Arbeitspapiere des SFB 340
Bericht Nr. 132, September 1999

A FORMAL INTERPRETATION OF RELATIONS AND QUANTIFICATION IN HPSG

FRANK RICHTER, MANFRED SAILER, AND GERALD PENN

Contents

1. Introduction

This paper formalizes the language used for expressing the principles found in Pollard and Sag 1994. In a sense, this is not a new language, but rather the language that Pollard and Sag (1994) and other HPSG

We are most grateful to Mike Calcagno, Tom Cornell, Thilo Götz, Tilman Höhle, Hans-Peter Kolb, Detmar Meurers, Uwe Mönnich, Frank Morawietz, Carl Pollard, Adam Przepiórkowski, Kiril Simov and Shuly Wintner for discussions, comments and support. Paul King assisted in the final formulation of RSRL.

ISSN 0947-6954/99

linguists have already been using. There have been many logics proposed for the task of unification-based processing with constraint-based grammars such as HPSG, such as Höhfeld and Smolka 1988, and Carpenter 1992; but because of certain practical considerations, they have eschewed some of the formal devices that a declarative, formal specification of the constraints themselves requires. The major properties that distinguish our formal language will be described below.

The language is called RSRL, a relational extension of SRL (Speciate Reentrant Logic, see King 1989). SRL is actually a logic that was proposed as a formalization and refinement of the language used in Pollard and Sag 1987; and Pollard and Sag 1994 incorporated many of those refinements. SRL is nevertheless not an adequate formalism for Pollard and Sag 1994, *pace* King (1999) and Pollard (1999), as it crucially lacks an explicit means of quantification, which is necessary to express the quantification either understood or explicitly stated in many of the principles of Pollard and Sag 1994. In fact, the principal contribution that RSRL makes is not its addition of relations but its formalization of what the quantification of Pollard and Sag 1994 really means: not quantification over some or all objects in a model, but over the components of a given object, such as a linguistic sign. The components of an object in an interpretation are those objects that are accessible from that object by a (finite) path. This different notion of quantification gives RSRL the ability to formalize linguistic principles from the HPSG literature very naturally; and it provides an important insight into the relation between HPSG and the logics underlying HPSG-based computational systems.

We do not suppose that the reader is familiar with SRL. Since the language of SRL is a proper sublanguage of RSRL, all relevant aspects of SRL will be introduced. In fact, from a more general perspective, our augmentation of SRL by restricted quantification and relations can be viewed as the characterization of an extension that could be applied to other description languages that are more loosely related to HPSG, just as Höhfeld and Smolka 1988 provides a method to augment constraint languages with relations. The reason that we choose SRL as our starting point is that, unlike other logics of descriptions, it was expressly designed for HPSG and, therefore, implicitly incorporates several HPSG-specific mechanisms that others do not. They include HPSG's view of appropriateness specifications for sorts and attributes, and the assumption that the maximally specific sorts partition the universe of linguistic objects. Moreover, SRL is furnished with a theory of what it means for an HPSG grammar to be true of a natural language, which we can easily adapt to RSRL, as will be discussed below.

2. Relations

Relations are used prolifically in Pollard and Sag 1994 and elsewhere in the HPSG literature. Some are explicitly referred to as such, e.g., the `o-command` and `local o-command` relations of the binding theory, while others are implicitly required by the natural language descriptions of principles. The Subject Condition (p. 400), for example, says:

"If the initial element of the SUBCAT *value of a word is slashed, then so is some other element of that list."*

The consequent of this implication requires the use of a `member` relation to pinpoint another slashed element on the SUBCAT list. The recursive nature of lists requires the use of the relation.

RSRL is not the first extension of a constraint language to provide relations. Höhfeld and Smolka 1988, for example, provides a restricted kind of relational extension that can be used for logic programming over a wide range of constraint languages, including several feature logics. This relational extension, however, does not permit relations to occur in the scope of negation, which includes the antecedents of implicational constraints. The Subject Condition above does not have the `member` relation in its antecedent; but the Relative Uniqueness Principle (p. 212) does:

"A member of the INHERITED REL *set*[1] *on a headed constituent may be inherited from ... at most one daughter."*

Perhaps the most famous instance of a negated HPSG relation can be found in principle B of HPSG's binding theory (p. 401), in which a personal pronoun is required to be locally o-free, i.e., not locally o-bound. The formalization of principle B and its relations can be found in Appendix 1.

True negation, of course, is very difficult to compute with; so it is only natural for definite clause languages such as Höhfeld and Smolka 1988 or systems such as ALE (Carpenter and Penn 1996) to rely on negation by failure or some other substitute. For a formal language with which

[1] RSRL can also be used to encode finite sets and the necessary access relations, such as membership and union, for them.

we want to precisely specify models of our theories, however, these substitutes will not provide us with the usual set-theoretic interpretation of negation that matches its intended use in linguistic principles. Junk slots, the encoding of relations as components of an object by means of extra attributes and extra sorts (Ait-Kaci 1984; Carpenter 1992), cannot also encode relational negation, and require the signature to contain uninteresting and unmotivated sorts.[2]

It has been suggested (Kasper, Kathol, and Pollard 1995) that one might be able to restrict the use of negated relations to an inventory of decidable ones that can be replaced by new (computable) relations that have the opposite meaning. For example, instead of using `(¬ member(x,y))` in a description, we might simply define a new relation, `non-member`, and use `non-member(x,y)` instead. The problem is that relations are often used with quantification over at least one of their arguments. Consider, as a simple example, the following constraint, taken from Abeillé and Godard 1996, that assists in predicting the distribution of *avoir* and *être* in French:

$$\begin{bmatrix} \text{VFORM } \textit{part-passé} \\ \text{S-ARG } \langle \ldots \ \textit{aff\_r} \ldots \rangle \end{bmatrix} \rightarrow \begin{bmatrix} \text{V-AUX } \textit{etre} \end{bmatrix}$$

Formalized in RSRL, this says:

$$\begin{aligned} &[: \text{VFORM} \sim \textit{part-passé} \wedge \exists x[x \sim \textit{aff\_r} \wedge \texttt{member}(x, : \text{S-ARG})]] \\ &\quad \rightarrow [: \text{V-AUX} \sim \textit{etre}] \end{aligned}$$

If the VFORM value is of sort *part-passé*, and there exists a member of the S-ARG list which is of sort *aff_r*, i.e., which is a reflexive clitic, then the V-AUX value must be *etre*. Because `member` is in the antecedent of an implication, it occurs in the scope of a negation. It is also in the scope of an implicit existential quantifier, made explicit in our formalization, which is in the scope of the negation as well. If, in the disjunctive form of the original implication, we simply used `non-member` for `member`, we would negate the relation but not the quantifier (which must become universal). Computability properties aside, introducing new relations with negation built-in simply will not preserve the intended meaning, even in straightforward principles such as this.

A formal language for HPSG must have true relations with true negation. We extend the usual SRL-style signatures with relational symbols and an arity function to express them:

[2]See below for further problems with a junk slot approach.

DEFINITION 1. 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 finite partial order,

$$\mathcal{S} = \left\{ \sigma \in \mathcal{G} \;\middle|\; \begin{array}{l} \text{for each } \sigma' \in \mathcal{G}, \\ \text{if } \sigma' \sqsubseteq \sigma \text{ then } \sigma = \sigma' \end{array} \right\},$$

$\mathcal{A}$ 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 $\mathbb{N}^+$.

$\langle \mathcal{G}, \sqsubseteq \rangle$ is a sort hierarchy for a theory, and $\mathcal{S}$ is the set of maximally specific sorts (*species*) that will partition the objects of our interpretations. $\mathcal{A}$ is the set of attributes; and $\mathcal{F}$ encodes the appropriateness conditions on an HPSG signature. $\mathcal{R}$ and $\mathcal{AR}$ are for the relation symbols and their arities, respectively. We interpret RSRL signatures as follows:

DEFINITION 2. For each signature Σ, 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 $\alpha \in \mathcal{A}$, for each $u \in \mathsf{U}$,

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

for each $\alpha \in \mathcal{A}$, for 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 U^*, and for each $\rho \in \mathcal{R}$, $\mathsf{R}(\rho) \subseteq \mathsf{U}^{\mathcal{AR}(\rho)}$.

where S^* is the set of finite sequences or tuples over elements of a set S. Similarly, for each natural number n, S^n is the set of n-tuples over elements of a set S. Below, we will also write S^+ for the set of nonempty finite tuples over elements of a set S, and $\overline{S}$ as an abbreviation for $S \cup S^*$.

U is the set of objects in the universe. S partitions the objects of the universe by assigning a maximally specific sort to each object. A provides an interpretation of attributes as functions that map from objects to their object values at that attribute, along with conditions to enforce appropriateness. R interprets a relation as a set of n-tuples of objects, where n is the arity of that relation.

In a definite clause extension of a constraint language, we would define descriptions for that constraint language, and then an additional stratum of descriptions in which relations could be used. Because we want to use relations on a par with other descriptions in our formal language, for example, in the scope of negation and/or quantification, relations in RSRL are part of the core description language itself. $\mathcal{VAR}$ is a countably infinite set of symbols, the variables of our language.

DEFINITION 3. For each signature Σ, $\mathcal{T}^\Sigma$ and $\mathcal{D}^\Sigma$ are the smallest sets such that

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

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

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

for each $\sigma \in \mathcal{G}$, 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 $\delta_2 \in \mathcal{D}^\Sigma$, $[\delta_1 \vee \delta_2] \in \mathcal{D}^\Sigma$.

$\mathcal{T}^{\Sigma}$ is a set of terms, consisting of either a variable or the reserved symbol ‘ : ’, which refers to the object being described, followed by a (possibly empty) string of attributes. The set of formulae, $\mathcal{D}^{\Sigma}$, provides sort assignments, path equations, and relations, closed under negation, conjunction, disjunction and a restricted form of existential and universal quantification (to which we return shortly). Implication and bi-implication can be obtained by combinations of these. Relations are expressed as tuples over variables. By using the aforementioned restricted quantification, and by requiring theories to consist only of descriptions with no free variables, *relations can only hold among components of a single object.* This allows us to interpret relations under the scope of negation, by restricting the set of objects in the denotation of a negated relation to exclude those over whose designated components the relation holds. The interpretation of the negation of relations is thus achieved, as elsewhere, by set complement.[3]

In the HPSG literature, some relations over lists have been used that, at first blush, seem to violate the componenthood restriction. Consider the description of a German verb phrase in Figure 1, adapted from Kathol and Pollard (1995, p. 176): `compaction` compacts the complement daughter,

$$\left[\begin{array}{l} \textit{phrase} \\ \text{DOM} \;\; \boxed{4} \left\langle [\text{PHON}\,\langle \textit{las}\rangle] \mid \boxed{0} \left\langle \boxed{2} [\text{PHON}\,\langle \textit{das, Buch}\rangle] \right\rangle \right\rangle \\ \text{H-DTR} \left[\begin{array}{l} \textit{word} \\ \text{DOM}\,\boxed{3}\,\langle [\text{PHON}\,\langle \textit{las}\rangle] \rangle \end{array}\right] \\ \text{C-DTR}\,\boxed{1} \left[\begin{array}{l} \textit{phrase} \\ \text{DOM}\,\langle [\text{PHON}\,\langle\textit{das}\rangle], [\text{PHON}\,\langle\textit{Buch}\rangle]\rangle \end{array}\right] \end{array}\right]$$

$$\wedge\ \texttt{compaction}(\boxed{1},\boxed{2}) \wedge \texttt{shuffle}(\langle\boxed{2}\rangle,\boxed{3},\boxed{4})$$

FIGURE 1. Description of a German verb phrase.

$\boxed{1}$, into a single domain object, $\boxed{2}$, at the mother. `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. Under the componenthood perspective presented so far, all the list arguments of `shuffle` must be present in the described object for the relation

[3]The fact that the syntax of RSRL in Definition 3 bears no apparent resemblance to the commonly used AVM diagrams as exemplified in figures 1 and 2 below, does not affect the suitability of our language for specifying HPSG grammars. For practical applications, Richter 1999 defines an AVM syntax for RSRL. The syntax in Definition 3 is more compact and, thus, more convenient for mathematical purposes.

to hold. With the given order of the elements on $\boxed{4}$, this is unproblematic: `shuffle`'s first argument can (and must) be the list $\boxed{0}$. If $\boxed{4}$ is reversed, however, the described object no longer contains a list such as the one designated by $\boxed{0}$, a list whose single element is a domain object with the phonology *das Buch* (see Figure 2). Thus, of the conceivable

$$\begin{bmatrix} \textit{phrase} \\ \textsc{dom} \left\langle \boxed{2}\left[\textsc{phon}\ \langle \textit{das, Buch}\rangle\right] \;\middle|\; \boxed{3}\left\langle\left[\textsc{phon}\ \langle \textit{las}\rangle\right]\right\rangle \right\rangle \end{bmatrix}$$

FIGURE 2. Description of a *phrase* with the DOM list $\boxed{4}$ reversed

phrase with the list $\boxed{4}$ reversed, `shuffle` would not hold. Note that the problem persists if we redefine `compaction` with the list in question built into its second argument and then use that second argument as the first argument of `shuffle`, i.e., `shuffle`($\boxed{0}$,$\boxed{3}$,$\boxed{4}$), as is done by Kathol (1995, pp. 145ff). Although the list designated by $\boxed{0}$ in Figure 1 is then an argument of both relations, it might still not be a component of all phrases we wish to describe. In fact, in that case, Figure 2 has the same problem with not only `shuffle` but `compaction` also.

It is now crucial to realize that, for the use of the `shuffle` relation in our example and for similar cases in the literature, the problematic lists in question only contain objects that are themselves components of the described objects, and they never contain objects that are not components. In this sense, the componenthood restriction also applies to these constructions. This suggests that the notion of componenthood is on the right track even for these apparent counterexamples. The solution to the described problem consists in a natural generalization of the concept of components and quantification over components, and in replacing the problematic lists with *chains*, a kind of "virtual list", which can be constructed from a finite number of components, over which relations can then be defined and across which we can navigate using two logical symbols, the quasi-attributes $\dagger$ and $\triangleright$. The symbols *chain, echain* and *nechain* are the corresponding quasi-sorts, which are ordered in a hierarchy. We thus extend the set of species, $\mathcal{S}$, the set of attributes, $\mathcal{A}$, and the sort hierarchy, $\langle \mathcal{G}, \sqsubseteq \rangle$, as well as signature interpretations and the constraint language, to handle chains, as shown in Appendix 2. When we refer to these extensions, we write them as $\widehat{\mathcal{S}}$, $\widehat{\mathcal{A}}$, $\left\langle \widehat{\mathcal{G}}, \widehat{\sqsubseteq} \right\rangle$, and so forth. The crucial idea behind chains is that they provide a very simple recursive structure, namely (effectively) lists of objects, that enables us to quantify over not only single components of an object but also an arbitrary structured collection of components of an object, without granting the same status to these lists as to objects of genuine sorts.

The problem with `shuffle` pointed out above and all similar relations in the HPSG literature can now be solved very generally by simply defining the relations in question for chains in a parallel manner as they were defined for lists. In cases in which linguists find it more natural to talk about relations between recursively ordered components than about relations between single components of an object, they can do so without having to simulate their specification with new junk relations or junk objects.

In fact, we conjecture that corresponding, if in some cases quite complicated, reformulations without chains can be found for all uses in the linguistic literature of relations that violate the naive componenthood restriction. But even if that were proven correct, this approach would miss the point: The purpose of our formalism is not to codify the minimal necessary apparatus for linguistic analyses, but to give a mathematical interpretation to existing grammars that explains the meaning of and the intuitions behind these grammars as they stand. We are interested in how the actual language that people are implicitly using could be formalized.

3. Quantification

As mentioned above, we only use a restricted form of quantification in RSRL, by restricting it to components of the described object, which are those objects that are accessible by paths. The vast majority of linguistic principles are not concerned with whether some particular kind of object exists somewhere in the universe, but with whether a particular object within the same unembedded sign exists in order to license that sign. The Trace Principle (p. 400) is an excellent example:

"The SYNSEM *value of any trace must be a (non-initial) member of the* SUBCAT *list of a substantive word."*

Which substantive word could that be? This principle, in fact, is not meant to be a constraint on traces, but rather a constraint on signs that contain traces. What Pollard and Sag (1994) meant to say is that for every sign, if there is a path from its root node to a trace, then there must also be a path from the root node to some substantive word on whose SUBCAT list the SYNSEM value of the trace appears as a non-initial member. It is irrelevant whether the trace's SYNSEM value appears on some other unrelated word's SUBCAT list. This principle is working, so to speak, inside an unembedded sign. Within that scope, of course, it is easy to formalize this principle. This perspective can be most saliently observed in Pollard and Sag 1994 in the Trace Principle, the Control Theory (which looks for potential controlled *local* objects), the Quantifier Binding Condition (which excludes occurrences of a quantifier's index not captured by the quantifier) and the Binding Theory (which looks for potential o-commanders). We may also notice that the quantification in the *être* example from Abeillé and Godard 1996 above is an instance of quantification over components as well.

There are a few proposed linguistic principles that have appealed to quantification over the world of objects, rather than over some particular sign, or more generally, over the components of a particular object. A crucial example is that of polarity items. At least in the tradition of Baker (1970) and Linebarger (1987), the grammaticality of a sentence that contains a Negative Polarity Item (NPI) may depend on the well-formedness of another sentence. The NPI *give a damn* in (1a) is licensed because it stands in the scope of a negation in the same sentence. In (1b), on the other hand, no such negation is present. In a Baker/Linebarger approach, the grammaticality of the sentence is accounted for indirectly by pragmatic licensing. (1b) has (1a) as one of its conventional implicatures. In this implicated sentence, the NPI is directly licensed, which also renders (1b) grammatical.

(1) a. John doesn't give a damn.
    b. John is too tired to give a damn.

From this brief characterization of the theory of NPI-licensing, it can be seen that the grammaticality of one sentence can possibly depend on the grammaticality of a second sentence.

The relation between these two sentences, however, is not arbitrary, but one of what Linebarger (1987) calls conventional implicature.[4] It has been argued by Pollard and Sag (1994, p. 27) that implicatures of exactly this kind should be part of at least some linguistic signs. Conventional implicatures, in that approach, are explicitly mentioned as candidates for

[4]The precise connection between the two sentences is, however, hard to state. See the differences among Baker 1970 and Linebarger 1987 as well as the criticism and alternative explanation put forward by Ladusaw 1979 and Giannakidou 1997 among others.

appearing as members of the BACKGROUND value. The existence of that value is motivated on independent grounds to account for the use of proper names and honorific forms. In our view, what few cases there are that would necessitate the more standard quantification can, and according to the philosophy of signs presented in Pollard and Sag 1994, should be reduced to instances of component quantification using this explicit BACKGROUND approach.

An alternative to explicit quantification in the description language could be the use of junk slots. On the one hand, this technique shares the assumption that linguistic theories need only to be able to talk about components of a given object. In the context of junk slots, however, this assumption includes the encoding of relations themselves, which also become components of the described objects. Even simpler formal systems such as SRL (King 1989; King 1999) can provide a means for a junk slot encoding of relations, including a junk slot encoding of "quantification," but only if the objects in question have only finitely many components. A straightforward junk slot encoding of `member`, for example, looks as follows:

$:\sim$ *finished-member* $\rightarrow$:ELT $\approx$:LIST FIRST

$:\sim$ *unfinished-member* $\rightarrow$:ELT $\approx$:REC ELT $\wedge$:LIST REST $\approx$:REC LIST

Here, the sort *member* has two subsorts, *finished-member* (*f-mem*) and *unfinished-member* (*u-mem*). The attributes ELT and LIST are appropriate to *member*, with the values *top*, the most general sort, and *nelist*, respectively. *f-mem* and *u-mem* inherit these specifications from *member*; and *u-mem* has an additional attribute, REC, with value *member*. Notice that any element can be a member of any infinite list, as witnessed by an object with an infinite embedding of *u-mem* objects.

One of the problems of the junk slot method is that it cannot differentiate between the linguistically interesting components of an object and its junk components. A more serious shortcoming is their semantics. Junk slots cannot be negated with the negation of the description language; but their negation must be constructed with a negative junk slot, in our example, *non-member*. If the LIST value of the *member* junk slot is infinite, any value can occur in the ELT slot; and the same observation holds for *non-member*. The positive and negative junk slot encodings of a relation might, thus, have overlapping denotations. *Non-member* has the same problems with quantification as `non-member` discussed on page 158.

Objects with infinitely many components are not just an artefact of junk slots. In principle, linguistic objects can have an infinite number of components. In a linguistic object in a Hinrichs-Nakazawa-style analysis (Hinrichs and Nakazawa 1994), nothing prevents the SUBCAT list of an argument raising verb, for example, from being infinite, if the argument from which it inherits SUBCAT elements is itself another argument raising verb; and argument inheritance can be formalized with an *append* junk slot. In RSRL, unlike junk slots, relations and their negations do not have overlapping denotations, because negation is classical. We can even impose a constraint that all linguistic objects contain only a finite number of components, thus eliminating the potential case of arbitrary membership of any element on an infinite list, or infinite argument raising in a Hinrichs-Nakazawa-style analysis. The legitimate question of whether linguists need to be concerned with the predictions of their theories in relation to objects with infinitely many components, particularly infinite phonology lists, will not be considered here.

We conjecture that most if not all of the linguistic principles of Pollard and Sag 1994 can be converted to junk slots, modulo problems with infinity, which cannot be excluded. While that might be the price to pay for computing with these principles, we have to be aware of how large that cost actually is. A junk slot encoding of principle B of the binding theory, for example, must look at the linguistic phenomenon from a completely different angle than the original formulation in Pollard and Sag 1994. For the purposes of explicating the language that linguists are using in their declarative theories, this cost is too much. RSRL eliminates that conversion step, and provides a direct formalization of the intended principle (see Appendix 1).

We are now ready to interpret our formulae:

DEFINITION 4. For each signature Σ, for each Σ interpretation

$\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, $\mathsf{Ass}_\mathsf{I} = \overline{\mathsf{U}}^{\mathcal{VAR}}$ is the set of variable assignments in I.

Note that variable assignments in I assign objects or chains of objects of the universe to variables. This is why we can refer to chains of objects in the arguments of relations. To be able to fix the meaning of quantifiers, we need the usual notational convention for variable assignment functions:

DEFINITION 5. For each signature Σ, for each Σ interpretation

$\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$, for each $ass \in \mathsf{Ass}_\mathsf{I}$, for each $v \in \mathcal{VAR}$, for each

$w \in \mathcal{VAR}$, for each $u \in \overline{\mathsf{U}}$,

$$ass\frac{u}{v}(w) = \begin{cases} u & \text{if } v = w \\ ass(w) & \text{otherwise.} \end{cases}$$

DEFINITION 6. 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 $ass \in \mathsf{Ass}_\mathsf{I}$, T_I^{ass} is the total function from $\mathcal{T}^\Sigma$ to the set of partial functions from U to $\overline{\mathsf{U}}$, and D_I^{ass} is the total function from $\mathcal{D}^\Sigma$ to the power set of U such that, for each $u \in \mathsf{U}$,

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

$$\mathsf{Co}_\mathsf{I}^u = \left\{ u' \in \mathsf{U} \;\middle|\; \begin{array}{l} \text{for some } ass \in \mathsf{Ass}_\mathsf{I}, \\ \text{for some } \pi \in \mathcal{A}^*, \\ \quad T_\mathsf{I}^{ass}(:\pi)(u) \text{ is defined, and} \\ \quad u' = T_\mathsf{I}^{ass}(:\pi)(u) \end{array} \right\},$$

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

$$D_\mathsf{I}^{ass}(\tau \sim \sigma) = \left\{ u \in \mathsf{U} \;\middle|\; \begin{array}{l} T_\mathsf{I}^{ass}(\tau)(u) \text{ is defined, and} \\ \widehat{\mathsf{S}}(T_\mathsf{I}^{ass}(\tau)(u)) \mathrel{\widehat{\sqsubseteq}} \sigma \end{array} \right\},$$

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

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

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 $\delta \in \mathcal{D}^\Sigma$,

$$D_\mathsf{I}^{ass}(\exists v\, \delta) = \left\{ u \in \mathsf{U} \;\middle|\; \begin{array}{l} \text{for some } u' \in \overline{\mathsf{Co}_\mathsf{I}^u}, \\ u \in D_\mathsf{I}^{ass\frac{u'}{v}}(\delta) \end{array} \right\},$$

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

$$D_{\mathsf{I}}^{ass}(\forall v\, \delta) = \left\{ u \in \mathsf{U} \,\middle|\, \begin{array}{l} \text{for each } u' \in \overline{\mathsf{Co}_{\mathsf{I}}^{u}}, \\ u \in D_{\mathsf{I}}^{ass\frac{u'}{v}}(\delta) \end{array} \right\},$$

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

T_{I}^{ass} interprets terms as partial functions from objects to objects or chains of objects in the interpretation. D_{I}^{ass} interprets formulae as sets of objects in the interpretation. $\overline{\mathsf{Co}_{\mathsf{I}}^{u}}$ is the set of components of an object, u, in an interpretation, I, closed under finite sequences. Notice the appeal to $\overline{\mathsf{Co}_{\mathsf{I}}^{u}}$ in the interpretation of quantification. This is where the restriction to quantification over components and chains of components is made. Notice also that an n-ary relational formula can simply be interpreted as the set of objects whose n-tuples of components or chains of components, as determined by a variable assignment under quantification, are in the denotation of that relation. Negation, of course, corresponds to set complement.

We restrict relations and variables to components and chains of components by requiring that, in grammatical principles, no free variables occur in formulae. Let $\mathcal{D}_0^{\Sigma}$ be the set of Σ formulae without free variables. We can easily prove that the denotation of each member of $\mathcal{D}_0^{\Sigma}$ is independent of variable assignments in I.

DEFINITION 7. 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 $\mathcal{D}_0^{\Sigma}$ to the power set of U, and 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 $\delta \in \mathcal{D}_0^{\Sigma}$, $D_{\mathsf{I}}(\delta) = \left\{ u \in \mathsf{U} \,\middle|\, \begin{array}{l} \text{for each } ass \in \mathsf{Ass}_{\mathsf{I}}, \\ u \in D_{\mathsf{I}}^{ass}(\delta) \end{array} \right\}$,

for each $\theta \subseteq \mathcal{D}_0^{\Sigma}$, $\Theta_{\mathsf{I}}(\theta) = \left\{ u \in \mathsf{U} \,\middle|\, \begin{array}{l} \text{for each } \delta \in \theta, \\ u \in D_{\mathsf{I}}(\delta) \end{array} \right\}$.

Every object in our models of a grammar, $\langle \Sigma, \theta \rangle$, must satisfy every description in the theory, θ:

DEFINITION 8. For each signature Σ, for each $\theta \subseteq \mathcal{D}_0^\Sigma$, for each Σ interpretation $\mathsf{I} = \langle \mathsf{U}, \mathsf{S}, \mathsf{A}, \mathsf{R} \rangle$,

I is a $\langle \Sigma, \theta \rangle$ model iff $\Theta_\mathsf{I}(\theta) = \mathsf{U}$.

The notion of a model formalizes the idea that, with a grammar, linguists want to characterize natural language by objects which, in all their components, obey every principle of the grammar. However, as King (1999) observes, arbitrary models are not adequate to characterize a natural language. The linguistically intended model is one that contains instances of all configurations of objects that are permitted by the theory. A trivial example of an uninteresting model is the model where U is the empty set. 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. We could say that the intended model is, in a linguistically significant sense, the largest possible model of the grammar. This is called an *exhaustive model*:

DEFINITION 9. 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 $\theta' \subseteq \mathcal{D}_0^\Sigma$, for each Σ interpretation I',

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

Among the objects in an exhaustive model of a grammar are, of course, all sentences that are licensed by that grammar. In linguistic terms this means that a grammar is true of a natural language if the natural language is one of the exhaustive models of the grammar. For a more thorough discussion of exhaustive models and alternative mathematical characterizations in SRL — which, *mutatis mutandis*, carry over to RSRL —, see King 1999 and Pollard 1999. The following important result has been proven by Richter and King (1997) for RSRL:

THEOREM 1. *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 1, if a grammar has a nonempty model, it also has a nonempty exhaustive model. In other words, a non-vacuous grammar has a linguistically sensible meaning in the class of its exhaustive models.

4. Computation

RSRL was designed as a formal language for the precise specification of HPSG grammars, not for computations such as parsing per se. It is, natural, however, to ask whether it might be possible to use RSRL for computational purposes as well. (Pollard and Sag 1994) certainly had this in mind when they wrote (p. 10) of the necessity for decidability in their view of linguistic theory.

The most basic question one can hope to answer in an implementation is whether there could possibly exist a non-trivial, i.e. non-empty, collection of objects that behaves in the way that the theory predicts. This question is formalized as the *satisfaction* problem:

Definition 10. An RSRL theory, θ, over a signature, Σ, is *satisfiable* iff there exists a Σ interpretation, I, such that $\Theta_{\mathsf{I}}(\theta) \neq \emptyset$.

Theorem 2. *RSRL-satisfiability is not compact, i.e., there exists a signature, Σ, and theory, θ, for which every finite $\theta' \subset \theta$ is satisfiable, but for which θ itself is not satisfiable* (proven by Paul King).

This non-compactness result has the practical consequence that no finite sound and complete calculus can exist for RSRL-satisfiability, nor, therefore, for many of the other problems we would like an implementation to be able to solve for us.

Looking at the situation from a broader perspective, what this really shows is that the formalization of the requirements put forth by Pollard and Sag 1994 and other HPSG literature, simply taken at face value, cannot serve as the basis of a computational system. HPSG, ever since Pollard and Sag 1987, has led a dual existence, being not only a scientifically stated theory of language, but also a fairly precise specification of how to compute with a language, in particular, of how to parse the sentences of a language. The non-compactness of RSRL-satisfiability, in our view, is a mathematical proof that these two views of HPSG are fundamentally incompatible. There are, of course, feature logics over which one can design parsers. Contrary to the objectives of systems that aspire to "directly" implement HPSG grammars, however, e.g., Götz and Meurers 1995, Götz and Meurers 1997, some rephrasal or transformation of linguistic principles will inevitably be necessary. This rephrasal effort can be felt most acutely in the case of principles such as the Binding Theory, Control Theory, etc., mentioned earlier, that make significant use of the different interpretation that HPSG requires of quantification and relations, and in our own experience, is far from easy.

5. Conclusion

The task of this paper was threefold: First, by quoting examples from Pollard and Sag 1994 and others, we intended to bring to the reader's attention that an adequate formal language for HPSG grammars not only needs a means to express true relations and true negation, but also explicit quantification over components. Second, we defined a formal language, RSRL, that provides these means and has the linguistically desired kinds of models, namely exhaustive models. Third, we proved that a discrepancy exists between the apparent intended use of feature-based languages in HPSG linguistics and computational considerations for languages underlying practical systems. For HPSG, conceived of as a theory of language rather than as a theory of parsing, RSRL is the formal language which is by far the closest to the language implicitly used by HPSG linguists.

6. Appendix 1

We illustrate the suitability of RSRL to HPSG by a straightforward formalization of principle B of the binding theory as given in the appendix of Pollard and Sag 1994. We proceed by quoting the verbal descriptions of the components of principle B from Pollard and Sag 1994, followed by their formalization in RSRL.

Principle B relies on the auxiliary notions *member*, *more oblique than*, *locally o-command* and *locally o-bind*, which are defined in turn.[5]

The relation `member` is defined as usual:

$$\forall x \forall y [\mathtt{member}(x, y) \leftrightarrow [x \approx y\textsc{first} \lor \mathtt{member}(x, y\textsc{rest})]]$$

"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

[5]According to the definitions above, only variables are allowed as arguments in relational formulae. In this section, we use the following abbreviatory notation, where ρ is an n-ary relation symbol, $\tau_1, \ldots, \tau_n$ are terms, at least one τ_i is not a variable, and $v_1, \ldots, v_n$ are distinct variables that are not free in $\tau_1, \ldots, \tau_n$: $\rho(\tau_1, \ldots, \tau_n) \equiv \exists v_1 \ldots \exists v_n [\tau_1 \approx v_1 \land \ldots \land \tau_n \approx v_n \land \rho(v_1, \ldots, v_n)]$

Sag 1994, p. 401) In the formalization, x is more oblique than y on the SUBCAT list of w.

$$\forall x \forall y\ [\texttt{more-oblique}(x, y) \leftrightarrow \\ \quad \exists w\ [w \sim \textit{word} \wedge \texttt{2-the-right}(x, y, w\text{SS LOC CAT SUBCAT})]]$$

$$\forall x \forall y \forall z\ [\texttt{2-the-right}(x, y, z) \leftrightarrow \\ \quad [[z\text{FIRST} \approx y \wedge \texttt{member}(x, z\text{REST})] \vee \texttt{2-the-right}(x, y, z\text{REST})]]$$

"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." (Pollard and Sag 1994, p. 401). In the formalization, x locally o-commands y.

$$\forall x \forall y\ [\texttt{loc-o-command}(x, y) \leftrightarrow \\ \quad x\text{LOC CONT INDEX} \sim \textit{ref} \wedge y\text{LOC CONT INDEX} \sim \textit{ref} \\ \quad \wedge \neg\ x\text{LOC} \approx y\text{LOC} \\ \quad \wedge\ [\texttt{more-oblique}(y, x) \\ \qquad \vee\ \exists v[\texttt{member}(y, v\text{LOC CAT SUBCAT}) \wedge \texttt{more-oblique}(v, x)\ ]]]$$

"A referential *synsem* object *locally o-binds* another provided it locally o-commands and is co-indexed with the other." (Pollard and Sag 1994, p. 401) The use of the relation `loc-o-command` in the definition of `loc-o-binds` makes sure that x and y are referential *synsem* objects.

$$\forall x \forall y\ [\texttt{loc-o-binds}(x, y) \leftrightarrow \\ \quad \texttt{loc-o-command}(x, y) \wedge x\text{LOC CONT INDEX} \approx y\text{LOC CONT INDEX}]$$

With these definitions to hand, we can formalize principle B of the binding theory directly: "Principle B: A personal pronoun must be locally o-free." (Pollard and Sag 1994, p. 401)

$$\forall x\ [x\text{LOC CONT} \sim \textit{ppro} \rightarrow \neg\ \exists y\ \texttt{loc-o-binds}(y, x)]$$

For every object o, for each *synsem* component, x, of o that is a personal pronoun, there is no component y of o such that y locally o-binds x.

7. Appendix 2

This section presents the necessary extensions of RSRL to chains, as discussed in Section 2. We extend the sort hierarchy by a chain hierarchy, and the set of attributes by the quasi-attributes in the obvious way:

Definition 11. For each signature $\Sigma = \langle \mathcal{G}, \sqsubseteq, \mathcal{S}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{AR} \rangle$,

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

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

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

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

Note that $\left\langle \widehat{\mathcal{G}}, \widehat{\sqsubseteq} \right\rangle$ is a finite partial order. As with the sort hierarchy and attribute set, we extend the species assignment function, S, and the attribute interpretation function, A, to chains.

Definition 12. 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 $u_n \in \mathsf{U}$,

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

$\widehat{\mathsf{A}}$ is the partial function from $\widehat{\mathcal{A}}$ to the set of partial functions from $\overline{\mathsf{U}}$ to $\overline{\mathsf{U}}$ such that for each $\alpha \in \mathcal{A}$, $\widehat{\mathsf{A}}(\alpha) = \mathsf{A}(\alpha)$, and

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

$\widehat{\mathsf{A}}(\triangleright)$ is the total function from U^+ to U^* such that for each $\langle u_0, \dots, u_n \rangle \in \mathsf{U}^+$, $\widehat{\mathsf{A}}(\triangleright)(\langle u_0, \dots, u_n \rangle) = \langle u_1, \dots, u_n \rangle$.

$\widehat{\mathsf{S}}$ is just like S, except that it assigns the quasi-sort *echain* to the empty chain, and *nechain* to each nonempty chain of objects. $\widehat{\mathsf{A}}$ is just like A, except that it 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 missing.

Finally, the definitions of an interpretation and of the constraint language must be slightly modified to accommodate chains. To allow chains as arguments of relations, we redefine the denotation of relations, R, as a total function from $\mathcal{R}$ to the power set of $(\overline{\mathsf{U}})^*$, and require for each $\rho \in \mathcal{R}$ that $\mathsf{R}(\rho)$ be a subset of $\overline{\mathsf{U}}^{\mathcal{AR}(\rho)}$. For the description of chains, the quasi-sorts and quasi-attributes must be available in the formulae of our language. Therefore, we replace the lines 5 and 6 of definition 3 and say now that for each $\alpha \in \widehat{\mathcal{A}}$ and for each $\tau \in \mathcal{T}^{\Sigma}$, $\tau\alpha \in \mathcal{T}^{\Sigma}$, and for each $\sigma \in \widehat{\mathcal{G}}$ and for each $\tau \in \mathcal{T}^{\Sigma}$, $\tau{\sim}\sigma \in \mathcal{D}^{\Sigma}$.

References

Abeillé, A. and D. Godard (1996, juin). La complémentation des auxiliaires français. *Langages 122*, 32–61.

Ait-Kaci, H. (1984). *A Lattice Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures.* Ph. D. thesis, University of Pennsylvania.

Baker, C. L. (1970). Double negatives. *Linguistic Inquiry 1*, 69–96.

Carpenter, B. (1992). *The Logic of Typed Feature Structures.* Cambridge University Press. Cambridge, Massachusetts, USA.

Carpenter, B. and G. Penn (1996). Efficient parsing of compiled typed attribute value grammars. In H. Bunt and M. Tomita (Eds.), *Recent Advances in Parsing Technology.* Kluwer.

Giannakidou, A. (1997). *The Landscape of Polarity Items.* Ph. D. thesis, University of Groningen.

Götz, T. and D. Meurers (1995). Compiling HPSG type constraints into definite clause programs. In *Proceedings of the 33rd ACL.*

Götz, T. and D. Meurers (1997). Interleaving universal principles and relational constraints over typed feature logic. In *Proceedings of the 35th ACL / 8th EACL*, pp. 1–8.

Hinrichs, E. and T. Nakazawa (1994). Linearizing AUXs in German verbal complexes. In J. Nerbonne, K. Netter, and C. J. Pollard (Eds.), *German in Head-Driven Phrase Structure Grammar*, Number 46 in CSLI Lecture Notes, Stanford University, pp. 11–37. CSLI Publications.

Höhfeld, M. and G. Smolka (1988). Definite relations over constraint languages. LILOG Report 53, IBM Deutschland.

Kasper, R., A. Kathol, and C. Pollard (1995). A relational interpretation of linear precedence constraints. *Paper presented at the Fourth Meeting on Mathematics of Language (MOL4).*

Kathol, A. (1995). *Linearization-Based German Syntax.* Ph. D. thesis, Ohio State University.

Kathol, A. and C. Pollard (1995). Extraposition via complex domain formation. *Proceedings of the 1995 Annual Meeting of the Association for Computational Linguistics*, 174–180.

King, P. J. (1989). *A logical Formalism for Head-Driven Phrase Structure Grammar.* Ph. D. thesis, University of Manchester.

King, P. J. (1999). Towards Truth in Head-driven Phrase Structure Grammar. This volume.

Ladusaw, W. (1979). *Polarity Sensitivity as Inherent Scope Relations.* Ph. D. thesis, University of Texas at Austin.

Linebarger, M. C. (1987). Negative polarity and grammatical representation. *Linguistics and Philosophy 10*, 325–387.

Pollard, C. and I. A. Sag (1987). *Information-Based Syntax and Semantics. Vol.1: Fundamentals.* CSLI Lecture Notes 13.

Pollard, C. and I. A. Sag (1994). *Head-Driven Phrase Structure Grammar.* University of Chicago Press.

Pollard, C. J. (1999). Strong generative capacity in HPSG. In G. Webelhuth, J.-P. Koenig, and A. Kathol (Eds.), *Lexical and Constructional Aspects of Linguistic Explanation*, pp. 281–297. CSLI Publications.

Richter, F. (1999). RSRL for HPSG. This volume.

Richter, F. and P. J. King (1997). On the Existence of Exhaustive Models in a Relational Feature Logic for Head-driven Phrase Structure Grammar. Manuscript.

Frank Richter

Seminar für Sprachwissenschaft
SFB 340, B8
Kleine Wilhelmstr. 113
72074 Tübingen
Germany

E-mail address: `fr@sfs.nphil.uni-tuebingen.de`
Homepage: `http://www.sfs.nphil.uni-tuebingen.de/~fr/`

Manfred Sailer

Seminar für Sprachwissenschaft
Abt. Computerlinguistik
Kleine Wilhelmstr. 113
72074 Tübingen
Germany

E-mail address: `mf@sfs.nphil.uni-tuebingen.de`
Homepage: `http://www.sfs.nphil.uni-tuebingen.de/~mf/`

Gerald Penn

Seminar für Sprachwissenschaft
SFB 340, B8
Kleine Wilhelmstr. 113
72074 Tübingen
Germany

E-mail address: `gpenn@sfs.nphil.uni-tuebingen.de`
Homepage: `http://www.sfs.nphil.uni-tuebingen.de/~gpenn/`