Bericht Nr. 59, 1994

Arbeitspapiere des
Sonderforschungsbereichs 340

ISSN 0947-6954/94

SPRACHTHEORETISCHE GRUNDLAGEN FÜR DIE COMPUTERLINGUISTIK

Paul John King:

An expanded logical formalism for head-driven phrase structure grammar

Universität Stuttgart Universität Tübingen
IBM Deutschland Informationssysteme GmbH

An expanded logical formalism for head-driven phrase structure grammar

Paul John King
Sonderforschungsbereich 340
Seminar für Sprachwissenschaft
Eberhard-Karls-Universität
Wilhelmstraße 113
72074 Tübingen
Germany
Email: king@sfs.nphil.uni-tuebingen.de

4th July, 1994

Abstract. Though [Pollard and Sag 1994] assumes that an unspecified variant of the formal logic of [Carpenter 1992] will provide a formalism for HPSG, a precise formulation of the envisaged formalism is not immediately obvious, primarily because a principal tenet of [Carpenter 1992], that feature structures represent partial information, seems to conflict with a principal tenet of [Pollard and Sag 1994], that feature structures represent abstract linguistic entities. This has caused many HPSGians to be mistakenly concerned with partial-information specific notions, such as subsumption, that are appropriate for the [Carpenter 1992] logic but inappropriate for the formalism [Pollard and Sag 1994] envisages. This paper hopes to allay this concern and the confusion it engenders by substituting [King 1989] for [Carpenter 1992] as the basis of the envisaged formalism. It demonstrates that the formal logic of [King 1989] provides a formalism for HPSG that meets all [Pollard and Sag 1994] asks of the envisaged formalism. It further shows that the most credible variant of the [Carpenter 1992] logic consistent with the aims of [Pollard and Sag 1994] is not only incompatible with the tenet that feature structures represent partial information, but also an instance of the [King 1989] logic.

1 Introduction

A grammar comprises two components: a means to express accounts of natural language fragments, and a collection of accounts of natural language fragments expressed via this means. For facility, I henceforth (mis)use the term 'formalism' to refer to the first of these components. Though a natural language can serve as a formalism, a grammar that yields painstaking accounts of extensive natural language fragments requires a more precise formalism. Since mathematics offers peerless precision of expression, many grammarians espouse mathematical formalisms. In particular, researchers have provided a succession of mathematical formalisms for the HPSG (head-driven phrase structure grammar) of [Pollard and Sag 1987] and [Pollard and Sag 1994]. This paper resolves a troubling aspect of the mathematical formalism [Pollard and Sag 1994] envisages for HPSG. However, an understanding of what troubles me and the method I use to resolve it requires a review of earlier HPSG formalisms and a summary of the formalism [Pollard and Sag 1994] envisages.

[Pollard and Sag 1987] originally gave HPSG a formalism that uses mathematical entities called feature structures to represent partial information about linguistic entities, and expresses an account of a natural language fragment as a feature structure representing the partial information about linguistic entities that is available to mature users of the fragment. The feature structures constitute a Heyting algebra (see [Partee et al. 1990, chapter 12]), and the formalism uses the algebraic operations to construct representations of the partial information available to mature users of natural language fragments. [Pollard and Sag 1987] held that some linguistic information is universal, being

> the information about language that is available to all normal adult humans by virtue of their shared genetic endowment, independent of which language communities they belong to. [Pollard and Sag 1987, page 44]

[Pollard and Sag 1987] codified this universal linguistic information into a small number of so-called universal principles. All of the universal principles are implicative. For example,

> To say that [the head feature principle] is a universal principle is to say: human beings implicitly have the information about any linguistic object—in any language whatsoever—that if it is a phrasal sign then its head features coincide with the head features on its head daughter. [Pollard and Sag 1987, page 44]

In order to construct accurate representations of implicative information, the formalism must meet the condition that a feature structure represents partial information about a linguistic entity iff the pseudocomplement of the feature structure represents partial information not about the entity. However, [King 1989] showed that the formalism does not meet this condition, and researchers consequently pursued three new approaches to providing an HPSG formalism, the first based on the algebra of [Pollard and Sag 1987], the second on the KRL (Kasper-Rounds logic) of [Rounds and Kasper 1986], and the third on the SRL (speciate re-entrant logic) of [King 1989].

HPSG uses a taxonomic system whereby each linguistic entity is assigned one or more symbols called sorts or types in order to encode linguistic-entity well-formedness conditions called appropriateness conditions. The [Pollard 1989] and [Pollard and Moshier 1990] HPSG formalisms retain the [Pollard and Sag 1987] Heyting algebra, but abandon the pseudocomplement operation, and use instead sorts to encode implicative information. The formalisms express an account of a natural language fragment as a total function from sorts to feature structures that assigns each sort a feature structure representing the partial information available to mature fragment users about linguistic entities of that sort. Informally, assigning sort σ feature structure F represents the implicative information 'if the linguistic entity is of sort σ then F is about the entity'. Thus, implication is not part of the algebra, but is reified as the algebra is used to express a natural language fragment. The formalisms can only express implicative information in which the antecedent is of the form 'the linguistic entity is of such-and-such a sort', but many HPSGians believe that, by judiciously reconfiguring the sort taxonomy, these formalisms can express any reasonable HPSG account of a natural language fragment.

Underlying KRL is the intuition that partial information can be accrued. That is, adding information to a first piece of partial information yields a second, more informative piece of partial information. I say that the second piece of partial information elaborates the first. Adding information ad infinitum to a piece of partial information therefore yields an infinite sequence of pieces of partial information, where each successor in the sequence elaborates its predecessor. I call such a sequence an elaboration sequence. One can picture an elaboration sequence as a succession of ever more explicit pieces of partial information tending asymptotically to some limiting piece of 'total' information, much like a succession of ever longer decimal numbers tending asymptotically to some limiting real number. I call such a limiting piece of 'total' information an ultimate elaboration of the initial piece of partial information. The fundamental presupposition of KRL is that a piece of partial information is significant not only for itself but also for its ultimate elaborations, these being indicated by the elaboration sequences it initiates.

Like the [Pollard and Sag 1987] formalism, KRL uses feature structures to represent partial information. KRL uses a subsumption relation to represent elaboration: a first feature structure subsumes a second feature structure iff the partial information the second represents elaborates the partial information the first represents. Thus, KRL represents an elaboration sequence as an infinite sequence of feature structures, where each member of the sequence subsumes its successor. However, unlike the [Pollard and Sag 1987] formalism, KRL uses a formal language to constrain the feature structures. KRL provides a set of formulae, and a satisfaction relation between the feature structures and the formulae. The set of formulae is closed under conjunction and disjunction, and a feature structure satisfies the conjunction/disjunction of two formulae iff the feature structure satisfies the first and/or the second formula. KRL expresses its concern with ultimate elaborations in a condition that I call subsumption monotonicity: if a first feature structure satisfies a formula and subsumes a second feature structure then the second feature structure satisfies the formula. Subsumption monotonicity ensures that if the representation of a piece of partial information satisfies a formula then the representation of each member of each elaboration chain the piece of partial information initiates satisfies the formula, or, more informally, that a formula cannot denote a piece of partial information without 'denoting' each of its ultimate elaborations. [Rounds and Kasper 1986] showed that KRL is subsumption monotonic.

A [Pollard and Sag 1987] feature structure and a KRL feature structure differ markedly. However, the EKR (extended Kasper-Rounds logic) of [Moshier 1988] incorporates sorts into KRL such that a [Pollard and Sag 1987] feature structure is a set of EKR feature structures with the set representing the disjunction of the partial information its members represent, and the LTFS (logic of typed feature structures) of [Carpenter 1992] incorporates appropriateness conditions and a richer formal language into EKR. This encouraged some researchers to attempt to base HPSG formalisms on KRL-style logics. However, KRL, EKR and LTFS cannot deliver representations of implicative information. Researchers overcame this limitation in two ways, the first mimicking the [Pollard and Sag 1987] formalism, the second the [Pollard 1989] and [Pollard and Moshier 1990] formalisms. The first method

> incorporates negation into a KRL-style logic so that the partial information the set of satisfiers of the negation of a formula represents is the negation of the partial information the set of satisfiers of the formula represents, and
>
> expresses an account of a natural language fragment as a KRL-style formula whose satisfiers constitute a [Pollard and Sag 1987] feature structure representing the partial information available to mature fragment users.

This method appears simple at first sight. For example, close the formulae under negation, and extend satisfaction classically so that a feature structure satisfies the negation of a formula iff the feature structure does not satisfy the formula. However, this apparent simplicity is deceptive. Firstly, [Moshier and Rounds 1987] showed that a KRL-style logic with classical negation is not subsumption monotonic. Secondly, in order to deliver accurate representations of implicative information, the formalism must meet the condition that some satisfier of a formula represents partial information about a linguistic entity iff no satisfier of the negation of the formula represents partial information about the entity. It is easy to show that a KRL-style logic with classical negation does not meet this condition. [Moshier and Rounds 1987], [Langholm 1989] and others have proposed various subsumption-monotonic extensions of KRL with nonclassical negations, and these extensions easily adapt to EKR and LTFS, but it remains an open question whether any of these extensions can deliver accurate representations of implicative information. The second method retains a negation-free KRL-style logic, but expresses an account of a natural language fragment as a total function from sorts to KRL-style formulae that assigns each sort a formula whose satisfiers constitute a [Pollard and Sag 1987] feature structure representing the partial information available to mature fragment users about linguistic entities of that sort. [Carpenter 1992] based such an HPSG formalism on LTFS. [Pollard 1989] quantified the parallels between this formalism and the [Pollard 1989] and [Pollard and Moshier 1990] formalisms, and the former both suffers the same expressive limitations and of course merits the same HPSGian confidence as the latter. Moreover, the ALE (attribute logic engine) of [Carpenter 1993] is a computer implementation of LTFS. As a result, the [Carpenter 1992] formalism is very popular among HPSGians.

Underlying SRL is the intuition that a description is a well-formed string of symbols that can be interpreted as true or false of an object. For example, the description 'it is black' can be conventionally interpreted as true of a soot particle but false of a snow flake. To formalise this intuition, SRL provides a class of formal languages, each comprising a set of descriptions and a class of interpretations, where each interpretation comprises a universe of objects and a denotation function that assigns each description the collection of objects in the universe of which the description is true. The set of descriptions is closed under negation, conjunction and disjunction. Formally, the denotation of the negation of a description is the complement of the denotation of the description, and the denotation of the conjunction/disjunction of two descriptions is the intersection/union of the denotations of the two descriptions. Informally, the negation of a description describes an entity iff the description does not describe the entity, and the conjunction/disjunction of two descriptions describes an entity iff the first and/or the second description describes the entity.

Though [King 1989] created SRL specifically to provide a formalism for HPSG, partial information plays no part in SRL. Rather, [King 1989] reasoned that partial information in HPSG is actually an irrelevance. [King 1989] observed that [Pollard and Sag 1987] used feature structures sometimes to represent partial information:

> The fundamental nature of linguistic information remains a mystery. We cannot look at a piece of linguistic information under a microscope; we aren't even sure whether to look inside human brains or somewhere else. Still, although we aren't sure what linguistic information is, we can still theorize about it by trying to say what it is like. To be more precise, we can try to construct formal *models* that reflect certain interesting aspects of the things we are studying. ...
>
> In the past several years, theoretical and computational linguists working within a number of distinct research traditions have found it useful to construct formal models of linguistic information. ...the formal objects of choice in information-based linguistics are things known as *feature structures*. [Pollard and Sag 1987, page 27]

sometimes to describe linguistic entities:

> In our applications, we will think of feature structures as partial descriptions of signs (or sign tokens) and other linguistic objects which occur as parts of signs. [Pollard and Sag 1987, page 28]

and sometimes to do both:

> Intuitively, a feature structure is just an information-bearing object that describes or represents another thing by specifying *values* for various *attributes* of the described thing; we think of the feature structure as providing partial information about the thing described. [Pollard and Sag 1987, page 28]

Evidently, [Pollard and Sag 1987] conflated 'partial information about a linguistic entity' with 'description of a linguistic entity'. In fact, description is the only aspect of partial information the [Pollard and Sag 1987] formalism uses to express an account of a natural language fragment. For example, see the explanation of the head feature principle I quoted earlier from [Pollard and Sag 1987]. [King 1989] argued that in order to meet concerns about irrelevant, nondescriptive aspects of partial information, [Pollard and Sag 1987] had formulated an overly complex formalism unable to meet concerns about the relevant, descriptive aspect of partial information. For example, in order to construct accurate representations of the descriptive aspect of implicative information, the formalism must meet the simple condition that the pseudocomplement of a feature structure describes a linguistic entity iff the feature structure does not describe the entity. Yet, as a means to meet this condition, the formalism is both unnecessarily complicated and demonstrably unsuccessful. Choosing to focus solely on the description of linguistic entities, [King 1989] took each linguistic entity to be an SRL object; substituted SRL descriptions for feature structures, SRL conjunction for meet, SRL disjunction for join, and SRL negation for pseudocomplement; and expressed an account of a natural language fragment as an SRL formula denoting the collection of linguistic entities acceptable to mature fragment users. The [King 1989] formalism's inherent narrowing of scope from partial information about linguistic entities to linguistic entities alone is both beneficial and detrimental. On the one hand, the [King 1989] formalism can express an account of a natural language fragment without the inaccuracies or expressive limitations of the other HPSG formalisms. On the other, the [King 1989] formalism cannot distinguish between natural language fragments as finely as the other HPSG formalisms can. For example, suppose that I_1 and I_2 are the partial information available to mature users of natural language fragments F_1 and F_2 respectively. Suppose further that I_1 and I_2 are different, but mature users of F_1 and F_2 deem exactly the same linguistic entities acceptable. Unlike the other HPSG formalisms, the [King 1989] formalism cannot distinguish between F_1 and F_2.

There is a sharp ontological distinction between KRL-style logics and SRL: a KRL-style logic formalises the accrual of partial information, whereas SRL formalises the description of objects. Thus, partial-information specific notions, such as subsumption, are central to a KRL-style logic, but irrelevant to SRL. The ontological distinction engenders an equally sharp distinction in the HPSG formalisms the logics provide: a KRL-based formalism delivers a representation of partial information, whereas an SRL-based formalism delivers a collection of linguistic entities. Thus, a formula constrains a linguistic entity directly in an SRL-based formalism, but indirectly in a KRL-based formalism: the KRL-style formula constrains a representation of partial information, and the partial information is about the linguistic entity. Classical negation illustrates both differences clearly. Firstly, classical negation conflicts with subsumption monotonicity, an intuition that underlies KRL-style logics, whereas classical negation accords with each intuition that underlies SRL. Secondly, KRL-based HPSG formalisms express each universal principle as implicative information whose accurate representation precludes classical negation, whereas SRL-based HPSG formalisms express each universal principle as an implicative description whose accurate interpretation requires classical negation. However, there is a widespread belief, even among researchers who acknowledge these differences, that, differences of ontology and application apart, SRL is a predicate-logic generalisation of LTFS – the KRL-style logic that most closely resembles SRL – with classical negation. This belief is false. Before showing how it is false I will show how it became prevalent, via a short detour into a formalism for another grammar, the LFG (lexical-functional grammar) of [Kaplan and Bresnan 1982].

[Johnson 1988] created AVL (attribute-value logic) specifically to provide a formalism for LFG. Despite their independent development and a number of nontrivial technical differences, AVL and SRL have much in common, and it is natural to consider AVL and SRL sister logics, the former tailored to LFG, the latter to HPSG. In particular, both are concerned with objects, not partial information. [Johnson 1988] did outline an embedding of KRL within AVL that generalised subsumption from KRL to AVL, but [Johnson 1988] also warned that whereas subsumption formalises the "partiality" that underlies what [Johnson 1988] called "unification approaches", it formalises nothing in AVL. Several researchers, including perhaps Johnson himself, ignored or underestimated this warning. For example, [Johnson 1988] offered the classical negation of AVL as a means to include negation in the "unification approaches", and [Smolka 1988] explicitly presented FL (feature logic), an extension of AVL that retains classical negation, as an extension not only of AVL but also of KRL. Conversely, [Carpenter 1992] criticised AVL for not being subsumption monotonic. Nevertheless, even if the [Johnson 1988] warning is heeded, it remains true that AVL is a predicate-logic generalisation of KRL with classical negation. Given the similarity between AVL and SRL, it is unsurprising therefore that many believe SRL is a predicate-logic generalisation of LTFS with classical negation.

The mendacity of this last belief arises from the different provenances of AVL and SRL. LFG is rather flexible about what constitutes a linguistic entity, and one AVL object can appear to be an elaboration of another. This provokes the suspicion that the partial-information/object distinction is an illusion, an ontological ruse to admit classical negation into KRL. However, the appropriateness conditions make HPSG quite strict about what constitutes a linguistic entity, and each SRL object meets the appropriateness conditions exhaustively. The LTFS feature structures also observe the appropriateness conditions, but only partially. Consequently, an LTFS feature structure cannot ordinarily serve as an SRL object. Later in this paper, I reconfigure the LTFS feature structures to meet the appropriateness conditions exhaustively, and I embed the reconfigured LTFS within SRL. However, the reconfigured feature structures are too large to be ordinary LTFS feature structures, and subsumption over the reconfigured feature structures flattens to feature structure isomorphism.[1] The reconfigured feature structures represent possibly infinite information that cannot be elaborated. Thus, the partial-information/object distinction that seemed immaterial between KRL and AVL becomes substantial between LTFS and SRL: LTFS can be embedded in SRL only by a reconfiguration of LTFS that nullifies the intuitions of [Carpenter 1992] concerning partial information.

[Pollard and Sag 1994] itself provides no formalism for HPSG, but envisages instead an HPSG formalism that synthesises components taken from several of the HPSG formalisms reviewed above. Like [King 1989], [Pollard and Sag 1994] seeks to express an account of a natural language fragment by denoting the collection of linguistic entities acceptable to mature fragment users. However, [King 1989] and [Pollard and Sag 1994] disagree over both the nature of the linguistic entities, and how to denote them.

[Pollard and Sag 1994] distinguishes between linguistic tokens and linguistic types.[2] A token is a concrete particular, whereas a type is an abstract universal that signifies a collection of tokens. I say that a type abstracts a token iff the type signifies a collection containing the token. For example, the individual disturbance in air pressure generated by a specific utterance of the English sentence 'I think' by a specific person at a specific time in a specific location is a linguistic token, the sequence of English phonemes represented by the string /aɪ θɪŋk/ of IPA (international phonetic alphabet) symbols is a linguistic type, and the latter abstracts the former. Though [King 1989] never mentions tokens, [King 1989] evidently considers linguistic entities to be token-like. [Pollard and Sag 1994] has a different opinion however.

> One thing that [language] certainly does not consist of is individual linguistic events or utterance tokens, for knowledge of these is not what is shared among the members of a linguistic community. Instead, what is known in common, that makes communication possible, is the system of linguistic types. For example, the type of the sentence *I'm sleepy* is part of that system, but no individual token of it is.
>
> [Pollard and Sag 1994, page 14]

[1] Thus, ironically, [Carpenter 1992] was not only wide of the mark, but also wrong to criticise SRL for not being subsumption monotonic!

[2] It is unfortunate that [Pollard and Sag 1987] and [Pollard and Sag 1994] use the same term 'type', but for different purposes. A [Pollard and Sag 1987] type is actually a [Pollard and Sag 1994] sort. For clarity, I henceforth keep to the [Pollard and Sag 1994] usage.

[Pollard and Sag 1994] concedes that the nature of a linguistic type is unclear.

> Just what sorts of things these linguistic types are is another question. Indeed, the precise ontological status of linguistic types is the subject of a very long-standing debate among various schools of conceptualists (e.g. Ferdinand de Saussure, Noam Chomsky), who take them to be mental objects, and realists (e.g. Leonard Bloomfield, Jerrold Katz, Paul Postal, Jon Barwise), who consider them to belong to extramental reality. Thus we might identify linguistic types with such psychological entities as Saussure's signs or with certain presumably nonmental objects of situation theory (situation types or perhaps infons). For our part, we doubt that the question of whether objects of knowledge are mental or extramental is an empirical one.
>
> [Pollard and Sag 1994, page 14]

In order to gain a purchase on the recondite linguistic types, [Pollard and Sag 1994] makes the methodological assumption that scientists model empirical objects with corresponding mathematical objects, theorise about the mathematical objects directly, and theorise about the empirical objects indirectly via the correspondence.

> In any mathematical theory about an empirical domain, the phenomena of interest are *modelled* by mathematical structures, certain aspects of which are conventionally understood as corresponding to observables of the domain. The theory itself does not talk directly about the empirical phenomena; instead, it talks about, or is *interpreted by*, the modelling structures. Thus the predictive power of the theory arises from the conventional correspondence between the model and the empirical domain.
>
> [Pollard and Sag 1994, page 6]

[Pollard and Sag 1994] re-enlists feature structures, albeit to represent linguistic types, not partial information, and envisages expressing an account of a natural language fragment by denoting the collection of feature structures that represent the linguistic entities acceptable to mature fragment users.

> In HPSG, the modelling domain ... is a system of *sorted feature structures*, which are intended to stand in a one-to-one relation with types of natural language expressions and their subparts. The role of the linguistic theory is to give a precise specification of which feature structures are to be considered admissible; the types of linguistic entities that correspond to the admissible feature structures constitute the predictions of the theory. [Pollard and Sag 1994, page 8]

[Pollard and Sag 1994] makes very clear that feature structures used to represent partial information differ markedly from feature structures used to represent linguistic types, and that partial-information specific notions, such as subsumption, play no part in the HPSG formalism it envisages. Nonetheless, [Pollard and Sag 1994] assumes that some variant of LTFS will be the basis of the envisaged formalism.

> [[Carpenter 1992]] in particular sets forth a logic very close to the one that we assume will underlie a fully formalized version of our theory. In very general terms, this can be characterized as a sorted variant of Kasper and Rounds's logic (1986) augmented with path inequalities, definite relations, and set values. [Pollard and Sag 1994, page 8]

This last assumption troubles me deeply. LTFS is a logic of partial information, but the formalism [Pollard and Sag 1994] envisages requires a logic of description, and such logics are immiscible in the presence of appropriateness conditions. To settle my disquiet, I do the following in this paper. I use SRL, a logic of description, to build an HPSG formalism that includes tokens, types, type abstraction and type admission. I then build a mathematical model of this formalism. I then use the model to prove that the formalism meets all [Pollard and Sag 1994] asks of the envisaged HPSG formalism. I lastly show that the most credible LTFS variant that meets the requirements of [Pollard and Sag 1994] is not just incompatible with the intuitions about partial information that underlie LTFS, but is in fact an instance of SRL.

2 The K-formalism

In this section I use SRL to build an HPSG formalism I call the K-formalism. I review the formal language of SRL, repeating the definitions of [King 1989], but with a small number of minor notational and terminological improvements. I then construct the K-formalism by giving SRL-based definitions of tokens, types, what it is for a type to abstract a token, and what it is for a theory to admit a type.

2.1 The logic

SRL provides a class of formal languages, each comprising a signature and a class of interpretations. The signature provides the syntactic resources from which descriptions are constructed, and each interpretation provides a universe of objects and assigns the syntactic resources a meaning from which a denotation function is generated.

1 Definition. *Σ is a signature iff*

Σ is a triple $\langle \mathfrak{S}, \mathfrak{A}, \mathfrak{F} \rangle$,
$\mathfrak{S}$ is a finite set,
$\mathfrak{A}$ is a set, and
$\mathfrak{F}$ is a total function from the Cartesian product of $\mathfrak{S}$ and $\mathfrak{A}$ to the powerset of $\mathfrak{S}$.

Strictly, all subsequent work should be done with respect to explicit signatures. However, for notational facility, I work with respect to an implicit signature $\langle \mathfrak{S}, \mathfrak{A}, \mathfrak{F} \rangle$. I call each member of $\mathfrak{S}$ a species, each member of $\mathfrak{A}$ an attribute, and $\mathfrak{F}$ the appropriateness function. I assume that none of the symbols $:$, $\sim$, $\approx$, $\neg$, $[$, $]$, $\wedge$, $\vee$ or $\rightarrow$ is a species or an attribute.

2 Definition. *I is an interpretation iff*

I is a triple $\langle U, S, A \rangle$,
U is a set,
S is a total function from U to $\mathfrak{S}$,
A is a total function from $\mathfrak{A}$ to the set of partial functions from U to U,
for each $\alpha \in \mathfrak{A}$ and each $u \in U$,
if $A(\alpha)(u)$ is defined then $S(A(\alpha)(u)) \in \mathfrak{F}\langle S(u), \alpha \rangle$, and
for each $\alpha \in \mathfrak{A}$ and each $u \in U$,
if $\mathfrak{F}\langle S(u), \alpha \rangle$ is not empty then $A(\alpha)(u)$ is defined.

Suppose that I is an interpretation, and $I = \langle U, S, A \rangle$. I call each member of U an object in I, S a species assignment function, and A an attribute interpretation function. Each species is a symbol that names one of a finite number of categories that partition U, and S assigns each object in I the name of the unique category to which it belongs. Each attribute is a symbol that names one of an arbitrary number of partial functions from U to U, and A assigns each attribute the partial function it names. The appropriateness function encodes, and the last two conditions in the definition of an interpretation enforce, a relationship between the interpretations of species and attributes: if $\mathfrak{F}\langle \sigma, \alpha \rangle$ is empty then the partial function that attribute α names acts upon no object in the category that species σ names, and if $\mathfrak{F}\langle \sigma, \alpha \rangle$ is not empty then the partial function that attribute α names acts upon each object in the category that species σ names to yield an object in a category named by one of the species in $\mathfrak{F}\langle \sigma, \alpha \rangle$.

A term is a $:$ symbol followed by a finite string of attributes, and each interpretation generates a term interpretation function that assigns each term a unary partial function in the interpretation.

3 Definition. *τ is a term iff τ is a member of the smallest set $\mathfrak{T}$ such that*

$: \in \mathfrak{T}$, and
for each $\alpha \in \mathfrak{A}$ and each $\tau \in \mathfrak{T}$, $\tau\alpha \in \mathfrak{T}$.

I write $\mathfrak{T}$ for the set of terms.

4 Definition. *T is a term interpretation function over I iff*

I is an interpretation $\langle U, S, A\rangle$,

T is a total function from $\mathfrak{T}$ to the set of partial functions from U to U,

$T(:)$ is the identity function from U to U, and

for each $\alpha \in \mathfrak{A}$ and each $\tau \in \mathfrak{T}$,

$T(\tau\alpha)$ is the functional composition of $T(\tau)$ and $A(\alpha)$.

For each interpretation I, there is a unique term interpretation function over I. I write T_I for the term interpretation function over I. Suppose that I is an interpretation, and $I = \langle U, S, A\rangle$. $T_I(:\alpha_1 \ldots \alpha_n)$ is the functional composition of the unary partial functions in interpretation I named by attributes $\alpha_1, \ldots, \alpha_n$.

A description is a finite, well-formed string of species, terms and other symbols, and each interpretation generates a description denotation function that assigns each description a set of objects in the interpretation.

5 Definition. *δ is a description iff δ is a member of the smallest set $\mathfrak{D}$ such that*

for each $\sigma \in \mathfrak{S}$ and each $\tau \in \mathfrak{T}$, $\tau{\sim}\sigma \in \mathfrak{D}$,

for each $\tau_1 \in \mathfrak{T}$ and each $\tau_2 \in \mathfrak{T}$, $\tau_1{\approx}\tau_2 \in \mathfrak{D}$,

for each $\delta \in \mathfrak{D}$, $\neg\delta \in \mathfrak{D}$,

for each $\delta_1 \in \mathfrak{D}$ and each $\delta_2 \in \mathfrak{D}$, $[\delta_1 \wedge \delta_2] \in \mathfrak{D}$,

for each $\delta_1 \in \mathfrak{D}$ and each $\delta_2 \in \mathfrak{D}$, $[\delta_1 \vee \delta_2] \in \mathfrak{D}$, and

for each $\delta_1 \in \mathfrak{D}$ and each $\delta_2 \in \mathfrak{D}$, $[\delta_1 \rightarrow \delta_2] \in \mathfrak{D}$.

I write $\mathfrak{D}$ for the set of descriptions. I call descriptions of the form $\tau{\sim}\sigma$ and $\tau_1{\approx}\tau_2$ atomic. descriptions, and call descriptions of the form $\neg\delta$, $[\delta_1 \wedge \delta_2]$, $[\delta_1 \vee \delta_2]$ and $[\delta_1 \rightarrow \delta_2]$ compound descriptions.

6 Definition. *D is a description denotation function over I iff*

I is an interpretation $\langle U, S, A\rangle$,

D is a total function from $\mathfrak{D}$ to the powerset of U,

for each $\sigma \in \mathfrak{S}$ and each $\tau \in \mathfrak{T}$,

$$D(\tau{\sim}\sigma) = \left\{ u \in U \;\middle|\; \begin{array}{l} T_I(\tau)(u) \textit{ is defined, and} \\ S(T_I(\tau)(u)) = \sigma \end{array} \right\},$$

for each $\tau_1 \in \mathfrak{T}$ and each $\tau_2 \in \mathfrak{T}$,

$$D(\tau_1{\approx}\tau_2) = \left\{ u \in U \;\middle|\; \begin{array}{l} T_I(\tau_1)(u) \textit{ is defined,} \\ T_I(\tau_2)(u) \textit{ is defined, and} \\ T_I(\tau_1)(u) = T_I(\tau_2)(u) \end{array} \right\},$$

for each $d \in \mathfrak{D}$, $D(\neg d) = U \setminus D(d)$,

for each $\delta_1 \in \mathfrak{D}$ and each $\delta_2 \in \mathfrak{D}$, $D([\delta_1 \wedge \delta_2]) = D(\delta_1) \cap D(\delta_2)$,

for each $\delta_1 \in \mathfrak{D}$ and each $\delta_2 \in \mathfrak{D}$, $D([\delta_1 \vee \delta_2]) = D(\delta_1) \cup D(\delta_2)$, and

for each $\delta_1 \in \mathfrak{D}$ and each $\delta_2 \in \mathfrak{D}$, $D([\delta_1 \rightarrow \delta_2]) = (U \setminus D(\delta_1)) \cup D(\delta_2)$.

For each interpretation I, there is a unique description denotation function over I. I write D_I for the description denotation function over I. Suppose that I is an interpretation, and $I = \langle U, S, A\rangle$. Informally, $D_I(\delta)$ is the set of those objects in interpretation I of which δ is true. Thus, $D_I(\tau{\sim}\sigma)$ is the set of those objects u in interpretation I of which

partial function $T_I(\tau)$ acts upon object u to yield object u', and

object u' is in the category named by species σ;

$D_I(\tau_1{\approx}\tau_2)$ is the set of those objects u in interpretation I of which

partial function $T_I(\tau_1)$ acts upon object u to yield object u_1,

partial function $T_I(\tau_2)$ acts upon object u to yield object u_2, and

object u_1 and object u_2 are the same object;

$D_I(\neg\delta)$ is the set of those objects in interpretation I of which description δ is false;

$D_I([\delta_1 \wedge \delta_2])$ is the set of those objects in interpretation I of which description δ_1 is true and description δ_2 is true; $D_I([\delta_1 \vee \delta_2])$ is the set of those objects in interpretation I of which description δ_1 is true or description δ_2 is true; and $D_I([\delta_1 \rightarrow \delta_2])$ is the set of those objects in interpretation I of which description δ_2 is true whenever description δ_1 is true.

For notational facility, I occasionally write

$\top$ for description $:\approx:$,
$\bot$ for description $\neg\top$,
$\bigvee\langle\rangle$ for description $\bot$,
$\bigvee\langle\delta\rangle$ for description δ, and
$\bigvee\langle\delta_0,\ldots,\delta_n,\delta\rangle$ for description $[\bigvee\langle\delta_0,\ldots,\delta_n\rangle \vee \delta]$.

Abbreviation $\top$ is true of each object in each interpretation, abbreviation $\bot$ is false of each object in each interpretation, and abbreviation $\bigvee\langle\delta_1,\ldots,\delta_n\rangle$ is the left associative disjunction of descriptions δ_1, …, δ_n.

A theory is a set of descriptions, and each interpretation builds from its description denotation function a theory denotation function that assigns each theory a set of objects in the interpretation.

7 Definition. *θ is a theory iff $\theta \subseteq \mathfrak{D}$.*

8 Definition. *Θ is a theory denotation function over I iff*

I is an interpretation,
Θ is a total function from the powerset of $\mathfrak{D}$ to the powerset of U, and
for each $\theta \subseteq \mathfrak{D}$, $\Theta(\theta) = \bigcap\{D_I(\delta) \mid \delta \in \theta\}$.

For each interpretation I, there is a unique theory denotation function over I. I write Θ_I for the theory denotation function over I. Suppose that I is an interpretation. $\Theta_I(\theta)$ is the set of those objects in interpretation I of which each description in θ is true.

[King 1989] furnished each SRL formal language with a Tarski-style entailment relation and a Hilbert and Ackermann-style deduction relation, and then used standard predicate logic techniques to show that deduction is sound and complete with respect to entailment. However, the K-formalism does not need entailment and deduction, and I do not reproduce this work here.

2.2 The formalism

Since a token is a concrete particular, a token might seem to be an object. However, imagine a type T that abstracts all the tokens that have the same surname as their father, and imagine a stepson who has taken the surname of his stepfather. A lawyer investigating the legal guardianship of the stepson might reasonably consider the stepson's father to be the stepson's stepfather, and hence the lawyer might reasonably consider type T to abstract the stepson. However, a doctor investigating the susceptibility of the stepson to a hereditary disorder might equally reasonably consider the stepson's father to be the stepson's genetic father, and hence the doctor might equally reasonably consider type T to *not* abstract the stepson. Thus, if a token were an object then the stepson would be a token that type T both abstracts and does not abstract. Clearly, a token is not an object. Rather, various objects (the stepson, the stepfather and the genetic father, for example) stand in various relations (fatherhood, for example), and a token must comprise at least an object and a record of the various relations in which the object stands. In SRL, an interpretation supplies such a record. Thus, a token is not an object in isolation, but an object in an interpretation.

9 Definition. *t is a token iff*

t is an ordered pair $\langle u, I\rangle$,
I is an interpretation $\langle U, S, A\rangle$, and
$u \in U$.

The tokens may constitute a proper class.

A type is an abstract universal that signifies a collection of tokens, and a type abstracts a token iff the type signifies a collection containing the token. Most simply, a type is a collection of tokens that signifies itself, and a type abstracts a token iff the token is in the type. Thus, I need only define what it is for two tokens to be of the same type in order to define types and type abstraction. Two tokens are indiscernible iff no description can tell them apart.

10 Definition. *t_1 and t_2 are indiscernible iff*
t_1 is a token $\langle u_1, I_1 \rangle$,
t_2 is a token $\langle u_2, I_2 \rangle$, and
for each $\delta \in \mathfrak{D}$,
$u_1 \in D_{I_1}(\delta)$ *iff* $u_2 \in D_{I_2}(\delta)$.

Two tokens are of the same type iff the tokens are indiscernible. Since indiscernibility is an equivalence relation over the class of tokens, a type is an equivalence class under indiscernibility.

11 Definition. *T is a type iff*
T is a class of tokens, and
for some t in T, for each token t',
t' is in T iff t and t' are indiscernible.

A type may be a proper class. A type abstracts a token iff the token is in the type.

12 Definition. *T is a type abstraction of t iff*
T is a type,
t is a token, and
t is in T.

Each token has a unique type abstraction. Also, two tokens are indiscernible iff the tokens have identical type abstractions.

13 Proposition. *For each token t_1 and each token t_2,*
t_1 and t_2 are indiscernible iff the type abstraction of t_1 is the type abstraction of t_2.

One might think that indiscernibility is too weak to capture the notion 'of the same type', and that some structural isomorphism between tokens would be a more suitable formalisation. For example, the components of a token $\langle u, I \rangle$ are the objects accessible in interpretation I from object u.

14 Definition. *u is a component of t iff*
t is a token $\langle u', I \rangle$,
I is an interpretation $\langle U, S, A \rangle$, and
for some $\tau \in \mathfrak{T}$,
$T_I(\tau)(u')$ *is defined and* $T_I(\tau)(u') = u$.

I write C_t for the set of components of t. Two tokens are congruent iff there is a structure and species preserving bijection between their components.

15 Definition. *f is a congruence from t_1 to t_2 iff*
t_1 is a token $\langle u_1, I_1 \rangle$,
t_2 is a token $\langle u_2, I_2 \rangle$,
I_1 is an interpretation $\langle U_1, S_1, A_1 \rangle$,
I_2 is an interpretation $\langle U_2, S_2, A_2 \rangle$,
f is a bijection from C_{t_1} to C_{t_2},
for each $\tau \in \mathfrak{T}$ and each $u \in C_{t_1}$,
$T_{I_1}(\tau)(u_1)$ *is defined and* $T_{I_1}(\tau)(u_1) = u$
iff $T_{I_2}(\tau)(u_2)$ *is defined and* $T_{I_2}(\tau)(u_2) = f(u)$, *and*
for each $u \in C_{t_1}$, $S_1(u) = S_2(f(u))$.

16 Definition. *t_1 and t_2 are congruent iff for some f, f is a congruence from t_1 to t_2.*
One might consider two tokens to be of the same type iff the tokens are congruent. Since congruence is an equivalence relation over the class of tokens, one might therefore consider a type to be an equivalence class under congruence. However, the 'weak' and 'strong' definitions of a type are actually equivalent.

17 Proposition. *For each token t_1 and each token t_2,*

t_1 and t_2 are indiscernible iff t_1 and t_2 are congruent.

Proof. For each token $\langle u_1, I_1\rangle$ and each token $\langle u_2, I_2\rangle$,

$\langle u_1, I_1\rangle$ and $\langle u_2, I_2\rangle$ are indiscernible

$$\Longrightarrow \left\{ \langle u'_1, u'_2\rangle \;\middle|\; \begin{array}{l} \text{for some } \tau \in \mathfrak{T}, \\ \quad T_{I_1}(\tau)(u_1) \text{ is defined,} \\ \quad T_{I_1}(\tau)(u_1) = u'_1, \\ \quad T_{I_2}(\tau)(u_2) \text{ is defined, and} \\ \quad T_{I_2}(\tau)(u_2) = u'_2 \end{array} \right\} \text{ is a congruence from } \langle u_1, I_1\rangle \text{ to } \langle u_2, I_2\rangle$$

$\Longrightarrow$ $\langle u_1, I_1\rangle$ and $\langle u_2, I_2\rangle$ are congruent,

and

$\langle u_1, I_1\rangle$ and $\langle u_2, I_2\rangle$ are congruent

$\Longrightarrow$ for some f, f is a congruence from $\langle u_1, I_1\rangle$ to $\langle u_2, I_2\rangle$

$\Longrightarrow$ for each $\delta \in \mathfrak{D}$,

$u_1 \in D_{I_1}(\delta)$ iff $u_2 \in D_{I_2}(\delta)$ ... by induction on the complexity of δ

$\Longrightarrow$ $\langle u_1, I_1\rangle$ and $\langle u_2, I_2\rangle$ are indiscernible.

■

[Pollard and Sag 1994] tacitly assumes that type admission is recursive. For example, [Pollard and Sag 1994, pages 33–34] uses the English sentence 'Kim gives Sandy Fido' to illustrate two universal principles, the head feature principle and the subcategorization principle, and it is evident throughout the illustration that the principles admit 'Kim gives Sandy Fido' iff they admit each constituent of 'Kim gives Sandy Fido'. No type has a structure over which to define admission recursively, but the components provide each token with just such a structure. Consequently, I first define an admission relation between the theories and the tokens such that token admission is recursive over the components of a token, and then naturally abstract the admission relation from tokens to types.

A theory admits a token iff the theory is true of each component of the token.

18 Definition. *θ token admits t iff*

$\theta \subseteq \mathfrak{D}$,

t is a token $\langle u, I\rangle$, and

$C_t \subseteq \Theta_I(\theta)$.

Notice two things. Firstly, token admission is recursive over the components of a token.

19 Lemma. *For each object u' and each token $\langle u, I\rangle$,*

if $u' \in C_{\langle u,I\rangle}$ then $C_{\langle u',I\rangle} \subseteq C_{\langle u,I\rangle}$.

20 Proposition. *For each $\theta \subseteq \mathfrak{D}$ and each token $\langle u, I\rangle$,*

θ token admits $\langle u, I\rangle$ iff for each $u' \in C_{\langle u,I\rangle}$, θ token admits $\langle u', I\rangle$.

Secondly, token admission cannot distinguish between indiscernible tokens.

21 Lemma. *For each object u, each token $\langle u_1, I_1\rangle$, each token $\langle u_2, I_2\rangle$ and each f,*

if f is a congruence from $\langle u_1, I_1\rangle$ to $\langle u_2, I_2\rangle$ and $u \in C_{\langle u_1,I_1\rangle}$

then the restriction of f to $C_{\langle u,I_1\rangle}$ is a congruence from $\langle u, I_1\rangle$ to $\langle f(u), I_2\rangle$.

22 Proposition. *For each $\theta \subseteq \mathfrak{D}$, each token t_1 and each token t_2,*

if t_1 and t_2 are indiscernible and θ token admits t_1

then θ token admits t_2.

Proof. For each $\theta \subseteq \mathfrak{D}$, each token $\langle u_1, I_1\rangle$ and each token $\langle u_2, I_2\rangle$,

$\langle u_1, I_1\rangle$ and $\langle u_2, I_2\rangle$ are indiscernible and θ token admits $\langle u_1, I_1\rangle$

$\Longrightarrow$ for some congruence f from $\langle u_1, I_1\rangle$ to $\langle u_2, I_2\rangle$, ...by proposition 17

for each object u_2',

$u_2' \in C_{\langle u_2, I_2\rangle}$

$\Longrightarrow$ for some $u_1' \in C_{\langle u_1, I_1\rangle}$, $f(u_1') = u_2'$

$\Longrightarrow$ for some $u_1' \in \Theta_{I_1}(\theta)$, the restriction of f to $C_{\langle u_1', I_1\rangle}$ is a congruence from $\langle u_1', I_1\rangle$ to $\langle u_2', I_2\rangle$ ...by lemma 21

$\Longrightarrow$ for some $u_1' \in \Theta_{I_1}(\theta)$, $\langle u_1', I_1\rangle$ and $\langle u_2', I_2\rangle$ are indiscernible ...by proposition 17

$\Longrightarrow$ $u_2' \in \Theta_{I_2}(\theta)$

$\Longrightarrow$ θ token admits $\langle u_2, I_2\rangle$.

■

By propositions 13 and 22, I can felicitously abstract admission from tokens to types.

23 Definition. *θ type admits T iff*

$\theta \subseteq \mathfrak{D}$,

T is a type, and

for some (or equivalently, each) t in T, θ token admits t.

3 A mathematical model of the K-formalism

In this section I build a mathematical model of the K-formalism that facilitates subsequent proofs that the K-formalism meets all [Pollard and Sag 1994] demands of the envisaged HPSG formalism. I construct mathematical entities I call morphs, and define what it is for a morph to abstract a token, and what it is for a theory to admit a morph. I then define a one-to-one correspondence between the types and the morphs, and show that a type abstracts a token iff the corresponding morph abstracts the token, and a theory admits a type iff the theory admits the corresponding morph.

3.1 The morphs

[Nerode 1958] gave an elegant representation of an automaton[3] that abstracts away from the states of the automaton. Pair $\langle \Delta, \Gamma\rangle$ is the Nerode representation of automaton A over input alphabet I iff

$$\Delta = \left\{ w \,\middle|\, \begin{array}{l} w \text{ is a finite string of symbols from } I \text{, and} \\ A \text{ transits from its initial state on input } w \end{array} \right\}, \text{ and}$$

$$\Gamma = \left\{ \langle w_1, w_2\rangle \,\middle|\, \begin{array}{l} w_1 \text{ is a finite string of symbols from } I, \\ w_2 \text{ is a finite string of symbols from } I, \\ A \text{ transits from its initial state to some state } q_1 \text{ on input } w_1, \\ A \text{ transits from its initial state to some state } q_2 \text{ on input } w_2 \text{, and} \\ q_1 = q_2 \end{array} \right\}.$$

Each automaton has a unique Nerode representation. Further, pair $\langle \Delta, \Gamma\rangle$ is the Nerode representation of some automaton over input alphabet I iff

Δ is a nonempty set of finite strings of symbols from I,

for each symbol i from I and each finite string w of symbols from I,

if $wi \in \Delta$ then $w \in \Delta$,

[3] [Hopcroft and Ullman 1979] covers much of the automata theory I use in this paper. Notice, I assume that each automaton is rooted and deterministic, but I impose no finiteness restrictions on the states, transitions or alphabets of automata.

Γ is an equivalence relation over Δ, and

for each symbol i from I, each finite string w_1 of symbols from I and each finite string w_2 of symbols from I,

if $w_1 i \in \Delta$ and $\langle w_1, w_2\rangle \in \Gamma$ then $\langle w_1 i, w_2 i\rangle \in \Gamma$.

[Moore 1956] allowed an automaton to output symbols. A Moore automaton over input alphabet I and output alphabet O is an automaton over input alphabet I that outputs a symbol from O at each state. [Moshier 1988, section 3.1.3] extended Nerode representation to give an equally elegant representation of a Moore automaton. Triple $\langle \Delta, \Gamma, \Lambda\rangle$ is the Moshier representation of Moore automaton A over input alphabet I and output alphabet O iff

$\langle \Delta, \Gamma\rangle$ is the Nerode representation of A, and

$$\Lambda = \left\{ \langle w, o\rangle \;\middle|\; \begin{array}{l} w \text{ is a finite string of symbols from } I, \\ o \text{ is a symbol from } O, \\ A \text{ transits from its initial state to some state } q \text{ on input } w, \text{ and} \\ A \text{ outputs } o \text{ at state } q \end{array} \right\}.$$

Each Moore automaton has a unique Moshier representation. Further, triple $\langle \Delta, \Gamma, \Lambda\rangle$ is the Moshier representation of some Moore automaton over input alphabet I and output alphabet O iff

Δ is a nonempty set of finite strings of symbols from I,

for each symbol i from I and each finite string w of symbols from I,

if $wi \in \Delta$ then $w \in \Delta$,

Γ is an equivalence relation over Δ,

for each symbol i from I, each finite string w_1 of symbols from I and each finite string w_2 of symbols from I,

if $w_1 i \in \Delta$ and $\langle w_1, w_2\rangle \in \Gamma$ then $\langle w_1 i, w_2 i\rangle \in \Gamma$,

Λ is a total function from Δ to O, and

for each finite string w_1 of symbols from I and each finite string w_2 of symbols from I,

if $\langle w_1, w_2\rangle \in \Gamma$ then $\Lambda(w_1) = \Lambda(w_2)$.

I say that A is a special automaton iff

A is a Moore automaton over input alphabet $\mathfrak{A}$ and output alphabet $\mathfrak{S}$,

for each $\sigma_1 \in \mathfrak{S}$, each $\sigma_2 \in \mathfrak{S}$, each $\alpha \in \mathfrak{A}$, each state q_1 of A and each state q_2 of A,

if A transits from q_1 to q_2 on input α,

A outputs σ_1 at q_1, and

A outputs σ_2 at q_2

then $\sigma_2 \in \mathfrak{F}\langle \sigma_1, \alpha\rangle$, and

for each $\sigma \in \mathfrak{S}$, each $\alpha \in \mathfrak{A}$ and each state q of A,

if A outputs σ at q, and

$\mathfrak{F}\langle \sigma, \alpha\rangle$ is not empty

then A transits from q on input α.

A morph is the Moshier representation of a special automaton.

24 Definition. *ω is a word iff*

ω is a finite string of attributes.

I write $\mathfrak{W}$ for the set of words.

25 Definition. *M is a morph iff*

M is a triple $\langle \Delta, \Gamma, \Lambda\rangle$,

Δ is a nonempty subset of $\mathfrak{W}$,

for each $\alpha \in \mathfrak{A}$ and each $\omega \in \mathfrak{W}$,

if $\omega\alpha \in \Delta$ then $\omega \in \Delta$,

Γ is an equivalence relation over Δ,
for each $\alpha \in \mathfrak{A}$, each $\omega_1 \in \mathfrak{W}$ and each $\omega_2 \in \mathfrak{W}$,
if $\omega_1\alpha \in \Delta$ and $\langle \omega_1, \omega_2 \rangle \in \Gamma$ then $\langle \omega_1\alpha, \omega_2\alpha \rangle \in \Gamma$,
Λ is a total function from Δ to $\mathfrak{S}$,
for each $\omega_1 \in \mathfrak{W}$ and each $\omega_2 \in \mathfrak{W}$,
if $\langle \omega_1, \omega_2 \rangle \in \Gamma$ then $\Lambda(\omega_1) = \Lambda(\omega_2)$,
for each $\alpha \in \mathfrak{A}$ and each $\omega \in \mathfrak{W}$,
if $\omega\alpha \in \Delta$ then $\Lambda(\omega\alpha) \in \mathfrak{F}\langle \Lambda(\omega), \alpha \rangle$, and
for each $\alpha \in \mathfrak{A}$ and each $\omega \in \mathfrak{W}$,
if $\omega \in \Delta$ and $\mathfrak{F}\langle \Lambda(\omega), \alpha \rangle$ is not empty then $\omega\alpha \in \Delta$.

I write $\mathfrak{M}$ for the set of morphs.

If morph M is the Moshier representation of special automaton A, and A transits from its initial state to state q on input word ω then the ω submorph of M is the Moshier representation of the special subautomaton of A that has initial state q.

26 Definition. *M' is an ω submorph of M iff*
M' is a morph $\langle \Delta', \Gamma', \Lambda' \rangle$
M is a morph $\langle \Delta, \Gamma, \Lambda \rangle$,
$\omega \in \Delta$,
$\Delta' = \left\{ \omega' \,\middle|\, \omega\omega' \in \Delta \right\}$,
$\Gamma' = \left\{ \langle \omega_1, \omega_2 \rangle \,\middle|\, \langle \omega\omega_1, \omega\omega_2 \rangle \in \Gamma \right\}$, and
$\Lambda' = \left\{ \langle \omega', \sigma \rangle \,\middle|\, \langle \omega\omega', \sigma \rangle \in \Lambda \right\}$.

For each $\langle \Delta, \Gamma, \Lambda \rangle \in \mathfrak{M}$ and each $\omega \in \Delta$, there is a unique ω submorph of $\langle \Delta, \Gamma, \Lambda \rangle$.

A morph abstracts a token iff the morph represents exactly the atomic descriptions that are true of the token.

27 Definition. *M is a morph abstraction of t iff*
M is a morph $\langle \Delta, \Gamma, \Lambda \rangle$,
t is a token $\langle u, I \rangle$,
for each $\omega_1 \in \mathfrak{W}$ and each $\omega_2 \in \mathfrak{W}$,
$\langle \omega_1, \omega_2 \rangle \in \Gamma$ iff $u \in D_I(:\omega_1 \approx :\omega_2)$, and
for each $\omega \in \mathfrak{W}$ and each $\sigma \in \mathfrak{S}$,
$\langle \omega, \sigma \rangle \in \Lambda$ iff $u \in D_I(:\omega \sim \sigma)$.

Each token has a unique morph abstraction. Also, two tokens are indiscernible iff the tokens have identical morph abstractions.

28 Proposition. *For each token t_1 and each token t_2,*
t_1 and t_2 are indiscernible
iff the morph abstraction of t_1 is the morph abstraction of t_2.

Proof. For each token $\langle u_1, I_1 \rangle$ and each token $\langle u_2, I_2 \rangle$,
$\langle u_1, I_1 \rangle$ and $\langle u_2, I_2 \rangle$ are indiscernible
$\Longrightarrow$ the morph abstraction of $\langle u_1, I_1 \rangle$ is the morph abstraction of $\langle u_2, I_2 \rangle$
and
the morph abstraction of $\langle u_1, I_1 \rangle$ is the morph abstraction of $\langle u_2, I_2 \rangle$
$\Longrightarrow$ for each $\delta \in \mathfrak{D}$,
$u_1 \in D_{I_1}(\delta)$ iff $u_2 \in D_{I_2}(\delta)$ ... by induction on the complexity of δ
$\Longrightarrow$ $\langle u_1, I_1 \rangle$ and $\langle u_2, I_2 \rangle$ are indiscernible.
■

A description approximates a morph iff the description and the set of atomic descriptions that the morph represents are consistent.

29 Definition. *δ approximates M iff $\langle \delta, M\rangle$ is a member of the smallest set A such that*
for each $\sigma \in \mathfrak{S}$, each $\omega \in \mathfrak{W}$ and each morph $\langle \Delta, \Gamma, \Lambda\rangle$,
if $\langle \omega, \sigma\rangle \in \Lambda$ then $\langle :\omega{\sim}\sigma, \langle \Delta, \Gamma, \Lambda\rangle\rangle \in A$,
for each $\omega_1 \in \mathfrak{W}$, each $\omega_2 \in \mathfrak{W}$ and each morph $\langle \Delta, \Gamma, \Lambda\rangle$,
if $\langle \omega_1, \omega_2\rangle \in \Gamma$ then $\langle :\omega_1{\approx}:\omega_2, \langle \Delta, \Gamma, \Lambda\rangle\rangle \in A$,
for each $\delta \in \mathfrak{D}$ and each morph M,
if $\langle \delta, M\rangle \notin A$ then $\langle \neg\delta, M\rangle \in A$,
for each $\delta_1 \in \mathfrak{D}$, each $\delta_2 \in \mathfrak{D}$ and each morph M,
if $\langle \delta_1, M\rangle \in A$ and $\langle \delta_2, M\rangle \in A$ then $\langle [\delta_1 \wedge \delta_2], M\rangle \in A$,
for each $\delta_1 \in \mathfrak{D}$, each $\delta_2 \in \mathfrak{D}$ and each morph M,
if $\langle \delta_1, M\rangle \in A$ or $\langle \delta_2, M\rangle \in A$ then $\langle [\delta_1 \vee \delta_2], M\rangle \in A$, and
for each $\delta_1 \in \mathfrak{D}$, each $\delta_2 \in \mathfrak{D}$ and each morph M,
if $\langle \delta_1, M\rangle \notin A$ or $\langle \delta_2, M\rangle \in A$ then $\langle [\delta_1 \rightarrow \delta_2], M\rangle \in A$.

A theory admits a morph iff every description in the theory approximates every submorph of the morph.

30 Definition. *θ morph admits M iff*
$\theta \subseteq \mathfrak{D}$,
M is a morph $\langle \Delta, \Gamma, \Lambda\rangle$, and
for each $\delta \in \theta$ and each $\omega \in \Delta$, δ approximates the ω submorph of M.

Notice that morph admission is recursive over the submorphs of a morph.

31 Proposition. *For each $\theta \subseteq \mathfrak{D}$ and each $\langle \Delta, \Gamma, \Lambda\rangle \in \mathfrak{M}$,*
θ morph admits $\langle \Delta, \Gamma, \Lambda\rangle$
iff for each $\omega \in \Delta$, θ morph admits the ω submorph of $\langle \Delta, \Gamma, \Lambda\rangle$.

3.2 A correspondence

A morph is an image of a type iff the morph and the type abstract a common token.

32 Definition. *M is an image of T iff*
$M \in \mathfrak{M}$,
T is a type, and
for some t in T, M is the morph abstraction of t.

Image is functional and injective by propositions 13 and 28, but proving that image is surjective requires proving that morph abstraction is surjective. To this end, I construct mathematical entities I call standard tokens, and show that each morph abstracts some standard token.

I say that pair $\langle A, q\rangle$ is a focused automaton over input alphabet I iff
A is an automaton over input alphabet I, and
q is a state of A.

Nerode representation easily extends to focused automata. Triple $\langle \Delta, \Gamma, \mathrm{E}\rangle$ is the Nerode representation of focused automaton $\langle A, q\rangle$ over input alphabet I iff
$\langle \Delta, \Gamma\rangle$ is the Nerode representation of A, and

$$\mathrm{E} = \left\{ w \;\middle|\; \begin{array}{l} w \text{ is a finite string of symbols from } I \text{, and} \\ A \text{ transits from its initial state to } q \text{ on input } w \end{array} \right\}.$$

Each focused automaton has a unique Nerode representation. Further, triple $\langle \Delta, \Gamma, \mathrm{E}\rangle$ is the Nerode representation of some focused automaton over input alphabet I iff
Δ is a nonempty set of finite strings of symbols from I,
for each symbol i from I and each finite string w of symbols from I,
if $wi \in \Delta$ then $w \in \Delta$,

Γ is an equivalence relation over Δ,
for each symbol i from I, each finite string w_1 of symbols from I and each finite string w_2 of symbols from I,
if $w_1 i \in \Delta$ and $\langle w_1, w_2 \rangle \in \Gamma$ then $\langle w_1 i, w_2 i \rangle \in \Gamma$, and
E is an equivalence class under Γ.

One can similarly speak of a focused Moore automaton, and extend Moshier representation to focused Moore automata. A standard object is the Moshier representation of a focused special automaton.

33 Definition. *u is a standard object iff*
u is a quadruple $\langle \Delta, \Gamma, \Lambda, \mathrm{E} \rangle$,
$\langle \Delta, \Gamma, \Lambda \rangle$ is a morph, and
E is an equivalence class under Γ.

I write $\widetilde{U}$ for the set of standard objects. If focused special automaton $\langle A, q \rangle$ outputs species σ at q then the standard species assignment function assigns σ to the Moshier representation of $\langle A, q \rangle$.

34 Definition. *f is a standard species assignment function iff*
f is a total function from $\widetilde{U}$ to $\mathfrak{S}$, and
for each $\sigma \in \mathfrak{S}$ and each standard object $\langle \Delta, \Gamma, \Lambda, \mathrm{E} \rangle$,
$f\langle \Delta, \Gamma, \Lambda, \mathrm{E} \rangle = \sigma$ iff for some (or equivalently, each) $\omega \in \mathrm{E}$, $\Lambda(\omega) = \sigma$.

There is a unique standard species assignment function. I write $\widetilde{S}$ for the standard species assignment function. If focused special automaton $\langle A, q_1 \rangle$ transits from q_1 to state q_2 on input attribute α then the standard attribute interpretation function assigns α a partial function that maps the Moshier representation of $\langle A, q_1 \rangle$ to the Moshier representation of $\langle A, q_2 \rangle$.

35 Definition. *f is a standard attribute interpretation function iff*
f is a total function from $\mathfrak{A}$ to the set of partial functions from $\widetilde{U}$ to $\widetilde{U}$, and
for each $\alpha \in \mathfrak{A}$, each $\langle \Delta, \Gamma, \Lambda, \mathrm{E} \rangle \in \widetilde{U}$ and each $\langle \Delta', \Gamma', \Lambda', \mathrm{E}' \rangle \in \widetilde{U}$,
$f(\alpha)\langle \Delta, \Gamma, \Lambda, \mathrm{E} \rangle$ is defined and $f(\alpha)\langle \Delta, \Gamma, \Lambda, \mathrm{E} \rangle = \langle \Delta', \Gamma', \Lambda', \mathrm{E}' \rangle$
iff $\langle \Delta, \Gamma, \Lambda \rangle = \langle \Delta', \Gamma', \Lambda' \rangle$ and for some $\omega \in \mathrm{E}$, $\omega\alpha \in \mathrm{E}'$.

There is a unique standard attribute interpretation function. I write $\widetilde{A}$ for the standard attribute interpretation function.

36 Proposition. *$\langle \widetilde{U}, \widetilde{S}, \widetilde{A} \rangle$ is an interpretation.*

37 Definition. *I is the standard interpretation iff I is the interpretation $\langle \widetilde{U}, \widetilde{S}, \widetilde{A} \rangle$.*

I write $\widetilde{I}$ for the standard interpretation.

38 Definition. *t is a standard token iff t is a token $\langle u, \widetilde{I} \rangle$.*

Each morph is the morph abstraction of some standard token.

39 Lemma. *For each $\omega \in \mathfrak{W}$, each $\langle \Delta, \Gamma, \Lambda, \mathrm{E} \rangle \in \widetilde{U}$ and each $\langle \Delta', \Gamma', \Lambda', \mathrm{E}' \rangle \in \widetilde{U}$,*
$T_{\widetilde{I}}(:\omega)\langle \Delta, \Gamma, \Lambda, \mathrm{E} \rangle$ is defined and $T_{\widetilde{I}}(:\omega)\langle \Delta, \Gamma, \Lambda, \mathrm{E} \rangle = \langle \Delta', \Gamma', \Lambda', \mathrm{E}' \rangle$
iff $\langle \Delta, \Gamma, \Lambda \rangle = \langle \Delta', \Gamma', \Lambda' \rangle$ and for some $\omega' \in \mathrm{E}$, $\omega'\omega \in \mathrm{E}'$.

Proof. By induction on the length of ω. ∎

40 Lemma. *For each $\omega \in \mathfrak{W}$ and each $\langle \Delta, \Gamma, \Lambda, \mathrm{E} \rangle \in \widetilde{U}$,*
if $\omega \in \mathrm{E}$ then the ω submorph of $\langle \Delta, \Gamma, \Lambda \rangle$ is the morph abstraction of $\langle \langle \Delta, \Gamma, \Lambda, \mathrm{E} \rangle, \widetilde{I} \rangle$.

41 Lemma. *For each $\langle \Delta, \Gamma, \Lambda, \mathrm{E} \rangle \in \widetilde{U}$,*
if the empty word is in E then $\langle \Delta, \Gamma, \Lambda \rangle$ is the morph abstraction of $\langle \langle \Delta, \Gamma, \Lambda, \mathrm{E} \rangle, \widetilde{I} \rangle$.

42 Proposition. *For each $M \in \mathfrak{M}$, for some standard token t,*
M is the morph abstraction of t.

Thus, image is a one-to-one correspondence between the types and the morphs.

43 Proposition.

For each type T, for some unique M, M is an image of T.

For each $M \in \mathfrak{M}$, for some unique T, M is an image of T.

Notice a nice consequence of this proposition. A type may be a proper class, but the types constitute a 'set sized' class of classes!

Image provides a correlation between type abstraction and morph abstraction.

44 Proposition. *For each token t,*

the image of the type abstraction of t is the morph abstraction of t.

Proof. From propositions 13 and 28. ■

Image also provides a correlation between type admission and morph admission.

45 Lemma. *For each token $\langle u, I\rangle$ and each $\langle \Delta, \Gamma, \Lambda\rangle \in \mathfrak{M}$,*

if $\langle \Delta, \Gamma, \Lambda\rangle$ is the morph abstraction of $\langle u, I\rangle$

then for each $M \in \mathfrak{M}$ and each $\omega \in \mathfrak{W}$,

$T_I(:\omega)(u)$ is defined and M is the morph abstraction of $\langle T_I(:\omega)(u), I\rangle$

iff $\omega \in \Delta$ and M is the ω submorph of $\langle \Delta, \Gamma, \Lambda\rangle$.

46 Lemma. *For each $\delta \in \mathfrak{D}$ and each token $\langle u, I\rangle$,*

$u \in D_I(\delta)$ iff δ approximates the morph abstraction of $\langle u, I\rangle$.

Proof. By induction on the complexity of δ. ■

47 Lemma. *For each $\theta \subseteq \mathfrak{D}$ and each token t,*

θ token admits t iff θ morph admits the morph abstraction of t.

Proof. For each $\theta \subseteq \mathfrak{D}$, each token $\langle u, I\rangle$ and each $\langle \Delta, \Gamma, \Lambda\rangle \in \mathfrak{M}$,

if $\langle \Delta, \Gamma, \Lambda\rangle$ is the morph abstraction of $\langle u, I\rangle$

then θ token admits $\langle u, I\rangle$

$\iff$ for each $\delta \in \theta$ and each $u' \in C_{\langle u, I\rangle}$, $u' \in D_I(\delta)$

$\iff$ for each $\delta \in \theta$ and each $u' \in C_{\langle u, I\rangle}$,

δ approximates the morph abstraction of $\langle u', I\rangle$ . . . by lemma 46

$\iff$ for each $\delta \in \theta$ and each $\omega \in \Delta$,

δ approximates the ω submorph of $\langle \Delta, \Gamma, \Lambda\rangle$ . . . by lemma 45

$\iff$ θ morph admits $\langle \Delta, \Gamma, \Lambda\rangle$.

■

48 Proposition. *For each $\theta \subseteq \mathfrak{D}$ and each type T,*

θ type admits T iff θ morph admits the image of T.

Proof. From lemma 47 and propositions 13 and 28. ■

Thus, a type abstracts a token iff the image of the type abstracts the token, and a theory admits a type iff the theory admits the image of the type.

4 The appositeness of the K-formalism

In this section I prove that the K-formalism meets all [Pollard and Sag 1994] asks of the envisaged HPSG formalism. I extract an exact definition of a [Pollard and Sag 1994] feature structure, demonstrate a one-to-one correspondence between the morphs and the feature structures, and show that the K-formalism types correspond to the types [Pollard and Sag 1994] envisages. I then propose a plausible definition of [Pollard and Sag 1994] feature structure admission, demonstrate that a theory admits a morph iff the theory admits the corresponding feature structure, and show that if my definition of feature structure admission is true then K-formalism type admission corresponds to the type admission [Pollard and Sag 1994] envisages.

4.1 The feature structures

[Pollard and Sag 1994] does not define a feature structure, but references [Moshier 1988], [Pollard and Moshier 1990] and [Carpenter 1992] instead. These sources define several kinds of feature structure, and [Pollard and Sag 1994] favours no one particular kind. However, [Moshier 1988], [Pollard and Moshier 1990] and [Carpenter 1992] all agree that a feature structure is either a Moore automaton over input alphabet $\mathfrak{A}$ or an isomorphism class of such automata. Additionally, [Pollard and Sag 1994] remarks that

> the feature structures employed in HPSG are *sorted*. This means simply that each node is labelled with a *sort symbol* that tells what type of object the structure is modelling ...
>
> Second, we require feature structures to be *well-typed* (Carpenter 1992). This means that what attribute (or, equivalently, feature) labels can appear in a feature structure are determined by its sort; this fact is the reflection within the model of the fact that what attributes (or, equivalently, features or components of structure) an empirical object has depends on its ontological category. ... In addition, given a sort of feature structure and an attribute label appropriate for that sort, the value of that attribute must also be of a sort appropriate for the given sort and attribute label. ...
>
> Third, feature structures that serve as models of linguistic entities are required to satisfy further criteria of completeness. Roughly what this means is that they are total (not merely partial) models of the objects that they represent. More precisely, they are required to be both *totally well-typed* (Carpenter 1992) and *sort-resolved*. A feature structure is *totally* well-typed in case it is well-typed, and, moreover, for each node, every feature that is appropriate for the sort assigned to that node is actually present. ... A feature structure is *sort-resolved* provided every node is assigned a sort label that is maximal (i.e. most specific) in the sort ordering.
>
> [Pollard and Sag 1994, pages 17 and 18]

Of the sorts, [Pollard and Sag 1994] says that

> The (finite) set of all sort symbols is assumed to be partially ordered,
>
> [Pollard and Sag 1994, pages 17 and 18]

and adds that

> The sort hierarchy is presented as a taxonomic tree, with the root labelled *object* (the sort of all linguistic entities with which the grammar deals). For each local tree in the hierarchy, the sorts $\sigma_1, \ldots, \sigma_n$, which label the daughters, partition the sort σ, which labels the mother; that is, they are necessarily disjoint subsorts of σ that exhaust σ. For two sorts σ and τ, τ is a *subsort* of σ if and only if it is dominated by σ; sorts that label terminal nodes are called *maximal* (in the sense of maximally informative or maximally specific). A *feature declaration* of the form
>
> $$\sigma : \begin{bmatrix} \mathrm{F}_1 & \tau_1 \\ \ldots & \\ \mathrm{F}_n & \tau_n \end{bmatrix}$$
>
> where $\sigma, \tau_1, \ldots, \tau_n$ are sorts and $\mathrm{F}_1, \ldots, \mathrm{F}_n$ are feature labels, signifies that for each $i = 1, \ldots, n$, (1) the feature F_i is appropriate for all objects of sort σ, and (2) for any such object, the value of the F_i feature must be an object of sort τ_i.
>
> If sorts σ_1 and σ_2 bear declarations $\left[\mathrm{F}\,\tau_1\right]$ and $\left[\mathrm{F}\,\tau_2\right]$ for the same feature F and σ_2 is a subsort of of σ_1, then τ_2 must be a subsort of τ_1. A sort inherits the feature declarations of its supersorts; hence any feature that is defined for a given sort is defined for all of that sort's subsorts. By convention, the features that are declared for a maximal sort (including those inherited from its supersorts) are the only features defined for that sort. [Pollard and Sag 1994, page 395]

Thus, the sorts comprise a finite partial order, a sort is maximal iff the sort has no proper subsorts (sic), a sort is root iff the sort is a supersort of each sort, and the sort partial order has a root sort. Further, under each interpretation, each sort denotes a set of objects in the interpretation, the root sort denotes the set of all objects in the interpretation, the denotation of a sort is partitioned by the denotations of its maximal subsorts, and each attribute names a unary partial function from objects to objects in the interpretation that is defined on all or no objects in the denotation of each maximal sort. From the foregoing, [King 1989, pages 46–48] deduced a simple relationship between the sorts and the species:

Each maximal sort is a species.

Attribute α is appropriate for maximal sort σ iff $\mathfrak{F}\langle\sigma,\alpha\rangle$ is not empty.

If maximal sort σ bears feature declaration $\left[\alpha\,\tau\right]$ then $\mathfrak{F}\langle\sigma,\alpha\rangle$ is the set of maximal subsorts of τ.

A [Pollard and Sag 1994] feature structure is therefore either a special automaton or an isomorphism class of special automata, and [Pollard and Sag 1994]'s stated intention that the feature structures "stand in a one-to-one relation" with the types precludes the former. To prove this, suppose that a feature structure is a special automaton, and that A_1 and A_2 are distinct but isomorphic feature structures that model types T_1 and T_2 respectively. The feature structures stand in a one-to-one relation with the types, and A_1 and A_2 are distinct. Therefore, T_1 and T_2 are distinct. However, states play no conceivable role in the one-to-one relation between feature structures and types, and A_1 and A_2 are isomorphic. Therefore, T_1 and T_2 are identical. To avoid reductio ad absurdum, a feature structure must be an isomorphism class of special automata. Notice that a feature structure may be a proper class, since the special automata may constitute a proper class.

There is a simple correlation between the morphs and the [Pollard and Sag 1994] feature structures. I say that feature structure F is the twin of morph M iff M is the Moshier representation of some special automaton in F. Reworking [Moshier 1988, corollary 2] and [Moshier 1988, lemma 5] shows that two special automata are isomorphic iff the automata have identical Moshier representations, and that each morph is the Moshier representation of a special automaton. Thus, twin is a one-to-one correspondence between the morphs and the feature structures. Further, by propositions 43, the composition of image and twin is a one-to-one correspondence between the K-formalism types and the feature structures. Thus, the K-formalism types correspond to the types [Pollard and Sag 1994] envisages.

4.2 Feature structure admission

[Pollard and Sag 1994] says very little about feature structure admission, and I cannot extract an exact definition. However, I base a plausible definition of feature structure admission on the [Pollard and Sag 1994] claim that LTFS is "very close to the [logic] that we assume will underlie a fully formalised version of our theory."

The LTFS formal language easily adapts to special automata and SRL descriptions. Satisfaction becomes the smallest relation between the special automata and $\mathfrak{D}$ such that

for each special automaton A, each $\sigma \in \mathfrak{S}$ and each $\omega \in \mathfrak{W}$,
    if A transits from its initial state to state q on input ω, and
      A outputs σ at q
    then A satisfies $:\omega{\sim}\sigma$,

for each special automaton A, each $\omega_1 \in \mathfrak{W}$ and each $\omega_2 \in \mathfrak{W}$,
    if A transits from its initial state to state q_1 on input ω_1,
      A transits from its initial state to state q_2 on input ω_2, and
      $q_1 = q_2$
    then A satisfies $:\omega_1{\approx}:\omega_2$,

for each special automaton A and each $\delta \in \mathfrak{D}$,

if A does not satisfy δ then A satisfies $\neg\delta$,

for each special automaton A, each $\delta_1 \in \mathfrak{D}$ and each $\delta_2 \in \mathfrak{D}$,

if A satisfies δ_1 and A satisfies δ_2 then A satisfies $[\delta_1 \wedge \delta_2]$,

for each special automaton A, each $\delta_1 \in \mathfrak{D}$ and each $\delta_2 \in \mathfrak{D}$,

if A satisfies δ_1 or A satisfies δ_2 then A satisfies $[\delta_1 \vee \delta_2]$, and

for each special automaton A, each $\delta_1 \in \mathfrak{D}$ and each $\delta_2 \in \mathfrak{D}$,

if A does not satisfy δ_1 or A satisfies δ_2 then A satisfies $[\delta_1 \rightarrow \delta_2]$.

As said earlier, [Pollard and Sag 1994] tacitly assumes that type admission, and hence feature structure admission, is recursive. A feature structure, being an isomorphism class of special automata, has no structure over which to define admission recursively, but the subautomata provide each special automaton with just such a structure. Echoing my strategy for defining type admission, I first define an admission relation between the theories and the special automata such that special automaton admission is recursive over the subautomata of a special automaton, and then naturally abstract the admission relation from special automata to feature structures.

I say that

$\theta \subseteq \mathfrak{D}$ admits special automata A iff each subautomaton of A satisfies each $\delta \in \theta$.

Clearly,

$\theta \subseteq \mathfrak{D}$ admits special automaton A iff θ admits each subautomaton of A.

Thus, admission is recursive over the subautomata of a special automaton. Equally clearly,

for each special automaton A_1, each special automaton A_2 and each state q in A_1,

the restriction of Moore automaton isomorphism f to the states of the q subautomaton of A_1 is a Moore automaton isomorphism from the q subautomaton of A_1 to the $f(q)$ subautomaton of A_2.

Thus, by induction on the complexity of δ,

if A_1 and A_2 are isomorphic special automata and A_1 satisfies $\delta \in \mathfrak{D}$

then A_2 satisfies δ.

Thus,

if A_1 and A_2 are isomorphic special automata and $\theta \subseteq \mathfrak{D}$ admits A_1

then θ admits A_2.

Thus, admission cannot distinguish between isomorphic special automata. I can therefore felicitously abstract admission from special automata to feature structures. I say that

$\theta \subseteq \mathfrak{D}$ admits feature structure F

iff θ admits some (or equivalently, each) special automaton in F.

There is a simple correlation between morph admission and feature structure admission. By induction on the complexity of δ,

special automaton A satisfies $\delta \in \mathfrak{D}$

iff δ approximates the Moshier representation of A.

Thus, a theory admits a morph iff the theory admits the twin of the morph. Further, by propositions 48, a theory admits a K-formalism type iff the theory admits the twin of the image of the type. Thus, if my definition of feature structure admission is true then K-formalism type admission corresponds to the type admission [Pollard and Sag 1994] envisages.

5 The C-formalism

The feature structures and feature structure admission of the previous section constitute an extremely credible candidate for the HPSG formalism [Pollard and Sag 1994] envisages. I call this formalism the C-formalism, and I call the variant of the LTFS formal language upon which it is based LTFS+. In the previous section I established a correlation between the K- and C-formalisms. Yet this correlation is genuinely surprising. The K-formalism is based upon the formal language of the SRL of [King 1989], whereas the C-formalism is based upon LTFS+, a variant of the formal language of the LTFS of [Carpenter 1992]. SRL is a logic of description, LTFS is a logic of partial information, and the appropriateness conditions of HPSG ought to prevent any correlation between such logics. In this section I solve this conundrum by showing that, despite its ancestry, LTFS+ does not share the partial-information hallmarks of LTFS, and is actually an instance of SRL. I occasionally assume a good knowledge of [Carpenter 1992] in this section.

LTFS+ is an adaptation of the formal language of LTFS to special automata and SRL descriptions, where a special automaton is a Moore automaton A over input alphabet $\mathfrak{A}$ and output alphabet $\mathfrak{S}$ such that

(1) for each $\sigma_1 \in \mathfrak{S}$, each $\sigma_2 \in \mathfrak{S}$, each $\alpha \in \mathfrak{A}$, each state q_1 of A and each state q_2 of A,
  if A transits from q_1 to q_2 on input α,
    A outputs σ_1 at q_1, and
    A outputs σ_2 at q_2
  then $\sigma_2 \in \mathfrak{F}\langle\sigma_1, \alpha\rangle$, and

(2) for each $\sigma \in \mathfrak{S}$, each $\alpha \in \mathfrak{A}$ and each state q of A,
  if A outputs σ at q, and
    $\mathfrak{F}\langle\sigma, \alpha\rangle$ is not empty
  then A transits from q on input α.

Let us suppose for the moment that special automata represent information. Conditions (1) and (2) have an extremely significant effect upon the kind of information a special automaton represents. As explained earlier, the appropriateness function $\mathfrak{F}$ encodes a relationship between the interpretations of species and attributes, and SRL encodes the HPSG appropriateness conditions in the appropriateness function. The definition of an interpretation enforces the relationship the appropriateness function encodes, and thus ensures every object in every universe exhaustively meets the appropriateness conditions. A special automaton that failed to meet condition (1) would represent species and attribute information that could only be about an ill-formed object, and adding species or attribute information to a special automaton that met condition (2) would yield a special automaton that failed to meet condition (1). Thus, conditions (1) and (2) ensure that a special automaton represents maximal species and attribute information that accords with the appropriateness conditions. However, a special automaton does not necessarily represent maximal information *simpliciter*, because a re-entrancy representing the information that two components of an object are identical can be added to a special automaton without contradicting condition (1) or (2). Subsumption supports all these intuitions: if a first special automaton subsumes a second then the only possible difference between the two is that the second has re-entrancies the first lacks. To summarise this first intuitive picture: a special automaton represents maximal species and attribute information, but partial equational information. It is as if one knows everything about species and attributes, but one can acquire information about equations.

I find this part maximal, part partial picture very unsatisfactory. The maximality of the species and attribute information is a deliberate consequence of conditions (1) and (2), but the partiality of the equational information is an accidental consequence of a representational limitation of the special automata. Though re-entrancy enables a special automaton to represent the information that two components of an object are identical, no mechanism enables a special automaton to represent the information that two components of an object are distinct. Faced with a similar limitation, [Carpenter 1992, chapter 7] added an anti-reflexive and symmetric "inequality relation" to the [Carpenter 1992] feature structures, took each pair of inequated states to represent the information that the relevant components of an object are distinct, and modified subsumption to prevent re-entrancies overriding inequations. I make a similar addition to the special automata. I assume that each pair of distinct states in a special automaton is tacitly inequated, and modify subsumption accordingly. Now, no information of any kind can be added to a special automaton. The modified subsumption supports this intuition: a first special automaton subsumes a second iff the second subsumes the first. Subsumption reduces to Moore automata isomorphism. To summarise this second intuitive picture: a special automaton represents information that cannot be elaborated. It is as if one knows everything. Such information is total, and is unlike partial information. For example, negation is classical in LTFS+, yet subsumption is monotonic.

There is yet another intuitive picture: the special automata are objects in an SRL interpretation. There is an initial problem in that the collection of special automata may be a proper class. However, predetermining a set from which the special automata draw their states solves this problem. Ensuring there are sufficient states only requires that the cardinality of the predetermined set be at least the cardinality of $\mathfrak{W}$. Let U be the set of all special automata (with states drawn from the predetermined set), let S be the total function from U to $\mathfrak{S}$ such that

$S(u) = \sigma$ iff u outputs σ at its initial state,

and let A be the total function from $\mathfrak{A}$ to the set of partial functions from U to U such that

$A(\alpha)(u)$ is defined iff u transits from its initial state on input α,

and

if u transits from its initial state to state q on input α

then $A(\alpha)(u)$ is the q subautomaton of u.

Clearly, $\langle U, S, A\rangle$ is an interpretation, and, by induction on the complexity of δ,

for each $\delta \in \mathfrak{D}$ and each $u \in U$,

$u \in D_I(\delta)$ iff u satisfies δ.

Thus, LTFS+ is an instance of SRL.

Acknowledgements

The research presented in this paper was sponsored by Teilprojekt B4 "Constraints on Grammar for Efficient Generation" of Sonderforschungsbereich 340 "Sprachtheoretische Grundlagen für die Computerlinguistik" of the Deutsche Forschungsgemeinschaft. This paper took a long time to write, and I have accumulated a huge debt of gratitude to a large number of people in writing it. As very partial repayment, I heartily thank Steve Abney, Dale Gerdemann, Thilo Götz, John Griffith, Erhard Hinrichs, Stephan Kepser, Jennifer King, Peter Krause, Detmar Meurers, Frank Morawietz, Drew Moshier, Larry Moss, Tsuneko Nakazawa, Stanley Peters, Carl Pollard, Sabine Reinhard and Ivan Sag for help, encouragement and advice concerning topics that range from finding IPA fonts in LaTeX to revealing the correspondence between morphs and feature structures. I can only humbly apologise to anyone whose help I have carelessly overlooked.

References

[Carpenter 1992] Bob Carpenter. *The logic of typed feature structures.* Cambridge tracts in theoretical computer science, volume 32. Cambridge University Press, Cambridge, England. 1992.

[Carpenter 1993] Bob Carpenter. *ALE: the attribute logic engine.* User's guide. Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. 1993.

[Hopcroft and Ullman 1979] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation.* Addison-Wesley series in computer science. Addison-Wesley Publishing Company, Reading, Massachusetts, USA. 1979.

[Johnson 1988] Mark Johnson. *Attribute-value logic and the theory of grammar.* CSLI lecture notes, volume 16. CSLI, Stanford, California, USA. 1988.

[Kaplan and Bresnan 1982] Ronald M. Kaplan and Joan Bresnan. "Lexical-functional grammar: a formal system for grammatical representation." Pages 173–281 of J. Bresnan (editor). *The mental representation of grammatical relations.* MIT Press, Cambridge, Massachusetts, USA. 1982

[King 1989] Paul J. King. *A logical formalism for head-driven phrase structure grammar.* Doctoral dissertation. Manchester University, Manchester, England. 1989.

[Langholm 1989] Tore Langholm. "How to say no with feature structures." COSMOS technical report, number 13. University of Oslo, Oslo, Norway. 1989.

[Moore 1956] E. F. Moore. "Gedanken experiments on sequential machines." Pages 129–153 of *Automata studies.* Princeton University Press, Princeton, New Jersey, USA. 1956.

[Moshier 1988] M. Andrew Moshier. *Extensions to unification grammars for the description of programming languages.* Doctoral dissertation. University of Michigan, Ann Arbor, Michigan, USA. 1988.

[Moshier and Rounds 1987] M. Andrew Moshier and William C. Rounds. "A logic for partially specified data structures." Pages 156–167 of *Proceedings of the 14th ACM symposium on principles of programming languages.* 1987.

[Nerode 1958] Anil Nerode. "Linear automaton transformations." Pages 541–544 of *Proceedings of the American Mathematical Society*, volume 9. 1958.

[Partee et al. 1990] Barbara H. Partee, Alice ter Meulen and Robert E. Wall. *Mathematical methods in linguistics.* Studies in linguistics and philosophy, volume 30. Kluwer Academic Publishers, Dordrecht, The Netherlands.

[Pollard 1989] Carl J. Pollard. "Sorts in unification-based grammar and what they mean." Unpublished manuscript. 1989.

[Pollard and Moshier 1990] Carl J. Pollard and M. Andrew Moshier. "Unifying partial descriptions of sets." In P. Hanson (editor). *Information, language and cognition.* Vancouver studies in cognitive science, volume 1. University of British Columbia Press, Vancouver, British Columbia, Canada. 1990.

[Pollard and Sag 1987] Carl J. Pollard and Ivan A. Sag. *Information-based syntax and semantics.* CSLI lecture notes, volume 13. CSLI, Stanford, California, USA. 1987.

[Pollard and Sag 1994] Carl J. Pollard and Ivan A. Sag. *Head-driven phrase structure grammar.* University of Chicago Press, Chicago, Illinois, USA. 1994.

[Rounds and Kaspar 1986] William C. Rounds and Robert T. Kasper. "A complete logical calculus for record structures representing linguistic information." Pages 89–96 of *Proceedings of the 25th annual conference of the Association for Computational Linguistics.* 1986.

[Smolka 1988] Gert Smolka. "A feature logic with subsorts." LILOG technical report, number 33. IBM Deutschland GmbH, Stuttgart, Germany. 1988.