2)).\end{aligned}$$

Let Σ be a signature, μ a Σ morph, ι a Σ insertion and δ a Σ formula. I call Δ^ι_Σ the *morph satisfaction function under* ι *in* Σ, and say μ *satisfies* δ *under* ι *in* Σ if and only if $\mu \in \Delta^\iota_\Sigma(\delta)$.

The set of Σ morphs that satisfy a Σ sort assignment formula or a Σ path equation under an insertion ι in Σ is determined in the obvious way by the paths that the path insertion function under ι in Σ assigns to the terms in the Σ formulae, and by the expanded label function and the expanded re-entrancy relation of the Σ morphs. Note the slight difference in the treatment of relational formulae regarding their denotation in a Σ interpretation I under a variable assignment in I and the set of Σ morphs that satisfy them according to the morph satisfaction function under ι in Σ. A Σ morph that satisfies a relational formula represents the arguments of the relation as paths that are members of its basis set by the definition of Σ morphs. The componenthood condition for the arguments of relations is thus already built into the Σ morphs. By contrast, variable assignments in interpretations assign an arbitrary entity in an interpretation to a variable, and only the component quantification in descriptions ensures that the arguments of relations are components of a common ancestor in a Σ interpretation. The denotation in interpretations of relational formulae itself does not enforce the componenthood condition on the arguments of relations.

Just as two term interpretations of the same term in an interpretation under different variable assignments agree if the two variable assignments assign the same entity or chain of entities to the variables that occur free in the term, two path insertions under two different insertions insert the same path or tuple of paths for the same term if the two insertions agree on the variables that occur free in the term. The corresponding result holds for formulae: The set of morphs that satisfy a formula δ under ι_1 equals the set of morphs that satisfy δ under ι_2 if ι_1 and ι_2 insert the same paths or tuples of paths for the variables that occur free in δ. PROPOSITION 13 thus replicates the result of PROPOSITION 9 about variable assignments in term interpretation functions and in formula interpretation functions as a result about the behavior of insertions in path insertions and in morph satisfaction functions.

Proposition 13 *For each signature Σ, for each $\iota_1 \in \mathrm{I}_\Sigma$, for each $\iota_2 \in \mathrm{I}_\Sigma$,*

for each $\tau \in \mathcal{T}^\Sigma$,

if for each $v \in FV(\tau)$, $\iota_1(v) = \iota_2(v)$

then $\Pi^{\iota_1}_\Sigma(\tau)$ is defined iff $\Pi^{\iota_2}_\Sigma(\tau)$ is defined, and
if $\Pi^{\iota_1}_\Sigma(\tau)$ is defined then $\Pi^{\iota_1}_\Sigma(\tau) = \Pi^{\iota_2}_\Sigma(\tau)$, and

for each $\delta \in \mathcal{D}^\Sigma$,

if for each $v \in FV(\delta)$, $\iota_1(v) = \iota_2(v)$
then $\Delta^{\iota_1}_\Sigma(\delta) = \Delta^{\iota_2}_\Sigma(\delta)$.

As a corollary of PROPOSITION 13, if δ is a Σ description, the value of the morph satisfaction function under ι in Σ on δ, $\Delta^{\iota}_\Sigma(\delta)$, is independent of the choice of the Σ insertion ι.

I can now put the results and definitions together and define morph admission:

Definition 77 *For each signature Σ,*

M_Σ is the total function from $Pow(\mathcal{D}^\Sigma_0)$ to $Pow(\mathcal{M}_\Sigma)$ such that for each $\theta \subseteq \mathcal{D}^\Sigma_0$,

$$M_\Sigma(\theta) = \left\{ \langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma \;\middle|\; \begin{array}{l} \textit{for each } \pi \in \beta, \\ \textit{for each } \iota \in \mathrm{I}_\Sigma, \textit{ and} \\ \textit{for each } \delta \in \theta, \\ \quad \langle \beta, \varrho, \lambda, \xi \rangle / \pi \in \Delta^{\iota}_\Sigma(\delta) \end{array} \right\}.$$

Let Σ be a signature. I call M_Σ the *morph admission function in* Σ. A Σ morph μ is in the set of Σ morphs admitted by a Σ theory θ exactly if μ and every possible π reduct of μ satisfy every description in θ.

From the definition of the morph admission function in Σ it is clear that the Σ morphs admitted by a theory θ cannot represent the tokens in a model of the grammar $\langle \Sigma, \theta \rangle$, because, unlike the intended use of tokens, there cannot be isomorphic but distinct configurations under a Σ morph in an interpretation whose entities are Σ morphs and whose attribute interpretation function of each attribute α is constructed in the apparently natural way as the partial function mapping each Σ morph μ to its α reduct μ/α. Informally, if the entities in an interpretation of the kind indicated were Σ morphs, all tokens in the interpretation that are of the same shape would necessarily have to be identical. For example, in such a hypothetical model of the grammar of Pollard and Sag 1994, there could only be one third person feminine singular index configuration that would be shared by all entities that are required to have a configuration of that shape as their component.

The obvious relationship of the set of morphs admitted by a theory to sets of 94 abstract feature structures admitted by theories, the fact that the set of admitted Σ morphs by definition contains all morphs that are admitted by the theory, and the fact that every possible π reduct of every morph in the set satisfies all descriptions in the theory suggest that the set of Σ morphs admitted by θ is a set of mathematical representations of the equivalence classes under congruence of the tokens in an exhaustive model of the grammar $\langle \Sigma, \theta \rangle$. If that is true—and the next section will demonstrate that it is—then the set of morphs admitted by a theory can justifiably be regarded as an accurate formalization of the object types of Pollard and Sag 1994.

3.1.2.3 Canonical Exhaustive Models

The Σ morphs admitted by a Σ theory θ are not themselves entities in an exhaustive model of the grammar $\langle \Sigma, \theta \rangle$. But under the preliminary assumption that, just as the 94 abstract feature structures admitted by an SRL grammar, they represent equivalence classes of congruent tokens in an exhaustive model of $\langle \Sigma, \theta \rangle$, entities in an exhaustive $\langle \Sigma, \theta \rangle$ model can be recovered from them. The idea is to reintroduce the nodes of concrete feature structures—from which morphs as some kind of abstract feature structure abstract away—as additional structure in an extension of morphs. Similar to Pollard's construction of exhaustive models for SRL grammars in (45) above, my extension of morphs adds a distinguished node to morphs. I call the resulting new structures *canonical entities*:

Definition 78 *For each signature* Σ,

> *o is a* ***canonical entity in*** Σ
> *iff o is a quintuple* $\langle \beta, \varrho, \lambda, \xi, \eta \rangle$,
> $\langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma$, *and*
> $\eta \in Quo(\langle \beta, \varrho \rangle)$.[6]

Suppose that Σ is a signature. I write $\mathcal{C}_\Sigma$ for the set of canonical entities in Σ. Suppose further that $\langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma$ and $\pi \in \beta$. I write $|\pi|_\varrho$ for the equivalence class of π in ϱ. Thus, I write $\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle$ for the canonical entity $\langle \beta, \varrho, \lambda, \xi, \eta \rangle$, where η is the equivalence class of π in ϱ.

Depending on the grammar, the arguments of (some) relations in an interpretation can be chains of entities instead of ordinary entities of the universe. In order to have a convenient notation that covers these cases in the definition of the relation interpretation function of the canonical exhaustive models of grammars, it is advantageous to introduce a compact notation for chains of canonical entities that generalizes the notation for canonical entities to chains of canonical entities:

Definition 79 *For each signature* Σ, *for each* $\theta \in \mathcal{D}_0^\Sigma$, *for each* $\langle \langle \beta, \varrho, \lambda, \xi, |\pi_1|_\varrho \rangle, \ldots, \langle \beta, \varrho, \lambda, \xi, |\pi_n|_\varrho \rangle \rangle \in (\mathcal{C}_\Sigma)^*$,

> $\langle \beta, \varrho, \lambda, \xi, |\langle \pi_1, \ldots, \pi_n \rangle|_\varrho \rangle$ *is an abbreviatory notation for*
> $\langle \langle \beta, \varrho, \lambda, \xi, |\pi_1|_\varrho \rangle, \ldots, \langle \beta, \varrho, \lambda, \xi, |\pi_n|_\varrho \rangle \rangle$.

Given the notational convention for chains of canonical entities, Definition 80 provides the canonical exhaustive model for each $\langle \Sigma, \theta \rangle$ grammar. The definition of the relation interpretation functions employs the notational convention of (79), because arguments of relations might be chains, which are represented as tuples of paths in the relation extension of morphs.

Definition 80 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each* $\theta \subseteq \mathcal{D}_0^\Sigma$,

[6] $Quo(\langle \beta, \varrho \rangle)$ is the quotient of ϱ, i.e., the set of equivalence classes of ϱ.

$\mathbf{U}^{\theta}_{\Sigma} = \left\{ \langle \beta, \varrho, \lambda, \xi, \eta \rangle \in \mathcal{C}_{\Sigma} \,\middle|\, \langle \beta, \varrho, \lambda, \xi \rangle \in M_{\Sigma}(\theta) \right\}$,

$\mathbf{S}^{\theta}_{\Sigma} = \left\{ \langle u, \sigma \rangle \in \mathbf{U}^{\theta}_{\Sigma} \times \mathcal{S} \,\middle|\, \begin{array}{l} \textit{for some } \langle \beta, \varrho, \lambda, \xi \rangle \in M_{\Sigma}(\theta), \\ \textit{for some } \pi \in \beta, \\ \quad u = \langle \beta, \varrho, \lambda, \xi, |\pi|_{\varrho} \rangle, \textit{ and} \\ \quad \sigma = \lambda(\pi) \end{array} \right\}$,

$\mathbf{A}^{\theta}_{\Sigma}$ *is the total function from* $\mathcal{A}$ *to* $Pow(\mathbf{U}^{\theta}_{\Sigma} \times \mathbf{U}^{\theta}_{\Sigma})$ *such that for each* $\alpha \in \mathcal{A}$,

$$\mathbf{A}^{\theta}_{\Sigma}(\alpha) = \left\{ \langle u, u' \rangle \in \mathbf{U}^{\theta}_{\Sigma} \times \mathbf{U}^{\theta}_{\Sigma} \,\middle|\, \begin{array}{l} \textit{for some } \langle \beta, \varrho, \lambda, \xi \rangle \in M_{\Sigma}(\theta), \\ \textit{for some } \pi \in \beta, \\ \quad u = \langle \beta, \varrho, \lambda, \xi, |\pi|_{\varrho} \rangle \textit{ and} \\ \quad u' = \langle \beta, \varrho, \lambda, \xi, |\pi\alpha|_{\varrho} \rangle \end{array} \right\},$$

$\mathbf{R}^{\theta}_{\Sigma}$ *is the total function from* $\mathcal{R}$ *to the power set of* $\bigcup_{n \in \mathbb{N}} \overline{\mathbf{U}^{\theta}_{\Sigma}}^{n}$ *such that for each* $\rho \in \mathcal{R}$,

$$\mathbf{R}^{\theta}_{\Sigma}(\rho) = \left\{ \langle u_1, \ldots, u_n \rangle \in \bigcup_{n \in \mathbb{N}} \overline{\mathbf{U}^{\theta}_{\Sigma}}^{n} \,\middle|\, \begin{array}{l} \textit{for some } \langle \beta, \varrho, \lambda, \xi \rangle \in M_{\Sigma}(\theta), \\ \textit{for some } \langle \rho, \pi_1, \ldots, \pi_n \rangle \in \xi, \\ \quad u_1 = \langle \beta, \varrho, \lambda, \xi, |\pi_1|_{\varrho} \rangle, \ \ldots, \\ \quad u_n = \langle \beta, \varrho, \lambda, \xi, |\pi_n|_{\varrho} \rangle \end{array} \right\},$$

$\mathbf{I}^{\theta}_{\Sigma} = \langle \mathbf{U}^{\theta}_{\Sigma}, \mathbf{S}^{\theta}_{\Sigma}, \mathbf{A}^{\theta}_{\Sigma}, \mathbf{R}^{\theta}_{\Sigma} \rangle$.

For each signature Σ, and for each Σ theory θ, the entities in $\mathbf{I}^{\theta}_{\Sigma}$ are the canonical entities provided by the set of Σ morphs admitted by θ with distinguished nodes. Each Σ morph admitted by θ occurs once with each node as distinguished node as an entity in $\mathbf{I}^{\theta}_{\Sigma}$. The species assignment function, $\mathbf{S}^{\theta}_{\Sigma}$, assigns each canonical entity in $\mathbf{U}^{\theta}_{\Sigma}$ the species that is assigned to its distinguished node in the respective Σ morph. The attribute interpretation function, $\mathbf{A}^{\theta}_{\Sigma}$, maps each attribute α to the partial function from canonical entities in Σ to canonical entities in Σ that comprise the same morph; and the distinguished node of the first canonical entity is in the ϱ equivalence class of some Σ path π and the distinguished node of the second is in the ϱ equivalence class of the Σ path $\pi\alpha$. Since Σ morphs obey the appropriateness conditions, we know that the attribute α is appropriate to the species that is assigned to π by λ in the relevant Σ morph. The relation interpretation function, $\mathbf{R}^{\theta}_{\Sigma}$, finally, interprets each relation symbol, ρ, with arity n as the set of n-tuples of canonical entities (and chains of canonical entities) that all comprise the same Σ morph, μ, such that, for each of these n-tuples of (chains of) canonical entities, there is an element e in the relation extension of μ such that the first path (or tuple of paths) in e is in the equivalence class of the distinguished node of the first canonical entity in the n-tuple of canonical entities (or, for tuples of paths, each path in the tuple of paths is in the equivalence class of the distinguished node of the corresponding canonical entity in the chain of canonical entities), and so on, up to the nth path (or tuple of paths) in e.

Informally, this construction turns each Σ morph $\mu = \langle \beta, \varrho, \lambda, \xi \rangle$ admitted by θ in Σ into a configuration of canonical entities under the canonical entity $\langle \beta, \varrho, \lambda, \xi, |\varepsilon|_\varrho \rangle$ that mirrors the configuration of abstract nodes in μ and the relations that hold among them by virtue of ξ. The abstract nodes of μ are, thus, reified as concrete entities. The way in which the morph admission function in Σ is defined ultimately guarantees that each configuration of canonical entities under a canonical entity with the empty path as distinguished node, $\langle \beta, \varrho, \lambda, \xi, |\varepsilon|_\varrho \rangle$, is a $\langle \Sigma, \theta \rangle$ model. Moreover, informally speaking, the morph admission function in Σ will, via this reification process, provide congruent copies of configurations under an entity of all possible $\langle \Sigma, \theta \rangle$ models, and since all of them are collected in $\mathbf{I}^\theta_\Sigma$, each $\mathbf{I}^\theta_\Sigma$ is an exhaustive $\langle \Sigma, \theta \rangle$ model. Before I can prove this claim, however, I have to establish more basic properties of $\mathbf{I}^\theta_\Sigma$. First of all, it is a Σ interpretation:

Proposition 14 *For each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, for each $\theta \subseteq \mathcal{D}^\Sigma_0$,*

> $\mathbf{U}^\theta_\Sigma$ *is a set,*
>
> $\mathbf{S}^\theta_\Sigma$ *is a total function from $\mathbf{U}^\theta_\Sigma$ to $\mathcal{S}$,*
>
> $\mathbf{A}^\theta_\Sigma$ *is a total function from $\mathcal{A}$ to the set of partial functions from $\mathbf{U}^\theta_\Sigma$ to $\mathbf{U}^\theta_\Sigma$,*
>
> $\mathbf{R}^\theta_\Sigma$ *is a total function from $\mathcal{R}$ to the power set of $\bigcup\limits_{n \in \mathbb{N}} \overline{\mathbf{U}^\theta_\Sigma}^n$, and*
>
> $\mathbf{I}^\theta_\Sigma$ *is a Σ interpretation.*

Let Σ be a signature and $\theta \subseteq \mathcal{D}^\Sigma_0$. I call $\mathbf{I}^\theta_\Sigma$ the *canonical Σ interpretation of θ*. A complex proof, due to Richter and King 1997 and contained in Appendix A, shows the following proposition about the canonical Σ interpretation of θ:

Proposition 15 *For each signature* Σ*, for each* $\theta \subseteq \mathcal{D}_0^{\Sigma}$*,*

> $\mathbf{I}_{\Sigma}^{\theta}$ *is a* $\langle \Sigma, \theta \rangle$ *model.*

All that is now left to show is that $\mathbf{I}_{\Sigma}^{\theta}$ is not simply a model of $\langle \Sigma, \theta \rangle$, but that it is an exhaustive $\langle \Sigma, \theta \rangle$ model. The basic idea of how to do this is simple, although its execution involves a number of technical complications that I will omit here.[7] According to THEOREM 3, it suffices to show that $\mathbf{I}_{\Sigma}^{\theta}$ simulates every other $\langle \Sigma, \theta \rangle$ model, or, in other words, that for each entity, u, in any other $\langle \Sigma, \theta \rangle$ model I, $\mathbf{I}_{\Sigma}^{\theta}$ contains an entity $\mathbf{u}$ such that $\langle \mathbf{u}, \mathbf{I}_{\Sigma}^{\theta} \rangle$ and $\langle u, \mathsf{I} \rangle$ are congruent in Σ. I use the previously discussed technique of defining a class of abstraction functions to do this: In the discussion of the feature structures of HPSG 87 we have seen that we can define abstraction functions from concrete feature structures to abstract feature structures that map isomorphic concrete feature structures to identical abstract feature structures. For HPSG 94, King defined corresponding abstraction functions in SRL that map entities in an interpretation directly to abstract feature structures, avoiding the extra step of going via concrete feature structures. King's abstraction functions map two entities in two SRL interpretations to the same abstract feature structure exactly if they are SRL congruent. If two entities in two interpretations are mapped to the same abstract feature structure we can thus conclude that they are SRL congruent.

King's abstraction functions are defined with respect to SRL interpretations, and their ranges are sets of 94 abstract feature structures. Below I extend King's abstraction functions, defining abstraction with respect to RSRL interpretations, and taking sets of morphs as their range. DEFINITION 82 first presents a relational definition of abstraction. For each Σ interpretation I, it defines an abstraction relation with respect to I between the entities in the universe of I and Σ morphs. In several propositions I will then claim that the abstraction relations with respect to an interpretation have the important properties of King's abstraction functions.

An abstraction relation relates each entity u in an interpretation to an abstract representation of the configuration of entities under u, abstracting away from the concrete entities under u and representing them as the equivalence classes of the paths with which they can be reached. A definition of the abstraction relation must, therefore, refer to components and chains of components of entities in interpretations and how they are reached by paths

[7]See Appendix A.2 for a complete presentation.

from the (unique) root entities. The auxiliary Definition 81 is designed to make this simple. Instead of accessing the components and chains of components by the interpretation of terms on entities, they are addressed by paths and tuples of paths.

Definition 81 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$*, for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$*,* T_{I} *is the partial function from the Cartesian product of* $\overline{\mathcal{A}^*}$ *and* U *to* $\overline{\mathsf{U}}$ *such that*

> *for each* $\pi \in \mathcal{A}^*$*, for each* $u \in \mathsf{U}$*,*
>
>> $T_{\mathsf{I}}(\pi, u)$ *is defined iff for some* $ass \in Ass_{\mathsf{I}}$*,* $T^{ass}_{\mathsf{I}}(:\pi)(u)$ *is defined, and*
>> *if* $T_{\mathsf{I}}(\pi, u)$ *is defined then for some* $ass \in Ass_{\mathsf{I}}$*,* $T_{\mathsf{I}}(\pi, u) = T^{ass}_{\mathsf{I}}(:\pi)(u)$*,*
>
> *for each* $\langle \pi_1, \ldots, \pi_n \rangle \in \mathcal{A}^{**}$*, for each* $u \in \mathsf{U}$*,*
>
>> $T_{\mathsf{I}}(\langle \pi_1, \ldots, \pi_n \rangle, u)$ *is defined*
>> *iff* $T_{\mathsf{I}}(\pi_1, u)$ *is defined, ...,* $T_{\mathsf{I}}(\pi_n, u)$ *is defined, and*
>> *if* $T_{\mathsf{I}}(\langle \pi_1, \ldots, \pi_n \rangle, u)$ *is defined*
>> *then* $T_{\mathsf{I}}(\langle \pi_1, \ldots, \pi_n \rangle, u) = \langle T_{\mathsf{I}}(\pi_1, u), \ldots, T_{\mathsf{I}}(\pi_n, u) \rangle$*.*

For each signature Σ and each Σ interpretation I, I call T_{I} the *extended* Σ *path interpretation function in* I. A Σ path π is defined on an entity u in I exactly if the Σ term that prefixes the path with a colon is defined on u, and if it is defined on u then the value of T_{I} on the pair $\langle \pi, u \rangle$ equals the term denotation of '$:\pi$' on u. In a straightforward extension, the interpretation of Σ paths on entities is generalized to the interpretation of tuples of Σ paths on entities. A tuple of Σ paths is defined on an entity exactly if each Σ path in the tuple is defined on the entity, and if the tuple of Σ paths is defined on the entity then the value of interpreting the tuple of Σ paths on the entity is the chain of components of the entity that is obtained by interpreting each individual Σ path in the tuple on the entity. The generalization to tuples of paths is needed for abstracting relations with chains in their arguments from interpretations:

Definition 82 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$*, for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$*,*

> $\mathsf{Abst_I}$ *is the binary relation on* $\mathsf{U} \times \mathcal{M}_\Sigma$ *such that, for each* $u \in \mathsf{U}$, *for each* $\langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma$, $\langle u, \langle \beta, \varrho, \lambda, \xi \rangle\rangle \in \mathsf{Abst_I}$ *iff*
>
> $$\beta = \left\{ \pi \in \mathcal{A}^* \,\middle|\, T_\mathsf{I}(\pi, u) \textit{ is defined} \right\},$$
>
> $$\varrho = \left\{ \langle \pi_1, \pi_2 \rangle \in \mathcal{A}^* \times \mathcal{A}^* \,\middle|\, \begin{array}{l} T_\mathsf{I}(\pi_1, u) \textit{ is defined,} \\ T_\mathsf{I}(\pi_2, u) \textit{ is defined, and} \\ T_\mathsf{I}(\pi_1, u) = T_\mathsf{I}(\pi_2, u) \end{array} \right\},$$
>
> $$\lambda = \left\{ \langle \pi, \sigma \rangle \in \mathcal{A}^* \times \mathcal{S} \,\middle|\, \begin{array}{l} T_\mathsf{I}(\pi, u) \textit{ is defined,} \\ \mathsf{S}(T_\mathsf{I}(\pi, u)) = \sigma \end{array} \right\},$$
>
> $$\xi = \left\{ \langle \rho, \pi_1, \ldots, \pi_n \rangle \in \mathcal{R} \times (\overline{\mathcal{A}^*})^* \,\middle|\, \begin{array}{l} T_\mathsf{I}(\pi_1, u) \textit{ is defined, } \ldots, \\ T_\mathsf{I}(\pi_n, u) \textit{ is defined, and} \\ \langle T_\mathsf{I}(\pi_1, u), \ldots, T_\mathsf{I}(\pi_n, u)\rangle \in \mathsf{R}(\rho) \end{array} \right\}.$$

For each signature Σ, for each Σ interpretation I, I call $\mathsf{Abst_I}$ the *abstraction relation with respect to* I. Each abstraction relation with respect to an interpretation is functional. It maps each entity in the interpretation to exactly one morph:

Proposition 16 *For each signature* Σ, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$,

> $\mathsf{Abst_I}$ *is a total function from* U *to* $\mathcal{M}_\Sigma$.

Proposition 16 justifies referring to an abstraction relation with respect to an interpretation I as a (total) abstraction function with respect to I.

Just as the related abstraction functions whose domains are concrete feature structures of HPSG 87, or the even more similar abstraction functions in the SRL formalism whose domains are entities in SRL interpretations and 94 concrete feature structures, the abstraction relations with respect to an RSRL interpretation map entities that occur in congruent pairs of an entity and an interpretation to one and the same element in their range, namely to one single morph; and if a morph is the abstraction of two distinct entities then the two entities in their interpretations are congruent:

Proposition 17 *For each signature* Σ, *for each* Σ *interpretation* $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1 \rangle$, *for each* Σ *interpretation* $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2 \rangle$, *for each* $o_1 \in \mathsf{U}_1$, *for each* $o_2 \in \mathsf{U}_2$,

$\mathsf{Abst}_{\mathsf{I}_1}(o_1) = \mathsf{Abst}_{\mathsf{I}_2}(o_2)$

iff $\langle o_1, \mathsf{I}_1 \rangle$ *and* $\langle o_2, \mathsf{I}_2 \rangle$ *are congruent in* Σ.

The next proposition establishes the necessary relationship between the set of Σ morphs that can be abstracted from the set of entities in a $\langle \Sigma, \theta \rangle$ model and the set of Σ morphs admitted by the Σ theory θ. Every Σ morph abstracted from an entity in a $\langle \Sigma, \theta \rangle$ model is a Σ morph that is admitted by θ:

Proposition 18 *For each signature* Σ, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, *for each* $\theta \subseteq \mathcal{D}_0^\Sigma$,

if I *is a* $\langle \Sigma, \theta \rangle$ *model*

then for each $o \in \mathsf{U}$, $\mathsf{Abst}_{\mathsf{I}}(o) \in M_\Sigma(\theta)$.

When I will show that for each entity u in some $\langle \Sigma, \theta \rangle$ model, the $\langle \Sigma, \theta \rangle$ model $\mathbf{I}_\Sigma^\theta$ contains an entity that abstracts to the same Σ morph as u, I will have to produce a particular entity in $\mathbf{I}_\Sigma^\theta$ that abstracts to the same Σ morph. With Proposition 19 I can always pick the canonical entity in $\mathbf{I}_\Sigma^\theta$ that comprises the Σ morph in question and the equivalence class containing the empty path as the distinguished node:

Proposition 19 *For each signature* Σ, *for each* $\theta \subseteq \mathcal{D}_0^\Sigma$, *for each* $\langle \beta, \varrho, \lambda, \xi \rangle \in M_\Sigma(\theta)$,

$\mathsf{Abst}_{\mathbf{I}_\Sigma^\theta}(\langle \beta, \varrho, \lambda, \xi, |\varepsilon|_\varrho \rangle) = \langle \beta, \varrho, \lambda, \xi \rangle$.

Proposition 19 shows very nicely how the abstraction function with respect to $\mathbf{I}_\Sigma^\theta$ reverses the reification of abstract nodes of the Σ morphs admitted by a theory θ that is implicit in the definition of the canonical $\langle \Sigma, \theta \rangle$ interpretations, Definition 80. The canonical interpretations reify each Σ morph, $\langle \beta, \varrho, \lambda, \xi \rangle$, admitted by θ as a configuration of entities under $\langle \beta, \varrho, \lambda, \xi, |\varepsilon|_\varrho \rangle$. An easy way to get the set of Σ morphs admitted by θ back from $\mathbf{I}_\Sigma^\theta$ is therefore to collect the abstractions of all those entities in it that have the empty path as distinguished node. All other entities in $\mathbf{I}_\Sigma^\theta$ are congruent to exactly one of the entities in $\mathbf{I}_\Sigma^\theta$ with the empty path as distinguished node.

Now I can put the pieces together and show that the canonical models that I defined for each grammar in Definition 80 are indeed exhaustive models of their grammars.

Proposition 20 *For each signature Σ, for each $\theta \subseteq \mathcal{D}_0^\Sigma$,*

$\mathbf{I}_\Sigma^\theta$ *is an exhaustive $\langle \Sigma, \theta \rangle$ model.*

Proof
For each signature Σ, for each $\theta \subseteq \mathcal{D}_0^\Sigma$,

$\mathbf{I}_\Sigma^\theta$ *is a $\langle \Sigma, \theta \rangle$ model, and* *by proposition 15*

for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$,

I *is a $\langle \Sigma, \theta \rangle$ model*

$\Longrightarrow$ *for each $o \in \mathsf{U}$,*
$\mathsf{Abst}_\mathsf{I}(o) \in M_\Sigma(\theta)$ *by proposition 18*

$\Longrightarrow$ *for each $o \in \mathsf{U}$, for some $\langle \beta, \varrho, \lambda, \xi \rangle \in M_\Sigma(\theta)$,*
$\mathsf{Abst}_\mathsf{I}(o) = \langle \beta, \varrho, \lambda, \xi \rangle$*, and*
$\mathsf{Abst}_{\mathbf{I}_\Sigma^\theta}(\langle \beta, \varrho, \lambda, \xi, |\varepsilon|_\varrho \rangle) = \langle \beta, \varrho, \lambda, \xi \rangle$ *by proposition 19*

$\Longrightarrow$ *for each $o \in \mathsf{U}$, for some $o' \in \mathbf{U}_\Sigma^\theta$,*
$\langle o', \mathbf{I}_\Sigma^\theta \rangle$ *and $\langle o, \mathsf{I} \rangle$ are congruent in Σ* *by proposition 17*

$\Longrightarrow$ $\mathbf{I}_\Sigma^\theta$ *simulates I in Σ*

$\Longrightarrow$ $\mathbf{I}_\Sigma^\theta$ *is an exhaustive $\langle \Sigma, \theta \rangle$ model.* *by theorem 3*

■

Theorem 2, which I repeat below, follows immediately from Proposition 20:

Theorem 2 *For each signature Σ, for each theory θ, there exists a Σ interpretation I such that*

I *is an exhaustive $\langle \Sigma, \theta \rangle$ model.*

3.1.3 The Different Perspectives Revisited

In Sections 2.2.2.2–2.2.2.4 of Chapter 2 I discussed three explanations of the meaning of HPSG 94 grammars. King 1999 holds that a grammar delimits the actual and non-actual tokens of a natural language. Pollard and Sag 1994 sees the task of a grammar in describing the object types rather than the individual tokens of a natural language. Pollard 1999 is interested in the strong generative capacity of grammars. According to Pollard, the strong generative capacity of a grammar is a set of structures that does not contain any two members that are isomorphic, but for every idealization of a possible linguistic token in the language it contains exactly one structure that is isomorphic to it. All three views of the task of grammars were formalized in the SRL formalism, and I discussed how they are mathematically related, and in which respects they differ. The most significant linguistic difference between them consists in the different relationships in which the structures with which the three approaches are concerned stand to the empirical domain of observable linguistic events. According to King 1999, possible tokens are directly subject to linguistic description, and tokens of the language are (at least partly) empirically observable in utterance events. The meaning of the grammar is the natural language, and a natural language is a system of possible linguistic tokens. The object types of Pollard and Sag 1994 are mathematical representations of types of linguistic events. They are further removed from empirically observable entities, because they abstract away from individual linguistic events; they are representations of equivalence classes of linguistic tokens. The elements of the strong generative capacity of a grammar are unique representations of linguistic events. All linguistic events that are captured by the same representation are indistinguishable by any descriptions generated by the signature of the grammar. In contrast to the object types of Pollard and Sag 1994, the representations in the strong generative capacity are isomorphic to the linguistic events whose representations they are.

The proof that for each RSRL grammar, a model exists that is an exhaustive model of the grammar contains almost everything that is necessary to reconstruct all three explanations of the meaning of HPSG 94 grammars in RSRL. In the present section, I will briefly summarize the mathematical aspects of the reconstruction of the three approaches. I will not consider the merits and problems of the three approaches that have to do with their relationship to the empirical domain of linguistic data. In that respect, their formalization in RSRL does not differ from their realization in SRL. Therefore, all considerations discussed in Chapter 2 above apply without any modification.

The central concern of King 1999 is to state the conditions under which a grammar is true of a natural language. Substituting RSRL signatures, RSRL theories and RSRL interpretations of grammars for their SRL counterparts is almost sufficient to transfer King's three conditions to the extended formal framework. There is only one terminological point in King's third condition that deserves to be explained. So far, whenever I used the notion of a configuration of entities under an entity in an interpretation in the context of RSRL, I implicitly appealed to the analogous notion in SRL without presenting its RSRL counterpart. It is clear that with the introduction of relations in interpretations, the relations that hold between the components of an entity in an interpretation are now part of the configuration of entities under the entity. An interpretation under an entity in an interpretation must, therefore, be defined as follows:

Definition 83 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, *for each* $u \in \mathsf{I}$,

$\mathsf{I}_u = \langle \mathsf{U}_u, \mathsf{S}_u, \mathsf{A}_u, \mathsf{R}_u \rangle$ *is the* Σ ***interpretation under*** u ***in*** I *iff*

$$\begin{aligned}
&\mathsf{U}_u = \left\{ u' \in \mathsf{U} \,\middle|\, u' \in \mathsf{Co}^u_{\mathsf{I}} \right\}, \\
&\mathsf{S}_u = \mathsf{S} \cap (\mathsf{U}_u \times \mathcal{S}), \\
&\mathsf{A}_u = \mathsf{A} \cap (\mathcal{A} \times \{\mathsf{U}_u \times \mathsf{U}_u\}), \\
&\mathsf{R}_u = \mathsf{R} \cap \left(\mathcal{R} \times Pow \left(\bigcup_{n \in \mathbb{N}} \left(\overline{\mathsf{U}}_u \right)^n \right) \right), \textit{ and} \\
&\mathsf{I}_u = \langle \mathsf{U}_u, \mathsf{S}_u, \mathsf{A}_u, \mathsf{R}_u \rangle.
\end{aligned}$$

Definition 83 leads directly to the new definition of configurations of entities under an entity in an interpretation. Except for the reference to extended signatures, extended interpretations, and configurations under an entity in extended interpretations, it does not differ at all from its SRL counterpart in Definition 24.

Definition 84 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, *for each* $u \in \mathsf{I}$,

$\langle u, \mathsf{I}_u \rangle$ *is the* ***configuration of entities under*** u ***in*** I *iff*

I_u *is the* Σ *interpretation under* u *in* I.

The three conditions of King 1999, p. 343, can now straightforwardly be cast in terms of RSRL. The only difference to the original three conditions is that I am claiming that natural languages include relations between components (and chains of components) of tokens. A grammar $\langle \Sigma, \theta \rangle$ is true of a natural language only if

1. the natural language can be construed as an interpretation, I, of the signature Σ, where I is a system of linguistic tokens,
2. each description in θ is true of each entity in the natural language, i.e., the natural language, I, is a $\langle \Sigma, \theta \rangle$ model, and
3. some description in θ is false of some component of each entity, u', in any Σ interpretation, I', for which no entity u in the natural language, I, has isomorphically configured components.

Except for the reference to the extended definitions of all relevant terms, the conditions 1–3 are identical to King's conditions on SRL grammars that are summarized on page 99. The condition 3 amounts to saying that the language that is described by a grammar is in the class of exhaustive models of the grammar. Since the previous section showed that for each grammar, there exists an exhaustive model, it follows that King's interpretation of the meaning of a grammar is fully compatible with RSRL.

According to Pollard and Sag 1994, an HPSG 94 grammar is about the object types of a language. Following King 1996, I can reconstruct the types of a natural language as equivalence classes of indiscernible tokens of the language (which I assume to be an exhaustive model of a given grammar). For each signature Σ, for each Σ interpretation I, two entities, u_1 and u_2, in I, are in the relation $\equiv_{\mathsf{I}}$ iff $\langle u_1, \mathsf{I} \rangle$ and $\langle u_2, \mathsf{I} \rangle$ are indiscernible in Σ. It is easy to see that $\equiv_{\mathsf{I}}$ is an equivalence relation. If I is a natural language, I call the elements of $Quo(\langle \mathsf{I}, \equiv_{\mathsf{I}} \rangle)$ the types of the natural language, I. However, inasmuch as Pollard and Sag (1994) are explicit about object types, it is clear that they have a more direct relationship between grammars and object types in mind than one that is mediated by the tokens in the language. They think of the set of object types of a language as a set of mathematical representations, namely as the set of totally well-typed, sort resolved and potentially infinite abstract feature structures that are admitted by the grammar. The prime candidates for corresponding structures in RSRL are morphs, since morphs are abstract feature structures of the relevant kind that are extended by representations of relations, and the feature structure admission function of RSRL is the morph admission function. In fact, a closer look reveals that the Σ morphs admitted by a theory θ have exactly the right properties to serve as mathematical representations of the object types of the grammar $\langle \Sigma, \theta \rangle$. For each signature Σ, for each Σ theory θ, and for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, I is a $\langle \Sigma, \theta \rangle$ model exactly if the abstraction function with respect to I, Abst_I, is a total function from U to the set of Σ morphs admitted by θ in Σ; and I is an exhaustive $\langle \Sigma, \theta \rangle$ model exactly if Abst_I is a surjection from U to the set of Σ morphs admitted by θ in Σ. As a corollary, for each signature Σ, for each Σ theory θ, and for each Σ interpretation I, I is an exhaustive $\langle \Sigma, \theta \rangle$ model exactly if the abstraction function with respect to I determines a bijection from the types of I to the set of Σ morphs admitted by θ. This means that the types of an exhaustive model of a grammar stand in a one-to-one correspondence to the morphs admitted by the theory of the grammar. The morphs admitted by the theory of a grammar can therefore be regarded as mathematical representations of the object types of the language, and the morph admission function in Σ of DEFINITION 77 is a mathematical reconstruction of Pollard and Sag's notion of (abstract) feature structure admission in HPSG 94.

The definitions of the strong generative capacity of a grammar in Pollard 1999 build on canonical representations of pairs of entities and interpretations whose elements are all components of the entity in the pair, and on the notions of indiscernibility, models, and exhaustive models. None of them are affected in any way by adding relations to interpretations. Substituting RSRL signatures and RSRL interpretations for SRL signatures and SRL interpretations in Pollard's definitions automatically yields appropriate definitions of the strong generative capacity of RSRL grammars. Following Pollard's terminology, what DEFINITION 83 calls a Σ interpretation I_u under an entity u in I, is a Σ *interpretation generated by* u *in* I; and each configuration of entities under an entity u in a Σ interpretation I, $\langle u, \mathsf{I}_u \rangle$ (DEFINITION 84) is an *extended Pollard feature structure*. In the definitions of node abstractions in a signature and extended Pollard abstract feature structures, only the references to SRL signatures and to SRL interpretations need to be replaced. I repeat the slightly modified definitions (36) and (37) for convenience:

Definition 85 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, ν *is a* ***node abstraction in*** Σ *iff*

> ν *is a pair* $\langle \beta, \varrho \rangle$,

$\beta \subseteq \mathcal{T}^{\Sigma}$,

$: \in \beta$,

for each $\tau \in \mathcal{T}^{\Sigma}$*, for each* $\varphi \in \mathcal{A}$*, if* $\tau\varphi \in \beta$ *then* $\tau \in \beta$,

ϱ *is an equivalence relation over* β,

for each $\tau_1 \in \mathcal{T}^{\Sigma}$*, for each* $\tau_2 \in \mathcal{T}^{\Sigma}$*, for each* $\varphi \in \mathcal{A}$,

if $\tau_1\varphi \in \beta$ *and* $\langle \tau_1, \tau_2 \rangle \in \varrho$ *then* $\langle \tau_1\varphi, \tau_2\varphi \rangle \in \varrho$.

β is a prefix closed set of terms that contains at least the colon, and ϱ is a right invariant equivalence relation over β.

Definition 86 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$*, for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$*, for each* $u \in \mathsf{U}$,

$\langle u, \mathsf{I} \rangle$ *is an* ***extended Pollard abstract feature structure under*** Σ *iff*

$\langle u, \mathsf{I} \rangle$ *is an extended Pollard feature structure,*
for some node abstraction in Σ*,* $\langle \beta, \varrho \rangle$,

$\mathsf{U} = Quo(\langle \beta, \varrho \rangle)$,
$u = |:|_{\varrho}$,
for each $u_1 \in Quo(\langle \beta, \varrho \rangle)$*, for each* $u_2 \in Quo(\langle \beta, \varrho \rangle)$*, for each* $\varphi \in \mathcal{A}$,

$\mathsf{A}(\varphi)(u_1) = u_2$ *iff*
for some $\tau \in \beta$*,* $u_1 = |\tau|_{\varrho}$*, and* $u_2 = |\tau\varphi|_{\varrho}$.

Extended Pollard abstract feature structures are canonical representations of extended Pollard feature structures. Their nodes are equivalence classes of terms in β, and their root entities are in the equivalence class $|:|_{\varrho}$. No special provisions are necessary for relations, because the meaning of relations is defined in the theory of grammars, and they hold between the entities in configurations under an entity or, in other words, between the entities in extended Pollard feature structures. They are independent of the ontological status of the entities in the universe. For each signature Σ, I write $\mathbb{AFS}^{\Sigma}_{\text{eP}}$ for the set of extended Pollard abstract feature structures under Σ.

Since nothing relevant changed in the definition of extended Pollard abstract feature structures compared to the definition of Pollard abstract feature structures, and since indiscernibility in RSRL behaves just like indiscernibility in SRL, we immediately get the counterpart to PROPOSITION 4: For each signature Σ, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, for each $u \in \mathsf{U}$, if $\langle u, \mathsf{I} \rangle$ is an extended Pollard feature structure then there exists exactly one $\langle u', \mathsf{I}' \rangle \in \mathbb{AFS}^{\Sigma}_{\text{eP}}$, such that $\langle u, \mathsf{I} \rangle$ and $\langle u', \mathsf{I}' \rangle$ are indiscernible. It is therefore possible to define Pollard's abstraction function from the set of entities in an SRL interpretation to Pollard abstract feature structures for RSRL interpretations and extended Pollard abstract feature structures. The extended abstraction function assigns each entity u in an interpretation the extended Pollard abstract feature structure that is indiscernible from the configuration of entities under u. With the extended abstraction function, the strong generative capacity of a grammar can then be defined analogously to DEFINITION 39 as the set of all extended Pollard abstract feature structures that can be abstracted from the models of the grammar. Since RSRL grammars have exhaustive models, Pollard's second definition is also available: If we choose an exhaustive model of the grammar as starting point, it suffices to collect all abstractions of all entities in the exhaustive model to obtain the strong generative capacity of the grammar. As a concrete, fully spelled out example of a definition of the strong generative capacity of an RSRL grammar, I adapt Pollard's third definition (DEFINITION 44), because it does not require any auxiliary definitions:

Definition 87 *For each signature Σ, SGC is the total function from the set of grammars to $Pow(\mathbb{AFS}^{\Sigma}_{\text{eP}})$ such that, for each $\theta \subseteq \mathcal{D}^{\Sigma}_{0}$,*

$$\mathsf{SGC}(\langle \Sigma, \theta \rangle) = \left\{ \langle u, \mathsf{I} \rangle \in \mathbb{AFS}^{\Sigma}_{\text{eP}} \;\middle|\; \begin{array}{l} \langle v, \mathsf{I} \rangle \in \mathbb{AFS}^{\Sigma}_{\text{eP}}, \textit{ and} \\ \mathsf{I} \textit{ is a } \langle \Sigma, \theta \rangle \textit{ model} \end{array} \right\}.$$

For each grammar Γ, $\mathsf{SGC}(\Gamma)$ is the *strong generative capacity of* Γ.

The definition of the strong generative capacity of an RSRL grammar completes the overview of the explanations of the meaning of HPSG 94 grammars within the formal framework of RSRL. I conclude that RSRL is apt to accommodate the major proposals concerning the meaning of HPSG 94 grammars that have been put forth in the literature.

When I refer to the meaning of a grammar in the discussion of linguistic principles in Chapter 4 and in Chapter 5, I will use the notion of the exhaustive models of the grammar. With the results that we obtained in this section, it is clear that this terminology can be replaced with appropriate references to the strong generative capacity of the grammar or to the set of morphs admitted by the grammar, depending on which philosophical view of the meaning of a grammar the reader prefers. The differences between the three views are of such a deep, fundamental nature that they will not play any role for the purposes of my discussion. In any case, we know that RSRL can express all three of them.

3.2 A Notation for Linguistics

While RSRL is apt to formalize current HPSG grammars rigorously, it leaves its potential users with a syntax of descriptions that is rather arcane. All that linguists are really interested in for writing grammars are descriptions and, perhaps, formulae. A further distinction between terms and formulae is superfluous for their purposes, and potentially confusing. In addition, the syntax of descriptions above is entirely linear in contrast to the conventional, two-dimensional AVM diagrams. A linear notation necessitates multiple repetitions of attributes where a two-dimensional matrix notation only needs to mention them once, and the linear notation is, thus, much harder to read.[8] But for the lack of a definition of AVM diagrams as a formal language with an interpretation in RSRL, AVM diagrams can so far only be understood as sketches of the theory of the intended grammar, and it takes a logician and a discussion with the linguist about the intended meaning of her or his sketches to translate them into meaningful expressions of RSRL.

My goal in the current section is to close the mathematical gap between AVM diagrams and their empirical predictions. First, I will define an AVM language for RSRL grammars. Then I will show that each theory of a grammar that can be written with descriptions can also be written with AVM descriptions. In linguistic terms, each natural language that can be characterized with descriptions can be characterized with AVM descriptions. The AVM language is simply a different notation for writing grammars in RSRL.

[8] See the multiple occurrences of the attributes DTRS, APPEND and EXTRACT-SS in the formalization in SRL of the SUBCATEGORIZATION PRINCIPLE, (20), on page 139.

3.2.1 An AVM Syntax for RSRL

The basic building blocks of AVM diagrams are matrices, enclosed between square brackets, which allow a two-dimensional arrangement of attribute names and sort symbols. This way it is possible to avoid redundant repetitions of attribute names in formulae. Matrices can optionally be preceded by so-called *tags*, which are conventionally depicted as boxed integers, and are usually used to indicate that two (or more) paths share their value. Again, this notation avoids repetitions of attribute names, and it can combine describing path values with stating that they have identical values. For lists, there is a special graphical notation that enumerates the descriptions of the elements of a list as AVM diagrams between angled brackets. Since this notation is built into the graphics of AVM diagrams, we need to presuppose that the underlying signature provides at least the usual sorts, hierarchical relations, attributes and appropriateness conditions for lists: The symbols *nelist* and *elist* are species, they are the only immediate subsorts of sort *list*, no attributes are appropriate to *list* and *elist*, exactly FIRST and REST are appropriate to *nelist*, and $\mathcal{F}\langle nelist, \text{REST}\rangle = list$. I call a signature which comprises these definitions a *list signature*. From now on, whenever signatures are mentioned, list signatures are meant. Finally, AVM diagrams occasionally contain the negation symbol or the disjunction symbol with AVM diagrams as their argument(s). DEFINITION 88 below generalizes this practice to the standard logical connectives. Since RSRL includes chains, it is necessary to add an appropriate syntax of chains, which I derive from the syntax of lists, to the AVM syntax.

The AVM syntax will be defined in two steps. First, I define simple AVM diagrams, which I call boxes. Then I take certain boxes as basic units for potentially complex formulae which, among other constructs, might involve quantification and relations. For tags, I use the set of variables, $\mathcal{VAR}$. I write $\mathcal{VAR}^{:}$ as an abbreviation for the set $\mathcal{VAR} \cup \{:\}$. As notation for variables, I henceforth use the common boxed integers and lower case latin letters in italics, typically *v, x, y,* and *z*.

Definition 88 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR}\rangle$, $\mathbb{BOX}^\Sigma$, $\mathbb{UBOX}^\Sigma$ *and* $\mathbb{TBOX}^\Sigma$ *are the smallest sets such that*

$\mathbb{TBOX}^\Sigma \cup \mathbb{UBOX}^\Sigma \subseteq \mathbb{BOX}^\Sigma$,

for each $\sigma \in \widehat{\mathcal{G}}$, *for each* $\alpha_1 \in \widehat{\mathcal{A}}$, $\ldots$, *for each* $\alpha_n \in \widehat{\mathcal{A}}$, *for each* $\beta_1 \in \mathbb{BOX}^\Sigma$, $\ldots$, *for each* $\beta_n \in \mathbb{BOX}^\Sigma$,

$$\begin{bmatrix} \sigma & \\ \alpha_1 & \beta_1 \\ \vdots & \\ \alpha_n & \beta_n \end{bmatrix} \in \mathbb{UBOX}^{\Sigma},$$

for each $\beta_1 \in \mathbb{BOX}^{\Sigma}$, ..., *for each* $\beta_n \in \mathbb{BOX}^{\Sigma}$,

$$\langle \beta_1, \ldots, \beta_n \rangle \in \mathbb{UBOX}^{\Sigma},$$

for each $\beta \in \mathbb{BOX}^{\Sigma}$, $\neg\beta \in \mathbb{UBOX}^{\Sigma}$,
for each $\beta_1 \in \mathbb{BOX}^{\Sigma}$, *for each* $\beta_2 \in \mathbb{BOX}^{\Sigma}$, $[\beta_1 \wedge \beta_2] \in \mathbb{UBOX}^{\Sigma}$,
for each $\beta_1 \in \mathbb{BOX}^{\Sigma}$, *for each* $\beta_2 \in \mathbb{BOX}^{\Sigma}$, $[\beta_1 \vee \beta_2] \in \mathbb{UBOX}^{\Sigma}$,
for each $\beta_1 \in \mathbb{BOX}^{\Sigma}$, *for each* $\beta_2 \in \mathbb{BOX}^{\Sigma}$, $[\beta_1 \rightarrow \beta_2] \in \mathbb{UBOX}^{\Sigma}$,
for each $\beta_1 \in \mathbb{BOX}^{\Sigma}$, *for each* $\beta_2 \in \mathbb{BOX}^{\Sigma}$, $[\beta_1 \leftrightarrow \beta_2] \in \mathbb{UBOX}^{\Sigma}$,
for each $v \in \mathcal{VAR}$, *for each* $\beta_1 \in \mathbb{BOX}^{\Sigma}$, ..., *for each* $\beta_n \in \mathbb{BOX}^{\Sigma}$,

$$v \, \|\beta_1, \ldots, \beta_n\| \in \mathbb{TBOX}^{\Sigma},$$

for each $v \in \mathcal{VAR}^{:}$, *for each* $\beta \in \mathbb{UBOX}^{\Sigma}$, $v \ \beta \in \mathbb{TBOX}^{\Sigma}$.

I call each element of $\mathbb{TBOX}^{\Sigma}$ a *tagged* Σ *box*, each element of $\mathbb{UBOX}^{\Sigma}$ an *untagged* Σ *box*, and each element of $\mathbb{BOX}^{\Sigma}$ a Σ *box*. Notice that $\mathbb{TBOX}^{\Sigma} \cap \mathbb{UBOX}^{\Sigma} = \emptyset$ and $\mathbb{TBOX}^{\Sigma} \cup \mathbb{UBOX}^{\Sigma} = \mathbb{BOX}^{\Sigma}$. Untagged Σ boxes of the form $[\ \sigma\ ]$ and tagged Σ boxes of the form $v[\ \sigma\ ]$, where v is a variable, I call *atomic* Σ *matrices*. A Σ matrix need not be atomic. There can be any finite number of symbols of the expanded attribute set in vertical order under the symbol, σ, of the expanded sort set, each of them followed by a Σ box. I call a Σ box of that form a(n) *(untagged)* Σ *matrix*. A Σ box of the form $\langle \beta_1, \ldots, \beta_n \rangle$, where a finite number of arbitrary Σ boxes is enumerated between angled brackets, I call a(n) *(untagged)* Σ *list description*. Finally, Σ boxes can be combined with the standard logical connectives in the usual way. Note that untagged atomic Σ matrices, which fall under the choice of $n = 0$ in the clause defining Σ matrices, and Σ list descriptions with zero Σ boxes are the base cases of the inductive definition of untagged Σ boxes.

Every untagged Σ box can be turned into a tagged Σ box by writing a variable or the colon in front of it. The graphical notation for chains is similar to the notation for Σ list descriptions, except that in this case Σ boxes are enumerated between two vertical bars instead of angled brackets on each side. I call a complex tagged Σ box of the form $v\,\|\beta_1,\ldots,\beta_n\|$ a *(tagged) Σ chain description*. Note that Σ chain descriptions only occur as tagged Σ boxes in which the tag must be a variable, whereas all other types of tagged Σ boxes have untagged counterparts and may have the colon as a tag. This is a syntactic reflection of the fact that only formulae whose terms start with a variable can possibly denote chains. A chain notation with untagged Σ boxes would, thus, only supply expressions which would always denote the empty set.

Definition 88 provides a syntax for single AVM diagrams. Crucially, it does not include quantification and relations. Linguistic notation usually does not make quantification explicit, and there is no standard notation of relations. The following definition provides a pragmatic solution that fits both linguistic practice and the need for mathematical brevity.

Definition 89 *For each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR}\rangle$, $\mathbb{AVM}^\Sigma$ is the smallest set such that*

for each $\beta \in \mathbb{TBOX}^\Sigma$, $\beta \in \mathbb{AVM}^\Sigma$,

for each $\rho \in \mathcal{R}$, for each $v_1 \in \mathcal{VAR}$, …, for each $v_{\mathcal{AR}(\rho)} \in \mathcal{VAR}$,

$$\rho(v_1,\ldots,v_{\mathcal{AR}(\rho)}) \in \mathbb{AVM}^\Sigma,$$

for each $v_1 \in \mathcal{VAR}^{:}$, for each $v_2 \in \mathcal{VAR}^{:}$, $v_1 = v_2 \in \mathbb{AVM}^\Sigma$,

for each $v \in \mathcal{VAR}$, for each $\kappa \in \mathbb{AVM}^\Sigma$, $\exists v\, \kappa \in \mathbb{AVM}^\Sigma$,

for each $v \in \mathcal{VAR}$, for each $\kappa \in \mathbb{AVM}^\Sigma$, $\forall v\, \kappa \in \mathbb{AVM}^\Sigma$,

for each $\kappa \in \mathbb{AVM}^\Sigma$, $\neg\kappa \in \mathbb{AVM}^\Sigma$,

for each $\kappa_1 \in \mathbb{AVM}^\Sigma$, for each $\kappa_2 \in \mathbb{AVM}^\Sigma$, $(\kappa_1 \wedge \kappa_2) \in \mathbb{AVM}^\Sigma$,

for each $\kappa_1 \in \mathbb{AVM}^\Sigma$, for each $\kappa_2 \in \mathbb{AVM}^\Sigma$, $(\kappa_1 \vee \kappa_2) \in \mathbb{AVM}^\Sigma$,

for each $\kappa_1 \in \mathbb{AVM}^\Sigma$, for each $\kappa_2 \in \mathbb{AVM}^\Sigma$, $(\kappa_1 \rightarrow \kappa_2) \in \mathbb{AVM}^\Sigma$,

for each $\kappa_1 \in \mathbb{AVM}^\Sigma$, for each $\kappa_2 \in \mathbb{AVM}^\Sigma$, $(\kappa_1 \leftrightarrow \kappa_2) \in \mathbb{AVM}^\Sigma$.

I call each element of $\mathbb{AVM}^\Sigma$ a Σ *AVM formula*. Every tagged Σ box is also an atomic Σ AVM formula. I call each Σ formula of the form $\rho(v_1,\ldots,v_{\mathcal{AR}(\rho)})$ a *relational* Σ *AVM formula*, and each equation between elements of $\mathcal{VAR}^{:}$ a Σ *AVM equation*. Both relational Σ AVM formulae and Σ AVM equations are atomic Σ AVM formulae. Complex Σ AVM formulae can be formed with quantification and the standard logical connectives in the usual way. When working with a fixed signature, I refer to Σ boxes and Σ AVM formulae as boxes and AVM formulae, respectively.

With Σ AVM formulae in hand, it is possible to write HPSG principles in a format that is much more familiar to linguists than the RSRL description syntax of Section 3.1.1. As an example, consider the SUBCATEGORIZATION PRINCIPLE of Pollard and Sag (1994) written as an AVM formula:[9]

$$(31)\quad {}^{:}\begin{bmatrix} \textit{phrase} \\ \textsc{dtrs}\ [\textit{headed-struc}] \end{bmatrix} \rightarrow$$

$$\exists\boxed{1}\ \exists\boxed{2}\ \exists\boxed{3}\ \exists\boxed{4}$$

$$\left(\begin{array}{l} {}^{:}\begin{bmatrix} \textit{phrase} & \\ \textsc{synsem} & \begin{bmatrix} \textit{synsem} & \\ \textsc{loc} & \begin{bmatrix}\textit{loc} & \\ \textsc{cat} & \begin{bmatrix}\textit{cat} \\ \textsc{subcat}\ \boxed{1}\,[\textit{list}]\end{bmatrix}\end{bmatrix}\end{bmatrix} \\ \textsc{dtrs} & \begin{bmatrix}\textit{headed-struc} & \\ \textsc{head-dtr} & \begin{bmatrix}\textit{sign} & \\ \textsc{synsem} & \begin{bmatrix}\textit{synsem} & \\ \textsc{loc} & \begin{bmatrix}\textit{loc} & \\ \textsc{cat} & \begin{bmatrix}\textit{cat}\\ \textsc{subcat}\ \boxed{2}\,[\textit{list}]\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix} \\ \textsc{comp-dtrs} & \boxed{3}\,[\textit{list}]\end{bmatrix}\end{bmatrix} \\ \wedge\ \texttt{extract-ss}(\boxed{3},\boxed{4}) \\ \wedge\ \texttt{append}(\boxed{1},\boxed{4},\boxed{2}) \end{array}\right)$$

Following linguistic conventions, the variables in (31) are written as boxed integers. The antecedent of the implication is a matrix which is tagged with the colon. I write the colon at the upper left-hand corner of matrices. All variables in the consequent—which is a complex AVM formula—are bound by existential quantifiers, and the outermost matrix of the complex matrix in the consequent is again preceded by the colon. The outermost matrices of (potentially nested) matrices in an AVM formula I call *top matrices*. The

[9]For a discussion of the SUBCATEGORIZATION PRINCIPLE, see pages 249–252. The two relations, `append` and `extract-ss`, are defined in (37) and (72), respectively. In (31), I follow the bracketing conventions of first-order logic and leave out round brackets of AVM formulae where that cannot lead to confusion.

two matrices tagged with the colon are the top matrices of (31). In Section 3.2.3, I will introduce a few abbreviatory conventions which further simplify the AVM notation.

What remains to be done in Section 3.2.2 is to give Σ AVM formulae a meaning in RSRL and to show that Σ AVM formulae are indeed a complete notation for grammars.

3.2.2 Equivalence to RSRL Syntax

The meaning of Σ AVM formulae is determined by their translation to corresponding Σ formulae. First, I define a translation from pairs of Σ terms and Σ boxes, where the term is thought of as leading to the box, to Σ formulae. Then I use that translation to define the translation of Σ AVM formulae to Σ formulae.[10]

Definition 90 *For each signature Σ,* btrans *is the total function from the Cartesian product of $\mathcal{T}^\Sigma$ and $\mathbb{BOX}^\Sigma$ to $\mathcal{D}^\Sigma$,* utrans *is the total function from the Cartesian product of $\mathcal{T}^\Sigma$ and $\mathbb{UBOX}^\Sigma$ to $\mathcal{D}^\Sigma$, and* ttrans *is the total function from the Cartesian product of $\mathcal{T}^\Sigma$ and $\mathbb{TBOX}^\Sigma$ to $\mathcal{D}^\Sigma$ such that*

for each $\tau \in \mathcal{T}^\Sigma$, for each $\beta \in \mathbb{BOX}^\Sigma$,

$$\mathsf{btrans}(\tau, \beta) = \begin{cases} \mathsf{utrans}(\tau, \beta) & \textit{if } \beta \in \mathbb{UBOX}^\Sigma, \\ \mathsf{ttrans}(\tau, \beta) & \textit{if } \beta \in \mathbb{TBOX}^\Sigma \end{cases},$$

for each $\tau \in \mathcal{T}^\Sigma$, for each $\sigma \in \widehat{\mathcal{G}}$, for each $\alpha_1 \in \widehat{\mathcal{A}}$, ..., for each $\alpha_n \in \widehat{\mathcal{A}}$, for each $\beta_1 \in \mathbb{BOX}^\Sigma$, ..., for each $\beta_n \in \mathbb{BOX}^\Sigma$,

$$\mathsf{utrans}\left(\tau, \begin{bmatrix} \sigma & \\ \alpha_1 & \beta_1 \\ \vdots & \\ \alpha_n & \beta_n \end{bmatrix}\right) =$$
$$\left[\tau \sim \sigma \;\wedge\; \bigwedge\left\{\mathsf{btrans}(\tau\alpha_i, \beta_i) \;\middle|\; i \in \{1, \dots, n\}\right\}\right],$$

for each $\tau \in \mathcal{T}^\Sigma$, for each $\beta_1 \in \mathbb{BOX}^\Sigma$, ..., for each $\beta_n \in \mathbb{BOX}^\Sigma$,

[10] For convenience, I adopt the following meta-formula notation in Definition 90: For each $\delta \in \mathcal{D}^\Sigma$, for each $\delta_0 \in \mathcal{D}^\Sigma$, ..., for each $\delta_{n+1} \in \mathcal{D}^\Sigma$, $\bigwedge\{\delta\} \stackrel{\text{def}}{=} \delta$, and $\bigwedge\{\delta_0, \dots, \delta_n, \delta_{n+1}\} \stackrel{\text{def}}{=} [\bigwedge\{\delta_0, \dots, \delta_n\} \wedge \delta_{n+1}]$.

$$\begin{aligned}&\mathsf{utrans}\,(\tau, \langle\beta_1, \ldots, \beta_n\rangle) = \\ &[\tau \underbrace{\text{REST} \ \ldots \text{REST}}_{n \ \textit{times}} \sim \textit{elist} \ \land \\ &\bigwedge \left\{ \mathsf{btrans}(\tau \underbrace{\text{REST} \ \ldots \text{REST}}_{i-1 \ \textit{times}} \ \text{FIRST}, \beta_i) \,\middle|\, i \in \{1, \ldots, n\} \right\}\Bigg],\end{aligned}$$

for each $\tau \in \mathcal{T}^\Sigma$, *for each* $\beta \in \mathbb{BOX}^\Sigma$,

$$\mathsf{utrans}(\tau, \neg\beta) = [\tau \approx \tau \land \neg \ \mathsf{btrans}(\tau, \beta)],$$

for each $\tau \in \mathcal{T}^\Sigma$, *for each* $\beta_1 \in \mathbb{BOX}^\Sigma$, *for each* $\beta_2 \in \mathbb{BOX}^\Sigma$,

$$\mathsf{utrans}\,(\tau, [\beta_1 \land \beta_2]) = [\mathsf{btrans}(\tau, \beta_1) \land \mathsf{btrans}(\tau, \beta_2)],$$

for each $\tau \in \mathcal{T}^\Sigma$, *for each* $\beta_1 \in \mathbb{BOX}^\Sigma$, *for each* $\beta_2 \in \mathbb{BOX}^\Sigma$,

$$\mathsf{utrans}\,(\tau, [\beta_1 \lor \beta_2]) = [\mathsf{btrans}(\tau, \beta_1) \lor \mathsf{btrans}(\tau, \beta_2)],$$

for each $\tau \in \mathcal{T}^\Sigma$, *for each* $\beta_1 \in \mathbb{BOX}^\Sigma$, *for each* $\beta_2 \in \mathbb{BOX}^\Sigma$,

$$\mathsf{utrans}\,(\tau, [\beta_1 \rightarrow \beta_2]) = [\mathsf{btrans}(\tau, \beta_1) \rightarrow \mathsf{btrans}(\tau, \beta_2)],$$

for each $\tau \in \mathcal{T}^\Sigma$, *for each* $\beta_1 \in \mathbb{BOX}^\Sigma$, *for each* $\beta_2 \in \mathbb{BOX}^\Sigma$,

$$\mathsf{utrans}\,(\tau, [\beta_1 \leftrightarrow \beta_2]) = [\mathsf{btrans}(\tau, \beta_1) \leftrightarrow \mathsf{btrans}(\tau, \beta_2)], \ \textit{and}$$

for each $\tau \in \mathcal{T}^\Sigma$, *for each* $v \in \mathcal{VAR}$, *for each* $\beta_1 \in \mathbb{BOX}^\Sigma$, $\ldots$, *for each* $\beta_n \in \mathbb{BOX}^\Sigma$,

$$\begin{aligned}&\mathsf{ttrans}\,(\tau, v \,\|\beta_1, \ldots, \beta_n\|) = \\ &[\tau \approx v \ \land \ [\tau \underbrace{\triangleright \ldots \triangleright}_{n \ \textit{times}} \sim \textit{echain} \ \land \\ &\bigwedge \left\{ \mathsf{btrans}(\tau \ \underbrace{\triangleright \ldots \triangleright}_{i-1 \ \textit{times}} \ \dagger, \beta_i) \,\middle|\, i \in \{1, \ldots, n\} \right\}\Bigg]\Bigg],\end{aligned}$$

for each $\tau \in \mathcal{T}^\Sigma$, *for each* $v \in \mathcal{VAR}^{:}$, *for each* $\beta \in \mathbb{UBOX}^\Sigma$,

$$\mathsf{ttrans}(\tau, v \ \beta) = [\tau \approx v \land \mathsf{utrans}(\tau, \beta)].$$

I call btrans the Σ *box translation function.* The basic intuition behind the definition of btrans is that the Σ term, τ, leads to the Σ box, β, and we prefix all paths with τ when we translate β. btrans distinguishes between the case when β is an untagged Σ box, which is treated by utrans, and the case when β is a tagged Σ box, which is treated by ttrans.

A Σ term, τ, and an untagged Σ matrix with the symbol, σ, from the expanded sort set, the symbols α_1 to α_n from the expanded attribute set followed by the Σ boxes β_1 to β_n, respectively, are translated to the Σ formula consisting of the sort assignment formula $\tau \sim \sigma$ and n conjuncts in which, for each line i of the Σ matrix, btrans is applied to the Σ term $\tau\alpha_i$ and the Σ box β_i. A Σ term, τ, and an untagged Σ list description, $\langle \beta_1, \ldots, \beta_n \rangle$, are translated to the Σ formula consisting of the sort assignment formula $\tau' \sim elist$, where τ' is the Σ term which consists of τ followed by a string of n REST attributes, and n conjuncts in which, for i from 1 to n, btrans is applied to τ followed by $i-1$ REST attributes and the ith Σ box in the Σ list description. Note that, according to the Σ formula interpretation function, our definition of btrans makes sure that the Σ formula resulting from the translation of any pair, $\langle \tau, \beta \rangle$, can only be true of an entity, u, if $T_{\mathsf{I}}^{ass}(\tau)(u)$ is defined. This property of the Σ box translation function also holds for the translation of Σ terms, τ, and a negated untagged Σ box, $\neg\beta$, where the only purpose of the first conjunct of the translation, $\tau \approx \tau$, is to enforce it. The reason I define btrans this way is, of course, that I think that it correctly reflects how linguists usually interpret their informal AVM diagrams. The underlying intuition is illustrated in (32). ‘: CASE’ is the relevant term, τ, that leads to the negated untagged box, $\neg[acc]$. Supposing the signature of the appendix of Pollard and Sag 1994, I assume that the AVM formula in (32) describes *head* entities on which $T_{\mathsf{I}}^{ass}(:\text{CASE})$ is defined and has values that are not of sort *acc*.

(32) $:\begin{bmatrix} head \\ \text{CASE} \ \neg \left[acc\right] \end{bmatrix}$

The translation of Σ terms and untagged Σ boxes combined by the standard logical connectives is straightforward.

For the translation of Σ terms, τ, and tagged Σ boxes, $v\ \beta$, the function ttrans simply uses utrans on τ and the untagged Σ box β, which we have already defined, and conjoins the resulting Σ formula with the path equation $\tau \approx v$. The only tagged Σ boxes this definition cannot apply to are Σ

chain descriptions, since they do not have untagged counterparts. Therefore, there is a special definition for Σ chain descriptions which mirrors the case of untagged Σ list descriptions but adds the path equation $\tau \approx v$.

We are now ready to translate AVM formulae:

Definition 91 *For each signature* Σ, trans *is the total function from* $\mathbb{AVM}^\Sigma$ *to* $\mathcal{D}^\Sigma$ *such that*

for each $v \in \mathcal{VAR}$, *for each* $\beta_1 \in \mathbb{BOX}^\Sigma$, $\ldots$, *for each* $\beta_n \in \mathbb{BOX}^\Sigma$,

$$\mathsf{trans}\,(v\,\|\beta_1, \ldots, \beta_n\|) = \mathsf{ttrans}\,(v, v\,\|\beta_1, \ldots, \beta_n\|),$$

for each $v \in \mathcal{VAR}^{:}$, *for each* $\beta \in \mathbb{UBOX}^\Sigma$,

$$\mathsf{trans}(v\ \beta) = \mathsf{ttrans}(v, v\ \beta),$$

for each $\rho \in \mathcal{R}$, *for each* $v_1 \in \mathcal{VAR}$, $\ldots$, *for each* $v_{\mathcal{AR}(\rho)} \in \mathcal{VAR}$,

$$\mathsf{trans}\left(\rho(v_1, \ldots, v_{\mathcal{AR}(\rho)})\right) = \rho(v_1, \ldots, v_{\mathcal{AR}(\rho)}),$$

for each $v_1 \in \mathcal{VAR}^{:}$, *for each* $v_2 \in \mathcal{VAR}^{:}$,

$$\mathsf{trans}\,(v_1 = v_2) = v_1 \approx v_2,$$

for each $v \in \mathcal{VAR}$, *for each* $\kappa \in \mathbb{AVM}^\Sigma$,

$$\mathsf{trans}\,(\exists v\ \kappa) = \exists v\ \mathsf{trans}(\kappa),$$

for each $v \in \mathcal{VAR}$, *for each* $\kappa \in \mathbb{AVM}^\Sigma$,

$$\mathsf{trans}\,(\forall v\ \kappa) = \forall v\ \mathsf{trans}(\kappa),$$

for each $\kappa \in \mathbb{AVM}^\Sigma$,

$$\mathsf{trans}\,(\neg\kappa) = \neg\mathsf{trans}(\kappa),$$

for each $\kappa_1 \in \mathbb{AVM}^\Sigma$, *for each* $\kappa_2 \in \mathbb{AVM}^\Sigma$,

$$\mathsf{trans}\,((\kappa_1 \wedge \kappa_2)) = [\mathsf{trans}(\kappa_1) \wedge \mathsf{trans}(\kappa_2)],$$

for each $\kappa_1 \in \mathbb{AVM}^{\Sigma}$, *for each* $\kappa_2 \in \mathbb{AVM}^{\Sigma}$,

$$\mathsf{trans}\left((\kappa_1 \vee \kappa_2)\right) = [\mathsf{trans}(\kappa_1) \vee \mathsf{trans}(\kappa_2)],$$

for each $\kappa_1 \in \mathbb{AVM}^{\Sigma}$, *for each* $\kappa_2 \in \mathbb{AVM}^{\Sigma}$,

$$\mathsf{trans}\left((\kappa_1 \rightarrow \kappa_2)\right) = [\mathsf{trans}(\kappa_1) \rightarrow \mathsf{trans}(\kappa_2)],$$

for each $\kappa_1 \in \mathbb{AVM}^{\Sigma}$, *for each* $\kappa_2 \in \mathbb{AVM}^{\Sigma}$,

$$\mathsf{trans}\left((\kappa_1 \leftrightarrow \kappa_2)\right) = [\mathsf{trans}(\kappa_1) \leftrightarrow \mathsf{trans}(\kappa_2)].$$

I call trans the Σ *AVM translation function.* On the basis of the Σ box translation function, the definition of trans is fairly trivial. The function trans applied to a Σ AVM formula which is a tagged Σ box is simply the ttrans value of the tag, v, of that formula and the formula itself. Note that the path equation $v \approx v$, which ttrans introduces into the translation of v and a Σ box, is always true of all entities in any interpretation. Σ chain descriptions must, again, be translated separately, since they do not have untagged twins. Relational Σ AVM formulae and Σ AVM equations are trivially translated to their counterparts, relational Σ formulae and path equations. For quantification and complex Σ AVM formulae involving the logical connectives, the translation is likewise straightforward. Henceforth, when I talk about the denotation of a Σ AVM formula, κ, I mean the denotation of the translation of κ, $\mathsf{trans}(\kappa)$.

For each Σ AVM formula, κ, I define the set of free occurrences of variables in κ. I will need this definition below to show that free variables are preserved under the translation from AVM syntax to RSRL syntax.

Definition 92 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$,

for each $\sigma \in \widehat{\mathcal{G}}$, *for each* $\alpha_1 \in \widehat{\mathcal{A}}$, ..., *for each* $\alpha_n \in \widehat{\mathcal{A}}$, *for each* $\beta_1 \in \mathbb{BOX}^{\Sigma}$, ..., *for each* $\beta_n \in \mathbb{BOX}^{\Sigma}$,

$$FV^{AVM}\left(\begin{bmatrix} \sigma & \\ \alpha_1 & \beta_1 \\ \vdots & \\ \alpha_n & \beta_n \end{bmatrix}\right) = \left\{ v \in \mathcal{VAR} \,\middle|\, \begin{array}{c} \textit{for some } i \in \{1, \ldots, n\}, \\ v \in FV^{AVM}(\beta_i) \end{array} \right\},$$

for each $\beta_1 \in \mathbb{BOX}^\Sigma$, ..., *for each* $\beta_n \in \mathbb{BOX}^\Sigma$,

$$FV^{AVM}(\langle \beta_1, \ldots, \beta_n \rangle) = \left\{ v \in \mathcal{VAR} \,\middle|\, \begin{array}{c} \textit{for some } i \in \{1, \ldots, n\}, \\ v \in FV^{AVM}(\beta_i) \end{array} \right\},$$

for each $\beta \in \mathbb{BOX}^\Sigma$, $FV^{AVM}(\neg\beta) = FV^{AVM}(\beta)$,

for each $\beta_1 \in \mathbb{BOX}^\Sigma$, *for each* $\beta_2 \in \mathbb{BOX}^\Sigma$,

$$FV^{AVM}([\beta_1 \wedge \beta_2]) = FV^{AVM}(\beta_1) \cup FV^{AVM}(\beta_2),$$

for each $\beta_1 \in \mathbb{BOX}^\Sigma$, *for each* $\beta_2 \in \mathbb{BOX}^\Sigma$,

$$FV^{AVM}([\beta_1 \vee \beta_2]) = FV^{AVM}(\beta_1) \cup FV^{AVM}(\beta_2),$$

for each $\beta_1 \in \mathbb{BOX}^\Sigma$, *for each* $\beta_2 \in \mathbb{BOX}^\Sigma$,

$$FV^{AVM}([\beta_1 \rightarrow \beta_2]) = FV^{AVM}(\beta_1) \cup FV^{AVM}(\beta_2),$$

for each $\beta_1 \in \mathbb{BOX}^\Sigma$, *for each* $\beta_2 \in \mathbb{BOX}^\Sigma$,

$$FV^{AVM}([\beta_1 \leftrightarrow \beta_2]) = FV^{AVM}(\beta_1) \cup FV^{AVM}(\beta_2),$$

for each $v \in \mathcal{VAR}$, *for each* $\beta_1 \in \mathbb{BOX}^\Sigma$, ..., *for each* $\beta_n \in \mathbb{BOX}^\Sigma$,

$$\begin{aligned} &FV^{AVM}(v \,\|\beta_1, \ldots, \beta_n\|) \\ &= \{v\} \cup \left\{ v' \in \mathcal{VAR} \,\middle|\, \begin{array}{c} \textit{for some } i \in \{1, \ldots, n\}, \\ v' \in FV^{AVM}(\beta_i) \end{array} \right\}, \end{aligned}$$

for each $v \in \mathcal{VAR}^{:}$, *for each* $\beta \in \mathbb{UBOX}^\Sigma$,

$$FV^{AVM}(v\ \beta) = FV(v) \cup FV^{AVM}(\beta),$$

for each $\rho \in \mathcal{R}$, *for each* $v_1 \in \mathcal{VAR}$, ..., *for each* $v_{\mathcal{AR}(\rho)} \in \mathcal{VAR}$,

$$FV^{AVM}\left(\rho(v_1, \ldots, v_{\mathcal{AR}(\rho)})\right) = \{v_1, \ldots, v_{\mathcal{AR}(\rho)}\},$$

for each $v_1 \in \mathcal{VAR}^{:}$, *for each* $v_2 \in \mathcal{VAR}^{:}$,

$$FV^{AVM}(v_1 = v_2) = FV(v_1) \cup FV(v_2),$$

for each $v \in \mathcal{VAR}$, *for each* $\kappa \in \mathbb{AVM}^{\Sigma}$,

$$FV^{AVM}(\exists v\, \kappa) = FV^{AVM}(\kappa)\backslash\{v\},$$

for each $v \in \mathcal{VAR}$, *for each* $\kappa \in \mathbb{AVM}^{\Sigma}$,

$$FV^{AVM}(\forall v\, \kappa) = FV^{AVM}(\kappa)\backslash\{v\},$$

for each $\kappa \in \mathbb{AVM}^{\Sigma}$, $FV^{AVM}(\neg\kappa) = FV^{AVM}(\kappa)$,
for each $\kappa_1 \in \mathbb{AVM}^{\Sigma}$, *for each* $\kappa_2 \in \mathbb{AVM}^{\Sigma}$,

$$FV^{AVM}((\kappa_1 \wedge \kappa_2)) = FV^{AVM}(\kappa_1) \cup FV^{AVM}(\kappa_2),$$

for each $\kappa_1 \in \mathbb{AVM}^{\Sigma}$, *for each* $\kappa_2 \in \mathbb{AVM}^{\Sigma}$,

$$FV^{AVM}((\kappa_1 \vee \kappa_2)) = FV^{AVM}(\kappa_1) \cup FV^{AVM}(\kappa_2),$$

for each $\kappa_1 \in \mathbb{AVM}^{\Sigma}$, *for each* $\kappa_2 \in \mathbb{AVM}^{\Sigma}$,

$$FV^{AVM}((\kappa_1 \rightarrow \kappa_2)) = FV^{AVM}(\kappa_1) \cup FV^{AVM}(\kappa_2),$$

for each $\kappa_1 \in \mathbb{AVM}^{\Sigma}$, *for each* $\kappa_2 \in \mathbb{AVM}^{\Sigma}$,

$$FV^{AVM}((\kappa_1 \leftrightarrow \kappa_2)) = FV^{AVM}(\kappa_1) \cup FV^{AVM}(\kappa_2).$$

Let $\mathbb{AVM}_0^{\Sigma}$ be the set of Σ AVM formulae without free variables. I call each element of $\mathbb{AVM}_0^{\Sigma}$ a Σ *AVM description.* When working with a fixed signature, I simply refer to them as AVM descriptions.

We can now show the correspondence between the set of variables that occur free in expressions of our AVM language and in the translations of those expressions. First, we must show the correspondence for btrans. Lemma 1 says that the set of variables that occur free in the Σ formula obtained from applying the Σ box translation function, btrans, to a Σ term, τ, and a Σ box, β, equals the union of the set of free variables in τ and the set of free variables in β.[11]

Lemma 1 *For each signature* Σ, *for each* $\tau \in \mathcal{T}^{\Sigma}$, *for each* $\beta \in \mathbb{BOX}^{\Sigma}$,

[11]The complete proofs of Lemma 1–Theorem 4 can be found in Appendix B.

$$FV\,(\mathsf{btrans}(\tau, \beta)) = FV(\tau) \cup FV^{AVM}(\beta).$$

Proof
By induction on the complexity of β. ■

With LEMMA 1 in hand it is easy to show that the set of free variables in a Σ AVM formula, κ, equals the set of free variables in the Σ formula obtained when we apply the Σ AVM translation function, trans, to κ:

Proposition 21 *For each signature Σ, for each $\kappa \in \mathbb{AVM}^{\Sigma}$,*

$$FV^{AVM}(\kappa) = FV(\mathsf{trans}(\kappa)).$$

Proof
By induction on the complexity of κ and LEMMA 1*.* ■

An important special case of this result is that our translation of a Σ AVM description is always a Σ description. With COROLLARY 2 we thus know that, as for Σ descriptions, the meaning of Σ AVM descriptions is independent of the variable assignments in I.

The next proposition is essential for the main result of this section, THEOREM 4. It says that, for each signature Σ and each Σ formula δ, we can find a Σ AVM formula κ such that the same variables occur free in δ and κ and, for certain variable assignments, δ and κ denote the same set of entities in each interpretation.

Proposition 22 *For each signature Σ, for each $\delta \in \mathcal{D}^{\Sigma}$, there exists a $\kappa \in \mathbb{AVM}^{\Sigma}$ such that, for each interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, for each $u \in \mathsf{U}$, for each $ass \in Ass_I$,*

$FV(\delta) = FV^{AVM}(\kappa)$, *and*

if for each $v \in FV(\delta)$, $ass(v) \in \overline{\mathsf{Co}_{\mathsf{I}}^{u}}$,

then $u \in D_{\mathsf{I}}^{ass}(\delta)$ iff $u \in D_{\mathsf{I}}^{ass}(\mathsf{trans}(\kappa))$.

Proof
By induction on the complexity of δ. Sketch: For each $\delta \in \mathcal{D}^{\Sigma}$ I have to provide a $\kappa \in \mathbb{AVM}^{\Sigma}$ such that the proposition holds. Base cases:

For each $n \in \mathbb{N}$, *for* $\delta = v\alpha_1 \ldots \alpha_n \sim \sigma$, *choose* $v\begin{bmatrix} metatop \\ \alpha_1 \ \ldots \begin{bmatrix} metatop \\ \alpha_n \ [\sigma] \end{bmatrix} \ldots \end{bmatrix}$.
For each $n \in \mathbb{N}$, *for each* $m \in \mathbb{N}$, *for each* $\delta = v\alpha_1 \ldots \alpha_n \approx v'\alpha'_1 \ldots \alpha'_m$,
if $n = 0$ *and* $m = 0$, *choose* $v = v'$,
if $n \geq 1$ *and* $m = 0$, *choose* $v\begin{bmatrix} metatop \\ \alpha_1 \ \ldots \begin{bmatrix} metatop \\ \alpha_n \ v' \ [metatop] \end{bmatrix} \ldots \end{bmatrix}$,
if $n = 0$ *and* $m \geq 1$, *analogous to the previous case,*
if $n \geq 1$ *and* $m \geq 1$, *choose*

$$\exists v'' \left(v\begin{bmatrix} metatop \\ \alpha_1 \ \ldots \begin{bmatrix} metatop \\ \alpha_n \ v'' \ [metatop] \end{bmatrix} \ldots \end{bmatrix} \wedge \ v'\begin{bmatrix} metatop \\ \alpha'_1 \ \ldots \begin{bmatrix} metatop \\ \alpha'_m \ v'' \ [metatop] \end{bmatrix} \ldots \end{bmatrix} \right).$$

For each relational Σ *formula, choose the identical relational* Σ *AVM formula.*
The proposition follows by induction. ■

Everything is now in place for the main theorem. Informally, THEOREM 4 says that for each Σ description, δ, there exists a Σ AVM description, κ, such that δ and κ denote the same set of entities in each interpretation.

Theorem 4 *For each signature* Σ, *for each* $\delta \in \mathcal{D}^\Sigma_0$, *there exists a* $\kappa \in \mathbb{AVM}^\Sigma_0$ *such that, for each interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, *for each* $u \in \mathsf{U}$, *for each* $ass \in Ass_I$,

$$u \in D_\mathsf{I}(\delta) \textit{ iff } u \in D_\mathsf{I}(\mathsf{trans}(\kappa)).$$

Proof
Follows immediately from PROPOSITION 22. ■

THEOREM 4 has significant consequences. Since the theory, θ, of a grammar, $\Gamma = \langle \Sigma, \theta \rangle$, is a set of descriptions and, for each description, δ, we can find an AVM description which is true of an entity in any interpretation of Γ exactly if δ is true of it, we can write θ as a set of AVM descriptions, θ^{AVM}, rather than as a set of descriptions. Since each element of θ has a corrensponding element with the same denotation in θ^{AVM}, the grammar $\langle \Sigma, \theta \rangle$ and the AVM grammar $\left\langle \Sigma, \theta^{AVM} \right\rangle$ have the same models and, thus, the same exhaustive models. But in linguistic terms that means that for each grammar, Γ, there is an AVM grammar, Γ^{AVM}, which is true of a natural language exactly if Γ is true of it. In other words, it does not matter whether one chooses RSRL's original syntax or the AVM syntax to write a grammar. For linguistic purposes, they are equivalent.

3.2.3 Abbreviatory Conventions

With AVM descriptions, we have a convenient syntax for specifying theories of HPSG 94 grammars. However, a few abbreviatory conventions tailored to the specific purpose of our formal language can improve the usefulness of the new notation further. The principal application of the AVM syntax is to write grammatical principles, or AVM descriptions, which lack free occurrences of variables. I will, therefore, introduce a number of notational conventions which take this into account and bring AVM descriptions even closer to actual linguistic practice. Throughout this section, if nothing else is indicated, I presuppose the signature of the appendix of Pollard and Sag 1994 and assume that it is augmented by the necessary relations, which the authors leave implicit.

As I have already done in example (31) above, I adopt the usual bracketing conventions of first-order predicate logic and leave out the round brackets of AVM formulae when the omission cannot lead to confusion. With those conventions, the Head Feature Principle of Pollard and Sag (1994, p. 399), written as an AVM description, looks as follows:

$$
\text{(33)}\quad \overset{:}{}\begin{bmatrix} \textit{phrase} \\ \text{DTRS}\ \left[\textit{headed-struc}\right] \end{bmatrix} \quad \rightarrow
$$

$$
\exists \boxed{1}\ \overset{:}{}\begin{bmatrix} \textit{phrase} \\ \text{SYNSEM} & \begin{bmatrix} \textit{synsem} \\ \text{LOC} & \begin{bmatrix} \textit{loc} \\ \text{CAT} & \begin{bmatrix} \textit{cat} \\ \text{HEAD}\ \boxed{1}\ \left[\textit{head}\right] \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{DTRS} & \begin{bmatrix} \textit{headed-struc} \\ \text{HEAD-DTR} & \begin{bmatrix} \textit{sign} \\ \text{SYNSEM} & \begin{bmatrix} \textit{synsem} \\ \text{LOC} & \begin{bmatrix} \textit{loc} \\ \text{CAT} & \begin{bmatrix} \textit{cat} \\ \text{HEAD}\ \boxed{1}\ \left[\textit{head}\right] \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}
$$

There are a few notational properties that distinguish (33) from how linguists would write the HEAD FEATURE PRINCIPLE. First, the colon, which functions as the tag of the top matrix in the antecedent and in the consequent, is a particular feature of our AVM descriptions. Since by definition every top matrix in an AVM description must be tagged, I can introduce a convention that determines which unique tag I mean when I leave top matrices untagged:

Convention 1 *The colon as the tag of a top matrix may be left out in AVM descriptions.*

Second, in every matrix there must be a symbol from the expanded sort set at the top. Clearly, this is not necessary for practical purposes, since most sort symbols in the matrices of (33) do not restrict the meaning of the description. On the other hand, they make the matrices look much more complex than they really are. It is, thus, convenient to leave those symbols out. Since *metatop* denotes all entities and chains of the universe, it can always be assumed that *metatop* should be inserted where no other symbol stands.

Convention 2 *At the top of matrices, the symbol from the expanded sort set may be left out. If it is missing, then I regard metatop as the missing symbol.*

Third, once superfluous sort symbols are left out, (33) contains many angled brackets of matrices that do not serve any real graphical purpose, because there is only one attribute in the matrix. If the brackets do anything at all, they make the description harder to read. This is inappropriate for a practical notation, and I eliminate this notational overhead:

Convention 3 *If a matrix which is not a top matrix contains only one attribute and no symbol from the expanded sort set or no attribute at all, then the brackets of that matrix may be left out. The brackets of a top matrix that only contains a symbol from the expanded sort set may also be omitted.*

Finally, the single variable in (33) is existentially bound. As an investigation of the use of tags in the HPSG literature in the light of the meaning of AVM descriptions shows, they are often to be understood as being existentially bound variables, and, as in the HEAD FEATURE PRINCIPLE, it can be assumed that the existential quantifier which binds the variables takes wide scope. It is, therefore, practical to leave wide scope existential quantification implicit in grammatical principles, and only notate quantifiers which are exceptions to this default rule.

Convention 4 *If a tag is not explicitly bound by a quantifier, then I assume that it is captured by an implicit existential closure with wide scope over the entire formula.*

Applying the notational conventions 1, 2, 3, and 4 to the HEAD FEATURE PRINCIPLE leads to the following notation:

$$(34)\quad \begin{bmatrix} \textit{phrase} \\ \text{DTRS} \ \ \textit{headed-struc} \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} \text{SYNSEM LOC CAT HEAD} & \boxed{1} \\ \text{DTRS HEAD-DTR SYNSEM LOC CAT HEAD} & \boxed{1} \end{bmatrix}$$

Notice that, in accordance with the conventions 2 and 3, the tag $\boxed{1}$ in the consequent of example (34) abbreviates the tagged matrix $\boxed{1}[\textit{metatop}]$. Notice also that, according to CONVENTION 4, the existential quantifier takes scope over the entire formula, in contrast to (33), where the existential only quantifies over the consequent. Standard results show that the two formulations are equivalent, since the tag $\boxed{1}$ does not occur in the antecedent.

Finally, it proves to be useful for the description of lists and chains to borrow the standard notation of the programming language Prolog for referring to the head and the tail of a list or chain:

Convention 5 *For each signature Σ, for each Σ AVM formula ϕ_1, and for each Σ AVM formula ϕ_2:*

$$\langle \phi_1 | \phi_2 \rangle = \begin{bmatrix} \textit{list} \\ \text{FIRST} & \phi_1 \\ \text{REST} & \phi_2 \end{bmatrix}, \ \textit{and} \ \ \| \phi_1 | \phi_2 \| = \begin{bmatrix} \textit{chain} \\ \dagger & \phi_1 \\ \triangleright & \phi_2 \end{bmatrix}.$$

ϕ_1 and ϕ_2 may of course employ any of the abbreviatory conventions which are defined in this section.

HPSG 94 grammars make intense use of relations. In theories, two types of appearances of relational AVM formulae can be distinguished. First, the theory of a grammar contains descriptions which serve to define the intended meaning of a relation, e.g., the meaning of `append` or `member`. Those descriptions are formally on a par with other principles of grammar such as the SUBCATEGORIZATION PRINCIPLE or the HEAD FEATURE PRINCIPLE, and, to emphasize this similarity, one could call them the `append` principle, the `member` principle, etc. Informally, they are sometimes called the definitions (of the meaning) of the relations. In Section 3.1.1, I have used the example of the `member` relation in (29) to discuss how a relation principle receives the intended meaning in the models of a grammar. Second, relational AVM formulae occur in other grammatical principles such us the SUBCATEGORIZATION PRINCIPLE, where they serve to ensure certain properties of configurations of linguistic entities. In this case, the occurrence of the relation makes use of the meaning of the relation which is defined in the relation principles.

Before I discuss the syntax of AVM descriptions which define the meaning of relations, I introduce a notational convention that is common linguistic practice. The lexical entry of the German verb *haben* shown in (35) is adapted from Hinrichs and Nakazawa 1994, p. 25. It is a typical example of a so-called argument raising predicate.[12]

(35)
$$\left[\begin{array}{ll} \textit{word} & \\ \text{PHON} & \langle \textit{haben} \rangle \\ \text{SYNSEM LOC CAT} & \left[\begin{array}{ll} \text{HEAD} & \left[\begin{array}{ll} \text{MAJ} & \textit{verb} \\ \text{AUX} & + \\ \text{FLIP} & \boxed{2} \end{array}\right] \\ \text{SUBCAT} & \boxed{3} \\ \text{NPCOMP} & - \end{array}\right] \end{array}\right]$$
$$\wedge\ \boxed{4} \left\langle \left[\begin{array}{ll} \text{LOC CAT} & \left[\begin{array}{ll} \text{HEAD} & \left[\begin{array}{ll} \text{MAJ} & \textit{verb} \\ \text{VFORM} & \textit{psp} \\ \text{FLIP} & \boxed{2} \end{array}\right] \\ \text{SUBCAT} & \boxed{1} \\ \text{NPCOMP} & - \end{array}\right] \end{array}\right] \right\rangle$$
$$\wedge\ \texttt{append}(\boxed{1}, \boxed{4}, \boxed{3})$$

The lexical entry of *haben* in (35) makes use of all conventions discussed so far, except for the list convention 5. The colon at the first top matrix is left implicit; most matrices do not contain sort symbols; most matrix brackets are omitted; and all tags are bound by implicit existential closure. However, contrary to the common notation in the HPSG literature, the description of the second argument of the `append` relation is written as a separate conjunct

[12]For this example to go through, I must change the signature of the appendix of Pollard and Sag 1994 slightly by adding the *head* attributes MAJ and FLIP, and the *cat* attribute NPCOMP, with their respective appropriateness specifications.

of the overall AVM description instead of being part of the relational AVM formula. Since the latter option seems to be an elegant way of representing the same description in a more compact fashion, I permit this notation:

Convention 6 *A tagged box, ϕ, may be written into an argument slot of a relational AVM formula. This notation is interpreted as abbreviating a conjunction of ϕ with the relational AVM formula which contains only the tag of ϕ in the respective argument slot. If the tag of ϕ is the colon, the notation is interpreted as described, except that the relevant argument slot of the relational AVM formula now contains the alphabetically first variable, v, which does not occur yet in the overall description, and the AVM equation $: = v$ is conjoined with the relational AVM formula and ϕ.*

According to Convention 6, the lexical entry of *haben* can be rewritten as follows:

$$(36)\quad \left[\begin{array}{ll} \textit{word} & \\ \text{PHON} & \langle \textit{haben} \rangle \\ \text{SYNSEM LOC CAT} & \left[\begin{array}{ll} \text{HEAD} & \left[\begin{array}{ll}\text{MAJ} & \textit{verb}\\ \text{AUX} & + \\ \text{FLIP} & \boxed{2}\end{array}\right] \\ \text{SUBCAT} & \boxed{3} \\ \text{NPCOMP} & - \end{array}\right]\end{array}\right]$$

$$\wedge\ \texttt{append}\left(\boxed{1}, \boxed{4}\left\langle \left[\begin{array}{ll}\text{LOC CAT} & \left[\begin{array}{ll} \text{HEAD} & \left[\begin{array}{ll}\text{MAJ} & \textit{verb}\\ \text{VFORM} & \textit{psp} \\ \text{FLIP} & \boxed{2}\end{array}\right] \\ \text{SUBCAT} & \boxed{1} \\ \text{NPCOMP} & - \end{array}\right]\end{array}\right]\right\rangle, \boxed{3}\right)$$

The append principle which must be a member of the theory of the grammar in order to give the relational AVM formula in the lexical entry in (36) the intended meaning looks as follows:

$$\begin{array}{ll}(37)\ \forall x \forall y \forall z\ (\texttt{append}(x,y,z) \leftrightarrow & \\ \qquad (x\,\langle\rangle\ \wedge\ y = z\ \wedge\ {}^{y}[\textit{list}]) & \\ \qquad \vee\ \exists\boxed{1}\,\exists\boxed{2}\,\exists\boxed{3}\ \left(x\,\langle\boxed{1}|\boxed{2}\rangle \wedge z\,\langle\boxed{1}|\boxed{3}\rangle \wedge \texttt{append}(\boxed{2}, y, \boxed{3})\right)) & \end{array}$$

Note that in (37) the existential quantification over the variables that are written as boxed integers cannot be left implicit. The existential closure convention, 4, cannot be evoked to leave out the existential quantifiers, since existential closure would wrongly lead to wide scope of the existentials over the universals. If the AVM description (37) is in the theory of a grammar, Γ, it determines, for each entity, u, in the universe of a model of Γ, which components, x, y, and z, of u must be in the `append` relation. It is exactly those triples of components of u that are either described by the first disjunct (second line of (37)) or the second disjunct (third line of (37)) after the bi-implication symbol.

In some cases, it proves to be advantageous to employ a slightly different notation for relation principles. That notation, again, owes its syntax to Prolog, and appeals to common notational habits of linguists working in HPSG. It breaks up the AVM description of relation principles like (37) into multiple clauses. The number of clauses determining the meaning of a relation depends on the number of disjuncts to the right of the bi-implication symbol in the corresponding AVM formula notation, and each clause is an independent piece of syntax. For example, `append` will be defined in two clauses, one for the second line and one for the third line of the definition of `append` in (37). Because of its design, the clause notation is very convenient when there are many disjuncts that define the meaning of a relation. If a relation principle is written as clauses, it is split up in a modular fashion and its parts can be presented, motivated and discussed separately in the course of linguistic or formal reasoning. This is an important characteristic if a relation has linguistic significance as in a relational formalization of word order rules; whereas, for merely technical relations such as `append`, it is usually not necessary. As we will presently see, a clause notation will also make it easy to leave all the quantifiers of the `append` principle, (37), implicit.

To define conventions for a clause notation, I first need an adaptation to AVM formulae of the standard notion of the simultaneous substitution of variables. In DEFINITION 93, curly brackets are used for disambiguation when necessary and have no other significance.

Definition 93 *For each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, for all pairwise distinct variables $x_1, \ldots x_n, v_1, \ldots, v_n$,*

$$\text{for each } v \in \mathcal{VAR}^{:},\ v\frac{v_1 \ldots v_n}{x_1 \ldots x_n} = \begin{cases} v, & \text{if } v \neq x_1, \ldots, v \neq x_n, \\ v_i, & \text{if } v = x_i, \end{cases}$$

for each $\sigma \in \widehat{\mathcal{G}}$, for each $\alpha_1 \in \widehat{\mathcal{A}}$, …, for each $\alpha_n \in \widehat{\mathcal{A}}$, for each $\beta_1 \in \mathbb{BOX}^{\Sigma}$, …, for each $\beta_n \in \mathbb{BOX}^{\Sigma}$,

$$\left[\begin{array}{ll}\sigma & \\ \alpha_1 & \beta_1 \\ \vdots & \\ \alpha_n & \beta_n\end{array}\right]\frac{v_1 \ldots v_n}{x_1 \ldots x_n} = \left[\begin{array}{ll}\sigma & \\ \alpha_1 & \beta_1 \frac{v_1 \ldots v_n}{x_1 \ldots x_n} \\ \vdots & \\ \alpha_n & \beta_n \frac{v_1 \ldots v_n}{x_1 \ldots x_n}\end{array}\right],$$

for each $\beta_1 \in \mathbb{BOX}^\Sigma$, ..., *for each* $\beta_n \in \mathbb{BOX}^\Sigma$,

$$\langle \beta_1, \ldots, \beta_n \rangle \frac{v_1 \ldots v_n}{x_1 \ldots x_n} = \left\langle \beta_1 \frac{v_1 \ldots v_n}{x_1 \ldots x_n}, \ldots, \beta_n \frac{v_1 \ldots v_n}{x_1 \ldots x_n} \right\rangle,$$

for each $\beta \in \mathbb{BOX}^\Sigma$, $\{\neg\beta\} \frac{v_1 \ldots v_n}{x_1 \ldots x_n} = \neg \left\{\beta \frac{v_1 \ldots v_n}{x_1 \ldots x_n}\right\}$,

for each $\beta_1 \in \mathbb{BOX}^\Sigma$, *for each* $\beta_2 \in \mathbb{BOX}^\Sigma$,

$$[\beta_1 \wedge \beta_2] \frac{v_1 \ldots v_n}{x_1 \ldots x_n} = [\beta_1 \frac{v_1 \ldots v_n}{x_1 \ldots x_n} \wedge \beta_2 \frac{v_1 \ldots v_n}{x_1 \ldots x_n}],$$

for each $\beta_1 \in \mathbb{BOX}^\Sigma$, *for each* $\beta_2 \in \mathbb{BOX}^\Sigma$,

$$[\beta_1 \vee \beta_2] \frac{v_1 \ldots v_n}{x_1 \ldots x_n} = [\beta_1 \frac{v_1 \ldots v_n}{x_1 \ldots x_n} \vee \beta_2 \frac{v_1 \ldots v_n}{x_1 \ldots x_n}],$$

for each $\beta_1 \in \mathbb{BOX}^\Sigma$, *for each* $\beta_2 \in \mathbb{BOX}^\Sigma$,

$$[\beta_1 \rightarrow \beta_2] \frac{v_1 \ldots v_n}{x_1 \ldots x_n} = [\beta_1 \frac{v_1 \ldots v_n}{x_1 \ldots x_n} \rightarrow \beta_2 \frac{v_1 \ldots v_n}{x_1 \ldots x_n}],$$

for each $\beta_1 \in \mathbb{BOX}^\Sigma$, *for each* $\beta_2 \in \mathbb{BOX}^\Sigma$,

$$[\beta_1 \leftrightarrow \beta_2] \frac{v_1 \ldots v_n}{x_1 \ldots x_n} = [\beta_1 \frac{v_1 \ldots v_n}{x_1 \ldots x_n} \leftrightarrow \beta_2 \frac{v_1 \ldots v_n}{x_1 \ldots x_n}],$$

for each $v \in \mathcal{VAR}$, *for each* $\beta_1 \in \mathbb{BOX}^\Sigma$, ..., *for each* $\beta_n \in \mathbb{BOX}^\Sigma$,

$$\{v \,\|\beta_1, \ldots, \beta_n\|\} \frac{v_1 \ldots v_n}{x_1 \ldots x_n} = v \frac{v_1 \ldots v_n}{x_1 \ldots x_n} \; \left\|\beta_1 \frac{v_1 \ldots v_n}{x_1 \ldots x_n}, \ldots, \beta_n \frac{v_1 \ldots v_n}{x_1 \ldots x_n}\right\|,$$

for each $v \in \mathcal{VAR}^{:}$, *for each* $\beta \in \mathbb{UBOX}^\Sigma$,

$$\{v \;\, \beta\} \frac{v_1 \ldots v_n}{x_1 \ldots x_n} = v \frac{v_1 \ldots v_n}{x_1 \ldots x_n} \; \beta \frac{v_1 \ldots v_n}{x_1 \ldots x_n},$$

for each $\rho \in \mathcal{R}$, *for each* $z_1 \in \mathcal{VAR}$, ..., *for each* $z_{\mathcal{AR}(\rho)} \in \mathcal{VAR}$,

$$\rho(z_1, \ldots, z_{\mathcal{AR}(\rho)}) \frac{v_1 \ldots v_n}{x_1 \ldots x_n} = \rho(z_1 \frac{v_1 \ldots v_n}{x_1 \ldots x_n}, \ldots, z_{\mathcal{AR}(\rho)} \frac{v_1 \ldots v_n}{x_1 \ldots x_n}),$$

for each $z_1 \in \mathcal{VAR}^{:}$, *for each* $z_2 \in \mathcal{VAR}^{:}$,

$$\{z_1 = z_2\}\,\frac{v_1 \ldots v_n}{x_1 \ldots x_n} = z_1\frac{v_1 \ldots v_n}{x_1 \ldots x_n} = z_2\frac{v_1 \ldots v_n}{x_1 \ldots x_n},$$

for each $v \in \mathcal{VAR}$, *for each* $\kappa \in \mathbb{AVM}^{\Sigma}$, *for the alphabetically first variable,* z, *which does not occur in* κ *and is distinct from* $v_1, \ldots, v_n$, *for the variables* $x_{i_1}, \ldots x_{i_m}$ *such that* $\{x_{i_1}, \ldots x_{i_m}\} \subseteq \{x_1, \ldots, x_n\}$, $i_1 < \ldots < i_m$, *for each* $x \in \{x_{i_1}, \ldots x_{i_m}\}$, $x \in FV^{AVM}(\exists v\, \kappa)$, *and for no* $x \in \{x_1, \ldots x_n\} \backslash \{x_{i_1}, \ldots x_{i_m}\}$, $x \in FV^{AVM}(\exists v\, \kappa)$,

$$\{\exists v\, \kappa\}\,\frac{v_1 \ldots v_n}{x_1 \ldots x_n} = \begin{cases} \exists v \left\{\kappa\frac{v_{i_1} \ldots v_{i_m}}{x_{i_1} \ldots x_{i_m}}\right\} & \textit{if } v \neq v_{i_1}, \ldots, v \neq v_{i_m}, \\ \exists z \left\{\kappa\frac{v_{i_1} \ldots v_{i_m}, z}{x_{i_1} \ldots x_{i_m}, v}\right\} & \textit{if for some } j \in \{1, \ldots, n\}, \\ & \quad v = v_{i_j}, \end{cases}$$

for each $v \in \mathcal{VAR}$, *for each* $\kappa \in \mathbb{AVM}^{\Sigma}$, *for the alphabetically first variable,* z, *which does not occur in* κ *and is distinct from* $v_1, \ldots, v_n$, *for the variables* $x_{i_1}, \ldots x_{i_m}$ *such that* $\{x_{i_1}, \ldots x_{i_m}\} \subseteq \{x_1, \ldots, x_n\}$, $i_1 < \ldots < i_m$, *for each* $x \in \{x_{i_1}, \ldots x_{i_m}\}$, $x \in FV^{AVM}(\forall v\, \kappa)$, *and for no* $x \in \{x_1, \ldots x_n\} \backslash \{x_{i_1}, \ldots x_{i_m}\}$, $x \in FV^{AVM}(\forall v\, \kappa)$,

$$\{\forall v\, \kappa\}\,\frac{v_1 \ldots v_n}{x_1 \ldots x_n} = \begin{cases} \forall v \left\{\kappa\frac{v_{i_1} \ldots v_{i_m}}{x_{i_1} \ldots x_{i_m}}\right\} & \textit{if } v \neq v_{i_1}, \ldots, v \neq v_{i_m}, \\ \forall z \left\{\kappa\frac{v_{i_1} \ldots v_{i_m}, z}{x_{i_1} \ldots x_{i_m}, v}\right\} & \textit{if for some } j \in \{1, \ldots, n\}, \\ & \quad v = v_{i_j}, \end{cases}$$

for each $\kappa \in \mathbb{AVM}^{\Sigma}$, $\{\neg\kappa\}\,\frac{v_1 \ldots v_n}{x_1 \ldots x_n} = \neg\left\{\kappa\frac{v_1 \ldots v_n}{x_1 \ldots x_n}\right\}$,

for each $\kappa_1 \in \mathbb{AVM}^{\Sigma}$, *for each* $\kappa_2 \in \mathbb{AVM}^{\Sigma}$,

$$(\kappa_1 \wedge \kappa_2)\frac{v_1 \ldots v_n}{x_1 \ldots x_n} = (\kappa_1\frac{v_1 \ldots v_n}{x_1 \ldots x_n} \wedge \kappa_2\frac{v_1 \ldots v_n}{x_1 \ldots x_n}),$$

for each $\kappa_1 \in \mathbb{AVM}^{\Sigma}$, *for each* $\kappa_2 \in \mathbb{AVM}^{\Sigma}$,

$$(\kappa_1 \vee \kappa_2)\frac{v_1 \ldots v_n}{x_1 \ldots x_n} = (\kappa_1\frac{v_1 \ldots v_n}{x_1 \ldots x_n} \vee \kappa_2\frac{v_1 \ldots v_n}{x_1 \ldots x_n}),$$

for each $\kappa_1 \in \mathbb{AVM}^{\Sigma}$, *for each* $\kappa_2 \in \mathbb{AVM}^{\Sigma}$,

$$(\kappa_1 \rightarrow \kappa_2)\frac{v_1 \ldots v_n}{x_1 \ldots x_n} = (\kappa_1\frac{v_1 \ldots v_n}{x_1 \ldots x_n} \rightarrow \kappa_2\frac{v_1 \ldots v_n}{x_1 \ldots x_n}),$$

for each $\kappa_1 \in \mathbb{AVM}^\Sigma$, *for each* $\kappa_2 \in \mathbb{AVM}^\Sigma$,

$$(\kappa_1 \leftrightarrow \kappa_2)\frac{v_1 \ldots v_n}{x_1 \ldots x_n} = (\kappa_1 \frac{v_1 \ldots v_n}{x_1 \ldots x_n} \leftrightarrow \kappa_2 \frac{v_1 \ldots v_n}{x_1 \ldots x_n}).$$

The only interesting case in Definition 93 is quantification, where it must be ensured that no bound variable is replaced and no new variable gets bound by quantification. For that reason, only those x_i are replaced which occur free in κ, and if a variable which is substituted for some x_i equals the variable that is bound by the quantification, the name of the variable that is quantified over is changed.

After these technical preliminaries, I can come back to the definition of the clause notation:

Definition 94 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$,
$\rho\left(x_1, \ldots, x_{\mathcal{AR}(\rho)}\right) \stackrel{\forall}{\Longleftarrow} \phi$ *is a* ***clause*** *iff*

> $\rho \in \mathcal{R}$, $x_1 \in \mathcal{VAR}$, ..., $x_{\mathcal{AR}(\rho)} \in \mathcal{VAR}$, x_1, ..., $x_{\mathcal{AR}(\rho)}$ *are pairwise distinct,* ϕ *is a* Σ *AVM formula, and* $FV^{AVM}(\phi) \subseteq \{x_1, \ldots, x_{\mathcal{AR}(\rho)}\}$.

Clauses employ a new reserved symbol, $\stackrel{\forall}{\Longleftarrow}$. I call it the *clause symbol* and assume that it is not a variable, a sort, an attribute nor a relation symbol. Notice that the variables in the relational Σ AVM formula to the left of the clause symbol must be pairwise distinct. The universal quantifier which is part of the clause symbol stands for the universal quantification over the variables in the relational formula to its left, which is left implicit. The quantification becomes explicit when we say what a set of clauses defining the meaning of a relation, ρ, abbreviates:

Convention 7 *Let*
$C = \{\rho(x_{1_1}, \ldots, x_{1_{\mathcal{AR}(\rho)}}) \stackrel{\forall}{\Longleftarrow} \phi_1, \ldots, \rho(x_{n_1}, \ldots, x_{n_{\mathcal{AR}(\rho)}}) \stackrel{\forall}{\Longleftarrow} \phi_n\}$ *be a finite set of clauses. Suppose that* v_1, ..., $v_{\mathcal{AR}(\rho)}$ *are the alphabetically first pairwise distinct variables which do not occur in any clause in* C. *Then* C *is equivalent to the AVM description*

$$\forall v_1 \ldots \forall v_{\mathcal{AR}(\rho)} \left(\rho(v_1, \ldots, v_{\mathcal{AR}(\rho)}) \leftrightarrow \left(\phi_1 \frac{v_1 \ldots v_{\mathcal{AR}(\rho)}}{x_{1_1} \ldots x_{1_{\mathcal{AR}(\rho)}}} \vee \ldots \vee \phi_n \frac{v_1 \ldots v_{\mathcal{AR}(\rho)}}{x_{n_1} \ldots x_{n_{\mathcal{AR}(\rho)}}}\right)\right).$$

Using clauses and the clause notation convention, 7, the `append` principle, (37), can be reformulated as two clauses:

(38) $\texttt{append}(x, y, z) \stackrel{\forall}{\Longleftarrow}$
$x\,\langle\rangle \ \wedge\ y = z\ \wedge\ {}^{y}[\mathit{list}]$

$\texttt{append}(x, y, z) \stackrel{\forall}{\Longleftarrow}$
$\exists\boxed{1}\ \exists\boxed{2}\ \exists\boxed{3}\ \left(x\,\langle\boxed{1}|\boxed{2}\rangle \wedge z\,\langle\boxed{1}|\boxed{3}\rangle \wedge \texttt{append}(\boxed{2}, y, \boxed{3})\right)$

The clause notation lends itself easily to further abbreviatory conventions. Two expedient moves are to allow multiple occurrences of the same variable name in the relational AVM formula to the left of the clause symbol and to apply a variant of the existential closure convention, 4, to the AVM formula to its right. For this purpose, I loosen the syntactic restrictions on the notation of clauses accordingly and introduce *pseudo-clauses*. A pseudo-clause looks just like a clause, i.e., it is of the form $\rho\left(x_1, \ldots, x_{\mathcal{AR}(\rho)}\right) \stackrel{\forall}{\Longleftarrow} \phi$, but either not all the variables $x_1, \ldots, x_{\mathcal{AR}(\rho)}$ are pairwise distinct, or the set of free variables in the AVM formula ϕ is not a subset of $\{x_1, \ldots, x_{\mathcal{AR}(\rho)}\}$, or both. For pseudo-clauses, I state which actual clauses are meant when variable names occur several times in the relational AVM formula, ϕ_ρ, on the left of the clause symbol, and when the AVM formula on the right contains free variables distinct from the ones that occur in ϕ_ρ:

Convention 8 *Suppose $\phi_1 \stackrel{\forall}{\Longleftarrow} \phi_2$ is a pseudo-clause. First, all free variables in ϕ_2, except for the ones occurring in ϕ_1, are regarded as being existentially bound, with the quantifiers scoping over ϕ_2.*

Second, if not all variables in ϕ_1 are pairwise distinct, then proceed as follows: For each variable, v, in ϕ_1, go through ϕ_1 from left to right and check if v occurs again. If a second occurrence is found, replace it by the alphabetically first variable, v', which is not yet in the clause. Furthermore, conjoin ϕ_2 with the AVM equation $v = v'$. Repeat this procedure until all variables in ϕ_1 are pairwise distinct.

With the pseudo-clause convention, 8, stating the `append` principle, (37), reduces to (39):

(39) $\texttt{append}(x, y, y) \stackrel{\forall}{\Longleftarrow}$
$x\,\langle\rangle \ \wedge\ {}^{y}[\mathit{list}]$

$\texttt{append}(x, y, z) \stackrel{\forall}{\Longleftarrow}$
$x\,\langle\boxed{1}|\boxed{2}\rangle \wedge z\,\langle\boxed{1}|\boxed{3}\rangle \wedge \texttt{append}(\boxed{2}, y, \boxed{3})$

Since multiple occurrences of the same variables in the relational AVM formula to the left of the clause symbol are permitted in pseudo-clauses, the equality between the second and the third argument of `append` in the first clause can now be expressed by using the same variable twice instead of enforcing the equality with an AVM equation as in (38).

In Section 2.2.3, I analyzed a grammatical principle from Abeillé and Godard 1996, which was formulated as an AVM diagram with an implicit `member` relation, and I investigated the possibility of formalizing the principle in SRL using the junk slot technique. For convenience, I repeat Abeillé and Godard's description of the principle in (40). In the end, I concluded that it was not possible to formalize the principle with a *member* junk slot, which is the natural formalization in terms of junk slots that the formulation of the principle suggests. Instead, I had to define a specialized junk slot, *non-aff_r*, that was not related to the `member` relation in any obvious sense. The generality of the underlying formalization idea was lost in favor of a task-specific solution that was dictated by the limitations of the junk slot approach to the encoding of relations.

(40) $\begin{bmatrix} \textsc{vform} & \textit{part-passé} \\ \textsc{s-arg} & \langle \ldots \textit{aff\_r} \ldots \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \textsc{v-aux} & \textit{etre} \end{bmatrix}$

With RSRL I promised a description language that permits a direct and faithful formalization of principles like the one in (40). In addition, the idea of the AVM syntax of RSRL is to provide a formal language that is so close to the common notational conventions of the linguistic literature that at most minimal adjustments to the notation of existing principles are necessary in order to turn them into fully formalized expressions of RSRL. These adjustments remove the ambiguities and vaguenesses that might be present in informal, complex diagrams, thus answering the open questions about their exact meaning.

In fact, I can now minimally modify the above AVM diagram and turn it into an AVM description (under Abeillé and Godard's signature) which, under certain assumptions about the empirical significance of the involved attributes and sorts, means what the authors intended to say.[13] The formalization of the original diagram in terms of RSRL shows what exactly was left

[13] For each entity in French, if the VFORM value is of sort *part-passé*, and there exists a member of the S-ARG list that is of sort *aff_r*, i.e., which is a reflexive clitic, then the V-AUX value must be *etre*.

implicit in the original AVM and how little the syntactic notation must be modified in order to turn the original diagram into a rigorously formalized and unambiguous statement.

$$(41)\quad \left(\begin{array}{l} \exists\boxed{0}\,\exists\boxed{1} \\ \left(\begin{array}{l} \begin{bmatrix} \text{VFORM} & \textit{part-passé} \\ \text{S-ARG} & \boxed{1} \end{bmatrix} \\ \wedge\, \texttt{member}\left(\boxed{0}\big[\textit{aff\_r}\big], \boxed{1}\right) \end{array} \right) \end{array} \right) \rightarrow \big[\text{V-AUX} \;\; \textit{etre}\big]$$

Notice that the existential quantifiers in (41) cannot be left implicit, since for the intended statement, they must take scope in the antecedent of the implication.[14] In other words, the existential quantifiers must be within the scope of the negation in the first disjunct if we rewrite the implication as a disjunction. In the equivalent description in which the quantifiers take wide scope over the entire implication, they would have to be universals, as shown in (42).

$$(42)\quad \forall\boxed{0}\,\forall\boxed{1} \left(\left(\begin{array}{l} \begin{bmatrix} \text{VFORM} & \textit{part-passé} \\ \text{S-ARG} & \boxed{1} \end{bmatrix} \\ \wedge\, \texttt{member}\left(\boxed{0}\big[\textit{aff\_r}\big], \boxed{1}\right) \end{array} \right) \rightarrow \big[\text{V-AUX} \;\; \textit{etre}\big] \right)$$

There are two notational issues that I postpone to later sections, because they are intimately tied to deeper questions about the interpretation of certain constructs and principles in the HPSG literature which deserve careful consideration.

Firstly, chains as arguments of relations are conventionally treated as if they were lists. As we will see, however, some relations between lists are occasionally used in a way which implies that one or more of their arguments cannot be an actual list that is a component of the described entities. Rather, it must be a chain of components of the described entities. The ubiquitous `shuffle` relation of so-called linearization grammars is a typical instance of a relation whose use presupposes such an extended definition, but even `append` is occasionally invoked with chains as arguments. In Section 4.3, I will investigate this issue, and I will present a notational convention that makes it possible to ignore the distinction between chains and lists if the linguist does not want to make it.

[14]The brackets around the antecedent are added to make that clear.

Secondly, I have not addressed the treatment of sets yet. In Section 4.4, I will show that finite sets in HPSG can be encoded in RSRL.[15] The basic idea is to enforce the relevant properties of sets with two principles that restrict the configurations of entities that represent sets, and to employ chains to define the set membership relation in such a general way that it covers all occurrences of sets in Pollard and Sag 1994. The necessary operations on sets such as union, intersection and set difference are defined with the generalized membership relation. A notational convention provides the standard set notation with curly brackets as an abbreviation for a complex expression that involves the membership relation. The usual functional notation for the operations on sets follows from another notational convention that makes the functional notation a notational variant of the corresponding relational AVM formulae.

3.3 Computation

RSRL is designed for the declarative specification of grammars of natural languages, especially for the precise and direct specification of HPSG 94 grammars. In Chapter 4, I will show that all syntactic and semantic extensions that RSRL adds to the logical language of SRL are needed in a faithful formalization of the grammar of English of Pollard and Sag 1994 and in the formalization of other HPSG 94 principles in the literature. I will also argue that the formal language of RSRL suffices to express all the principles of the grammar in the appendix of Pollard and Sag 1994. If the two claims are true then RSRL can indeed be regarded as the formalism that HPSG linguists are implicitly using, and its formal language is capable of specifying natural languages as they are conceived by the practitioners of current theoretical HPSG linguistics.

From its very beginning with Pollard and Sag 1987, HPSG has not only served as a fairly precise formal framework for expressing purely declarative grammars of natural languages, but also as a basis and point of reference for applications of natural language processing, especially for systems that are designed to parse the sentences of a language. In the light of the history and widespread perception of HPSG in the linguistic community, it is thus natural to ask about the computational properties of RSRL, and whether

[15] I am not aware of any proposal in the HPSG literature to employ infinite sets in grammars.

it might be possible to use RSRL for computational purposes as well. Under the assumption that the formal language and the model theory of RSRL make the language of HPSG 94 mathematically explicit, this is ultimately a question about the computational properties of HPSG 94 as it is actually used by theoretical linguists. In this section, I give a brief overview of some fundamental computational results about RSRL. My goal is to sketch the computational properties that we can expect in the general case from HPSG 94 grammars that are not specifically rephrased for implementational purposes. In particular, I will address the satisfiability and the modelability problem, and I will discuss the prediction problem of grammars.

A very basic question that a computational system might be able to answer for us is whether for a given theory, there exists an interpretation that contains a configuration of entities under an entity u that behaves in the way that the theory demands, i.e, some u in the interpretation is described by every description in the theory. This question is formalized as the *satisfaction* problem.

Definition 95 (satisfiability)
*For each signature Σ, for each Σ theory θ, θ is **satisfiable** iff there is a Σ interpretation I such that $\Theta_{\mathsf{I}}(\theta) \neq \emptyset$.*

Kepser (1994) shows that SRL satisfiability is decidable for signatures with a finite set of species, a countable set of attributes, and a recursive appropriateness function, and he provides an effective algorithm that determines if a given set of descriptions is satisfiable. The situation changes when we move from SRL to RSRL:

Theorem 5 (non-compactness of RSRL satisfiability)
RSRL-satisfiability is not compact, i.e., there exists a signature Σ and a Σ theory θ for which every finite subset $\theta' \subset \theta$ is satisfiable, but for which θ itself is not satisfiable.[16]

Proof
Let $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, with

$\mathcal{G} = \{object\}$,

$\sqsubseteq = \emptyset$,

[16]The idea of the following proof is due to Paul King.

$$\mathcal{S} = \{object\},$$
$$\mathcal{A} = \{\alpha\},$$
$$\mathcal{F} = \{\langle\langle object, \alpha\rangle, object\rangle\},$$
$$\mathcal{R} = \emptyset, \text{ and}$$
$$\mathcal{AR} = \emptyset.$$

Let $\theta = S_1 \cup S_2$, *with*

$$S_1 = \{\exists x :\approx x\alpha\}, \text{ and}$$
$$S_2 = \left\{ \neg :\approx : \underbrace{\alpha \ldots \alpha}_{n+1 \text{ times}} \middle| n \in \mathbb{N} \right\}.$$

By virtue of the signature, every component of an entity, u, in a Σ *interpretation can be reached by a sequence of the attribute* α*. Every entity that satisfies* S_1 *can itself be reached from one of its components by* α*. However, because of* S_2*, if an entity satisfies* θ*, no sequence of* α *attributes leads from the entity back to itself. Therefore, there cannot be an entity in any* Σ *interpretation that satisfies* θ*.*

For every finite subset of θ' *of* θ*,* S_2 *is finite. Therefore, for some natural numbers n greater than 0,* $\neg :\approx : \underbrace{\alpha \ldots \alpha}_{n \text{ times}}$ *is not in* S_2*, and each entity, u, in a* Σ *interpretation on which the denotation of n* α *attributes yields u itself satisfies* θ'*, because it satisfies both* S_1 *and* S_2*. Therefore,* θ' *is satisfiable.* ■

The non-compactness result of RSRL satisfiability has the practical consequence that no sound and complete calculus can exist for RSRL satisfiability.

In (R)SRL, unlike in many other interpretable logics, satisfiability and modelability are not the same. Satisfiability is a very weak property that only indicates whether a given set of descriptions is consistent or not. If a Σ theory is satisfiable then there exists at least one entity in some Σ interpretation such that every description in the theory is true of the entity. As the discussion of the intended meaning of grammars in (R)SRL has shown, linguists are interested in the existence of a much more restricted class of interpretations of grammars. Given a grammar $\langle\Sigma, \theta\rangle$, linguists do not only want to know whether the theory of the grammar is consistent; linguists want to know whether there is a non-trivial Σ interpretation I in which every entity is described by every description in the theory θ, i.e., whether there exists a nonempty model of the grammar.[17] This question is formalized as the *modelability* problem.

Definition 96 (modelability)
For each signature Σ, for each Σ theory θ, θ is ***modelable*** *iff there is a Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$ with nonempty universe U such that $\Theta_{\mathsf{I}}(\theta) = \mathsf{U}$.*

If Σ is a signature and θ is a Σ theory, then I call the grammar $\langle \Sigma, \theta \rangle$ *modelable* exactly if θ is modelable. Since satisfiability is undecidable in RSRL, it follows immediately that the stronger modelability problem is undecidable as well. In fact, based on an encoding of a tiling problem, King et al. 1999 proves that modelability is already undecidable in SRL. Since each SRL grammar is an RSRL grammar, the tiling problem of King et al. 1999 can automatically be encoded in RSRL, and we obtain an independent second proof that RSRL modelability is undecidable.

Another typical question that linguists might ask is a question about the *predictions* of a grammar: Given a grammar and a candidate description δ, does the grammar predict δ? In other words, does the grammar predict that δ describes something in the language that the grammar captures? A closer look shows that this question comprises the parsing problem in a constraint-based setting. The parsing problem expresses the question whether a given grammar predicts a given string as grammatical or as ungrammatical. In a constraint-based formalism like RSRL, we can ask that question as a question of prediction. For example, given the RSRL formalization of the grammar of Pollard and Sag 1994, does the grammar predict the description

$$\begin{bmatrix} \textit{sign} \\ \textsc{phon} \ \langle \textit{kim, likes, bagels} \rangle \end{bmatrix} ?$$

If we had a computational system that could answer that question, we would be able to falsify a grammar by letting the system test descriptions of phonological strings that are not in the language to find out if the grammar wrongly predicts a phonological string that is not in the language. If it did, the grammar would overlicense the language, or, in the terminology of generative grammar, it would overgenerate. Of course, in an HPSG 94 grammar, we do not need to restrict our queries to phonological values of signs; we can also ask if a grammar predicts a certain description of the CONTENT value

[17] To be more precise, linguists are interested in an even more restricted class of interpretations, namely in the exhaustive models of a grammar. However, the existence of a non-trivial model of a grammar entails the existence of a non-trivial exhaustive model.

of phrases, or some description of the SUBCAT list of a certain class of verbs, or any other conceivable description. The prediction relation of (R)SRL is defined as follows:

Definition 97 (prediction)
For each signature Σ, for each Σ theory θ, and for each Σ description δ, the grammar $\langle \Sigma, \theta \rangle$ ***predicts*** *δ iff there exists a $\langle \Sigma, \theta \rangle$ model I such that $\{\delta\}$ is satisfiable in I.*

The co-problem of the prediction problem is the non-prediction problem. If we test for non-prediction, we ask if a grammar does *not* predict a description. A computational system for non-prediction could serve to detect whether a grammar underlicenses the language it is supposed to capture. Since it accepts the descriptions that do not describe anything in the language, one could test descriptions of configurations of entities that are expected to be in the language. If the grammar wrongly non-predicts a description, we have found configurations of entities that should be in the language but, according to the grammar, are not. In terms of generative grammar, the grammar undergenerates.

Aldag 1997 is an investigation of the prediction relation and of the non-prediction relation in SRL with the goal to determine whether a finitary, sound and complete calculus for SRL prediction is conceivable. For that purpose, Aldag first establishes correspondences between SRL and standard first-order logic (FOL). From translations of SRL signatures (with finite sets of species), SRL interpretations and SRL descriptions to their FOL analogues, Aldag obtains a translation of triples of SRL signatures, SRL theories and SRL descriptions, such that for each SRL signature Σ, for each SRL theory θ, and for each SRL description δ, $\langle \Sigma, \theta \rangle$ predicts δ iff the translation of $\langle \Sigma, \theta, \delta \rangle$—which is a set of FOL sentences—is FOL satisfiable. Similarly, an SRL theory non-predicts an SRL description exactly if the translation of the signature and the theory infers the negation of the translation of the description in FOL. Given these two correspondences, Aldag uses standard results of the theory of computation and the undecidability result of SRL modelability of King et al. 1999 to show that, (1) SRL prediction is not Turing decidable, (2) SRL non-prediction under signatures with finite sets of species and finite sets of attributes is Turing acceptable, and (3) SRL prediction is not Turing acceptable.[18] Aldag concludes that for some SRL

[18]The terminology of Aldag 1997 is based on Lewis and Papadimitriou 1981, which

signature Σ, there is no sound and complete calculus such that for each Σ theory θ and for each Σ description δ, $\langle\Sigma,\theta\rangle$ predicts δ iff the calculus furnishes a proof that $\langle\Sigma,\theta\rangle$ predicts δ. Aldag thus arrives at his ultimate thesis which says that there is no finitary, sound and complete calculus for SRL prediction under SRL signatures with a finite set of attributes and a finite set of species (Aldag, 1997, p. 40). However, a corresponding calculus for SRL non-prediction is conceivable.

As with the undecidability result of the modelability problem, the undecidability results obtained for SRL persist in RSRL, because every SRL grammar is also an RSRL grammar. It thus follows immediately from Aldag 1997 that the prediction problem as well as the non-prediction problem are undecidable in RSRL. Moreover, there cannot be a sound and complete calculus for RSRL prediction. However, we do not know yet if there is a sound and complete calculus for RSRL non-prediction.

What all of the above results show is that the specification language that HPSG 94 employs in its grammatical theories cannot easily serve as the basis of a general processing system that can mechanically and reliably answer some of the typical questions that linguists ask about grammars. This result has interesting implications for our understanding of HPSG 94 and its architecture.

For the practical work of the computational linguist it means that usually some rephrasal or transformation of linguistic principles will be necessary in order to bring an HPSG 94 grammar into a form that can be processed by a computational system. This rephrasal effort can be felt most acutely in the case of principles that make significant use of component quantification in combination with relations and negation. Typical instances of such principles in Pollard and Sag 1994 are the BINDING THEORY, the CONTROL THEORY, and the TRACE PRINCIPLE.

While it is impossible to build computational systems that solve the satisfiability, modelability or prediction problem of arbitrary RSRL grammars, it might still be possible to design systems for tractable subclasses of grammars that meet certain restrictions, or systems that can solve these problems at least most of the time in interesting cases. For example, Götz 1999 translates grammars written in a language very similar to the language of SRL into con-

defines the notions of decidability and acceptability with Turing machines. (Turing-) acceptability is shown to be equivalent to (Turing-) enumerability (Lewis and Papadimitriou, 1981, p. 280f.).

straint logic programs. The translation preserves the prediction problem for an interesting class of grammars, and the resulting constraint logic programs are, again under certain reasonable conditions, guaranteed to terminate for any given query if the query is not predicted by the grammar. Although the ConTroll grammar development system that is based on Götz's architecture (Götz and Meurers, 1995, 1997a,b) is, in principle, not capable of parsing effectively with any possible input grammar that the specification language permits, it can still be successfully used for many actual HPSG grammars that are theoretically interesting. Of course, processing with general component quantification and with relational formulae that may be in the scope of negation is an even harder problem than processing with a grammar formalism that is only a mild extension of SRL, and the possibilities of a computational application of RSRL remains an open research question.

With regard to human sentence processing, the results above make a successful explanation in terms of RSRL (non-) prediction seem improbable. However, there is no reason to believe that the formal language in which HPSG 94 chooses to express the principles that constrain the possible structures that constitute a natural language is the language of the human processor. Moreover, as Johnson 1988, pp. 94–95, points out, the human parser might use additional structure that natural languages possess outside the realm of the structures that are described in our formal specification language. Independent general processing strategies might guide the human parser and narrow down the parsing problem, and it is not unlikely that these strategies employ "probabilistic techniques, which although not always correct, yield on the average sufficient useful information for human communication in the majority of situations they are used" (Johnson, 1988, p. 95). The use of a variety of extra-logical parsing strategies by the human parser is even more plausible if we consider the fact that empirically, there is no conclusive evidence that the class of strings that is generated by a language is decidable. Quite to the contrary, it is a well-known empirical fact that sometimes humans cannot tell whether or not a given sentence belongs to their native language.

For a theoretical linguist, general computational systems for prediction and non-prediction would facilitate the falsification of grammars due to overlicensing or underlicensing. They would thus be a very welcome feature of a linguistic formalism. However, their existence is by no means a necessary condition for falsifying a grammar that overlicenses or underlicenses. Even if a general computational system does not exist, the (non-) prediction or the satisfiability problem might well be decidable for a particular RSRL grammar in which the linguist is interested. Even if they are not, it is still possible to find counterexamples and prove that they are counterexamples in the traditional way. It might just be impossible to build an algorithm that performs that task automatically for the linguist.

Most importantly, it is necessary to distinguish sharply between mathematically rigorous theories of language with a crisp and clear model theoretic interpretation that allows for an empirical falsification of the theories, and theories of human or computational language processing. A theory that specifies a natural language and a theory of how natural languages are processed are two fundamentally different issues. Nothing in a declarative specification is supposed to directly reflect anything about processing. What is important about the specification is that it is empirically correct and makes falsifiable predictions. A formalism that makes this task easy for linguists by providing a tailor-made, expressive specification language for saying exactly what linguists want to say in exactly the way in which they want to say it is by far superior to a framework with a formal language with minimal expressive power that makes it very difficult or impossible for linguists to express their observations concisely. It is not the formalism that should impose constraints on a linguistic theory. Instead, linguistic theories express constraints on natural languages, and the formalism to which they belong must be sufficiently expressive to support the right constraints.[19] Once such a formalism exists and is used for the specification of increasingly comprehensive descriptions of natural languages, the models of the theories that are expressed in the formalism can be investigated with the model theoretic tools of mathematics, and they give theories of language processing a well-defined target which they can then try to supplement with appropriate computational theories. With HPSG 94 we are in the lucky situation that the formalism that RSRL makes explicit has already been widely used implicitly. Once it is made explicit, we immediately obtain a significant number of models of natural languages that were proposed in the literature and that we can study.

[19]See Pollard 1996 for further arguments supporting this view. Pollard expands on many of the issues surrounding the purpose of constraint-based grammar formalisms which I have only briefly mentioned here. In general, the criteria that a constraint-based grammar formalism should meet according to Pollard 1996 are met by RSRL.

Chapter 4

The Grammar of Pollard & Sag 94

In Section 2.2.3 above, I have argued that the expressive means of the language of SRL are insufficient for an adequate formalization of many linguistic principles that can be found in the HPSG literature. In particular, SRL does not provide genuine relations, and it lacks any means to express the kind of bounded quantification that underlies a significant number of grammatical principles. In response to that criticism, I have presented an extension of SRL, RSRL, which comprises a syntax and semantics of relations and bounded quantification. I have shown that this extension is consistent with the different views of the relationship between a grammar and a natural language which have been discussed in the context of SRL and of the HPSG of Pollard and Sag 1994. Finally, I have provided an alternative syntax for RSRL which to a large extent agrees with the notational conventions of the linguistic literature. If my argument that RSRL is a suitable formal language for the formalization of HPSG 94 grammars is correct, it should now be possible to specify HPSG 94 grammars rigorously in RSRL in a straightforward way.

In this section, I will defend my claim that RSRL is a formalism which provides everything that is necessary to treat HPSG 94 with a formalization of selected principles from the appendix of Pollard and Sag 1994 and some additional examples from elsewhere in the literature. The examples have been chosen to demonstrate that all new syntactic constructions that I have added to SRL are necessary for a faithful formalization of HPSG 94. The stronger claim that RSRL provides everything that is needed to formalize

HPSG 94 grammars is harder to defend. On various occasions Pollard and Sag 1994 itself is vague enough about the underlying formalism it envisions to be open to different interpretations. However, I will argue that all grammatical well-formedness conditions that Pollard and Sag (1994) apparently want to express with additional formal devices can be expressed just as easily and faithfully with the means that we already have. I will discuss examples in which it might be open to debate and taste if the formalization in RSRL comes sufficiently close to the apparent intentions of Pollard and Sag 1994, or if it is justified to add more machinery to the formalism in order to come even closer to what the authors might exactly have had in mind. Three topics that deserve special attention in this context are the treatment of sets, parametric sorts (sometimes called polymorphic types), and the specification of the lexicon. Sets are the topic of Section 4.4, and parametric sorts and lexical specifications will be discussed in Section 4.5.

Although grammatical principles, in particular the principles of English put forth in Pollard and Sag 1994, are the topic of this section, I will ignore all questions concerning their theoretical adequacy and empirical scope. My question here is not which linguistic predictions the principles make nor how they carve up the empirical domain. The question is how the informally given principles can be succinctly formalized. The most important goal is to demonstrate that the relationship between their informal rendering in natural language and their formal rendering in RSRL is completely transparent. Because of this transparency we can be sure that the AVM descriptions capture exactly what the authors of the principles meant to say. That certainty is a prerequisite for any reliable formal reasoning with a grammar, and for an investigation of the common properties of the exhaustive models that different HPSG grammars specify.

Throughout the discussion of the principles of Pollard and Sag 1994 I presuppose the signature of the appendix of the book. For long attribute names and sort symbols I occasionally use the transparent abbreviations that are common in the literature.

All principles of the grammar of Pollard and Sag 1994 cannot be discussed in this section. Once one has seen the formalization of the selected examples, it is, however, a purely mechanical task to treat the remaining principles accordingly. In order to substantiate this claim, a complete formalization of all other principles of the appendix of Pollard and Sag 1994 is provided in Appendix C below.

4.1 Quantification

Most grammatical principles that use quantification do so in combination with the use of relations. When quantification occurs as quantification over the arguments of a relational formula, one might think that it is or should be tied to the interpretation of the relational formula. In extensions of constraint languages with definite relations, quantification is typically part of the semantic interpretation of the relational expressions.[1] In HPSG 94, however, the use of quantification is much more liberal. The CONTROL THEORY is an excellent example of a grammatical principle that does not involve relations at all, but makes use of both universal and existential quantification. It also illustrates the perspective of quantification as quantification over the components of an entity:

(43) *The* CONTROL THEORY *(Pollard and Sag, 1994, p. 401)*[2]

> If the SOA-ARG value of a *control-qfpsoa* is token-identical with the CONTENT value of a *local* entity whose CATEGORY | SUBCAT value is a list of length one, then the member of that list is (1) reflexive, and (2) coindexed with the INFLUENCED (respectively, COMMITTOR, EXPERIENCER) value of the *control-qfpsoa* if the latter is of sort *influence* (respectively, *commitment*, *orientation*).

Several features of the CONTROL THEORY are remarkable: It does not talk about an entity of a particular sort and its components in the way the HEAD FEATURE PRINCIPLE talks about phrases with DTRS value *headed-struc* and the CLAUSAL REL PROHIBITION talks about *synsem* entities and (some of) their components. It talks about entities of three maximally specific subsorts of *control-qfpsoa* and about entities of sort *local* that are components of an entity which it does not specify at all. We may only deduce that they are components of a common ancestor, for example a sentence, because it would not make sense to interpret the CONTROL THEORY as demanding the identities it mentions of *control-qfpsoa* entities and *local* entities that do not at least occur in the same utterance. The CONTROL THEORY does not even specify

[1] For examples of this practice, see Höhfeld and Smolka 1988, Dörre and Dorna 1993, and Götz 1999.

[2] The appendix version of the CONTROL THEORY differs from the one given in the text, Pollard and Sag 1994, p. 302. Tilman Höhle pointed out to me that the two versions make different empirical predictions.

the relative position of the *control-qfpsoa* entity and the *local* entity within their common ancestor. It simply says that if there are an *influence* entity and a *local* entity that have the same SOA-ARG and CONTENT value, then two other conditions must hold. In particular, it does not specify relations that would hold between the two entities within a given ancestor.

As a result, the only way to specify the CONTROL THEORY in exactly the same way in which it is stated by Pollard and Sag (1994) is to demand that, whenever an *influence* entity and a *local* entity with exactly one element on its SUBCAT list that share their SOA-ARG and CONTENT value occur as components of an entity, then conditions (1) and (2) must hold of that *influence* entity and of that *local* entity: (1) the LOCAL CONTENT value of the *synsem* entity on the SUBCAT list of the *local* entity is *reflexive*, and (2) its LOCAL CONTENT INDEX value is identical with the INFLUENCED value of the *influence* entity. The same conditions hold of the corresponding components of *local* entities of the relevant kind and of *commitment* and *orientation* entities and the values of their attributes COMMITTOR and EXPERIENCER, respectively.

(44) *The* CONTROL THEORY *formalized*

$$\forall\boxed{1}\,\forall\boxed{2}\,\forall\boxed{3}\left(\begin{array}{l}\exists\boxed{4}\left(\boxed{1}\begin{bmatrix}\textit{influence} & \\ \textsc{soa-arg} & \boxed{4}\end{bmatrix}\wedge\boxed{2}\begin{bmatrix}\textit{local} & \\ \textsc{cat subcat} & \langle\boxed{3}\rangle\\ \textsc{cont} & \boxed{4}\end{bmatrix}\right)\rightarrow\\ \exists\boxed{5}\left(\boxed{3}\begin{bmatrix}\textsc{loc cont} & \begin{bmatrix}\textit{reflexive} & \\ \textsc{index} & \boxed{5}\end{bmatrix}\end{bmatrix}\wedge\boxed{1}\begin{bmatrix}\textsc{influenced} & \boxed{5}\end{bmatrix}\right)\end{array}\right)\wedge$$

$$\forall\boxed{1}\,\forall\boxed{2}\,\forall\boxed{3}\left(\begin{array}{l}\exists\boxed{4}\left(\boxed{1}\begin{bmatrix}\textit{commitment} & \\ \textsc{soa-arg} & \boxed{4}\end{bmatrix}\wedge\boxed{2}\begin{bmatrix}\textit{local} & \\ \textsc{cat subcat} & \langle\boxed{3}\rangle\\ \textsc{cont} & \boxed{4}\end{bmatrix}\right)\rightarrow\\ \exists\boxed{5}\left(\boxed{3}\begin{bmatrix}\textsc{loc cont} & \begin{bmatrix}\textit{reflexive} & \\ \textsc{index} & \boxed{5}\end{bmatrix}\end{bmatrix}\wedge\boxed{1}\begin{bmatrix}\textsc{committor} & \boxed{5}\end{bmatrix}\right)\end{array}\right)\wedge$$

$$\forall\boxed{1}\,\forall\boxed{2}\,\forall\boxed{3}\left(\begin{array}{l}\exists\boxed{4}\left(\boxed{1}\begin{bmatrix}\textit{orientation} & \\ \textsc{soa-arg} & \boxed{4}\end{bmatrix}\wedge\boxed{2}\begin{bmatrix}\textit{local} & \\ \textsc{cat subcat} & \langle\boxed{3}\rangle\\ \textsc{cont} & \boxed{4}\end{bmatrix}\right)\rightarrow\\ \exists\boxed{5}\left(\boxed{3}\begin{bmatrix}\textsc{loc cont} & \begin{bmatrix}\textit{reflexive} & \\ \textsc{index} & \boxed{5}\end{bmatrix}\end{bmatrix}\wedge\boxed{1}\begin{bmatrix}\textsc{experiencer} & \boxed{5}\end{bmatrix}\right)\end{array}\right)$$

The CONTROL THEORY consists of three analogous conjuncts, one for each of the three relevant subsorts of *control-qfpsoa* and their respective attributes.

Notice also that whereas it shows the necessity of universal quantification over components, it cannot be used as a compelling argument for existential quantification over components. Both instances of existential quantification in each conjunct could be replaced by path equations, if the syntax of the formal language provided general path equations. A clear case of existential quantification over components occurs in the BINDING THEORY, which I will discuss next in the context of relations.

4.2 Relations

It is uncontroversial that relations are needed in a formalization of HPSG 94. Pollard and Sag (1994, p. 21) themselves mention that they employ a number of relation symbols in AVM descriptions, and they suggest that a formalism for HPSG must contain definite relations (Pollard and Sag, 1994, p. 8). As I have argued in my discussion of SRL in Section 2.2.3, definite relations are not sufficient, because HPSG 94 treats relations on a par with other formulae of the description language. Relations occur in the scope of negation, and negation is supposed to be classical. This rules out a formalization of relational expressions as definite relations, because definite relations do not have the required properties and rely on an interpretation of negation as negation by failure or some other substitute.

However, there are examples of relations in Pollard and Sag 1994 where definite relations would suffice. The use of relations in the SUBCATEGORIZATION PRINCIPLE is a particularly simple example.

(45) *The* SUBCATEGORIZATION PRINCIPLE *(Pollard and Sag, 1994, p. 399)*

In a headed phrase, the list value of DAUGHTERS | HEAD-DAUGHTER | SYNSEM | LOCAL | CATEGORY | SUBCAT is the concatenation of the list value of SYNSEM | LOCAL | CATEGORY | SUBCAT with the list consisting of the SYNSEM values (in order) of the elements of the list value of DAUGHTERS | COMPLEMENT-DAUGHTERS.

The SUBCATEGORIZATION PRINCIPLE is a restriction on headed phrases. Headed phrases are entities of sort *phrase* with DTRS value *headed-phrase*. The relation between three lists that the SUBCATEGORIZATION PRINCIPLE demands is only referred to in the consequent of the principle, and the three entities that are in that relation are referred to as particular components of headed phrases. In the terminology of RSRL, they are the values of applying (the functions denoted by) three explicitly given terms that are defined on all phrases to a phrase.[3] It is the SUBCAT list of the head daughter, the list of complement daughters, and the SUBCAT list of the phrase that must stand in a relation which is not named but clearly described in the principle. Recall that SUBCAT lists are lists of *synsem* entities, whereas lists of complement daughters are lists of signs. There are at least two obvious ways to formalize the required relationship between those three lists.

First, one could specify a relation, `extract-ss`, that relates each list of signs in an entity to each list of entities in the entity consisting of the values of the SYNSEM attribute of each sign on the list of signs, where the order of the *synsem* entities corresponds to the order of the signs of which they are a component. For each headed phrase in the grammar of Pollard and Sag 1994, the list of complement daughters and a list consisting of the SYNSEM values of all complement daughters in the same order would be in the relation `extract-ss`. Then one could use the `append` relation specified in (37) to enforce the envisioned relationship by saying that the SUBCAT list of the mother, the list of *synsem* entities extracted from the complement daughters, and the SUBCAT list of the head daughter are in the `append` relation. The advantage of this formalization of the SUBCATEGORIZATION PRINCIPLE would be that it uses an essential relation, `append`, that can be employed in the formalization of many principles. Similarly, a relation for "extracting" a corresponding list of *synsem* entities from a list of signs seems to be a promising candidate for a basic relationship that can potentially be recycled in other principles of grammar. This first possible formulation of the SUBCATEGORIZATION PRINCIPLE is given in Section 3.2.1 on page 215 as an example of the AVM notation of RSRL without additional notational conventions. It also corresponds to the junk slot specification of the SUBCATEGORIZATION PRINCIPLE presented in Section 2.2.3, (19)–(20). The two relations `extract-ss` and `append` are defined in (72) and (37), respectively.

If one pursues the goal of identifying a small set of basic relations that are sufficient to formalize all grammatical principles while avoiding junk attributes and junk sorts, a formalization in terms of `append` and the extraction relation might be the preferred option. Putting considerations of reusing independently needed relations aside, there is a second option which is more

[3] Recall that, on page 163, I said that a term is defined on an entity if the term starts with the colon and the function denoted by the term is defined on the entity.

elegant with respect to the number of relations that are needed to formalize the SUBCATEGORIZATION PRINCIPLE. One could simply define a single relation that relates two lists of *synsem* entities and one list of signs in headed phrases in the appropriate way. This option results in a leaner formalization of the SUBCATEGORIZATION PRINCIPLE itself, because one relation suffices.

(46) $\texttt{sign-ss-append}(x, y, z) \stackrel{\forall}{\Longleftarrow}$
$x \langle\rangle \land y \langle\rangle \land z \langle\rangle$

$\texttt{sign-ss-append}(x, y, z) \stackrel{\forall}{\Longleftarrow}$
$x \langle \boxed{1} | \boxed{2} \rangle \land z \langle \boxed{1} | \boxed{3} \rangle \land \texttt{sign-ss-append}(\boxed{2}, y, \boxed{3})$

$\texttt{sign-ss-append}(x, y, z) \stackrel{\forall}{\Longleftarrow}$
$y \left\langle [\text{SYNSEM } \boxed{1}] \middle| \boxed{2} \right\rangle \land z \langle \boxed{1} | \boxed{3} \rangle \land \texttt{sign-ss-append}(x \langle\rangle, \boxed{2}, \boxed{3})$

I need a relation between three entities such that each triple of components of each entity is in that relation exactly if the last argument, z, is obtained by appending the SYNSEM values of the list of signs in the second argument, y, in the order of the occurrence of the signs on y to the list in the first argument, x. The relation `sign-ss-append` is slightly more general than what is needed for the SUBCATEGORIZATION PRINCIPLE as it does not insist that the lists in its first and in its third argument are lists of *synsem* entities. The first clause of (46) says that for each entity, each triple of components that are empty lists is in the relation. According to the second clause, for each entity, each triple of components x, y, and z is in the relation if the first element of x and z are identical, and `sign-ss-append` holds between the tail of x, y, and the tail of z. Thirdly, if x is the empty list, the SYNSEM value of the first element of y is the first element of z, and x, the tail of y and the tail of z are in `sign-ss-append`, then x, y and z are in the relation. This is the clause that "converts" the list of signs in the second argument into a correspondingly ordered list of *synsem* entities in the third argument. There is no other clause for `sign-ss-append`. With that definition of the relation, (45) can be formalized as in (47):

(47) *The* Subcategorization Principle *formalized*

$$
\begin{array}{l}
\left[\begin{array}{ll} \textit{phrase} & \\ \textsc{dtrs} & \textit{headed-struc} \end{array}\right] \quad \rightarrow \\
\left[\begin{array}{ll}
\textit{phrase} & \\
\textsc{ss loc cat subcat} & \boxed{1} \\
\textsc{dtrs} & \left[\begin{array}{ll}
\textit{headed-struc} & \\
\textsc{head-dtr} & \left[\begin{array}{l} \textit{sign} \\ \textsc{ss loc cat subcat} \ \boxed{3} \end{array}\right] \\
\textsc{comp-dtrs} & \boxed{2}
\end{array}\right]
\end{array}\right] \\
\wedge \ \texttt{sign-ss-append}(\boxed{1}, \boxed{2}, \boxed{3})
\end{array}
$$

The Subcategorization Principle cannot justify the necessity of explicit quantification over components as the correct interpretation of quantification over variables in relational formulae in HPSG 94. All three variables in (47) refer to components of the described entities independent of the kind of quantification, because they are bound to the value of paths that are defined on phrases.

The Binding Theory is formally much more interesting, since it exhibits a complex interaction between relations, existential and universal quantification, and negation. Pollard and Sag (1994) summarize their Binding Theory as follows:

(48) *The* Binding Theory *(Pollard and Sag, 1994, p. 401)*

Principle A. A locally o-commanded anaphor must be locally o-bound.
Principle B. A personal pronoun must be locally o-free.
Principle C. A nonpronoun must be o-free.

Similar to the Control Theory, and in contrast to the Subcategorization Principle, the Binding Theory does not talk about entities of a particular sort and components of them which are specified by terms that are defined on the entities, but about certain components of any kind of entity that stand in certain relations: Whenever there is an anaphor that is locally o-commanded by some other component, then that anaphor is also locally o-bound; for no personal pronoun in the language is there any other entity such that the entity is in the `loc-o-bind` relation with the pronoun, i.e., every pronoun is locally o-free; and wherever a nonpronoun occurs in the language, it is o-free. The componenthood perspective is particularly salient in Principle A. In its antecedent, Principle A talks about locally o-commanded anaphors. Where could that local o-commander be? Clearly, it does not make sense to assume that the o-commander could be anywhere in the (exhaustive model of the grammar that is the) language. Potential o-commanders must be part of the same configuration of linguistic entities. They precede the anaphor on the SUBCAT list of some word that is in the same configuration of entities. The anaphor and its o-commander occur as components of the same entity, whatever kind of entity it may be.

The BINDING THEORY relies on a number of terminological definitions and on several relations, which are all informally specified in Pollard and Sag, 1994, p. 401. Anaphors, personal pronouns, and nonpronouns are *synsem* entities with the LOCAL CONTENT values *ana*, *ppro*, and *npro*, respectively. Recall that all three sorts are maximally specific subsorts of *nom-obj*, which is the supersort of all sorts that denote the typical CONTENT values of nouns. The relations that are needed for the BINDING THEORY, (local) o-command and (local) o-binding, are based on the notion of the relative obliqueness of two entities, which is given as follows:

> "One *synsem* object is *more oblique* than another provided it appears to the right of the other on the SUBCAT list of some word." (Pollard and Sag, 1994, p. 401)

This specification can be directly expressed as a clause that specifies the relation `more-oblique`. In the formalization (49), x is more oblique than y.

(49) $\texttt{more-oblique}(x, y) \stackrel{\forall}{\Longleftarrow}$
$$ {}^{w}\begin{bmatrix} \textit{word} \\ \textsc{ss loc cat subcat } \boxed{1} \end{bmatrix} \land \texttt{to-the-right}(x, y, \boxed{1}) $$

In the pseudo-clause specification of `more-oblique`, the variable w is implicitly existentially bound, with the existential quantifier taking scope over the entire AVM formula to the right of the symbol '$\stackrel{\forall}{\Longleftarrow}$' (see CONVENTION 8). Thus, for every entity, a component x is more oblique than a component y if and only if there exists a word in that entity on whose SUBCAT list y precedes x. The latter condition is enforced by and depends on the auxiliary relation `to-the-right`:

(50) $\texttt{to-the-right}(x, y, z) \overset{\forall}{\Longleftarrow}$
$\exists \boxed{1} \ (z \langle y | \boxed{1} \rangle \wedge \texttt{member}(x, \boxed{1}))$
$\vee \exists \boxed{1} \ (z \langle \textit{object} | \boxed{1} \rangle \wedge \texttt{to-the-right}(x, y, \boxed{1}))$

There are exactly two conditions under which I want to say that component x is to the right of component y on the component list z. Firstly, y is the first element on z and x can be found somewhere in the tail of z (second line of (50)); or, in the recursive case, `to-the-right` holds of x, y, and the tail of z (third line of (50)). It follows that, (a), provided z is finite and non-cyclic, x occurs to the right of y on z, and (b), all components x, y and z of any entity for which that is the case are in the relation `to-the-right`. That in turn guarantees that the relation `more-oblique`, (49), does indeed have the intended meaning, and can be used for the definition of the first relation that is needed in the formalization of Principle A:

> "One referential *synsem* object *locally o-commands* another provided they have distinct LOCAL values and either (1) the second is more oblique than the first, or (2) the second is a member of the SUBCAT list of a *synsem* object that is more oblique than the first."[4] (Pollard and Sag, 1994, p. 401)

Pollard and Sag (1994) call a *synsem* entity *referential* exactly if its LOCAL CONTENT INDEX value is of sort *ref*. According to the sort hierarchy, *ref* is a maximally specific subsort of *index*. The attribute INDEX is defined on entities in the denotation of sort *nom-obj*. Typically, referential *synsem* entities will, therefore, be nouns, i.e., *synsem* entities with LOCAL CATEGORY HEAD value *noun*. In the formalization, x locally o-commands y. Note that the relation `loc-o-command` is not recursively defined.

[4] This definition of local o-command differs from the version given on page 278 of Pollard and Sag 1994, where only the o-commander is required to be referential. According to Carl Pollard (personal communication), the failure to require only the first argument of the local o-command relation referential in the definition of local o-command in the appendix is an error. The same error occurs in the definition in the appendix of o-command, which I quote on page 255 and formalize in (52); it should be compared to the (intended) version as given on page 279 of Pollard and Sag 1994. Since I am not concerned with the correctness of the specifications in the appendix of Pollard and Sag 1994 but with their precise formalization as they are given, I keep the erroneous version. The error can be fixed in the obvious way by removing the requirement that the second argument of the (local) o-command relation be referential from my definitions of the relations.

(51) $\texttt{loc-o-command}(x, y) \overset{\forall}{\Longleftarrow}$

$$\left({}^{x}\left[\text{LOC } \boxed{1} \left[\text{CONT INDEX } \textit{ref}\right]\right] \land {}^{y}\left[\text{LOC } \boxed{2} \left[\text{CONT INDEX } \textit{ref}\right]\right] \land \neg \boxed{1} = \boxed{2}\right) \land$$
$$\left(\begin{array}{l} \texttt{more-oblique}(y, x) \lor \\ \left(\texttt{more-oblique}\left({}^{s}\begin{bmatrix} \textit{synsem} \\ \text{LOC CAT SUBCAT } \boxed{3} \end{bmatrix}, x\right) \land \texttt{member}(y, \boxed{3})\right) \end{array}\right)$$

Again, the formalization follows the informal specification very closely. The second line of (51) ensures that, independent of whether we are in case (1) or in case (2) of the following disjunction, the components x and y are always referential *synsem* entities with distinct LOCAL values. Given this, for every entity, every pair of components x and y with these properties is in the `loc-o-command` relation if and only if x precedes y on the SUBCAT list of some other component which is a word (third line of (51)) or y is the member of some *synsem* component's SUBCAT list, where x precedes that *synsem* component on some word's SUBCAT list. What is remarkable in this definition and what makes it so complex is its repeated appeal to existential quantification over components. Relative obliqueness is defined with reference to some word's SUBCAT list, but it is not specified where in the configuration of entities that word is located. In case (2) of the definition of `loc-o-command`, one more *synsem* entity enters into the definition, whose location is only defined by its obliqueness relative to x, which means that it occurs to the right of x on the SUBCAT list of some word somewhere in the configuration of entities.

From a logical point of view, the notion of o-command generalizes the non-recursive local o-command to a recursive relation. The relation `o-command` keeps the condition that the two entities in the relation are referential *synsem* entities with distinct LOCAL values, and it also keeps case (1). But case (2) is generalized to a recursive definition, and a third case is added which is also recursive:

> "One referential *synsem* object *o-commands* another provided they have distinct LOCAL values and either (1) the second is more oblique than the first, (2) the second is a member of the SUBCAT list of a *synsem* object that is o-commanded by the first, or (3) the second has the same LOCAL | CATEGORY | HEAD value as a *synsem* object that is o-commanded by the first." (Pollard and Sag, 1994, p. 401)

(52) $\texttt{o-command}(x, y) \overset{\forall}{\Longleftarrow}$

$$\left({}^{x}\Big[\text{LOC } \boxed{1} \big[\text{CONT INDEX } \mathit{ref}\big]\Big] \wedge {}^{y}\left[\text{LOC } \boxed{2} \begin{bmatrix}\text{CAT HEAD} & \boxed{4} \\ \text{CONT INDEX} & \mathit{ref}\end{bmatrix}\right] \wedge \neg\boxed{1} = \boxed{2} \right) \wedge$$

$$\left(\begin{array}{l} \texttt{more-oblique}(y, x) \vee \\ \left(\texttt{o-command}\left(x, {}^{s_1}\begin{bmatrix}\mathit{synsem} \\ \text{LOC CAT SUBCAT } \boxed{3}\end{bmatrix} \right) \wedge \texttt{member}(y, \boxed{3}) \right) \vee \\ \texttt{o-command}\left(x, {}^{s_2}\begin{bmatrix}\mathit{synsem} \\ \text{LOC } \big[\text{CAT HEAD } \boxed{4}\big]\end{bmatrix} \right) \end{array} \right)$$

The second line repeats the condition on the components x and y that they be referential *synsem* entities with distinct LOCAL values. The HEAD value of y, $\boxed{4}$, plays a role in the third case, to which I will turn presently. Case (1) in the third line of (52) is the base case of the definition of `o-command` and identical to case (1) of `loc-o-command`: If a component y of the described kind is more oblique than a component x of the described kind, then x o-commands y. Two referential *synsem* components x and y with distinct LOCAL values are also in `o-command` if x o-commands a third *synsem* component, s_1, on whose SUBCAT list we can find y (line 4 of (52)). The last possibility for two x and y of the relevant kind to be in `o-command` is the existence of a *synsem* component s_2 that is o-commanded by x and happens to share its HEAD value with y.

Everything is now in place for the specification of the final two relations that are necessary for the principles of the BINDING THEORY.

> "One referential *synsem* object *(locally) o-binds* another provided it (locally) o-commands and is coindexed with the other. A referential *synsem* object is *(locally) o-free* provided it is not (locally) o-bound. Two *synsem* entities are *coindexed* provided their LOCAL | CONTENT | INDEX values are token-identical." (Pollard and Sag, 1994, p. 401)

The specification of both local o-binding and o-binding is non-recursive. They simply reduce to (local) o-command with an extra condition of coindexation put on their arguments. The fact that the two *synsem* entities in the (local) o-binding relations are referential is automatically guaranteed by their definition as a subset of (local) o-command, because the (local) o-command relation requires that its arguments be referential *synsem* entities.

(53) $\texttt{loc-o-bind}(x, y) \stackrel{\forall}{\Longleftarrow}$
$$\texttt{loc-o-command}\Big({}^{x}\big[\textsc{loc cont index } \boxed{1}\big], {}^{y}\big[\textsc{loc cont index } \boxed{1}\big]\Big)$$

(54) $\texttt{o-bind}(x, y) \stackrel{\forall}{\Longleftarrow}$
$$\texttt{o-command}\Big({}^{x}\big[\textsc{loc cont index } \boxed{1}\big], {}^{y}\big[\textsc{loc cont index } \boxed{1}\big]\Big)$$

Saying that an entity is (locally) o-free is simply a different way of saying that the entity is not (locally) o-bound. Since negation in RSRL is classical, this can easily be expressed with the relations `loc-o-bind` and `o-bind`, and I refrain from introducing extra relations into the grammar to express (local) o-freedom.

With all necessary relations being defined, the principles of the BINDING THEORY can now be formally expressed:

(55) *The* BINDING THEORY *formalized*

Principle A:
$$\forall x\ \Big(\exists y\ \texttt{loc-o-command}\Big(y, {}^{x}\big[\textsc{loc cont } \textit{ana}\big]\Big) \rightarrow \exists z\ \texttt{loc-o-bind}(z, x)\Big)$$

Principle B:
$$\forall x\ \Big({}^{x}\big[\textsc{loc cont } \textit{ppro}\big] \rightarrow \neg\exists y\ \texttt{loc-o-bind}(y, x)\Big)$$

Principle C:
$$\forall x\ \Big({}^{x}\big[\textsc{loc cont } \textit{npro}\big] \rightarrow \neg\exists y\ \texttt{o-bind}(y, x)\Big)$$

(55) relies on the correctness of my mathematical interpretation of Pollard and Sag's (1994) specification of the relations as given in (49)–(54).[5] In

[5] As Carl Pollard pointed out to me, the theory of argument raising might suggest a slightly different formulation of the BINDING THEORY. In argument raising constructions, the same anaphor might occur on the SUBCAT lists of two or more syntactic constituents. The question arises whether an o-commanded anaphor must be o-bound on the SUBCAT list of each word on which it appears, or whether it suffices if it is o-bound on one of the SUBCAT lists. The formalization above follows the latter interpretation. If the former interpretation is desired, `more-oblique` could be defined as follows (as suggested by Carl Pollard):

$$\texttt{more-oblique}(x, y, z) \stackrel{\forall}{\Longleftarrow}$$
$$^{w}\begin{bmatrix} \textit{word} \\ \textsc{ss loc cat subcat } z \end{bmatrix} \wedge \texttt{to-the-right}(x, y, z)$$

mathematical jargon, Principle A can be paraphrased as saying that, for each entity o in the language, for all components x of o that are anaphora, if there is some component y of o that o-commands x, then there is also some component z of o that locally o-binds x. The AVM description that formalizes Principle B says that, for each entity o of the language, each component, x, of o that is a personal pronoun is not locally o-bound by any component, y, of o. Analogously, the AVM description that expresses Principle C can be paraphrased as saying that, for each entity o of the language, each component x of o that is a nonpronoun is not o-bound by any component, y, of o.

Clearly, these somewhat technical paraphrases of (55) are also straightforward paraphrases of the Principles A–C of the BINDING THEORY, quoted in (48) above. If one believes that all parts of my formalization of Pollard and Sag's relations are adequate, one must therefore conclude that (55) is not only a rigorous mathematical specification of their BINDING THEORY, but also maximally transparent and faithful to its original phrasing. Moreover, the formalization of the BINDING THEORY provides a substantial justification of the interpretation of the quantification in HPSG 94 as quantification over components, and it demonstrates the intuitively correct interaction of quantification over components, the definition of the meaning of relations on the basis of quantification over components, and the availability of full classical negation over each kind of formula of the description language, including relational formulae.

4.3 Chains

Up to this point, I have taken my justification of RSRL as the appropriate formalism for HPSG 94 exclusively from Pollard and Sag 1994. In the present section, I will depart from this working strategy, and turn to a paper on

---

According to this definition, `more-oblique` is a set of triples, x, y, and z, where x is more oblique than y relative to the SUBCAT list z. Adding a third argument, z, to the definitions of `loc-o-command` and `loc-o-bind` in the obvious way and without any other modifications, makes the SUBCAT lists available to Principle A:

$\forall x\, \forall v\ \big(\exists y\ \texttt{loc-o-command}\big(y, {}^{x}\big[\text{LOC CONT}\ \ ana\big], v\big) \rightarrow \exists z\ \texttt{loc-o-bind}(z, x, v)\big)$

For each entity o in the language, for all components x of o that are anaphora, for all v, if there is some component y of o that o-commands x on the SUBCAT list of a word, v, then there is also some component z of o that locally o-binds x on v.

It is one of the advantages of formal precision to bring out the differences between alternative formulations like the ones discussed here.

the formalization of certain aspects of linearization grammars, Kathol and Pollard 1995. The reason for that departure is not that an implicit use of chains does not occur in Pollard and Sag 1994. As we will see later, they are implicitly present in several major principles of the grammar, but they always co-occur with sets. Since sets and their formalization are an issue which itself deserves a scrupulous investigation, the discussion will benefit from a clean separation of the issues of chains and sets. For that reason, I take examples from elsewhere in the HPSG literature, and I shall make the argument for chains first, so their status in grammars will have been clarified when I turn to the discussion of sets in Section 4.4. This will help to clearly distinguish the issue of the treatment of sets from the issue of chains, and we will see that the treatment of sets that I will propose is conceptually independent from the use of chains in grammars.

In the present section I will argue that chains must be part of the formalization of some of the principles of Pollard and Sag 1994 and other HPSG literature if one accepts component quantification as the adequate formalization of quantification in HPSG, and if one takes the signature and principles that linguists give when they express their grammatical principles at face value.

4.3.1 An Example

Kathol and Pollard 1995 provides an excellent example of chains in the description of phrases, and in one particular grammatical principle that can be infered from their discussion and has explicitly been put forth in Kathol 1995. The relevance of chains in a faithful formalization of HPSG 94 is further emphasized by the fact that they occur in the context of one of the crucial relations that formalize the description of word order in word order domains of linearization grammars. Linearization grammars are a research topic that has attracted much interest in HPSG in recent years.[6] At their heart lies the idea that the linear ordering in time of utterance events is not explained by positing a precedence relation between the phrase structural components of phrases. A typical instance of the latter approach are classical phrase structure grammars, in which a notion of linear order between the sisters in a tree indirectly induces a temporal ordering of the words that are the yield of the tree. In effect, the linear relationship between constituents

[6] See Reape 1994a, Kathol 1995, Penn 1999c, and Richter and Sailer 2000, among others.

is determined on the basis of hierarchical dominance relations.[7]

From the very beginning, HPSG suggested a different view. The attribute PHON is declared appropriate for every sign.[8] Its value determines the temporal order of the words that are components of the sign. The phonology of the words of which a sign is composed is, therefore, directly represented at each sign in the value of the PHON attribute and need not be traced back to the leaves of a tree.[9] The ordering of words must then be regulated by some sort of CONSTITUENT ORDER PRINCIPLE which must guarantee that the phonology of each phrase consists only of the phonology of its daughters and that the admitted orderings of the elements on the PHON list are empirically correct. Pollard and Sag 1987, p. 169, observes already that a formal explication of the CONSTITUENT ORDER PRINCIPLE may involve a more complex relationship between the phonology of a phrase and the phonology of its daughters than a relationship that only expects the phonology of each phrase to be a subset of the possible permutations of the phonologies of its daughters. Instead, Pollard and Sag 1987 envisages that for certain phenomena such as scrambling, the possibility of interleaving the phonologies of the daughters might be necessary and analytically adequate. In cases where the phonologies of the daughters are interleaved, the phonology of the phrase is not a permutation of the phonologies of its daughters. It was not before the advent of so-called linearization approaches, however, that these suggestions were made concrete in HPSG.[10]

Most discussions of linearization grammars since the work of Reape have involved a list-valued attribute DOM, appropriate for signs, whose elements

[7]Partee et al. 1990, pp. 439–446, presents the standard assumptions in phrase structure grammars about dominance and linear precedence in constituent structure trees. See also Gazdar et al. 1985, pp. 44–50, for a different example of a strictly phrase structural approach to word order.

[8]According to Pollard and Sag 1994, p. 397, the value of PHON is a list of entities of sort *phonstring*, where *phonstring* is an immediate supersort of a set of species which exhaustively enumerates the well-formed phonetic strings of English. Höhle 1999 shows how these simplistic working assumptions can be replaced with a serious architecture of phonology that can be formalized in RSRL.

[9]If certain conditions hold of an HPSG grammar, a tree can be recovered from, (1), the immediate dominance relations that are embodied in the functions in the denotation of the daughters attributes that recursively embed signs into phrases and, (2), an additional definition of precedence relations between the values of all daughters attributes of each sign.

[10]The fundamental ideas of linearization grammars in HPSG were initially pioneered by Mike Reape in Reape 1989, 1990, 1992, 1994a.

mediate between the syntactic or tectogrammatical structure and the phonology of signs. The suggestions about the nature and internal structure of the entities on that list have varied, ranging from Reape's original assumption that they are signs over the largely syntactic domain entities of Kathol 1995 to the domain entities of Penn 1999c, which integrate conditions on constituency stemming from syntax, prosody, and discourse.

Kathol and Pollard 1995 is working with assumptions about domain entities that are largely consistent with Kathol 1995. The goal of the paper is to provide an account of extraposition from complex nominal phrases to the *Nachfeld* in German in terms of a dissociation of domain entities on the DOM list of certain phrases from the syntactic constituents of which these phrases consist. Some constituents can be liberated from nominal phrases, which means that they introduce domain entities which are distinct from the domain entity that comes from the nominal phrase to which they syntactically belong.

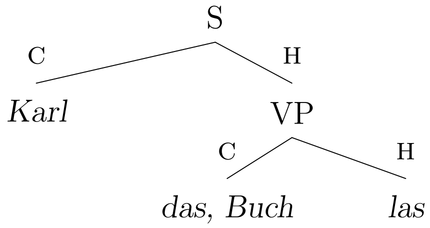
The feature of Kathol and Pollard's analysis in which I am interested here is, however, completely independent of the particular syntactic analysis of extraposition of Kathol and Pollard 1995. What I am concerned with is rather a property of the linearization approach to syntax that occurs independent of the empirical domain to which it is applied. Before we can discuss an example, I must introduce a number of assumptions about the signature in which Kathol and Pollard 1995 differs from Pollard and Sag 1994. As already mentioned above, there is a new attribute, DOM, appropriate to signs, and $\mathcal{F}(sign, \text{DOM}) = list(dom\text{-}obj)$. The notation *list(dom-obj)* is meant to indicate that the list values of DOM lists only contain entities of sort *dom-obj*. For the moment I ignore the question of how so-called parametric sorts of this kind can be treated in RSRL, and simply assume that they can be expressed.[11] Two attributes are defined on entities of the maximally specific sort *dom-obj*, PHON and SYNSEM, and their respective values are the same as their values on signs, a list of *phonstring* and *synsem* entities. In addition, the syntactic head complement structure is binary branching. The list-valued COMP-DTRS attribute is replaced by the attribute COMP-DTR, whose value is a sign. With this signature, the syntactic structure of the German verb first sentences in (56a) and (56b) is uniformly analyzed as sketched in (56c).[12]

[11] See Section 4.5.2 for discussion.

[12] The structure of the corresponding verb second and verb final sentences in Kathol 1995 is also described by (56c) if one maintains the assumption of daughters attributes,

(56) a. Las Karl das Buch?
read Karl the book
'Did Karl read the book?'

b. Las das Buch Karl?
read the book Karl
'Did Karl read the book?'

c. 

By convention, the symbols 'S' and 'VP' at the phrasal nodes of the description (56c) abbreviate descriptions of phrases with HEAD value *verb* and an empty SUBCAT list or a SUBCAT list containing exactly one element, respectively. A line labelled with H is a graphical abbreviation of DTRS HEAD-DTR, and a line labelled with C is a graphical abbreviation of DTRS COMP-DTR. These two paths are supposed to be in the description of the signs at the points where the labelled lines originate. At the end of the labelled lines we see recursively embedded descriptions of signs. At the leaves of the graphical tree stand the *phonstrings* *Karl*, *das*, *Buch*, and *las*, which indicate the PHON values of the respective signs. (56c) is, thus, simply a shorter, more traditional, and in some respects more perspicuous way of writing (57):

(57)
$$\begin{bmatrix} \textit{phrase} & \\ \text{SS LOC CAT} & \begin{bmatrix} \text{HEAD} & \textit{verb} \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \\ \text{DTRS} & \begin{bmatrix} \text{HEAD-DTR} & \begin{bmatrix} \textit{phrase} & \\ \text{SS LOC CAT} & \begin{bmatrix} \text{HEAD} & \textit{verb} \\ \text{SUBCAT} & \langle \textit{object} \rangle \end{bmatrix} \\ \text{DTRS} & \begin{bmatrix} \text{HEAD-DTR PHON} & \langle \textit{las} \rangle \\ \text{COMP-DTR PHON} & \langle \textit{das, Buch} \rangle \end{bmatrix} \end{bmatrix} \\ \text{COMP-DTR PHON} & \langle \textit{Karl} \rangle \end{bmatrix} \end{bmatrix}$$

which is dropped in the later chapters of Kathol 1995.

Obviously, this description says nothing about the phonology of the sentence it describes. The phonology of phrases is determined by the order of the elements on the DOM lists of the phrases. As I have already mentioned above, domain entities have a phonology, and the phonology of a phrase is taken to be the concatenation of the phonologies of the elements on the DOM list in the same order. In a concrete example Kathol and Pollard (1995, p. 276) describe how the DOM list of the VP node of my description (56c) is in their theory related to the DOM list of its daughters. To give their description a direct meaning in RSRL, I have expanded most of their abbreviatory notations in a more explicit rendering of their description:

(58) *Complex domain formation in Kathol and Pollard 1995, p. 176*

$$
\begin{bmatrix}
\textit{phrase} & \\
\text{SS LOC CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{verb} \\ \text{INV} & + \end{bmatrix} \\ \text{SUBCAT} & \langle \text{NP}[\textit{nom}] \rangle \end{bmatrix} \\
\text{DOM} & \boxed{4} \left\langle \begin{bmatrix} \textit{dom-obj} & \\ \text{PHON} & \langle \textit{las} \rangle \\ \text{SS LOC CAT HEAD} & \begin{bmatrix} \textit{verb} \\ \text{INV} & + \end{bmatrix} \end{bmatrix}, \boxed{2} \begin{bmatrix} \textit{dom-obj} & \\ \text{PHON} & \langle \textit{das, Buch} \rangle \\ \text{SS LOC CAT HEAD} & \begin{bmatrix} \textit{noun} \\ \text{CASE} & \textit{acc} \end{bmatrix} \end{bmatrix} \right\rangle \\
\text{DTRS} & \begin{bmatrix} \text{HEAD-DTR} & \begin{bmatrix} \textit{word} & \\ \text{SS LOC CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{verb} \\ \text{INV} & + \end{bmatrix} \\ \text{SUBCAT} & \langle \text{NP}[\textit{nom}],\ \text{NP}[\textit{acc}] \rangle \end{bmatrix} \\ \text{DOM} & \boxed{3} \left\langle \begin{bmatrix} \textit{dom-obj} & \\ \text{PHON} & \langle \textit{las} \rangle \\ \text{SS LOC CAT HEAD} & \begin{bmatrix} \textit{verb} \\ \text{INV} & + \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix} \\ \text{COMP-DTR} \ \boxed{1} & \begin{bmatrix} \textit{phrase} & \\ \text{SS LOC CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{noun} \\ \text{CASE} & \textit{acc} \end{bmatrix} \\ \text{SUBCAT} & \langle \rangle \end{bmatrix} \\ \text{DOM} & \left\langle \begin{bmatrix} \textit{dom-obj} \\ \text{PHON} & \langle \textit{das} \rangle \end{bmatrix}, \begin{bmatrix} \textit{dom-obj} \\ \text{PHON} & \langle \textit{Buch} \rangle \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix}
\end{bmatrix}
$$

$$\wedge\ \texttt{compaction}(\boxed{1}, \boxed{2})$$

$$\wedge\ \texttt{shuffle}(\boxed{0}\ \langle \boxed{2} \rangle, \boxed{3}, \boxed{4})$$

Tag $\boxed{0}$ is not present in the original description. I have introduced it because the syntax of the AVM language requires an explicit variable for the arguments of relations. NP[*nom*] is an abbreviation for a description of a *synsem* entity with LOC CAT HEAD CASE value *nom* and LOC CAT SUBCAT value *elist*. NP[*acc*] is to be understood analogously but with *acc* instead of *nom* case.

The two place relation `compaction` holds between the complement daughter phrase *das Buch*, $\boxed{1}$, and the second domain entity, $\boxed{2}$, on the DOM list of the overall phrase. (59) is a formalization of `compaction` parallel to its informal description in Kathol and Pollard 1995, p. 175.

(59) $\texttt{compaction}(x, y) \overset{\forall}{\Longleftarrow}$
$$ {}^{x}\begin{bmatrix} \textit{sign} & \\ \text{SYNSEM} & \boxed{1} \\ \text{DOM} & \boxed{2} \end{bmatrix} \wedge {}^{y}\begin{bmatrix} \textit{dom-obj} & \\ \text{PHON} & \boxed{3} \\ \text{SYNSEM} & \boxed{1} \end{bmatrix} \wedge \texttt{join-phon}(\boxed{2}, \boxed{3}) $$

The SYNSEM values of the sign and the corresponding compacted domain entity are identical. They have identical syntactic properties. The phonology of the compacted domain entity is the concatenation of the phonology of the domain entities on the DOM list of the sign. This relationship is expressed by the relation `join-phon`.[13]

(60) $\texttt{join-phon}(x, y) \overset{\forall}{\Longleftarrow}$
$$ x\,\langle\rangle \wedge y\,\langle\rangle $$

$\texttt{join-phon}(x, y) \overset{\forall}{\Longleftarrow}$
$$ x\left\langle \begin{bmatrix} \text{PHON} & \boxed{1} \end{bmatrix} \middle| \boxed{2} \right\rangle \wedge \texttt{join-phon}(\boxed{2}, \boxed{3}) \wedge \texttt{append}(\boxed{1}, \boxed{3}, y) $$

The first clause is the base case: In the context of the use of `join-phon` in the definition of `compaction` in (59) it means that an empty list of domain entities is related to an empty phonology list. If the domain list is nonempty (second clause), y is obtained by appending the list of *phonstrings*, $\boxed{3}$, to which

[13] Kathol and Pollard (1995) specify a relation $\texttt{join}_F$ that fulfills the same function as `join-phon` in (60) in a slightly different way, because they want to leave open which attribute's values are concatenated by the relation. While it is possible to formalize precisely that relation, it is much more space-consuming to do so. Since, for present purposes, I only need a relation that relates the phonology values of list members of the first argument to the list-valued second argument, it is much more economical for me to define just that relation.

the tail of the DOM list, $\boxed{2}$, recursively stands in the `join-phon` relation to the PHON value of the first element on the DOM list. For appending lists of *phonstrings*, (60) presupposes the definition of `append` as given in (39) above.

The relation `shuffle` describes the relationship between the DOM list of the head daughter, $\boxed{3}$, a singleton list, $\boxed{0}$, that only contains the domain entity that stands in the `compaction` relation to the complement daughter, and the DOM list, $\boxed{4}$, of the phrase. By convention, `shuffle` holds of a triple of lists if and only if exactly the elements on the first two lists occur on the third, and the relative order of each pair of elements on the first two lists is preserved on the third list; in the present framework, the three lists are components of a common ancestor:

(61)
$$\begin{array}{l} \texttt{shuffle}(x,y,z) \stackrel{\forall}{\Longleftarrow} \\ \quad x\,\langle\rangle \land\ y\,\langle\rangle \land\ z\,\langle\rangle \\[1ex] \texttt{shuffle}(x,y,z) \stackrel{\forall}{\Longleftarrow} \\ \quad x\,\langle\boxed{0}|\,\boxed{1}\rangle \land\ z\,\langle\boxed{0}|\,\boxed{2}\rangle \land\ \texttt{shuffle}(\boxed{1},y,\boxed{2}) \\[1ex] \texttt{shuffle}(x,y,z) \stackrel{\forall}{\Longleftarrow} \\ \quad y\,\langle\boxed{0}|\,\boxed{2}\rangle \land\ z\,\langle\boxed{0}|\,\boxed{3}\rangle \land\ \texttt{shuffle}(x,\boxed{2},\boxed{3}) \end{array}$$

With the given definitions of the meaning of `compaction` and `shuffle`, (58) is a consistent description that denotes the intended set of VPs with the phonology *las das Buch* in an appropriately specified linearization grammar of German. However, this example is only a specific instance of a general property of binary head complement structures that Kathol and Pollard 1995 observes. The ultimate intention of Kathol and Pollard 1995 is to say that for all head complement structures, it is the case that the DOM list of the mother is a shuffling of the DOM list of the head daughter and a singleton list containing the domain entity that stands in the `compaction` relation to the complement daughter. This is in fact made explicit in the BINARY HEAD-ARGUMENT SCHEMA of Kathol 1995:[14,15]

[14] In Kathol 1995, the second argument of `compaction` is a list with one element, and that list is identical with the second argument of `shuffle`. I have changed that detail in order to make the BINARY HEAD-ARGUMENT SCHEMA consistent with the assumptions of Kathol and Pollard 1995. I will briefly return to the difference below.

[15] Strictly speaking, only the consequent of (62) is called the BINARY HEAD-ARGUMENT SCHEMA, just as in the appendix of Pollard and Sag 1994, p. 399 and p. 402, only the disjuncts in the consequent of the ID PRINCIPLE are called ID schemata. Since I want to

(62) BINARY HEAD-ARGUMENT SCHEMA, *after Kathol 1995, p. 148*

$$\left[\text{DTRS}\ \textit{head-comp-struc}\right] \longrightarrow$$

$$\begin{bmatrix} \text{SS LOC CAT SUBCAT} & \boxed{5} \\ \text{DOM} & \boxed{4} \\ \text{DTRS} & \begin{bmatrix} \text{HEAD-DTR} & \begin{bmatrix} \text{DOM} & \boxed{3} \\ \text{SS LOC CAT SUBCAT} & \boxed{7} \end{bmatrix} \\ \text{COMP-DTR} & \boxed{1}\left[\text{SS}\ \boxed{6}\right] \end{bmatrix} \end{bmatrix}$$

$$\wedge\ \texttt{compaction}(\boxed{1}, \boxed{2})\ \wedge\ \texttt{shuffle}(\boxed{0}\langle\boxed{2}\rangle, \boxed{3}, \boxed{4})$$
$$\wedge\ \texttt{append}(\boxed{5}, \boxed{8}\langle\boxed{6}\rangle, \boxed{7})$$

Note that in Kathol 1995, the SUBCATEGORIZATION PRINCIPLE is folded into the BINARY HEAD-ARGUMENT SCHEMA. The last line of (62) states that the SUBCAT list of the head daughter, $\boxed{7}$, is the concatenation of the SUBCAT list of the mother, $\boxed{5}$, with the singleton list, $\boxed{8}$, containing the SYNSEM value of the complement daughter.

It becomes clear from the grammar of Kathol 1995 and the structure of example sentences given therein that (62) envisages the case in which the domain entity $\boxed{2}$ is not the last domain entity on the DOM list of the mother, but instead either intervenes somewhere between the domain entities coming from the head daughter or precedes all of them. Applied to the concrete example of Kathol and Pollard 1995 in (58), that leaves two possibilities. Either, as in the given example itself, *las* precedes *das Buch* (which is motivated by the fact that the INV value is '+', i.e., the verb is inverted or in sentence initial position); or *das Buch* precedes *las*, which could be the case if the verb happened to have the INV value '−'. In (63), the two possibilities are depicted, omitting all nonessential details of the VP:

---

stress the fact that the BINARY HEAD-ARGUMENT SCHEMA imposes its conditions by way of being part of a principle of the grammar and since the principles of grammar are usually implicative descriptions, I take the liberty of adding an appropriate antecedent throughout the discussion of this example. Kathol himself follows the traditional terminology.

(63) a.
$$\left[\begin{array}{l} \textit{phrase} \\ \text{DOM}\ \ \boxed{4}\left\langle [\text{PHON}\ \ \langle \textit{las}\rangle] \,\middle|\, \boxed{0}\left\langle \boxed{2}[\text{PHON}\ \ \langle \textit{das, Buch}\rangle]\right\rangle \right\rangle \\ \text{DTRS}\ \left[\begin{array}{ll} \text{HEAD-DTR} & \left[\begin{array}{l}\textit{word}\\ \text{DOM}\ \ \boxed{3}\left\langle[\text{PHON}\ \ \langle\textit{las}\rangle]\right\rangle\end{array}\right] \\ \text{COMP-DTR}\ \ \boxed{1} & \left[\begin{array}{l}\textit{phrase}\\ \text{DOM}\ \ \left\langle[\text{PHON}\ \ \langle\textit{das}\rangle], [\text{PHON}\ \ \langle\textit{Buch}\rangle]\right\rangle\end{array}\right]\end{array}\right]\end{array}\right]$$
$$\wedge\ \texttt{compaction}(\boxed{1},\boxed{2}) \wedge \texttt{shuffle}(\boxed{0}\,\langle\boxed{2}\rangle,\boxed{3},\boxed{4})$$

b.
$$\left[\begin{array}{l} \textit{phrase} \\ \text{DOM}\ \ \boxed{4}\left\langle \boxed{2}[\text{PHON}\ \ \langle \textit{das, Buch}\rangle] \,\middle|\, \boxed{3}\left\langle [\text{PHON}\ \ \langle \textit{las}\rangle]\right\rangle \right\rangle \\ \text{DTRS}\ \left[\begin{array}{ll} \text{HEAD-DTR} & \left[\begin{array}{l}\textit{word}\\ \text{DOM}\ \ \boxed{3}\left\langle[\text{PHON}\ \ \langle\textit{las}\rangle]\right\rangle\end{array}\right] \\ \text{COMP-DTR}\ \ \boxed{1} & \left[\begin{array}{l}\textit{phrase}\\ \text{DOM}\ \ \left\langle[\text{PHON}\ \ \langle\textit{das}\rangle], [\text{PHON}\ \ \langle\textit{Buch}\rangle]\right\rangle\end{array}\right]\end{array}\right]\end{array}\right]$$
$$\wedge\ \texttt{compaction}(\boxed{1},\boxed{2}) \wedge \texttt{shuffle}(\boxed{0}\,\langle\boxed{2}\rangle,\boxed{3},\boxed{4})$$

In (63), I use CONVENTION 5, the head and tail convention, to emphasize the structure of the DOM list, $\boxed{4}$, of the described phrases. In the first example, (63a), the first element on the list is the domain entity with the phonology *las*, and its tail is the singleton list with the domain entity with the phonology *das Buch* that is also referred to as the first argument of `shuffle`.

Now consider (63b), which differs from (63a) only in the order of the two elements on the DOM list, $\boxed{4}$, of the described phrase: The domain entity *das Buch*, $\boxed{2}$, is its head, and the singleton list, $\boxed{3}$, containing the domain entity *las* is its tail. Notice that the singleton list $\boxed{0}$ is no longer a component of the DOM list $\boxed{4}$, nor does it occur elsewhere in the described phrase. But the semantics of $\texttt{shuffle}(\boxed{0}\langle\boxed{2}\rangle,\boxed{3},\boxed{4})$ requires that its first argument, the list $\boxed{0}$, be a component of the described entity. Since that is not the case, `shuffle` cannot hold of the entity that (63b) is trying to get at, and the phrase with the reversed DOM list does not exist, contrary to the manifest intentions of the authors. To put it slightly differently, (63a) is the only kind of VP with the head daughter *las* and the complement daughter *das Buch* of which the BINARY HEAD-ARGUMENT SCHEMA, (62), holds, because the `shuffle` relation as imposed by (62) only holds for that order of the elements on the DOM list. In particular, it does not hold for the order described in (63b).

I conclude that the definition of `shuffle` that I have given in (61) apparently misses the intuitions that inform the shuffling relation that Pollard and Kathol have in mind. For them it seems to be irrelevant whether a list like $\boxed{0}$ really exists as a component of the described entity. If we pretend it exists, however, it also stands in the `shuffle` relation with the lists $\boxed{3}$ and $\boxed{4}$. Clearly, the virtual list $\boxed{0}$ is a means to express the idea lying at the heart of the analysis, namely that the component list $\boxed{4}$ is structured in a way that could be described as a shuffling of $\boxed{0}$ and the list $\boxed{3}$ if $\boxed{0}$ were a list. To make the meaning of `shuffle` comply with that intuition, it is necessary to extend its meaning in a way that captures that kind of would-be list.

Whereas the virtual list $\boxed{0}$ is not a component of the phrase with the reversed DOM list in (63b), the domain entity, $\boxed{2}$, that stands in the `compaction` relation to the phrase *das Buch*, $\boxed{1}$, is. It follows that, relative to the described phrase, there exists a singleton chain whose only member is $\boxed{2}$. That singleton chain could stand in for the non-existent singleton list $\boxed{0}$ in (63b). In other words, if we allow chains in the first argument of `shuffle` alongside lists, the problem of its too restrictive semantics disappears. Chains of entities are ideal candidates for representing the intended kind of entity, because they are effectively structured like lists and their formalization maintains the basic intuition that quantification in linguistic descriptions is always relative to components of a given entity, but never ranges over unrelated entities in the interpretation. With an appropriately redefined `shuffle` relation—to which I will turn presently in more detail—the BINARY HEAD-ARGUMENT SCHEMA, (62), no longer requires that the compacted complement daughter occur on a singleton list. Remember that it is that requirement that entails that the compacted complement daughter must be the last element on the phrase's DOM list if it is not the last element on some other list that is a component of the phrase.

If one examines the cases in the literature in which problematic lists of the kind illustrated above are used in the arguments of relational formulae, one finds that they always contain entities that are themselves components of the described entities, and they never contain entities that are not components. This suggests that RSRL's generalization of variable assignments and quantification to chains is on the right track. It is sufficient to treat all problematic cases of relations over lists in HPSG 94. The problem with `shuffle` and all similar relations in the HPSG literature is solved very generally by simply defining the relations in question for chains in a parallel manner as they were defined for lists.

In order to make the parallel definition of relations with lists and chains as arguments as convenient as possible in grammar writing, it proves to be advantageous to introduce a notational convention. In practice, the linguist often does not want to think about whether a particular instance of using a relation might involve chains in one or more arguments of that relation, and which argument(s) are which type of entity. To make that kind of underspecification as easy as possible, an abbreviatory notation is needed which can be used to capture all conceivable distributions of lists or chains over a relation's arguments. CONVENTION 9 defines that notation.

Convention 9 *For each signature* Σ*, for each* $v \in \mathcal{VAR}$*, for each* $\beta_1 \in \mathbb{BOX}^{\Sigma}$*, ..., for each* $\beta_n \in \mathbb{BOX}^{\Sigma}$*,*

$$v \ll \beta_1, \ldots, \beta_n \gg = v\,[\langle \beta_1, \ldots, \beta_n \rangle \vee \| \beta_1, \ldots, \beta_n \|].$$

I call each $v \ll \beta_1, \ldots, \beta_n \gg$ a (Σ) tape description. Tape descriptions may involve all other abbreviatory conventions, especially the head and tail convention, CONVENTION 5, for lists and chains. With tape descriptions, principles for relations which involve both lists and chains can easily be written in such a way that they enforce the intended relationship between lists and chains. Consider the meaning of `shuffle` defined with tape descriptions:

(64) $\mathtt{shuffle}(x, y, z) \overset{\forall}{\Longleftarrow}$
$\quad x \ll\gg \wedge\ y \ll\gg \wedge\ z \ll\gg$

$\mathtt{shuffle}(x, y, z) \overset{\forall}{\Longleftarrow}$
$\quad x \ll \boxed{0} | \boxed{1} \gg \wedge\ z \ll \boxed{0} | \boxed{2} \gg \wedge\ \mathtt{shuffle}(\boxed{1}, y, \boxed{2})$

$\mathtt{shuffle}(x, y, z) \overset{\forall}{\Longleftarrow}$
$\quad y \ll \boxed{0} | \boxed{2} \gg \wedge\ z \ll \boxed{0} | \boxed{3} \gg \wedge\ \mathtt{shuffle}(x, \boxed{2}, \boxed{3})$

It is easy to verify that the new definition of `shuffle` entails that `shuffle` holds between each appropriate triple of lists and chains in an exhaustive model of a grammar containing (64). Crucially, that means that the BINARY HEAD-ARGUMENT SCHEMA, (62), can be rewritten with the new `shuffle` relation as in (65):

(65) *The* BINARY HEAD-ARGUMENT SCHEMA *with the revised* `shuffle`

$$\left[\text{DTRS } \textit{head-comp-struc}\right] \longrightarrow$$

$$\begin{bmatrix} \text{SS LOC CAT SUBCAT } \boxed{5} \\ \text{DOM } \boxed{4} \\ \text{DTRS } \begin{bmatrix} \text{HEAD-DTR } \begin{bmatrix}\text{DOM } \boxed{3} \\ \text{SS LOC CAT SUBCAT } \boxed{7}\end{bmatrix} \\ \text{COMP-DTR } \boxed{1}\left[\text{SS } \boxed{6}\right]\end{bmatrix}\end{bmatrix}$$
$$\wedge\ \texttt{compaction}(\boxed{1},\boxed{2})\ \wedge\ \texttt{shuffle}(\boxed{0} \ll \boxed{2} \gg, \boxed{3}, \boxed{4})$$
$$\wedge\ \texttt{append}(\boxed{5}, \boxed{8}\langle\boxed{6}\rangle, \boxed{7})$$

The new BINARY HEAD-ARGUMENT SCHEMA has the originally intended effect on the possible orderings of domain entities on the DOM list of the mother node. Now `shuffle` holds of the VP with the phonology *das Buch las*, because its first argument can be a chain. Thus, the description given in (66), which should be compared to (63b), has a nonempty denotation in exhaustive models of the grammar that Kathol and Pollard 1995 envisages:

(66)
$$\begin{bmatrix} \textit{phrase} \\ \text{DOM } \boxed{4}\left\langle \boxed{2}\left[\text{PHON } \langle \textit{das, Buch}\rangle\right] \| \boxed{3}\left\langle\left[\text{PHON } \langle \textit{las}\rangle\right]\right\rangle\right\rangle \\ \text{DTRS } \begin{bmatrix} \text{HEAD-DTR } \begin{bmatrix} \textit{word} \\ \text{DOM } \boxed{3}\left\langle\left[\text{PHON } \langle \textit{las}\rangle\right]\right\rangle \end{bmatrix} \\ \text{COMP-DTR } \boxed{1} \begin{bmatrix} \textit{phrase} \\ \text{DOM } \left\langle \left[\text{PHON } \langle \textit{das}\rangle\right], \left[\text{PHON } \langle \textit{Buch}\rangle\right]\right\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix}$$
$$\wedge\ \texttt{compaction}(\boxed{1},\boxed{2}) \wedge \texttt{shuffle}(\boxed{0} \|\boxed{2}\|, \boxed{3}, \boxed{4})$$

I conclude that the BINARY HEAD-ARGUMENT SCHEMA as envisioned by Kathol and Pollard 1995 can be maintained and concisely formalized in RSRL, provided one extends the meaning of the `shuffle` relation to include the possibility of chains as arguments.

4.3.2 On the Status of Chains in Grammars

In this section, I want to elaborate on the exact status of my proposal to treat the arguments of some relations simultaneously as lists and as chains. My main objective is to forestall a misunderstanding that might easily arise, namely that the use of chains is a logically inevitable side product of the independently motivated assumption that quantification in HPSG 94 is quantification over components. To see that the use of chains in the above example of domain formation in Kathol and Pollard 1995 is not inevitable, consider the following relation:

(67) $\texttt{blend}(x, y, z) \stackrel{\forall}{\Longleftarrow}$
$\quad z\,\langle x |\, y\rangle$

$\texttt{blend}(x, y, z) \stackrel{\forall}{\Longleftarrow}$
$\quad y\,\langle \boxed{1} | \boxed{2}\rangle \;\wedge\; z\,\langle \boxed{1} | \boxed{3}\rangle \;\wedge\; \texttt{blend}(x, \boxed{2}, \boxed{3})$

`blend` is a three place relation that holds between three components, x, y, and z, of an entity exactly if z is a list whose first element is the entity x and whose tail is the list y, or if the first element of list y is the first element of list z, and `blend` holds of x, the tail of y, and the tail of z. Obviously, `blend` guarantees that z is a list which contains all and only the elements of y in the same order in which they occur on y, except that x has been blended somewhere into that list. With the relation `blend`, we can reformulate the BINARY HEAD-ARGUMENT SCHEMA once again:

(68) *The* BINARY HEAD-ARGUMENT SCHEMA *with the relation* `blend`

$$\left[\textsc{dtrs}\ \textit{head-comp-struc}\right] \longrightarrow$$

$$\begin{bmatrix} \textsc{ss loc cat subcat}\ \boxed{5} \\ \textsc{dom}\ \ \boxed{4} \\ \textsc{dtrs}\ \begin{bmatrix} \textsc{head-dtr}\ \begin{bmatrix} \textsc{dom}\ \ \boxed{3} \\ \textsc{ss loc cat subcat}\ \boxed{7} \end{bmatrix} \\ \textsc{comp-dtr}\ \boxed{1}\left[\textsc{ss}\ \boxed{6}\right] \end{bmatrix} \end{bmatrix}$$

$$\wedge\ \texttt{compaction}(\boxed{1}, \boxed{2})\ \wedge\ \texttt{blend}(\boxed{2}, \boxed{3}, \boxed{4})$$
$$\wedge\ \texttt{append}(\boxed{5}, \boxed{8}\,\langle \boxed{6}\rangle, \boxed{7})$$

In this version of the BINARY HEAD-ARGUMENT SCHEMA, it is no longer claimed that the DOM list of the mother is a shuffling of a tape containing the compacted complement daughter with the DOM list of the head daughter, but it is still said that the DOM list of the mother is the DOM list of the head daughter with the compacted complement daughter somewhere blended in. Of course, that is the same proposition with respect to the possible configurations of the DOM list of the mother. It is just expressed slightly differently with a different relation. But now consider the consequences of this reformulation for the possible orderings of the domain entities in our VP example from Kathol and Pollard 1995. In (69), I repeat (63), but with the relation `blend` substituted for `shuffle` of lists:

(69) a.

$$\left[\begin{array}{ll} \textit{phrase} & \\ \text{DOM} & \boxed{4}\left\langle \left[\text{PHON}\ \langle \textit{las}\rangle\right] \mid \left\langle \boxed{2}\left[\text{PHON}\ \langle \textit{das, Buch}\rangle\right]\right\rangle\right\rangle \\ \text{DTRS} & \left[\begin{array}{ll} \text{HEAD-DTR} & \left[\begin{array}{l} \textit{word} \\ \text{DOM}\ \boxed{3}\left\langle\left[\text{PHON}\ \langle \textit{las}\rangle\right]\right\rangle \end{array}\right] \\ \text{COMP-DTR} & \boxed{1}\left[\begin{array}{l} \textit{phrase} \\ \text{DOM}\ \left\langle\left[\text{PHON}\ \langle \textit{das}\rangle\right], \left[\text{PHON}\ \langle \textit{Buch}\rangle\right]\right\rangle \end{array}\right] \end{array}\right] \end{array}\right]$$
$$\wedge\ \texttt{compaction}(\boxed{1}, \boxed{2}) \wedge \texttt{blend}(\boxed{2}, \boxed{3}, \boxed{4})$$

b.

$$\left[\begin{array}{ll} \textit{phrase} & \\ \text{DOM} & \boxed{4}\left\langle \boxed{2}\left[\text{PHON}\ \langle \textit{das, Buch}\rangle\right] \mid \boxed{3}\left\langle\left[\text{PHON}\ \langle \textit{las}\rangle\right]\right\rangle\right\rangle \\ \text{DTRS} & \left[\begin{array}{ll} \text{HEAD-DTR} & \left[\begin{array}{l} \textit{word} \\ \text{DOM}\ \boxed{3}\left\langle\left[\text{PHON}\ \langle \textit{las}\rangle\right]\right\rangle \end{array}\right] \\ \text{COMP-DTR} & \boxed{1}\left[\begin{array}{l} \textit{phrase} \\ \text{DOM}\ \left\langle\left[\text{PHON}\ \langle \textit{das}\rangle\right], \left[\text{PHON}\ \langle \textit{Buch}\rangle\right]\right\rangle \end{array}\right] \end{array}\right] \end{array}\right]$$
$$\wedge\ \texttt{compaction}(\boxed{1}, \boxed{2}) \wedge \texttt{blend}(\boxed{2}, \boxed{3}, \boxed{4})$$

Since `blend` does not refer to a list containing the domain entity $\boxed{2}$, it holds for both orderings of the domain entities on the DOM list of the mother, $\boxed{4}$, i.e., `blend`($\boxed{2}$,$\boxed{3}$,$\boxed{4}$) holds of some entity u in the exhaustive model of the envisioned grammar, where u is described either by (69a) or by (69b). In other words, (68) is a reformulation of the BINARY HEAD-ARGUMENT SCHEMA which in effect does exactly what Kathol and Pollard 1995 wants to do. It eschews the use of chains and still does not introduce any kind of junk slot into the grammar. Instead, it simply replaces one relation, `shuffle`, by another relation, `blend`. This observation shows that even for a purely relational formulation of the intended principle, it is not necessary to employ chains. However, if one chooses not to use chains, the relation that must hold is no longer a shuffling relation.

Since a reformulation of the BINARY HEAD-ARGUMENT SCHEMA can avoid the use of chains, it might be in order to note that its alternative formulation in Kathol 1995, p. 148, does not do so. To see why, consider the BINARY HEAD-ARGUMENT SCHEMA as it is stated there:

(70) *The* Binary Head-Argument Schema *in Kathol 1995, p. 148*

$$
\begin{bmatrix}\textsc{dtrs} & \textit{head-comp-struc}\end{bmatrix} \longrightarrow
$$

$$
\begin{bmatrix}
\textsc{ss loc cat subcat} \ \boxed{5} \\
\textsc{dom} \ \boxed{4} \\
\textsc{dtrs} \ \begin{bmatrix}
\textsc{head-dtr} \ \begin{bmatrix}\textsc{dom} \ \boxed{3} \\ \textsc{ss loc cat subcat} \ \boxed{7}\end{bmatrix} \\
\textsc{comp-dtr} \ \boxed{1}\begin{bmatrix}\textsc{ss} \ \boxed{6}\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

$$
\wedge\ \texttt{compaction}(\boxed{1}, \boxed{0}\langle\boxed{2}\rangle) \ \wedge\ \texttt{shuffle}(\boxed{3}, \boxed{0}, \boxed{4}) \\
\wedge\ \texttt{append}(\boxed{5}, \boxed{8}\langle\boxed{6}\rangle, \boxed{7})
$$

The difference between the formulation in (70) and my adaptation of it in (62) that makes it compatible with the definitions of the relevant relations of Kathol and Pollard 1995 does not reside in the `shuffle` relation, but in the definition of `compaction`. In Kathol 1995, `compaction` is not a relation between a sign and a domain entity but between a sign and a singleton list that contains a domain entity with the appropriate properties. That singleton list, $\boxed{0}$, is then referred to again as the second argument of `shuffle` with the intention of shuffling it into the DOM list of the head daughter, $\boxed{3}$. As before, that means that that singleton list must exist somewhere in the described entity, and if it does not occur independently anywhere else as a component of the described entity, it must be the tail of the DOM list of the mother. Its single element must then necessarily be the last element of that list, contrary to what is intended. With the formulation of the Binary Head-Argument Schema in Kathol 1995, the necessity for chains in relations arises, thus, not only for `shuffle` but for `compaction` also.

A second example may help to elucidate which kind of possible generalization can become unavailable when one bans the use of chains from the arguments of relations. It also illustrates how easy it is in principle to avoid chains. Furthermore, the example helps to get an idea under which circumstances chains may have to enter into the grammar unnoticed by the grammar writer.

Pollard and Sag 1987, p. 71, formulates a Subcategorization Principle that is very similar to the Subcategorization Principle of Pollard and Sag 1994, which I have cited and formalized in (45) and (47) above. There are only two differences. One is in the signature. Pollard and Sag 1987 assumes that SUBCAT lists are lists of signs, as opposed to the list of *synsem* entities in Pollard and Sag 1994. The second difference concerns the order of the elements on SUBCAT lists. While Pollard and Sag 1987 stipulates that they are ordered with the more oblique complements of the lexical head preceding the less oblique complements, Pollard and Sag 1994 assumes an order where the least oblique complement comes first. For the following thought experiment, I drop the assumptions about the signature that distinguish HPSG 87 from HPSG 94, but I keep the second difference. That is, with Pollard and Sag 1987 I reverse the order of obliqueness of the elements on the SUBCAT list and with Pollard and Sag 1994 I assume that SUBCAT lists are lists of *synsem* entities. For ease of reference, I call the principle to which this leads the P&S:87 SUBCATEGORIZATION PRINCIPLE. Informally, it can be characterized as follows:[16]

(71) *The P&S:87* SUBCATEGORIZATION PRINCIPLE

> In a headed phrase, the list value of DAUGHTERS | HEAD-DAUGHTER | SYNSEM | LOCAL | CATEGORY | SUBCAT is the concatenation of the list consisting of the SYNSEM values (in order) of the elements of the list value of DAUGHTERS | COMPLEMENT-DAUGHTERS with the list value of SYNSEM | LOCAL | CATEGORY | SUBCAT.

Note that whereas in Pollard and Sag 1994 the SUBCAT list of the phrase, the list of SYNSEM values of the complement daughters, and the SUBCAT list of the head daughter stand in the `append` relation (in that order), the first two arguments of the `append` relation have switched their position in (71). The SUBCAT list of the phrase is appended to the list of SYNSEM values of the complement daughter to yield the SUBCAT list of the head daughter. As in the case of the SUBCATEGORIZATION PRINCIPLE of Pollard and Sag 1994, the first idea to formalize (71) might be to use a relation that "extracts" the list of *synsem* entities that are the SYNSEM values of the signs on a list of signs, and then append the SUBCAT list of the phrase to that list.

(72) $\texttt{extract-ss}(x, y) \overset{\forall}{\Longleftarrow}$
$$x\,\langle\rangle \;\wedge\; y\,\langle\rangle$$

$\texttt{extract-ss}(x, y) \overset{\forall}{\Longleftarrow}$
$$x\left\langle \left[\text{SYNSEM } \boxed{1}\right] \middle| \boxed{2} \right\rangle \;\wedge\; y\,\langle \boxed{1} | \boxed{3} \rangle \;\wedge\; \texttt{extract-ss}(\boxed{2}, \boxed{3})$$

[16]This formulation of the SUBCATEGORIZATION PRINCIPLE should be compared to (45).

For each entity, `extract-ss` holds of each pair of component lists in which the first list is a list of signs and the second list is a list of the SYNSEM values of the signs on the first list in the same order. One might then believe that (73) is an appropriate formalization of (71):

(73) *The P&S:87* SUBCATEGORIZATION PRINCIPLE *formalized (first attempt)*

$$\begin{bmatrix} \textit{phrase} \\ \text{DTRS } \textit{headed-struc} \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} \textit{phrase} & \\ \text{SS LOC CAT SUBCAT } \boxed{3} & \\ \text{DTRS} & \begin{bmatrix} \textit{headed-struc} & \\ \text{HEAD-DTR} & \begin{bmatrix} \textit{sign} \\ \text{SS LOC CAT SUBCAT } \boxed{4} \end{bmatrix} \\ \text{COMP-DTRS} & \boxed{1} \end{bmatrix} \end{bmatrix}$$
$$\land\ \texttt{extract-ss}(\boxed{1}, \boxed{2})\ \land\ \texttt{append}(\boxed{2}, \boxed{3}, \boxed{4})$$

However, a closer look reveals that nothing guarantees that the list of *synsem* entities $\boxed{2}$ really exists in the headed phrases that (73) is supposed to license. This is so, because $\boxed{2}$ is not a tail of the SUBCAT list of the head daughter, $\boxed{4}$, and there is no other candidate list of *synsem* entities in the entities supposedly described by the consequent of (73) that could potentially be identical to $\boxed{2}$. Metaphorically speaking, the list $\boxed{2}$ is only an intermediary result which is needed as input to the `append` relation. Note that a parallel formalization with `extract-ss` of the P&S:94 SUBCATEGORIZATION PRINCIPLE would not have that problem,[17] since in that case the extracted list of *synsem* entities is a tail of the SUBCAT list of the head daughter, because it is appended to the SUBCAT list of the phrase rather than the other way around.

If one does not think of the SUBCATEGORIZATION PRINCIPLE as involving a distinct *synsem* extraction that is independent from the appending of the lists, and if one instead views the relationship it expresses in terms of a single relation between the three lists involved, it can be formalized with one relation without chains. I call the relevant relation `ss-sign-append`:

(74) $\texttt{ss-sign-append}(x, y, z) \overset{\forall}{\Longleftarrow}$
$x\,\langle\rangle\ \land\ y\,\langle\rangle\ \land\ z\,\langle\rangle$

[17]See (31).

$$\texttt{ss-sign-append}(x, y, z) \overset{\forall}{\Longleftarrow}$$
$$x \left\langle \left[\text{SYNSEM } \boxed{1}\right] \middle| \boxed{2} \right\rangle \wedge z \left\langle \boxed{1} \middle| \boxed{3} \right\rangle \wedge \texttt{ss-sign-append}(\boxed{2}, y, \boxed{3})$$

$$\texttt{ss-sign-append}(x, y, z) \overset{\forall}{\Longleftarrow}$$
$$x \langle\rangle \wedge y = z \wedge {}^{y}[\textit{list}]$$

ss-sign-append is very similar to the relation sign-ss-append, (46), which I used in the formalization of the P&S:94 SUBCATEGORIZATION PRINCIPLE in (47). The only difference resides in the sort of the entities on the list arguments of the relation. In the case of ss-sign-append, the elements in the first list, x, must be signs, as can be seen in the second clause of (74), because SYNSEM is only defined on words and phrases, and in the first and in the third clause x is the empty list. In correspondence to the definition of sign-ss-append, (74) is more general than what is strictly needed for our purpose, since it does not insist that y and z be lists of *synsem* entities. It does ensure, however, that y and z are always lists, as can be seen in the two base cases of the definition of the meaning of ss-sign-append (first and third clause).

(75) *The P&S:87* SUBCATEGORIZATION PRINCIPLE *formalized (final version)*

$$\begin{bmatrix} \textit{phrase} \\ \text{DTRS } \textit{headed-struc} \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} \textit{phrase} \\ \text{SS LOC CAT SUBCAT } \boxed{2} \\ \text{DTRS} \begin{bmatrix} \textit{headed-struc} \\ \text{HEAD-DTR} \begin{bmatrix} \textit{sign} \\ \text{SS LOC CAT SUBCAT } \boxed{3} \end{bmatrix} \\ \text{COMP-DTRS } \boxed{1} \end{bmatrix} \end{bmatrix}$$
$$\wedge\ \texttt{ss-sign-append}(\boxed{1}, \boxed{2}, \boxed{3})$$

The essential difference between the formalization of (71) with ss-sign-append and the attempt with a combination of extract-ss and append lies in the fact that the former relation avoids an intermediary result that is only an apparent list because it is not a component. Instead, it expresses the relationship between the three components in question directly. On the other hand, we must pay a price for that. ss-sign-append is a more task-specific relation than extract-ss and append are. It is very likely that in a given grammar it can only be used for the SUBCATEGORIZATION PRINCIPLE and for nothing else.

The flexibility that is gained by using chains is illustrated by the fact that with the more general meaning that `extract-ss` and `append` obtain through chains, these relations can formalize both versions of the SUBCATEGORIZATION PRINCIPLE, the version of 94 and the slightly modified version of 87, whereas at least two distinct and much more specialized independent relations are needed to achieve their formalization without chains.[18] Moreover, there are many other places in a grammar where `append` is needed, and it is possible to imagine that `extract-ss` may be of use for other principles as well.

The broader meaning of relations that chains afford in effect allows a more modular approach to relational conditions on configurations of entities. Especially in large grammars, the use of chains may lead to a smaller number of relations, because the same general relation can be used for different purposes. The semantically more liberal definitions also help to capture generalizations that would otherwise be inaccessible formally. For example, the meaning of relations with both list and chain arguments allows to express the intuition that the relationship between DOM lists which we observe in the BINARY HEAD-ARGUMENT SCHEMA is a kind of shuffling which is conceptually related to the shuffling that we observe in other principles concerning linearization domains. In that sense, chains make it possible to bring out generalizations that are otherwise unavailable but might be conceptually important to the linguist.

Whether or not chains should be used in a grammar is a matter of design decisions, and without a convincing catalog of empirically testable criteria it is not *a priori* clear if relations with chains are to be preferred over task-specific relations or *vice versa*. Therefore, I remain agnostic as to which route to choose. It should be stressed, however, that it is not the task of a formalism to prescribe one of two choices which are inherently linguistic in nature and can hopefully ultimately be decided upon on linguistic grounds. Accordingly, the purpose of RSRL is not to codify the minimal necessary apparatus for linguistic analyses that were expressed in HPSG, but to give a mathematical interpretation of existing grammars that explains the meaning and the intuitions behind these grammars as they stand. I am interested in how the actual language that people are implicitly using can be formalized.

[18] For a formalization of `append` with chains and lists as arguments, see `append-t` in (77).

From this point of view I observe that the only way to use `shuffle` in the BINARY HEAD-ARGUMENT SCHEMA that is consistent with the assumptions about quantification and the interpretation of relations and relational negation manifest in HPSG 94 which I have discussed in Sections 4.1 and 4.2 is by evoking chains.

4.3.3 Relations between Chains and Lists

For the formulation of relation principles that use tape descriptions it turns out to be useful to have a single relation which can be used to express the identity of two lists, the identity of two chains, and the identity of a list and a chain in the sense that the list and the chain contain exactly the same members in the same order. I call that two place relation *tape equation* and reserve the symbol '$\doteq$' for it. It is defined in (76), followed by the definition of a generalized append relation, `append-t`, that illustrates its use.

(76)
$$\begin{array}{l}
\doteq(x,y) \stackrel{\forall}{\Longleftarrow} \\
\quad {}^{x}[\mathit{list}] \wedge {}^{y}[\mathit{list}] \wedge x = y \\[1ex]
\doteq(x,y) \stackrel{\forall}{\Longleftarrow} \\
\quad {}^{x}[\mathit{chain}] \wedge {}^{y}[\mathit{chain}] \wedge x = y \\[1ex]
\doteq(x,y) \stackrel{\forall}{\Longleftarrow} \\
\quad (x \,\|\ \| \wedge y \,\langle\rangle) \vee (x \,\langle\rangle \wedge y \,\|\ \|) \\[1ex]
\doteq(x,y) \stackrel{\forall}{\Longleftarrow} \\
\quad \Big(\big({}^{x}[\mathit{list}] \wedge {}^{y}[\mathit{chain}]\big) \vee \big({}^{x}[\mathit{chain}] \wedge {}^{y}[\mathit{list}]\big)\Big) \\
\quad \wedge\ (x \ll\boxed{1}|\boxed{2}\gg \wedge\, y \ll\boxed{1}|\boxed{3}\gg \wedge \doteq(\boxed{2},\boxed{3}))
\end{array}$$

If both arguments of a tape equation are either lists or chains, then tape equation is simply identity.[19] If one is a list and the other is a chain, then

[19] It could be argued that a weaker formulation that says that two lists (or two chains) are considered identical for linguistic purposes if they are not necessarily identical but contain the same elements in the same order is more adequate for tape equations. Since that formulation would, however, permit more pairs of entities in the denotation of tape equations, this can mean that more structures are permitted by a principle of the grammar that uses tape equations. Since linguists normally avoid formulations of principles that admit non-isomorphic configurations of entities that do not correspond to an empirically observable difference—non-isomorphic configurations of this kind are often called *spurious ambiguities*—I choose the more restrictive formulation.

there is an identity of members in the same order. For convenience, I will always use the infix notation for tape equations.

(77) $\texttt{append-t}(x, y, z) \overset{\forall}{\Longleftarrow}$
$x \ll\gg \land y \doteq z$

$\texttt{append-t}(x, y, z) \overset{\forall}{\Longleftarrow}$
$x \ll \boxed{1} | \boxed{3} \gg \land z \ll \boxed{1} | \boxed{4} \gg \land \texttt{append-t}(\boxed{3}, y, \boxed{4})$

(77) defines a generalized version of the `append` relation, (39), which corresponds to the generalized definition of `shuffle`, which I have given in (64) above. Each clause of `append-t` permits each of its three arguments to be either a list or a chain. The tape notation of CONVENTION 9 and the tape equation relation permit a very compact formulation of `append-t`, whose correspondence to the simple `append` of lists is immediately transparent.

4.4 Sets

RSRL, just as SRL, does not provide any direct means for the description of sets. At first, that might appear to be a problem, because grammars written in the framework of HPSG 94 regularly employ so-called set-valued features. The attribute SLASH, which is involved in the description of unbounded dependency constructions, is only the most eminent example. In this section, I will argue that the quantification and the relations of RSRL are sufficient to simulate set descriptions. I will present a slight extension to the signature of Pollard and Sag 1994, the definition of a small number of relations, a SET PRINCIPLE, and an EXTENSIONALITY SIMULATION PRINCIPLE which, together with a meta-assumption about exhaustive models of a grammar, yield an approximation to set values that satisfies the needs of HPSG 94. The common, informal AVM notation for sets is then taken to be a functional notation for some of the relations defined before. As an illustration of this "syntactic sugar method" of describing sets, I will formalize the NONLOCAL FEATURE PRINCIPLE and clause (a) of the SEMANTICS PRINCIPLE, both of which talk about relationships between set values.

4.4.1 A Problem and a Solution

When Pollard and Sag 1994, p. 19, explains sets in HPSG 94, it gives a semantic characterization in terms of nodes that model sets, and with the word "node" it refers to nodes in (some kind of) feature structures. The terminology of the single sentence that describes the intention of the authors, the time of publication, and the identity of the first author suggest that these remarks were written with the set-valued feature structures of Pollard and Moshier 1990 and Moshier and Pollard 1994 in mind. However, these two papers build on background assumptions that differ from those of RSRL. First, they are committed to feature structure models of grammars. As we have seen in Sections 2.2.2.2–2.2.2.4, this is only one possible interpretation of the more general models of RSRL grammars. Second, and even more important, their feature structures can be partial models of linguistic entities, which is reminiscent of the view of HPSG 87, which takes the task of a grammar to be a specification of a feature structure algebra whose feature structures represent partial information about linguistic entities (see Section 2.1.1). Consequently, the notions of unification and subsumption play a central role in Pollard and Moshier 1990 and Moshier and Pollard 1994. Finally, aspects of computational feasibility with which I am not concerned here are implicitly present as a guiding idea of these papers. In summary, most of the issues discussed there are orthogonal or unrelated to questions that are relevant in the present discussion of HPSG 94.

Interestingly, a second kind of entity that is ubiquitous in grammars receives a formalization in HPSG 94 which can be regarded as an example of a generally accepted indirect treatment. Lists are not modelled as lists. Instead, grammarians are content to represent them as complex configurations of entities in the denotation of grammars, just as other entities of linguistic interest. Empty lists are entities of species *elist*, and nonempty lists are entities of species *nelist*, on which two attributes are defined. FIRST names a function that, when interpreted on *nelist* entities, returns an entity which is conventionally understood to be the first element on the list; and REST returns an entity of a subsort of *list* which is taken to be the tail of the list. This state of affairs is expressed by the following section of the sort hierarchy:

(78) *The sort hierarchy of list*

list			
	elist		
	nelist	FIRST	*object*
		REST	*list*

The sort *list* is a direct subsort of *object*. It is probably futile to speculate why lists receive this indirect treatment in HPSG 94. Whatever the ulterior motive might have been originally, one of its effects is a leaner formalism, because lists are not epistemologically distinguished from other entities. If one is willing to accept a uniform account of linguistic entities as a welcome consequence of the indirect treatment of lists, then one should also prefer a treatment of sets that does not enrich the formalism with distinguished entities but treats all entities alike.

In the eyes of Pollard and Sag (1994), there are at least two reasons to employ sets rather than lists for the description of certain phenomena, although one might think that the recursive nature of lists satisfies their descriptive needs. First, each element of the envisaged entities occurs at most once, as opposed to the elements of lists, which can potentially occur twice in different positions. Second, the elements in question are not ordered with respect to each other, as they are on lists. On the other hand, phenomena such as unbounded dependency constructions may involve an arbitrary number of moved elements, which motivates the use of a representation that does not presuppose a fixed number of entities. However, we observe that in each actual example of sets in Pollard and Sag 1994, they have only finitely many members.

Pollard and Sag 1994, p. 19, says that the symbols *eset* and *neset*, which it uses as if they were sort symbols, are in fact no sort symbols.[20] Since I want to approximate sets as used in HPSG 94 in a formalism without special descriptive means for sets, I must use the devices which are given for the rest of the grammar. Therefore, I treat the symbols for sets on a par with other sort symbols, and I define two attributes as being appropriate for *neset*:

(79) *The sort hierarchy of set*

set			
	eset		
	neset	ELE	*object*
		MORE	*set*

The set hierarchy (79) looks essentially like the list hierarchy in (78), except that the attributes that are appropriate for *neset* are called ELE and MORE

[20] In the appendix of the book they are listed in the sort hierarchy (Pollard and Sag, 1994, p. 396).

instead of FIRST and REST. The different symbols are meant to recall the intuition that *set* entities are not understood as inducing an order on the elements of the sets for which the *set* entities stand. But the crucial difference between *list* entities and *set* entities is how their elements are conventionally addressed. Whereas with lists, knowledge of the relative position of their elements is sometimes relevant, I will talk about *set* entities only via a very restricted, small set of access relations. The access relations do neither fix a specific order of elements nor do they make use of any particular configuration of the members of the sets. This should be contrasted with relations over lists such as `to-the-right`, (50), in the BINDING THEORY, which exploit the conventional precedence structure of lists.

Sets are defined by their members. Thus, the crucial relation that I will need to talk about entities of sort *set* is a membership relation, `member`. In fact, all other relations on sets that are needed in grammatical principles such as `union`, `intersection` and equality of sets will be defined on the basis of `member` alone. There are two minor complications with an appropriate definition of membership.

First, in HPSG the notion of membership is conventionally used for both lists and sets indiscriminately. In a formalization, either one distinguishes a relation symbol for list membership from a separate symbol for set membership, or one conflates both kinds of membership in a single relation. In my definition of the meaning of `member` I take the second route, because the signature normally prescribes whether the value of a path in a description with a nonempty denotation is a list or a set, and no harm follows from allowing both lists and sets in the second argument of the `member` relation. Moreover, if necessary one can still explicitly demand in descriptions that only sets or lists be allowed in the second argument of `member` in a particular principle.

Second, as argued in Section 4.3, linguists sometimes use relations in a way which presupposes that lists and chains are interchangable. It is, therefore, a reasonable strategy to define relations which have lists as their arguments simultaneously for chain arguments also, and to extend the meaning of `member` to the membership of entities in chains.

An additional motivation for a broader definition of the meaning of `member` comes from the use of sets themselves. As we will see below in the discussion of the NONLOCAL FEATURE PRINCIPLE, (95) and (96), and clause (a) of the SEMANTICS PRINCIPLE, (99) and (102), some principles that concern set values are formulated in such a way that in the arguments of relations they refer to sets that are not a component of all of the entities these principles are intended to describe; but all elements of the sets in question are components of these entities. As I have already mentioned, all set relations are defined on the basis of `member`. It is an obvious solution to the problem at hand to use chains for the treatment of the virtual sets in the relevant principles, just as we have used chains for virtual lists in relations such as `shuffle`, (64), above. Since all set relations are based on `member`, this effect will be achieved automatically and without further assumptions with a definition of `member` that includes chains. Thus, I define the relation `member` simultaneously for chains, lists, and sets:

(80) $\texttt{member}(x, y) \stackrel{\forall}{\Longleftarrow}$
$$y \, \| x | \, chain \| \;\vee\; y \, \langle x | \, list \rangle \;\vee\; {}^{y}\!\begin{bmatrix} set \\ \text{ELE} \;\; x \end{bmatrix}$$

$\texttt{member}(x, y) \stackrel{\forall}{\Longleftarrow}$
$$(y \, \| \, object | \, z \| \;\wedge\; \texttt{member}(x, z)) \;\vee$$
$$(y \, \langle \, object | \, z \rangle \;\wedge\; \texttt{member}(x, z)) \;\vee$$
$$\left({}^{y}\!\begin{bmatrix} set \\ \text{MORE} \;\; z \end{bmatrix} \wedge \texttt{member}(x, z) \right)$$

According to (80), each tuple of entities, x and y, is in the `member` relation relative to some entity u if and only if y is is a component set or component list of u, and x can be reached from y by the interpretation of a path consisting of a finite string of MORE or REST attributes followd by one ELE or one FIRST attribute, respectively;[21] or y is a chain of components of u, and x is a value of a function named by a finite sequence of the quasi-attribute $\triangleright$ followed by one †. This definition follows the intuitive sense in which list configurations of entities, set configurations of entities and chains of entities are understood. The description '*object*' for the first element of lists and chains in the recursive clause of (80) has been chosen specifically for the grammar of Pollard and Sag 1994, because in that grammar *object* is the top element of the sort hierarchy. For arbitrary grammars with a top element with a different name or no top element of the sort hierarchy at all, `member` obtains the same meaning as in (80) if *object* is replaced by *metatop*.

Before I define a number of relations that will serve to talk about sets in linguistic principles, it is necessary to think about the configurations of entities that are used to approximate sets. Clearly, these configurations of

[21] I ignore the special case of infinite sets and infinite lists. This is justified here, since infinity will be excluded in the SET PRINCIPLE.

entities are not sets themselves; they are merely meant to specify sets. It must then be ensured that they behave in a way that is appropriate to their task.[22] The basic idea is that entities of sort *set* behave like a simple notation with which one can unambiguously specify sets of entities that occur in utterances. That means that each possible configuration of entities under a *set* entity is one that does specify a set. Moreover, it is reasonable to assume that exactly those entities are the members of a nonempty set specified by an *neset* entity that can be reached from the *neset* entity by all paths consisting of a finite string of MORE attributes and one ELE attribute whose interpretation is defined on it.

We can now introduce a few conditions on our specification language for sets, i.e., on the configurations of components of *set* entities, that assimilate the specification language to what its configurations of entities are specifying. I collect these requirements in the definition of a one place relation which I call `set-properties`. In the SET PRINCIPLE, (84), I will require that `set-properties` holds of each *set* entity in the models of the grammar. I consider three properties relevant for *set* entities. First, they are non-cyclic. Non-cyclicity means that there is no nonempty sequence of MORE attributes whose interpretation on a *neset* entity, u, has u as its value. I exclude that kind of cycle because it does not correspond to any property of sets. Second, only finitely many elements stand in the `member` relation with *set* entities. That requirement derives from the fact that, given the definition of `member` in (80), any element u stands in the `member` relation to a *neset* entity with an infinite embedding of *neset* entities relative to a common ancestor if the entity u is a component of every *neset* entity in the configuration. But I want to be able to control the behavior of sets properly with the `member` relation. Undesired effects that would be caused by infinity are therefore excluded by excluding infinity. Finally, each element of a set occurs only once in the set specification. These considerations lead to the following definition of the meaning of `set-properties`:

(81) $\texttt{set-properties}(x) \overset{\forall}{\Longleftarrow}$
$\qquad \texttt{nc-finite}(x) \land \texttt{no-copies}(x)$

[22]Throughout the discussion of the function of *set* entities, it might be helpful to keep in mind that the same considerations apply, *mutatis mutandis*, to HPSG's well established *list* entities. But list simulation is much simpler and can be achieved with the signature alone.

The first defining relation of `set-properties` in the clause (81), `nc-finite`, excludes cyclicity and infinity; `no-copies` excludes repetitions of member entities in *set* entities. The definition of `nc-finite` is straightforward. For reasons of generality, I define it on chains, lists, and sets.

(82) $\texttt{nc-finite}(x) \overset{\forall}{\Longleftarrow}$
$${}^{y}\big[\mathit{chain}\big] \land \left({}^{x}\big[\mathit{list}\big] \lor {}^{x}\big[\mathit{chain}\big]\right) \land x \doteq y$$

$\texttt{nc-finite}(x) \overset{\forall}{\Longleftarrow}$
$${}^{x}\big[\mathit{set}\big] \land \texttt{set-eq-chain}(x, y)$$

$\texttt{set-eq-chain}(x, y) \overset{\forall}{\Longleftarrow}$
$${}^{x}\big[\mathit{eset}\big] \land {}^{y}\big[\mathit{echain}\big]$$

$\texttt{set-eq-chain}(x, y) \overset{\forall}{\Longleftarrow}$
$${}^{x}\begin{bmatrix}\mathit{neset} & \\ \text{ELE} & \boxed{1} \\ \text{MORE} & \boxed{2}\end{bmatrix} \land y \left\|\boxed{1}\middle|\boxed{2}\right\|$$

The clause that defines the meaning of `nc-finite` exploits the fact that chains are finite and non-cyclic by definition. Note that `nc-finite` is defined on chains, on lists and on sets. If x is a chain or a list, I can use the previously defined tape equation, (76): If x is already a chain, the tape equation simply says that x is identical with some chain denoted by y. If x is a list, the tape equation says that its elements occur in the same order on the (finite) chain y. If x is a set, the second clause of the definition of `nc-finite` says that there is a y that stands in the `set-eq-chain` relation with set x. That relation means essentially the same for *set* entities and chains as a tape equation means for lists and chains. It takes the configuration of members of the set as given by the interpretation of sequences of MORE attributes followed by one ELE attribute and requires that there be a chain whose configuration of elements in terms of the interpretation of the quasi-attributes $\triangleright$ and $\dagger$ corresponds to the configuration of the set. The set configuration must, thus, be finite and non-cyclic in the relevant respect. In summary, for each entity in the universe, each list component, each set component, or each chain of components is in the `nc-finite` relation if and only if it is finite and non-cyclic.[23]

[23]Of course, for chains that is true by definition. However, a definition of `nc-finite` that includes chains is convenient, because it allows a well-defined use of the relation in principles together with variables that can and should be assigned both components and chains of components. With the given definition one then avoids awkward case distinctions.

The one place relation that forbids repetitions of elements, `no-copies`, is also defined on chains, on lists, and on sets:

(83) $\texttt{no-copies}(x) \overset{\forall}{\Longleftarrow}$
$x \ll\gg \lor {}^{x}[\textit{eset}]$

$\texttt{no-copies}(x) \overset{\forall}{\Longleftarrow}$
$$\left(x \ll y|\, z\gg \lor {}^{x}\begin{bmatrix} \textit{set} & \\ \text{ELE} & y \\ \text{MORE} & z \end{bmatrix}\right) \land \neg\texttt{member}(y, z) \land \texttt{no-copies}(z)$$

Empty chains, lists, and sets trivially do not have repetitive occurrences of elements (first clause). The second clause forbids repetitions by saying that the first element of the chain or list, y, is not a member of the tail, z, of the chain or list. Moreover, the recursion makes sure that the tail is also a list or chain in which repetitions do not occur. For each *neset* entity, u, the same effect is achieved by recursively excluding that its ELE value is an ELE value in the *set* entity that is u's MORE value, and recursively demanding that property for u's MORE value.

With the relations in (82) and (83) defined, I have spelled out the meaning of `set-properties`, (81), and I have a relation with which I can enforce the three properties under discussion for lists, sets and chains. In particular, I demand that each *set* entity in the exhaustive models of my formalization of the grammar of Pollard and Sag 1994 have those three properties:

(84) *The* SET PRINCIPLE

Each *set* entity is non-cyclic and has finitely many members, and no member of a set occurs twice in the set:

$\textit{set} \rightarrow \texttt{set-properties}\big([\textit{set}]\big)$

Since my description language is not equiped with quantification over the entire universe of entities, and since in the general case, it is not the case that all entities of the universe are components of one special entity over which I could quantify to mimic standard first order quantification, I cannot enforce that all *neset* entities that specify the same set be identical. In other words, extensionality of sets cannot be enforced with the description language. However, it is important to keep in mind that the goal is not to introduce sets themselves into the exhaustive models of grammars, but only to introduce well-behaved set specifications. Which properties I need to enforce for them is a matter of achieving a workable approximation to the behavior of sets.

Instead of the unavailable requirement that all *set* entities in the universe that have identical members be identical, I impose the weaker condition that all *set* entities are identical that are components of a common ancestor and have the same members. It follows that, whatever the entities that are subject to linguistic description are, all *set* entities with identical members within those entities are identical. For example, if utterances are the relevant entities in the grammar, then all sets with identical members that occur in one and the same utterance are guaranteed to be identical by the EXTENSIONALITY SIMULATION PRINCIPLE in (86).

To be able to express that principle, I need an additional relation that expresses that two sets have the same members. That relation, which I write as $\stackrel{s}{=}$, will also be independently needed in grammatical principles.[24] As with the other relations on sets that were discussed so far, it is convenient to keep the new relation general and define it for chains, lists, and sets:

(85) $\stackrel{s}{=}(x, y) \stackrel{\forall}{\Longleftarrow}$
$$\forall a\, (\mathtt{member}(a, x) \leftrightarrow \mathtt{member}(a, y)) \wedge$$
$$^{x}[\mathit{chain} \vee \mathit{list} \vee \mathit{set}] \wedge {}^{y}[\mathit{chain} \vee \mathit{list} \vee \mathit{set}]$$

For convenience, I will henceforth use the infix notation for $\stackrel{s}{=}$. The relation $\stackrel{s}{=}$ holds of pairs, x and y, of chains of components, component lists and component sets, in each possible combination, that have identical members. The third line of (85) restricts x and y to chains, lists, or sets, because, without that restriction, arbitrary pairs of components of other sorts would be in the denotation of the relation. With (85), the EXTENSIONALITY SIMULATION PRINCIPLE can be formulated:

(86) EXTENSIONALITY SIMULATION PRINCIPLE

For each entity u, if two sets x and y with identical members are components of u, then x and y are identical.

$$\forall x \forall y \left({}^{x}[\mathit{set}] \stackrel{s}{=} {}^{y}[\mathit{set}] \rightarrow x = y \right)$$

[24] For an example, see the formalization of clause (a) of the SEMANTICS PRINCIPLE, (102), below.

By the SET PRINCIPLE and the EXTENSIONALITY SIMULATION PRINCIPLE it is now ensured that configurations of *set* entities always behave in the desired way: Each configuration of *set* entities uniquely specifies a set. Which set each *set* entity specifies can be stated as a convention:

Convention 10 *For each signature with sets, for each grammar with sets, for each exhaustive model* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$ *of that grammar, for each* $u \in \mathsf{U}$*, if* $\mathsf{S}(u) \sqsubseteq$ *set, then* u *specifies the set* σ*, where*

$$\sigma = \left\{ o \in \mathsf{U} \;\middle|\; \begin{array}{l} \textit{for some } ass \in Ass_{\mathsf{I}}, \\ \textit{for some } n \in \mathbb{N}, \\ \quad T^{ass}_{\mathsf{I}}(:\underbrace{\text{MORE}\ldots\text{MORE}}_{n\ times}\ \text{ELE})(u) \textit{ is defined, and} \\ \quad o = T^{ass}_{\mathsf{I}}(:\underbrace{\text{MORE}\ldots\text{MORE}}_{n\ times}\ \text{ELE})(u) \end{array} \right\}.$$

In CONVENTION 10, the notion "signature with sets" refers to signatures that contain the sort hierarchy and the appropriateness specifications of the *set* hierarchy, (79), and the relation symbols of the relations in the preceding principles; and a "grammar with sets" is a grammar whose signature is a signature with sets and whose theory contains the SET PRINCIPLE and the EXTENSIONALITY SIMULATION PRINCIPLE along with the definitions of their respective relations.

The final ingredient of the present indirect approach to sets in HPSG 94 is a number of set access relations or set operations. These are the usual relations that are needed to express grammatical principles that refer to sets, such as set union, set intersection, and set difference. They are all defined on the basis of the `member` relation, but in contrast to the membership relation their arguments are restricted to chains and *set* entities; they do not have lists as arguments.

In the remainder of this section, I will illustrate briefly how the relevant set operations can be defined as RSRL relations by providing definitions for those set operations that are most frequently used or will be needed in the formalization of the grammar of Pollard and Sag 1994.

(87) *Approximating the standard set operations*

$$
\begin{array}{l}
\texttt{union}(x, y, z) \stackrel{\forall}{\Longleftarrow} \\
\quad \forall a\, (\texttt{member}(a, z) \;\leftrightarrow\; (\texttt{member}(a, x) \;\vee\; \texttt{member}(a, y))) \;\wedge \\
\quad \texttt{set-properties}({}^{x}[\mathit{chain} \vee \mathit{set}]) \wedge \\
\quad \texttt{set-properties}({}^{y}[\mathit{chain} \vee \mathit{set}]) \wedge \\
\quad \texttt{set-properties}({}^{z}[\mathit{chain} \vee \mathit{set}])
\end{array}
$$

$$
\begin{array}{l}
\texttt{difference}(x, y, z) \stackrel{\forall}{\Longleftarrow} \\
\quad \forall a\, (\texttt{member}(a, z) \;\leftrightarrow\; (\texttt{member}(a, x) \;\wedge\; \neg\texttt{member}(a, y))) \;\wedge \\
\quad \texttt{set-properties}({}^{x}[\mathit{chain} \vee \mathit{set}]) \wedge \\
\quad \texttt{set-properties}({}^{y}[\mathit{chain} \vee \mathit{set}]) \wedge \\
\quad \texttt{set-properties}({}^{z}[\mathit{chain} \vee \mathit{set}])
\end{array}
$$

$$
\begin{array}{l}
\texttt{intersection}(x, y, z) \stackrel{\forall}{\Longleftarrow} \\
\quad \forall a\, (\texttt{member}(a, z) \;\leftrightarrow\; (\texttt{member}(a, x) \;\wedge\; \texttt{member}(a, y))) \;\wedge \\
\quad \texttt{set-properties}({}^{x}[\mathit{chain} \vee \mathit{set}]) \wedge \\
\quad \texttt{set-properties}({}^{y}[\mathit{chain} \vee \mathit{set}]) \wedge \\
\quad \texttt{set-properties}({}^{z}[\mathit{chain} \vee \mathit{set}])
\end{array}
$$

$$
\begin{array}{l}
\texttt{disjoint-union}(x, y, z) \stackrel{\forall}{\Longleftarrow} \\
\quad \texttt{union}(x, y, z) \;\wedge\; \exists w\, \texttt{intersection}(x, y, w \,\|\; \|)
\end{array}
$$

$$
\begin{array}{l}
\texttt{subseteq}(x, y) \stackrel{\forall}{\Longleftarrow} \\
\quad \forall a\, (\texttt{member}(a, x) \;\rightarrow\; \texttt{member}(a, y)) \;\wedge \\
\quad \texttt{set-properties}({}^{x}[\mathit{chain} \vee \mathit{set}]) \wedge \\
\quad \texttt{set-properties}({}^{y}[\mathit{chain} \vee \mathit{set}])
\end{array}
$$

The explicit restriction to `set-properties` of the denotation of all variables in the arguments of the relations is for the most part vacuous, since *set* entities already obey the Set Principle, (84). In the case of chains, however, the extra condition enforces the intuition of Pollard and Sag that the entities which, in their view, represent sets are such that every member of each set occurs only once. Except for the special provision for chains, the definitions of the set operations follow the standard definitions of set union, set difference, and set intersection; the subset relation is likewise standard. The definition of disjoint union uses an empty chain, denoted by w, in order to require that x and y be disjoint sets (or chains), because this is the simplest way to capture all cases of possible denotations of x, y and z, including the one in which x, y, and z denote chains and no *eset* entity is a component of the entity of which the relation is supposed to hold.

Occasionally, linguists use the more general notions of the union and the disjoint union of a collection of sets. In the literature, the respective operations are called successive union and successive disjoint union. In practice, these operations are applied to an arbitrary but finite number of sets. A convenient way to capture the necessary relations is to generalize the relations `union` and `disjoint-union` accordingly. The manner in which I define the generalized relations in (88) is derived from the way in which they are used in the literature: Some sets are unioned together whose relative positions in a configuration of entities are (recursively) described. For a modular design of principles and relations, it is best to define a first relation that relates the sets in the relevant configurations to a chain that enumerates them. In a second step, the successive (disjoint) union relation holds between that chain and the set which we are after.[25] Since in some cases, the sets that we wish to union together might already stand on a list, I choose a somewhat more general definition and allow tapes of sets to be unioned:

(88) $\texttt{successive-union}(x, y) \stackrel{\forall}{\Longleftarrow}$
$x \ll\gg \land {}^{y}[\mathit{echain} \lor \mathit{eset}]$

$\texttt{successive-union}(x, y) \stackrel{\forall}{\Longleftarrow}$
$x \ll s|\, r \gg \land \texttt{successive-union}(r, z) \land \texttt{union}(s, z, y)$

$\texttt{successive-disjoint-union}(x, y) \stackrel{\forall}{\Longleftarrow}$
$x \ll\gg \land {}^{y}[\mathit{echain} \lor \mathit{eset}]$

$\texttt{successive-disjoint-union}(x, y) \stackrel{\forall}{\Longleftarrow}$
$x \ll s|\, r \gg \land \texttt{successive-disjoint-union}(r, z) \land$
$\texttt{disjoint-union}(s, z, y)$

The definitions of the two relations are identical, except that in the recursive case, `successive-disjoint-union` is defined with `disjoint-union` where `successive-union` simply employs `union`. The base case is the successive union of the sets on an empty tape, which results in an empty set or an empty

[25] The formalization of the NONLOCAL FEATURE PRINCIPLE in (96) below will illustrate this design idea.

chain. In the recursive case, the first set on the tape is (disjoint) unioned with the second argument (the "result") of the application of successive (disjoint) union to the tail of the tape.

In the linguistic literature, the relations whose meanings I have defined in (87) and (88) do not occur in the notation in which I have presented them. In the following section I will show with examples how the set notations of informal AVM diagrams can be understood as notational shorthands for formulae that comprise the above relations. To make the intuitive relationship between the actual notation that one finds in the literature and my definitions explicit, I will capture the meaning of the informal AVM diagrams in notational conventions that integrate a version of them with RSRL's AVM descriptions.

4.4.2 Notation and Examples

In the previous section, I have outlined an approach to sets in HPSG 94 that simulates sets with entities of sort *set* and the configurations of entities that are components of them. I have precisely characterized these configurations of entities and which set each *set* entity in the model of a grammar specifies. Finally, I have defined the most common set operations as relations between *set* entities (and chains). This being accomplished, the purely technical side of my indirect explication of sets is completed. However, I have set myself the task to get as close to existing notational conventions as possible in order to bridge the gap between existing grammar specifications and their rigorous model-theoretic interpretation. In this respect, the present proposal can and should still be improved upon. The presentation of two representative principles, the NONLOCAL FEATURE PRINCIPLE and clause (a) of the SEMANTICS PRINCIPLE, in Section 4.4.3 will benefit from the notational amendments.

The actual notational conventions for the description of sets can be interpreted in terms of the `member` relation and in terms of the set operations that are defined on the basis of `member`. In informal AVM diagrams, linguists write the standard set-theoretic symbols, such as '$\cup$' and '$\cap$', for the union and for the intersection of two sets. They can be understood as functional variants of the corresponding relational notation in an obvious way:

Convention 11 *Suppose a grammar $\langle \Sigma, \theta \rangle$ containing the relations* `union`*,* `intersection`*,* `difference` *and* `disjoint-union` *with their meanings as* *defined in (87). The relational* Σ *AVM formulae,* `union`(x, y, z)*,* `intersection`(x, y, z)*,* `difference`(x, y, z) *and* `disjoint-union`(x, y, z) *can then be written functionally as* ${}^{z}[x \cup y]$*,* ${}^{z}[x \cap y]$*,* ${}^{z}[x \setminus y]$ *and* ${}^{z}[x \uplus y]$*, respectively. In addition to the syntactic distribution of their relational counterparts, the functional variants can be written where tagged* Σ *AVM boxes are permitted, and they are understood as standing in conjunction with the* Σ *AVM formula in which they occur, leaving a copy of the variable* z *in their place behind. If they are nested in a* Σ *AVM matrix, the variable corresponding to* z *and the bracktes may be omitted. In that case, the missing variable is taken to be the alphabetically first variable that does not yet occur in the overall* Σ *AVM formula.*

Suppose furthermore that a grammar contains the relation `subseteq` *with the meaning defined in (87). Then* $x \subseteq y$ *is the infix notation for* `subseteq`(x, y)*.*

The effects of Convention 11 can be observed in (89). (89a) is a description that is adapted from Sag 1997, p. 449, and illustrates a lexical mechanism for the propagation of the values of nonlocal attributes in a theory of relative clause constructions. Note that the signature underlying the AVM descriptions in (89) differs slightly from the signature of Pollard and Sag 1994.[26] The ss nonloc rel value of the *word* entities in the denotation of (89a) is functionally described as the disjoint union of the nonloc rel values of the elements on the arg-st list of the *word* entities. By our convention, this is just another, more compact and more perspicious way of writing (89b):

(89) a.
$$\begin{bmatrix} \textsc{phon} & \langle \textit{picture} \rangle \\ \textsc{arg-st} & \left\langle \boxed{1}\begin{bmatrix}\textsc{nonloc rel} & \boxed{3}\end{bmatrix}, \boxed{2}\begin{bmatrix}\textsc{nonloc rel} & \boxed{4}\end{bmatrix}\right\rangle \\ \textsc{ss} & \begin{bmatrix} \textsc{loc cat} & \begin{bmatrix}\textsc{head} & \textit{noun} \\ \textsc{valence} & \begin{bmatrix}\textsc{spr} & \langle\boxed{1}\rangle \\ \textsc{comps} & \langle\boxed{2}\rangle\end{bmatrix}\end{bmatrix} \\ \textsc{nonloc rel} & \boxed{3} \uplus \boxed{4}\end{bmatrix}\end{bmatrix}$$

[26] In fact, it is not clear from Sag 1997 for which sort the attribute arg-st is appropriate. Fortunately, that is not relevant in the present discussion. In (89), I omit non-pertinent details of Sag's description that demand that the complement be a PP[of].

$$\text{b.}\quad \begin{bmatrix} \text{PHON} & \langle \textit{picture} \rangle \\ \text{ARG-ST} & \left\langle \boxed{1}\begin{bmatrix}\text{NONLOC REL } \boxed{3}\end{bmatrix}, \boxed{2}\begin{bmatrix}\text{NONLOC REL } \boxed{4}\end{bmatrix}\right\rangle \\ \text{SS} & \begin{bmatrix} \text{LOC CAT} & \begin{bmatrix}\text{HEAD} & \textit{noun} \\ \text{VALENCE} & \begin{bmatrix}\text{SPR} & \langle\boxed{1}\rangle \\ \text{COMPS} & \langle \boxed{2}\rangle\end{bmatrix}\end{bmatrix} \\ \text{NONLOC REL } \boxed{5} & \end{bmatrix}\end{bmatrix}$$
$$\wedge\ \texttt{disjoint-union}(\boxed{3},\boxed{4},\boxed{5})$$

The disjoint union in (89a) is an instance of the most interesting case of the CONVENTION 11. The functional expression is inserted into a complex AVM matrix at the place of an AVM box, and the "result" variable is left implicit. Hence '$\boxed{3} \uplus \boxed{4}$' abbreviates '$\boxed{5}[\boxed{3} \uplus \boxed{4}]$', which is in turn expanded in (89b) to the appropriate relational AVM formula standing in conjunction with the AVM formula of (89a) minus the functional expression.[27]

Another typical AVM notation that is inspired by the standard set-theoretic notation specifies the members of a set with AVM boxes that are enumerated between curly brackets, similar to the description of lists in HPSG. Again, interpreting the AVM notation in terms of the `member` relation is straightforward:

Convention 12 *For each signature Σ, for each $v \in \mathcal{VAR}$, for each $v_1 \in \mathcal{VAR}$, $\dots$, for each $v_n \in \mathcal{VAR}$, for each $\beta_1 \in \mathbb{UBOX}^{\Sigma}$, $\dots$, for each $\beta_n \in \mathbb{UBOX}^{\Sigma}$, $v\,\{v_1\,\beta_1, \dots, v_n\,\beta_n\}$ is an alternative way of expressing the Σ AVM formula*

$$\texttt{member}(v_1, v)\ \wedge \dots \wedge\ \texttt{member}(v_n, v)\ \wedge\ v_1\,\beta_1 \wedge \dots \wedge\ v_n\,\beta_n$$
$$\wedge\ \forall x\,(\texttt{member}(x, v)\ \rightarrow\ (x = v_1\ \vee \dots \vee\ x = v_n)).$$

Parallel to the functional notation of set relations, $v\,\{v_1\,\beta_1, \dots, v_n\,\beta_n\}$ is treated syntactically as a tagged AVM box. If it occurs inside of an AVM matrix, the variable corresponding to v may again be omitted. If it is omitted, it is taken to be the alphabetically first variable that does not yet occur in the overall AVM formula. If $v\,\{v_1\,\beta_1, \dots, v_n\,\beta_n\}$ occurs nested in another AVM formula, it is understood as standing in conjunction with that AVM formula, leaving a copy of its tag behind.

[27]The correspondence between the functional notation of relations between sets discussed here and the common functional AVM notation for relations such as `append` and `shuffle`, '$\textit{list}_1 \oplus \textit{list}_2$' and '$\textit{list}_1 \bigcirc \textit{list}_2$', is easy to see. The latter can just as easily be regarded as a notational variant of relational formulae containing the former.

I call each $v\,\{v_1\,\beta_1, \ldots, v_n\,\beta_n\}$ a Σ set specification. As already noted in Pollard and Moshier 1990, p. 292, the interpretation of that HPSG notation differs slightly from the interpretation of the standard set-theoretic notation; and CONVENTION 12 captures the difference. In the standard notation, $\{a, a\}$ and $\{a\}$ both denote the singleton set to which only a belongs. Crucially, repetitions of the same symbol between the brackets do not change the meaning of the expression. This it not the case in AVM diagrams. The difference is due to the different interpretations that the symbols between the curly brackets receive. Whereas in the usual set-theoretic notation, each type of symbol denotes one set-theoretic object, AVM expressions denote sets of entities; and the formulae between curly brackets denote components of some entity. As a consequence, different occurrences of identical expressions within one formula can denote different—possibly identical-looking but still not token identical—components of entities. Consider the following examples:

(90) a. $\begin{bmatrix} \textit{phrase} \\ \text{SS NL INH SLASH} \left\{ \boxed{1}\begin{bmatrix}\text{CAT} \begin{bmatrix}\text{HEAD} & \textit{noun} \\ \text{SUBCAT} & \langle\rangle\end{bmatrix}\end{bmatrix}, \boxed{2}\begin{bmatrix}\text{CAT} \begin{bmatrix}\text{HEAD} & \textit{noun} \\ \text{SUBCAT} & \langle\rangle\end{bmatrix}\end{bmatrix} \right\} \end{bmatrix}$

b. $\begin{bmatrix} \textit{phrase} \\ \text{SS NL INH SLASH} \left\{ \boxed{1}\begin{bmatrix}\text{CAT} \begin{bmatrix}\text{HEAD} & \textit{noun} \\ \text{SUBCAT} & \langle\rangle\end{bmatrix}\end{bmatrix}, \boxed{2}\begin{bmatrix}\text{CAT} \begin{bmatrix}\text{HEAD} & \textit{prep} \\ \text{SUBCAT} & \langle\rangle\end{bmatrix}\end{bmatrix} \right\} \end{bmatrix}$

At first glance, the set specification in (90a) seems to say that the SYNSEM NONLOCAL INHERITED SLASH value of the described phrases must be sets that contain the *local* entities of two saturated nominal signs. On closer inspection, however, it turns out that nothing prevents these two *local* entities from actually being one and the same entity, i.e., $\boxed{1} = \boxed{2}$. A look at the AVM formula that the set specification abbreviates makes that clear:[28] What the set specification is saying is only that, (1), a saturated NP *local* entity, $\boxed{1}$, belongs to the set, (2), a saturated NP *local* entity, $\boxed{2}$, belongs to the set, and, (3) if there is anything else in the set, it actually equals either $\boxed{1}$ or $\boxed{2}$; or, in other words, there is nothing in the set except $\boxed{1}$ and $\boxed{2}$. But note

[28]In my discussion, I tacitly assume (as I always do when nothing else is indicated) that (90a) is subject to the existential closure convention, CONVENTION 4, i.e., it is a description, and $\boxed{1}$ and $\boxed{2}$ are existentially bound by quantifiers that take scope over the entire formula.

that nothing in these three conditions excludes that $\boxed{1}$ and $\boxed{2}$ are identical. Therefore, we only know that at least one and maximally two saturated NP *local* entities are in the set. If we want to make sure that they are distinct, we must explicitly say so and conjunctively add the formula $\neg\boxed{1} = \boxed{2}$.

Of course, that does not mean that there must always be an inequation if we want to exclude the identity of two members of a set that we describe. The identity might already be excluded by the fact that the two descriptions are not consistent. Their conjunction always denotes the empty set in any model of the grammar. This situation can be caused by the signature. In (90b), the SYNSEM NONLOCAL INHERITED SLASH value of the described phrases must be sets with a saturated NP *local* entity, $\boxed{1}$, and a saturated PP *local* entity, $\boxed{2}$. The HEAD values of the two *local* entities, entities of sort *noun* and *prep*, can never be identical, because they are of different species; and the species partition the entities in the models. It follows immediately that $\boxed{1}$ and $\boxed{2}$ cannot be identical, since their CATEGORY HEAD values are necessarily distinct.

In summary, if some descriptions of the elements of a set in a set description are consistent, their number equals the maximum number of members that they denote; but they may denote any smaller number of elements in the set, down to one. If, however, all the descriptions of the members are pairwise inconsistent, the number of descriptions equals the number of members of the set. Inspecting CONVENTION 12 also reveals that, as a generalization of its underlying idea, writing $\{\}$ in the place of a (tagged) AVM box can simply be seen as another way of writing $[eset]$.

Note that the convention for the functional notation of set operations, CONVENTION 11, interacts interestingly with CONVENTION 6, which allows tagged boxes in the arguments of relations instead of mere variables. In the light of the second convention, it is legitimate to write tagged boxes in the arguments of the functional notation of set operations, and, as a result, we can even interpret AVM formulae as the one in (91) by applying the notational conventions:

(91) *Description of the* LOCAL *value of the adjective* 'red' *(after Pollard and Sag 1994, p. 55)*

$$\begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{adj} & \\ \text{MOD LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix}\text{HEAD} & \textit{noun} \\ \text{SUBCAT} & \langle \textit{synsem} \rangle\end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{INDEX} & \boxed{1} \\ \text{RESTR} & \boxed{2}\end{bmatrix}\end{bmatrix}\end{bmatrix} \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{INDEX} & \boxed{1} \\ \text{RESTR} & \left\{ \boxed{3}\begin{bmatrix}\text{RELN} & \textit{red} \\ \text{ARG} & \boxed{1}\end{bmatrix}\right\} \cup \boxed{2}\end{bmatrix}\end{bmatrix}$$

First, the set specification in (91) leaves its tag implicit. Adding it leads to the following description of the CONTENT RESTR value of the *local* entity of *red*:

(92) $$\boxed{4} \left\{ \boxed{3}\begin{bmatrix}\text{RELN} & \textit{red} \\ \text{ARG} & \boxed{1}\end{bmatrix}\right\} \cup \boxed{2}$$

This formula can in turn be expanded to (93) by adding the omitted variable that denotes the result of the set union:

(93) $$\boxed{5} \left[\boxed{4} \left\{ \boxed{3}\begin{bmatrix}\text{RELN} & \textit{red} \\ \text{ARG} & \boxed{1}\end{bmatrix}\right\} \cup \boxed{2} \right]$$

One final application of the conventions 11 and 12 leads to the following formula in which no abbreviatory set notation occurs:

(94) $$\begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{adj} & \\ \text{MOD LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix}\text{HEAD} & \textit{noun} \\ \text{SUBCAT} & \langle \textit{synsem} \rangle\end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{INDEX} & \boxed{1} \\ \text{RESTR} & \boxed{2}\end{bmatrix}\end{bmatrix}\end{bmatrix} \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{INDEX} & \boxed{1} \\ \text{RESTR} & \boxed{5}\end{bmatrix}\end{bmatrix}$$

$$\wedge\ \texttt{union}(\boxed{4}, \boxed{2}, \boxed{5})\ \wedge\ \texttt{member}\left(\boxed{3}\begin{bmatrix}\text{RELN} & \textit{red} \\ \text{ARG} & \boxed{1}\end{bmatrix}, \boxed{4}\right)$$
$$\wedge\ \forall x\ (\texttt{member}(x, \boxed{4})\ \rightarrow\ x = \boxed{3})$$

Note also that $\boxed{4}$ might necessarily denote a chain if no singleton set $\boxed{4}$ containing the entity $\boxed{3}$ is a component of the *local* entity of a given instance of the adjective *red*.

Finally, for further convenience, I extend the functional notation convention to successive (disjoint) union:

Convention 13 *Suppose a grammar* $\langle \Sigma, \theta \rangle$ *containing the relations* `successive-union`(y, z) *and* `successive-disjoint-union`(y, z) *with their meanings as defined in (88). I write these relational formulae in functional notation as* ${}^{z}[\bigcup y]$ *and* ${}^{z}[\biguplus y]$*, respectively, where* y *is the first argument and* z *is the second argument of each relation. The usual conventions about the syntactic distribution and abbreviations of the functional notation apply (see* CONVENTION *11).*

We will see an application of CONVENTION 13 in the formalization of the NONLOCAL FEATURE PRINCIPLE in the next section.

4.4.3 Application

In the preceding sections I have argued that the sets that occur in the descriptions of linguistic entities in HPSG 94 can be characterized as configurations of entities that are components of entities in the denotation of a sort, *set*. I have defined a number of access relations to talk about the configurations of entities that picture sets, and I have stated some notational conventions that extend the AVM syntax of RSRL in a way that mimics the common set notation of the informal AVM diagrams of the literature. It is now time to illustrate the whole machinery that I have put into place with the formalization of two of the most prominent principles concerning set values of Pollard and Sag 1994, the NONLOCAL FEATURE PRINCIPLE and clause (a) of the SEMANTICS PRINCIPLE. Consider the NONLOCAL FEATURE PRINCIPLE first:

(95) *The* NONLOCAL FEATURE PRINCIPLE *(*NFP*), (Pollard and Sag, 1994, p. 400)*

> In a headed phrase, for each nonlocal feature F = SLASH, QUE or REL, the value of SYNSEM | NONLOCAL| INHERITED | F is the set difference of the union of the values on all the daughters, and the value of SYNSEM | NONLOCAL | TO-BIND | F on the HEAD-DAUGHTER.

The NFP regulates the relationships between three set-valued attributes of the mother and the daughters in four kinds of phrases. Recall that phrases are distinguished by their DTRS values. The headed phrases are head adjunct phrases, head complement phrases, head filler phrases, and head marker phrases. Since each kind of headed phrase has different kinds of daughters besides the head daughter, the conditions that the NFP imposes on each type of phrase are in fact slightly different, because they must refer to different daughter attributes. However, the conditions are largely parallel for each type of phrase. Only head complement phrases require a slightly different treatment, because they do not have a fixed number of complement daughters, whereas the number of daughters is fixed for head adjunct phrases, head filler phrases, and head marker phrases. In the formalization below, the necessary distinctions are captured in a relation, `collect-dependencies`, that holds between the DTRS value and the (possibly successive) unions of each one of the three nonlocal inherited sets of the head daughter and of each nonhead daughter:

(96) *The* NONLOCAL FEATURE PRINCIPLE *formalized*

$$\left[\textsc{dtrs}\ \ \textit{headed-struc}\right] \longrightarrow$$

$$\begin{bmatrix} \textsc{ss nonloc inherited} & \begin{bmatrix} \textsc{slash} & \boxed{5} \setminus \boxed{1} \\ \textsc{que} & \boxed{6} \setminus \boxed{2} \\ \textsc{rel} & \boxed{7} \setminus \boxed{3} \end{bmatrix} \\ \textsc{dtrs}\ \boxed{4} & \begin{bmatrix} \textsc{head-dtr ss nonloc to-bind} & \begin{bmatrix} \textsc{slash} & \boxed{1} \\ \textsc{que} & \boxed{2} \\ \textsc{rel} & \boxed{3} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\wedge\ \texttt{collect-dependencies}(\boxed{4}, \boxed{5}, \boxed{6}, \boxed{7})$$

Suppose for the moment that for the DTRS value, $\boxed{4}$, of each phrase, $\boxed{5}$ is the union of the INHERITED SLASH sets of all the daughters, $\boxed{6}$ is the union of the INHERITED QUE sets of all the daughters, and $\boxed{7}$ is the union of the INHERITED REL sets of all the daughters. The idea of the NFP is that by subtracting from each of these three sets the members of the corresponding NONLOCAL TO-BIND set of the phrase's HEAD-DAUGHTER, we know the respective INHERITED values of the phrase itself. In (96), the sets resulting from forming the set difference are specified in the functional notation of the set difference relation. Therefore, their functional description is directly written behind the appropriate nonlocal inherited attributes of the mother, in accordance with CONVENTION 11. Note that, if one of the NONLOCAL TO-BIND sets is nonempty, the union of the inherited sets of all the daughters from which it is subtracted is a set that is not a component of the phrase; yet, the NFP is obviously intended to be true of the phrase. It is exactly for these cases that all set operations are defined for both components and chains of components as their arguments: A set in the argument of each set relation does not have to be a component of the described entity; it suffices if each of its members is a component.

Since the entities in the denotation of the tags $\boxed{5}$, $\boxed{6}$ and $\boxed{7}$ are the ones that can be either sets or chains, the relation `collect-dependencies` must allow both cases. Its meaning is defined in four clauses, one for each maximally specific subsort of *headed-struc*. The idea is to union the relevant sets of the head daughter and of the nonhead daughter(s). While this is trivial for the headed structures that have exactly one nonhead daughter, head complement structures, which have an arbitrary finite number of complement daughters, constitute a more complex case. I treat them with an additional relation that provides an enumeration on a tape of the relevant sets, which, in turn, are successively unioned together:

(97)
$$\texttt{collect-dependencies}(v, x, y, z) \stackrel{\forall}{\Longleftarrow}$$
$$^{v}\begin{bmatrix} \textit{head-comp-struc} \\ \text{HEAD-DTR SS NONLOC INHERITED} \begin{bmatrix} \text{SLASH} & \boxed{1} \\ \text{QUE} & \boxed{2} \\ \text{REL} & \boxed{3} \end{bmatrix} \\ \text{COMP-DTRS } \boxed{4} \end{bmatrix}$$
$$\wedge\ \texttt{enumerate-slashes}(\boxed{4}, \boxed{5})\ \wedge\ ^{x}\big[\textstyle\bigcup c_1 \ll \boxed{1} \mid \boxed{5} \gg\big]$$
$$\wedge\ \texttt{enumerate-ques}(\boxed{4}, \boxed{6})\ \wedge\ ^{y}\big[\textstyle\bigcup c_2 \ll \boxed{2} \mid \boxed{6} \gg\big]$$
$$\wedge\ \texttt{enumerate-rels}(\boxed{4}, \boxed{7})\ \wedge\ ^{z}\big[\textstyle\bigcup c_3 \ll \boxed{3} \mid \boxed{7} \gg\big]$$

$$\texttt{collect-dependencies}(v, x, y, z) \stackrel{\forall}{\Longleftarrow}$$
$$^{v}\begin{bmatrix} \textit{head-adj-struc} \\ \text{HEAD-DTR SS NONLOC INHERITED} \begin{bmatrix} \text{SLASH} & \boxed{1} \\ \text{QUE} & \boxed{2} \\ \text{REL} & \boxed{3} \end{bmatrix} \\ \text{ADJUNCT-DTR SS NONLOC INHERITED} \begin{bmatrix} \text{SLASH} & \boxed{4} \\ \text{QUE} & \boxed{5} \\ \text{REL} & \boxed{6} \end{bmatrix} \end{bmatrix}$$
$$\wedge\ ^{x}\big[\boxed{1} \cup \boxed{4}\big]\ \wedge\ ^{y}\big[\boxed{2} \cup \boxed{5}\big]\ \wedge\ ^{z}\big[\boxed{3} \cup \boxed{6}\big]$$

$$\texttt{collect-dependencies}(v,x,y,z) \overset{\forall}{\Longleftarrow}$$
$$\begin{array}{l} {}^{v}\left[\begin{array}{ll} \textit{head-filler-struc} & \\ \text{HEAD-DTR SS NONLOC INHERITED} & \left[\begin{array}{ll}\text{SLASH} & \boxed{1}\\ \text{QUE} & \boxed{2}\\ \text{REL} & \boxed{3}\end{array}\right] \\ \text{FILLER-DTR SS NONLOC INHERITED} & \left[\begin{array}{ll}\text{SLASH} & \boxed{4}\\ \text{QUE} & \boxed{5}\\ \text{REL} & \boxed{6}\end{array}\right] \end{array}\right] \\ \wedge\ {}^{x}\left[\boxed{1}\cup\boxed{4}\right] \ \wedge\ {}^{y}\left[\boxed{2}\cup\boxed{5}\right] \ \wedge\ {}^{z}\left[\boxed{3}\cup\boxed{6}\right] \end{array}$$

$$\texttt{collect-dependencies}(v,x,y,z) \overset{\forall}{\Longleftarrow}$$
$$\begin{array}{l} {}^{v}\left[\begin{array}{ll} \textit{head-mark-struc} & \\ \text{HEAD-DTR SS NONLOC INHERITED} & \left[\begin{array}{ll}\text{SLASH} & \boxed{1}\\ \text{QUE} & \boxed{2}\\ \text{REL} & \boxed{3}\end{array}\right] \\ \text{MARKER-DTR SS NONLOC INHERITED} & \left[\begin{array}{ll}\text{SLASH} & \boxed{4}\\ \text{QUE} & \boxed{5}\\ \text{REL} & \boxed{6}\end{array}\right] \end{array}\right] \\ \wedge\ {}^{x}\left[\boxed{1}\cup\boxed{4}\right] \ \wedge\ {}^{y}\left[\boxed{2}\cup\boxed{5}\right] \ \wedge\ {}^{z}\left[\boxed{3}\cup\boxed{6}\right] \end{array}$$

For all headed structures except for head complement structures, each pair of corresponding nonlocal inherited attributes of the single nonhead daughter and of the head daughter is unioned. In head complement structures, however, there can be any (finite) number of nonhead daughters. Therefore, we need to take the successive union of the sets of each nonlocal inherited attribute of the head daughter with the appropriate sets of all complement daughters. For that purpose, the relations `enumerate-slashes`, `enumerate-ques` and `enumerate-rels` specify three tapes of each nonlocal inherited set of all complement daughters. The first argument of the three relations is a list of signs (the list of complement daughters), their second argument is a tape of the SLASH, QUE and REL sets, respectively, of all the signs on the (complement daughters) list:

(98) $\texttt{enumerate-slashes}(x,y) \overset{\forall}{\Longleftarrow}$
$$x\,\langle\rangle \ \wedge\ y \ll\gg$$

$$\texttt{enumerate-slashes}(x,y) \overset{\forall}{\Longleftarrow}$$
$$x\left\langle\left[\text{SS NONLOC INHERITED SLASH } \boxed{1}\right]\middle|\,\boxed{2}\right\rangle \wedge\ y \ll\boxed{1}\,|\,\boxed{3}\gg$$
$$\wedge\ \texttt{enumerate-slashes}(\boxed{2},\boxed{3})$$

$$\texttt{enumerate-ques}(x,y) \stackrel{\forall}{\Longleftarrow}$$
$$x\,\langle\rangle \;\wedge\; y \ll\gg$$

$$\texttt{enumerate-ques}(x,y) \stackrel{\forall}{\Longleftarrow}$$
$$x\left\langle \left[\text{SS NONLOC INHERITED QUE } \boxed{1}\right] \middle| \boxed{2} \right\rangle \wedge\; y \ll \boxed{1} \,|\, \boxed{3} \gg$$
$$\wedge\; \texttt{enumerate-ques}(\boxed{2}, \boxed{3})$$

$$\texttt{enumerate-rels}(x,y) \stackrel{\forall}{\Longleftarrow}$$
$$x\,\langle\rangle \;\wedge\; y \ll\gg$$

$$\texttt{enumerate-rels}(x,y) \stackrel{\forall}{\Longleftarrow}$$
$$x\left\langle \left[\text{SS NONLOC INHERITED REL } \boxed{1}\right] \middle| \boxed{2} \right\rangle \wedge\; y \ll \boxed{1} \,|\, \boxed{3} \gg$$
$$\wedge\; \texttt{enumerate-rels}(\boxed{2}, \boxed{3})$$

The definition of the meaning of `enumerate-slashes`, `enumerate-ques` and `enumerate-rels` concludes the formalization of the NFP, (95). The example has shown that the set relations of Section 4.4.2 suffice to express the relationships between sets that are required by a typical principle that governs set values. Moreover, it could be observed that the formulation of principles in HPSG 94 treats sets similar to lists in some respects. Occasionally it is not required that an intermediary (set-valued) result in the relational characterization of a relationship between two set components of an entity is itself a set that is a component of all the entities that are supposed to be in the denotation of the principle; but all the members of the intermediary set are components.

Again, the use of chains could be avoided by rephrasing the NFP, and by defining specific relations that formalize the rephrased principle. However, just as in the case of the reformulation of the `shuffle` relation of Kathol and Pollard 1995 that avoids chains in Section 4.3.2, this would unnecessarily obscure the intuitions that originally led to the straightforward and intuitively transparent formulation of the NFP quoted in (95), and it would increase the number of relations that one needs to formalize the set principles of Pollard and Sag 1994.

Clause (a) of the Semantics Principle highlights more aspects of principles that talk about sets. In particular, it illustrates how HPSG 94 switches the perspective from the membership of elements on lists to the membership of elements on sets, where one of the two might be a "virtual" list or set, i.e., it is not a component of the described entity whereas its members are.

(99) *Clause (a) of the* SEMANTICS PRINCIPLE *(*SP-*a) (Pollard and Sag, 1994, p. 401–402)*[29]

(a) In a headed phrase, the RETRIEVED value is a list whose set of elements is disjoint from the QSTORE value set, and the union of those two sets is the union of the QSTORE values of the daughters.

Note the repeated reference to a set containing the elements of the RETRIEVED value, which itself is a list. While a RETRIEVED list exists in each headed phrase by virtue of the signature, it is not guaranteed that every headed phrase that is supposed to be well-formed according to the principle in (99) also contains a set whose members are the members of its RETRIEVED list as its component. Given the definitions of the set operations in Section 4.4.1, this will not be a problem for a direct formalization.

Parallel to the NFP, the SP-a refers to a union of sets that are a component of each daughter of each headed phrase. Hence I define a relation, `collect-qstores`, that relates each entity in the denotation of *headed-struc* to the (successive) union of the QSTORE sets of all its daughters:

(100) $\texttt{collect-dtrs-qstores}(x, y) \stackrel{\forall}{\Longleftarrow}$
$$^{x}\begin{bmatrix} \textit{head-comp-struc} \\ \text{HEAD-DTR QSTORE} & \boxed{1} \\ \text{COMP-DTRS} & \boxed{2} \end{bmatrix}$$
$$\land\ \texttt{enumerate-qstores}(\boxed{2}, \boxed{3}) \land\ ^{y}\!\left[\bigcup c \ll \boxed{1} | \boxed{3} \gg\right]$$

$\texttt{collect-dtrs-qstores}(x, y) \stackrel{\forall}{\Longleftarrow}$
$$^{x}\begin{bmatrix} \textit{head-adj-struc} \\ \text{HEAD-DTR QSTORE} & \boxed{1} \\ \text{ADJUNCT-DTR QSTORE} & \boxed{2} \end{bmatrix}$$
$$\land\ ^{y}\!\left[\boxed{1} \cup \boxed{2}\right]$$

$\texttt{collect-dtrs-qstores}(x, y) \stackrel{\forall}{\Longleftarrow}$
$$^{x}\begin{bmatrix} \textit{head-filler-struc} \\ \text{HEAD-DTR QSTORE} & \boxed{1} \\ \text{FILLER-DTR QSTORE} & \boxed{2} \end{bmatrix}$$
$$\land\ ^{y}\!\left[\boxed{1} \cup \boxed{2}\right]$$

[29] Clause (b) will receive a formalization in Appendix C, (179).

$$\texttt{collect-dtrs-qstores}(x, y) \stackrel{\forall}{\Longleftarrow} {}^{x}\begin{bmatrix} \textit{head-mark-struc} \\ \text{HEAD-DTR QSTORE} & \boxed{1} \\ \text{MARKER-DTR QSTORE} & \boxed{2} \end{bmatrix} \wedge {}^{y}\big[\boxed{1} \cup \boxed{2}\big]$$

As in the case of `collect-dependencies`, (97), the clauses for head adjunct structures, head filler structures and head marker structures are almost identical, since the union of exactly two sets, the QSTORE values of the head daughter and the QSTORE value of the nonhead daughter, stands in the `collect-dtrs-qstores` relation to the *headed structure* entity. For the remaining most specific subsort of *headed-struc*, *head-comp-struc*, another relation, `enumerate-qstores`, is needed. It relates each list of signs to the successive union of the QSTORE sets of all its members, only supposing that the list is finite and non-cyclic:

(101) $$\texttt{enumerate-qstores}(x, y) \stackrel{\forall}{\Longleftarrow} x\,\langle\rangle \wedge y \ll\gg$$

$$\texttt{enumerate-qstores}(x, y) \stackrel{\forall}{\Longleftarrow} x\left\langle \big[\text{QSTORE } \boxed{1}\big] \middle| \boxed{2} \right\rangle \wedge\ y \ll \boxed{1} | \boxed{3} \gg \wedge \texttt{enumerate-qstores}(\boxed{2}, \boxed{3})$$

Given (101), each *head-comp-struc* entity, x, with a finite, non-cyclic list of complement daughters stands in the `collect-dtrs-qstores` relation to the successive union, y, of all its daughters. Since we already know that for all other headed structures, we know now that each *headed-struc* entity stands in the `collect-dtrs-qstores` relation to the (successive) union, y, of all its daughters. With that result, the formalization of the SP-a is very simple:

(102) *The* SEMANTICS PRINCIPLE*, clause (a), formalized*

$$\big[\text{DTRS } \textit{headed-struc}\big] \rightarrow \begin{bmatrix} \text{DTRS} & \boxed{1} \\ \text{QSTORE} & \boxed{2} \\ \text{RETRIEVED} & \boxed{3} \end{bmatrix} \wedge \boxed{3} \stackrel{s}{=} \boxed{4} \wedge \texttt{collect-dtrs-qstores}\big(\boxed{1}, \boxed{5}\big[\boxed{4} \uplus \boxed{2}\big]\big)$$

Recall that $\stackrel{s}{=}$, (85), holds between all possible combinations of component lists, component sets, and chains of components of each entity. Since the entity denoted by $\boxed{4}$ stands in the `disjoint-union` relation with the QSTORE set of each phrase in the denotation of the SP-a, $\boxed{4}$ can only be a set or a chain of which the relation `set-properties`, (81), holds (by (87)). Since $\boxed{4}$ also stands in the $\stackrel{s}{=}$ relation to the RETRIEVED list, it is, thus, a set that contains the same members as the RETRIEVED list. By virtue of `collect-dtrs-qstore`, its disjoint union with the QSTORE set equals the (successive) union of the QSTORE sets of all daughters of the phrase. This is exactly what the informal characterization of the SP-a in (99) demands.

4.5 Parerga and Paralipomena

The purpose of this section is to discuss a number of diverse issues that arise when RSRL is applied to HPSG 94 that did not fit properly into any of the previous sections. First, I investigate and formalize a grammatical principle whose formulation initially suggests that it might pose problems for a faithful and intuitively direct formalization. Second, I turn to the issue of parametric sorts and say how RSRL can implement the parametric sorts of HPSG 94 grammars without admitting them as an extension of the notion of sorts in the sort hierarchy. Finally, I briefly indicate where the lexicon is located in grammars that are written in RSRL and what that means for two different approaches to lexical rules.

4.5.1 An Apparently Problematic Principle

There is one principle in the appendix of Pollard and Sag 1994 that seems to be less amenable to a straightforward formalization in RSRL than the ones I have discussed so far. It is the QUANTIFIER BINDING CONDITION, (QBC). In contrast to other principles, the formulations of the QBC that Pollard and Sag 1994 provides in the presentation of the main text and in the appendix differ drastically:

(103) *The* QUANTIFIER BINDING CONDITION *(Pollard and Sag, 1994, p. 327 and 402)*

First version (p. 327):

Given a quantifier contained within a CONTENT value, every occurrence within that CONTENT value of the quantifier's index must be captured by that quantifier.

Second version (p. 402):

Let X be a *quantifier*, with RESTINDEX | INDEX value Y, on the QUANTS list of a *psoa* Z, and P a path in Z whose value is Y. Let the address of X in Z be QUANTS | Q | FIRST. Then P must have a prefix of one of the following three forms:

(1) QUANTS | Q | FIRST | RESTINDEX | RESTRICTION;

(2) QUANTS | Q | REST; or

(3) NUCLEUS.

In an explanation of the terminology of the first version of the QBC, Pollard and Sag say what it means for a quantifier in the QUANTS list of a *psoa* to capture an occurrence of an index:

> ". . . the occurrence is either (1) in the RESTRICTION of the quantifier, (2) in another quantifier to the right of the quantifier in question on the same QUANTS list, or (3) in the NUCLEUS of the psoa in question." (Pollard and Sag, 1994, pp. 327–328)

They also add that with an occurrence of an index in a *psoa* they refer to different paths whose interpretations lead from the *psoa* entity to that index. Each path that does so constitutes an occurrence of the index. With that in mind, it is easy to see how the two formulations of the QBC correspond. Essentially, both of them start with a given quantifier and its index on the QUANTS list of a *psoa*, and they exhaustively enumerate other places in that *psoa* where the index may legitimately be found. The first version does that in semantic terms by saying of which entities the index may be a component. The second version is syntactic and makes statements about the syntactic shape of the paths whose interpretations may lead from the *psoa* entity to the index, thus forbidding other conceivable paths to that index. The following sketch of a description, (104), pictures the relevant configuration:

(104) *Illustration: Sketch of a description of entities of sort psoa*

$$
\begin{bmatrix}
\textit{psoa} & \\
\textsc{quants} & \left\langle \ldots, \begin{bmatrix}
\textit{quant} & \\
\textsc{det} & \textit{exists} \\
\textsc{restind} & \begin{bmatrix}
\textit{nom-obj} & \\
\textsc{index} & \begin{bmatrix}
\textit{index} & \\
\textsc{per} & \textit{per} \\
\textsc{num} & \textit{num} \\
\textsc{gend} & \textit{gend}
\end{bmatrix} \\
\textsc{restr} & \left\{\ldots, \textit{psoa}, \ldots\right\}
\end{bmatrix}
\end{bmatrix}, \ldots \right\rangle \\
\textsc{nucleus} & \begin{bmatrix}
\textit{qfpsoa} & \\
\textsc{soa-arg} & \textit{psoa} \\
\textsc{influence} & \begin{bmatrix}
\textit{ref} & \\
\textsc{per} & \textit{per} \\
\textsc{num} & \textit{num} \\
\textsc{gend} & \textit{gend}
\end{bmatrix} \\
\textsc{influenced} & \begin{bmatrix}
\textit{ref} & \\
\textsc{per} & \textit{per} \\
\textsc{num} & \textit{num} \\
\textsc{gend} & \textit{gend}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Assume that we are interested in the quantifier and its index on the QUANTS list whose description I have sketched. According to the (second version of the) QBC, it is only allowed to occur in the RESTR set of the quantifier, further down the QUANTS list, or in the NUCLEUS of the overall *psoa*. Or, in other words, the only place where it is not allowed to occur is as a component of any element on the QUANTS list that precedes the quantifier in question. This is of course a condition that should be easy to express in RSRL, since its language is designed to talk about components of entities.

A closer investigation reveals that the second or appendix version of the QBC is not equivalent to the first version. The first version considers the indices of quantifiers at any level of embedding within a CONTENT value. In terms of the illustration of the QBC in (104), the first version of the QBC additionally forbids occurrences of the index of a quantifier of the QUANTS list of the NUCLEUS SOA-ARG value of the described psoa as a component of an element of the QUANTS list and as a component of the NUCLEUS INFLUENCE and the NUCLEUS INFLUENCED values of the described psoa. The second version of the QBC does not restrict the distribution of indices of quantifiers on the QUANTS lists of embedded psoas in the psoas that embed them (except for the distribution in the embedded psoas themselves). Loosely speaking, the second version of the QBC only restricts the occurrences of the index of a quantifier within the predicate, p, over which the quantifier takes scope, whereas the first version of the QBC in addition forbids occurrences of the index within all predicates that embed p (except for p itself, where the index may appear in those places where it is captured by the quantifier). The predictions of the two versions of the QBC differ in interesting ways that concern the analysis of so-called donkey sentences.[30] However, since I am only concerned with the formalization of informally stated principles and not with their empirical or theoretical adequacy, I will not pursue this issue. Since I have set myself the task of formalizing the principles of the appendix of Pollard and Sag 1994, I choose the second version of the QBC.

On the other hand, it seems to be awkward to go the way of Pollard and Sag's second formulation of the principle and encode the QBC in terms of the possible paths that lead to a quantifier's index. Paths are sequences of attributes, and sequences of attributes are names for functions in the interpretations of a grammar. While, in principle, it is possible to encode conditions on these functions within the grammar itself, it is very cumbersome and should, therefore, best be avoided. For that reason, I take the second version of the QBC just as a precise description of the intended effects rather than as an informal specification of the way in which the principle should be formalized. What I am going to formalize below is, instead, a variant of Pollard and Sag's first formulation that is equivalent to the second version. However, I turn the statement around and say that, in each *psoa*, the index of each quantifier on its QUANTS list may not be a component of any quantifier preceding that quantifier. It is easy to see that this is equivalent to Pollard and Sag's second formulation of the QBC.

In order to be able to express that principle, I need to know where to find the indices that are components of a quantifier, because I must be able to prevent a given index from being the component of some quantifiers. The relation `index-of-a-quantifier` identifies indices in quantifiers. In fact, each index that is a component of a given quantifier stands in the `index-of-a-quantifier` relation with it. For a quantifier, y, x is an index that is a component of y:

[30]See Pollard and Sag 1994, p. 328, example (29), for the relevant construction.

(105) $\texttt{index-of-a-quantifier}(x, y) \stackrel{\forall}{\Longleftarrow}$
$\quad {}^{y}\left[\textsc{restind index } x\right]$

$\texttt{index-of-a-quantifier}(x, y) \stackrel{\forall}{\Longleftarrow}$
$\quad {}^{y}\left[\textsc{restind restr } \boxed{1}\right]$
$\quad \wedge\ \texttt{member}(\boxed{2}, \boxed{1})$
$\quad \wedge\ \texttt{index-of-a-psoa}(x, \boxed{2})$

Since the RESTR value of a quantifier can be a nonempty set of psoas, a second relation, `index-of-a-psoa`, is needed which relates psoas to the indices that are their components. According to the signature, there are two locations in *psoas* where we can find indices, namely their QUANTS list and their NUCLEUS. The case of the QUANTS list is easy, because its value is a list of quantifiers, and we already have a relation that relates quantifiers to their indices:

(106) $\texttt{index-of-a-psoa}(x, y) \stackrel{\forall}{\Longleftarrow}$
$\quad {}^{y}\left[\textsc{quants } \boxed{1}\right]$
$\quad \wedge\ \texttt{member}(\boxed{2}, \boxed{1})$
$\quad \wedge\ \texttt{index-of-a-quantifier}(x, \boxed{2})$

The case of the NUCLEUS is more difficult. The location of indices as components of a NUCLEUS value depends, of course, on the attributes that are defined on it. Unfortunately, Pollard and Sag (1994, p. 397) are not fully explicit about that part of their signature. In particular, they leave open which subsorts and how many subsorts the value of $\mathcal{F}\langle \mathit{psoa}, \textsc{nucleus}\rangle$, *qfpsoa*, subsumes, and which attributes are defined on the species that are subsumed by the sort *qfpsoa*. They insist, however, that the values of the attributes defined on those species be either of sort *psoa* or of sort *ref*. That is specific enough to give a characterization of the remaining clauses of `index-of-a-psoa` relative to the possible completions of Pollard and Sag's (1994) signature.

Assume that we have a completed signature which complies with the requirements put forth by Pollard and Sag (1994). Especially, we have $\mathcal{F}\langle \mathit{psoa}, \textsc{nucleus}\rangle = \mathit{qfpsoa}$. Let $\mathit{Species}_{qfpsoa} = \left\{\sigma \in \mathcal{S} \,\middle|\, \sigma \sqsubseteq \mathit{qfpsoa}\right\}$. For each $\sigma \in \mathit{Species}_{qfpsoa}$, let $A^{\sigma}_{\mathit{index}} = \left\{\alpha \in \mathcal{A} \,\middle|\, \mathcal{F}\langle \sigma, \alpha\rangle \sqsubseteq \mathit{index}\right\}$, and, for each $\sigma \in \mathit{Species}_{qfpsoa}$, let $A^{\sigma}_{\mathit{psoa}} = \left\{\alpha \in \mathcal{A} \,\middle|\, \mathcal{F}\langle \sigma, \alpha\rangle \sqsubseteq \mathit{psoa}\right\}$. $A^{\sigma}_{\mathit{index}}$ is the set of all *index*-valued attributes that are defined on *qfpsoa*, $A^{\sigma}_{\mathit{psoa}}$ is the set of all *psoa*-valued attributes that are defined on *qfpsoa*. These two sets are all I need to specify the remaining clauses of `index-of-a-psoa`:

(107) For each $\sigma \in Species_{qfpsoa}$, for each $\alpha \in A^{\sigma}_{index}$,

$$\texttt{index-of-a-psoa}(x, y) \overset{\forall}{\Longleftarrow} {}^{y}\left[\text{NUCLEUS} \begin{bmatrix}\sigma \\ \alpha \;\; x\end{bmatrix}\right]$$

is in the set of clauses of `index-of-a-psoa`, and
for each $\sigma \in Species_{qfpsoa}$, for each $\alpha \in A^{\sigma}_{psoa}$,

$$\texttt{index-of-a-psoa}(x, y) \overset{\forall}{\Longleftarrow} {}^{y}\left[\text{NUCLEUS} \begin{bmatrix}\sigma \\ \alpha \;\; \boxed{1}\end{bmatrix}\right] \wedge \texttt{index-of-a-psoa}(x, \boxed{1})$$

is in the set of clauses of `index-of-a-psoa`.

The set of clauses of the relation `index-of-a-psoa` is the smallest set that contains the clauses specified in (106) and (107).

Everything is now in place for the formalization of the QBC. Besides the relations we have just defined, it also employs `to-the-right`, (50), a relation which specifies that its first argument succeeds its second argument on the list in its third argument. It was needed above in the BINDING THEORY for the definition of the notion of one entity being more oblique than another entity.

(108) *The* QUANTIFIER BINDING CONDITION *formalized*

$$psoa \rightarrow$$

$$\neg\exists\boxed{1}\,\exists\boxed{2}\,\exists\boxed{3}\,\exists\boxed{4} \left(\begin{array}{l} \left[\text{QUANTS } \boxed{4}\right] \\ \wedge\ \texttt{to-the-right}\Big(\boxed{3}\left[\text{RESTIND INDEX } \boxed{1}\right], \boxed{2}, \boxed{4}\Big) \\ \wedge\ \texttt{index-of-a-quantifier}(\boxed{1}, \boxed{2}) \end{array}\right)$$

In this formulation, the QBC says that in every psoa, for each quantifier q with index i on the QUANTS list of the psoa, no quantifier that precedes q on the QUANTS list contains i as a component. That statement is clearly equivalent to the second version of the QBC given in (103).

4.5.2 Parametric Sorts

Up to now, I have always treated the sorts in the sort hierarchies of grammars as atomic symbols. While this treatment is unquestionably correct for most sorts in the signatures of HPSG 94 grammars, there are a few exceptions in which sort names are not meant to be atomic symbols. For lists and for sets, it is common practice to appeal to so-called *parametric sorts*.[31] Pollard and Sag explain the use of parametric sorts in HPSG 94 as follows:

> We employ limited parametric polymorphism for lists and sets as follows: where σ is a meta-variable over (nonparametric) sorts, we partition *list*(σ) into *elist* and *nelist*(σ) and declare *nelist*(σ)[FIRST σ, REST *list*(σ)]; likewise we partition *set*(σ) into *eset* and *neset*(σ). (Pollard and Sag, 1994, p. 396, fn. 2)

Since my treatment of sets—as explained in Section 4.4—equals the encoding of lists in all respects that are relevant here, I will ignore sets in the following discussion. It will be trivial to see how my solution to the treatment of parametric lists carries over to sets.

In my graphical notation for the appropriateness function and the sort hierarchy of signatures, Pollard and Sag's specification of parametric lists can be pictured as follows, where σ is a meta-variable over nonparametric sorts:

(109) *list(σ)*

elist		
nelist(σ)	FIRST	σ
	REST	*list(σ)*

With the specification in (109), Pollard and Sag intend to declare the sort hierarchy and the appropriateness function for all lists whose members are

[31] Parametric sorts are sometimes called parametric types. Since I adopt the terminology of Pollard and Sag 1994 that speaks of a sort hierarchy instead of a type hierarchy and reserves the term "type" for a different concept, I will call them parametric sorts throughout, even when I refer to literature that calls them parametric types. The idea of parametric polymorphism comes from logic programming (see Yardeni et al. 1992 and the references cited therein).

of exactly one nonparametric sort.[32] For example, (109) comprises the specification in the signature of their grammar of lists of *synsem* entities. (110) expresses that specification without a meta-variable:

(110) *list(synsem)*

elist		
nelist(synsem)	FIRST	*synsem*
	REST	*list(synsem)*

Given the fact that the set of nonparametric sorts in Pollard and Sag's grammar is finite, (109) induces a finite number of declarations of possible lists, of which (110) is one specific instance. Pollard and Sag 1994 uses parametric lists for specifying in the signature that lists that are appropriate for certain attributes may only contain entities of some given sort. For example, the entities on SUBCAT lists must be of sort *synsem* by virtue of the appropriateness specification for the sort *category*:

(111) *category*	HEAD	*head*
	SUBCAT	*list(synsem)*
	MARKING	*marking*

It is thus clear what the purpose of the limited use of parametric sorts in HPSG 94 is. The remaining question is how we can express parametric sorts in RSRL.

Penn 1998 investigates the meaning of parametric sorts in feature logics, and defines what is called *parametric sort hierarchies* for the attribute-value logic of Carpenter 1992. The main insight of Penn 1998 is that parametric sorts are not sorts but functions that provide access to a set of sorts as determined by the sorts that are the arguments of these functions (the parameters). In Penn's approach, parametric sorts of the very restricted kind that is employed by HPSG 94 ultimately induce a sort hierarchy without parametric sorts. Spelling out the details of the construction is slightly complicated by the fact that it must be ensured that the appropriateness of attributes and attribute values is respected in the desired way. Moreover, Penn's parametric sort hierarchies are much more general than what is needed to treat the parametric sorts of HPSG 94. In the present discussion, the technical details are

[32] Note that this does not imply that the members of a list must be of the same species, because it is not required that σ be a maximally specific sort. In fact, since σ may be the sort *object*, the members of some lists can, in principle, be of any species.

of no immediate relevance, and it is only important to note that, in principle, we could augment RSRL by a definition of parametric sort hierarchies.

However, since RSRL already augments basic feature logics with powerful new constructs, relations and bounded quantification, it seems to be reasonable to ask whether another augmentation by parametric sort hierarchies is really necessary in order to interpret what linguists want to express with parametric lists. We could even ask the more fundamental question if parametric lists are necessary at all in existing theoretical HPSG grammars. Recall that so far I have avoided the issue of parametric lists by permitting entities of any sort on any list in the models of grammars, at least as far as the restrictions on the members of lists that are expressed in the signature are concerned. In (78) above, I have adopted a simple sort hierarchy and a definition of the appropriateness function for the sort *list* and its two subsorts *elist* and *nelist* for the grammar of Pollard and Sag that I repeat here for convenience:

(112) *list*

elist		
nelist	FIRST	*object*
	REST	*list*

The sort *object* is a supersort of all sorts in Pollard and Sag's sort hierarchy. If we replace every appropriateness specification in the signature of Pollard and Sag 1994 in which a parametric list is the appropriate value of some pair of a sort and an attribute with a specification that is the same as the original except that the sort *list* is appropriate for the respective pair, then the new signature obviously permits more structurally distinct interpretations. For example, based on the modification of the signature, it would be possible that SUBCAT lists of signs suddenly contain not only *synsem* entities, as the original appropriateness specification with a parametric list that is illustrated in (111) requires for the SUBCAT attribute, but entities of any other sort as well.

A look at the structure of grammars, however, shows that this effect can hardly arise in models of actual grammars. The possible sorts of members of lists in signs are usually determined by the members of the corresponding lists of the words that make up the sign. And the members of the lists in words are given in lexical specifications, which determine the sorts of the members of the lists. To put it differently, if the members of the SUBCAT list of each word in the grammar are *synsem* entities, and if the grammar determines that the SUBCAT list of each sign is by some operation composed only of entities on the SUBCAT list of its daughters, then the SUBCAT list of each sign can only contain *synsem* entities; and similarly for other lists (and sets) in HPSG grammars. In this situation, parametric lists in signatures provide a redundant specification, and using the general list specification of (112) would achieve the same result as parametric lists as far as the models of the grammars are concerned.

There are cases, however, in which linguists want to put stronger restrictions on the permitted sorts of elements of lists than what the principles of the grammar require independently. An example in the grammar of Pollard and Sag 1994 are the lists of complement daughters, which the signature with parametric sorts requires to be lists of phrases. Another principle of the grammar that restricts the list of complement daughters in the relevant respect is the SUBCATEGORIZATION PRINCIPLE.[33] It relates the *synsem* entities on the SUBCAT list of the head daughter of a head complement structure to the SYNSEM values of its complement daughters, thus implying that the complement daughters are signs, i.e., words or phrases. This is almost the same as what the parametric list specification requires. The exclusion of words as potential complement daughters, however, comes from from the parametric list specification of complement daughters alone.[34] If we do not have parametric sort hierarchies to express this restriction, is is thus necessary to impose it somewhere else in the grammar.

This brings me back to the original question: Can the parametric lists of HPSG 94 be expressed in RSRL without the introduction of parametric sort hierarchies? In fact, it is trivial to do so with relations. As we have seen, the idea of parametric lists is that the members of the list value of certain attributes are all of a certain sort. Instead of expressing the different kinds of lists with separate sort names for each kind of list, a common name (*list*) is kept and augmented with parameters, which is supposed to indicate that

[33] See (45) and (47).

[34] As the grammar of Pollard and Sag 1994 stands, restricting complement daughters to phrases is actually problematic. Pronouns like *she* and proper names like *Kim* can be complement daughters. If words are not permitted as complement daughters, *she* and *Kim* must first project to phrase level before they can be constructed as complement daughters. Since their SUBCAT lists are empty, there is no ID schema in the grammar that permits the necessary unary projection. Dropping the phrase restriction is one possible solution to the problem.

there is a generalization over related kinds of structures. They are all lists. For the limited use that HPSG 94 makes of parametric sorts, the same result can be obtained in a slightly different way.

Suppose that the sort hierarchy under the sort *list* and the respective appropriateness specifications of (112) are part of the signature of a grammar. For every parametric list with parameter σ (where σ is a sort symbol), I introduce a relation symbol `list-of-`σ and define the unary relation `list-of-`σ that holds of all component lists of all entities in a model of the grammar that only contain elements of sort σ. For lists of *synsem* entities, we get the following definition:

(113) $\texttt{list-of-synsems}(x) \stackrel{\forall}{\Longleftarrow}$
$\quad {}^{x}[\textit{elist}]$

$\texttt{list-of-synsems}(x) \stackrel{\forall}{\Longleftarrow}$
$\quad x\,\langle \textit{synsem} | \boxed{1} \rangle \land \texttt{list-of-synsems}(\boxed{1})$

Secondly, for each (pseudo-) appropriateness specification of the form $\mathcal{F}\langle \sigma_1, \alpha\rangle = \textit{list}(\sigma_2)$, where σ_1 and σ_2 are sort symbols and α is an attribute name, we let the sort *list* be appropriate for $\mathcal{F}\langle \sigma_1, \alpha\rangle$ instead, and add the description $\sigma_1 \rightarrow ([\alpha\,\boxed{1}] \land \texttt{list-of-}\sigma_2(\boxed{1}))$ to the theory of the grammar. For example, to restrict the list values of SUBCAT lists to *synsem* entities, as intended with the parametric list specification *list(synsem)* in (111), I add the description (114) to the theory of the grammar of Pollard and Sag:

(114) $\textit{category} \rightarrow [\textsc{subcat}\,\boxed{1}] \land \texttt{list-of-synsems}(\boxed{1})$

It is not hard to see that each signature with a parametric sort hierarchy for lists (and sets) that is limited in the way Pollard and Sag indicate can be translated into an RSRL signature with general lists as given in (112), a number of relations of the form `list-of-`σ for the parameters, σ, and principles that restrict the possible members of the list (or set) values of certain attributes just as the interpretation of the parametric sort hierarchy in the sense of Penn would do. The notation for parametric lists and sets of HPSG 94 can, therefore, be viewed as a convenient shorthand for these relations and principles in an RSRL grammar.

Treating the limited parametric polymorphism of HPSG 94 in the way just sketched has the additional advantage that the technique generalizes easily to other habits in the literature that cannot be interpreted as uses of parametric sorts but seem to be based on the same intuitions of the grammarians as their use of parametric lists. With parametric lists, linguists restrict the possible elements of certain lists to entities of some sort. But there are similar conditions on lists according to which all members of a list must satisfy some complex description. For example, in an analysis of the Polish reflexive marker, *się*, Kupść (1999, p. 110) assumes that *się* is subcategorized for by a specialized valence attribute. As a consequence, she wants to exclude that *się* can be subcategorized for by the standard valence attribute for complements, COMPS. Therefore, she requires that no element of any COMPS list satisfy her description of anaphoric clitics. Using Kupść's description of anaphoric clitics and tacitly presupposing her signature, which differs slightly from the signature of Pollard and Sag 1994, this condition can be formalized with a unary relation, `list-of-no-ana-cl`, defined in (115a), and a principle based on that relation that restricts the possible members of COMPS lists (115b):

(115) a. $\texttt{list-of-no-ana-cl}(x) \stackrel{\forall}{\Longleftarrow}$

$$^{x}\left[\textit{elist}\right]$$

$\texttt{list-of-no-ana-cl}(x) \stackrel{\forall}{\Longleftarrow}$

$$x\left\langle \neg \begin{bmatrix} \textit{cl-synsem} \\ \text{LOC} \begin{bmatrix} \textit{local} \\ \text{CAT HEAD} \;\; \textit{noun} \\ \text{CONTENT} \;\; \textit{ana} \end{bmatrix} \end{bmatrix} \middle| \boxed{1} \right\rangle \land \texttt{list-of-no-ana-cl}(\boxed{1})$$

b. $\textit{val} \rightarrow \left[\text{COMPS}\,\boxed{1}\right] \land \texttt{list-of-no-ana-cl}(\boxed{1})$

The definition of the relation and of the principle that express Kupść's restriction on COMPS lists of Polish are analogous to Pollard and Sag's restriction which says that SUBCAT lists are lists of *synsem* entities as formalized in (113) and (114). While complex restrictions on the members of lists of the kind illustrated in (115) cannot be interpreted as a parametric list specification, they fit into the schema of a relational specification of parametric lists.

4.5.3 The Lexicon and Lexical Rules

Pollard and Sag 1994 does not say much about the lexicon and about the integration of the lexicon with the rest of the architecture of the grammar, and the appendix of the book does not contain a specification of a lexicon nor examples of lexical entries. The appendix also omits the three lexical rules that are contained in the book, because the authors do not expect that their formalization is possible in the formalism that they envisage. They assume that for lexical rules a higher-order formalism is needed that takes the expressions of the formal languages of the envisaged formalism as its object language (Pollard and Sag, 1994, p. 395, fn. 1). Insofar as the purpose of the present chapter is to justify RSRL as an adequate formalism for HPSG 94 in which the grammar of the appendix of Pollard and Sag 1994 can be fully formalized, problems of the lexicon and lexical rules could, strictly speaking, be ignored here, because they are not part of the grammar that Pollard and Sag specify explicitly. However, HPSG is often called a lexicalist framework for linguistic theories, and since the publication of Pollard and Sag's book the question of the status of the lexicon and of lexical rules in HPSG 94 has received satisfactory answers. Moreover, HPSG grammars rely heavily on rich lexical structures, and the lexicon plays a prominent role in many recent HPSG analyses.

For these reasons, this section very briefly indicates where the lexicon is located in the architecture of HPSG 94 grammars. It is only intended to complete the picture of how to formalize an HPSG 94 grammar in RSRL, and it is deliberately restricted to a very short summary of how the lexicon of constraint-based HPSG can be specified. The task of the present section is not to discuss any of the complex issues surrounding the various possibilities of organizing lexical specifications, nor to discuss how lexical generalizations can be expressed best. The most comprehensive discussion of lexical specifications in HPSG 94 is contained in "Part II, Lexical Generalizations in HPSG", of Meurers 2000, pp. 101–165, and the interested reader is referred there and to the literature cited therein for a comprehensive discussion of the lexicon in HPSG 94. The HPSG formalism that Meurers 2000 uses in the discussion of the lexicon and of lexical generalizations is SRL, which makes its presentation fully compatible with the present formalism.

According to the signature of Pollard and Sag 1994, there are two kinds of signs: Phrases, which are of sort *phrase*, and words, which are of sort *word*. Obviously, the lexicon of a grammar must express restrictions on the possible configurations under *word* entities in models of the grammar. As long as the theory of the grammar does not contain such restrictions, any configuration under *word* entities that is an interpretation of the signature of the grammar is in an exhaustive model of the grammar, as long as none of its components violates any other description in the theory of the grammar.[35] This means that many configurations are in the models of the grammar that are not actual words of the language. The basic task of *lexical entries* is to impose the proper restrictions on permissible words.

The standard way of integrating lexical entries with an HPSG 94 grammar is the formulation of a so-called WORD PRINCIPLE as an element in the theory of the grammar. Just as any other description in the theory of a grammar, the WORD PRINCIPLE is an implication. Its antecedent is a description of all words, and its consequent is a disjunctive enumeration of the lexical entries:

(116) *The logical structure of the* WORD PRINCIPLE

$$word \rightarrow (LE_1 \vee LE_2 \vee \ldots \vee LE_n)$$

The meta-variables LE_1 to LE_n stand for the actual lexical entries in the grammar, which are descriptions of the words that the grammar contains. Given the WORD PRINCIPLE, every word in the models of the grammar must be described by at least one lexical entry. The set of descriptions LE_1 to LE_n is the lexicon. Note that in this formulation the lexicon is necessarily finite, since the formal languages of RSRL do not allow infinitary disjunctions. Nevertheless, depending on the formulation of the lexical entries and on the structure of words, there might still be infinitely many non-isomorphic words that are described by the lexical entries.

In principle, the descriptions of words in the lexical entries can be left very general. For example, the lexical entry of a verb might only mention the phonology of the verb stem, its part of speech, the lexical semantics of the word, and its syntactic argument structure. Such an impoverished lexical entry in the consequent of the WORD PRINCIPLE would of course license many structural variants of the verb in question that should not be grammatical, and there must be other principles in the grammar that exclude them. As Meurers (2000, pp. 109ff.) explains, *lexical principles* can be used to further constrain the admissible words in the denotation of lexical entries. Lexical principles are principles that constrain the structure of words by generalizing over word classes. In our example of a hypothetical, very general

[35] For example, the psoa components of words must obey the QUANTIFIER BINDING CONDITION, formalized in (108); and the anaphors, pronouns, and nonpronouns on their SUBCAT lists are restricted by the BINDING THEORY, formalized in (55).

lexical entry of a verb, lexical principles might be needed in the theory of the grammar that make sure that for all finite verbs in the denotation of the lexical entry (and of all other lexical entries that describe finite verbs), the case of its subject is *nominative*; and other lexical principles might have to ensure the correct behavior of subject-verb agreement of verbs.

This leaves the question of how to formalize lexical rules. Two views of lexical rules can be distinguished. The first view holds that lexical rules express a relationship between lexical entries, and is the view of Pollard and Sag 1994. Under this view the grammarian specifies a finite set of base lexical entries and a finite set of binary relations between expressions of the logical language in which the base lexical entries are expressed. The lexicon is then defined as the closure of the lexical entries under the defined set of binary relations, the lexical rules. Lexical rules of this kind are also known as meta-level lexical rules (MLRs), because they are not defined in the logical language in which the grammar is specified, but in a meta-language that talks about the language of the grammar. Therefore, this view of lexical rules places them outside of RSRL.

The MLR perspective still awaits a formalization in HPSG 94, and for the time being it remains unclear if it is compatible with RSRL. The closure of lexical entries under MLRs might yield an infinite lexicon, which either necessitates an extension of the logical languages of the formalism to infinitary languages in order to formulate a Word Principle with infinitely many lexical entries, or a modification of the notion of a grammar. For example, we could define a grammar as a triple consisting of a signature Σ, a set of Σ descriptions (the theory of the grammar), and a (potentially infinite) set of lexical entries (the lexicon) that is the closure of the base lexical entries under the lexical rules. A Σ interpretation would be a model of the grammar exactly if each entity in it satisfies each description of the theory, and each entity of sort *word* satisfies at least one lexical entry.

The second view of lexical rules holds that lexical rules express relations between word entities. It says that if the model of a grammar contains words of a certain structure then it must also contain corresponding words of a given different structure. This interpretation of lexical rules is known as description-level lexical rules (DLRs), because it puts lexical rules on the same level with all other principles of the grammar: Lexical rules as DLRs become part of (descriptions in) the theory of the grammar. A formalization in SRL of the DLR approach to lexical rules is put forth and discussed by Meurers (2000, pp. 123ff.). The basic idea is to introduce entities of sort *lexical_rule* into the grammar with a *word*-valued IN attribute (the input of the lexical rule) and a *word*-valued OUT attribute (the output of the lexical rule). In the ordinary description language of all principles the linguist describes which properties the input words share with the output words, and where they differ. The WORD PRINCIPLE is extended by exactly one disjunct which adds the possibility that a word in a model of the grammar may also be described by the OUT value of a *lexical_rule* entity.[36] To enhance the practical usability of DLRs, Meurers provides a lexical rule specification language that essentially takes lexical rules as they are usually written by linguists and translates them into descriptions of *lexical_rule* entities.

As Meurers 2000 argues at length, DLRs achieve exactly the effect that linguists have in mind when they specify lexical rules, and their formalization in SRL helps to clarify open questions about ambiguities in the notation that linguists use in their notation of lexical rules. Meurers and Minnen 1997 describes a computational implementation of DLRs.

With the formalization of DLRs in SRL, there exists an interpretation of lexical rules that is compatible with RSRL and allows an interpretation of the three lexical rules of Pollard and Sag 1994 in the present formalism for linguistic theories. I conclude that RSRL indeed allows a complete formal specification of all aspects of HPSG 94, including the lexicon and lexical rules.

[36]To make this specification possible, the *lexical_rule* entity must be a component of the word that is the output of the lexical rule. For the technical details, see Meurers 2000, p. 124.

Chapter 5

Beyond Classical HPSG

Like most grammars in theoretical linguistics, the grammar of Pollard and Sag 1994 was not written on the basis of a complete and explicit mathematical formalism. To a considerable extent, the formulation of the grammar was guided by existing feature logics, but where their expressive means did not coincide with linguistically motivated needs or where the authors felt that a strict adherence to mathematically more precise potential formulations would compromise linguistic perspicuity, Pollard and Sag did not hesitate to go beyond what was formalized at the time and relied on intuitions about possible extensions of existing formalisms. Part of the motivation underlying the design of RSRL was to construct a custom-made formalism that directly supports the intuitions behind Pollard and Sag's implicit extensions of traditional feature logics, and makes possible a rigorous formulation of HPSG 94 grammars without loss of linguistic perspicuity.

In the course of making the underlying intuitions explicit, RSRL introduced mathematical constructs that were not known to play a role in HPSG before, such as generalized component quantification and chains. Relations and relational negation received a semantics that differs from their semantics in computational treatments of feature logics. On the one hand, these extensions make it possible to fully formalize the grammar of English of the appendix of Pollard and Sag 1994; and, as I have argued in Chapter 4, all new constructs are necessary to do so. On the other hand, a systematic application of the extensions of feature logics that become available with RSRL is likely to permit the formulation of new kinds of HPSG grammars. These grammars might contain linguistic principles whose form would have made them dubious candidates for HPSG principles before the formulation of

RSRL, because it would not have been clear whether they can be expressed in the proposed description languages for the HPSG framework. A systematic use of component quantification offers possibilities that did not seem to be available before. However, according to our present knowledge, they must be available in HPSG, since without component quantification there is no known mathematical explication of traditional HPSG grammars.

In the present chapter, I give a very brief summary of recent research in HPSG that made use of RSRL. As in Chapter 4, the focus is not on a discussion of the specifically linguistic contributions of this research. Although empirical questions are at the center of the original research reported here, I will ignore them as much as possible. In the present context, my interest in this work is on a more abstract, conceptual level that is largely independent of its empirical content. My goal is to obtain a first impression of what kind of new analytical perspectives, if any, open up to linguistic analyses in HPSG with RSRL. At the same time, one should keep in mind that linguistic research that is formalized in RSRL could ultimately allow for an evaluation of whether the high expectations regarding mathematically rigorous formalisms for linguistic theories from which the development of RSRL drew a significant portion of its initial motivation are really justified. Taking HPSG 94 and RSRL as a specific instance of the general question, it should become possible to evaluate whether this formalism for a pre-existing linguistic theory really broadens the perspective of practical linguistic research in that specific framework, and whether it allows the natural, fruitful, and straightforward expression of grammatical principles beyond those that are contained in the early literature of HPSG 94. While it is certainly too early to draw final conclusions, we can already get a first impression of how the formalism influences the formulation of linguistic principles.

5.1 Overview

In this section I give a short characterization of RSRL-based analyses in HPSG, and I briefly indicate what kinds of principles these analyses use. In Sections 5.2, 5.3 and 5.4, we will investigate them in more detail.

The empirical domain of Meurers 2000 is the syntax of non-finite constructions in German. To a large extent, the formalization of its HPSG analysis consists in lexical specifications, in implicational descriptions whose antecedents can be formalized in SRL, and in revisions of equally straightforward traditional principles from the literature. These grammatical specifications are all given in the familiar AVM syntax that can by and large be directly interpreted as the AVM syntax of RSRL if we suppose the notational conventions that I have introduced in Section 3.2.3.[1] That means that, for the most part, Meurers 2000 is formally very conservative and does not even need to refer to quantification explicitly, because all instances of quantification are taken care of by the notational conventions of existential closure and the clause notation for relations. An interesting exception to this formally conservative approach are Meurers's principles of structural case assignment, which are formulated and formalized in a way that requires explicit component quantification. In Section 5.2, we will discuss the formal structure of these case principles.

The situation that we find in Przepiórkowski 1999a is different. Like Meurers 2000, it is mostly concerned with syntax. Przepiórkowski 1999a proposes a comprehensive, non-configurational theory of case assignment in Polish and, building on the application of a wide range of empirical tests and on previous theoretical work in HPSG, challenges the widespread view that there is a configurational distinction between adjuncts and complements. The analytical results support the conclusion that tree-configurationality is less important in grammars than is often assumed. An appendix (Przepiórkowski, 1999a, pp. 417–442) formalizes the main aspects of the HPSG analysis in RSRL. Since the formalization is not written in an AVM syntax but in the original syntax of RSRL, quantification is always explicit, except for the clause notation of the definition of relations. Moreover, the descriptions are usually formally complex, using relations in the antecedents of descriptions and exhibiting a rich interaction between existential quantification, universal quantification, negation and relational formulae. The form of these principles is reminiscent of the RSRL formalization of the TRACE PRINCIPLE or the BINDING THEORY of Pollard and Sag 1994.[2] It is interesting to note that in Pollard and Sag 1994, principles of this formal complexity are not sketched in the traditional AVM notation, which would be poorly suited to indicate the necessary quantification. They are given as precise verbal descriptions.

[1] The only exception is the functional notation for relations, which Meurers 2000 uses throughout, even in the clausal definitions of the meanings of relations. For a reader who is familiar with the functional clause notation of Meurers's definitions of relations and with the relational clause notation of RSRL, it is not hard to translate the syntax of one into the syntax of the other.

[2] See (168) in Appendix C and (55) in Section 4.2, respectively.

Przepiórkowski 1997, 1998 present self-contained, slight variants of parts of the analysis of Przepiórkowski 1999a. They focus on problems of a theory of quantifier scope at the syntax-semantics interface. They are a response to a paper by Pollard and Yoo 1998 that fixes problems of the theory of quantifier scope originally proposed in Pollard and Sag 1994. While Pollard and Yoo 1998 solves the problems with a theory of phrasal quantifier retrieval, Przepiórkowski proposes a theory of lexical quantifier retrieval, and argues that the resulting theory needs fewer principles, is conceptually simpler, and avoids spurious ambiguities that occur in Pollard and Yoo's theory. In addition, Przepiórkowski shows that his analysis is compatible with an adjuncts-as-complements analysis and with different theory-internal assumptions about unbounded dependency extraction. Just as in Przepiórkowski 1999a, the formalization of the analysis is presented in the original RSRL syntax. Representative principles from the work of Przepiórkowski will be discussed together with the theory of case assignment of Meurers in Section 5.2.

Penn 1999a,b,c reconsider the structure of previous linearization-based approaches to syntax as proposed by Reape 1994a and Kathol 1995. In an analysis of second-position clitics in Serbo-Croatian, Penn argues that the domain objects that are used by Reape and Kathol are too closely related to syntactic structures to be able to express the interaction of constraints from different modules of the grammar, namely from syntax, prosody, and discourse, on linear structure in Serbo-Croatian. Penn's own analysis assumes a strict separation of linear or phenogrammatical structure and combinatory syntactic or tectogrammatical structure. Following the proposal by Kathol 1995, the realization of linear structure is based on a model of topological fields. Topological fields are defined relative to regions in a structure, which allows for multiple layers of regions with their own relative topological fields. The RSRL formalization of the central topological principles of the analysis can be found in Penn 1999a, pp. 132–135.[3] Section 5.3 will give an overview of the kinds of principles that occur in Penn's linearization theory.

The topic of Richter and Sailer 1999a,b is the syntax and semantics of negative polarity items in French and in Polish, respectively. The goal of the papers is to determine the logical form of various lexical elements that may participate in negative concord constructions, and to formulate the mu-

[3] See also Penn 1999b, pp. 192–196, which adds explicit definitions of the relations used in the topological principles.

tually contingent conditions on syntax and on logical form that determine the characteristic distribution of these elements. The technically interesting aspect of the analyses is the definition of a semantic representation language, Ty2 (Gallin, 1975), as logical forms of signs. Ty2 is one of the standard languages of natural language semantics, and its integration makes methods of model-theoretic semantics available to HPSG grammars. The appendix of Richter and Sailer 1999a, pp. 282–295, contains the specification in RSRL of principles that define structures in the denotation of the grammar that can be regarded as the expressions of Ty2. Sailer 2000 introduces important modifications and proves the relevant properties of the models of the grammar of Ty2. In Section 5.4, we will discuss the necessary principles.

As this brief overview shows, although there is only a small number of RSRL-based analyses, they are concerned with different languages and with diverse empirical phenomena in different areas of linguistics. We may thus hope that the principles that they formulate give a realistic idea of what to expect from fully explicit RSRL grammars of natural languages.

5.2 Meurers 2000 and Przepiórkowski 1999a

In this section, I discuss some formal aspects of the case theory of Meurers 2000 and Przepiórkowski 1999a, and of the theory of quantifier retrieval of Przepiórkowski 1999a. As we will see, these analyses of distinct empirical domains share the same technique of making information about configurations of entities at some "unknown" location in linguistic structures available to lexical principles, i.e., to principles that restrict admissible configurations of entities under *words*. Their common technique crucially depends on the availability of component quantification in the formalism of HPSG.

A historical note on the case theory that is discussed in Section 5.2.1 might be in order, since the publication dates of the referenced literature do not reflect the chronology of its formulation, and since my discussion of its logical aspects does not reveal the mutual influences. The essential ideas underlying the non-configurational case theory in HPSG are due to Adam Przepiórkowski. A fully developed version of Przepiórkowski's non-configurational case theory was first published in Przepiórkowski 1999b. Meurers 1999 refines this original theory in an application to German data; and Meurers 2000 contains essentially the same analysis of case assignment in German as Meurers 1999 with minor differences in formulation. Przepiórkowski 1999a finally modifies his own earlier formulation of the case theory to be able to account for Meurers's data as well. For reasons of exposition, I present the case assignment principles of Meurers 2000 before discussing the (revised) case theory of Przepiórkowski 1999a, although Przepiórkowski's case theory preceded the case principles of Meurers.

5.2.1 Case Theory

Meurers 2000 proposes a lexical theory of structural case assignment. The main idea of the analysis is that case assignment is not determined by tree-configurational notions as in many other theories of case, i.e., case assignment does not refer to the syntactic tree structure in which an argument is realized. Instead, case is assigned lexically, using the rich structure of words to formulate the right lexical generalizations. As a consequence, case assignment is local in a very strict sense, because—metaphorically speaking—it happens inside words. However, there is an interesting condition on this local case assignment that goes beyond the word level. From a discussion of complex German data, Meurers 2000 concludes that a word only assigns case to those of its arguments that are not raised to a higher predicate. More specifically, if an auxiliary like *haben* (*have*) raises the arguments from a subcategorized verb like *kaufen* (*to buy*) in a verb cluster,[4] then *kaufen* does not assign (structural) case to the raised arguments. Instead, the verb that inherits the arguments (here: *haben*) becomes responsible for case assignment, unless there is another, even higher argument raiser that inherits the arguments again. In other words, it is always the highest verb that contains the argument on its SUBCAT list that assigns structural case to that argument in an argument raising construction.

Which arguments of a word are raised does not depend on the word itself but on the particular syntactic structure in which (an instance of) the word occurs. It is not a property of the word. The properties of the predicate that selects the word in a syntactic structure determine whether argument raising occurs. The interesting question is how a lexical theory of case assignment gets access to the necessary information about argument raising

[4]See the sketch of a lexical entry of the argument raising verb *haben* in (36) on page 229. Its SUBCAT list, $\boxed{2}$, contains the SYNSEM value of the subcategorized verb as the most oblique element, and it raises all elements of the SUBCAT list, $\boxed{1}$, of that verb by requiring that its own SUBCAT list is the concatenation of $\boxed{1}$ and a list, $\boxed{4}$, that contains exactly the SYNSEM value of the subcategorized verb.

that determines which words must assign case to which arguments. Before Meurers formalizes his two case principles of nominative case assignment and accusative case assignment, he gives a description in natural language that reveals the underlying idea:[5]

(117) *Nominative Case Assignment (Meurers, 2000, p. 323)*

> In an utterance, the least oblique subcategorization requirement with structural case of each verb which is not raised from that verb receives nominative case.

In Meurers's theory, the assignment of structural case in words is relativized with respect to the utterances in which the words occur. For every particular utterance, the theory of argument raising determines which arguments of which words are raised, and by which words they are raised. Relative to a given utterance, it is thus clear which verb of a verb cluster must assign structural case to which argument. Note that the nominative case assignment in (117) is only concerned with structural case, which must be distinguished from idiosyncratic case.[6] Idiosyncratic case cannot be inferred from grammatical regularities but must be specified individually for a particular word. In HPSG grammars, the distinction between lexical and structural case is traditionally realized with different sorts in a sort hierarchy under *case*. For example, one might posit several distinct sorts for accusative case, the speciate sorts structural accusative (*s-acc*) and lexical accusative (*l-acc*), and a common supersort of them, *acc*. In addition, structural cases are subsorts of the sort *struc(tural case)*; lexical cases are subsorts of the sort *lex(ical case)*.[7]

How does Meurers 2000 formalize the case principle in (117)? Obviously, the crucial concept is the concept of *utterances*. As soon as we know what utterances are, we can use component quantification to impose the conditions expressed in (117) on all words occurring in utterances. In a discussion of German sentence structure, Richter 1997, pp. 134ff., introduces a partitioning of signs into *unembedded-signs* and *embedded-signs*, which is motivated by syntactic, semantic and pragmatic reasons. The sort *unembedded-sign* denotes

[5]I omit the principle of accusative case assignment. Its logical structure is parallel to nominative case assignment. It differs only with respect to the obliqueness of the element to which case is assigned.

[6]Idiosyncratic case is sometimes called lexical case. I use both terminologies.

[7]See Przepiórkowski 1999a, p. 424, for such a case hierarchy for a grammar of Polish.

signs that occur as independent utterances with illocutionary force. With an attribute ILLOCUTION appropriate to *unembedded-sign*, it constitutes the interface to pragmatics. By contrast, *embedded-signs* are those signs that occur as syntactic daughters. Embedded signs lack an illocutionary force of their own.

Adopting the idea of formalizing utterances as unembedded signs, it is not hard to express nominative case assignment with universal component quantification. In a first approximation, Meurers 2000, p. 325, says that for each verb, w, in an unembedded sign, if the most oblique argument, a, of the verb bears structural case, *struc*, and there is no SUBCAT list in the unembedded sign on which both the SYNSEM value of a projection of w (or the SYNSEM value of w) and a *synsem* entity that represents a occur, i.e., if the argument a is not raised from w to a higher functor in the utterance, then a is assigned nominative case (*nom*). For technical reasons having to do with details of the analysis, Meurers 2000 identifies the argument, a, on the SUBCAT list of the *category* entity, $\boxed{5}$, (the CATEGORY value of the hypothetical argument raiser) by its HEAD value, $\boxed{4}$, and projections of the verb by its HEAD value, $\boxed{2}$.[8]

(118) *Nominative Case Assignment formalized (after Meurers 2000, p. 325)*

$$
\forall\boxed{1}\ \forall\boxed{2}\ \forall\boxed{3}\ \forall\boxed{4}
\left(
\begin{array}{l}
\left(
\begin{array}{l}
\textit{unembedded-sign} \wedge \\
\boxed{1}\begin{bmatrix} \textit{word} \\ \textsc{ss loc cat} \begin{bmatrix} \textsc{head} & \boxed{2}\ \textit{verb} \\ \textsc{subcat} & \left\langle \boxed{3}\begin{bmatrix}\textsc{loc cat head}\ \boxed{4}\begin{bmatrix}\textsc{case}\ \textit{struc}\end{bmatrix}\end{bmatrix} \middle| \textit{list} \right\rangle \end{bmatrix} \end{bmatrix} \wedge \\
\neg\ \exists\boxed{5}\ \exists\boxed{6}\ \exists\boxed{7}\ \exists\boxed{8}
\left(
\begin{array}{l}
\boxed{5}\begin{bmatrix}\textsc{subcat}\ \boxed{6}\end{bmatrix} \wedge \\
\mathtt{member}\left(\boxed{7}\begin{bmatrix}\textsc{loc cat head}\ \boxed{4}\end{bmatrix}, \boxed{6}\right) \wedge \\
\mathtt{member}\left(\boxed{8}\begin{bmatrix}\textsc{loc cat head}\ \boxed{2}\end{bmatrix}, \boxed{6}\right)
\end{array}
\right)
\end{array}
\right) \\
\rightarrow\ \boxed{3}\begin{bmatrix}\textsc{loc cat head case}\ \textit{nom}\end{bmatrix}
\end{array}
\right)
$$

The word $\boxed{1}$ is the word whose nominative case assignment to its most oblique argument, $\boxed{3}$, is constrained, if the most oblique argument bears structural

[8] As Meurers 2000, p. 326, fn. 42, notes, the possibility of identifying projections of words by unique HEAD values cannot be taken for granted; it requires additional principles in the grammar. The most natural way to formalize them is as conditions on the distribution of identical HEAD values in unembedded signs.

case.[9] $\boxed{5}$ is the CATEGORY value of an argument raiser that raises $\boxed{3}$ (identified by its HEAD value, $\boxed{4}$) from (a projection of) $\boxed{1}$ (identified by its HEAD value, $\boxed{2}$). If no $\boxed{5}$ with the described argument raising configuration on its SUBCAT list exists in the utterance then $\boxed{3}$ receives nominative case.

The idea of using component quantification over unembedded signs is a typical application of RSRL-specific technical means to formalize a linguistic generalization. In the case assignment principle in (118), Meurers deliberately uses component quantification in order to avoid a conceivable alternative encoding of the principle that would involve a Boolean-valued book-keeping attribute for marking an argument on a SUBCAT list as being raised. Component quantification thus helps to avoid what Meurers views as a merely technical attribute without a linguistic interpretation. The intuition is to use attributes only to express genuinely linguistic concepts. Moreover, the formalization in (118) is more faithful to the original, verbal formulation of the principle in (117) than a formalization with an additional book-keeping device would be.

The final formulation of Przepiórkowski's (1999a) non-configurational case theory is in many respects similar to the two case principles of Meurers 2000, but there are also interesting formal differences. The basic idea is the same: structural case is assigned to arguments that are not raised to a higher predicate. One of Przepiórkowski's goals is to formulate a case theory whose principles reflect the idea that case assignment is strictly local. Another important consideration is to build a modular case theory that is compatible with as many independently motivated assumptions in the current HPSG literature as possible. For example, the case theory is carefully constructed in a way that makes it compatible with a classical theory of unbounded dependencies that posits traces and with theories of extraction without traces; and it can be combined with current HPSG analyses of clitics as affixes in Romance languages. The architectural considerations lead Przepiórkowski to split the part of the case theory that determines which arguments of a predicate are raised in an utterance from the case principles. This separation into distinct principles implies that arguments must be explicitly marked in their structures as being raised or not, so the case principles can refer to

[9]Note that at this point the formalization differs slightly from the informal description of the principle in (117). In (117), the principle talks about the least oblique subcategorization requirement with structural case of verbs. The formalization only restricts the least oblique subcategorization requirement of verbs if it happens to have structural case. Thanks are due to Detmar Meurers and Tilman Höhle for bringing this to my attention.

arguments that are not raised in their antecedents.

Before we look at two of Przepiórkowski's principles, I need to introduce some background information about how the signature of the grammar of Polish of Przepiórkowski 1999a differs from the signature of the appendix to Pollard and Sag 1994. Instead of letting predicates select all arguments with a single SUBCAT list, Przepiórkowski follows the proposal of Chapter 9 of Pollard and Sag 1994 and splits the subcategorization frame into three different lists. They are the values of the new attributes SUBJECT, SPECIFIER, and COMPLEMENTS, which are appropriate to the sort *valence*, which in turn is the value of the new attribute VALENCE that replaces the SUBCAT attribute as one of the attributes that are appropriate to *category*. In addition, the members of the three valence lists are not of sort *synsem*. They are of sort *argument*, to which two attributes are appropriate: First, there is the attribute ARGUMENT, which is *synsem*-valued, and second, there is the Boolean-valued attribute RAISED. The purpose of the sort *argument* and the attribute RAISED is to provide a place in the structure of the syntactic argument representation where the arguments can be marked as being raised or not being raised. Finally, in Przepiórkowski's signature the attribute ARG-ST, which essentially replaces the former SUBCAT list for principles that talk about all syntactic arguments of predicates like the BINDING THEORY, is appropriate to *head*. The ARG-ST list encodes the argument structure of a word. In a first approximation, its value can be thought of as the concatenation of the SUBJECT list, the SPECIFIER list, and the COMPLEMENTS list, in that order.[10] (119) summarizes the modifications to the signature:[11]

(119) a. *argument* ARGUMENT *synsem*
RAISED *boolean*

b. *category* HEAD *head*
VALENCE *valence*

c. *head* ARG-ST *list(argument)*

[10] There are various, systematic exceptions to this basic relationship between the valence lists and the ARG-ST list, which do not need to concern us here. See Przepiórkowski 1999a, pp. 24ff., for a brief overview of the pertinent issues.

[11] Since RSRL does not allow parametric sort hierarchies, the parametric list specifications must be understood as an abbriviatory notation for simple list specifications and appropriate descriptions in the theory of the grammar as explained in Section 4.5.2. I keep the parametric sort notation for lists as a convenient shorthand.

d. $\begin{array}{lll} \textit{valence} & \text{SUBJECT} & \textit{list(argument)} \\ & \text{SPECIFIER} & \textit{list(argument)} \\ & \text{COMPLEMENTS} & \textit{list(argument)} \end{array}$

With the new structure of signs that follows from (119), the RAISED Principle in (120) fixes the RAISED value of *argument* entities on ARG-ST lists in utterances, depending on whether or not the arguments are raised from their ARG-ST list. The value '+' means that an argument is raised from its ARG-ST list; the value '−' means that it is not raised. The case principles express conditions on arguments with the RAISED value '−'. Despite the differences, the formal and conceptual relationship between the RAISED Principle and the nominative case assignment, (118), are quite transparent. While, metaphorically speaking, (118) assigns nominative case to the least oblique element on a SUBCAT list in an utterance exactly if it is not raised, (120) requires that the RAISED value of an *argument* entity on an ARG-ST list in an utterance be '+' exactly if it is raised to a higher ARG-ST list, and '−' otherwise:[12]

(120) *The* RAISED *Principle (after Przepiórkowski 1999a, p. 424)*

$$
\begin{array}{l}
\textit{unembedded-sign} \rightarrow \\
\forall\boxed{1}\ \forall\boxed{2}\ \forall\boxed{3}\ \forall\boxed{4}\forall\boxed{5} \\
\left(\left(\begin{array}{l} \boxed{1}\left[\text{HEAD}\ \boxed{2}\left[\text{ARG-ST}\ \boxed{3}\right]\right] \wedge \\ \texttt{member}\left(\boxed{4}\left[\text{ARG}\ \boxed{5}\right], \boxed{3}\right) \end{array} \right) \rightarrow
\left(\boxed{4}\left[\text{RAISED}\ +\right] \leftrightarrow
\begin{array}{l} \exists\boxed{6}\ \exists\boxed{7}\ \exists\boxed{8}\ \exists\boxed{9} \\
\left(\begin{array}{l} \boxed{7}\left[\text{ARG-ST}\ \boxed{6}\right] \wedge \\ \texttt{member}\left(\boxed{8}\left[\text{ARG}\ \boxed{5}\right], \boxed{6}\right) \wedge \\ \texttt{member}\left(\boxed{9}\left[\text{ARG LOC CAT HEAD}\ \boxed{2}\right], \boxed{6}\right) \end{array} \right) \end{array} \right) \right)
\end{array}
$$

The technique that is used in the RAISED Principle to determine whether an argument is raised is very similar to the one that is used in (118). Just as Meurers 2000, Przepiórkowski 1999a assumes that the HEAD values of two *category* entities are identical if and only if one is the CATEGORY value of a syntactic projection of the other. Supposing that this is the case, (120) says that in each utterance, for each *category* component, $\boxed{1}$, with HEAD value $\boxed{2}$, and for each *argument* entity, $\boxed{4}$, on the ARG-ST list of $\boxed{1}$, $\boxed{4}$ is raised if

[12]Przepiórkowski 1999a notates the principles in the original syntax of RSRL. For reasons of readability, I translate the descriptions into equivalent AVM descriptions.

and only if there is a *head* entity, $\boxed{7}$, in the utterance with an ARG-ST list, $\boxed{6}$, that contains an *argument* entity with the *synsem* entity that is also the ARGUMENT value of $\boxed{4}$ as its ARGUMENT value and another *argument* that has the same HEAD value as $\boxed{1}$, i.e., a projection of $\boxed{1}$ is selected by means of the ARG-ST list of $\boxed{7}$.

Marking the syntactic arguments of predicates as being raised or not being raised in a given structure enables Przepiórkowski (1999a) to formulate the case principles themselves without direct reference to the utterances in which the case assignment takes place. Nevertheless, the case principles make use of the particular expressive means of RSRL. Typically, they have relational formulae in their antecedents, and they require universal quantification over the arguments of the relations. (121) is a representative example. Informally, it says that an argument of a negated verb[13] that is not its least oblique argument, has a referential index, and is a "complete functional complex" with structural case receives structural genitive case. A *complete functional complex* is defined as an *argument* entity that is marked as not being raised and whose SUBJ list only contains arguments that are marked as not being raised. The unary relation `str-cfc` holds of exactly those entities in a model that are a complete functional complex with structural case (Przepiórkowski, 1999a, p. 425).

(121) *A case principle of Polish (after Przepiórkowski 1999a, p. 427)*

$$\forall\boxed{1}\ \forall\boxed{2}$$
$$\left(\begin{array}{l}\left(\begin{array}{l}\begin{bmatrix}\textit{verbal} & \\ \textsc{neg} & + \\ \textsc{arg-st} & \langle\, \textit{object} | \boxed{2}\ \textit{list}\rangle\end{bmatrix}\\ \wedge\ \mathtt{member}\left(\boxed{1}\big[\textsc{arg loc cont index}\ \ \textit{ref}\big], \boxed{2}\right)\\ \wedge\ \mathtt{str\text{-}cfc}(\boxed{1})\end{array}\right) \rightarrow \\ \boxed{1}\big[\textsc{arg loc cat head agr case}\ \ \textit{sgen}\big]\end{array}\right)$$

5.2.2 Quantifier Retrieval

The case theory is not the only place where Przepiórkowski 1999a formalizes a principle with universal quantification over the components of utterances

[13]More accurately, the sort *verbal* is not only a supersort of the HEAD value of verbs; among others, it also subsumes the HEAD values of adjectival and adverbial participles (Przepiórkowski, 1999a, pp. 418–419).

in order to be able to formulate another principle as a local condition. A second example is the theory of quantifier scope, which is also the topic of Przepiórkowski 1997, 1998. The main point of Przepiórkowski's theory of quantifier scope is to treat quantifier retrieval lexically in a single principle that expresses a condition on words, and to forbid non-lexical quantifier retrieval. Obligatorily lexical quantifier retrieval avoids spurious ambiguities that arise in the analysis of Pollard and Yoo 1998 from the possibility of retrieving the same quantifiers in the same order at different semantic projections of the same head with the same overall result of relative quantifier scope.

In contrast to the previous signature specifications in Pollard and Sag 1994 and Pollard and Yoo 1998, Przepiórkowski defines the attribute QSTORE as being appropriate to *content*. Therefore, QSTORE values percolate with *content* entities according to the SEMANTICS PRINCIPLE along semantic projections. Quantifier retrieval is only allowed at words with a *psoa*-valued CONTENT and with a non-vacuous semantics. That means that in the standard HPSG analysis of the English copula *be*, there can be no quantifier retrieval in the word *be*, because it is semantically vacuous (it inherits the CONTENT value of one of the elements on its ARG-ST list). Quantifier retrieval at words with a CONTENT of sort *psoa* and a non-vacuous semantics proceeds as follows: They split the elements of the QSTORE sets of their selected arguments between their own QSTORE and QUANTS lists, thus fixing the scope of the retrieved quantifiers on QUANTS. Selected arguments are those elements on the ARG-ST list that are not raised from a lower ARG-ST list. Since the QUANTS list is a component of *psoas*, it percolates along semantic projections inside the *content* entity to which the *psoas* belong.

Words whose CONTENT value is not of sort *psoa* amalgamate all quantifiers of their selected arguments and pass them on using their QSTORE, which also contains the quantifier that they may contribute themselves.

Assuming for the sake of argument that the determiner of the nominal phrase *a unicorn* is a quantifier, Przepiórkowski 1998 illustrates the analysis with the sentence in (122).

(122) A unicorn appears to be approaching.

There are two semantically non-vacuous words with a *psoa*-valued CONTENT in (122), *appears* and *approaching*. Therefore, these are the two signs in the sentence where the quantifier can be retrieved. If it is retrieved by *approaching*, we obtain the so-called *de dicto* reading. If it is not retrieved by *approaching* but retrieved by *appears*, we obtain the *de re* reading. In the first reading, no unicorn needs to exist for the sentence to be true, because the semantic contribution of the modal *appears* takes scope over the quantifier. Something seems to be coming closer, and it seems to be a unicorn (but it might in fact be a horse). In the second reading, there is a specific unicorn out there in the world, and it appears to be coming closer. Each possible retrieval site corresponds to an actual reading.

There is an apparent problem with strictly lexical retrieval: Pollard and Yoo (1998) support their theory of phrasal quantifier retrieval with English data that shows that quantifiers stemming from *wh*-phrases *in situ* can be retrieved only when there is a left peripheral daughter (a subject daughter or a filler daughter of a sentence) that is a *wh*-phrase:

(123) a. *Which books did Mary wonder Bill read?

b. Who remembers where we bought which book?

(123a) cannot mean *Mary wondered which books Bill read.* This reading would be available if the *wh*-quantifier associated with *which book* could be retrieved in the embedded sentence. (123b) is ambiguous. Felicitous answers are (a), John and Martha remember where we bought which book, and (b), John remembers where we bought the physics book and Martha and Ted remember where we bought *The Wizard of Oz.* The nominal phrase *which book* can be retrieved either with the filler of the embedded clause, *where*, or with the subject of the matrix clause, *who.* Pollard and Yoo (1998, p. 431) formulate the following generalization to capture the data in (123):

(124) "At any node, retrieval, if any, of *wh*-operators must include the member of the left peripheral daughters's QUE value."

The QUE set of a *wh*-phrase contains the CONTENT value of the *wh*-quantifier of the phrase. At first, capturing the principle (124) of Pollard and Yoo seems to be a problem for a lexical theory of quantifier retrieval, because it might seem that it shows that the retrieval of *wh*-operators must take place at the phrasal node that directly dominates a left peripheral daughter of the relevant kind, and is excluded otherwise. Przepiórkowski (1999a) proves that this is not the case. He reformulates the principle in (124) as a principle on utterances. Intuitively speaking, configurations of retrieved *wh*-quantifiers in *psoas* are made sensitive to the tree-configurational environments in which the *psoas* occur. As a consequence, only if there is a certain tree configuration in the utterance, can a word retrieve a *wh*-operator:

(125) "If the QUANTS of a *psoa* contains a *wh*-quantifier, it must also contain the QUE member of a left peripheral daughter of some semantic projection of this *psoa*." (Przepiórkowski, 1999a, p. 433)

The formalization of (125) is more explicit then the informal description of the principle. First of all, the antecedent of (126) makes clear that it is indeed a condition on utterances. Second, the formalization says exactly what a *wh*-quantifier and a semantic projection are. These notions are formalized in relations. The unary relation `wh-quantifier` is defined to hold of all entities in the universe that satisfy the grammar-specific description of *wh*-quantifiers. The binary relation `semantic-projection` is defined on the basis of the relation `semantic-daughter`, which follows the explanation of the notion of the semantic head of a phrase of Pollard and Sag (1994, p. 402, fn. 16). In a head-adjunct structure, the adjunct daughter is the semantic head. In all other headed structures, the head daughter is the semantic head. For two components, x and y, of an entity, x is defined to be a semantic projection of y if and only if they are in the `semantic-projection` relation (Przepiórkowski, 1999a, p. 434).

(126) *A restriction on quantifier retrieval (Przepiórkowski, 1999a, pp. 433–434)*

$$
\begin{array}{l}
\textit{unembedded-sign} \rightarrow \\
\forall\boxed{1}\ \forall\boxed{2}\ \forall\boxed{3} \\
\left(
\begin{array}{l}
\left(
\begin{array}{l}
\boxed{1}\begin{bmatrix}\textit{word}\\ \text{SS LOC CONT QUANTS } \boxed{2}\end{bmatrix} \\
\wedge\ \texttt{wh-quantifier}(\boxed{3}) \\
\wedge\ \texttt{member}(\boxed{3},\boxed{2})
\end{array}
\right) \rightarrow \\[2ex]
\exists\boxed{4}\ \exists\boxed{5}\ \exists\boxed{6} \\
\left(
\begin{array}{l}
\left(
\begin{array}{c}
\texttt{semantic-projection}\Big(\boxed{4}\Big[\text{DTRS SUBJ-DTR } \big\langle \big[\text{SS NLOC QUE } \boxed{5}\big] \big|\ \textit{list}\big\rangle\Big],\boxed{1}\Big) \\
\vee \\
\texttt{semantic-projection}\Big(\boxed{4}\Big[\text{DTRS FILL-DTR } \big\langle \big[\text{SS NLOC QUE } \boxed{5}\big] \big|\ \textit{list}\big\rangle\Big],\boxed{1}\Big)
\end{array}
\right) \\
\wedge\ \texttt{member}(\boxed{6},\boxed{2}) \\
\wedge\ \texttt{member}(\boxed{6},\boxed{5})
\end{array}
\right)
\end{array}
\right)
\end{array}
$$

For each word, w, in an utterance, if there is a *wh*-quantifier on w's QUANTS list, then a member of the QUE set of a left peripheral daughter (a filler daughter or a subject daughter) of a semantic projection of w is also a member of w's QUANTS list. In other words, a *wh*-quantifier *in situ* can only be retrieved together with the *wh*-quantifier of a left-peripheral *wh*-phrase.

5.2.3 Summary

The nominative case assignment of Meurers 2000, and the case theory and the restriction on *wh*-quantifier retrieval in the theory of quantifier scope of Przepiórkowski 1999a have in common that they are or contain lexical principles that in some way use information about the structure of the utterance in which the pertinent words occur. The words need access to that information to be able to assign case or to retrieve quantifiers locally.

One way to make information about certain configurations of entities somewhere in an utterance locally available is by "traversing" the syntactic structure with relations and passing on the necessary non-local information as the value of some book-keeping attribute. Alternatively, some so-called feature-percolation mechanism without relations can be defined to the same end. The problem with this method is that it tends to be highly dependent on structural assumptions of specific analyses and thus yields principles that must be modified every time to be compatible with different, independently motivated architectural assumptions.

Meurers 2000 and Przepiórkowski 1999a use the expressive capabilities of RSRL to go a different route. Universal quantification over utterances enables them to describe the structures that are relevant to their principles without stating where exactly in the utterance they are structurally located. It is enough to state that they are there. Thus the lexical conditions that depend on configurations of entities elsewhere in the structure become independent of details of particular tree configurations and other design decisions. In the principle for nominative case assignment and in the principle that restricts *wh*-quantifier retrieval, the restriction is even expressed without introducing a new attribute and new attribute values. In the case theory of Przepiórkowski 1999a, the Boolean-valued attribute RAISED is only introduced in order to keep the local case principles conceptually separate from the principle that records the non-local information.

Comparing the principles of this section to the principles of the grammar of Pollard and Sag 1994, we observe that the non-local aspects of the principles of Meurers and Przepiórkowski are closely related to the logical structure of the TRACE PRINCIPLE. But unlike the formulation of the TRACE PRINCIPLE, whose informal description leaves determining the right antecedent of the principle to the reader,[14] the formulation in RSRL of their principles forces Meurers and Przepiórkowski to be explicit about the entities to which their conditions apply. In this respect, their decision to formalize the principles rigorously contributes to a high level of conceptual clarity. Their complete and concrete proposals of how their principles can be realized makes it easier for the reader to examine the conceptual and formal prerequisites of all details of the analyses.

5.3 Linearization

An analysis of second-position clitics in Serbo-Croatian leads Penn 1999a,b,c to a radical reformulation of the architecture of previous linearization-based theories in HPSG. Of the three papers, Penn 1999c contains the most comprehensive discussion of the theoretical background of linearization-based grammars, of previous analyses of second-position clitics, and of the data that motivate the new approach. The sketch of an HPSG formalization of Penn 1999c is substantially extended in Penn 1999a,b, which only give a very brief empirical motivation, followed by a presentation of technical details. The definitions of the relations and of the principles of the grammar are formulated in a syntax that closely follows the syntax of RSRL of Section 3.1.1. The only difference between Penn 1999a and Penn 1999b is that the latter, but not the former, includes explicit definitions of the relations that are needed for the principles of the grammar.

5.3.1 The Architecture

Following the general method of the present chapter, I will focus on the technical rather than on the linguistic and empirical aspects of Penn's theory. Some of the linguistic background, however, is necessary in order to understand the theory.

In the discussion of the use of chains in the `shuffle` relation of linearization-based HPSG in Section 4.3, we have seen that the entities of species

[14]See Appendix C, pp. 421–423 for discussion.

dom-obj that stand on the DOM lists of signs of Kathol 1995 have two attributes defined on them, PHON and SYNSEM, with lists of *phonstrings* and *synsem* entities as their respective values. The compaction of domain objects is achieved by identifying the SYNSEM value of a daughter sign, s, in a syntactic structure with the SYNSEM value of the compacted domain object and by concatenating the phonologies of the domain objects on the DOM list of s to obtain the PHON value of the compacted domain object on the DOM list of the mother node. An illustration of this can be found in example (58) on page 263. Their SYNSEM values and their mode of compaction give the domain objects of Kathol 1995 a very syntactic flavor. For example, they necessarily have a syntactic category; and they even have a semantic representation.

The most salient innovation of the linearization theory of Penn is the strict separation of the internal structure of domain objects from syntactic structures.[15] While Penn maintains Kathol's idea of encoding the linear order of words with topological fields and with the relative linear order of domain objects on a domain list, the domain objects become free of all tectogrammatical structure; they do not contain structure that is unrelated to topological fields, the compaction of domain objects, and the embedding of topological fields into regions. In short, all of their internal structure is related to phenogrammatical properties. Different modules of the grammar, but at least prosody and syntax, contain their own DOM lists that comprise the domain objects of the words in the structure. The different modules of grammar impose their own conditions on the DOM lists. In the DOM lists of signs, then, the requirements of the different modules come together through the identification of their DOM lists, and all conditions on them must be fulfilled simultaneously.

In (127), I depict the new appropriateness function for *dom-obj*:

(127) *The structure of domain objects in Penn 1999a,b,c*

dom-obj	PHON	*phonstring*
	COMPACTION	*field*

Since the PHON value of a domain object is always the phonological string of a word, there is no need to make it list-valued, in contrast to the PHON values of domain objects in Kathol 1995. The value of the second attribute

[15]Dowty 1996 can be seen as a precursor of this idea.

that is appropriate for *dom-obj*, COMPACTION, is a topological field marker, i.e., an entity that indicates in which topological field the domain object is located. I will presently turn to the sort hierachy under *field*.

The restructuring of domain objects allows Penn to account for seemingly inconsistent compaction requirements from different modules of grammar. For example, in the prepositional phrase *u lepi grad (in the beautiful city)*, prosody requires compaction of the proclitic *u* and the adjective *lepi* to form a prosodic word. But syntax requires the compaction of the nominal phrase, *lepi grad*, if *u lepi grad* is not a clause initial phrase. With the traditional representation of compaction, these would be inconsistent demands. The new architecture of domain objects allows for a consistent representation of these two grammatical facts.

(128) *Different compaction requirements from syntax and prosody (after Penn 1999a, p. 126)*

$$\left\langle \begin{bmatrix} \textit{dom-obj} & \\ \textsc{phon} & \textit{u} \\ \textsc{compaction} & \boxed{1}\ \textit{field} \end{bmatrix}, \begin{bmatrix} \textit{dom-obj} & \\ \textsc{phon} & \textit{lepi} \\ \textsc{compaction} & \boxed{1}\ \textit{field} \end{bmatrix}, \begin{bmatrix} \textit{dom-obj} & \\ \textsc{phon} & \textit{grad} \\ \textsc{compaction} & \textit{field} \end{bmatrix} \right\rangle$$

Prosodic compaction

$$\left\langle \begin{bmatrix} \textit{dom-obj} & \\ \textsc{phon} & \textit{u} \\ \textsc{compaction} & \textit{field} \end{bmatrix}, \begin{bmatrix} \textit{dom-obj} & \\ \textsc{phon} & \textit{lepi} \\ \textsc{compaction} & \boxed{1}\ \textit{field} \end{bmatrix}, \begin{bmatrix} \textit{dom-obj} & \\ \textsc{phon} & \textit{grad} \\ \textsc{compaction} & \boxed{1}\ \textit{field} \end{bmatrix} \right\rangle$$

Syntactic compaction

$$\left\langle \begin{bmatrix} \textit{dom-obj} & \\ \textsc{phon} & \textit{u} \\ \textsc{compaction} & \boxed{1}\ \textit{field} \end{bmatrix}, \begin{bmatrix} \textit{dom-obj} & \\ \textsc{phon} & \textit{lepi} \\ \textsc{compaction} & \boxed{1}\ \textit{field} \end{bmatrix}, \begin{bmatrix} \textit{dom-obj} & \\ \textsc{phon} & \textit{grad} \\ \textsc{compaction} & \boxed{1}\ \textit{field} \end{bmatrix} \right\rangle$$

Combining the requirements of prosody and syntax

Crucially, in the new representation, it is the identity of fields in domain objects that signifies compaction. Prosody demands that the COMPACTION values of *u* and *lepi* be identical; and syntax requires the identity of the COMPACTION values of *lepi* and *grad*. In the DOM list that represents the compaction requirements of both prosody and syntax in the sign *u lepi grad*, all three COMPACTION values must, therefore, be identical.

Topological field markers are potentially recursively structured entities. This assumption expresses the insight that the assignment of topological fields to domain objects is not absolute; topological fields are assigned relative to a region. For example, a German sentence constitutes a relative region for the domain objects that it contains, and the domain objects in that region are assigned the topological field markers of German sentences in sentential regions, such as complementizer field, middle field, or verb cluster. But the clause itself could be in the *Nachfeld* of a matrix clause. As a whole, the sentence is then assigned the topological field marker *nachfeld*. As this brief example also shows, regions are often defined syntactically. But as Penn points out, as soon as there is a more general theory available, this no longer needs to be the case in general.

For the analysis of Serbo-Croatian sentences, Penn 1999c suggests the following topological field markers:

(129) *The sort hierarchy of topological fields*

field

  matrix

  pre-cf REGION *field*

  cf REGION *field*

  post-cf REGION *field*

  rf REGION *field*

  ef REGION *field*

The topological field markers given in (129) stand for the pre-clitic field (*pre-cf*), the clitic field (*cf*), the post-clitic field (*post-cf*), the remainder field (*rf*), and the extraposed field (*ef*) for embedded clauses. For all five of them, the *field*-valued attribute REGION is appropriate. The REGION value of entities indicates to which region they belong. The grammatical theory determines that the only possible REGION values of the given field markers are *ef* and *matrix*.

The atomic field marker *matrix* is the only one for which the attribute REGION is not appropriate. Intuitively, the matrix region is the region of unembedded signs, i.e., entities of sort *matrix* are the intuitive counterpart in the theory of topological fields to entities of sort *unembedded-sign* in the structure of signs. Besides restricting the admissible REGION values of specific topological field markers, the theory of topological fields imposes restrictions on the number of topological field markers of one species in a region, and on the linear order of domain objects with certain topological field markers relative to a region. In every sentential region, there must be exactly one *pre-cf* field, and there can be at most one *cf*, at most one *post-cf*, and at most one *ef* field. In every region, the *pre-cf* field precedes the *cf* field, the *cf* field precedes the *post-cf* field, and so on, in the order in which the fields are listed in (129). We will see formalizations of some of these topological principles below.

Penn 1999a,b,c considers the structure of domain objects a linguistic universal. The particular topological field markers, on the other hand, are language-specific. Moreover, the topological field markers given in (129) are not meant to be an exhaustive list for Serbo-Croatian. In fact, Penn 1999a,b states another set of topological field markers for a more fine-grained distinction between various kinds of clitics, and assumes that there might be more topological field markers that structure the region of nominal phrases. For the present overview of the formal structure of the theory, the topological field markers of (129) suffice.

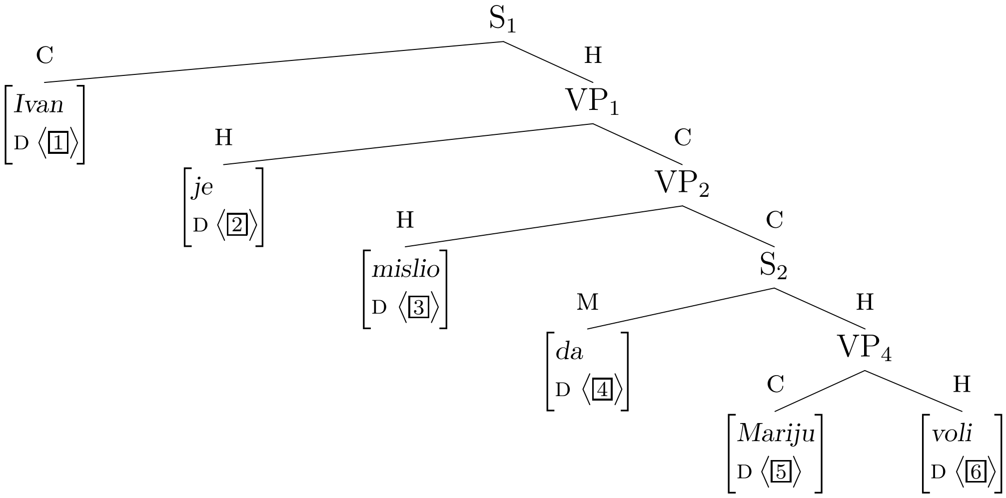
The best way to get an impression about how the pieces of Penn's linearization theory fall into place is to consider an example. The sentence in (130) and its analysis in (131)–(132) are quoted from Penn 1999a, pp. 172–173.

(130)	Ivan	je	mislio	da	voli	Mariju.
	Ivan	CL-3s	thought	COMP	loves	Mary-ACC

'Ivan thought that he loves Mary.'

In (131), I sketch a hypothetical syntactic structure of the sentence (130). At the leaves of the tree, I indicate the PHON value of each sign (first line) and the element on its DOM list, where DOM is abbreviated as D. The succeeding DOM list of the sentence, S_1, reveals the internal configuration of entities under the domain objects on the DOM list of each leaf. I abbreviate the attribute COMPACTION with C and the attribute REGION with R. Of course, nothing of what we have said so far implies the given intended structure of the domain objects, and it will be the task of the topological principles to specify it.

(131) *The analysis of sentence (130)*



The DOM list of S_1:

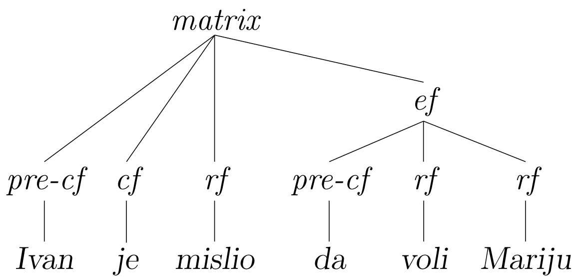
$$\left\langle \boxed{1}\begin{bmatrix} \textit{dom-obj} \\ \text{PHON} \;\; \textit{Ivan} \\ \text{C} \begin{bmatrix} \textit{pre-cf} \\ \text{R} \; \boxed{0} \; \textit{matrix} \end{bmatrix} \end{bmatrix}, \boxed{2}\begin{bmatrix} \textit{dom-obj} \\ \text{PHON} \;\; \textit{je} \\ \text{C} \begin{bmatrix} \textit{cf} \\ \text{R} \; \boxed{0} \; \textit{matrix} \end{bmatrix} \end{bmatrix}, \boxed{3}\begin{bmatrix} \textit{dom-obj} \\ \text{PHON} \;\; \textit{mislio} \\ \text{C} \begin{bmatrix} \textit{rf} \\ \text{R} \; \boxed{0} \; \textit{matrix} \end{bmatrix} \end{bmatrix}, \right.$$

$$\left. \boxed{4}\begin{bmatrix} \textit{dom-obj} \\ \text{PHON} \;\; \textit{da} \\ \text{C} \begin{bmatrix} \textit{pre-cf} \\ \text{R} \; \boxed{7} \begin{bmatrix} \textit{ef} \\ \text{R} \; \boxed{0} \; \textit{matrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}, \boxed{5}\begin{bmatrix} \textit{dom-obj} \\ \text{PHON} \;\; \textit{voli} \\ \text{C} \begin{bmatrix} \textit{rf} \\ \text{R} \; \boxed{7} \begin{bmatrix} \textit{ef} \\ \text{R} \; \boxed{0} \; \textit{matrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}, \boxed{6}\begin{bmatrix} \textit{dom-obj} \\ \text{PHON} \;\; \textit{Mariju} \\ \text{C} \begin{bmatrix} \textit{rf} \\ \text{R} \; \boxed{7} \begin{bmatrix} \textit{ef} \\ \text{R} \; \boxed{0} \; \textit{matrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \right\rangle$$

Note that the configuration under each domain object, $\boxed{1}$–$\boxed{6}$, persists throughout the structure. Wherever the domain object, $\boxed{5}$, of the verb *voli* occurs on a DOM list, it carries the representation of the fact that *voli* stands in the *ef* field relative to the *matrix* region of the matrix sentence, S_1, and it stands in an *rf* field relative to the region, *ef*, of the embedded clause, S_2. Note also that we must appeal to the usual `shuffle` relation of linearization grammars to relate the DOM lists of mother nodes to the DOM lists of their daughters. For example, the DOM lists of the sign *Ivan*, of VP_1, and of S_1 stand in the `shuffle` relation; and so do the DOM lists of *Mariju*, of *voli*, and of VP_4. The DOM list of each phrase thus contains exactly the shuffled domain objects of the domain lists of its daughters.

The topological facts of (131) can be depicted more perspicuously in a graph notation. The graph notation preserves the information about the identities between REGION values and the assignment of topological field markers to domain objects. In (132) it can easily be seen that the overall sentence (which is an *unembedded-sign*) forms a unique *matrix* region; and the embedded sentence, *da voli Mariju*, stands in the (unique) *ef* field of the matrix sentence. By contrast, the *rf* fields of *voli* and *Mariju* are not depicted as being identified.

(132)



With the signature given in (127) and (129), I have everything in place for a discussion of some of Penn's topological relations and principles that lead to the given analysis of sentence (130). Penn notates his relation definitions and his principles in a syntax that bears a very close resemblance to the RSRL syntax of Section 3.1.1. For consistency with my presentation of the other RSRL literature, and to make it easier for the reader, I translate all of Penn's original descriptions into equivalent AVM descriptions. My reformulation includes a translation of Penn's parametric list descriptions into the treatment of parametric lists with relations that was introduced in Section 4.5.2.

5.3.2 Topological Relations and Principles

I start my summary with the definitions of the three relations, `topo-field`, `region`, and `topo-order`. Almost all topological principles, representative examples of which I will present in (136), refer to at least one of these relations. Without definition, I presuppose an appropriate specification of the binary relation, `exist`, that is heavily used in Penn's descriptions. The relation `exist`(x, y) is supposed to hold of all components of entities such that y is a component of x.[16]

[16] Penn 1999b, pp. 195–196, defines `exist` to hold of all pairs of entities for which it is needed in the topological principles. A more general definition is omitted. On page 358, I present a schema for defining a relation, `component`, relative to signatures with finite sets

The relation `topo-field` determines the topological field of a domain object relative to some region. In Penn's words, "the relative topological field of domain object, d, with respect to region, r, is f." (Penn, 1999b, p. 194)

(133) $\texttt{topo-field}(d, r, f) \overset{\forall}{\Longleftarrow}$

$$^{d}\begin{bmatrix} \textit{dom-obj} \\ \text{COMPACTION } \boxed{1} \end{bmatrix} \wedge \boxed{1} = r \ \wedge \ f = \boxed{1}$$

$\texttt{topo-field}(d, r, f) \overset{\forall}{\Longleftarrow}$

$$^{d}\begin{bmatrix} \textit{dom-obj} \\ \text{COMPACTION } \boxed{1} \end{bmatrix} \wedge \neg \boxed{1} = r \ \wedge \ \texttt{exist}(\boxed{1}, f) \ \wedge \ ^{f}\big[\text{REGION } r\big]$$

If the region, r, with respect to which the topological field, f, is determined is the COMPACTION value of the domain object, then f equals r (first clause). If r is not the COMPACTION value of d, then the field f with respect to r of the domain object d is that topological field component of d's COMPACTION value whose REGION value is r (second clause). For example, the relation `topo-field` says that the relative topological field of the domain object of *Mariju* with respect to the region $\boxed{7}$[*ef*] in (131) is *rf*.

Typically, regions are nested into other regions in domain objects. This implies that there might not be a unique region that the domain objects of a domain list have in common. For example, the domain objects *da*, *voli* and *Mariju* on the domain list of S_2 in (131) share the *matrix* region and the *ef* region. Nevertheless there is exactly one region that determines the relative topological fields and thus the linear order of the domain objects on the domain list of S_2 (and thus on every domain list higher in the syntactic tree than S_2). For *da*, *voli*, and *Mariju*, it is the *ef* region. In general, it is the region that all domain objects on a list have as their component and that can be reached by the shortest path from the domain objects compared to the length of the paths by which all other regions that all of them share can be reached. Penn identifies the relevant region as the smallest region that contains all domain objects on the list. Each list of domain objects is related to the smallest region to which all elements of the list belong by the binary relation, `region`. Its definition employs an auxiliary relation,

---

of attributes. The relation `component` corresponds to the generalized relation `exist` that Penn envisages, but with the order of the arguments reversed: In $\texttt{component}(x, y)$, x is a component of y.

region_recurse. In the definition of region, "r is the smallest region that contains the domain objects on the domain list, ds." (Penn, 1999b, p. 195)

(134)
$$
\begin{array}{l}
\texttt{region}(ds, r) \stackrel{\forall}{\Longleftarrow} \\
\quad \texttt{list-of-domain-objects}\Big(ds \left\langle {}^{d}[\textsc{compaction}\ \boxed{1}] \middle| \boxed{2} \right\rangle\Big) \\
\quad \land\ \texttt{region\_recurse}(\boxed{2}, d, \boxed{1}, r)
\end{array}
$$

$$
\begin{array}{l}
\texttt{region\_recurse}(ds, d_1, r_1, r) \stackrel{\forall}{\Longleftarrow} \\
\quad ds\ \langle\rangle\ \land\ r_1 = r
\end{array}
$$

$$
\begin{array}{l}
\texttt{region\_recurse}(ds, d_1, r_1, r) \stackrel{\forall}{\Longleftarrow} \\
\quad ds \left\langle {}^{d_2}[\textsc{compaction}\ \boxed{1}] \middle| \boxed{2} \right\rangle \\
\quad \land\ \exists \boxed{3} \left(\begin{array}{l}
\left(\begin{array}{l}
(\texttt{exist}(\boxed{1}, \boxed{3})\ \land\ \boxed{3} = r_1) \\
\lor\ \forall \boxed{4} \left(\begin{array}{l}
\texttt{exist}(\boxed{1}, \boxed{4})\ \rightarrow \\
\neg \boxed{4} = r_1 \\
\land\ \exists f_1\ \exists f_2 \\
\left(\begin{array}{l}
\texttt{exist}\Big(r_1, {}^{f_1}[\textsc{region}\ \boxed{3}]\Big) \\
\land\ \texttt{exist}\Big(\boxed{1}, {}^{f_2}[\textsc{region}\ \boxed{3}]\Big) \\
\land\ \neg f_1 = f_2
\end{array} \right)
\end{array} \right)
\end{array} \right) \\
\land\ \texttt{region\_recurse}(\boxed{2}, d_2, \boxed{3}, r)
\end{array} \right)
\end{array}
$$

The relation region_recurse is responsible for determining the smallest region, r, that the domain objects on the domain list, ds, have in common. If there is only one element on ds, then its COMPACTION value is trivially the smallest region.

In the general case, the first argument of region_recurse contains the current rest of the domain list, the second argument the current first element, d_1, of the domain list, the third argument the smallest region as determined from the beginning of the domain list up to the current element, and the last argument the smallest region in the overall domain list, i.e., the "result". When we go into the recursion (second clause of region_recurse), there are two possibilities: Either the smallest region, r_1, that was determined so far is a component of the COMPACTION value of the domain object, d_2. Then we pass it on (with $\boxed{3}$) to the next recursion step along with the rest of the domain list, $\boxed{2}$, the current domain object, d_2, and the overall smallest region, r. Or the smallest region so far is not a component of the COMPACTION value of the domain object, d_2, on the domain list. Then it cannot be the overall smallest region, because the region is in fact too small to include the current domain object d_2, and we need to find a bigger region that is a component of the COMPACTION value of d_2. As the bigger region, we take a region $\boxed{3}$ that is a component of the region that we considered to be the smallest region so far (r_1) and a component of the COMPACTION value of d_2. $\boxed{3}$ is chosen in such a way that the topological field markers, f_1 and f_2, whose REGION value $\boxed{3}$ is are distinct, i.e., $\boxed{3}$ is the smallest common region. We go into the next recursion step with the rest of the domain list, $\boxed{2}$, the current domain object, d_2, the current smallest region, $\boxed{3}$, and the overall smallest region, r. When we reach the end of the domain list in the first clause of `region_recurse`, the current smallest region is identified with the overall smallest region.[17]

The meaning of the relation `region` in models of the grammar can be illustrated with the structure of the sentence in (131). For example, `region` holds of the pair consisting of the singleton DOM list of the verb of the matrix sentence, *mislio*, and the COMPACTION value, *rf*, of the domain object of *mislio*. Another pair in the `region` relation is the DOM list of the embedded clause, S_2, and the *ef* entity $\boxed{7}$, which is the smallest region that contains all domain objects of the embedded clause. Finally, the DOM list of the matrix clause and the *matrix* entity, $\boxed{0}$, are in the `region` relation.

The last relation definition that I want to include in my summary of the relation definitions of Penn 1999b is the relation `topo-order`. It relates a domain list, ds, to a region, r, and ensures that the topological ordering of region r holds on the domain list, ds. In Penn's words, "the domain list, ds, respects the topological ordering defined on region, r." (Penn, 1999b, p. 195)

(135) $\texttt{topo-order}(ds, r) \overset{\forall}{\Longleftarrow}$
$\quad ds\ \langle\rangle$

[17] Note that the second argument of `region_recurse` is never used in the definition of its meaning and could thus be dropped without consequences.

$$
\begin{array}{l}
\texttt{topo-order}(ds, r) \overset{\forall}{\Longleftarrow} \\
\quad ds\,\langle \boxed{1} | \boxed{2} \rangle \;\wedge\; \texttt{topo-order}(\boxed{2}, r) \;\wedge\; \texttt{topo-field}(\boxed{1}, r, f) \\
\quad \wedge \left({}^{f}[\mathit{cf}] \;\rightarrow\; \forall d_2 \left(\begin{array}{l} \texttt{exist}\Big(\boxed{2}, {}^{d_2}[\mathit{dom\text{-}obj}]\Big) \;\rightarrow \\ \exists f_2\; \texttt{topo-field}\Big(d_2, r, {}^{f_2}[\neg\; \mathit{prec\text{-}cf}]\Big) \end{array} \right) \right) \\
\quad \wedge \left(\begin{array}{l} {}^{f}[\mathit{post\text{-}cf}] \;\rightarrow \\ \forall d_2 \\ \left(\begin{array}{l} \texttt{exist}\Big(\boxed{2}, {}^{d_2}[\mathit{dom\text{-}obj}]\Big) \;\rightarrow \\ \exists f_2\; \texttt{topo-field}\Big(d_2, r, {}^{f_2}[\neg\; [\; \mathit{prec\text{-}cf} \vee \mathit{cf}]]\Big) \end{array} \right) \end{array} \right) \\
\quad \wedge \left(\begin{array}{l} {}^{f}[\mathit{rf}] \;\rightarrow \\ \forall d_2 \\ \left(\begin{array}{l} \texttt{exist}\Big(\boxed{2}, {}^{d_2}[\mathit{dom\text{-}obj}]\Big) \;\rightarrow \\ \exists f_2\; \texttt{topo-field}\Big(d_2, r, {}^{f_2}[\neg\; [\; \mathit{prec\text{-}cf} \vee \mathit{cf} \vee \mathit{post\text{-}cf}\,]]\Big) \end{array} \right) \end{array} \right) \\
\quad \wedge \left(\begin{array}{l} {}^{f}[\mathit{ef}] \;\rightarrow \\ \forall d_2 \\ \left(\begin{array}{l} \texttt{exist}\Big(\boxed{2}, {}^{d_2}[\mathit{dom\text{-}obj}]\Big) \;\rightarrow \\ \exists f_2\; \texttt{topo-field}\Big(d_2, r, {}^{f_2}[\neg\; [\; \mathit{prec\text{-}cf} \vee \mathit{cf} \vee \mathit{post\text{-}cf} \vee \mathit{rf}\,]]\Big) \end{array} \right) \end{array} \right)
\end{array}
$$

According to the first clause of `topo-order`, the relation trivally holds between each empty list and any other entity with a common ancestor. According to the second clause, the relation `topo-order` holds between each nonempty list, ds, and a region, r, if and only if it holds between the rest of ds and r, and, furthermore, if f is the topological field of the first element of ds relative to r, and (1), if f is of sort *cf*, then no domain object in the rest of ds is of sort *pre-cf* relative to r; and (2), if f is of sort *post-cf*, then no domain object in the rest of ds is of sort *pre-cf* or of sort *cf* relative to r; and (3), if f is of sort *rf*, then no domain object in the rest of ds is of sort *pre-cf*, of sort *cf* or of sort *post-cf* relative to r; and, finally, if f is of sort *ef*, then no domain object in the rest of ds is of sort *pre-cf*, of sort *cf*, of sort *post-cf*, or of sort *rf* relative to r.

The definitions of the relations `region` and `topo-order` are fairly complex. Once they are defined, it is straightforward to formulate the topological principles of Penn 1999a, pp. 133–134, that build on the new structure of domain objects. In (136), I present five representative examples of topological principles that show how the new structure of domain objects is used to specify their linear order and their topological properties.

(136) a. Topological Order:

$$\texttt{list-of-domain-objects}\Big([\textit{nelist}]\Big) \rightarrow$$
$$\exists\boxed{1}\ (\texttt{region}(:, \boxed{1}) \wedge \texttt{topo-order}(:, \boxed{1}))$$

b. Field Existence:

$$\Big(\textit{unembedded-sign} \vee [\textsc{dtrs}\ \textit{head-marker-struc}]\Big) \rightarrow$$
$$\exists\boxed{1}\ \exists\boxed{2}\ \exists\boxed{3}\ \exists\boxed{4} \begin{pmatrix} [\textsc{dom}\ \boxed{1}] \wedge \texttt{region}(\boxed{1}, \boxed{2}) \wedge \texttt{exist}(\boxed{1}, \boxed{3}) \\ \wedge\ \texttt{topo-field}\Big(\boxed{3}, \boxed{2}, \boxed{4}[\textit{pre-cf}]\Big) \end{pmatrix}$$

c. Embedded Clause Compaction:

$$[\textsc{dtrs}\ \textit{head-marker-struc}] \rightarrow$$
$$\exists\boxed{1}\ \exists\boxed{2}\ \Big([\textsc{dom}\ \boxed{1}] \wedge \texttt{region}\Big(\boxed{1}, \boxed{2}[\textit{ef}]\Big)\Big)$$

d. Matrix Compaction:

$$\textit{dom-obj} \rightarrow \exists\boxed{1}\ \boxed{1}[\textit{matrix}]$$

$$\textit{unembedded-sign} \rightarrow \forall\boxed{1}\ \forall\boxed{2}\ \Big(\Big(\boxed{1}[\textit{matrix}] \wedge \boxed{2}[\textit{matrix}]\Big) \rightarrow \boxed{1} = \boxed{2}\Big)$$

e. Planarity:

$$\forall\boxed{1}\ \forall\boxed{2}\ \forall\boxed{3}\ \forall\boxed{4}\ \forall\boxed{5}$$
$$\left(\begin{array}{l} \begin{pmatrix} \texttt{list-of-domain-objects}(\langle\boxed{1}, \boxed{2}, \boxed{3}|\ \textit{list}\rangle) \\ \wedge\ \texttt{region}(:, \boxed{4}) \\ \wedge\ \texttt{topo-field}(\boxed{1}, \boxed{4}, \boxed{5}) \\ \wedge\ \texttt{topo-field}(\boxed{3}, \boxed{4}, \boxed{5}) \end{pmatrix} \rightarrow \\ \texttt{topo-field}(\boxed{2}, \boxed{4}, \boxed{5}) \end{array}\right)$$

The first principle, Topological Order (136a), imposes the desired order on domain lists: For each nonempty domain list, its elements obey the topological ordering defined on the smallest region that contains the elements on the domain list. Given the assignments of topological fields to the domain objects in (131), Topological Order fully determines the order of the domain objects on the DOM list of S_1 as given there, with the exception of the relative ordering of $\boxed{5}$ and $\boxed{6}$. Since $\boxed{5}$ and $\boxed{6}$ have the same topological fields in the same regions, the reversed ordering would also be consistent with Topological Order.

Field Existence, (136b), is a different type of topological principle. It requires the presence of a topological field in certain syntactic environments, or, using a different metahpor, Field Existence requires that a certain field always be filled in certain syntactic environments. In every unembedded sign and in every head-marker structure—which Penn assumes to subsume all Serbo-Croatian sentences—the domain list contains a domain object whose topological field relative to the smallest region of the domain list is *pre-cf*. In other words, the *pre-cf* field of every sentence is filled. In (131), Field existence is satisfied, because in the matrix clause, *Ivan* is in the *pre-cf* field, and in the embedded clause, the complementizer *da* fills the *pre-cf* field.

The grammar of Penn 1999a contains more principles that are similar to Field Existence. They are Field Uniqueness principles that demand that in unembedded signs and in head-marker structures, there be at most one *pre-cf*, *cf*, *post-cf* and *ef* field. Notice that this does not mean that there can only be one domain object in the *ef* field in the relevant region. It means that all domain objects in the *ef* field in the relevant region have the same *ef* entity, i.e., they are compacted.

Embedded Clause Compaction, (136c), is an example of another important class of topological principles. Compaction principles say that in certain environments, the domain objects on the domain list must compact. Embedded Clause Compaction requires that the smallest region of all domain objects of a head-marker structure—which, in Penn's theory, identifies embedded clauses—is one and the same *ef* entity. There are similar, if more complex, compaction principles for nominal phrases, for prepositional phrases and for second-position clitic compaction. What makes the compaction of nominal phrases more complex is the fact that NP compaction is sensitive to interacting prosodic and syntactic conditions that constrain the linear order of second position clitics. Although this point is crucial for the linguistic analysis of Serbo-Croatian, and it is the cornerstone of Penn's analytical contribution, NP and PP compaction as well as second-position clitic compaction do not add anything new from a purely technical perspective, and I refer the reader to Penn 1999a, pp. 133–134, for their formalization.

Matrix Compaction, (136d), consists of two descriptions. The first description ensures the existence of a *matrix* region in every domain object. Since the REGION attribute is not appropriate for *matrix*, this implies that the embedding of regions into regions is always finite and non-cyclic. The second description is another example of the use of the concept of *unembedded-signs* in a formally explicit HPSG theory. While the first part of Matrix Compaction implies the existence of a *matrix* component in every sign with a nonempty DOM list, the second part of Matrix Compaction adds the constraint that this *matrix* entity be unique in every unembedded sign. In short, every unembedded sign has a unique topological matrix region.

The last topological principle, Planarity (136e), is an example of the principles that are necessitated by the constraint-based approach to topological fields relative to regions. These principles can be understood as guaranteeing that the topological structures of all unembedded signs have a tree-shaped representation of the kind illustrated in (132). The task of Planarity is to ensure that constituents in a multiply-fillable topological field (such as *rf*) cannot be made discontinuous by intervening domain objects. More specifically, in any list of domain objects with three elements and the smallest region $\boxed{4}$, if the first and the third element on the domain list are assigned the same topological field marker, $\boxed{5}$, relative to $\boxed{4}$ then the intervening domain object must also be assigned the topological field marker $\boxed{5}$.

5.3.3 Summary

The topological principles and the relations of Penn employ all of the descriptive extensions to standard feature logics of RSRL, with the exception of chains.[18] Not only do the principles make use of complex, new relations, the definitions of the meaning of the relations `region_recurse` and `topo-order` also contain explicit existential and universal quantification, and relational formulae in the scope of negation. Similarly, some of the topological principles contain relational formulae in antecedents of implications, and they use existential and universal quantification. The perspective of component quantification can be saliently observed in the principle Matrix Compaction, in which the presence of a unique *matrix* entity in *unembedded-signs* is enforced. RSRL is thus a necessary prerequisite for expressing the linearization theory of second-position clitics in Serbo-Croation of Penn 1999a,b,c as a formalized HPSG grammar.

The formalization in RSRL of Penn's theory is not a mere addition to an independently developed linearization architecture, but it is an integral part of the formulation of a new type of linearization grammar. Bracketing mismatches between syntax and phonology of the kind exemplified by the Serbo-Croatian prepositional phrase *u lepi grad* in (128) are known to be hard to account for in formal grammars. The data of Serbo-Croatian second

[18]Chains may occur, however, in the definition of the `shuffle` relation. Penn's theory uses `shuffle` for relating the DOM list of phrases to the DOM lists of their daughters, just like the previous approaches to linearization-based grammar.

position clitics that is at the heart of Penn's analysis is a famous instance of the same theoretical problem. Bracketing paradoxes have been an unsolved puzzle for many linguistic theories, and it is not obvious how the traditional linearization theories of HPSG could be extended naturally in order to give a satisfactory account of the empirical data. The expressive power and the flexible grammar architecture of RSRL enable Penn to provide a genuinely new solution for a long-standing, theoretical problem. Penn's work is thus an excellent example of how a mathematical formalism can support precise and innovative answers to open questions of linguistic theory.

5.4 Model-theoretic Semantics[19]

The traditional semantic structures of HPSG grammars were built on ideas that came from situation semantics (Barwise and Perry, 1983). That means that in Pollard and Sag 1994, the configurations under *content* entities that are defined by the signature and restricted by the SEMANTICS PRINCIPLE, the QUANTIFIER BINDING CONDITION and the CONTROL THEORY are inspired by situation semantics, and it is not straightforward to compare them with other, model-theoretic approaches to natural language. In an attempt to bridge the gap between HPSG and much of the current linguistic literature of model-theoretic semantics, Richter and Sailer 1999a,b introduce a completely different signature for *content* entities together with a new set of principles that restrict the admissible configurations under *content* entities. The idea is to furnish signs with CONTENT values that can be regarded as representations of a symbolic language with a model-theoretic interpretation. Richter and Sailer choose the language Ty2 of two-sorted type theory (Gallin, 1975) as the semantic object language under CONTENT of their grammars of French and Polish negative polarity items, because Ty2 is one of the standard representation languages of natural language semantics.[20]

In this section, I will sketch how a signature and a theory of Ty2 can be defined in RSRL. For that purpose, I will specify an RSRL grammar whose intended exhaustive model is the language of Ty2. In fact, a proof by

[19]The text of this section is partly based on joint work with Manfred Sailer published in the appendix to Richter and Sailer 1999a.

[20]In particular, the fragment of Ty2 that is actually used in linguistic analyses is equivalent to Richard Montague's Intensional Logic, but it has a number of technical advantages. See Zimmermann 1989 for discussion.

Sailer (2000) shows that the models of that grammar can be thought of as containing expressions of Ty2. In contrast to the theories of Sections 5.2 and 5.3, the formalization of Ty2 in RSRL does not in and by itself constitute a new analysis of some empirical linguistic phenomenon. It is a technical way of combining two formal systems that already existed independently before, a formalism for HPSG and two-sorted type theory. The integration of Ty2 with HPSG offers the opportunity to study the interaction between the techniques of model-theoretic semantics and of constraint-based grammar in rigorously formalized HPSG grammars. The existence of a mathematical formalism for HPSG grammars is a necessary prerequisite for making Ty2 available to HPSG linguists.

Before I go into the details of the specification of Ty2 in RSRL, it might be useful to illustrate the objective with a small example. The goal is to specify a signature and the principles of a grammar that defines a standard semantic representation language. This grammar can then be added to a grammar that is lacking semantic representations. With additional principles that constrain the combination of semantic and syntactic (and, potentially, other) structures, we obtain an analysis of signs whose model-theoretic meaning is determined by the semantic representations that are their CONTENT value. With the PRINCIPLE OF FULL INTERPRETATION (PFI), we will later look at an example of an interface principle between syntax and semantics. The PFI constrains the admissible semantic representations, which are often called the *logical form*, of utterances.[21]

We start with lexical specifications. What we want to be able to do in the end is to specify the semantics of the universal quantifier *every* in its lexical entry as illustrated in (137). The variable w is a world index.

(137) *The relevant part of the lexical entry of* every

$$\begin{bmatrix} \textit{word} \\ \textsc{phon} \ \langle \textit{every} \rangle \\ \textsc{ss loc cont lf} \ \ \lambda P \lambda Q. \forall x[P(w)(x) \rightarrow Q(w)(x)] \end{bmatrix}$$

The value of the attribute LOGICAL-FORM (LF), which is appropriate to *content*, is the expression $\lambda P \lambda Q. \forall x[P(w)(x) \rightarrow Q(w)(x)]$. Of course, what we really have to do is to *describe* that expression, but once we will have

[21] More interface principles within the present architecture of logical forms can be found in Richter and Sailer 1999a,b, and in Sailer 2000.

explained how we can do that, we may as well write down the expression directly, which is much more readable than an AVM formula describing it. The standard interpretation of the given Ty2 expression determines the meaning of the word *every*.

Once we have lexical specifications of logical forms of the kind exemplified in (137), we can introduce a new SEMANTICS PRINCIPLE. In the new SEMANTICS PRINCIPLE, we will define functional application as the mode of combining the semantic contributions of two signs when they form a phrase. (138) illustrates the effect with the nominal phrase *every unicorn*. The semantic contributions of the quantifier *every* and of the noun *unicorn* are combined by functional application of the former to the latter. Note that, in addition, the resulting expression is fully β-reduced. To obtain that result, we will also have to define a mechanism for β-reduction in RSRL.[22]

(138) *Functional application in HPSG*

NP

$$\begin{bmatrix} \text{PHON} & \langle \textit{every, unicorn} \rangle \\ \text{SS LOC CONT LF} & \lambda Q.\forall x[\mathsf{unicorn}'(w)(x) \rightarrow Q(w)(x)] \end{bmatrix}$$

C: Det

$$\begin{bmatrix} \text{PHON} & \langle \textit{every} \rangle \\ \text{SS LOC CONT LF} & \lambda P \lambda Q.\forall x[P(w)(x) \rightarrow Q(w)(x)] \end{bmatrix}$$

H: N

$$\begin{bmatrix} \text{PHON} & \langle \textit{unicorn} \rangle \\ \text{SS LOC CONT LF} & \mathsf{unicorn}'(w) \end{bmatrix}$$

5.4.1 A Grammar of the Syntax of Ty2

To be explicit about what the RSRL grammar of Ty2 is supposed to specify, it is necessary to recall the definition of the syntax of Ty2. Assume that e, t, and s are three distinct objects, none of which is an ordered pair. Let $Types$ be the the smallest set that contains e, t, and s, and is closed under pairs. $Types$ is the set of *two-sorted types* (hence *types*). If $\tau \in Types$, then Var_τ is a denumerably infinite set of variables of type τ. I write $v_{n,\tau}$ for the $(1+n)$th variable of type τ. Similarly, if $\tau \in Types$, then $Const_\tau$ is a denumerably infinite set of constants of type τ, and I write $c_{n,\tau}$ for the $(1+n)$th constant of type τ.

[22] β-reduction is sometimes also called lambda conversion in the literature. My terminology follows the introductory textbook by Hindley and Seldin (1986).

Definition 98 *The* ***meaningful expressions of Ty2*** *are the smallest family* $(\mathrm{Ty2}_\tau)_{\tau \in Types}$ *such that*

> *for each* $\tau \in Types$, $Var_\tau \cup Const_\tau \subset \mathrm{Ty2}_\tau$,
>
> *for each* $\tau \in Types$, *for each* $\tau' \in Types$,
>
> > *if* $\alpha \in \mathrm{Ty2}_{\langle \tau', \tau \rangle}$ *and* $\beta \in \mathrm{Ty2}_{\tau'}$, *then* $\alpha(\beta) \in \mathrm{Ty2}_\tau$,
>
> *for each* $\tau \in Types$, *for each* $\tau' \in Types$, *for each* $n \in \mathbb{N}$, *for each* $v_{n,\tau'} \in Var_{\tau'}$, *for each* $\alpha \in \mathrm{Ty2}_\tau$,
>
> > $(\lambda v_{n,\tau'}.\alpha) \in \mathrm{Ty2}_{\langle \tau', \tau \rangle}$,
>
> *for each* $\tau \in Types$, *for each* $\alpha \in \mathrm{Ty2}_\tau$, *for each* $\beta \in \mathrm{Ty2}_\tau$,
>
> > $(\alpha = \beta) \in \mathrm{Ty2}_\tau$,
>
> *for each* $\alpha \in \mathrm{Ty2}_t$,
>
> > $\neg\alpha \in \mathrm{Ty2}_t$,
>
> *for each* $\alpha \in \mathrm{Ty2}_t$, *for each* $\beta \in \mathrm{Ty2}_t$,
>
> > $(\alpha \wedge \beta) \in \mathrm{Ty2}_t$, *and* *(analogously for* $\vee, \rightarrow$, *and* $\leftrightarrow$*)*
>
> *for each* $\tau \in Types$, *for each* $n \in \mathbb{N}$, *for each* $v_{n,\tau} \in Var_\tau$, *for each* $\alpha \in \mathrm{Ty2}_t$,
>
> > $\exists v_{n,\tau}\alpha \in \mathrm{Ty2}_t$. *(and analogously for* $\forall$*)*

For simplicity, the RSRL grammar of the meaningful expressions of Ty2 will describe a slight notational variant of the expressions of DEFINITION 98. In the grammar, I give every meaningful expression a subscript that indicates its type instead of only giving type subscripts to variables and constants. For example, if α is a variable of type $\langle e, t\rangle$ and $c_{n,e}$ is a constant, the RSRL grammar will specify structures of the form $\alpha(c_{n,e})_t$ as the functional application of α to $c_{n,e}$ instead of $\alpha(c_{n,e})$, as it is done in DEFINITION 98.

In a complete definition of Ty2, it would now be necessary to say what a Ty2 model is and how the meaningful expressions of Ty2 are interpreted in models. These definitions, however, are not needed here, because we only want to specify the syntax of Ty2 with RSRL. The semantics of the expressions will only become interesting in linguistic applications when we ask ourselves what signs mean. Since I will not discuss linguistic applications, I omit the definition of Ty2 models and the semantics of Ty2 expressions.[23]

We are now ready to define the RSRL grammar of the meaningful expressions of Ty2. My specification of the grammar of Ty2 closely follows the appendix to Richter and Sailer 1999a with the modifications proposed in Sailer 2000. The ontology of Ty2 expressions is determined by a hierarchy of sorts under the supersort *meaningful-expressions* (*me*). The sort *me* will be defined as appropriate value for the pair ⟨*content*, LOGICAL-FORM⟩, and thus be anchored in the feature geometry of signs.

(139) The sort hierarchy of the sort *meaningful-expression* (*me*):

```
me   TYPE  type
    var(iable)   N-NUM(BER)  n-num(ber)
    const(ant)   N-NUM(BER)  n-num(ber)
    appl(ication)   FUNC(TOR)    me
                    ARG(UMENT)   me
    abstr(action)   VAR   variable
                    ARG   me
    equ(ation)   ARG1   me
                 ARG2   me
    neg(ation)   ARG   me
    l(ogical)-const(ant)   AGR1   me
                           ARG2   me
        dis(junction)
        con(junction)
        imp(lication)
        bi-imp(lication)
    quant(ifiers)   VAR     variable
                    SCOPE   me
        uni(versal)
        exi(stential)
```

[23]For models and a semantics of Ty2 expressions that are directly compatible with my DEFINITION 98, see Zimmermann 1989, p. 66.

The entities in the denotation of the new species are the variables, non-logical constants, and logical operators of the expressions of Ty2. The attributes whose interpretation is defined on them bear intuitively clear names. A functional application consists of a FUNCTOR and of an ARGUMENT. Since the functor and the argument in an application are meaningful expressions, the values of both attributes are again meaningful expressions. The same holds for the two arguments, ARG1 and ARG2, of equations, and for the two arguments of the binary logical operators (*l-const*). Quantifiers and (λ-) abstraction are different. Since quantification is quantification over variables and abstraction is abstraction over variables, the entities that model them bear an attribute, VAR, whose value is a variable, i.e., an entity of sort *var*. The second attribute that is defined on quantifiers and on abstractions is the attribute ARG or SCOPE, respectively. Both attributes have meaningful expressions as values. Finally, entities of sort *negation* have one argument, which is a meaningful expression.

The attribute TYPE is defined on every entity of sort *me*. Its value encodes the type of the expression and can be read as a subscript of the expression. Entities of sort *var* and entities of sort *const* have one more subscript. The second subscript is the natural number, n, that is used to number the variables (and constants) of each type. To number the variables and constants of each type, a representation of natural numbers is needed. With existential component quantification, a theory of natural numbers can easily be defined on the basis of a simple signature:

(140) The natural numbers:

n-number

 zero

 non-zero N-NUMBER *n-number*

The theory of the natural numbers:

$\textit{n-number} \rightarrow \ \exists x \ ^{x}[\textit{zero}]$

A natural number (*n-number*) is either zero (*zero*) or non-zero (*non-zero*). If a natural number is not zero, then it has a predecessor, which we find by interpreting the attribute N-NUMBER on it. The theory of entities of sort *n-number* says that after finitely many iterations of applying the function N-NUMBER to predecessors of a natural number, we will finally find the last predecessor, *zero*, of each natural number. 0 does not have a predecessor.

For 1, we find the last predecessor immediately. For 2, we find it after the second application of N-NUMBER, and so on.

Every meaningful expression has a subscript that encodes its type. A type is either atomic or a pair. There is a species for each atomic type, *entity* for the type e, *truth* for the type t, and *w-index* for the type s. Complex types are modeled by entities of sort *complex-type*. They have two attributes, IN and OUT, which are again *type*-valued. The entity under IN models the first type in the pair that constitutes a complex type, and the entity under OUT models the second type in the pair.

(141) The sort hierarchy of *type*:

$$\begin{array}{llll}
\textit{type} & & & \\
\quad \textit{atomic-type} & & & \\
\qquad \textit{entity} & & & \\
\qquad \textit{truth} & & & \\
\qquad \textit{w-index} & & & \\
\quad \textit{complex-type} & & \text{IN} & \textit{type} \\
 & & \text{OUT} & \textit{type}
\end{array}$$

The ontology of meaningful expressions that is given with an interpretation of the signature of (139), (140), and (141) may contain configurations of entities that obviously do not model expressions of Ty2. For example, the first and second argument of logical constants do not need to be of type t, contrary to DEFINITION 98. A grammar of Ty2 expressions must guarantee the proper types of the Ty2 expressions in the models of our grammar of the meaningful expressions of Ty2. In (142), I state the necessary type restrictions for meaningful expressions. The order in which they are given mirrors the order of the corresponding clauses of DEFINITION 98.

(142) Type restrictions on the logical operators:

$$\textit{appl} \rightarrow \begin{bmatrix} \text{TYPE} & \boxed{2} \\ \text{FUNC TYPE} & \begin{bmatrix} \text{IN} & \boxed{1} \\ \text{OUT} & \boxed{2} \end{bmatrix} \\ \text{ARG TYPE} & \boxed{1} \end{bmatrix},$$

$$\textit{abstr} \rightarrow \begin{bmatrix} \text{TYPE} & \begin{bmatrix} \text{IN} & \boxed{1} \\ \text{OUT} & \boxed{2} \end{bmatrix} \\ \text{VAR TYPE} & \boxed{1} \\ \text{ARG TYPE} & \boxed{2} \end{bmatrix},$$

$$equ \rightarrow \begin{bmatrix} \text{TYPE} & truth \\ \text{ARG1 TYPE} & \boxed{1} \\ \text{ARG2 TYPE} & \boxed{1} \end{bmatrix},$$

$$neg \rightarrow \begin{bmatrix} \text{TYPE} & truth \\ \text{ARG TYPE} & truth \end{bmatrix},$$

$$l\text{-}const \rightarrow \begin{bmatrix} \text{TYPE} & truth \\ \text{ARG1 TYPE} & truth \\ \text{ARG2 TYPE} & truth \end{bmatrix},$$

$$quant \rightarrow \begin{bmatrix} \text{TYPE} & truth \\ \text{SCOPE TYPE} & truth \end{bmatrix}$$

In an entity of sort *application* in a model of the grammar, the first element in the pair that is the complex type of the functor (its IN value) must be identical to the type of the argument. And the type of the *application* entity itself is identical to the OUT value of the type of the functor. Analogously, the other descriptions in (142) restrict the types of abstraction entities, equations, and so forth.

With the signature for meaningful expressions, types, and natural numbers, and the theory of natural numbers and the type restrictions on the logical operators, the RSRL grammar of Ty2 expressions is almost complete. We only need three more restrictions. Firstly, cyclic structures must be excluded, because cyclic expressions or cyclic types are not permitted in the syntax of Ty2. If they were not explicitly excluded, our grammar of Ty2 would license them. Secondly, all expressions must be finite: Each meaningful expression entity has a finite number of components. Again, the grammar would license infinite expressions if we did not explicitly exclude them. The third condition, adopted from Sailer 2000, is not strictly necessary, but it makes the correspondence between the representations of Ty2 expressions in models of our grammar and the regular notation of Ty2 expressions more transparent. The condition requires that in each meaningful expression, isomorphic configurations of entities be identical. For example, that means that there can only be one entity of sort *truth* or of the sort *w-index* in each expression; and each occurrence of the variable $v_{n,\langle e,t\rangle}$ in an expression will be modeled by the same configuration of entities. Loosely speaking, the principle that enforces the identity of isomorphic configurations of entities prevents spurious ambiguities in the representations of Ty2 expressions, because it entails that all the configurations of entities that model a certain expression form an indiscernibility class.

In the following paragraphs, I discuss these three additional restrictions in turn. For the principles, I presuppose that the sort hierarchy of the signature of the RSRL grammar of Ty2 contains a sort, *ty2*, that is an immediate supersort of the sorts *me*, *type*, and *n-number*.

Sailer 2000 shows how to formulate a General Non-Cyclicity Principle (GNP) over arbitrary signatures with a finite set of attributes. The GNP can be used to exclude cyclic structures in components of arbitrary entities in interpretations of a given signature. Informally, it says that for each component of each entity, the component is not the entity itself. By adding an appropriate antecedent, the GNP can be turned into a principle that excludes cyclic structures in entities in the denotation of some sort(s), as determined by the antecedent. Below I will formulate a principle that restricts the effect of the GNP to entities of sort *ty2*.

Sailer's General Non-Cyclicity Principle uses a binary relation, `component`. The appropriate meaning of `component` can be defined in grammars with signatures whose set of attributes is finite and whose set of relation symbols contains a symbol `component` that is assigned arity two by the arity function. Assume that $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$ is such a signature. The set of clauses that define `component` is the smallest set C such that

$$\texttt{component}(x, y) \overset{\forall}{\Longleftarrow} x = y \ \in C, \text{ and}$$

for each $\alpha \in \mathcal{A}$,

$$\texttt{component}(x, y) \overset{\forall}{\Longleftarrow} {}^{y}[\alpha\ \boxed{1}] \wedge \texttt{component}(x, \boxed{1}) \ \in C.$$

In the definition of the meaning of `component`, x is a component of y. The Ty2 Non-Cyclicity Principle, (143), uses that relation for requiring that no complex meaningful expression occur in any proper subexpression of itself. The principle is formulated with respect to a signature, $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, that includes the signature of Ty2 and has a finite set of attributes.

(143) The Ty2 Non-Cyclicity Principle:

$$ty2 \rightarrow$$

$$\forall \boxed{1} \left(\left(\bigvee_{\alpha \in \mathcal{A}} [\alpha\ \boxed{1}] \right) \rightarrow \neg\, \texttt{component}(:, \boxed{1}) \right)$$

The consequent of the principle in (143) is Sailer's GNP. The NON-CYCLICITY PRINCIPLE can be paraphrased as saying that for each entity, o, in the grammar of Ty2, if an attribute α is defined on o and $\boxed{1}$ is its value, then o is not a component of $\boxed{1}$. Informally, o is not cyclic.

Similarly, Sailer 2000 shows that it is possible to formulate a GENERAL FINITENESS PRINCIPLE (GFP) over signatures with a finite set of attributes. The GFP requires that every entity in every model of the grammar has finitely many components. Again, Sailer's general principle can easily be restricted to some subset of entities in the model of a grammar by turning it into the consequent of a principle with an appropriate antecedent. In our grammar of Ty2, an appropriate antecedent is again *ty2*.

(144) The TY2 FINITENESS PRINCIPLE:

$$\textit{ty2} \rightarrow$$
$$\exists \boxed{1} \forall \boxed{2} \Big(\texttt{component}(\boxed{2}, :) \rightarrow \texttt{member}\Big(\boxed{2}, \boxed{1}[\textit{chain}]\Big)\Big)$$

The crucial point of the TY2 FINITENESS PRINCIPLE is the reference to chains. Since chains are finite by definition, requiring that there is a chain of which every component of an entity is a member amounts to demanding that there only be finitely many components of that entity. Note that it is not necessary to require that every component occur only once on the chain. Repetitions of elements on the chain can be allowed without causing any harm. Note also that for *n-number* entities, (144) does not introduce any new restrictions, because they are already finite due to the theory of natural numbers, given in (140).

The TY2 IDENTITY PRINCIPLE follows the pattern of the preceding two principles. The GENERAL IDENTITY PRINCIPLE (GIP) of Sailer 2000 requires that whenever there is a pair of entities, o_1 and o_2, that have a common ancestor and that are the root of isomorphic configurations of entities, then o_1 and o_2 are identical. In other words, there is the maximal number of possible identities between the components of each entity. It is clear that the GIP is not an appropriate principle for all configurations of entities in the models of linguistic grammars. For example, if it were added to the grammar of Pollard and Sag 1994, it would require that in each sentence, each pair of indices whose NUMBER, GENDER and PERSON values are of the same species be identical. This would rule out all sentences in which the BINDING THEORY demands that two isomorphic indices be distinct. In the grammar of Ty2, however, an Identity Principle is useful. A very simple example may illustrate this. Consider the possible representations of the constant $c_{1,\langle e,\langle e,t\rangle\rangle}$ in models of the grammar of Ty2 as we have defined it so far. Without the Ty2 Identity Principle, models of the grammar may contain two non-isomorphic representations of $c_{1,\langle e,\langle e,t\rangle\rangle}$, one satisfying the description in (145a), and the other one satisfying the description in (145b).

(145) a.
$$\begin{bmatrix} \textit{const} & \\ \textsc{n-num}\ \textsc{n-num} & \textit{zero} \\ \textsc{type} & \begin{bmatrix} \textsc{in} & \boxed{1}\ \textit{entity} \\ \textsc{out} & \begin{bmatrix} \textsc{in} & \boxed{1}\ \textit{entity} \\ \textsc{out} & \textit{truth} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

b.
$$\begin{bmatrix} \textit{const} & \\ \textsc{n-num}\ \textsc{n-num} & \textit{zero} \\ \textsc{type} & \begin{bmatrix} \textsc{in} & \boxed{1}\ \textit{entity} \\ \textsc{out} & \begin{bmatrix} \textsc{in} & \boxed{2}\ \textit{entity} \\ \textsc{out} & \textit{truth} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \wedge \neg \boxed{1} = \boxed{2}$$

The only difference between the configurations under a *const* entity satisfying (145a) and (145b), respectively, is that those that satisfy (145a) have one *entity* component, and those that satisfy (145b) have two distinct *entity* components. Therefore, they are not isomorphic, although they represent the same non-logical constant. With the Ty2 Identity Principle, we eliminate the second possibility. It should again be stressed that eliminating the possibility of non-isomorphic representations of the same Ty2 expressions is not a logical necessity, as long as we are able to identify the equivalence classes of entities that represent any given Ty2 expression.[24] However, turning the representations into unique indiscernibility classes is aesthetically preferable and makes the correspondence proof more straightforward.

Sailer's GIP uses a relation that can be defined with respect to any signature with a finite set of attributes. I call that relation `copy`. Assume that $\langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$ is a signature with a finite set of attributes, $\mathcal{A}$, whose set of relation symbols contains the symbol `copy`, which is assigned arity two by the arity function of the signature. Following Sailer 2000, I define the meaning of `copy` in one clause:

[24]For example, Richter and Sailer 1999c does not introduce the identity requirement in its grammar of the semantic representation language Ty2U. Ty2U is a variant of Ty2 that allows for underspecified semantic representations.

(146) $\texttt{copy}(x, y) \overset{\forall}{\Longleftarrow}$

$$\bigvee_{\sigma \in \mathcal{S}} \left({}^{x}[\sigma] \wedge {}^{y}[\sigma] \right) \wedge$$
$$\bigwedge_{\alpha \in \mathcal{A}} \left(\forall \boxed{1} \left({}^{x}[\alpha\ \boxed{1}] \rightarrow \exists \boxed{2} \left({}^{y}[\alpha\ \boxed{2}] \wedge \texttt{copy}(\boxed{1}, \boxed{2}) \right) \right) \right)$$

y is a copy of x if x and y have the same species (second line), and if an attribute α with value $\boxed{1}$ is defined on x, then α is also defined on y, and $\boxed{1}$ and the value of α on y, $\boxed{2}$, are in the copy relation (third line).

Two entities that stand in the copy relation are of the same species, and they have the same attributes defined on them; and corresponding attribute values stand again in the copy relation. In the Ty2 Identity Principle, I require that all the corresponding entities in the copy relation of representations of Ty2 expressions be identical:

(147) The Ty2 Identity Principle:

$$\textit{ty2} \rightarrow$$
$$\forall \boxed{1} \forall \boxed{2} \left(\texttt{copy}(\boxed{1}, \boxed{2}) \rightarrow \boxed{1} = \boxed{2} \right)$$

The consequent of my principle is Sailer's GIP.

The Ty2 Identity Principle concludes the specification of the RSRL grammar of Ty2 expressions. Sailer 2000 proves the crucial properties of the RSRL grammar of Ty2. Firstly, there is an exhaustive model of the grammar whose universe contains exactly the natural numbers, the set of types, *Types*, and the meaningful expressions of Ty2. This result shows that the grammar indeed specifies what we want to specify with it. Secondly, for each indiscernibility class of entities of sort *me* in an exhaustive model of the grammar, there exists a corresponding Ty2 expression that has the same meaning. To prove this result, it is of course necessary to define the notion of *correspondence* between indiscernibility classes of *me* entities and Ty2 expressions, and to define a semantics for indiscernibility classes of *me* entities. In fact, both definitions are straightforward: Equivalence classes of indiscernible *me* entities receive the standard Ty2 semantics, and the correspondence is defined in the way that is intuitively expected. For example, an entity in an exhaustive model of the Ty2 grammar that is described by (145a) corresponds to the constant $c_{1,\langle e,\langle e,t\rangle\rangle}$. Thirdly, for each Ty2 expression there exists an AVM description that picks out exactly the indiscernibility class of entities in models of the RSRL grammar of Ty2 whose *me* entities correspond to the Ty2 expression. It is this result that allows us to use Ty2 expressions in AVM descriptions in places where the AVM syntax would require a description of the Ty2 expression. Since those descriptions are very cumbersome to write and to read, the abbreviatory convention of notating the described expression instead is very convenient.

With the Ty2 grammar in place, I can formulate a first approximation to a SEMANTICS PRINCIPLE (SP). For expository reasons, I assume a binary branching syntactic structure, where every phrase has a head daughter and exactly one nonhead daughter. It is not hard to generalize this specialized SP to different assumptions about the syntactic phrase structure. The SP should license semantic composition by functional application. This leaves two possibilities: Either the head daughter is the functor, and the nonhead daughter is the argument, or *vice versa.* The type of the semantic representations of the daughters determines which one of the two possibilities holds. If neither one of the two cases applies, the SP in (148) excludes the phrase from models of the grammar.

(148) SEMANTICS PRINCIPLE (first approximation)

$$
\textit{phrase} \;\rightarrow
$$

$$
\left(
\begin{array}{l}
\begin{bmatrix}
\textsc{synsem} & \begin{bmatrix}\textsc{loc content lf} & \begin{bmatrix}\textit{application} \\ \textsc{func}\ \boxed{3} \\ \textsc{arg} & \begin{bmatrix}\textit{abstraction} \\ \textsc{var} & \begin{bmatrix}\textsc{type} & \textit{w-index} \\ \textsc{n-num} & \textit{zero}\end{bmatrix} \\ \textsc{arg}\ \boxed{4} \end{bmatrix}\end{bmatrix}\end{bmatrix} \\
\textsc{dtrs} & \begin{bmatrix}\textsc{head-dtr synsem loc content lf} & \boxed{1} \\ \textsc{nonhead-dtr synsem loc content lf} & \boxed{2}\end{bmatrix}
\end{bmatrix} \\
\wedge \left(
\begin{array}{c}
(\boxed{1} = \boxed{3} \quad \wedge \quad \boxed{2} = \boxed{4}) \\
\vee \\
(\boxed{1} = \boxed{4} \quad \wedge \quad \boxed{2} = \boxed{3})
\end{array}
\right)
\end{array}
\right)
$$

If $\boxed{1} = \boxed{3}$ and $\boxed{2} = \boxed{4}$, then the head daughter is the functor, and the nonhead daughter is the argument. If $\boxed{1} = \boxed{4}$ and $\boxed{2} = \boxed{3}$, then the nonhead daughter is the functor, and the head daughter is the argument. (148) also shows that the standard analysis of functional application in natural language semantics in Ty2 involves lambda abstraction over the world index of the argument. For that reason, the ARG value of the *application* is an *abstraction.* To ensure that the SP abstracts over the right world index, I assume with Groenendijk and Stokhof 1982, p. 187, that all our basic expressions, i.e., the expressions in the lexical entries of the linguistic grammar, contain the same free index variable. The SP abstracts over the world index variable that is the world index variable of the lexical entries. Like Groenendijk and Stokhof, I reserve the first variable of type s, $v_{0,s}$ for that purpose. In the Ty2 notation below, I write w for that variable.

The example in (149) illustrates the effect of the SP with the nominal phrase *every unicorn*. The type of the semantic expressions determines that the syntactic nonhead, *every*, must be the functor, and *unicorn* is the argument. In addition, the SP introduces the lambda abstraction over the world index, w, of *unicorn*.

(149) *An illustration of the* SEMANTICS PRINCIPLE

NP

$$\begin{bmatrix} \text{PHON} & \langle \textit{every, unicorn} \rangle \\ \text{SS LOC CONT LF} & \lambda P \lambda Q.\forall x[P(w)(x) \rightarrow Q(w)(x)](\lambda w.\mathsf{unicorn}'(w)) \end{bmatrix}$$

C — Det

$$\begin{bmatrix} \text{PHON} & \langle \textit{every} \rangle \\ \text{SS LOC CONT LF} & \lambda P \lambda Q.\forall x[P(w)(x) \rightarrow Q(w)(x)] \end{bmatrix}$$

H — N

$$\begin{bmatrix} \text{PHON} & \langle \textit{unicorn} \rangle \\ \text{SS LOC CONT LF} & \mathsf{unicorn}'(w) \end{bmatrix}$$

In the introductory example, (138), I have indicated that my goal is to specify the principles of semantics in a way that leads to a fully β-reduced expression when the logical forms of two phrases are combined by functional application. As we can see at the top node in (149), this is not the case yet with the SP in (148). If we want the expressions that represent the meaning of linguistic signs to be redex free, we need to add a mechanism that expresses β-reduction.

5.4.2 An Operational Extension

In this section, I sketch an extension of the grammar of Ty2 that furnishes our logical forms with β-reduction. This operational extension of Ty2 is another example of the descriptive power of RSRL. It shows how a technically desirable mechanism can be made part of a declarative grammar. I proceed in three steps: First, I anchor the sorts and attributes of the signature of the meaningful expressions of Ty2 and of the theory of β-reduction in the standard signature of HPSG grammars by declaring the appropriate attributes and attribute values for *content*, and I specify the theory of β-reduction as a principle in the theory of the grammar. Second, I introduce a relation that allows me to require that a meaningful expression be redex free. The new relation leads directly to the β-REDUCTION PRINCIPLE, which relies on the theory of β-reduction specified before. The β-REDUCTION PRINCIPLE requires that all LOGICAL-FORM values of *content* entities be redex free. Third, the SEMANTICS PRINCIPLE is modified to respect the addition of β-reduction to the grammar.

My specification of β-reduction essentially follows the proposal by Richter and Sailer (1999a), who choose to implement β-reduction as a recursive sort within CONTENT values. To provide the "work space" for β-reduction, two attributes are appropriate to *content*: The attribute LOGICAL-FORM, which we have already seen above, and the attribute β-REDUCTION. The meaningful expressions of Ty2 that determine the model-theoretic meaning of a linguistic sign are the value of the attribute LOGICAL-FORM. A new sort, *β-reduction*, is appropriate for the attribute β-REDUCTION. With the theory of the sort *β-reduction*, to which I will turn presently, the β-REDUCTION values in models of the grammar are recursively structured configurations of *β-reduction* entities that specify the β-reduction steps of complex expressions.[25]

(150) The sort hierarchies of *content* and of *β-reduction*:

content	LOGICAL-FORM	*me*
	β-REDUCTION	*β-reduction*

β-reduction	TERM	*me*
	AUX	*me*

    no-reduction

    one-reduction REC *β-reduction*

[25]The present encoding of *β-reduction* as recursively structured configurations of entities bears a noticeable similarity to the junk slot method of encoding relations that is discussed in Section 2.2.3. As mentioned there, a reification of relations as configurations of entities is regarded a serious conceptual problem that keeps linguists from using junk slots, and the same argument could be leveled against the related encoding of *β-reduction*. Fortunately, Sailer 2000 shows how in RSRL *β-reduction* can alternatively be encoded as chains in the arguments of relations, i.e., Sailer's encoding avoids additional attributes and entities of new species in signs, and the reduction steps are no longer present in the components of signs. The reason I do not follow Sailer's construction here is that the present approach is technically much simpler. My choice should not be taken as an indication of preference.

According to the principle formulated in (151) below, the value of the attribute TERM is an expression that is either identical to the expression under AUX, or it is derived from the AUX value by β-reduction. The idea of the recursive structure of *β-reduction* is that if no reduction step is performed, the AUX value equals the TERM value. If there is a reduction, the REC AUX value is deduced from the AUX value by one reduction step, and the TERM value is passed up from the REC TERM value, i.e., ultimately it is passed up from the bottom of the reduction chain deeper down in the recursive embedding of *β-reduction* entities.

(151) The theory of *β-reduction*:

$$\textit{no-reduction} \rightarrow \begin{bmatrix} \textsc{term} & \boxed{1} \\ \textsc{aux} & \boxed{1} \end{bmatrix}$$

$$\begin{aligned} \textit{one-reduction} \rightarrow\ & \begin{bmatrix} \textsc{term} & \boxed{1} \\ \textsc{aux} & \boxed{2} \\ \textsc{rec} & \begin{bmatrix} \textsc{term} & \boxed{1} \\ \textsc{aux} & \boxed{3} \end{bmatrix} \end{bmatrix} \\ & \wedge\ \texttt{component}\left({}^{y}\begin{bmatrix} \textit{appl} & \\ \textsc{func} & \begin{bmatrix} \textit{abstr} & \\ \textsc{var} & \boxed{5} \\ \textsc{arg} & \boxed{4} \end{bmatrix} \\ \textsc{arg} & \boxed{6} \end{bmatrix}, \boxed{2} \right) \\ & \wedge\ \texttt{replace}(\boxed{4}, \boxed{5}, \boxed{7}, \boxed{6}) \\ & \wedge\ \texttt{replace}(\boxed{2}, y, \boxed{3}, \boxed{7}) \end{aligned}$$

In the non-recursive case, the TERM value and the AUX value are simply identical. In the recursive case, the β-reduced TERM value is passed up from REC TERM, and the reduction step is taken care of by the relations `component` and `replace`. The relation `component` is used to identify a redex sub-expression, y, of $\boxed{2}$. Informally, y is a redex in the input formula, $\boxed{2}$, which is the AUX value of the *one-reduction* entity. $\boxed{7}$ is a sub-expression of the REC AUX value, $\boxed{3}$, such that $\boxed{7}$ is obtained from y by β-reduction. In the general case, the replacement of the variable, $\boxed{5}$, by $\boxed{6}$ in the argument of the abstraction, $\boxed{4}$, is not the only replacement. The second occurrence of the `replace` relation ensures that we get the overall β-reduced expression, $\boxed{3}$, by replacing the former abstraction, y, in $\boxed{2}$, by the expression, $\boxed{7}$, that we obtain from contracting y.

The real work is done by the relation `replace`, whose definition is sketched in (152).

(152) The relation `replace`:

The base cases:

$$\texttt{replace}(x, y, v, w) \overset{\forall}{\Longleftarrow} x = y \ \wedge\ v = w$$

$$\texttt{replace}(x, y, v, w) \overset{\forall}{\Longleftarrow} \neg x = y \ \wedge\ {}^{x}[\mathit{var} \vee \mathit{const}] \wedge x = v$$

An example of a recursive case:

$$\texttt{replace}(x, y, v, w) \overset{\forall}{\Longleftarrow} \neg x = y \ \wedge\ {}^{x}\begin{bmatrix} \mathit{abstr} & \\ \text{TYPE} & \boxed{1} \\ \text{VAR} & \boxed{2} \\ \text{ARG} & \boxed{3} \end{bmatrix} \wedge\ {}^{v}\begin{bmatrix} \mathit{abstr} & \\ \text{TYPE} & \boxed{1} \\ \text{VAR} & \boxed{2} \\ \text{ARG} & \boxed{4} \end{bmatrix} \wedge\ \texttt{replace}(\boxed{3}, y, \boxed{4}, w)$$

v is obtained by replacing y by w in x. The recursion goes over the structure of the expression x. There are two base cases: If x is identical to y, then the reduced version of y, namely w, is identical to v. If x is not identical to y, but an atomic expression, i.e., either a variable or a constant, then x is identical to v. In this case, we were checking a sub-expression of x that did not contain y. In the recursive case, the structure of x is copied to v, and the relation `replace` is applied to all recursive sub-expressions of x.

Note that as it stands, our β-reduction differs from the standard definition in textbooks:[26] We never rename bound variables or, in other words, α-conversion is not provided by the above specification of the theory of β-reduction. Except for the variable whose occurrences are replaced in the reduction, the variables in the redex and in the contracted expression are identical. Variables that are free in the argument of a β-redex may, thus, become bound in the contracted expression. It follows that our β-reduction does not necessarily preserve the meaning of an expression. In (153), I illustrate the (potential) problem. The symbol '$\triangleright_{1\beta}$' indicates that one β-reduction step is performed to get from the expression to its left to the expression to its right. The symbol '$\not\triangleright_{1\beta}$ signals non-standard β-reduction as permitted by (151). The symbol '$\equiv_\alpha$' signals α-conversion, sometimes also called a change of a bound variable.

(153) a. $(\lambda v_{1,e}.\exists v_{2,e}\, \mathsf{pred}(v_{2,e}, v_{1,e}))\,(v_{2,e}) \ \not\triangleright_{1\beta}\ \exists v_{2,e}\, \mathsf{pred}(v_{2,e}, v_{2,e})$

[26] For example, see Hindley and Seldin 1986, p. 11.

b. $(\lambda v_{1,e}.\exists v_{2,e}\, \mathsf{pred}(v_{2,e}, v_{1,e}))\, (v_{2,e})$
$\equiv_\alpha$ $(\lambda v_{1,e}.\exists v_{3,e}\, \mathsf{pred}(v_{3,e}, v_{1,e}))\, (v_{2,e})$
$\rhd_{1\beta}$ $\exists v_{3,e}\, \mathsf{pred}(v_{3,e}, v_{2,e})$

The first example, (153a), is an instance of the pseudo-β-reduction licensed by our theory (151) but not licensed by standard β-reduction, as indicated by '$\not\rhd_{1\beta}$.' This pseudo-reduction step does not preserve the meaning of the expression, because the variable $v_{2,e}$ that is free in the argument of the redex in the first expression is bound by the existential quantification in the expression to the right of '$\not\rhd_{1\beta}$.' The second example, (153b), is an instance of standard β-reduction, which involves α-conversion, i.e., a change of the variable bound by the existential quantifier. Procedurally speaking, we first replace the bound occurrences of the variable, $v_{2,e}$, by $v_{3,e}$, thus obtaining the expression $(\lambda v_{1,e}.\exists v_{3,e}\, \mathsf{pred}(v_{3,e}, v_{1,e}))\, (v_{2,e})$, and then we perform the β-reduction.

Sailer 2000 shows how (151) can be modified to forbid the non-standard β-reduction indicated in (153a). For that purpose, Sailer enriches the sort hierarchy under *one-reduction* with two additional subsorts of *one-reduction*, *β1-reduction* and *α-conversion*. The former theory of *one-reduction* becomes the theory of *β1-reduction* by adding a conjunct to the consequent of the principle that blocks contraction if an unbound variable would get into the scope of some logical operator, and enforces α-conversion instead. *α-conversion* is of course governed by an appropriate principle that restricts the admissible configurations under *α-conversion* entities. I omit Sailer's extension here, since it does not add any new insights into the descriptive flexibility of RSRL to what we have already seen.[27]

Given the theory of the sort *β-reduction*, we only need to define one more auxiliary relation, `redex-free`, before we can state a principle that demands that the LOGICAL-FORM value of each *content* entity be redex free. The definition of `redex-free` in (154) uses the relation `component`, defined on page 358.

[27] It should be mentioned that the linguistic theories in Richter and Sailer 1999a,b deliberately exploit the non-standard definition of β-reduction given in (151). Some analyses of indefinites in natural languages postulate that their logical forms contain free variables that must be bound by a mechanism of existential closure within certain syntactic domains. Richter and Sailer show that the non-standard β-reduction offers the possibility of introducing the necessary existential closure by appropriate lexical elements, such as Polish finite verbs with the negation prefix *nie* (Richter and Sailer, 1999b, pp. 272–274).

(154) The relation `redex-free`:

$$\texttt{redex-free}(x) \stackrel{\forall}{\Longleftarrow} \neg\, \exists y \left({}^{y}\begin{bmatrix} appl \\ \text{FUNC} \;\; abstr \end{bmatrix} \wedge \texttt{component}(y, {}^{x}[me]) \right)$$

According to (154), a meaningful expression x is redex free exactly if it does not contain an application whose functor is an abstraction.

The β-REDUCTION PRINCIPLE, (155), requires that the LF value of a *content* entity be always fully β-reduced. It equals the "output" of our mechanism for β-reduction, the β-REDUCTION TERM value of the *content* entity:

(155) The β-REDUCTION PRINCIPLE:

$$content \rightarrow \left(\begin{bmatrix} \text{LOGICAL-FORM} \;\; \boxed{1} \\ \beta\text{-REDUCTION TERM} \;\; \boxed{1} \end{bmatrix} \wedge \texttt{redex-free}(\boxed{1}) \right)$$

We are now ready to reformulate the SEMANTICS PRINCIPLE. Recall that, for ease of exposition, we assume an overall binary branching syntactic structure when we state the SP. For each phrase, the SP provides the possibility of functional application of one syntactic daughter to the other. In conjunction with the theory of *β-reduction*, (151), and the β-REDUCTION PRINCIPLE, (155), the new SP says that the LOGICAL-FORM value of a phrase is the redex free expression that is derived by β-reduction from the expression obtained by functional application of the LOGICAL-FORM value of one syntactic daughter to the LOGICAL-FORM value of the other syntactic daughter of the phrase. As already explained in connection with the first formulation of the SP, we always abstract over the world index, $v_{0,s}$, of the argument of the functional application.

(156) *The* SEMANTICS PRINCIPLE *(final version)*

$$phrase \rightarrow \begin{bmatrix} \text{SS} & \begin{bmatrix} \text{LOC CONT } \beta\text{-REDUCT} & \begin{bmatrix} \beta\text{-}reduction \\ \text{AUX} & \begin{bmatrix} application \\ \text{FUNC} \;\; \boxed{3} \\ \text{ARG} & \begin{bmatrix} abstraction \\ \text{VAR} & \begin{bmatrix} \text{TYPE} & w\text{-}index \\ \text{N-NUM} & zero \end{bmatrix} \\ \text{ARG} \;\; \boxed{4} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{DTRS} & \begin{bmatrix} \text{HEAD-DTR SS LOC CONT LF} & \boxed{1} \\ \text{NONHEAD-DTR SS LOC CONT LF} & \boxed{2} \end{bmatrix} \end{bmatrix}$$
$$\wedge \left(\left(\boxed{1} = \boxed{3} \wedge \boxed{2} = \boxed{4}\right) \vee \left(\boxed{1} = \boxed{4} \wedge \boxed{2} = \boxed{3}\right) \right)$$

The formulation of the SP in (156) leads to the desired result, as illustrated in (157):

(157) *An illustration of the final* SEMANTICS PRINCIPLE

$$\text{NP}$$

$$\begin{bmatrix} \textsc{phon} & \langle \textit{every, unicorn} \rangle \\ \textsc{ss loc cont} & \begin{bmatrix} \textit{content} & \\ \textsc{lf} & \boxed{1}\ \lambda Q.\forall x[\mathsf{unicorn}'(w)(x) \rightarrow Q(w)(x)] \\ \beta\text{-}\textsc{reduction} & \begin{bmatrix} \textsc{term} & \boxed{1} \\ \textsc{aux} & (\boxed{2})(\lambda w\ \boxed{3}) \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

C: Det

$$\begin{bmatrix} \textsc{phon} & \langle \textit{every} \rangle \\ \textsc{ss loc cont lf} & \boxed{2}\ \lambda P \lambda Q.\forall x[P(w)(x) \rightarrow Q(w)(x)] \end{bmatrix}$$

H: N

$$\begin{bmatrix} \textsc{phon} & \langle \textit{unicorn} \rangle \\ \textsc{ss loc cont lf} & \boxed{3}\ \mathsf{unicorn}'(w) \end{bmatrix}$$

The semantic types of the quantifier and of the common noun determine that the logical form of the former is applied to the logical form of the latter. This application can be observed in the β-REDUCTION AUX value of the *content* of the mother node. The β-REDUCTION PRINCIPLE in combination with the theory of β-reduction leads to the fully contracted expression in the β-REDUCTION TERM value, which is in turn identified with the LOGICAL-FORM value of the mother node (due to the β-REDUCTION PRINCIPLE).

5.4.3 A Principle at the Syntax-Semantics Interface

I conclude the discussion of the integration of two-sorted type theory into HPSG grammars with an example of a principle that illustrates how the interaction of syntax and semantics can be accounted for in the present architecture. It demonstrates how to express restrictions on logical form straightforwardly as restrictions on expressions of Ty2.

It is well-known that every arbitrary logical expression may not be the logical form of a matrix sentence. Richter and Sailer 1999a,b subsume restrictions on the possible logical form of matrix sentences under their PRINCIPLE OF FULL INTERPRETATION (PFI). The PFI imposes well-formedness conditions on the possible logical forms of certain distinguished expressions of a natural language. Richter and Sailer's PFI restricts the logical form of unembedded signs, i.e., expressions that carry illocutionary force.[28] Matrix sentences are a prototypical instance of unembedded signs.

(158) *The* PRINCIPLE OF FULL INTERPRETATION *(PFI) (after Richter and Sailer 1999a,b)*

In the logical form of an unembedded sign,

1. every quantifier binds an occurrence of a variable, and
2. except for the world index variable, $v_{0,s}$, there are no free variables.

The PFI has a number of non-trivial implications for unembedded sentences. For example, those theories in which indefinites are analyzed as introducing free variables must provide a mechanism for binding off these variables in unembedded sentences. This prompts the question where the logical operators that do that come from, where they can enter into the logical form of a sentence, and which principles control their occurrence. Here, I will not pursue questions of the empirical and technical adequacy of the PFI. In the present discussion, the PFI is only meant as an illustration of how principles at the interface between the logical form of signs and some module of the grammar outside of semantics are formalized when the logical form of a sign is an expression of some logical language.

What is formally interesting about the PFI is that for some syntactic environments (unembedded signs), it imposes complex restrictions on the shape of the expressions that may occur as their logical form. Since these expressions determine the meaning of a sign, the PFI also restricts the possible meanings of unembedded signs. Just like the principles of Pollard and Sag 1994 restrict admissible configurations of syntactic or situation-semantic entities, the PFI looks at logical forms as configurations of entities on which it imposes well-formedness conditions.

The kind of principle that we see in (158) is reminiscent of the QUANTIFIER BINDING CONDITION and of the BINDING THEORY. Where the QUANTIFIER BINDING CONDITION declares that every occurrence of a certain index must be captured by a certain quantifier, the first clause of the

[28]The PFI in (158) differs slightly from the version given in Richter and Sailer 1999a,b. In particular, I omit their third clause that is motivated by specific technical assumptions of their grammars of French and Polish.

PFI says that every quantifier binds at least one variable; and where Principle A of the BINDING THEORY declares that certain anaphors must be locally o-bound, the second clause of the PFI says that with one exception, all variables in the logical form of unembedded signs must be bound by a quantifier. Given the conceptual similarities between the principles, and since it is not hard to express the QUANTIFIER BINDING CONDITION and Principle A of the BINDING THEORY in RSRL, a formalization of the PFI does not pose any problems either.

For the formalization of (158), we need one new binary relation, `free`, which relates variables to the complex expressions in which they occur free. In (159), x occurs free in y:

(159) The relation `free`:

$$\texttt{free}(x, y) \stackrel{\forall}{\Longleftarrow} x = y \wedge {}^{y}\big[\textit{var}\big]$$

$$\texttt{free}(x, y) \stackrel{\forall}{\Longleftarrow} {}^{y}\begin{bmatrix} \textit{appl} & \\ \text{FUNC} & \boxed{1} \\ \text{ARG} & \boxed{2} \end{bmatrix} \wedge \big(\texttt{free}(x, \boxed{1}) \vee \texttt{free}(x, \boxed{2})\big)$$

$$\texttt{free}(x, y) \stackrel{\forall}{\Longleftarrow} {}^{y}\begin{bmatrix} \text{ARG1} & \boxed{1} \\ \text{ARG2} & \boxed{2} \end{bmatrix} \wedge \big(\texttt{free}(x, \boxed{1}) \vee \texttt{free}(x, \boxed{2})\big)$$

$$\texttt{free}(x, y) \stackrel{\forall}{\Longleftarrow} {}^{y}\begin{bmatrix} \textit{neg} & \\ \text{ARG} & \boxed{1} \end{bmatrix} \wedge \texttt{free}(x, \boxed{1})$$

$$\texttt{free}(x, y) \stackrel{\forall}{\Longleftarrow} {}^{y}\begin{bmatrix} \textit{quant} & \\ \text{VAR} & \boxed{1} \\ \text{SCOPE} & \boxed{2} \end{bmatrix} \wedge \neg x = \boxed{1} \wedge \texttt{free}(x, \boxed{2})$$

$$\texttt{free}(x, y) \stackrel{\forall}{\Longleftarrow} {}^{y}\begin{bmatrix} \textit{abstr} & \\ \text{VAR} & \boxed{1} \\ \text{ARG} & \boxed{2} \end{bmatrix} \wedge \neg x = \boxed{1} \wedge \texttt{free}(x, \boxed{2})$$

With the relation `free` in hand, I can express the PFI without any further definitions. The only other relation that is needed is the relation `component`, which was defined on page 358.

(160) *The* Principle of Full Interpretation *formalized (after Richter and Sailer 1999a, p. 289)*

$$
\begin{array}{l}
\textit{unembedded-sign} \quad \rightarrow \\
\left(\begin{array}{l}
\left[\textsc{ss loc cont lf}\ \boxed{1}\right] \\
\wedge\ \forall x \left(\begin{array}{l}
\mathtt{component}\left({}^{x}[\textit{quant}],\ \boxed{1}\right) \\
\rightarrow \exists\boxed{3}\,\exists\boxed{2}\ \left({}^{x}\begin{bmatrix}\textsc{var} & \boxed{3} \\ \textsc{scope} & \boxed{2}\end{bmatrix} \wedge\ \mathtt{free}(\boxed{3}, \boxed{2})\right)
\end{array}\right) \\
\wedge\ \forall x \left(\mathtt{component}\left({}^{x}[\textit{var}], \boxed{1}\right) \rightarrow \left({}^{x}\begin{bmatrix}\textit{var} & \\ \textsc{n-number} & \textit{zero} \\ \textsc{type} & \textit{w-index}\end{bmatrix} \vee\ \neg\,\mathtt{free}(x,\ \boxed{1})\right)\right)
\end{array}\right)
\end{array}
$$

The second conjunct in the consequent says that if a *quantifier* entity, x, in the logical form, $\boxed{1}$, of an unembedded sign, quantifies over the variable $\boxed{3}$, then $\boxed{3}$ occurs free in the expression, $\boxed{2}$, that is the SCOPE value of the quantifier. In other words, each quantifier binds a variable. The third, and last, conjunct says that for each *variable* entity in the logical form, $\boxed{1}$, of an unembedded sign, the variable does not occur free in $\boxed{1}$, with the possible exception of the world index variable, $v_{0,s}$. In other words, no variable except for $v_{0,s}$ may occur unbound in the logical form of an unembedded sign.

5.4.4 Summary

In this section, we saw a number of technical principles that tell us more about the expressive power of RSRL. Inasmuch as RSRL is a formal explication of the formalism that is implicit in Pollard and Sag 1994, they also tell us more about possible HPSG 94 grammars of natural languages.

The main result of this section is that HPSG 94 is sufficiently expressive to allow for the specification of symbolic languages like Ty2, including an operational extension for β-reduction and α-conversion as they are usually defined in the λ-calculus. The linguistic significance of this result is that without an extension of the formalism that is implicit in HPSG 94 from the very beginning, HPSG grammars may employ expressions of standard logical languages, just as it is usually done elsewhere in the literature on the formal semantics of natural languages. It is therefore possible to write rigorously formalized HPSG grammars of natural languages that comprise sophisticated syntactic structures as well as expressions of standard logical languages with well-studied model-theoretic interpretations. The integration of Ty2 with the semantically uninterpreted syntactic structures of HPSG grammars offers new perspectives to grammar writers, allows for a more direct comparison of the semantic structures of HPSG grammars with the semantic structures of other linguistic theories that employ logical forms, and makes the empirical and analytical insights of the pertinent literature more easily applicable to the design of HPSG grammars. With Richter and Sailer 1999a and Richter and Sailer 1999b, there already exist two studies that illustrate this new line of research.

The specification of a grammar of Ty2 with an operational extension crucially requires the use of all extensions that RSRL adds to SRL. Existential and universal component quantification, relations in the scope of negation, and chains are indispensable for the formalization of the semantic representations that I discussed above. Two technical principles that are needed in the specification of Ty2 deserve to be mentioned in this context, since they are potentially relevant for other grammatical specifications as well. The Ty2 Non-Cyclicity Principle, (143), showed that, as long as the set of attributes of the signature of a grammar is finite, cyclic structures can be forbidden either throughout all models of the grammar or in designated parts of the models. For example, cyclic structures might be allowed everywhere except for the expressions of Ty2. The Ty2 Finiteness Principle, (144), showed that, under the same restriction on the signatures of a grammar, entities with infinitely many components can be excluded from all models of the grammar. Apart from being an important formal result about the expressive power of a formalism of HPSG 94, a general finiteness principle might, at least to some linguists, be a theoretically welcome constraint on models of natural languages.

Appendix A

Existence Proof of Exhaustive Models of RSRL grammars

This appendix contains the details of the proof that shows that for every RSRL grammar Γ, there exists an exhaustive Γ model. Since the idea of the proof is explained in Section 3.1.2, the comments are kept very brief, and they are restricted to those definitions and propositions that are not addressed in Sections 3.1.2.1–3.1.2.3. Nevertheless, I will repeat all definitions and propositions, lemmata etc. for convenience. The numbering is identical to the numbering in Section 3.1.2. The mathematics reported here is due to Richter and King 1997.

Section A.1 shows that there is an algebraic characterization of exhaustive models in terms of simulation that is equivalent to the characterization provided by Definition 64. This result will be needed in Section A.2, where for each grammar, a model is defined that is shown to be an exhaustive model of the grammar.

A.1 A Different Characterization of Exhaustive Models

Recall that that for each set S, I write $\overline{S}$ for $(S \cup S^*)$, $\overline{S}^*$ for $(\overline{S})^*$, and S^{**} for $(S^*)^*$.

Definition 65 *For each signature Σ, for each Σ interpretation $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1 \rangle$, for each $u_1 \in \mathsf{U}_1$, for each Σ interpretation $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2 \rangle$,* *for each* $u_2 \in \mathsf{U}_2$,

> f *is a* ***congruence from*** $\langle u_1, \mathsf{I}_1\rangle$ ***to*** $\langle u_2, \mathsf{I}_2\rangle$ ***in*** Σ
> *iff* f *is a bijection from* $\overline{\mathsf{Co}}^{u_1}_{\mathsf{I}_1}$ *to* $\overline{\mathsf{Co}}^{u_2}_{\mathsf{I}_2}$,
> *for each* $u \in \overline{\mathsf{Co}}^{u_1}_{\mathsf{I}_1}$, $\widehat{\mathsf{S}_1}(u) = \widehat{\mathsf{S}_2}(f(u))$,
> *for each* $\alpha \in \widehat{\mathcal{A}}$, *for each* $u \in \overline{\mathsf{Co}}^{u_1}_{\mathsf{I}_1}$,
> $\widehat{\mathsf{A}_1}(\alpha)(u)$ *is defined iff* $\widehat{\mathsf{A}_2}(\alpha)(f(u))$ *is defined, and*
> *if* $\widehat{\mathsf{A}_1}(\alpha)(u)$ *is defined then* $f(\widehat{\mathsf{A}_1}(\alpha)(u)) = \widehat{\mathsf{A}_2}(\alpha)(f(u))$,
> *for each* $\rho \in \mathcal{R}$, *for each* $o_1 \in \overline{\mathsf{Co}}^{u_1}_{\mathsf{I}_1}, \ldots,$ *for each* $o_{\mathcal{AR}(\rho)} \in \overline{\mathsf{Co}}^{u_1}_{\mathsf{I}_1}$,
> $\langle o_1, \ldots, o_{\mathcal{AR}(\rho)}\rangle \in \mathsf{R}_1(\rho)$ *iff* $\langle f(o_1), \ldots, f(o_{\mathcal{AR}(\rho)})\rangle \in \mathsf{R}_2(\rho)$, *and*
> $f(u_1) = u_2$.

A first entity in a first interpretation and a second entity in a second interpretation are congruent iff there is a (quasi-) species, (quasi-) attribute and relation preserving bijection from the components and chains of components of the first entity in the first interpretation to the components and chains of components of the second entity in the second interpretation such that the bijection maps the first entity to the second entity.

Definition 66 *For each signature* Σ, *for each* Σ *interpretation* $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1\rangle$, *for each* $u_1 \in \mathsf{U}_1$, *for each* Σ *interpretation* $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2\rangle$, *for each* $u_2 \in \mathsf{U}_2$,

> $\langle u_1, \mathsf{I}_1\rangle$ *and* $\langle u_2, \mathsf{I}_2\rangle$ *are* ***congruent in*** Σ
> *iff for some* f, f *is a congruence from* $\langle u_1, \mathsf{I}_1\rangle$ *to* $\langle u_2, \mathsf{I}_2\rangle$ *in* Σ.

Definition 67 *For each signature* Σ, *for each* Σ *interpretation* $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1\rangle$, *for each* $u_1 \in \mathsf{U}_1$, *for each* Σ *interpretation* $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2\rangle$, *for each* $u_2 \in \mathsf{U}_2$,

> $\langle u_1, \mathsf{I}_1\rangle$ *and* $\langle u_2, \mathsf{I}_2\rangle$ *are* ***indiscernible in*** Σ
> *iff for each* $\delta \in \mathcal{D}^{\Sigma}_0$, $u_1 \in D_{\mathsf{I}_1}(\delta)$ *iff* $u_2 \in D_{\mathsf{I}_2}(\delta)$.

I use a standard definition of functional composition:

Definition 99 *For each set U, for each set V, for each set W, for each total function f from U to V, for each total function g from V to W,*

> *$g \circ f$ is the function from U to W such that, for each $u \in U$,*
>
> $g \circ f(u) = g(f(u))$.

By PROPOSITION 9 the following definition of the extended path interpretation functions in interpretations is well-made. They will be needed presently to show that congruence and indiscernibility are equivalent.

Definition 81 *For each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, T_{I} is the partial function from the Cartesian product of $\overline{\mathcal{A}^*}$ and U to $\overline{\mathsf{U}}$ such that*

> *for each $\pi \in \mathcal{A}^*$, for each $u \in \mathsf{U}$,*
>
> > *$T_{\mathsf{I}}(\pi, u)$ is defined iff for some $ass \in Ass_{\mathsf{I}}$, $T_{\mathsf{I}}^{ass}(:\pi)(u)$ is defined, and*
> > *if $T_{\mathsf{I}}(\pi, u)$ is defined then for some $ass \in Ass_{\mathsf{I}}$, $T_{\mathsf{I}}(\pi, u) = T_{\mathsf{I}}^{ass}(:\pi)(u)$,*
>
> *for each $\langle \pi_1, \ldots, \pi_n \rangle \in \mathcal{A}^{**}$, for each $u \in \mathsf{U}$,*
>
> > *$T_{\mathsf{I}}(\langle \pi_1, \ldots, \pi_n \rangle, u)$ is defined*
> > *iff $T_{\mathsf{I}}(\pi_1, u)$ is defined, ..., $T_{\mathsf{I}}(\pi_n, u)$ is defined, and*
> > *if $T_{\mathsf{I}}(\langle \pi_1, \ldots, \pi_n \rangle, u)$ is defined*
> > *then $T_{\mathsf{I}}(\langle \pi_1, \ldots, \pi_n \rangle, u) = \langle T_{\mathsf{I}}(\pi_1, u), \ldots, T_{\mathsf{I}}(\pi_n, u) \rangle$.*

Proposition 10 *For each signature Σ, for each Σ interpretation $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1 \rangle$, for each $o_1 \in \mathsf{U}_1$, for each Σ interpretation $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2 \rangle$, for each $o_2 \in \mathsf{U}_2$,*

> *$\langle o_1, \mathsf{I}_1 \rangle$ and $\langle o_2, \mathsf{I}_2 \rangle$ are congruent in Σ iff $\langle o_1, \mathsf{I}_1 \rangle$ and $\langle o_2, \mathsf{I}_2 \rangle$ are indiscernible in Σ.*

Proof

Firstly, for each signature Σ, for each Σ interpretation $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1 \rangle$, for each $o_1 \in \mathsf{U}_1$, for each Σ interpretation $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2 \rangle$, for each $o_2 \in \mathsf{U}_2$, for each congruence f from $\langle o_1, \mathsf{I}_1 \rangle$ to $\langle o_2, \mathsf{I}_2 \rangle$, *for each* $\tau \in \mathcal{T}^{\Sigma}$*, for each total function ass from* $\mathcal{VAR}$ *to* $\overline{\mathsf{Co}^{o_1}_{\mathsf{I}_1}}$*,*

$T^{ass}_{\mathsf{I}_1}(\tau)(o_1)$ *is defined iff* $T^{f \circ ass}_{\mathsf{I}_2}(\tau)(o_2)$ *is defined, and*
if $T^{ass}_{\mathsf{I}_1}(\tau)(o_1)$ *is defined then* $f(T^{ass}_{\mathsf{I}_1}(\tau)(o_1)) = T^{f \circ ass}_{\mathsf{I}_2}(\tau)(o_2)$*, and*
by induction on the length of τ

for each $\delta \in \mathcal{D}^{\Sigma}$*, for each total function ass from* $\mathcal{VAR}$ *to* $\overline{\mathsf{Co}^{o_1}_{\mathsf{I}_1}}$*,*

$o_1 \in D^{ass}_{\mathsf{I}_1}(\delta)$ *iff* $o_2 \in D^{f \circ ass}_{\mathsf{I}_2}(\delta)$*. by induction on the complexity of* δ
(since, for each $o \in \mathsf{U}_1$*,* $f \circ (ass\frac{o}{x}) = (f \circ ass)\frac{f(o)}{x}$*)*

Thus, for each signature Σ*, for each* Σ *interpretation* $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1 \rangle$*, for each* $o_1 \in \mathsf{U}_1$*, for each* Σ *interpretation* $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2 \rangle$*, for each* $o_2 \in \mathsf{U}_2$*,*

$\langle o_1, \mathsf{I}_1 \rangle$ *and* $\langle o_2, \mathsf{I}_2 \rangle$ *are congruent in* Σ

$\Longrightarrow$ *for some congruence* f*,* f *is a congruence from* $\langle o_1, \mathsf{I}_1 \rangle$ *to* $\langle o_2, \mathsf{I}_2 \rangle$ *in* Σ

$\Longrightarrow$ *for each* $\delta \in \mathcal{D}^{\Sigma}_0$*,*

$o_1 \in D_{\mathsf{I}_1}(\delta)$

$\Longleftrightarrow$ *for each* $ass \in Ass_{\mathsf{I}_1}$*,* $o_1 \in D^{ass}_{\mathsf{I}_1}(\delta)$

$\Longleftrightarrow$ *for some* $ass \in Ass_{\mathsf{I}_1}$*,*

$o_1 \in D^{ass}_{\mathsf{I}_1}(\delta)$ *and, for each* $v \in \mathcal{VAR}$*,* $ass(v) \in \overline{\mathsf{Co}^{o_1}_{\mathsf{I}_1}}$
by corollary 2

$\Longleftrightarrow$ *for some* $ass \in Ass_{\mathsf{I}_2}$*,*

$o_2 \in D^{ass}_{\mathsf{I}_2}(\delta)$ *and, for each* $v \in \mathcal{VAR}$*,* $ass(v) \in \overline{\mathsf{Co}^{o_2}_{\mathsf{I}_2}}$

$\Longleftrightarrow$ *for each* $ass \in Ass_{\mathsf{I}_2}$*,* $o_2 \in D^{ass}_{\mathsf{I}_2}(\delta)$

$\Longleftrightarrow$ $o_2 \in D_{\mathsf{I}_2}(\delta)$

$\Longrightarrow$ $\langle o_1, \mathsf{I}_1 \rangle$ *and* $\langle o_2, \mathsf{I}_2 \rangle$ *are indiscernible in* Σ*.*

Secondly, suppose $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$ *is a signature,* $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1 \rangle$ *is a* Σ *interpretation,* $o_1 \in \mathsf{U}_1$, $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2 \rangle$ *is a* Σ *interpretation,* $o_2 \in \mathsf{U}_2$, *and* $\langle o_1, \mathsf{I}_1 \rangle$ *and* $\langle o_2, \mathsf{I}_2 \rangle$ *are indiscernible in* Σ.

Firstly, for each $\pi \in \mathcal{A}^*$, $T_{\mathsf{I}_1}(\pi, o_1)$ *is defined iff* $T_{\mathsf{I}_2}(\pi, o_2)$ *is defined.*
by induction on the length of π

Secondly, let $f = \left\{ \langle o_1', o_2' \rangle \in \mathsf{Co}^{o_1}_{\mathsf{I}_1} \times \mathsf{Co}^{o_2}_{\mathsf{I}_2} \;\middle|\; \begin{array}{l} \text{for some } \pi \in \mathcal{A}^* \\ \quad T_{\mathsf{I}_1}(\pi, o_1) \text{ is defined,} \\ \quad T_{\mathsf{I}_2}(\pi, o_2) \text{ is defined,} \\ \quad o_1' = T_{\mathsf{I}_1}(\pi, o_1), \text{ and} \\ \quad o_2' = T_{\mathsf{I}_2}(\pi, o_2) \end{array} \right\}$.

f *is a bijection from* $\mathsf{Co}^{o_1}_{\mathsf{I}_1}$ *to* $\mathsf{Co}^{o_2}_{\mathsf{I}_2}$.

Let $\overline{f}$ *be the total function from* $\overline{\mathsf{Co}^{o_1}_{\mathsf{I}_1}}$ *to* $\overline{\mathsf{Co}^{o_2}_{\mathsf{I}_2}}$ *such that,*

for each $u \in \mathsf{Co}^{o_1}_{\mathsf{I}_1}$, $\overline{f}(u) = f(u)$, *and*
for each $\langle u_1, \ldots, u_n \rangle \in (\mathsf{Co}^{o_1}_{\mathsf{I}_1})^*$, $\overline{f}(\langle u_1, \ldots, u_n \rangle) = \langle f(u_1), \ldots, f(u_n) \rangle$.

Clearly, $\overline{f}$ *is a bijection.*

Thirdly, for each $u \in \mathsf{Co}^{o_1}_{\mathsf{I}_1}$, $\mathsf{S}_1(u) = \mathsf{S}_2(f(u))$. *Thus, for each* $u \in \overline{\mathsf{Co}^{o_1}_{\mathsf{I}_1}}$, $\widehat{\mathsf{S}_1}(u) = \widehat{\mathsf{S}_2}(\overline{f}(u))$.

Fourthly, for each $\alpha \in \mathcal{A}$, *for each* $u \in \mathsf{Co}^{o_1}_{\mathsf{I}_1}$,

$\mathsf{A}_1(\alpha)(u)$ *is defined iff* $\mathsf{A}_2(\alpha)(f(u))$ *is defined, and*
if $\mathsf{A}_1(\alpha)(u)$ *is defined then* $f(\mathsf{A}_1(\alpha)(u)) = \mathsf{A}_2(\alpha)(f(u))$.

Thus, for each $\alpha \in \widehat{\mathcal{A}}$, *for each* $u \in \overline{\mathsf{Co}^{o_1}_{\mathsf{I}_1}}$,

$\widehat{\mathsf{A}_1}(\alpha)(u)$ *is defined iff* $\widehat{\mathsf{A}_2}(\alpha)(\overline{f}(u))$ *is defined, and*
if $\widehat{\mathsf{A}_1}(\alpha)(u)$ *is defined then* $\overline{f}(\widehat{\mathsf{A}_1}(\alpha)(u)) = \widehat{\mathsf{A}_2}(\alpha)(\overline{f}(u))$.

Fifthly, for each $\rho \in \mathcal{R}$, *for each* $u_1 \in \overline{\mathsf{Co}^{o_1}_{\mathsf{I}_1}}$, $\ldots$, *for each* $u_n \in \overline{\mathsf{Co}^{o_1}_{\mathsf{I}_1}}$,

$\langle u_1, \ldots, u_n \rangle \in \mathsf{R}_{\mathsf{I}_1}(\rho)$,
$\Longleftrightarrow$ *for some* $\pi_1 \in \overline{\mathcal{A}^*}$, $\ldots$, *for some* $\pi_n \in \overline{\mathcal{A}^*}$,
$u_1 = T_{\mathsf{I}_1}(\pi_1, o_1), \ldots, u_n = T_{\mathsf{I}_1}(\pi_n, o_1)$, *and*
$\langle u_1, \ldots, u_n \rangle \in \mathsf{R}_{\mathsf{I}_1}(\rho)$

$$\begin{aligned}
&\Longleftrightarrow \textit{for some } \pi_1 \in \overline{\mathcal{A}^*},\ \ldots,\ \textit{for some } \pi_n \in \overline{\mathcal{A}^*},\\
&\quad u_1 = T_{\mathsf{I}_1}(\pi_1, o_1), \ldots, u_n = T_{\mathsf{I}_1}(\pi_n, o_1),\ \textit{and}\\
&\quad o_1 \in D_{\mathsf{I}_1}\left(\exists x_1 \ldots \exists x_n \left[\begin{array}{l} \rho(x_1, \ldots, x_n) \land \\ x_1 \approx :\pi_1 \ \land \ldots \land x_n \approx :\pi_n \end{array} \right] \right)^1\\
&\Longleftrightarrow \textit{for some } \pi_1 \in \overline{\mathcal{A}^*},\ \ldots,\ \textit{for some } \pi_n \in \overline{\mathcal{A}^*},\\
&\quad \overline{f}(u_1) = T_{\mathsf{I}_2}(\pi_1, o_2), \ldots, \overline{f}(u_n) = T_{\mathsf{I}_2}(\pi_n, o_2),\ \textit{and}\\
&\quad o_2 \in D_{\mathsf{I}_2}\left(\exists x_1 \ldots \exists x_n \left[\begin{array}{l} \rho(x_1, \ldots, x_n) \land \\ x_1 \approx :\pi_1 \ \land \ldots \land \ x_n \approx :\pi_n \end{array} \right] \right)\\
&\Longleftrightarrow \textit{for some } \pi_1 \in \overline{\mathcal{A}^*},\ \ldots,\ \textit{for some } \pi_n \in \overline{\mathcal{A}^*},\\
&\quad \overline{f}(u_1) = T_{\mathsf{I}_2}(\pi_1, o_2), \ldots, \overline{f}(u_n) = T_{\mathsf{I}_2}(\pi_n, o_2),\ \textit{and}\\
&\quad \langle \overline{f}(u_1), \ldots, \overline{f}(u_n) \rangle \in \mathsf{R}_{\mathsf{I}_2}(\rho)\\
&\Longleftrightarrow \langle \overline{f}(u_1), \ldots, \overline{f}(u_n) \rangle \in \mathsf{R}_{\mathsf{I}_2}(\rho).
\end{aligned}$$

Finally, $\overline{f}(o_1) = o_2$.

Therefore, $\overline{f}$ *is a congruence from* $\langle o_1, \mathsf{I}_1 \rangle$ *to* $\langle o_2, \mathsf{I}_2 \rangle$ *in* Σ. $\langle o_1, \mathsf{I}_1 \rangle$ *and* $\langle o_2, \mathsf{I}_2 \rangle$ *are, thus, congruent in* Σ. ■

Definition 68 *For each signature* Σ*, for each* Σ *interpretation* $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1 \rangle$*, for each* Σ *interpretation* $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2 \rangle$*,*

I_1 ***simulates*** I_2 ***in*** Σ

iff for each $u_2 \in \mathsf{U}_2$*, for some* $u_1 \in \mathsf{U}_1$*,* $\langle u_1, \mathsf{I}_1 \rangle$ *and* $\langle u_2, \mathsf{I}_2 \rangle$ *are congruent in* Σ*.*

A Σ interpretation I_1 simulates Σ interpretation I_2 in Σ just in case every entity in I_2 has a congruent counterpart in I_1.

Proposition 23 *For each signature* Σ*,*

[1] I write $x \approx :\langle \pi_1, \ldots, \pi_n \rangle$ as an abbreviation for
$x\dagger \approx :\pi_1 \ \land \ldots \land \ x \underbrace{\triangleright \ldots \triangleright}_{i \ times} \dagger \approx :\pi_{i+1} \ \land \ldots \land \ x \underbrace{\triangleright \ldots \triangleright}_{n \ times} \sim \mathit{echain}$.

for each Σ *interpretation* I,

> I *simulates* I *in* Σ, *and*

for each Σ *interpretation* I_1, *for each* Σ *interpretation* I_2, *for each* Σ *interpretation* I_3,

> *if* I_1 *simulates* I_2 *in* Σ *and* I_2 *simulates* I_3 *in* Σ *then* I_1 *simulates* I_3 *in* Σ.

PROPOSITION 23 shows that for each signature Σ, the collection of Σ interpretations under simulation almost constitutes a preorder. Note, however, that the collection of Σ interpretations is a proper class.

Proposition 11 *For each signature* Σ, *for each* $\theta \subseteq \mathcal{D}_0^\Sigma$, *for each* Σ *interpretation* I,

for each Σ *interpretation* I',

> *if* I' *is a* $\langle \Sigma, \theta \rangle$ *model then* I *simulates* I' *in* Σ

iff for each $\theta' \subseteq \mathcal{D}_0^\Sigma$, *for each* Σ *interpretation* I',

> *if* I' *is a* $\langle \Sigma, \theta \rangle$ *model and* $\Theta_{\mathsf{I}'}(\theta') \neq \emptyset$ *then* $\Theta_{\mathsf{I}}(\theta') \neq \emptyset$.

Proof
Firstly, for each signature Σ, *for each* $\theta \subseteq \mathcal{D}_0^\Sigma$, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$,

for each Σ *interpretation* $\mathsf{I}' = \langle \mathsf{U}', \mathsf{S}', \mathsf{A}', \mathsf{R}' \rangle$,
if I' *is a* $\langle \Sigma, \theta \rangle$ *model then* I *simulates* I'

$\Longrightarrow$ *for each* $\theta' \subseteq \mathcal{D}_0^\Sigma$, *for each* Σ *interpretation* $\mathsf{I}' = \langle \mathsf{U}', \mathsf{S}', \mathsf{A}', \mathsf{R}' \rangle$,
I' *is a* $\langle \Sigma, \theta \rangle$ *model and* $\Theta_{\mathsf{I}'}(\theta') \neq \emptyset$

$\Longrightarrow$ I *simulates* I' *in* Σ *and, for some* $u' \in \mathsf{U}'$, $u' \in \Theta_{\mathsf{I}'}(\theta')$

$\Longrightarrow$ *for some* $u \in \mathsf{U}$, *for some* $u' \in \mathsf{U}'$,
$\langle u, \mathsf{I} \rangle$ *and* $\langle u', \mathsf{I}' \rangle$ *are congruent in* Σ *and* $u' \in \Theta_{\mathsf{I}'}(\theta')$

$\Longrightarrow$ *for some* $u \in \mathsf{U}$, *for some* $u' \in \mathsf{U}'$,
$\langle u, \mathsf{I} \rangle$ *and* $\langle u', \mathsf{I}' \rangle$ *are indiscernible in* Σ *and* $u' \in \Theta_{\mathsf{I}'}(\theta')$
by proposition 10

$\Longrightarrow$ *for some* $u \in \mathsf{U}$, $u \in \Theta_{\mathsf{I}}(\theta')$

$\Longrightarrow \Theta_{\mathsf{I}}(\theta') \neq \emptyset$.

Secondly, for each signature Σ, *for each* $\theta \subseteq \mathcal{D}_0^\Sigma$, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$,
for each $\theta' \subseteq \mathcal{D}_0^\Sigma$, *for each* Σ *interpretation* I',
if I' *is a* $\langle \Sigma, \theta \rangle$ *model and* $\Theta_{\mathsf{I}'}(\theta') \neq \emptyset$ *then* $\Theta_{\mathsf{I}}(\theta') \neq \emptyset$

$\Longrightarrow$ *for each* Σ *interpretation* $\mathsf{I}' = \langle \mathsf{U}', \mathsf{S}', \mathsf{A}', \mathsf{R}' \rangle$,
I' *is a* $\langle \Sigma, \theta \rangle$ *model*

$\Longrightarrow$ *for each* u',

$u' \in \mathsf{U}'$

$\Longrightarrow u' \in \Theta_{\mathsf{I}'} \left\{ \delta \in \mathcal{D}_0^\Sigma \,\middle|\, u' \in D_{\mathsf{I}'}(\delta) \right\}$

$\Longrightarrow \Theta_{\mathsf{I}'} \left\{ \delta \in \mathcal{D}_0^\Sigma \,\middle|\, u' \in D_{\mathsf{I}'}(\delta) \right\} \neq \emptyset$

$\Longrightarrow \Theta_{\mathsf{I}} \left\{ \delta \in \mathcal{D}_0^\Sigma \,\middle|\, u' \in D_{\mathsf{I}'}(\delta) \right\} \neq \emptyset$

$\Longrightarrow$ *for some* $u \in \mathsf{U}$, $u \in \Theta_{\mathsf{I}} \left\{ \delta \in \mathcal{D}_0^\Sigma \,\middle|\, u' \in D_{\mathsf{I}'}(\delta) \right\}$

$\Longrightarrow$ *for some* $u \in \mathsf{U}$, *for each* $\delta \in \mathcal{D}_0^\Sigma$,

$u \in D_{\mathsf{I}}(\delta)$

$\Longrightarrow u \notin D_{\mathsf{I}}(\neg\delta)$

$\Longrightarrow \neg\delta \notin \left\{ \delta \in \mathcal{D}_0^\Sigma \,\middle|\, u' \in D_{\mathsf{I}'}(\delta) \right\}$

$\Longrightarrow u' \notin D_{\mathsf{I}'}(\neg\delta)$

$\Longrightarrow u' \in D_{\mathsf{I}'}(\delta)$, *and*

$u' \in D_{\mathsf{I}'}(\delta)$

$\Longrightarrow \delta \in \left\{ \delta \in \mathcal{D}_0^\Sigma \,\middle|\, u' \in D_{\mathsf{I}'}(\delta) \right\}$

$$\Longrightarrow u \in D_{\mathsf{I}}(\delta)$$

$$\Longrightarrow \textit{for some } u \in \mathsf{U},\ \langle u, \mathsf{I}\rangle \textit{ and } \langle u', \mathsf{I}'\rangle \textit{ are indiscernible in } \Sigma$$

$$\Longrightarrow \textit{for some } u \in \mathsf{U},\ \langle u, \mathsf{I}\rangle \textit{ and } \langle u', \mathsf{I}'\rangle \textit{ are congruent in } \Sigma$$

by proposition 10

$$\Longrightarrow \mathsf{I} \textit{ simulates } \mathsf{I}' \textit{ in } \Sigma.$$ ■

Theorem 3 follows directly from Proposition 11 and the definition of the exhaustive models of a grammar (Definition 64).

Theorem 3 *For each signature Σ, for each $\theta \subseteq \mathcal{D}_0^\Sigma$, for each Σ interpretation I,*

I is an exhaustive $\langle\Sigma, \theta\rangle$ model iff

I is a $\langle\Sigma, \theta\rangle$ model, and
for each Σ interpretation I',
if I' is a $\langle\Sigma, \theta\rangle$ model then I simulates I' in Σ.

A.2 Existence Proof for Exhaustive Models

In this section I show that, for each grammar, there exists an exhaustive model. For each grammar $\langle\Sigma, \theta\rangle$, I construct a Σ interpretation which I call the canonical Σ interpretation of θ. I then show that the canonical Σ interpretation is an exhaustive model of the grammar $\langle\Sigma, \theta\rangle$.

In the first part of this section, from Definition 69 to Definition 77, I prepare the construction of the canonical exhaustive models by defining the extended abstract feature structures for RSRL grammars, which I call morphs, and I say what it means for a theory under a signature to admit a set of morphs. Morphs and morph admission then play a central role in the second part of this section, where a canonical interpretation of each grammar is defined and and then shown to be an exhaustive model of the grammar (Definition 78–Theorem 2).

Suppose $\Sigma = \langle\mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR}\rangle$ is a signature. I call each member of $\mathcal{A}^*$ a Σ path, and write ε for the empty path, the unique path of length zero.

Definition 69 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$,

μ *is a* ***morph in*** Σ
iff μ *is a quadruple* $\langle \beta, \varrho, \lambda, \xi \rangle$,
$\beta \subseteq \mathcal{A}^*$,
$\varepsilon \in \beta$,
for each $\pi \in \mathcal{A}^*$, *for each* $\alpha \in \mathcal{A}$,
if $\pi\alpha \in \beta$ *then* $\pi \in \beta$,
ϱ *is an equivalence relation over* β,
for each $\pi_1 \in \mathcal{A}^*$, *for each* $\pi_2 \in \mathcal{A}^*$, *for each* $\alpha \in \mathcal{A}$,
if $\pi_1\alpha \in \beta$ *and* $\langle \pi_1, \pi_2 \rangle \in \varrho$ *then* $\langle \pi_1\alpha, \pi_2\alpha \rangle \in \varrho$,
λ *is a total function from* β *to* $\mathcal{S}$,
for each $\pi_1 \in \mathcal{A}^*$, *for each* $\pi_2 \in \mathcal{A}^*$,
if $\langle \pi_1, \pi_2 \rangle \in \varrho$ *then* $\lambda(\pi_1) = \lambda(\pi_2)$,
for each $\pi \in \mathcal{A}^*$, *for each* $\alpha \in \mathcal{A}$,
if $\pi\alpha \in \beta$ *then* $\mathcal{F}\langle \lambda(\pi), \alpha \rangle$ *is defined and* $\lambda(\pi\alpha) \sqsubseteq \mathcal{F}\langle \lambda(\pi), \alpha \rangle$,
for each $\pi \in \mathcal{A}^*$, *for each* $\alpha \in \mathcal{A}$,
if $\pi \in \beta$ *and* $\mathcal{F}\langle \lambda(\pi), \alpha \rangle$ *is defined then* $\pi\alpha \in \beta$,
$\xi \subseteq \mathcal{R} \times \overline{\beta}^*$,
for each $\rho \in \mathcal{R}$, *for each* $\pi_1 \in \overline{\beta}$,..., *for each* $\pi_n \in \overline{\beta}$,
if $\langle \rho, \pi_1, \ldots, \pi_n \rangle \in \xi$ *then* $n = \mathcal{AR}(\rho)$, *and*
for each $\rho \in \mathcal{R}$, *for each* $\pi_1 \in \overline{\beta}$, ..., *for each* $\pi_n \in \overline{\beta}$, *for each* $\pi'_1 \in \overline{\beta}$, ..., *for each* $\pi'_n \in \overline{\beta}$,
if $\langle \rho, \pi_1, \ldots, \pi_n \rangle \in \xi$, *and*
for each $i \in \{1, \ldots, n\}$,
$\pi_i \in \beta$ *and* $\langle \pi_i, \pi'_i \rangle \in \varrho$, *or*
for some $m \in \mathbb{N}$,

$$\begin{aligned}
&\qquad \pi_i \in \beta^*,\\
&\qquad \pi_i = \langle \pi_{i_1}, \ldots, \pi_{i_m} \rangle,\\
&\qquad \pi'_i = \langle \pi'_{i_1}, \ldots, \pi'_{i_m} \rangle, \textit{ and}\\
&\qquad \langle \pi_{i_1}, \pi'_{i_1} \rangle \in \varrho, \ldots, \langle \pi_{i_m}, \pi'_{i_m} \rangle \in \varrho,\\
&\textit{then } \langle \rho, \pi'_1, \ldots, \pi'_n \rangle \in \xi.
\end{aligned}$$

Suppose Σ is a signature and $\mu = \langle \beta, \varrho, \lambda, \xi \rangle$ is a Σ morph. I call β the *basis set in* μ, ϱ the *re-entrancy relation in* μ, λ the *label function in* μ, and ξ the *relation extension in* μ. I write $\mathcal{M}_\Sigma$ for the set of Σ morphs. The Σ morphs are a straightforward extension of the abstract feature structures of Moshier 1988.

Definition 70 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$*, for each* $\pi \in \mathcal{A}^*$*, for each* $\langle \pi_1, \ldots, \pi_n \rangle \in \mathcal{A}^{**}$*,*

$$\pi\langle \pi_1, \ldots, \pi_n \rangle \textit{ is an abbreviatory notation for } \langle \pi\pi_1, \ldots, \pi\pi_n \rangle.$$

Definition 71 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$*, for each* $\mu = \langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma$*, for each* $\pi \in \mathcal{A}^*$*,*

$$\begin{aligned}
&\beta/\pi = \{\pi' \in \mathcal{A}^* \mid \pi\pi' \in \beta\},\\
&\varrho/\pi = \{\langle \pi_1, \pi_2 \rangle \in \mathcal{A}^* \times \mathcal{A}^* \mid \langle \pi\pi_1, \pi\pi_2 \rangle \in \varrho\},\\
&\lambda/\pi = \{\langle \pi', \sigma \rangle \in \mathcal{A}^* \times \mathcal{S} \mid \langle \pi\pi', \sigma \rangle \in \lambda\},\\
&\xi/\pi = \{\langle \rho, \pi_1, \ldots, \pi_n \rangle \in \mathcal{R} \times \overline{(\beta/\pi)}^* \mid \langle \rho, \pi\pi_1, \ldots, \pi\pi_n \rangle \in \xi\}, \textit{ and}\\
&\mu/\pi = \langle \beta/\pi, \varrho/\pi, \lambda/\pi, \xi/\pi \rangle.
\end{aligned}$$

If Σ is a signature, μ is a Σ morph and π is a Σ path then I call μ/π the π *reduct of* μ.

Proposition 12 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$*, for each* $\mu = \langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma$*, for each* $\pi \in \mathcal{A}^*$*,*

$$\textit{if } \pi \in \beta \textit{ then } \mu/\pi \in \mathcal{M}_\Sigma.$$

Definition 72 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$*,*

I_Σ *is the set of total functions from* $\mathcal{VAR}$ *to* $\overline{\mathcal{A}^*}$.

Let Σ be a signature. Then I call each member of I_Σ a Σ *insertion*.

Definition 73 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$,

T_Σ *is the smallest partial function from* $\overline{\mathcal{A}^*} \times \widehat{\mathcal{A}}$ *to* $\overline{\mathcal{A}^*}$ *such that,*

for each $\pi \in \mathcal{A}^*$, *for each* $\alpha \in \mathcal{A}$,

$$\mathrm{T}_\Sigma(\pi, \alpha) = \pi\alpha,$$

for each $\langle \pi_0, \ldots, \pi_n \rangle \in \mathcal{A}^{**}$,

$$\mathrm{T}_\Sigma(\langle \pi_0, \ldots, \pi_n \rangle, \dagger) = \pi_0,$$
$$\mathrm{T}_\Sigma(\langle \pi_0, \ldots, \pi_n \rangle, \triangleright) = \langle \pi_1, \ldots, \pi_n \rangle.$$

Definition 74 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each* $\iota \in \mathrm{I}_\Sigma$,

Π^ι_Σ *is the smallest partial function from* $\mathcal{T}^\Sigma$ *to* $\overline{\mathcal{A}^*}$ *such that*

$\Pi^\iota_\Sigma(:) = \varepsilon$,
for each $v \in \mathcal{VAR}$, $\Pi^\iota_\Sigma(v) = \iota(v)$, *and*
for each $\tau \in \mathcal{T}^\Sigma$, *for each* $\alpha \in \widehat{\mathcal{A}}$, $\Pi^\iota_\Sigma(\tau\alpha) = \mathrm{T}_\Sigma(\Pi^\iota_\Sigma(\tau), \alpha)$.

Suppose Σ is a signature, and ι is a Σ insertion. Then I call each Π^ι_Σ the *path insertion function under* ι *in* Σ.

In order to say when a morph satisfies a formula that contains descriptions of chains, the re-entrancy relation and the label function in morphs must be extended to tuples of paths from the basis set in morphs.

Definition 75 *For each signature* Σ, *for each* $\langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma$,

$\hat{\varrho}$ *is the smallest subset of* $\overline{\beta} \times \overline{\beta}$ *such that*

$\varrho \subseteq \hat{\varrho}$, *and*
for each $\pi_1 \in \beta, \ldots, \pi_n \in \beta$, *for each* $\pi'_1 \in \beta, \ldots, \pi'_n \in \beta$,

if $\langle \pi_1, \pi'_1 \rangle \in \varrho$, $\ldots$, *and* $\langle \pi_n, \pi'_n \rangle \in \varrho$
then $\langle \langle \pi_1, \ldots, \pi_n \rangle, \langle \pi'_1, \ldots, \pi'_n \rangle \rangle \in \hat{\varrho}$,

$\hat{\lambda}$ *is the total function from* $\overline{\beta}$ *to* $\widehat{\mathcal{S}}$ *such that*

for each $\pi \in \beta$, $\hat{\lambda}(\pi) = \lambda(\pi)$,
for each $\pi_1 \in \beta$, ..., *for each* $\pi_n \in \beta$,

$$\hat{\lambda}(\langle \pi_1, \ldots, \pi_n \rangle) = \begin{cases} \textit{echain} & \textit{if } n = 0, \\ \textit{nechain} & \textit{if } n > 0. \end{cases}$$

Everything is now in place for the definition of the morph satisfaction functions:

Definition 76 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each* $\iota \in \mathrm{I}_\Sigma$, Δ^ι_Σ *is the total function from* $\mathcal{D}^\Sigma$ *to* $Pow(\mathcal{M}_\Sigma)$ *such that*

for each $\tau \in \mathcal{T}^\Sigma$, *for each* $\sigma \in \widehat{\mathcal{G}}$,

$$\Delta^\iota_\Sigma(\tau \sim \sigma) = \left\{ \langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma \;\middle|\; \begin{array}{l} \Pi^\iota_\Sigma(\tau) \textit{ is defined, and,} \\ \textit{for some } \sigma' \in \widehat{\mathcal{S}}, \\ \quad \langle \Pi^\iota_\Sigma(\tau), \sigma' \rangle \in \hat{\lambda} \textit{ and } \sigma' \mathrel{\widehat{\sqsubseteq}} \sigma \end{array} \right\},$$

for each $\tau_1 \in \mathcal{T}^\Sigma$, *for each* $\tau_2 \in \mathcal{T}^\Sigma$,

$$\Delta^\iota_\Sigma(\tau_1 \approx \tau_2) = \left\{ \langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma \;\middle|\; \begin{array}{l} \Pi^\iota_\Sigma(\tau_1) \textit{ is defined,} \\ \Pi^\iota_\Sigma(\tau_2) \textit{ is defined, and} \\ \langle \Pi^\iota_\Sigma(\tau_1), \Pi^\iota_\Sigma(\tau_2) \rangle \in \hat{\varrho} \end{array} \right\},$$

for each $\rho \in \mathcal{R}$, *for each* $v_1 \in \mathcal{VAR}$, ..., *for each* $v_n \in \mathcal{VAR}$,

$$\Delta^\iota_\Sigma(\rho(v_1, \ldots, v_n)) = \left\{ \langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma \;\middle|\; \langle \rho, \iota(v_1), \ldots, \iota(v_n) \rangle \in \xi \right\},$$

for each $v \in \mathcal{VAR}$, *for each* $\delta \in \mathcal{D}^\Sigma$,

$$\Delta^\iota_\Sigma(\exists v\, \delta) = \left\{ \langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma \;\middle|\; \begin{array}{l} \textit{for some } \pi \in \overline{\beta}, \\ \quad \langle \beta, \varrho, \lambda, \xi \rangle \in \Delta^{\iota[\frac{\pi}{v}]}_\Sigma(\delta) \end{array} \right\},$$

for each $v \in \mathcal{VAR}$, *for each* $\delta \in \mathcal{D}^\Sigma$,

$$\Delta^\iota_\Sigma(\forall v\, \delta) = \left\{ \langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma \;\middle|\; \begin{array}{l} \textit{for each } \pi \in \overline{\beta}, \\ \quad \langle \beta, \varrho, \lambda, \xi \rangle \in \Delta^{\iota[\frac{\pi}{v}]}_\Sigma(\delta) \end{array} \right\},$$

for each $\delta \in \mathcal{D}^{\Sigma}$,

$$\Delta^{\iota}_{\Sigma}(\neg\delta) = \mathcal{M}_{\Sigma} \backslash \Delta^{\iota}_{\Sigma}(\delta),$$

for each $\delta_1 \in \mathcal{D}^{\Sigma}$, *for each* $\delta_2 \in \mathcal{D}^{\Sigma}$,

$$\Delta^{\iota}_{\Sigma}([\delta_1 \wedge \delta_2]) = \Delta^{\iota}_{\Sigma}(\delta_1) \cap \Delta^{\iota}_{\Sigma}(\delta_2),$$

for each $\delta_1 \in \mathcal{D}^{\Sigma}$, *for each* $\delta_2 \in \mathcal{D}^{\Sigma}$,

$$\Delta^{\iota}_{\Sigma}([\delta_1 \vee \delta_2]) = \Delta^{\iota}_{\Sigma}(\delta_1) \cup \Delta^{\iota}_{\Sigma}(\delta_2),$$

for each $\delta_1 \in \mathcal{D}^{\Sigma}$, *for each* $\delta_2 \in \mathcal{D}^{\Sigma}$,

$$\Delta^{\iota}_{\Sigma}([\delta_1 \rightarrow \delta_2]) = (\mathcal{M}_{\Sigma} \backslash \Delta^{\iota}_{\Sigma}(\delta_1)) \cup \Delta^{\iota}_{\Sigma}(\delta_2), \textit{ and}$$

for each $\delta_1 \in \mathcal{D}^{\Sigma}$, *for each* $\delta_2 \in \mathcal{D}^{\Sigma}$,

$$\begin{aligned}\Delta^{\iota}_{\Sigma}([\delta_1 \leftrightarrow \delta_2]) = {} & ((\mathcal{M}_{\Sigma} \backslash \Delta^{\iota}_{\Sigma}(\delta_1)) \cap (\mathcal{M}_{\Sigma} \backslash \Delta^{\iota}_{\Sigma}(\delta_2))) \cup \\ & (\Delta^{\iota}_{\Sigma}(\delta_1) \cap \Delta^{\iota}_{\Sigma}(\delta_2)).\end{aligned}$$

Let Σ be a signature, μ a Σ morph, ι a Σ insertion and δ a Σ formula. I call Δ^{ι}_{Σ} the *morph satisfaction function under* ι *in* Σ, and say μ *satisfies* δ *under* ι *in* Σ if and only if $\mu \in \Delta^{\iota}_{\Sigma}(\delta)$.

According to Proposition 13, the same sets of Σ morphs satisfy a Σ formula δ under morph satisfaction functions in Σ under two different Σ insertions if the two Σ insertions agree on the variables that occur free in δ:

Proposition 13 *For each signature* Σ, *for each* $\iota_1 \in \mathrm{I}_{\Sigma}$, *for each* $\iota_2 \in \mathrm{I}_{\Sigma}$,

for each $\tau \in \mathcal{T}^{\Sigma}$,

if for each $v \in FV(\tau)$, $\iota_1(v) = \iota_2(v)$

then $\Pi^{\iota_1}_{\Sigma}(\tau)$ *is defined iff* $\Pi^{\iota_2}_{\Sigma}(\tau)$ *is defined, and*

if $\Pi^{\iota_1}_{\Sigma}(\tau)$ *is defined then* $\Pi^{\iota_1}_{\Sigma}(\tau) = \Pi^{\iota_2}_{\Sigma}(\tau)$, *and*

for each $\delta \in \mathcal{D}^{\Sigma}$,

if for each $v \in FV(\delta)$, $\iota_1(v) = \iota_2(v)$
then $\Delta^{\iota_1}_{\Sigma}(\delta) = \Delta^{\iota_2}_{\Sigma}(\delta)$.

Proof
By induction on the length of τ *and the complexity of* δ*, respectively.* ■

Definition 77 *For each signature* Σ,

M_Σ *is the total function from* $Pow(\mathcal{D}^{\Sigma}_{0})$ *to* $Pow(\mathcal{M}_\Sigma)$ *such that for each* $\theta \subseteq \mathcal{D}^{\Sigma}_{0}$,

$$M_\Sigma(\theta) = \left\{ \langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma \;\middle|\; \begin{array}{l} \textit{for each } \pi \in \beta, \\ \quad \textit{for each } \iota \in \mathrm{I}_\Sigma, \textit{ and} \\ \quad \textit{for each } \delta \in \theta, \\ \quad \langle \beta, \varrho, \lambda, \xi \rangle / \pi \in \Delta^{\iota}_{\Sigma}(\delta) \end{array} \right\}.$$

Let Σ be a signature. I call M_Σ the *morph admission function in* Σ.

All relevant notions surrounding Σ morphs have now been defined. In the next three definitions I use them to define a canonical interpretation for each grammar. Then I will show that the canonical interpretations are exhaustive models of their respective grammars.

Definition 78 *For each signature* Σ,

o is a ***canonical entity in*** Σ

iff o is a quintuple $\langle \beta, \varrho, \lambda, \xi, \eta \rangle$,

$\langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma$, *and*

$\eta \in Quo(\langle \beta, \varrho \rangle)$.[2]

Suppose that Σ is a signature. I write $\mathcal{C}_\Sigma$ for the set of canonical entities in Σ. Suppose further that $\langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma$ and $\pi \in \beta$. I write $|\pi|_\varrho$ for the equivalence class of π in ϱ. Thus, I write $\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle$ for the canonical entity $\langle \beta, \varrho, \lambda, \xi, \eta \rangle$, where η is the equivalence class of π in ϱ.

Definition 79 *For each signature* Σ, *for each* $\theta \in \mathcal{D}^{\Sigma}_{0}$, *for each* $\langle \langle \beta, \varrho, \lambda, \xi, |\pi_1|_\varrho \rangle, \ldots, \langle \beta, \varrho, \lambda, \xi, |\pi_n|_\varrho \rangle \rangle \in (\mathcal{C}_\Sigma)^*$,

[2] $Quo(\langle \beta, \varrho \rangle)$ is the quotient of ϱ, i.e., the set of equivalence classes of ϱ.

$\langle \beta, \varrho, \lambda, \xi, |\langle \pi_1, \ldots, \pi_n \rangle|_\varrho \rangle$ *is an abbreviatory notation for* $\langle \langle \beta, \varrho, \lambda, \xi, |\pi_1|_\varrho \rangle, \ldots, \langle \beta, \varrho, \lambda, \xi, |\pi_n|_\varrho \rangle \rangle$.

Definition 80 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each* $\theta \subseteq \mathcal{D}_0^\Sigma$,

$$\mathbf{U}_\Sigma^\theta = \left\{ \langle \beta, \varrho, \lambda, \xi, \eta \rangle \in \mathcal{C}_\Sigma \;\middle|\; \langle \beta, \varrho, \lambda, \xi \rangle \in M_\Sigma(\theta) \right\},$$

$$\mathbf{S}_\Sigma^\theta = \left\{ \langle u, \sigma \rangle \in \mathbf{U}_\Sigma^\theta \times \mathcal{S} \;\middle|\; \begin{array}{l} \textit{for some } \langle \beta, \varrho, \lambda, \xi \rangle \in M_\Sigma(\theta), \\ \textit{for some } \pi \in \beta, \\ \quad u = \langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle, \textit{ and} \\ \quad \sigma = \lambda(\pi) \end{array} \right\},$$

$\mathbf{A}_\Sigma^\theta$ *is the total function from* $\mathcal{A}$ *to* $Pow(\mathbf{U}_\Sigma^\theta \times \mathbf{U}_\Sigma^\theta)$ *such that for each* $\alpha \in \mathcal{A}$,

$$\mathbf{A}_\Sigma^\theta(\alpha) = \left\{ \langle u, u' \rangle \in \mathbf{U}_\Sigma^\theta \times \mathbf{U}_\Sigma^\theta \;\middle|\; \begin{array}{l} \textit{for some } \langle \beta, \varrho, \lambda, \xi \rangle \in M_\Sigma(\theta), \\ \textit{for some } \pi \in \beta, \\ \quad u = \langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle \textit{ and} \\ \quad u' = \langle \beta, \varrho, \lambda, \xi, |\pi\alpha|_\varrho \rangle \end{array} \right\},$$

$\mathbf{R}_\Sigma^\theta$ *is the total function from* $\mathcal{R}$ *to the power set of* $\bigcup_{n \in \mathbb{N}} \overline{\mathbf{U}_\Sigma^\theta}^n$ *such that for each* $\rho \in \mathcal{R}$,

$$\mathbf{R}_\Sigma^\theta(\rho) = \left\{ \langle u_1, \ldots, u_n \rangle \in \bigcup_{n \in \mathbb{N}} \overline{\mathbf{U}_\Sigma^\theta}^n \;\middle|\; \begin{array}{l} \textit{for some } \langle \beta, \varrho, \lambda, \xi \rangle \in M_\Sigma(\theta), \\ \textit{for some } \langle \rho, \pi_1, \ldots, \pi_n \rangle \in \xi, \\ \quad u_1 = \langle \beta, \varrho, \lambda, \xi, |\pi_1|_\varrho \rangle, \ \ldots, \\ \quad u_n = \langle \beta, \varrho, \lambda, \xi, |\pi_n|_\varrho \rangle \end{array} \right\},$$

$\mathbf{I}_\Sigma^\theta = \langle \mathbf{U}_\Sigma^\theta, \mathbf{S}_\Sigma^\theta, \mathbf{A}_\Sigma^\theta, \mathbf{R}_\Sigma^\theta \rangle$.

Proposition 14 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each* $\theta \subseteq \mathcal{D}_0^\Sigma$,

$\mathbf{U}_\Sigma^\theta$ *is a set,*

$\mathbf{S}_\Sigma^\theta$ *is a total function from* $\mathbf{U}_\Sigma^\theta$ *to* $\mathcal{S}$,

$\mathbf{A}^{\theta}_{\Sigma}$ *is a total function from* $\mathcal{A}$ *to the set of partial functions from* $\mathbf{U}^{\theta}_{\Sigma}$ *to* $\mathbf{U}^{\theta}_{\Sigma}$,

$\mathbf{R}^{\theta}_{\Sigma}$ *is a total function from* $\mathcal{R}$ *to the power set of* $\bigcup\limits_{n \in \mathbb{N}} \overline{\mathbf{U}^{\theta}_{\Sigma}}^{n}$, *and*

$\mathbf{I}^{\theta}_{\Sigma}$ *is a* Σ *interpretation.*

Let Σ be a signature and $\theta \subseteq \mathcal{D}^{\Sigma}_{0}$. I call $\mathbf{I}^{\theta}_{\Sigma}$ the *canonical* Σ *interpretation of* θ.

The following proposition is an important result about the values of path insertions under a Σ insertion ι in Σ relative to canonical entities under certain restrictions on ι. It will be needed in many of the inductive proofs below to treat the case when Σ insertions insert tuples of Σ paths. It says that if we have a canonical entity in Σ, o, with a distinguished node that is in the equivalence class of the Σ path π, and we restrict ι to insert only Σ paths and tuples of Σ paths that would be in the basis set of the morph in o if they were appended to π, then that means that if the path insertion under the restricted Σ insertion ι in Σ is defined on a Σ term τ and yields a tuple of paths as value on τ, then we obtain a tuple of paths in the basis set of the morph in o from that tuple of paths by prefixing every path in it with π.

Proposition 24 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each* $\langle \beta, \varrho, \lambda, \xi, |\pi|_{\varrho} \rangle \in \mathcal{C}_{\Sigma}$, *for each total function* ι *from* $\mathcal{VAR}$ *to* $\overline{\beta/\pi}$, *for each* $\tau \in \mathcal{T}^{\Sigma}$,

if $\Pi^{\iota}_{\Sigma}(\tau)$ *is defined and* $\Pi^{\iota}_{\Sigma}(\tau) \in \mathcal{A}^{**}$
then $\pi\Pi^{\iota}_{\Sigma}(\tau) \in \beta^{*}$.

Proof
By arithmetic induction on the length of τ. ■

PROPOSITION 24 will be needed presently in the proof of PROPOSITION 15.

Proposition 15 *For each signature* Σ, *for each* $\theta \subseteq \mathcal{D}^{\Sigma}_{0}$,

$\mathbf{I}^{\theta}_{\Sigma}$ *is a* $\langle \Sigma, \theta \rangle$ *model.*

In order to show PROPOSITION 15, I have to show that for each signature Σ, for each $\theta \subseteq \mathcal{D}^{\Sigma}_{0}$, each element of $\mathbf{U}^{\theta}_{\Sigma}$ is in the theory denotation with respect to $\mathbf{I}^{\theta}_{\Sigma}$, i.e., every description in θ describes every entity in $\mathbf{U}^{\theta}_{\Sigma}$. The proof exploits the internal structure of the canonical entities as Σ morphs with a distinguished node, where the morphs are admitted by θ, which means that every π reduct of them satisfies each description in θ. The proposition is then shown by using an equivalence between morph satisfaction functions and theory denotation functions. The interesting step in the proof is to show that if the π reduct of a morph $\mu = \langle \beta, \varrho, \lambda, \xi \rangle$ that occurs in a canonical entity in the universe of $\mathbf{I}^{\theta}_{\Sigma}$ with the distinguished node $|\pi|_{\varrho}$ satisfies every description in the theory θ under some restricted Σ insertion ι then the canonical entity comprising μ and the distinguished node $|\pi|_{\varrho}$ is in the set value of the description interpretation function with respect to $\mathbf{I}^{\theta}_{\Sigma}$ under some particular variable assignment in $\mathbf{I}^{\theta}_{\Sigma}$ for every description in θ. Of course, neither the choice of ι nor the choice of the variable assignment are relevant for Σ descriptions without free occurrences of variables. However, the choice of the Σ insertion and the variable assignment function in the proof are dictated by the fact that I first have to show for arbitrary Σ formulae under which variable assignments and insertions a canonical entity with morph μ and distinguished node $|\pi|_{\varrho}$ is in the denotation of a formula if and only if the π reduct of μ satisfies the formula, since this is the crucial link between morph satisfaction and formula denotation that can be exploited. The first half of the proof is thus concerned with establishing that link, starting with the denotation of terms on canonical entities.

Proof

Throughout this proof, suppose that, for each signature Σ, for each $\theta \subseteq \mathcal{D}^{\Sigma}_{0}$, for each $o = \langle \beta, \varrho, \lambda, \xi, |\pi|_{\varrho} \rangle \in \mathbf{U}^{\theta}_{\Sigma}$, for each total function ι from $\mathcal{VAR}$ to $\overline{\beta/\pi}$,

> *ass^{ι}_{o} is the total function from $\mathcal{VAR}$ to $\overline{\mathbf{U}^{\theta}_{\Sigma}}$ such that*
>
> > *for each $v \in \mathcal{VAR}$, $ass^{\iota}_{o}(v) = \langle \beta, \varrho, \lambda, \xi, |\pi\iota(v)|_{\varrho} \rangle$.*

Firstly, for each signature Σ, for each $\theta \subseteq \mathcal{D}^{\Sigma}_{0}$, for each $o = \langle \beta, \varrho, \lambda, \xi, |\pi|_{\varrho} \rangle \in \mathbf{U}^{\theta}_{\Sigma}$, for each $\tau \in \mathcal{T}^{\Sigma}$, for each total function ι from $\mathcal{VAR}$ to $\overline{\beta/\pi}$,

> *$T^{ass^{\iota}_{o}}_{\mathbf{I}^{\theta}_{\Sigma}}(\tau)(\langle \beta, \varrho, \lambda, \xi, |\pi|_{\varrho} \rangle)$ is defined iff $\Pi^{\iota}_{\Sigma}(\tau)$ is defined and $\pi\Pi^{\iota}_{\Sigma}(\tau) \in \overline{\beta}$, and*
> *if $T^{ass^{\iota}_{o}}_{\mathbf{I}^{\theta}_{\Sigma}}(\tau)(\langle \beta, \varrho, \lambda, \xi, |\pi|_{\varrho} \rangle)$ is defined*
> *then $T^{ass^{\iota}_{o}}_{\mathbf{I}^{\theta}_{\Sigma}}(\tau)(\langle \beta, \varrho, \lambda, \xi, |\pi|_{\varrho} \rangle) = \langle \beta, \varrho, \lambda, \xi, |\pi\Pi^{\iota}_{\Sigma}(\tau)|_{\varrho} \rangle$.*
> *by proposition 24 and arithmetic induction on the length of τ*

Secondly, for each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$*, for each* $\theta \subseteq \mathcal{D}_0^\Sigma$*, for each* $o = \langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle \in \mathbf{U}_\Sigma^\theta$*, for each total function* ι *from* $\mathcal{VAR}$ *to* $\overline{\beta/\pi}$*,*

for each $\tau \in \mathcal{T}^\Sigma$*, for each* $\sigma \in \mathcal{G}$*,*

$$\begin{aligned}
&\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle \in D_{\mathbf{I}_\Sigma^\theta}^{ass_o^\iota}(\tau \sim \sigma) \\
\iff\ & T_{\mathbf{I}_\Sigma^\theta}^{ass_o^\iota}(\tau)(\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle) \textit{ is defined, and} \\
& \widehat{\mathbf{S}_\Sigma^\theta}(T_{\mathbf{I}_\Sigma^\theta}^{ass_o^\iota}(\tau)(\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle)) \mathrel{\widehat{\sqsubseteq}} \sigma \\
\iff\ & \Pi_\Sigma^\iota(\tau) \textit{ is defined, } \pi\Pi_\Sigma^\iota(\tau) \in \overline{\beta}\textit{, and} \\
& \widehat{\mathbf{S}_\Sigma^\theta}(\langle \beta, \varrho, \lambda, \xi, |\pi\Pi_\Sigma^\iota(\tau)|_\varrho \rangle) \mathrel{\widehat{\sqsubseteq}} \sigma \\
\iff\ & \Pi_\Sigma^\iota(\tau) \textit{ is defined, and } \widehat{\lambda/\pi}(\Pi_\Sigma^\iota(\tau)) \mathrel{\widehat{\sqsubseteq}} \sigma \\
\iff\ & \langle \beta, \varrho, \lambda, \xi \rangle/\pi \in \Delta_\Sigma^\iota(\tau \sim \sigma),
\end{aligned}$$

for each $\tau_1 \in \mathcal{T}^\Sigma$*, for each* $\tau_2 \in \mathcal{T}^\Sigma$*,*

$$\begin{aligned}
&\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle \in D_{\mathbf{I}_\Sigma^\theta}^{ass_o^\iota}(\tau_1 \approx \tau_2) \\
\iff\ & T_{\mathbf{I}_\Sigma^\theta}^{ass_o^\iota}(\tau_1)(\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle) \textit{ is defined,} \\
& T_{\mathbf{I}_\Sigma^\theta}^{ass_o^\iota}(\tau_2)(\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle) \textit{ is defined, and} \\
& T_{\mathbf{I}_\Sigma^\theta}^{ass_o^\iota}(\tau_1)(\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle) = T_{\mathbf{I}_\Sigma^\theta}^{ass_o^\iota}(\tau_2)(\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle) \\
\iff\ & \Pi_\Sigma^\iota(\tau_1) \textit{ is defined, } \pi\Pi_\Sigma^\iota(\tau_1) \in \overline{\beta}\textit{, } \Pi_\Sigma^\iota(\tau_2) \textit{ is defined,} \\
& \pi\Pi_\Sigma^\iota(\tau_2) \in \overline{\beta}\textit{, and} \\
& \langle \beta, \varrho, \lambda, \xi, |\pi\Pi_\Sigma^\iota(\tau_1)|_\varrho \rangle = \langle \beta, \varrho, \lambda, \xi, |\pi\Pi_\Sigma^\iota(\tau_2)|_\varrho \rangle \\
\iff\ & \Pi_\Sigma^\iota(\tau_1) \textit{ is defined, } \Pi_\Sigma^\iota(\tau_2) \textit{ is defined, and} \\
& \langle \Pi_\Sigma^\iota(\tau_1), \Pi_\Sigma^\iota(\tau_2) \rangle \in \widehat{\varrho/\pi} \\
\iff\ & \langle \beta, \varrho, \lambda, \xi \rangle/\pi \in \Delta_\Sigma^\iota(\tau_1 \approx \tau_2),
\end{aligned}$$

for each $\rho \in \mathcal{R}$*, for each* $x_1 \in \mathcal{VAR}$*, ..., for each* $x_{\mathcal{AR}(\rho)} \in \mathcal{VAR}$*,*

$$\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle \in D^{ass^\iota_o}_{\mathbf{I}^\theta_\Sigma}(\rho(x_1, \ldots, x_{\mathcal{AR}(\rho)}))$$
$$\iff \langle ass^\iota_o(x_1), \ldots, ass^\iota_o(x_{\mathcal{AR}(\rho)}) \rangle \in \mathbf{R}^\theta_\Sigma(\rho)$$
$$\iff \langle \langle \beta, \varrho, \lambda, \xi, |\pi\iota(x_1)|_\varrho \rangle, \ldots, \langle \beta, \varrho, \lambda, \xi, |\pi\iota(x_{\mathcal{AR}(\rho)})|_\varrho \rangle \rangle \in \mathbf{R}^\theta_\Sigma(\rho)$$
$$\iff \langle \rho, \pi\iota(x_1), \ldots, \pi\iota(x_{\mathcal{AR}(\rho)}) \rangle \in \xi$$
$$\iff \langle \rho, \iota(x_1), \ldots, \iota(x_{\mathcal{AR}(\rho)}) \rangle \in \xi/\pi$$
$$\iff \langle \beta, \varrho, \lambda, \xi \rangle/\pi \in \Delta^\iota_\Sigma(\rho(x_1, \ldots, x_{\mathcal{AR}(\rho)})).$$

Thus, for each signature Σ, for each $\theta \subseteq \mathcal{D}^\Sigma_0$, for each $o = \langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle \in \mathbf{U}^\theta_\Sigma$, for each $\delta \in \mathcal{D}^\Sigma$, for each total function ι from $\mathcal{VAR}$ to $\overline{\beta/\pi}$,

$$\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle \in D^{ass^\iota_o}_{\mathbf{I}^\theta_\Sigma}(\delta) \text{ iff } \langle \beta, \varrho, \lambda, \xi \rangle/\pi \in \Delta^\iota_\Sigma(\delta).$$

by arithmetic induction on the complexity of δ

(since, for each $\pi' \in \overline{\beta/\pi}$, $ass^\iota_o\frac{\langle \beta,\varrho,\lambda,\xi,|\pi\pi'|_\varrho\rangle}{v} = ass_o^{\iota\frac{\pi'}{v}}$)

Thus, for each signature Σ, for each $\theta \subseteq \mathcal{D}^\Sigma_0$, for each $\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle \in \mathcal{C}_\Sigma$,

$$\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle \in \mathbf{U}^\theta_\Sigma,$$
$$\Longrightarrow \langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle \in \mathbf{U}^\theta_\Sigma,\ \langle \beta, \varrho, \lambda, \xi \rangle \in M_\Sigma(\theta) \text{ and } \pi \in \beta$$
$$\Longrightarrow \langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle \in \mathbf{U}^\theta_\Sigma,$$
for each $\delta \in \theta$, for each total function ι from $\mathcal{VAR}$ to $\overline{\beta/\pi}$,
$$\langle \beta, \varrho, \lambda, \xi \rangle/\pi \in \Delta^\iota_\Sigma(\delta)$$
$\Longrightarrow$ *for each $\delta \in \theta$, for each total function ι from $\mathcal{VAR}$ to $\overline{\beta/\pi}$,*
$$\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle \in D^{ass^\iota_o}_{\mathbf{I}^\theta_\Sigma}(\delta)$$
$\Longrightarrow$ *for each $\delta \in \theta$, for some $ass \in Ass_{\mathbf{I}^\theta_\Sigma}$,*
$$\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle \in D^{ass}_{\mathbf{I}^\theta_\Sigma}(\delta)$$
$\Longrightarrow$ *for each $\delta \in \theta$, for each $ass \in Ass_{\mathbf{I}^\theta_\Sigma}$,*
$$\langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle \in D^{ass}_{\mathbf{I}^\theta_\Sigma}(\delta) \qquad \text{by corollary 2}$$

$\Longrightarrow \langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle \in \Theta_{\mathbf{I}_\Sigma^\theta}(\theta)$. ■

Next I must show that $\mathbf{I}_\Sigma^\theta$ is not just an arbitrary $\langle \Sigma, \theta \rangle$ model; it is an *exhaustive* $\langle \Sigma, \theta \rangle$ model. As explained at length in Section 3.1.2.3, I prove this by showing that for each entity u in each $\langle \Sigma, \theta \rangle$ model, there is an entity in $\mathbf{I}_\Sigma^\theta$ that abstracts to the same Σ morph as u. I start with the definition of the abstraction relations with respect to an interpretation:

Definition 82 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$,

> $\mathsf{Abst_I}$ *is the binary relation on* $\mathsf{U} \times \mathcal{M}_\Sigma$ *such that, for each* $u \in \mathsf{U}$, *for each* $\langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma$, $\langle u, \langle \beta, \varrho, \lambda, \xi \rangle \rangle \in \mathsf{Abst_I}$ *iff*
>
> $$\beta = \left\{ \pi \in \mathcal{A}^* \,\middle|\, T_{\mathsf{I}}(\pi, u) \textit{ is defined} \right\},$$
>
> $$\varrho = \left\{ \langle \pi_1, \pi_2 \rangle \in \mathcal{A}^* \times \mathcal{A}^* \,\middle|\, \begin{array}{l} T_{\mathsf{I}}(\pi_1, u) \textit{ is defined,} \\ T_{\mathsf{I}}(\pi_2, u) \textit{ is defined, and} \\ T_{\mathsf{I}}(\pi_1, u) = T_{\mathsf{I}}(\pi_2, u) \end{array} \right\},$$
>
> $$\lambda = \left\{ \langle \pi, \sigma \rangle \in \mathcal{A}^* \times \mathcal{S} \,\middle|\, \begin{array}{l} T_{\mathsf{I}}(\pi, u) \textit{ is defined,} \\ \mathsf{S}(T_{\mathsf{I}}(\pi, u)) = \sigma \end{array} \right\},$$
>
> $$\xi = \left\{ \langle \rho, \pi_1, \ldots, \pi_n \rangle \in \mathcal{R} \times (\overline{\mathcal{A}^*})^* \,\middle|\, \begin{array}{l} T_{\mathsf{I}}(\pi_1, u) \textit{ is defined, } \ldots, \\ T_{\mathsf{I}}(\pi_n, u) \textit{ is defined, and} \\ \langle T_{\mathsf{I}}(\pi_1, u), \ldots, T_{\mathsf{I}}(\pi_n, u) \rangle \in \mathsf{R}(\rho) \end{array} \right\}.$$

Proposition 16 *For each signature* Σ, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$,

> $\mathsf{Abst_I}$ *is a total function from* U *to* $\mathcal{M}_\Sigma$.

The proof that $\mathbf{I}_\Sigma^\theta$ is an exhaustive $\langle \Sigma, \theta \rangle$ model uses the fact that $\mathbf{I}_\Sigma^\theta$ simulates every other $\langle \Sigma, \theta \rangle$ model. Simulation can only be inferred from abstraction of two entities to the same morph if two congruent entities in two interpretations always abstract to the same morph and if two entities in two interpretations that abstract to the same morph are necessarily congruent.

PROPOSITION 17 claims that the abstraction relations with respect to an interpretation have this property:

Proposition 17 *For each signature* Σ, *for each* Σ *interpretation* $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1 \rangle$, *for each* Σ *interpretation* $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2 \rangle$, *for each* $o_1 \in \mathsf{U}_1$, *for each* $o_2 \in \mathsf{U}_2$,

$$\begin{aligned} &\mathsf{Abst}_{\mathsf{I}_1}(o_1) = \mathsf{Abst}_{\mathsf{I}_2}(o_2) \\ &\textit{iff } \langle o_1, \mathsf{I}_1 \rangle \textit{ and } \langle o_2, \mathsf{I}_2 \rangle \textit{ are congruent in } \Sigma. \end{aligned}$$

Proof

Firstly, for each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each* Σ *interpretation* $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1 \rangle$, *for each* Σ *interpretation* $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2 \rangle$, *for each* $o_1 \in \mathsf{U}_1$, *for each* $o_2 \in \mathsf{U}_2$,

$$\begin{aligned} &\mathsf{Abst}_{\mathsf{I}_1}(o_1) = \mathsf{Abst}_{\mathsf{I}_2}(o_2) \\ \Longrightarrow\ &\left\{ \langle o'_1, o'_2 \rangle \in \overline{\mathsf{U}_1} \times \overline{\mathsf{U}_2} \;\middle|\; \begin{array}{l} \textit{for some } \pi \in \overline{\mathcal{A}^*}, \\ \quad T_{\mathsf{I}_1}(\pi, o_1) \textit{ is defined,} \\ \quad T_{\mathsf{I}_2}(\pi, o_2) \textit{ is defined,} \\ \quad o'_1 = T_{\mathsf{I}_1}(\pi, o_1), \textit{ and} \\ \quad o'_2 = T_{\mathsf{I}_2}(\pi, o_2) \end{array} \right\} \\ &\quad \textit{is a congruence from } \langle o_1, \mathsf{I}_1 \rangle \textit{ to } \langle o_2, \mathsf{I}_2 \rangle \textit{ in } \Sigma \\ \Longrightarrow\ &\langle o_1, \mathsf{I}_1 \rangle \textit{ and } \langle o_2, \mathsf{I}_2 \rangle \textit{ are congruent in } \Sigma. \end{aligned}$$

Secondly, for each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each* Σ *interpretation* $\mathsf{I}_1 = \langle \mathsf{U}_1, \mathsf{S}_1, \mathsf{A}_1, \mathsf{R}_1 \rangle$, *for each* Σ *interpretation* $\mathsf{I}_2 = \langle \mathsf{U}_2, \mathsf{S}_2, \mathsf{A}_2, \mathsf{R}_2 \rangle$, *for each* $o_1 \in \mathsf{U}_1$, *for each* $o_2 \in \mathsf{U}_2$,

$$\begin{aligned} &\langle o_1, \mathsf{I}_1 \rangle \textit{ and } \langle o_2, \mathsf{I}_2 \rangle \textit{ are congruent in } \Sigma \\ \Longrightarrow\ &\textit{for some } f,\ f \textit{ is a congruence from } \langle o_1, \mathsf{I}_1 \rangle \textit{ to } \langle o_2, \mathsf{I}_2 \rangle \\ \Longrightarrow\ &\textit{for some congruence } f \textit{ from } \langle o_1, \mathsf{I}_1 \rangle \textit{ to } \langle o_2, \mathsf{I}_2 \rangle, \textit{ for each } \pi \in \mathcal{A}^*, \\ &\quad T_{\mathsf{I}_1}(\pi, o_1) \textit{ is defined iff } T_{\mathsf{I}_2}(\pi, o_2) \textit{ is defined, and} \\ &\quad \textit{if } T_{\mathsf{I}_1}(\pi, o_1) \textit{ is defined then } f(T_{\mathsf{I}_1}(\pi, o_1)) = T_{\mathsf{I}_2}(\pi, o_2) \\ &\hfill \textit{by induction on the length of } \pi \end{aligned}$$

$$\Longrightarrow \mathsf{Abst}_{\mathsf{I}_1}(o_1) = \mathsf{Abst}_{\mathsf{I}_2}(o_2). \qquad \blacksquare$$

The following lemmata, propositions, and definitions all prepare PROPOSITION 18, which ensures that each entity in a $\langle \Sigma, \theta \rangle$ model abstracts to a Σ morph that is admitted by θ in Σ. To show that, a number of preliminary results must be provided.

These preliminary results essentially come in two groups. The first group, LEMMA 2, DEFINITION 100 with PROPOSITION 25, and LEMMA 3 prepare PROPOSITION 26, which says that for each entity o in a Σ interpretation I and for each Σ morph μ, if we restrict the Σ insertion ι to paths and tuples of paths in the basis set of μ, and if μ is the abstraction of o, then μ satisfies each Σ formula δ under ι exactly if o is described by δ in I under some appropriately chosen variable assignment in I. In the second group, LEMMA 4 and LEMMA 5 lead to PROPOSITION 27, which says that we obtain the same result if we first interpret a path π on an entity and then take its abstraction as if we first abstract a morph from the entity and then take the π reduct of the morph.

LEMMA 2 states that for a Σ morph μ to satisfy each Σ formula δ with respect to a morph satisfaction function under a Σ insertion ι_1 exactly if it satisfies δ with respect to a morph satisfaction function under some other Σ insertion ι_2, it is not necessary that ι_1 and ι_2 agree on all free variables in δ; it suffices if ι_1 and ι_2 only assign paths or tuples of paths from the basis set of μ to the free variables, and if the paths or tuples of paths that they assign to the same free variables in δ are re-entrant, i.e., they are in the extension, $\hat{\varrho}$, of the re-entrancy relation, ϱ, in μ.

Lemma 2 *For each signature Σ, for each $\langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma$,*

for each $\tau \in \mathcal{T}^\Sigma$, for each $\iota_1 \in \mathrm{I}_\Sigma$, for each $\iota_2 \in \mathrm{I}_\Sigma$,

if for each $v \in \mathcal{VAR}$, $\iota_1(v) \in \overline{\beta}$ and $\iota_2(v) \in \overline{\beta}$, and for each $v \in FV(\tau)$, $\langle \iota_1(v), \iota_2(v) \rangle \in \hat{\varrho}$

then $\Pi^{\iota_1}_\Sigma(\tau)$ is defined and $\Pi^{\iota_1}_\Sigma(\tau) \in \overline{\beta}$ iff $\Pi^{\iota_2}_\Sigma(\tau)$ is defined and $\Pi^{\iota_2}_\Sigma(\tau) \in \overline{\beta}$, and

if $\Pi^{\iota_1}_\Sigma(\tau)$ is defined and $\Pi^{\iota_1}_\Sigma(\tau) \in \overline{\beta}$ then $\langle \Pi^{\iota_1}_\Sigma(\tau), \Pi^{\iota_2}_\Sigma(\tau) \rangle \in \hat{\varrho}$, and *for each* $\delta \in \mathcal{D}^{\Sigma}$, *for each* $\iota_1 \in \mathrm{I}_{\Sigma}$, *for each* $\iota_2 \in \mathrm{I}_{\Sigma}$,

if for each $v \in \mathcal{VAR}$, $\iota_1(v) \in \overline{\beta}$ *and* $\iota_2(v) \in \overline{\beta}$, *and*
for each $v \in FV(\delta)$, $\langle \iota_1(v), \iota_2(v) \rangle \in \hat{\varrho}$

then $\langle \beta, \varrho, \lambda, \xi \rangle \in \Delta^{\iota_1}_{\Sigma}(\delta)$ *iff* $\langle \beta, \varrho, \lambda, \xi \rangle \in \Delta^{\iota_2}_{\Sigma}(\delta)$.

Proof
By proposition 24 and induction on the length of τ*, and by induction on the complexity of* δ*, respectively.* ■

In Definition 100, I define a collection of functions, some of which will be used as variable assignment functions. They are functions from variables to entities in an interpretation I that are constructed in such a way that the values assigned to the variables are all components or chains of components of some entity o in I; and the components are determined by interpreting a path that is inserted for a variable by an insertion ι:

Definition 100 *For each signature* Σ, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, *for each* $o \in \mathsf{U}$, *for each* $\iota \in \mathrm{I}_{\Sigma}$,

$$ass^{\iota}_{o,\mathsf{I}} = \left\{ \langle v, o' \rangle \in \mathcal{VAR} \times \overline{\mathsf{U}} \;\middle|\; \begin{array}{l} T_{\mathsf{I}}(\Pi^{\iota}_{\Sigma}(v), o) \textit{ is defined, and} \\ T_{\mathsf{I}}(\Pi^{\iota}_{\Sigma}(v), o) = o' \end{array} \right\}.$$

With some restrictions on the choice of the insertion ι, the interpretation I, and the entity o in I, $ass^{\iota}_{o,\mathsf{I}}$ is a variable assignment in I:

Proposition 25 *For each signature* Σ, *for each* Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, *for each* $o \in \mathsf{U}$, *for each* $\langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_{\Sigma}$, *for each* $\iota \in \mathrm{I}_{\Sigma}$,

if for each $v \in \mathcal{VAR}$, $\iota(v) \in \overline{\beta}$, *and*
$\mathsf{Abst}_{\mathsf{I}}(o) = \langle \beta, \varrho, \lambda, \xi \rangle$

then $ass^{\iota}_{o,\mathsf{I}} \in Ass_{\mathsf{I}}$.

In the contexts in which I will need $ass^{\iota}_{o,\mathsf{I}}$, the relevant restrictions are fulfilled.

The following lemma builds on these restrictions. It says that if they hold, then each term is defined on the entity exactly if the path that the path insertion inserts for the term is defined on the entity, and if the term is defined on the entity, then the interpretation of the term on the entity (using the term interpretation function) yields the same value as the interpretation of the inserted path on the entity (using the extended path interpretation function).

Lemma 3 *For each signature Σ, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, for each $o \in \mathsf{U}$, for each $\langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma$, for each $\tau \in \mathcal{T}^\Sigma$, for each $\iota \in \mathrm{I}_\Sigma$,*

if for each $v \in \mathcal{VAR}$, $\iota(v) \in \overline{\beta}$, and
$\mathsf{Abst}_\mathsf{I}(o) = \langle \beta, \varrho, \lambda, \xi \rangle$

then $T_\mathsf{I}^{ass^\iota_{o,\mathsf{I}}}(\tau)(o)$ is defined iff $T_\mathsf{I}(\Pi^\iota_\Sigma(\tau), o)$ is defined, and
if $T_\mathsf{I}^{ass^\iota_{o,\mathsf{I}}}(\tau)(o)$ is defined then $T_\mathsf{I}^{ass^\iota_{o,\mathsf{I}}}(\tau)(o) = T_\mathsf{I}(\Pi^\iota_\Sigma(\tau), o)$.

Proof
By proposition 24 and induction on the length of τ. ■

With the proof of LEMMA 3, I come to the proposition that I have been preparing since LEMMA 2:

Proposition 26 *For each signature Σ, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, for each $o \in \mathsf{U}$, for each $\langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma$, for each $\delta \in \mathcal{D}^\Sigma$, for each $\iota \in \mathrm{I}_\Sigma$,*

if for each $v \in \mathcal{VAR}$, $\iota(v) \in \overline{\beta}$, and
$\mathsf{Abst}_\mathsf{I}(o) = \langle \beta, \varrho, \lambda, \xi \rangle$

then $o \in D_\mathsf{I}^{ass^\iota_{o,\mathsf{I}}}(\delta)$ iff $\langle \beta, \varrho, \lambda, \xi \rangle \in \Delta^\iota_\Sigma(\delta)$.

Proof
For each signature Σ, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, for each $o \in \mathsf{U}$, for each $o' \in \overline{\mathsf{Co}^o_\mathsf{I}}$, let $\#(o')$ be a $\pi \in \overline{\beta}$ such that

$o' = T_\mathsf{I}(\pi, o)$.

We can then show that for each signature Σ, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, for each $o \in \mathsf{U}$, for each $\langle \beta, \varrho, \lambda, \xi \rangle \in \mathcal{M}_\Sigma$, for each $\iota \in \mathrm{I}_\Sigma$, *if for each* $v \in \mathcal{VAR}$, $\iota(v) \in \overline{\beta}$, *and*

$\mathsf{Abst_I}(o) = \langle \beta, \varrho, \lambda, \xi \rangle$

then for each $o' \in \overline{\mathsf{Co}^o_{\mathsf{I}}}$, *for each* $v \in \mathcal{VAR}$, $ass^{\iota}_{o,\mathsf{I}}\frac{o'}{v} = ass^{\iota\frac{\#(o')}{v}}_{o,\mathsf{I}}$, *and*

for each $\pi \in \overline{\beta}$, *for each* $o' \in \overline{\mathsf{Co}^o_{\mathsf{I}}}$,

$o' = T_{\mathsf{I}}(\pi, o)$

$\Longrightarrow T_{\mathsf{I}}(\#(o'), o) = T_{\mathsf{I}}(\pi, o)$

$\Longrightarrow \langle \#(o'), \pi \rangle \in \widehat{\varrho}$

$\Longrightarrow$ *for each* $\delta \in \mathcal{D}^{\Sigma}$,

$$\langle \beta, \varrho, \lambda, \xi \rangle \in \Delta^{\iota\frac{\#(o')}{v}}_{\Sigma}(\delta) \text{ iff } \langle \beta, \varrho, \lambda, \xi \rangle \in \Delta^{\iota\frac{\pi}{v}}_{\Sigma}(\delta).$$

by lemma 2

Using this result, the proposition follows by lemma 3 and induction on the complexity of δ. ■

The two lemmata, LEMMA 4 and LEMMA 5, that lead to PROPOSITION 27 are closely related. The first lemma says that the interpretation of a path consisting of the prefix π and the tail π' on an entity is defined exactly if the interpretation of the prefix on the entity is defined, yielding another entity as value, and the interpretation of the tail is defined on that entity. And if the overall path is defined on the entity, then its interpretation on the entity yields the same value as the stepwise interpretation of the prefix and the tail. The second lemma generalizes this result to the case where the tail of the first lemma is a tuple of paths instead of a path.

Lemma 4 *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, *for each* $o \in \mathsf{U}$, *for each* $\pi \in \mathcal{A}^*$, *for each* $\pi' \in \mathcal{A}^*$,

$T_{\mathsf{I}}(\pi\pi', o)$ *is defined iff* $T_{\mathsf{I}}(\pi', T_{\mathsf{I}}(\pi, o))$ *is defined, and*

if $T_{\mathsf{I}}(\pi\pi', o)$ *is defined then* $T_{\mathsf{I}}(\pi\pi', o) = T_{\mathsf{I}}(\pi', T_{\mathsf{I}}(\pi, o))$.

Proof
By induction on the length of π'. ■

Lemma 5 *For each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, for each interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, for each $o \in \mathsf{U}$, for each $\pi \in \mathcal{A}^*$, for each $\pi' \in \mathcal{A}^{**}$,*

> *$T_\mathsf{I}(\pi\pi', o)$ is defined iff $T_\mathsf{I}(\pi', T_\mathsf{I}(\pi, o))$ is defined, and*
>
> *if $T_\mathsf{I}(\pi\pi', o)$ is defined then $T_\mathsf{I}(\pi\pi', o) = T_\mathsf{I}(\pi', T_\mathsf{I}(\pi, o))$.*

Proof
Follows from lemma 4. ■

Proposition 27 *For each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, for each $o \in \mathsf{U}$, for each $\pi \in \mathcal{A}^*$,*

> *if $T_\mathsf{I}(\pi, o)$ is defined then $\mathsf{Abst}_\mathsf{I}(T_\mathsf{I}(\pi, o)) = \mathsf{Abst}_\mathsf{I}(o)/\pi$.*

Proof
Uses lemma 4 and lemma 5. ■

All preliminary results that are necessary to show PROPOSITION 18 are now in place.

Proposition 18 *For each signature Σ, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, for each $\theta \subseteq \mathcal{D}_0^\Sigma$,*

> *if I is a $\langle \Sigma, \theta \rangle$ model*
>
> *then for each $o \in \mathsf{U}$, $\mathsf{Abst}_\mathsf{I}(o) \in M_\Sigma(\theta)$.*

Proof
For each signature Σ, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, for each $\theta \subseteq \mathcal{D}_0^\Sigma$,

> *I is a $\langle \Sigma, \theta \rangle$ model*

$\Longrightarrow$ *for each* $o \in \mathsf{U}$, *for each* $\langle\beta, \varrho, \lambda, \xi\rangle \in \mathcal{M}_\Sigma$,

$\quad\langle\beta, \varrho, \lambda, \xi\rangle = \mathsf{Abst}_{\mathsf{I}}(o)$

$\quad\Longrightarrow$ *for each* $\pi \in \beta$,

$\quad\quad\langle\beta, \varrho, \lambda, \xi\rangle/\pi = \mathsf{Abst}_{\mathsf{I}}(T_{\mathsf{I}}(\pi, o))$ *by proposition 27*

$\quad\Longrightarrow$ *for each* $\pi \in \beta$, *for each* $ass \in Ass_{\mathsf{I}}$, *for each* $\delta \in \theta$,

$\quad\quad\langle\beta, \varrho, \lambda, \xi\rangle/\pi = \mathsf{Abst}_{\mathsf{I}}(T_{\mathsf{I}}(\pi, o))$ *and* $T_{\mathsf{I}}(\pi, o) \in D_{\mathsf{I}}^{ass}(\delta)$

$\quad\Longrightarrow$ *for each* $\pi \in \beta$, *for each total function* $\iota \colon \mathcal{VAR} \to \overline{\beta/\pi}$, *for each* $\delta \in \theta$,

$\quad\quad\langle\beta, \varrho, \lambda, \xi\rangle/\pi = \mathsf{Abst}_{\mathsf{I}}(T_{\mathsf{I}}(\pi, o))$ *and* $T_{\mathsf{I}}(\pi, o) \in D_{\mathsf{I}}^{ass^{\iota}_{T_{\mathsf{I}}(\pi,o),\mathsf{I}}}(\delta)$ *by proposition 25*

$\quad\Longrightarrow$ *for each* $\pi \in \beta$, *for each total function* $\iota \colon \mathcal{VAR} \to \overline{\beta/\pi}$, *for each* $\delta \in \theta$,

$\quad\quad\langle\beta, \varrho, \lambda, \xi\rangle/\pi \in \Delta^{\iota}_{\Sigma}(\delta)$ *by proposition 26*

$\quad\Longrightarrow$ *for each* $\pi \in \beta$, *for some* $\iota \in \mathrm{I}_\Sigma$, *for each* $\delta \in \theta$,

$\quad\quad\langle\beta, \varrho, \lambda, \xi\rangle/\pi \in \Delta^{\iota}_{\Sigma}(\delta)$ *since* $\beta/\pi \neq \emptyset$

$\quad\Longrightarrow$ *for each* $\pi \in \beta$, *for each* $\iota \in \mathrm{I}_\Sigma$, *for each* $\delta \in \theta$,

$\quad\quad\langle\beta, \varrho, \lambda, \xi\rangle/\pi \in \Delta^{\iota}_{\Sigma}(\delta)$ *by proposition 13*

$\quad\Longrightarrow \langle\beta, \varrho, \lambda, \xi\rangle \in M_\Sigma(\theta)$

$\Longrightarrow$ *for each* $o \in \mathsf{U}$, $\mathsf{Abst}_{\mathsf{I}}(o) \in M_\Sigma(\theta)$. *by proposition 16*

■

PROPOSITION 19 is the last piece in the puzzle that I need in order to show that each $\mathbf{I}^{\theta}_{\Sigma}$ is an exhaustive $\langle\Sigma, \theta\rangle$ model. It tells us which entity in $\mathbf{I}^{\theta}_{\Sigma}$ to choose when we are looking for an entity that abstracts to a given morph:

Proposition 19 *For each signature* Σ, *for each* $\theta \subseteq \mathcal{D}^{\Sigma}_{0}$, *for each* $\langle\beta, \varrho, \lambda, \xi\rangle \in M_\Sigma(\theta)$,

$$\mathsf{Abst}_{\mathbf{I}^{\theta}_{\Sigma}}(\langle\beta, \varrho, \lambda, \xi, |\varepsilon|_{\varrho}\rangle) = \langle\beta, \varrho, \lambda, \xi\rangle.$$

Proof
Follows from the observation that, for each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each* $\theta \subseteq \mathcal{D}_0^\Sigma$, *for each* $\langle \beta, \varrho, \lambda, \xi, |\varepsilon|_\varrho \rangle \in \mathbf{U}_\Sigma^\theta$, *for each* $\pi \in \mathcal{A}^*$,

$\pi \in \beta$ *iff* $T_{\mathbf{I}_\Sigma^\theta}(\pi, \langle \beta, \varrho, \lambda, \xi, |\varepsilon|_\varrho \rangle)$ *is defined, and*

if $\pi \in \beta$ *then* $T_{\mathbf{I}_\Sigma^\theta}(\pi, \langle \beta, \varrho, \lambda, \xi, |\varepsilon|_\varrho \rangle) = \langle \beta, \varrho, \lambda, \xi, |\pi|_\varrho \rangle$.
by induction on the length of π

■

Proposition 20 *For each signature* Σ, *for each* $\theta \subseteq \mathcal{D}_0^\Sigma$,

$\mathbf{I}_\Sigma^\theta$ *is an exhaustive* $\langle \Sigma, \theta \rangle$ *model.*

Proof
For each signature Σ, *for each* $\theta \subseteq \mathcal{D}_0^\Sigma$,

$\mathbf{I}_\Sigma^\theta$ *is a* $\langle \Sigma, \theta \rangle$ *model, and* *by proposition 15*

for each Σ *interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$,

I *is a* $\langle \Sigma, \theta \rangle$ *model*

$\Longrightarrow$ *for each* $o \in \mathsf{U}$,
$\mathsf{Abst}_\mathsf{I}(o) \in M_\Sigma(\theta)$ *by proposition 18*

$\Longrightarrow$ *for each* $o \in \mathsf{U}$, *for some* $\langle \beta, \varrho, \lambda, \xi \rangle \in M_\Sigma(\theta)$,
$\mathsf{Abst}_\mathsf{I}(o) = \langle \beta, \varrho, \lambda, \xi \rangle$, *and*
$\mathsf{Abst}_{\mathbf{I}_\Sigma^\theta}(\langle \beta, \varrho, \lambda, \xi, |\varepsilon|_\varrho \rangle) = \langle \beta, \varrho, \lambda, \xi \rangle$ *by proposition 19*

$\Longrightarrow$ *for each* $o \in \mathsf{U}$, *for some* $o' \in \mathbf{U}_\Sigma^\theta$,
$\langle o', \mathbf{I}_\Sigma^\theta \rangle$ *and* $\langle o, \mathsf{I} \rangle$ *are congruent in* Σ *by proposition 17*

$\Longrightarrow$ $\mathbf{I}_\Sigma^\theta$ *simulates* I *in* Σ

$\Longrightarrow$ $\mathbf{I}_\Sigma^\theta$ *is an exhaustive* $\langle \Sigma, \theta \rangle$ *model.* *by theorem 3*

■

Theorem 2 *For each signature Σ, for each theory θ, there exists a Σ interpretation I such that*

> I *is an exhaustive $\langle \Sigma, \theta \rangle$ model.*

It is an immediate corollary of THEOREM 2 and the definition of an exhaustive model (DEFINITION 64) that if a grammar has a nonempty model then it has a nonempty exhaustive model.

Appendix B

Proof: AVM Syntax for RSRL

This appendix contains the technical details of the proofs that lead to THEOREM 4, which essentially says that the syntax of RSRL as defined in Section 3.1.1 and the AVM syntax of Section 3.2.1 are notational variants of each other. To prove that result, I show first that the translation function, trans, from Σ AVM formulae to Σ formulae (DEFINITION 91) preserves free occurrences of variables. Then I use this to show that for each Σ formula there is a Σ AVM formula with the same free variables and the same denotation in each interpretation, if the variable assignment function is restricted to components and chains of components of the described entity. Since that condition holds for the denotation of all variables in formulae without free variables, and Σ AVM descriptions are translated into Σ descriptions, the theorem will follow immediately from that result. For convenience, I repeat the lemmata and propositions of Section 3.2.2 before giving the proofs.

Before I can show that the translation of AVM formulae preserves free occurrences of variables, I have to show that the translation of each pair of an AVM box and the term that leads to it preserves free occurrences of variables.

Lemma 1 *For each signature Σ, for each $\tau \in \mathcal{T}^\Sigma$, for each $\beta \in \mathbb{BOX}^\Sigma$,*

$$FV\left(\mathsf{btrans}(\tau, \beta)\right) = FV(\tau) \cup FV^{AVM}(\beta).$$

Proof

Proof by induction on the complexity of β. Suppose that $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$ is a signature, and $\tau \in \mathcal{T}^\Sigma$.

Firstly, for each $\sigma \in \widehat{\mathcal{G}}$*, for each* $\alpha_1 \in \widehat{\mathcal{A}}$*, ..., for each* $\alpha_n \in \widehat{\mathcal{A}}$*, for each* $\beta_1 \in \mathbb{BOX}^\Sigma$*, ..., for each* $\beta_n \in \mathbb{BOX}^\Sigma$ *where we have already shown the result for* β_1*, ...,* β_n*,*

$$
\begin{aligned}
& FV\left(\mathsf{btrans}\left(\tau, \begin{bmatrix} \sigma & \\ \alpha_1 & \beta_1 \\ \vdots & \\ \alpha_n & \beta_n \end{bmatrix}\right)\right) \\
&= FV\left(\mathsf{utrans}\left(\tau, \begin{bmatrix} \sigma & \\ \alpha_1 & \beta_1 \\ \vdots & \\ \alpha_n & \beta_n \end{bmatrix}\right)\right) \\
&= FV\left(\tau \sim \sigma \wedge \bigwedge\left\{ \mathsf{btrans}(\tau\alpha_i, \beta_i) \,\middle|\, i \in \{1, \dots, n\} \right\}\right) \\
&= FV(\tau) \cup FV\left(\bigwedge\left\{ \mathsf{btrans}(\tau\alpha_i, \beta_i) \,\middle|\, i \in \{1, \dots, n\} \right\}\right) \\
&= FV(\tau) \cup FV^{AVM}(\beta_1) \cup \dots \cup FV^{AVM}(\beta_n) \\
&= FV(\tau) \cup FV^{AVM}\left(\begin{bmatrix} \sigma & \\ \alpha_1 & \beta_1 \\ \vdots & \\ \alpha_n & \beta_n \end{bmatrix}\right).
\end{aligned}
$$

Secondly, for each $\beta_1 \in \mathbb{BOX}^\Sigma$*, ..., for each* $\beta_n \in \mathbb{BOX}^\Sigma$ *where we have already shown the result for* $\beta_1, \dots, \beta_n$*,*

$$
\begin{aligned}
& FV(\mathsf{btrans}(\tau, \langle \beta_1, \dots, \beta_n \rangle)) \\
&= FV(\mathsf{utrans}(\tau, \langle \beta_1, \dots, \beta_n \rangle)) \\
&= FV\left(\begin{array}{l} \tau \underbrace{\text{REST} \dots \text{REST}}_{n \ \textit{times}} \sim \textit{elist} \wedge \\ \bigwedge\left\{ \mathsf{btrans}(\tau \underbrace{\text{REST} \dots \text{REST}}_{i-1 \ \textit{times}} \ \text{FIRST}, \beta_i) \,\middle|\, i \in \{1, \dots, n\} \right\} \end{array}\right) \\
&= FV(\tau) \cup FV\left(\bigwedge\left\{ \mathsf{btrans}(\tau \underbrace{\text{REST} \dots \text{REST}}_{i-1 \ \textit{times}} \ \text{FIRST}, \beta_i) \,\middle|\, i \in \{1, \dots, n\} \right\}\right) \\
&= FV(\tau) \cup FV^{AVM}(\beta_1) \cup \dots \cup FV^{AVM}(\beta_n) \\
&= FV(\tau) \cup FV^{AVM}(\langle \beta_1, \dots, \beta_n \rangle).
\end{aligned}
$$

Thirdly, for each $\beta \in \mathbb{BOX}^\Sigma$ *where we have already shown the result for* β*,*

$$\begin{aligned}
& FV(\mathsf{btrans}(\tau, \neg\beta)) \\
& = FV(\mathsf{utrans}(\tau, \neg\beta)) \\
& = FV(\tau \approx \tau \;\wedge\; \neg\mathsf{btrans}(\tau, \beta)) \\
& = FV(\tau) \cup FV(\tau) \cup FV^{AVM}(\beta) \\
& = FV(\tau) \cup FV^{AVM}(\beta).
\end{aligned}$$

Fourthly, for each $\beta_1 \in \mathbb{BOX}^{\Sigma}$, ..., *for each* $\beta_n \in \mathbb{BOX}^{\Sigma}$ *where we have already shown the result for* $\beta_1, \ldots, \beta_n$,

$$\begin{aligned}
& FV(\mathsf{btrans}(\tau, [\beta_1 \wedge \beta_2])) \\
& = FV(\mathsf{utrans}(\tau, [\beta_1 \wedge \beta_2])) \\
& = FV(\mathsf{btrans}(\tau, \beta_1) \;\wedge\; \mathsf{btrans}(\tau, \beta_2)) \\
& = FV(\tau) \cup FV^{AVM}(\beta_1) \cup FV(\tau) \cup FV^{AVM}(\beta_2) \\
& = FV(\tau) \cup FV^{AVM}([\beta_1 \wedge \beta_2]).
\end{aligned}$$

(analogously for disjunction, implication, and bi-implication)

Fifthly, for each $v \in \mathcal{VAR}$, *for each* $\beta_1 \in \mathbb{BOX}^{\Sigma}$, ..., *for each* $\beta_n \in \mathbb{BOX}^{\Sigma}$ *where we have already shown the result for* $\beta_1, \ldots, \beta_n$,

$$\begin{aligned}
& FV(\mathsf{btrans}(\tau, v\, \|\beta_1, \ldots, \beta_n\|)) \\
& = FV(\mathsf{ttrans}(\tau, v\, \|\beta_1, \ldots, \beta_n\|)) \\
& = FV\left(\begin{array}{l} \tau \approx v \wedge \tau \underbrace{\triangleright \ldots \triangleright}_{n \ times} \sim elist \;\wedge \\ \bigwedge \left\{ \mathsf{btrans}(\tau \underbrace{\triangleright \ldots \triangleright}_{i-1 \ times} \dagger, \beta_i) \,\middle|\, i \in \{1, \ldots, n\} \right\} \end{array}\right) \\
& = FV(\tau) \cup FV(v) \cup FV\left(\bigwedge \left\{ \mathsf{btrans}(\tau \underbrace{\triangleright \ldots \triangleright}_{i-1 \ times} \dagger, \beta_i) \,\middle|\, i \in \{1, \ldots, n\} \right\}\right) \\
& = FV(\tau) \cup FV(v) \cup FV^{AVM}(\beta_1) \cup \ldots \cup FV^{AVM}(\beta_n) \\
& = FV(\tau) \cup FV^{AVM}(v\, \|\beta_1, \ldots, \beta_n\|).
\end{aligned}$$

Lastly, for each $v \in \mathcal{VAR}^{:}$, *for each* $\beta \in \mathbb{UBOX}^{\Sigma}$ *where we have already shown the result for* β,

$$\begin{aligned}
& FV(\mathsf{btrans}(\tau, v\ \beta)) \\
& = FV(\mathsf{ttrans}(\tau, v\ \beta)) \\
& = FV(\tau \approx v \;\wedge\; \mathsf{utrans}(\tau, \beta)) \\
& = FV(\tau) \cup FV(v) \cup FV(\tau) \cup FV^{AVM}(\beta) \\
& = FV(\tau) \cup FV^{AVM}(v\ \beta). \qquad \blacksquare
\end{aligned}$$

With LEMMA 1 the proof of PROPOSITION 21, which states that free occurrences of variables are preserved under the translation from Σ AVM formulae to Σ formulae, is a straightforward induction:

Proposition 21 *For each signature Σ, for each $\kappa \in \mathbb{AVM}^{\Sigma}$,*

$$FV^{AVM}(\kappa) = FV(\mathsf{trans}(\kappa)).$$

Proof

By induction on the complexity of κ and lemma 1. Suppose that $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$ is a signature.

Firstly, for each $v \in \mathcal{VAR}^{:}$, for each $\sigma \in \widehat{\mathcal{G}}$, for each $\alpha_1 \in \widehat{\mathcal{A}}$, …, for each $\alpha_n \in \widehat{\mathcal{A}}$, for each $\beta_1 \in \mathbb{BOX}^{\Sigma}$, …, for each $\beta_n \in \mathbb{BOX}^{\Sigma}$,

$$\begin{aligned}
& FV^{AVM}\left(v \begin{bmatrix} \sigma & \\ \alpha_1 & \beta_1 \\ \vdots & \\ \alpha_n & \beta_n \end{bmatrix} \right) \\
&= FV(v) \cup \left\{ v' \in \mathcal{VAR} \;\middle|\; \begin{array}{c} \textit{for some } i \in \{1, \ldots, n\}, \\ v' \in FV^{AVM}(\beta_i) \end{array} \right\} \\
&= FV(v) \cup FV(v) \cup \left\{ v' \in \mathcal{VAR} \;\middle|\; \begin{array}{c} \textit{for some } i \in \{1, \ldots, n\}, \\ v' \in FV^{AVM}(\beta_i) \end{array} \right\} \\
&= FV(v) \cup FV^{AVM}\left(v \begin{bmatrix} \sigma & \\ \alpha_1 & \beta_1 \\ \vdots & \\ \alpha_n & \beta_n \end{bmatrix} \right) \\
&= FV\left(\mathsf{btrans}\left(v, v \begin{bmatrix} \sigma & \\ \alpha_1 & \beta_1 \\ \vdots & \\ \alpha_n & \beta_n \end{bmatrix} \right) \right) && \textit{by lemma 1} \\
&= FV\left(\mathsf{trans}\left(v \begin{bmatrix} \sigma & \\ \alpha_1 & \beta_1 \\ \vdots & \\ \alpha_n & \beta_n \end{bmatrix} \right) \right).
\end{aligned}$$

(analogously for Σ AVM formulae of the form $v\,\langle \beta_1, \ldots, \beta_n \rangle$ and $v\,\|\beta_1, \ldots, \beta_n\|$, except that for chains, unlike for AVM matrices and lists, v cannot be colon)

Secondly, for each $v \in \mathcal{VAR}^{:}$, for each $\beta \in \mathbb{BOX}^{\Sigma}$,

$FV^{AVM}(v\ \neg\beta)$
$= FV(v) \cup FV^{AVM}(\beta)$
$= FV(v) \cup FV(v) \cup FV^{AVM}(\beta)$
$= FV(v) \cup FV^{AVM}(v\ \neg\beta)$
$= FV(\mathsf{btrans}(v, v\ \neg\beta))$ *by lemma 1*
$= FV(\mathsf{trans}(v\ \neg\beta))$.

Thirdly, for each $v \in \mathcal{VAR}^{:}$, *for each* $\beta_1 \in \mathbb{BOX}^{\Sigma}$, *for each* $\beta_2 \in \mathbb{BOX}^{\Sigma}$,

$FV^{AVM}(v\ [\beta_1 \wedge \beta_2])$
$= FV(v) \cup FV^{AVM}(\beta_1) \cup FV^{AVM}(\beta_2)$
$= FV(v) \cup FV(v) \cup FV^{AVM}(\beta_1) \cup FV^{AVM}(\beta_2)$
$= FV(v) \cup FV^{AVM}(v\ [\beta_1 \wedge \beta_2])$
$= FV(\mathsf{btrans}(v, v\ [\beta_1 \wedge \beta_2]))$ *by lemma 1*
$= FV(\mathsf{trans}(v\ [\beta_1 \wedge \beta_2]))$.
(analogously for disjunction, implication, and bi-implication)

Fourthly, for each $\rho \in \mathcal{R}$, *for each* $v_1 \in \mathcal{VAR}$, *…for each* $v_{\mathcal{AR}(\rho)} \in \mathcal{VAR}$,

$FV^{AVM}(\rho(v_1, \ldots, v_{\mathcal{AR}(\rho)}))$
$= \{v_1, \ldots, v_{\mathcal{AR}(\rho)}\}$
$= FV(\rho(v_1, \ldots, v_{\mathcal{AR}(\rho)}))$
$= FV(\mathsf{trans}(\rho(v_1, \ldots, v_{\mathcal{AR}(\rho)})))$.

Fifthly, for each $v_1 \in \mathcal{VAR}^{:}$, *for each* $v_2 \in \mathcal{VAR}^{:}$,

$FV^{AVM}(v_1 = v_2)$
$= FV(v_1) \cup FV(v_2)$
$= FV(v_1 \approx v_2)$
$= FV(\mathsf{trans}(v_1 = v_2))$.

Hence, for each $\kappa \in \mathbb{AVM}^{\Sigma}$, $FV^{AVM}(\kappa) = FV(\mathsf{trans}(\kappa))$.
by induction on the complexity of κ

■

In the proof of the following proposition I can now use the fact that the translation function, trans, preserves free occurrences of variables. PROPOSITION 22 says that for each Σ formula δ one can always find some Σ AVM formula in whose translation occur the same variables free as in δ and that has the same denotation in each interpretation, as long as the variable assignment function maps each variable to a component (or a chain of components) of the described entity.

Proposition 22 *For each signature* Σ, *for each* $\delta \in \mathcal{D}^{\Sigma}$, *there exists a* $\kappa \in \mathbb{AVM}^{\Sigma}$ *such that, for each interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, *for each* $u \in \mathsf{U}$, *for each* $ass \in Ass_{\mathsf{I}}$,

$$FV(\delta) = FV(\mathsf{trans}(\kappa)), \textit{ and}$$

$$\textit{if for each } v \in \mathcal{VAR},\ ass(v) \in \overline{\mathsf{Co}_{\mathsf{I}}^{u}},$$

$$\textit{then } u \in D_{\mathsf{I}}^{ass}(\delta) \textit{ iff } u \in D_{\mathsf{I}}^{ass}(\mathsf{trans}(\kappa)).$$

For each signature Σ and for each Σ formula, I have to specify a Σ AVM formula that has exactly the same denotation in each interpretation, given the restriction on variable assignments to components (and chains of components) of the described entity.

Proof

By induction on the complexity of δ. *For each signature* $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$, *for each interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, *for each* $u \in \mathsf{U}$, *for each* $ass \in Ass_{\mathsf{I}}$,

for each $v \in \mathcal{VAR}^{:}$, *for each* $n \in \mathbb{N}$, *for each* $\alpha_1 \in \widehat{\mathcal{A}}$, ..., *for each* $\alpha_n \in \widehat{\mathcal{A}}$, *for each* $\sigma \in \widehat{\mathcal{G}}$, *for each* $v\alpha_1 \ldots \alpha_n \sim \sigma$,

firstly, if $n = 0$,

$$FV(v \sim \sigma) = \{v\} = FV([v \approx v \wedge v \sim \sigma]) = FV(\mathsf{trans}(v[\sigma])), \textit{ and}$$

$$\begin{aligned}
& u \in D_{\mathsf{I}}^{ass}(v \sim \sigma) \\
\iff & u \in D_{\mathsf{I}}^{ass}([v \approx v \wedge v \sim \sigma]) \\
\iff & u \in D_{\mathsf{I}}^{ass}(\mathsf{ttrans}(v, v[\sigma])) \\
\iff & u \in D_{\mathsf{I}}^{ass}(\mathsf{trans}(v[\sigma])),
\end{aligned}$$

secondly, if $n > 0$,

$$
\begin{aligned}
& FV(v\alpha_1 \dots \alpha_n \sim \sigma) = \{v\} \\
& = FV\left(\begin{array}{l} v \approx v \ \wedge \ v \sim metatop \ \wedge \\ \bigwedge\left\{ v\alpha_1 \dots \alpha_i \sim metatop \,\middle|\, i \in \{1, \dots, n-1\} \right\} \ \wedge \\ v\alpha_1 \dots \alpha_n \sim \sigma \end{array}\right) \\
& = FV\left(\mathsf{trans}\left(v\begin{bmatrix} metatop \\ \alpha_1 \ \dots \begin{bmatrix} metatop \\ \alpha_n \ [\sigma] \end{bmatrix} \dots \end{bmatrix}\right)\right), \textit{ and}
\end{aligned}
$$

$$
\begin{aligned}
& u \in D^{ass}_{\mathsf{I}}(v\alpha_1 \dots \alpha_n \sim \sigma) \\
& \Longleftrightarrow T^{ass}_{\mathsf{I}}(v\alpha_1 \dots \alpha_n)(u) \textit{ is defined, and} \\
& \qquad \widehat{\mathsf{S}}(T^{ass}_{\mathsf{I}}(v\alpha_1 \dots \alpha_n)(u)) \mathrel{\widehat{\sqsubseteq}} \sigma \\
& \Longleftrightarrow T^{ass}_{\mathsf{I}}(v)(u) \textit{ is defined,} \\
& \qquad \textit{for each } i \in \{1, \dots, n\}, \\
& \qquad\qquad T^{ass}_{\mathsf{I}}(v\alpha_1 \dots \alpha_i)(u) \textit{ is defined,} \\
& \qquad\qquad \widehat{\mathsf{A}}(\alpha_i)(T^{ass}_{\mathsf{I}}(v\alpha_1 \dots \alpha_{i-1})(u)) \textit{ is defined,} \\
& \qquad\qquad T^{ass}_{\mathsf{I}}(v\alpha_1 \dots \alpha_i)(u) = \widehat{\mathsf{A}}(\alpha_i)(T^{ass}_{\mathsf{I}}(v\alpha_1 \dots \alpha_{i-1})(u)), \\
& \qquad \textit{for each } i \in \{0, \dots, n-1\}, \\
& \qquad\qquad \widehat{\mathsf{S}}(T^{ass}_{\mathsf{I}}(v\alpha_1 \dots \alpha_i)(u)) \mathrel{\widehat{\sqsubseteq}} metatop, \textit{ and} \\
& \qquad \widehat{\mathsf{S}}(T^{ass}_{\mathsf{I}}(v\alpha_1 \dots \alpha_n)(u)) \mathrel{\widehat{\sqsubseteq}} \sigma \\
& \Longleftrightarrow T^{ass}_{\mathsf{I}}(v)(u) \textit{ is defined,} \\
& \qquad \textit{for each } i \in \{1, \dots, n\}, \\
& \qquad\qquad T^{ass}_{\mathsf{I}}(v\alpha_1 \dots \alpha_i)(u) \textit{ is defined,} \\
& \qquad \textit{for each } i \in \{0, \dots, n-1\}, \\
& \qquad\qquad \widehat{\mathsf{S}}(T^{ass}_{\mathsf{I}}(v\alpha_1 \dots \alpha_i)(u)) \mathrel{\widehat{\sqsubseteq}} metatop, \textit{ and} \\
& \qquad \widehat{\mathsf{S}}(T^{ass}_{\mathsf{I}}(v\alpha_1 \dots \alpha_n)(u)) \mathrel{\widehat{\sqsubseteq}} \sigma \\
& \Longleftrightarrow T^{ass}_{\mathsf{I}}(v)(u) \textit{ is defined, } T^{ass}_{\mathsf{I}}(v)(u) = T^{ass}_{\mathsf{I}}(v)(u), \\
& \qquad \textit{for each } i \in \{0, \dots, n-1\},
\end{aligned}
$$

$$\begin{array}{l} T^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_i)(u) \textit{ is defined,} \\ \widehat{\mathsf{S}}(T^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_i)(u)) \mathrel{\widehat{\sqsubseteq}} \textit{metatop, and} \end{array}$$

$$T^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_n)(u) \textit{ is defined, } \widehat{\mathsf{S}}(T^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_n)(u)) \mathrel{\widehat{\sqsubseteq}} \sigma$$

$$\Longleftrightarrow u \in \left(\begin{array}{l} D^{ass}_{\mathsf{I}}(v \approx v) \cap \\ \bigcap \left\{ D^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_i \sim \textit{metatop}) \,\middle|\, i \in \{0, \ldots, n-1\} \right\} \cap \\ D^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_n \sim \sigma) \end{array} \right)$$

$$\Longleftrightarrow u \in D^{ass}_{\mathsf{I}} \left(\begin{array}{l} v \approx v \,\wedge \\ \bigwedge \left\{ v\alpha_1 \ldots \alpha_i \sim \textit{metatop} \,\middle|\, i \in \{0, \ldots, n-1\} \right\} \,\wedge \\ v\alpha_1 \ldots \alpha_n \sim \sigma \end{array} \right)$$

$$\Longleftrightarrow u \in D^{ass}_{\mathsf{I}} \left(\mathsf{ttrans} \left(v,\ v \begin{bmatrix} \textit{metatop} \\ \alpha_1 \ \ldots \begin{bmatrix} \textit{metatop} \\ \alpha_n \ [\sigma] \end{bmatrix} \ldots \end{bmatrix} \right) \right)$$

$$\Longleftrightarrow u \in D^{ass}_{\mathsf{I}} \left(\mathsf{trans} \left(v \begin{bmatrix} \textit{metatop} \\ \alpha_1 \ \ldots \begin{bmatrix} \textit{metatop} \\ \alpha_n \ [\sigma] \end{bmatrix} \ldots \end{bmatrix} \right) \right),$$

for each $v \in \mathcal{VAR}^{:}$, *for each* $v' \in \mathcal{VAR}^{:}$, *for each* $n \in \mathbb{N}$, *for each* $m \in \mathbb{N}$, *for each* $\alpha_1 \in \widehat{\mathcal{A}}$, …, *for each* $\alpha_n \in \widehat{\mathcal{A}}$, *for each* $\alpha'_1 \in \widehat{\mathcal{A}}$, …, *for each* $\alpha'_m \in \widehat{\mathcal{A}}$,

firstly, if $n = 0$ *and* $m = 0$,

$$FV(v \approx v') = \{v, v'\} = FV(\mathsf{trans}(v = v')), \textit{ and}$$

$$\begin{array}{l} u \in D^{ass}_{\mathsf{I}}(v \approx v') \\ \Longleftrightarrow u \in D^{ass}_{\mathsf{I}}(\mathsf{trans}(v = v')), \end{array}$$

secondly, if $n > 0$ *and* $m = 0$,

$$\begin{array}{l} FV(v\alpha_1 \ldots \alpha_n \approx v') \\ = \{v, v'\} \\ = FV \left(\begin{array}{l} v \approx v \,\wedge \\ \bigwedge \left\{ v\alpha_1 \ldots \alpha_i \sim \textit{metatop} \,\middle|\, i \in \{0, \ldots, n-1\} \right\} \,\wedge \\ v\alpha_1 \ldots \alpha_n \approx v' \,\wedge\, v\alpha_1 \ldots \alpha_n \sim \textit{metatop} \end{array} \right) \end{array}$$

$$= FV\left(\mathsf{trans}\left(v\begin{bmatrix} metatop \\ \alpha_1 \ \ldots \begin{bmatrix} metatop \\ \alpha_n \ v'\,[metatop] \end{bmatrix} \ldots \end{bmatrix}\right)\right), \textit{ and}$$

$u \in D^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_n \approx v')$

$\iff T^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_n)(u)$ *is defined,*

$T^{ass}_{\mathsf{I}}(v')(u)$ *is defined, and*

$T^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_n)(u) = T^{ass}_{\mathsf{I}}(v')(u)$

$\iff T^{ass}_{\mathsf{I}}(v)(u)$ *is defined,*

for each $i \in \{1, \ldots, n\}$,

$T^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_i)(u)$ *is defined,*

$\widehat{\mathsf{A}}(\alpha_i)(T^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_{i-1})(u))$ *is defined,*

$T^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_i)(u) = \widehat{\mathsf{A}}(\alpha_i)(T^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_{i-1})(u))$,

for each $i \in \{0, \ldots, n\}$,

$\widehat{\mathsf{S}}(T^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_i)(u)) \mathrel{\widehat{\sqsubseteq}} metatop$,

$T^{ass}_{\mathsf{I}}(v')(u)$ *is defined, and*

$T^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_n)(u) = T^{ass}_{\mathsf{I}}(v')(u)$

$\iff T^{ass}_{\mathsf{I}}(v)(u)$ *is defined,*

for each $i \in \{1, \ldots, n\}$,

$T^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_i)(u)$ *is defined,*

for each $i \in \{0, \ldots, n\}$,

$\widehat{\mathsf{S}}(T^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_i)(u)) \mathrel{\widehat{\sqsubseteq}} metatop$,

$T^{ass}_{\mathsf{I}}(v')(u)$ *is defined, and*

$T^{ass}_{\mathsf{I}}(v\alpha_1 \ldots \alpha_n)(u) = T^{ass}_{\mathsf{I}}(v')(u)$

$\iff T^{ass}_{\mathsf{I}}(v)(u)$ *is defined,* $T^{ass}_{\mathsf{I}}(v)(u) = T^{ass}_{\mathsf{I}}(v)(u)$,

for each $i \in \{0, \ldots, n\}$,

$T_{\mathsf{I}}^{ass}(v\alpha_1 \ldots \alpha_i)(u)$ *is defined,*
$\widehat{\mathsf{S}}(T_{\mathsf{I}}^{ass}(v\alpha_1 \ldots \alpha_i)(u)) \mathrel{\widehat{\sqsubseteq}}$ *metatop,*

$T_{\mathsf{I}}^{ass}(v\alpha_1 \ldots \alpha_n)(u)$ *is defined,* $T_{\mathsf{I}}^{ass}(v')(u)$ *is defined, and*
$T_{\mathsf{I}}^{ass}(v\alpha_1 \ldots \alpha_n)(u) = T_{\mathsf{I}}^{ass}(v')(u)$

$$\iff u \in \left(\begin{array}{l} D_{\mathsf{I}}^{ass}(v \approx v) \cap \\ \bigcap \left\{ D_{\mathsf{I}}^{ass}(v\alpha_1 \ldots \alpha_i \sim \mathit{metatop}) \,\middle|\, i \in \{0, \ldots, n\} \right\} \cap \\ D_{\mathsf{I}}^{ass}(v\alpha_1 \ldots \alpha_n \approx v') \end{array} \right)$$

$$\iff u \in D_{\mathsf{I}}^{ass} \left(\begin{array}{l} v \approx v \ \wedge \\ \bigwedge \left\{ v\alpha_1 \ldots \alpha_i \sim \mathit{metatop} \,\middle|\, i \in \{0, \ldots, n-1\} \right\} \wedge \\ v\alpha_1 \ldots \alpha_n \approx v' \wedge \ v\alpha_1 \ldots \alpha_n \sim \mathit{metatop} \end{array} \right)$$

$$\iff u \in D_{\mathsf{I}}^{ass} \left(\mathsf{ttrans} \left(v, \ v \begin{bmatrix} \mathit{metatop} \\ \alpha_1 \ \ldots \begin{bmatrix} \mathit{metatop} \\ \alpha_n \ \ v' \left[\mathit{metatop}\right] \end{bmatrix} \ldots \end{bmatrix} \right) \right)$$

$$\iff u \in D_{\mathsf{I}}^{ass} \left(\mathsf{trans} \left(\ v \begin{bmatrix} \mathit{metatop} \\ \alpha_1 \ \ldots \begin{bmatrix} \mathit{metatop} \\ \alpha_n \ \ v' \left[\mathit{metatop}\right] \end{bmatrix} \ldots \end{bmatrix} \right) \right),$$

thirdly, if $n = 0$ *and* $m > 0$,

analogous to the second case,

lastly, if $n > 0$ *and* $m > 0$,

$$\begin{array}{l} FV(v\alpha_1 \ldots \alpha_n \approx v'\alpha'_1 \ldots \alpha'_m) \\ = \{v, v'\} \\ = FV \left(\exists v'' \left[\begin{array}{l} \left[\begin{array}{l} v \approx v \ \wedge \\ \bigwedge \left\{ v\alpha_1 \ldots \alpha_i \sim \mathit{metatop} \,\middle|\, i \in \left\{ \begin{array}{c} 0, \ldots, \\ n-1 \end{array} \right\} \right\} \wedge \\ v\alpha_1 \ldots \alpha_n \approx v'' \ \wedge \ v\alpha_1 \ldots \alpha_n \sim \mathit{metatop} \end{array} \right] \\ \wedge \\ \left[\begin{array}{l} v' \approx v' \ \wedge \\ \bigwedge \left\{ v'\alpha'_1 \ldots \alpha'_j \sim \mathit{metatop} \,\middle|\, j \in \left\{ \begin{array}{c} 0, \ldots, \\ m-1 \end{array} \right\} \right\} \wedge \\ v'\alpha'_1 \ldots \alpha'_m \approx v'' \ \wedge \ v'\alpha'_1 \ldots \alpha'_m \sim \mathit{metatop} \end{array} \right] \end{array} \right] \right) \end{array}$$

$$= FV\left(\textsf{trans}\left(\exists v''\left(\begin{array}{l} v\begin{bmatrix} \textit{metatop} \\ \alpha_1 \ \ldots \begin{bmatrix}\textit{metatop} \\ \alpha_n \ \ v'' \left[\textit{metatop}\right]\end{bmatrix}\ldots\end{bmatrix} \\ \wedge \\ v'\begin{bmatrix} \textit{metatop} \\ \alpha'_1 \ \ldots \begin{bmatrix}\textit{metatop} \\ \alpha'_m \ \ v'' \left[\textit{metatop}\right]\end{bmatrix}\ldots\end{bmatrix}\end{array}\right)\right)\right), \textit{ and}$$

if, for each $x \in \mathcal{VAR}$, $ass(x) \in \overline{\textsf{Co}_{\textsf{I}}^{u}}$, *then*

$u \in D_{\textsf{I}}^{ass}(v\alpha_1 \ldots \alpha_n \approx v'\alpha'_1 \ldots \alpha'_m)$

$\Longleftrightarrow T_{\textsf{I}}^{ass}(v\alpha_1 \ldots \alpha_n)(u)$ *is defined,*

$T_{\textsf{I}}^{ass}(v'\alpha'_1 \ldots \alpha'_m)(u)$ *is defined,*

$T_{\textsf{I}}^{ass}(v\alpha_1 \ldots \alpha_n)(u) = T_{\textsf{I}}^{ass}(v'\alpha'_1 \ldots \alpha'_m)(u)$

$\Longleftrightarrow T_{\textsf{I}}^{ass}(v)(u)$ *is defined,*

for each $i \in \{1, \ldots, n\}$,

$T_{\textsf{I}}^{ass}(v\alpha_1 \ldots \alpha_i)(u)$ *is defined,*

$\widehat{\textsf{A}}(\alpha_i)(T_{\textsf{I}}^{ass}(v\alpha_1 \ldots \alpha_{i-1})(u))$ *is defined,*

$T_{\textsf{I}}^{ass}(v\alpha_1 \ldots \alpha_i)(u) = \widehat{\textsf{A}}(\alpha_i)(T_{\textsf{I}}^{ass}(v\alpha_1 \ldots \alpha_{i-1})(u))$,

for each $i \in \{0, \ldots, n\}$,

$\widehat{\textsf{S}}(T_{\textsf{I}}^{ass}(v\alpha_1 \ldots \alpha_i)(u)) \mathrel{\widehat{\sqsubseteq}} \textit{metatop}$,

$T_{\textsf{I}}^{ass}(v')(u)$ *is defined,*

for each $j \in \{1, \ldots, m\}$,

$T_{\textsf{I}}^{ass}(v'\alpha'_1 \ldots \alpha'_j)(u)$ *is defined,*

$\widehat{\textsf{A}}(\alpha'_j)(T_{\textsf{I}}^{ass}(v'\alpha'_1 \ldots \alpha'_{j-1})(u))$ *is defined,*

$T_{\textsf{I}}^{ass}(v'\alpha'_1 \ldots \alpha'_j)(u) = \widehat{\textsf{A}}(\alpha'_j)(T_{\textsf{I}}^{ass}(v'\alpha'_1 \ldots \alpha'_{j-1})(u))$,

for each $j \in \{0, \ldots, m\}$,

$\widehat{\textsf{S}}(T_{\textsf{I}}^{ass}(v'\alpha'_1 \ldots \alpha'_j)(u)) \mathrel{\widehat{\sqsubseteq}} \textit{metatop}$,

$T_{\textsf{I}}^{ass}(v\alpha_1 \ldots \alpha_n)(u) = T_{\textsf{I}}^{ass}(v'\alpha'_1 \ldots \alpha'_m)(u)$

$\iff T_{\mathsf{I}}^{ass}(v)(u)$ *is defined,*

for each $i \in \{1, \ldots, n\}$,

$\quad T_{\mathsf{I}}^{ass}(v\alpha_1 \ldots \alpha_i)(u)$ *is defined,*

for each $i \in \{0, \ldots, n\}$,

$\quad \widehat{\mathsf{S}}(T_{\mathsf{I}}^{ass}(v\alpha_1 \ldots \alpha_i)(u)) \mathrel{\widehat{\sqsubseteq}} metatop$,

$T_{\mathsf{I}}^{ass}(v')(u)$ *is defined,*

for each $j \in \{1, \ldots, m\}$,

$\quad T_{\mathsf{I}}^{ass}(v'\alpha'_1 \ldots \alpha'_j)(u)$ *is defined,*

for each $j \in \{0, \ldots, m\}$,

$\quad \widehat{\mathsf{S}}(T_{\mathsf{I}}^{ass}(v'\alpha'_1 \ldots \alpha'_j)(u)) \mathrel{\widehat{\sqsubseteq}} metatop$,

$T_{\mathsf{I}}^{ass}(v\alpha_1 \ldots \alpha_n)(u) = T_{\mathsf{I}}^{ass}(v'\alpha'_1 \ldots \alpha'_m)(u)$, *and*

for some $o_1 \in \overline{\mathsf{Co}_{\mathsf{I}}^{u}}$, *for some* $o_2 \in \overline{\mathsf{Co}_{\mathsf{I}}^{u}}$,

$\quad T_{\mathsf{I}}^{ass}(v)(u) = o_1$,

$\quad T_{\mathsf{I}}^{ass}(v')(u) = o_2$

$\iff$ *for some* $o \in \overline{\mathsf{Co}_{\mathsf{I}}^{u}}$,

$\quad T_{\mathsf{I}}^{ass\frac{o}{v''}}(v)(u)$ *is defined,* $T_{\mathsf{I}}^{ass\frac{o}{v''}}(v)(u) = T_{\mathsf{I}}^{ass\frac{o}{v''}}(v)(u)$,

$\quad$ *for each* $i \in \{0, \ldots, n\}$,

$\quad\quad T_{\mathsf{I}}^{ass\frac{o}{v''}}(v\alpha_1 \ldots \alpha_i)(u)$ *is defined,*

$\quad\quad \widehat{\mathsf{S}}(T_{\mathsf{I}}^{ass\frac{o}{v''}}(v\alpha_1 \ldots \alpha_i)(u)) \mathrel{\widehat{\sqsubseteq}} metatop$,

$\quad T_{\mathsf{I}}^{ass\frac{o}{v''}}(v\alpha_1 \ldots \alpha_n)(u)$ *is defined,*

$\quad T_{\mathsf{I}}^{ass\frac{o}{v''}}(v'')(u)$ *is defined, and*

$\quad T_{\mathsf{I}}^{ass\frac{o}{v''}}(v\alpha_1 \ldots \alpha_n)(u) = T_{\mathsf{I}}^{ass\frac{o}{v''}}(v'')(u)$,

$\quad T_{\mathsf{I}}^{ass\frac{o}{v''}}(v')(u)$ *is defined,* $T_{\mathsf{I}}^{ass\frac{o}{v''}}(v')(u) = T_{\mathsf{I}}^{ass\frac{o}{v''}}(v')(u)$,

$\quad$ *for each* $j \in \{0, \ldots, m\}$,

$$T_{\mathsf{I}}^{ass\frac{o}{v''}}(v'\alpha'_1 \ldots \alpha'_j)(u) \textit{ is defined,}$$
$$\widehat{\mathsf{S}}(T_{\mathsf{I}}^{ass\frac{o}{v''}}(v'\alpha'_1 \ldots \alpha'_j)(u)) \mathrel{\widehat{\sqsubseteq}} \textit{metatop},$$

$$T_{\mathsf{I}}^{ass\frac{o}{v''}}(v'\alpha'_1 \ldots \alpha'_m)(u) \textit{ is defined,}$$
$$T_{\mathsf{I}}^{ass\frac{o}{v''}}(v'')(u) \textit{ is defined, and}$$
$$T_{\mathsf{I}}^{ass\frac{o}{v''}}(v'\alpha'_1 \ldots \alpha'_m)(u) = T_{\mathsf{I}}^{ass\frac{o}{v''}}(v'')(u)$$

for some $o_1 \in \overline{\mathsf{Co}_{\mathsf{I}}^u}$, *for some* $o_2 \in \overline{\mathsf{Co}_{\mathsf{I}}^u}$,

$$T_{\mathsf{I}}^{ass\frac{o}{v''}}(v)(u) = o_1,$$
$$T_{\mathsf{I}}^{ass\frac{o}{v''}}(v')(u) = o_2$$

$\iff$ *for some* $o \in \overline{\mathsf{Co}_{\mathsf{I}}^u}$,

$$u \in D_{\mathsf{I}}^{ass\frac{o}{v''}}\left(\left[\begin{array}{l}\left[\begin{array}{l} v \approx v \;\wedge \\ \bigwedge\left\{ v\alpha_1 \ldots \alpha_i \sim \textit{metatop} \;\middle|\; i \in \left\{\begin{array}{c} 0, \ldots, \\ n-1 \end{array}\right\}\right\} \wedge \\ v\alpha_1 \ldots \alpha_n \approx v'' \;\wedge\; v\alpha_1 \ldots \alpha_n \sim \textit{metatop} \end{array}\right] \\ \wedge \\ \left[\begin{array}{l} v' \approx v' \;\wedge \\ \bigwedge\left\{ v'\alpha'_1 \ldots \alpha'_j \sim \textit{metatop} \;\middle|\; j \in \left\{\begin{array}{c} 0, \ldots, \\ m-1 \end{array}\right\}\right\} \wedge \\ v'\alpha'_1 \ldots \alpha'_m \approx v'' \;\wedge\; v'\alpha'_1 \ldots \alpha'_m \sim \textit{metatop} \end{array}\right]\end{array}\right]\right)$$

$$\iff u \in D_{\mathsf{I}}^{ass}\left(\exists v'' \left[\begin{array}{l}\left[\begin{array}{l} v \approx v \;\wedge \\ \bigwedge\left\{ v\alpha_1 \ldots \alpha_i \sim \textit{metatop} \;\middle|\; i \in \left\{\begin{array}{c} 0, \ldots, \\ n-1 \end{array}\right\}\right\} \wedge \\ v\alpha_1 \ldots \alpha_n \approx v'' \;\wedge\; v\alpha_1 \ldots \alpha_n \sim \textit{metatop} \end{array}\right] \\ \wedge \\ \left[\begin{array}{l} v' \approx v' \;\wedge \\ \bigwedge\left\{ v'\alpha'_1 \ldots \alpha'_j \sim \textit{metatop} \;\middle|\; j \in \left\{\begin{array}{c} 0, \ldots, \\ m-1 \end{array}\right\}\right\} \wedge \\ v'\alpha'_1 \ldots \alpha'_m \approx v'' \;\wedge\; v'\alpha'_1 \ldots \alpha'_m \sim \textit{metatop} \end{array}\right]\end{array}\right]\right)$$

$$\iff u \in D_{\mathsf{I}}^{ass}\left(\exists v'' \left[\begin{array}{l} \mathsf{ttrans}\left(v,\ v\begin{bmatrix} \mathit{metatop} \\ \alpha_1\ \ \ldots \begin{bmatrix} \mathit{metatop} \\ \alpha_n\ \ v''\left[\mathit{metatop}\right]\end{bmatrix} \ldots \end{bmatrix}\right) \\ \wedge \\ \mathsf{ttrans}\left(v',\ v'\begin{bmatrix} \mathit{metatop} \\ \alpha'_1\ \ \ldots \begin{bmatrix} \mathit{metatop} \\ \alpha'_m\ \ v''\left[\mathit{metatop}\right]\end{bmatrix} \ldots \end{bmatrix}\right)\end{array}\right]\right)$$

$$\iff u \in D_{\mathsf{I}}^{ass}\left(\mathsf{trans}\left(\exists v''\left(\begin{array}{l} v\begin{bmatrix} \mathit{metatop} \\ \alpha_1\ \ \ldots \begin{bmatrix} \mathit{metatop} \\ \alpha_n\ \ v''\left[\mathit{metatop}\right]\end{bmatrix} \ldots \end{bmatrix} \\ \wedge \\ v'\begin{bmatrix} \mathit{metatop} \\ \alpha'_1\ \ \ldots \begin{bmatrix} \mathit{metatop} \\ \alpha'_m\ \ v''\left[\mathit{metatop}\right]\end{bmatrix} \ldots \end{bmatrix}\end{array}\right)\right)\right),$$

for each $\rho \in \mathcal{R}$, *for each* $v_1 \in \mathcal{VAR}$, $\ldots$, *for each* $v_{\mathcal{AR}(\rho)} \in \mathcal{VAR}$,

$$\begin{aligned} & FV(\rho(v_1, \ldots, v_{\mathcal{AR}(\rho)})) \\ & = \{v_1, \ldots, v_{\mathcal{AR}(\rho)}\} \\ & = FV(\mathsf{trans}(\rho(v_1, \ldots, v_{\mathcal{AR}(\rho)}))), \text{ and} \end{aligned}$$

$$\begin{aligned} & u \in D_{\mathsf{I}}^{ass}(\rho(v_1, \ldots, v_{\mathcal{AR}(\rho)})) \\ & \iff u \in D_{\mathsf{I}}^{ass}(\mathsf{trans}(\rho(v_1, \ldots, v_{\mathcal{AR}(\rho)}))). \end{aligned}$$

Thus, for each signature Σ, *for each* $\delta \in \mathcal{D}^{\Sigma}$, *there exists a* $\kappa \in \mathbb{AVM}^{\Sigma}$ *such that, for each interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R}\rangle$, *for each* $u \in \mathsf{U}$, *for each* $ass \in Ass_{\mathsf{I}}$,

$FV(\delta) = FV(\mathsf{trans}(\kappa))$, *and*

if for each $v \in \mathcal{VAR}$, $ass(v) \in \overline{\mathsf{Co}_{\mathsf{I}}^{u}}$,

then $u \in D_{\mathsf{I}}^{ass}(\delta)$ *iff* $u \in D_{\mathsf{I}}^{ass}(\mathsf{trans}(\kappa))$.

by induction on the complexity of δ

■

Theorem 4 *For each signature* Σ, *for each* $\delta \in \mathcal{D}_0^{\Sigma}$, *there exists a* $\kappa \in \mathbb{AVM}_0^{\Sigma}$ *such that, for each interpretation* $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R}\rangle$, *for each* $u \in \mathsf{U}$,

$u \in D_{\mathsf{I}}(\delta)$ *iff* $u \in D_{\mathsf{I}}(\mathsf{trans}(\kappa))$.

Proof
Follows immediately from PROPOSITION *22.* ■

Appendix C

The Principles of Pollard & Sag 94

In this appendix, I formalize those grammatical principles of the appendix of Pollard and Sag 1994 that I have not discussed in Chapter 4. With the exception of the ID PRINCIPLE, my presentation follows the order in which the principles are presented in their appendix, omitting the ones that are discussed in detail in Chapter 4. Since Pollard and Sag 1994 presents the ID schemata in a separate subsection after all principles of the grammar have been stated,[1] I have moved the entire ID PRINCIPLE to the position of the ID schemata in order to make its presentation more coherent.

The formalizations of the principles given in this section are hardly ever interesting by themselves, and the techniques that I apply have all been discussed before. The comments will, therefore, be kept brief. The purpose of the presentation of all principles is to be precise about how every principle of the grammar of Pollard and Sag 1994 can be captured in RSRL. It plays an important role in substantiating my claim that RSRL provides everything to make an HPSG 94 grammar faithfully explicit.

(161) *The* MARKING PRINCIPLE *(Pollard and Sag, 1994, p. 400)*

> In a headed phrase, the MARKING value is token-identical with that of the MARKER-DAUGHTER if any, and with that of the HEAD-DAUGHTER otherwise.

[1]Except for the RAISING PRINCIPLE, to which I will get back at the end of this section.

According to the signature, the attribute MARKER-DTR is only appropriate for *head-marker-struc*. The second case of the MARKING PRINCIPLE can therefore be expressed by demanding that in headed phrases whose DTRS value is not of sort *head-mark-struc*, the MARKING value of the phrase and of the HEAD-DTR are identical:

(162) *The* MARKING PRINCIPLE *formalized*

$$\begin{array}{l}\left[\text{DTRS}\ \textit{headed-struc}\right] \longrightarrow \\ \left(\begin{array}{l}\left[\begin{array}{l}\text{SS LOC CAT MARKING}\ \boxed{1} \\ \text{DTRS MARKER-DTR SS LOC CAT MARKING}\ \boxed{1}\end{array}\right] \vee \\ \left(\left[\begin{array}{l}\text{SS LOC CAT MARKING}\ \boxed{1} \\ \text{DTRS HEAD-DTR SS LOC CAT MARKING}\ \boxed{1}\end{array}\right] \wedge \neg\left[\text{DTRS}\ \textit{head-mark-struc}\right]\right)\end{array}\right)\end{array}$$

Although the informal characterization of the SPEC PRINCIPLE seems as simple as the MARKING PRINCIPLE, its complex antecedent makes it more interesting:

(163) *The* SPEC PRINCIPLE *(Pollard and Sag, 1994, p. 400)*

> In a headed phrase whose nonhead daughter (either the MARKER-DAUGHTER or COMPLEMENT-DAUGHTERS | FIRST) has a SYNSEM | LOCAL | CATEGORY | HEAD value of sort *functional*, the SPEC value of that value must be token-identical with the phrase's DAUGHTERS | HEAD-DAUGHTER | SYNSEM value.

The distinction in the antecedent between headed phrases with a marker daughter and headed phrases with a first complement daughter, where in each case the respective daughter must be functional, makes a formalization in terms of universal quantification over a disjunction of the appropriate nonhead daughters and their SPEC value in the antecedent attractive. The obvious (if boring) alternative solution would be to formulate the SPEC PRINCIPLE as two implications, one for phrases with a functional marker daughter and one for phrases whose first complement daughter is functional.

(164) *The* Spec Principle *formalized*

$$\forall \boxed{1}\ \forall \boxed{2} \left(\left(\left[\text{DTRS } \left[\left[\text{MARKER-DTR } \boxed{1}\right] \lor \left[\text{COMP-DTRS } \langle \boxed{1} |\, \textit{list} \rangle\right] \right]\right] \land \boxed{1}\left[\text{SS LOC CAT HEAD } \begin{bmatrix} \textit{functional} \\ \text{SPEC } \boxed{2} \end{bmatrix}\right] \right) \rightarrow \left[\text{DTRS HEAD-DTR SS } \boxed{2}\right] \right)$$

The Trace Principle, like the Control Theory and the Binding Theory, is one of the prime examples of quantification over components:

(165) *The* Trace Principle *(Pollard and Sag, 1994, p. 400)*

> The SYNSEM value of any trace must be a (noninitial) member of the SUBCAT list of a substantive word.

At first glance, the Trace Principle seems to talk about traces, or, in other words, it might seem to be a condition on traces. By definition, traces are entities in the denotation of the following description (Pollard and Sag, 1994, p. 164):

(166)
$$\begin{bmatrix} \textit{sign} & \\ \text{PHON} & \langle\rangle \\ \text{SS} & \begin{bmatrix} \text{LOC} & \boxed{1} \\ \text{NONLOC} & \begin{bmatrix} \text{INHERITED} & \begin{bmatrix} \text{QUE} & \{\} \\ \text{REL} & \{\} \\ \text{SLASH} & \{\boxed{1}\} \end{bmatrix} \\ \text{TO-BIND} & \begin{bmatrix} \text{QUE} & \{\} \\ \text{REL} & \{\} \\ \text{SLASH} & \{\} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

When one considers the fact that the Trace Principle demands that there be a substantive word on whose SUBCAT list the SYNSEM value of a trace occurs, it becomes clear that it does not express a condition on traces in the same sense in which the Head Feature Principle expresses a condition on headed phrases. There is no component of a trace that could be the substantive word that is asked for. Neither is it reasonable to assume that the identity of a trace's SYNSEM value with an element of the SUBCAT list of any otherwise unrelated substantive word somewhere in the linguistic universe is what the TRACE PRINCIPLE requires. What is meant is rather that a substantive word that occurs in the same utterance—or some other suitably specified linguistic context—with the trace has the trace's SYNSEM value as a non-initial member on its SUBCAT list. Whatever the linguistic context is that the authors of the TRACE PRINCIPLE ultimately envision, that context constitutes an entity of which both the trace and the substantive word are components.

Since there is no notion of utterance or unembedded sign in the grammar of Pollard and Sag 1994, I pick finite, saturated sentences in which no unbounded dependencies are open as a first practical approximation to the unembedded signs in which the TRACE PRINCIPLE must be satisfied. The sentences that I have just characterized can be described as follows:

$$(167)\quad \begin{bmatrix} \text{PHON} & \textit{nelist} \\ \text{SS} & \begin{bmatrix} \text{LOC CAT} & \begin{bmatrix} \text{HEAD VFORM} & \textit{finite} \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \\ \text{NONLOC INHERITED} & \begin{bmatrix} \text{QUE} & \{\} \\ \text{REL} & \{\} \\ \text{SLASH} & \{\} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Within the scope of the finite sentences of (167), it is straightforward to formalize the TRACE PRINCIPLE: Within each finite sentence, s, in the sense of (167), and for each component trace of s, there exists a substantive word that is a component of s such that the trace's SYNSEM value is a non-initial member of the substantive word's SUBCAT list:

(168) *The* TRACE PRINCIPLE *formalized*

$$
\forall x\, \forall \boxed{2}
\left(\begin{array}{l}
\exists \boxed{1} \\
\left(\begin{array}{l} \begin{bmatrix} \text{PHON} & \textit{nelist} \\ \text{SS} & \begin{bmatrix} \text{LOC CAT} & \begin{bmatrix}\text{HEAD VFORM} & \textit{finite} \\ \text{SUBCAT} & \langle\rangle\end{bmatrix} \\ \text{NONLOC INHERITED} & \begin{bmatrix} \text{QUE} & \{\} \\ \text{REL} & \{\} \\ \text{SLASH} & \{\}\end{bmatrix}\end{bmatrix}\end{bmatrix} \\ \land \ {}^{x}\begin{bmatrix} \textit{sign} & \\ \text{PHON} & \langle\rangle \\ \text{SS} & \boxed{2}\begin{bmatrix} \text{LOC} & \boxed{1} \\ \text{NONLOC} & \begin{bmatrix} \text{INHERITED} & \begin{bmatrix}\text{QUE} & \{\} \\ \text{REL} & \{\} \\ \text{SLASH} & \{\boxed{1}\}\end{bmatrix} \\ \text{TO-BIND} & \begin{bmatrix}\text{QUE} & \{\}\\ \text{REL} & \{\} \\ \text{SLASH} & \{\}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix} \end{array} \right) \rightarrow \\
\exists y\, \exists \boxed{3} \\
\left({}^{y}\begin{bmatrix} \textit{word} & \\ \text{SS LOC CAT} & \begin{bmatrix} \text{HEAD} & \textit{substantive} \\ \text{SUBCAT} & \langle \textit{object} | \boxed{3} \rangle \end{bmatrix}\end{bmatrix} \land \texttt{member}(\boxed{2}, \boxed{3}) \right)
\end{array} \right)
$$

The SUBJECT CONDITION is a simple example of the need for quantification in HPSG 94:

(169) *The* SUBJECT CONDITION *(Pollard and Sag, 1994, p. 400)*

> If the initial element of the SUBCAT value of a word is slashed, then so is some other element of that list.

(170) *The* SUBJECT CONDITION *formalized*

$$
\forall \boxed{1} \left(\begin{array}{l} \begin{bmatrix} \textit{word} \\ \text{SS LOC CAT SUBCAT} \ \left\langle [\text{NONLOC INHERITED SLASH } \textit{neset}] | \boxed{1} \right\rangle \end{bmatrix} \rightarrow \\ \exists x\, \texttt{member}\left({}^{x}[\text{NONLOC INHERITED SLASH } \textit{neset}], \boxed{1} \right) \end{array} \right)
$$

In the appendix of Pollard and Sag 1994, the SUBJECT CONDITION is followed by the WEAK COORDINATION PRINCIPLE. It says that, in a coordinate structure, the CATEGORY and NONLOCAL values of each conjunct daughter are subsumed by those of the mother. As the authors themselves note, the notion of subsumption that they evoke in this principle is not consistent with the assumptions about the formalism that they make throughout the rest of their theorizing. With the notion of subsumption, the principle refers to the subsumption of partial feature structures of HPSG 87, as is made clear on page 203 of Pollard and Sag 1994. For that single principle, Pollard and Sag would have to give up the epistemological hypothesis that the maximally specific sorts of the sort hierarchy partition the universe of linguistic entities in favor of admitting entities of nonmaximal sorts, and they would have to give up the idea of *total* well-typedness. They concede that the WEAK COORDINATION PRINCIPLE, thus, lacks a formal interpretation in their grammar. Since RSRL shares the assumptions about the interpretation of the sort hierarchy and total well-typedness with HPSG 94, the WEAK COORDINATION PRINCIPLE lies beyond its expressive boundaries. Since the WEAK COORDINATION PRINCIPLE presupposes a different epistemology, I consider it justified to ignore it here.

The SINGLETON REL CONSTRAINT returns to the strictures of the formalism:

(171) *The* SINGLETON REL CONSTRAINT *(Pollard and Sag, 1994, p. 400)*

For any sign, the SYNSEM | NONLOCAL | INHERITED | REL value is a set of cardinality at most one.

(172) *The* SINGLETON REL CONSTRAINT *formalized*

$$\begin{bmatrix} sign \end{bmatrix} \rightarrow$$
$$\begin{bmatrix}\text{SS NONLOC INHERITED REL } \begin{bmatrix}\{\} \vee \{ object \}\end{bmatrix}\end{bmatrix}$$

The RELATIVE UNIQUENESS PRINCIPLE belongs to those principles that require a few new relations to make them explicit. This is remarkable insofar as its informal rendering is very simple:

(173) *The* RELATIVE UNIQUENESS PRINCIPLE, RUP *(Pollard and Sag, 1994, p. 400)*

> For any phrase, a member of the set value of SYNSEM | NONLOCAL | INHERITED | REL may belong to the value of that same path on at most one daughter.

Note that, unlike most of the conditions on phrases that we have seen so far, the antecedent of the RUP is not only true of headed phrases, but also of coordinate phrases. However, this is not the case in the first formulation of the RUP on page 212 of Pollard and Sag 1994, where the antecedent of the principle is each "headed constituent". The solution to this puzzle can be found in footnote 6, page 397, which says that the fragment of English of the book is to be thought of as limited to headed phrases, except for the "roughly formulated section 6.1 of Chapter 4" (which is the section on coordinate structures in unbounded dependency constructions that gives rise to the WEAK COORDINATION PRINCIPLE). I take this remark to imply that the RUP is in fact meant to apply only to headed phrases, because other phrases are not treated in the formalized fragment of English. Since the sort *coord-struc* is, however, part of the sort hierarchy of the appendix, its presence must be honored in the formulation of principles with the given signature. I conclude that a formalization of the RUP that talks about headed phrases in its antecedent comes closest to the intentions of the authors of (173), and I will go that route.

In the formalization of the RUP, I need two relations: One that expresses that each member of a component set occurs at most once in a collection of component sets; and one that gives me convenient access to the collection of sets in which I am interested, namely to the INHERITED REL values of all daughters of a phrase. As usual, this idea can be expressed by relating each headed structure to a tape of all INHERITED REL sets of its daughters. I define the first relation first:

(174) $\texttt{at-most-once}(x, y) \overset{\forall}{\Longleftarrow}$
$y \ll\gg$

$\texttt{at-most-once}(x, y) \overset{\forall}{\Longleftarrow}$
$y \ll a|\, z \gg \land \neg\texttt{member}(x, a) \land \texttt{at-most-once}(x, z)$

$\texttt{at-most-once}(x, y) \overset{\forall}{\Longleftarrow}$
$y \ll a|\, z \gg \land \texttt{member}(x, a) \land \texttt{successive-not-member}(x, z)$

$$\texttt{successive-not-member}(x, y) \stackrel{\forall}{\Longleftarrow}$$
$$y \ll\gg$$

$$\texttt{successive-not-member}(x, y) \stackrel{\forall}{\Longleftarrow}$$
$$y \ll a|\, z \gg \wedge \neg\texttt{member}(x, a) \wedge \texttt{successive-not-member}(x, z)$$

x occurs at most once in the sets that are on the tape, y,[2] because either (clause 1) y is the empty tape, or (clause 2) x is not a member of the first set on the tape and it occurs at most once in the sets in the rest of the tape, or (clause 3) x is a member of the first set on the tape, but it is not a member of any of the sets on the rest of the tape. The latter condition is expressed by `successive-not-member`, which holds between a component, x, and a tape, y, iff x is not a member of any of the sets on y.

As mentioned above, the second relation that the RUP requires provides a tape of the INHERITED REL sets of all daughters for each entity in the denotation of the sort *headed-struc*. I call it `all-dtrs-rels`. It relates all entities in the denotation of *headed-struc* to a tape of the INHERITED REL values of their daughters. For the headed structures with exactly two daughters, the definition is trivial; for head complement structures, the relation `enumerate-rels`, (98), is invoked that was originally defined in the formalization of the NONLOCAL FEATURE PRINCIPLE, (96). For a component list of signs in the first argument, `enumerate-rels` contains a tape with the INHERITED REL values of these signs in the second argument.

(175)
$$\texttt{all-dtrs-rels}(x, y) \stackrel{\forall}{\Longleftarrow}$$
$$\begin{array}{l} {}^{x}\begin{bmatrix} \textit{head-comp-struc} \\ \text{HEAD-DTR SS NONLOC INHERITED REL } \boxed{1} \\ \text{COMP-DTRS } \boxed{2} \end{bmatrix} \\ \wedge\ \texttt{enumerate-rels}(\boxed{2}, \boxed{3}) \wedge\ y \ll \boxed{1} |\, \boxed{3} \gg \end{array}$$

$$\texttt{all-dtrs-rels}(x, y) \stackrel{\forall}{\Longleftarrow}$$
$$\begin{array}{l} {}^{x}\begin{bmatrix} \textit{head-adj-struc} \\ \text{HEAD-DTR SS NONLOC INHERITED REL } \boxed{1} \\ \text{ADJUNCT-DTR SS NONLOC INHERITED REL } \boxed{2} \end{bmatrix} \\ \wedge\ y \ll \boxed{1}, \boxed{2} \gg \end{array}$$

[2]The definition in (174) of course also allows lists as elements of a nonempty tape, y. I restrict my attention to sets, since in the formalization of the RUP, only tapes of sets play a role.

$$\texttt{all-dtrs-rels}(x, y) \stackrel{\forall}{\Longleftarrow} {}^{x}\begin{bmatrix} \textit{head-filler-struc} \\ \text{HEAD-DTR SS NONLOC INHERITED REL} \; \boxed{1} \\ \text{FILLER-DTR SS NONLOC INHERITED REL} \; \boxed{2} \end{bmatrix} \wedge\, y \ll \boxed{1}, \boxed{2} \gg$$

$$\texttt{all-dtrs-rels}(x, y) \stackrel{\forall}{\Longleftarrow} {}^{x}\begin{bmatrix} \textit{head-mark-struc} \\ \text{HEAD-DTR SS NONLOC INHERITED REL} \; \boxed{1} \\ \text{MARKER-DTR SS NONLOC INHERITED REL} \; \boxed{2} \end{bmatrix} \wedge\, y \ll \boxed{1}, \boxed{2} \gg$$

With the necessary relations defined, I can now express the RUP. For every headed phrase, for each member, u, of its INHERITED REL set, u is a member of at most one of the INHERITED REL sets of the phrase's daughters:

(176) *The* RELATIVE UNIQUENESS PRINCIPLE *formalized*

$$\forall \boxed{1}\; \forall \boxed{3} \left(\begin{array}{l} \exists \boxed{2} \left(\begin{bmatrix} \textit{phrase} \\ \text{SS NONLOC INHERITED REL} \; \boxed{2} \\ \text{DTRS} \; \boxed{3} \; \textit{headed-struc} \end{bmatrix} \wedge \texttt{member}(\boxed{1}, \boxed{2}) \right) \rightarrow \\ \exists \boxed{4} \left(\texttt{all-dtrs-rels}(\boxed{3}, \boxed{4}) \wedge \texttt{at-most-once}(\boxed{1}, \boxed{4}) \right) \end{array} \right)$$

The CLAUSAL REL PROHIBITION is a very simple principle that does not require any relations at all. It differs from the majority of the principles of Pollard and Sag 1994 by describing *synsem* entities in its antecedent.

(177) *The* CLAUSAL REL PROHIBITION *(Pollard and Sag, 1994, p. 401)*

For any *synsem* object, if the LOCAL | CATEGORY | HEAD value is *verb* and the LOCAL | CATEGORY | SUBCAT value is $\langle\rangle$, then the NONLOCAL | INHERITED | REL value must be $\{\}$.

(178) *The* CLAUSAL REL PROHIBITION *formalized*

$$\begin{bmatrix} \text{LOC CAT} \begin{bmatrix} \text{HEAD} & \textit{verb} \\ \text{SUBCAT} & \langle\rangle \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \text{NONLOC INHERITED REL} \; \{\} \end{bmatrix}$$

The informal characterization of the clause (b) of the SEMANTICS PRINCIPLE (SP-b) sounds complex, because it distinguishes phrases with a *psoa*-valued CONTENT from phrases with other CONTENT values:

(179) *The* SEMANTICS PRINCIPLE, *clause (b), (Pollard and Sag, 1994, p. 402)*

> If the semantic head's SYNSEM | LOCAL | CONTENT value is of sort *psoa*, then the SYNSEM | LOCAL | CONTENT | NUCLEUS value is token-identical with that of the semantic head, and the SYNSEM | LOCAL | CONTENT | QUANTS value is the concatenation of the RETRIEVED value and the semantic head's SYNSEM | LOCAL | CONTENT | QUANTS value; otherwise the RETRIEVED value is the empty list, and the SYNSEM | LOCAL | CONTENT value is token-identical with that of the semantic head.

In footnote 16, page 402, Pollard and Sag 1994 explains what the semantic head of a phrase is: "In a headed phrase, the *semantic head* is the ADJUNCT-DAUGHTER if any and the HEAD-DAUGHTER otherwise." Hence not only must I distinguish phrases with a *psoa*-valued CONTENT from other phrases, in each case there is also a further distinction between head adjunct phrases and all others.[3]

In (180), I split the SP-b into two implications, where the first is concerned with phrases with CONTENT values of sort *psoa*, and the second with phrases that do not have a *psoa*-valued CONTENT. In the both cases, a universal quantification using tag $\boxed{1}$ over one more implication then picks out the sign that is the semantic head of the phrase, and the condition of the SP-b is formulated with reference to the semantic head, $\boxed{1}$: In the first case, the RETRIEVED list of the phrase and the QUANTS list of its semantic head stand in the `append` relation with the QUANTS list of the phrase, and the NUCLEUS value of the phrase equals the NUCLEUS value of the semantic head. In the second case, the RETRIEVED list of the phrase is empty, and its CONTENT value equals the CONTENT value of the semantic head.

[3] Bearing in mind the comments regarding coordinate structures on page 425, I exclude coordinate phrases from the requirements of the SP-b. In (180), their exclusion follows from the distinction between head adjunct phrases and phrases that are not head adjunct phrases but have a head daughter. Coordinate phrases are neither head adjunct phrases nor do they have a head daughter.

(180) *The* Semantics Principle, *clause (b), formalized*

$$
\left(\begin{array}{l}
\begin{bmatrix} \textit{phrase} \\ \textsc{ss loc cont} \;\; \textit{psoa} \end{bmatrix} \rightarrow \\
\forall \boxed{1} \\
\left(\begin{array}{l}
\left[\textsc{dtrs} \;\; \left[\begin{bmatrix}\textit{head-adj-struc}\\ \textsc{adjunct-dtr} \;\; \boxed{1}\end{bmatrix} \vee \left[\neg[\textit{head-adj-struc}] \wedge [\textsc{head-dtr} \;\; \boxed{1}]\right]\right]\right] \rightarrow \\
\exists\boxed{2}\ \exists\boxed{3}\ \exists\boxed{4}\ \exists\boxed{5} \\
\left(\begin{array}{l}
\begin{bmatrix} \textsc{ss loc cont} & \begin{bmatrix}\textsc{quants} & \boxed{2}\\ \textsc{nucleus} & \boxed{3}\end{bmatrix} \\ \textsc{retrieved} & \boxed{4}\end{bmatrix} \wedge \boxed{1}\left[\textsc{ss loc cont} \;\; \begin{bmatrix}\textsc{quants} & \boxed{5}\\ \textsc{nucleus} & \boxed{3}\end{bmatrix}\right] \\
\wedge\ \texttt{append}(\boxed{4},\boxed{5},\boxed{2})
\end{array}\right)
\end{array}\right)
\end{array}\right)
$$

$\wedge$

$$
\left(\begin{array}{l}
\begin{bmatrix} \textit{phrase} \\ \textsc{ss loc cont} \;\; \neg\textit{psoa} \end{bmatrix} \rightarrow \\
\forall \boxed{1} \\
\left(\begin{array}{l}
\left[\textsc{dtrs} \;\; \left[\begin{bmatrix}\textit{head-adj-struc}\\ \textsc{adjunct-dtr} \;\; \boxed{1}\end{bmatrix} \vee \left[\neg[\textit{head-adj-struc}] \wedge [\textsc{head-dtr} \;\; \boxed{1}]\right]\right]\right] \rightarrow \\
\exists\boxed{2} \\
\left(\begin{bmatrix}\textsc{ss loc cont} & \boxed{2} \\ \textsc{retrieved} & \langle\rangle\end{bmatrix} \wedge \boxed{1}\left[\textsc{ss loc cont} \;\; \boxed{2}\right]\right)
\end{array}\right)
\end{array}\right)
$$

Note that the meaning of (180) would not change if the antecedents of the two conjuncts were not restricted to phrases, i.e., if the antecedents described signs. For readability, I prefer a formulation where it is made explicit in the beginning that it is only phrases that are restricted by the principle.

The Principle of Contextual Consistency (PCC) is similar to the Nonlocal Feature Principle: A set value of the mother is characterized as the (successive) union of the corresponding set values of the daughters. The PCC is simpler than the NFP, because it does not demand the subtraction of another set from the union of sets:

(181) *The* Principle of Contextual Consistency *(Pollard and Sag, 1994, p. 402)*

> The context | background value of a given phrase is the union of the context | background values of the daughters.

Although the remarks on the exclusion of coordinate phrases that I have cited above apply to all principles of the appendix, I have included coordinate phrases in my formalization of the PCC below in order to illustrate what a complex principle over sets that are embedded in other sets looks like. The preliminary nature of the PCC (Pollard and Sag, 1994, pp. 333–334) also justifies a broader coverage of the initial hypothesis, which must be revised in any case in a refined theory of presupposition inheritance.

For the PCC I define two relations whose basic idea is known from the NFP. The first one is called `collect-dtrs-backgrounds`, and it relates each entity in the denotation of the sort *con-struc* to the set (or chain) that is the (successive) union of the BACKGROUND sets of all of its daughters (provided that there are only finitely many daughters and the list of complement daughters, if there is any, is acyclic). Second, for the clause in the definition of `collect-dtrs-backgrounds` that treats head complement structures, I define a relation, `enumerate-backgrounds`, that relates a list of signs to a tape of the BACKGROUND values of each sign. It should be compared to the analogous relations `enumerate-slashes`, `enumerate-ques`, and `enumerate-rels` in (98), and to `enumerate-qstores` in (101).

(182) $\texttt{collect-dtrs-backgrounds}(x, y) \stackrel{\forall}{\Longleftarrow}$

$$^{x}\begin{bmatrix} \textit{head-comp-struc} \\ \text{HEAD-DTR SS LOC CONTEXT BACKGROUND } \boxed{1} \\ \text{COMP-DTRS } \boxed{2} \end{bmatrix}$$
$$\wedge\ \texttt{enumerate-backgrounds}(\boxed{2}, \boxed{3}) \wedge\ ^{y}\!\left[\bigcup z \ll \boxed{1} \mid \boxed{3} \gg\right]$$

$\texttt{collect-dtrs-backgrounds}(x, y) \stackrel{\forall}{\Longleftarrow}$

$$^{x}\begin{bmatrix} \textit{head-filler-struc} \\ \text{HEAD-DTR SS LOC CONTEXT BACKGROUND } \boxed{1} \\ \text{FILLER-DTR SS LOC CONTEXT BACKGROUND } \boxed{2} \end{bmatrix} \wedge\ ^{y}\!\left[\boxed{1} \cup \boxed{2}\right]$$

$\texttt{collect-dtrs-backgrounds}(x, y) \stackrel{\forall}{\Longleftarrow}$

$$^{x}\begin{bmatrix} \textit{head-mark-struc} \\ \text{HEAD-DTR SS LOC CONTEXT BACKGROUND } \boxed{1} \\ \text{MARKER-DTR SS LOC CONTEXT BACKGROUND } \boxed{2} \end{bmatrix} \wedge\ ^{y}\!\left[\boxed{1} \cup \boxed{2}\right]$$

$\texttt{collect-dtrs-backgrounds}(x, y) \stackrel{\forall}{\Longleftarrow}$

$$^{x}\begin{bmatrix} \textit{head-adj-struc} \\ \text{HEAD-DTR SS LOC CONTEXT BACKGROUND } \boxed{1} \\ \text{ADJUNCT-DTR SS LOC CONTEXT BACKGROUND } \boxed{2} \end{bmatrix} \wedge\ ^{y}\!\left[\boxed{1} \cup \boxed{2}\right]$$

$$
\begin{array}{l}
\texttt{collect-dtrs-backgrounds}(x, y) \overset{\forall}{\Longleftarrow} \\
\quad {}^{x}\!\left[\begin{array}{l} \textit{coord-struc} \\ \textsc{conjunction-dtr ss loc context background } \boxed{1} \\ \textsc{conj-dtrs } \boxed{2} \end{array}\right] \\
\quad \wedge\ \exists \boxed{5} \\
\quad \left(\begin{array}{l}
\forall \boxed{3}\ \forall \boxed{4} \\
\left(\begin{array}{l} \texttt{member}\Big(\boxed{3}\big[\textsc{ss loc context background } \boxed{4}\big], \boxed{2}\Big) \rightarrow \\ \texttt{member}\Big(\boxed{4}, \boxed{5}\big[\textit{chain}\big]\Big) \end{array} \right) \\
\wedge\ \texttt{member}\big(\boxed{1}, \boxed{5}\big) \\
\wedge\ \forall \boxed{6} \\
\left(\begin{array}{l} \texttt{member}\big(\boxed{6}, \boxed{5}\big) \rightarrow \\
\left(\begin{array}{l} \exists \boxed{7}\ \exists \boxed{8} \\
\left(\begin{array}{l} \texttt{member}\Big(\boxed{7}\big[\textsc{ss loc context background } \boxed{8}\big], \boxed{2}\Big) \\ \wedge\ \boxed{8} = \boxed{6} \end{array} \right) \\
\vee\ \boxed{6} = \boxed{1} \end{array} \right) \end{array} \right) \\
\wedge\ {}^{y}\big[\textstyle\bigcup \boxed{5}\big]
\end{array} \right)
\end{array}
$$

(183) $\texttt{enumerate-backgrounds}(x, y) \overset{\forall}{\Longleftarrow}$

$$x\, \langle\rangle \wedge y \ll\gg$$

$\texttt{enumerate-backgrounds}(x, y) \overset{\forall}{\Longleftarrow}$

$$
\begin{array}{l}
x \left\langle \big[\textsc{ss loc context background } \boxed{1}\big] \middle| \boxed{2} \right\rangle \wedge y \ll \boxed{1} | \boxed{3} \gg \\
\wedge\ \texttt{enumerate-backgrounds}\big(\boxed{2}, \boxed{3}\big)
\end{array}
$$

The clause for coordinate structures of the definition of `collect-dtrs-backgrounds` is a good example of how expressing a relationship in a configuration of entities can be greatly simplified by introducing chains in a relation. Assume we allow chains in the first argument of `enumerate-backgrounds` in addition to lists. Then the unwieldy second conjunct of the clause can be reduced to

$$
\begin{array}{l}
\boxed{2} \subseteq \boxed{3}\,[\textit{chain}] \ \wedge\ \boxed{3} \subseteq \boxed{2} \ \wedge\ \texttt{enumerate-backgrounds}\big(\boxed{3}, \boxed{4}\big) \\
\wedge\ {}^{y}\big[\textstyle\bigcup z \ll \boxed{1} | \boxed{4} \gg\big].
\end{array}
$$

(184) *The* PRINCIPLE OF CONTEXTUAL CONSISTENCY *formalized*

$$[\textit{phrase}] \rightarrow$$

$$\begin{bmatrix}\text{SS LOC CONTEXT BACKGROUND } \boxed{1} \\ \text{DTRS } \boxed{2}\end{bmatrix}$$
$$\wedge\ \texttt{collect-dtrs-backgrounds}(\boxed{2}, \boxed{1})$$

Finally, we obtain a complete informal version of the ID PRINCIPLE by combining it with the description of the six ID schemata for English:

(185) *The* ID PRINCIPLE *(Pollard and Sag, 1994, p. 399 and p. 402–403)*

Every headed phrase must satisfy exactly one of the ID schemata.

SCHEMA 1 (HEAD-SUBJECT SCHEMA)

The SYNSEM | LOCAL | CATEGORY | SUBCAT value is $\langle\rangle$, and the DAUGHTERS value is an object of sort *head-comp-struc* whose HEAD-DAUGHTER is a phrase whose SYNSEM | NONLOCAL | TO-BIND | SLASH value is $\{\}$, and whose COMPLEMENT-DAUGHTERS value is a list of length one.

SCHEMA 2 (HEAD-COMPLEMENT SCHEMA)

The SYNSEM | LOCAL | CATEGORY | SUBCAT value is a list of length one, and the daughters value is an object of sort *head-comp-struc* whose HEAD-DAUGHTER value is a word.

SCHEMA 3 (HEAD-SUBJECT-COMPLEMENT SCHEMA)

The SYNSEM | LOCAL | CATEGORY | SUBCAT value is $\langle\rangle$, and the DAUGHTERS value is an object of sort *head-comp-struc* whose HEAD-DAUGHTER value is a word.

SCHEMA 4 (HEAD-MARKER SCHEMA)

The DAUGHTERS value is an object of sort *head-marker-struc* whose HEAD-DAUGHTER | SYNSEM | NONLOCAL | TO-BIND | SLASH value is $\{\}$, and whose MARKER-DAUGHTER | SYNSEM | LOCAL | CATEGORY | HEAD value is of sort *marker*.

SCHEMA 5 (HEAD-ADJUNCT SCHEMA)

The DAUGHTERS value is an object of sort *head-adjunct-struc* whose HEAD-DAUGHTER | SYNSEM value is token-identical to its

ADJUNCT-DAUGHTER | SYNSEM | LOCAL | CATEGORY | HEAD | MOD value and whose HEAD-DAUGHTER | SYNSEM | NONLOCAL | TO-BIND | SLASH value is {}.

SCHEMA 6 (HEAD-FILLER SCHEMA)

The DAUGHTERS value is an object of sort *head-filler-struc* whose HEAD-DAUGHTER | SYNSEM | LOCAL | CATEGORY value satisfies the description [HEAD *verb*[VFORM *finite*], SUBCAT ⟨⟩], whose HEAD-DAUGHTER | SYNSEM | NONLOCAL | INHERITED | SLASH value contains an element token-identical to the FILLER-DAUGHTER | SYNSEM | LOCAL value, and whose HEAD-DAUGHTER | SYNSEM | NONLOCAL | TO-BIND | SLASH value contains only that element.

Pollard and Sag 1994, footnote 17, p. 402, mentions additional restrictions in the ID schemata for English: "In the parochial versions of Schemata 1 and 2, the SYNSEM | LOCAL | CATEGORY | HEAD | INV value, if any, must be *minus*; in the parochial versions of Schema 3, it must be *plus*." (186) integrates these additional conditions with the informal version of the ID PRINCIPLE.

(186) *The* ID PRINCIPLE *formalized*

$$\begin{bmatrix}\textsc{dtrs} & \textit{headed-struc}\end{bmatrix} \longrightarrow$$

$$(\textsc{Schema1} \lor \textsc{Schema2} \lor \textsc{Schema3} \lor \textsc{Schema4} \lor \textsc{Schema5} \lor \textsc{Schema6})$$

where SCHEMA1 equals

$$\begin{bmatrix} \textsc{ss loc cat} & \begin{bmatrix} \textsc{head} & \begin{bmatrix}\begin{bmatrix}\textsc{inv} & \textit{minus}\end{bmatrix} \lor \neg\ \textit{verb}\end{bmatrix} \\ \textsc{subcat} & \langle\rangle \end{bmatrix} \\ \textsc{dtrs} & \begin{bmatrix} \textit{head-comp-struc} \\ \textsc{head-dtr} & \begin{bmatrix} \textit{phrase} \\ \textsc{ss nonloc to-bind slash} \ \{\} \end{bmatrix} \\ \textsc{comp-daughters} & \langle \textit{object} \rangle \end{bmatrix} \end{bmatrix},$$

SCHEMA2 equals

$$\begin{bmatrix} \textsc{ss loc cat} & \begin{bmatrix} \textsc{head} & \begin{bmatrix}\begin{bmatrix}\textsc{inv} & \textit{minus}\end{bmatrix} \lor \neg\ \textit{verb}\end{bmatrix} \\ \textsc{subcat} & \langle \textit{object} \rangle \end{bmatrix} \\ \textsc{dtrs} & \begin{bmatrix} \textit{head-comp-struc} \\ \textsc{head-dtr} & \textit{word} \end{bmatrix} \end{bmatrix},$$

SCHEMA3 equals

$$\begin{bmatrix} \textsc{ss loc cat} & \begin{bmatrix}\textsc{head inv} & \textit{plus}\\ \textsc{subcat} & \langle\rangle\end{bmatrix} \\ \textsc{dtrs} & \begin{bmatrix}\textit{head-comp-struc}\\ \textsc{head-dtr} \ \ \textit{word}\end{bmatrix}\end{bmatrix},$$

SCHEMA4 equals

$$\begin{bmatrix} \textsc{dtrs} & \begin{bmatrix}\textit{head-marker-struc}\\ \textsc{head-dtr ss nonloc to-bind slash} \ \{\}\\ \textsc{marker-dtr ss loc cat head} \ \ \textit{marker}\end{bmatrix}\end{bmatrix},$$

SCHEMA5 equals

$$\begin{bmatrix} \textsc{dtrs} & \begin{bmatrix}\textit{head-adjunct-struc}\\ \textsc{head-dtr ss} \ \ \boxed{1}\begin{bmatrix}\textsc{nonloc to-bind slash} \ \{\}\end{bmatrix}\\ \textsc{adjunct-dtr ss loc cat head mod} \ \ \boxed{1}\end{bmatrix}\end{bmatrix},$$

and SCHEMA6 equals

$$\begin{bmatrix} \textsc{dtrs} & \begin{bmatrix}\textit{head-filler-struc}\\ \textsc{filler-dtr ss loc} \ \ \boxed{1}\\ \textsc{head-dtr ss} & \begin{bmatrix}\textsc{loc cat} & \begin{bmatrix}\textsc{head vform} & \textit{finite}\\ \textsc{subcat} & \langle\rangle\end{bmatrix}\\ \textsc{nonloc} & \begin{bmatrix}\textsc{inherited slash} & \boxed{2}\\ \textsc{to-bind slash} & \{\boxed{1}\}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$
$$\wedge \ \texttt{member}(\boxed{1}, \boxed{2})$$

Notice that, although the formalization of the ID PRINCIPLE does not use logical exclusive disjunction in its consequent, it complies with the requirement of Pollard and Sag 1994 that each phrase satisfy exactly one ID schema. This is simply a consequence of the fact that the denotations of the ID schemata cannot overlap in any interpretation of the given signature.

One principle in the appendix of Pollard and Sag 1994 is listed separately from the others. It is called the RAISING PRINCIPLE:

(187) *The* RAISING PRINCIPLE *(Pollard and Sag, 1994, p. 403)*

> Let E be a lexical entry in which the (description of the) SUBCAT list L contains (a description corresponding to) a member X (of L) that is not explicitly described in E as an expletive. Then in (the

description of) the CONTENT value, X is (described as) assigned no semantic role if and only if L (is described as if it) contains a non-subject whose own SUBCAT value is $\langle$X$\rangle$.

The authors point out that the RAISING PRINCIPLE has a different status than the other principles of their grammar. Whereas all other principles describe linguistic entities (or, under the perspective of Pollard and Sag 1994, feature structures), the RAISING PRINCIPLE expresses a restriction on descriptions: It prohibits certain ways in which words could otherwise be described in the lexicon.[4] As a meta-principle about admissible lexical entries, the RAISING PRINCIPLE is not a principle that is itself expressed in the descriptive formalism. If one would want to formalize it at all, the formalization would have to be carried out in a meta-language that makes it possible to talk about expressions of the description language of the linguistic formalism. In the present framework, I would therefore need a meta-language to talk about AVM formulae.[5] Designing such a meta-language is beyond the scope of my thesis.

[4]To be more precise, if we adopt the WORD PRINCIPLE, (116), discussed in Section 4.5.3 to formalize the lexicon, the RAISING PRINCIPLE expresses restrictions on the lexical entries, LE_i, in the grammar.

[5]In that sense, the RAISING PRINCIPLE is reminiscent of the meta-level approach to lexical rules as characterized in Section 4.5.3.

Bibliography

Abeillé, Anne and Godard, Danièle 1996. La complémentation des auxiliaires français. *Langages*, 122:32–61.

Abraham, Ralph and Marsden, Jerrold E. 1967. *Foundations of Mechanics*. Benjamin/Cummings.

Ait-Kaci, Hassan 1984. *A Lattice Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures*. PhD thesis, University of Pennsylvania.

Aldag, Bjørn 1997. A proof theoretic investigation of prediction in HPSG. Master's thesis, Eberhard-Karls-Universität Tübingen.

Barwise, Jon and Perry, John 1983. *Situations and Attitudes*. Cambridge, Mass.: The MIT Press.

Birkhoff, Garrett 1967. *Lattice Theory*. American Mathematical Society, 3rd edition.

Blackburn, Patrick 1993. Modal logic and attribute value structures. In Maarten de Rijke (ed), *Diamonds and Defaults*, 19–65. Kluwer Academic Publishers.

Blackburn, Patrick 1994. Structures, languages, and translations: the structural approach to feature logic. In C. J. Rupp, M. A. Rosner, and R. L. Johnson (eds), *Constraints, Language and Computation*, 1–27. Academic Press Limited.

Blackburn, Patrick and Spaan, Edith 1993. A modal perspective on the computational complexity of attribute value grammar. *Journal of Logic, Language, and Information*, 2:129–169.

Borsley, Robert D. and Przepiórkowski, Adam (eds) 1999. *Slavic in Head-Driven Phrase Structure Grammar*. CSLI Publications.

Carpenter, Bob 1992. *The Logic of Typed Feature Structures*. Cambridge University Press. Cambridge, Massachusetts, USA.

Carpenter, Bob and Penn, Gerald 1996. Efficient parsing of compiled typed attribute value grammars. In H. Bunt and M. Tomita (eds), *Recent Advances in Parsing Technology*. Kluwer.

Dörre, Jochen and Dorna, Michael 1993. CUF - a formalism for linguistic knowledge representation. In Jochen Dörre (ed), *Computational aspects of constraint based linguistic descriptions I*, 1–22. Universität Stuttgart: DYANA-2 Deliverable R1.2.A.

Dowty, David 1996. Toward a minimalist theory of syntactic structure. In Harry Bunt and Arthur van Horck (eds), *Discontinuous Constituency*. Berlin: Mouton.

Ebbinghaus, Heinz-Dieter, Flum, Jörg, and Thomas, Wolfgang 1992. *Einführung in die mathematische Logik*. B.I.-Wissenschaftsverlag, 3rd edition.

Gallin, Daniel 1975. *Intensional and Higher-Order Modal Logic*. North-Holland, Amsterdam.

Gazdar, Gerald, Klein, Ewan, Pullum, Geoffrey K., and Sag, Ivan 1985. *Generalized Phrase Structure Grammar*. Harvard University Press. Cambridge Massachusetts.

Götz, Thilo 1999. *Feature Constraint Grammars*. PhD thesis, Eberhard-Karls-Universität Tübingen.

Götz, Thilo and Meurers, Walt Detmar 1995. Compiling HPSG type constraints into definite clause programs. In *Proceedings of the 33rd Annual Meeting of the ACL*, 85–91, Cambridge, MA: MIT. Association for Computational Linguistics.

Götz, Thilo and Meurers, Walt Detmar 1997a. The ConTroll system as large grammar development platform. In *Proceedings of the Workshop "Computational Environments for Grammar Development and Linguistic Engineering (ENVGRAM)" held in conjunction with the 35th Annual Meeting* *of the ACL and 8th Conference of the EACL*, 38–45, Madrid: Universidad Nacional de Educación a Distancia. Association for Computational Linguistics.

Götz, Thilo and Meurers, Walt Detmar 1997b. Interleaving universal principles and relational constraints over typed feature logic. In *Proceedings of the 35th Annual Meeting of the ACL and 8th Conference of the EACL*, 1–8, Madrid: Universidad Nacional de Educación a Distancia. Association for Computational Linguistics.

Green, Georgia M. 2000. Modelling grammar growth: Universal grammar without innate principles or parameters. *Note:* Unpublished manuscript, available from `http://www.cogsci.uiuc.edu/~green/`, version dated January 2000.

Groenendijk, Jeroen and Stokhof, Martin 1982. Semantic analysis of *wh*-complements. *Linguistics and Philosophy*, 5:175–233.

Hindley, J. Roger and Seldin, Jonathan P. 1986. *Introduction to Combinators and the lambda-Calculus*. Cambridge University Press.

Hinrichs, Erhard, Meurers, Detmar, Richter, Frank, Sailer, Manfred, and Winhart, Heike (eds) 1997. *Ein HPSG-Fragment des Deutschen, Teil 1: Theorie*, (= *Arbeitspapiere des SFB 340, Nr. 95*). Eberhard-Karls-Universität Tübingen.

Hinrichs, Erhard and Nakazawa, Tsuneko 1994. Linearizing AUXs in German Verbal Complexes. In John Nerbonne, Klaus Netter, and Carl Pollard (eds), *German in Head-Driven Phrase Structure Grammar*, 11–37. CSLI Publications.

Höhfeld, Markus and Smolka, Gerd 1988. Definite relations over constraint languages. LILOG Report 53, IBM Deutschland.

Höhle, Tilman N. 1999. An Architecture for Phonology. In Robert D. Borsley and Adam Przepiórkowski (eds), *Slavic in HPSG*, 61–90. CSLI Publications.

Johnson, Mark 1988. *Attribute-Value Logic and the Theory of Grammar*. CSLI Publications.

Johnson, Mark 1995. Logic and feature structures. In Mary Dalrymple, Annie Zaenen, John Maxwell III., and Ronald M. Kaplan (eds), *Formal Issues in Lexical-Functional Grammar*, 369–380. CSLI Publications.

Kaplan, Ronald M. 1995. Three seductions of computational linguistics. In Mary Dalrymple, Annie Zaenen, John Maxwell III., and Ronald M. Kaplan (eds), *Formal Issues in Lexical-Functional Grammar*, 339–367. CSLI Publications.

Kasper, Robert T. and Rounds, William C. 1986. A logical semantics for feature structures. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, 257–266.

Kathol, Andreas 1995. *Linearization-Based German Syntax*. PhD thesis, Ohio State University.

Kathol, Andreas and Pollard, Carl 1995. Extraposition via complex domain formation. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, 174–180.

Kepser, Stephan 1994. A satisfiability algorithm for a typed feature logic. Arbeitspapiere des SFB 340 60, Eberhard-Karls-Universität Tübingen.

King, Paul J. 1989. *A logical Formalism for Head-Driven Phrase Structure Grammar*. PhD thesis, University of Manchester.

King, Paul J. 1994. An expanded logical formalism for Head-Driven Phrase Structure Grammar. Arbeitspapiere des SFB 340 59, Eberhard-Karls-Universität Tübingen.

King, Paul J. 1995. From unification to constraint. *Note:* Unpublished lecture notes. Eberhard-Karls-Universität Tübingen.

King, Paul J. 1996. From unification to constraint: An evolving formalism for Head-driven Phrase Structure Grammar. *Note:* Lecture Notes for the European Summer School in Logic Language and Information, 12–16 August 1996.

King, Paul J. 1999. Towards Truth in Head-driven Phrase Structure Grammar. In Valia Kordoni (ed), *Tübingen Studies in Head-Driven Phrase Structure Grammar*, (= *Arbeitspapiere des SFB 340, Nr. 132, Volume 2*), 301–352. Eberhard-Karls-Universität Tübingen.

King, Paul John, Simov, Kiril Ivanov, and Aldag, Bjørn 1999. The complexity of modellability in finite and computable signatures of a constraint logic for Head-driven Phrase Structure Grammar. *The Journal of Logic, Language and Information*, 8.1:83–110.

Kordoni, Valia (ed) 1999. *Tübingen Studies in Head-Driven Phrase Structure Grammar*, (= *Arbeitspapiere des SFB 340, Nr. 132, Volume 1 & 2*). Eberhard-Karls-Universität Tübingen.

Kracht, Marcus 1995. Is there a genuine modal perspective on feature structures? *Linguistics and Philosophy*, 18:401–458.

Kupść, Anna 1999. Haplology of the Polish Reflexive Marker. In Robert D. Borsley and Adam Przepiórkowski (eds), *Slavic in HPSG*, 91–124. CSLI Publications.

Lewis, Harry R. and Papadimitriou, Christos H. 1981. *Elements of the Theory of Computation*. Prentice-Hall, Inc. Englewood Cliffs, New Jersey 07632.

Meurers, Walt Detmar 1999. Raising spirits (and assigning them case). *Groninger Arbeiten zur Germanistischen Linguistik (GAGL)*, 43:173–226.

Meurers, Walt Detmar 2000. *Lexical Generalizations in the Syntax of German Non-Finite Constructions, (= Arbeitspapiere des SFB 340, Nr. 145)*. PhD thesis, Eberhard-Karls-Universität Tübingen.

Meurers, W. Detmar and Minnen, Guido 1997. A computational treatment of lexical rules in HPSG as covariation in lexical entries. *Computational Linguistics*, 23.4:543–568.

Moshier, Michael Andrew 1988. *Extensions to Unification Grammar for the Description of Programming Languages*. PhD thesis, University of Michigan.

Moshier, M. Andrew and Pollard, Carl J. 1994. The domain of set-valued feature structures. *Linguistics and Philosophy*, 17:607–631.

Moshier, M. Andrew and Rounds, William C. 1987. A logic for partially specified data structures. In *Proceedings of the 14th ACM symposium on principles of programming languages*, 156–167.

Nerode, Anil 1958. Linear automaton transformations. In *Proceedings of the American Mathematical Society*, 541–544.

Partee, Barbara H., ter Meulen, Alice, and Wall, Robert E. 1990. *Mathematical Methods in Linguistics*. Kluwer Academic Publishers.

Penn, Gerald 1998. Parametric types for typed attribute-value logic. In *Proceedings of th 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics, Montreal, Canada*.

Penn, Gerald 1999a. A Generalized-Domain-Based Approach to Serbo-Croation Second Position Clitic Placement. In Gosse Bouma, Erhard Hinrichs, Geert-Jan M. Kruijff, and Richard T. Oehrle (eds), *Constraints and Resources in Natural Language Syntax and Semantics*, 119–136. CSLI Publications.

Penn, Gerald 1999b. An RSRL Formalization of Serbo-Croatian Second Position Clitic Placement. In Valia Kordoni (ed), *Tübingen Studies in Head-Driven Phrase Structure Grammar*, (= *Arbeitspapiere des SFB 340, Nr. 132, Volume 1*), 177–197. Eberhard-Karls-Universität Tübingen.

Penn, Gerald 1999c. Linearization and *WH*-Extraction in HPSG: Evidence from Serbo-Croatian. In Robert D. Borsley and Adam Przepiórkowski (eds), *Slavic in HPSG*, 149–182. CSLI Publications.

Pereira, Fernando C. N. and Shieber, Stuart M. 1984. The semantics of grammar formalisms seen as computer languages. In *Proceedings of COLING 84*, 123–129.

Pollard, Carl and Sag, Ivan A. 1987. *Information-Based Syntax and Semantics. Vol.1: Fundamentals*. CSLI Lecture Notes 13.

Pollard, Carl and Sag, Ivan A. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Pollard, Carl J. 1989. Sorts in unification-based grammar and what they mean. To appear in: Pollard 2000. *Note:* Manuscript. Carnegie-Mellon University.

Pollard, Carl J. 1996. The nature of constraint-based grammar. To appear in: Pollard 2000. *Note:* Keynote Lecture, Pacific Asia Conference on Language, Information, and Computation. Kyung Hee University, Seoul, Korea, December 20, 1996.

Pollard, Carl J. 1999. Strong generative capacity in HPSG. In Gert Webelhuth, Jean-Pierre Koenig, and Andreas Kathol (eds), *Lexical and Constructional Aspects of Linguistic Explanation*, 281–297. CSLI Publications.

Pollard, Carl J. 2000. *Lectures on Constraint-Based Grammar*. CSLI Publications. *Note:* Forthcoming.

Pollard, Carl J. and Moshier, M. Drew 1990. Unifying partial descriptions of sets. In Philip P. Hanson (ed), *Information, Language, and Cognition*, 285–322. University of British Columbia Press. Vancouver. *Note:* Cited after the 1994 edition, Oxford University Press, Oxford, England.

Pollard, Carl J. and Yoo, Eun Jung 1998. A unified theory of scope for quantifiers and *wh*-phrases. *Journal of Linguistics*, 34:415–445.

Przepiórkowski, Adam 1997. Quantifiers, adjuncts as complements, and sope ambiguities. *Note:* To appear in *Journal of Linguistics*. Draft of June 20, 1997.

Przepiórkowski, Adam 1998. 'A unified theory of scope' revisited. Quantifier retrieval without spurious ambiguities. In Gosse Bouma, Geert-Jan M. Kruijff, and Richard T. Oehrle (eds), *Proceedings of the FHCG-98, 14–16 August 1998, Saarbrücken*, 185–195.

Przepiórkowski, Adam 1999a. *Case Assignment and the Complement-Adjunct Dichotomy: A Non-Configurational Constraint-Based Approach*. PhD thesis, Eberhard-Karls-Universität Tübingen.

Przepiórkowski, Adam 1999b. On case assignment and "adjuncts as complements". In Gert Webelhuth, Jean-Pierre Koenig, and Andreas Kathol (eds), *Lexical and Constructional Aspects of Linguistic Explanation*, 231–245. CSLI Publications.

Reape, Mike 1989. A logical treatment of semi-free word order and bounded discontinuous constituency. In *Proceedings of the 4th Conference of the European Chapter of the ACL*, 103–110.

Reape, Mike 1990. Getting things in order. In *Proceedings of the Symposium on Discontinuous Constituency*. Institute for Language Technology and Artificial Intelligence.

Reape, Mike 1991. An introduction to the semantics of unification-based grammar formalisms. DYANA Deliverable R3.2.A, Centre for Cognitive Science, University of Edinburgh.

Reape, Mike 1992. *A Formal Theory of Word Order: A Case Study of West Germanic*. PhD thesis, University of Edinburgh.

Reape, Mike 1994a. Domain Union and Word Order Variation in German. In John Nerbonne, Klaus Netter, and Carl Pollard (eds), *German in Head-Driven Phrase Structure Grammar*, 151–197. CSLI Publications.

Reape, Mike 1994b. A feature value logic with intensionality, nonwellfoundedness and functional and relational dependencies. In C. J. Rupp, M. A. Rosner, and R. L. Johnson (eds), *Constraints, Language and Computation*, 77–110. Academic Press Limited.

Richter, Frank 1997. Die Satzstruktur des Deutschen und die Behandlung langer Abhängigkeiten in einer Linearisierungsgrammatik. Formale Grundlagen und Implementierung in einem HPSG-Fragment. In: Hinrichs et al. 1997.

Richter, Frank 1999. RSRL for HPSG. In Valia Kordoni (ed), *Tübingen Studies in Head-Driven Phrase Structure Grammar*, (= *Arbeitspapiere des SFB 340, Nr. 132*), 74–115. Eberhard-Karls-Universität Tübingen.

Richter, Frank and King, Paul J. 1997. On the Existence of Exhaustive Models in a Relational Feature Logic for Head-driven Phrase Structure Grammar. *Note:* Manuscript.

Richter, Frank and Sailer, Manfred 1995. Remarks on Linearization. Reflections on the Treatment of LP-Rules in HPSG in a Typed Feature Logic. Master's thesis, Eberhard-Karls-Universität Tübingen.

Richter, Frank and Sailer, Manfred 1999a. A lexicalist collocation analysis of sentential negation and negative concord in French. In Valia Kordoni (ed), *Tübingen Studies in Head-Driven Phrase Structure Grammar*, (= *Arbeitspapiere des SFB 340, Nr. 132, Volume 1*), 231–300. Eberhard-Karls-Universität Tübingen.

Richter, Frank and Sailer, Manfred 1999b. LF Conditions on Expressions of Ty2: An HPSG Analysis of Negative Concord in Polish. In Robert D. Borsley and Adam Przepiórkowski (eds), *Slavic in HPSG*, 247–282. CSLI Publications.

Richter, Frank and Sailer, Manfred 1999c. Underspecified Semantics in HPSG. In Harry Bunt and Reinhard Muskens (eds), *Computing Meaning*, (= *Studies in Linguistics and Philosophy*), 95–112. Kluwer academic publishers.

Richter, Frank and Sailer, Manfred 2000. On the left Periphery of German Finite Sentences. In Detmar Meurers and Tibor Kiss (eds), *Topics in Constraint-Based Approaches to Germanic Syntax (Working Title)*. CSLI Publications. *Note:* Forthcoming.

Richter, Frank, Sailer, Manfred, and Penn, Gerald 1999. A Formal Interpretation of Relations and Quantification in HPSG. In Gosse Bouma, Erhard Hinrichs, Geert-Jan M. Kruijff, and Richard T. Oehrle (eds), *Constraints and Resources in Natural Language Syntax and Semantics*, 281–298. CSLI Publications.

Rounds, William C. 1997. Feature logics. In Johan van Benthem and Alice ter Meulen (eds), *Handbook of Logic and Language*, 475–533. Elsevier.

Rounds, William C. and Kasper, Robert 1986. A complete logical calculus for record structures representing linguistic information. In *Proceedings of the IEEE Symposium on Logic in Computer Science*, 38–43.

Sag, Ivan A. 1997. English relative clause constructions. *Journal of Linguistics*, 33:431–483.

Sailer, Manfred 2000. *The Content of* CONTENT. *In progress. Draft version of January 31, 2000*. PhD thesis, Eberhard-Karls-Universität Tübingen.

Shieber, Stuart M. 1986. *An Introduction to Unification-based Approaches to Grammar*, (= *CSLI Lecture Notes*, 4). CSLI Publications.

Smolka, Gert 1988. A feature logic with subsorts. Technical report, LILOG technical report 33, IBM Deutschland GmbH, Stuttgart, Germany.

Smolka, Gert 1992. Feature-constraint logics for unification grammars. *Journal of Logic Programming*, 12:51–87.

Smolka, Gert and Treinen, Ralf 1994. Records for logic programming. *The Journal of Logic Programming*, 18:229–258.

Vickers, Steven 1989. *Topology via Logic*. Cambridge University Press.

Yardeni, Eyal, Frühwirth, Thom, and Shapiro, Ehud 1992. Polymorphically typed logic programs. In Frank Pfenning (ed), *Types in Logic Programming*, 63–90. Cambridge, Mass.: The MIT Press.

Zimmermann, Thomas Ede 1989. Intensional logic and two-sorted type theory. *The Journal of Symbolic Logic*, 54:65–77.