Expressing generalizations in unification-based grammar formalisms *

Marc Moens, Jo Calder
Ewan Klein, Mike Reape, Henk Zeevat

Centre for Cognitive Science, University of Edinburgh
2, Buccleuch Place, Edinburgh EH8 9LW
Scotland, UK

Abstract

This paper shows how higher levels of generalization can be introduced into unification grammars by exploiting methods for typing grammatical objects. We discuss the strategy of using global declarations to limit possible linguistic structures, and sketch a few unusual aspects of our type-checking algorithm. We also describe the sort system we use in our semantic representation language and illustrate the expressive power gained by being able to state global constraints over these sorts. Finally, we briefly illustrate the sort system by applying it to some agreement phenomena and to problems of adjunct resolution.

1 Introduction

Since Kay's seminal work (Kay 1979), the utility of unification as a general tool in computational linguistics has gained widespread recognition. One major point on which the methodology of unification grammars differs radically from that assumed by linguistic theories lies in the way they deal with generalizations that hold over the domain of description. In unification-based theories, such generalizations are typically implicit, or extremely limited in their import. The reasons for this are easy to pinpoint. First, in such theories one has to be explicit about the feature structures that the grammar manipulates, and these structures have to be described more or less directly. In PATR-II for example (Shieber *et al* 1983) the only means of expressing a generalization is via the notion of *template*, a structure which merely represents recurring information—i.e. information that recurs in different lexical items, combination rules, lexical rules or other templates. A second reason why unification-based theories do not lend themselves easily to the expression of general statements is that there is no explicit quantification in unification formalisms. In fact, every statement in these formalisms represents a simple existential constraint, never a universal generalization.

The work reported here is an attempt to introduce higher levels of organization into unification grammars. The notions we employ to do this come from sorted logics and from strong data typing in programming language theory. We will show that the typing of grammatical objects offers a way of stating structural constraints on, or equivalently universal properties of, the objects that constitute the grammar.

The grammatical framework in which these ideas have been implemented is *Unification Categorial Grammar* (UCG) and its semantic representation language InL, both developed as part of the ESPRIT-funded project ACORD. Introductions to UCG and InL can be found in Calder *et al* (1988b) and Zeevat (1988). For present purposes it is sufficient to note that UCG uses a sorted logic which requires being able to express complex constraints over clusters of features. While there is no real distinction between this technique and that of data typing mentioned above, we will nevertheless continue to use the term *typing* only to refer to constraints on the global structure of an object and reserve the term *sort* to refer to constraints that hold of a variable in InL.

In the following sections, we will first discuss our strategy of using global declarations to limit possible linguistic structures. We will briefly describe some of the type declarations currently implemented in UCG and discuss the unusual aspects of our type-checking algorithm. We will also infor-

*The work reported here was carried out as part of ESPRIT project P393 ACORD. A longer version of this paper can be found in Calder *et al* (1988a).

mally describe the InL sort system and will show how the ability to express global constraints on the sort lattice is both perspicuous and expressively powerful. Detailed discussion of the underlying formal theory and the implementation can be found in Calder *et al* (1988a) and will not be attempted here.

Next, we will demonstrate the usefulness of the sort system by describing UCG's adjunct resolution system, the declarative semantics of which depends crucially on our use of a logic of sorts. This treatment allows the grammar writer to write and add adjunct resolution conditions using the same notation as that used to express sort descriptions in the grammar and without having to modify any implementation code.

2 Types in UCG

Importing the notion of data typing into unification-based grammars has several advantages (cf. also Calder *et al* 1986, Calder 1987). To begin with, the use of data typing allows one to show whether a grammar is consistent with a set of statements about the possible structures allowed within the grammar. This compile-time type-checking of the structures designed by the grammar writer allows more useful error information to be presented to the grammar writer. We have found such information essential in writing large grammars for the ACORD project.

Second, data typing forces the grammar writer to make the structure of linguistic objects explicit. This higher level of organization makes it easier to pinpoint aspects of the grammar which are inelegant or inefficient.

Finally, the notion of typing represents a further step towards the goal of making local structures reflect global restrictions. This move is an essential part of the programme of characterizing, within a formal computational theory, linguistic devices such as GPSG's feature co-occurrence restrictions.

A standard way of defining categorial grammars is to provide a set of basic categories and one or more recursive rules for defining complex categories. A very similar definition holds in UCG. Following Pollard & Sag (1987), we treat every UCG object, apart from the rules, as a sign. That is, it represents a complex conjunction of phonological, syntactic and semantic information. We can further specify a sign by adding constraints on legal instantiations of each of the sign's attributes: for example, semantics in UCG has a tripartite structure, consisting of an index, a predicate and an argument list.

It is obvious that the abstract structure of each of these categories must be known in advance to the interpreter. The formalism we will use here for declaring types is borrowed from Smolka (1988), and the following illustrates his matrix notation for record structures, where type symbols are written in bold face, and feature symbols are written in italics[1]:

(1)
$$\begin{bmatrix} \textbf{sign} \\ \textit{phonology} : \textbf{phonlist} \sqcup \textbf{basic} \\ \textit{category} : \textbf{complex} \sqcup \textbf{bcat} \sqcup \textbf{basic} \\ \textit{semantics} : \textbf{variable} \sqcup \textbf{formula} \end{bmatrix}$$

The structure as a whole is declared to be of type **sign**, and it is defined for exactly three features, namely *phonology*, *category*, and *semantics*. We also show, for each feature, the types of the values that it takes; as it happens, these are all disjunctive. So, for example, the feature *semantics* has a value either of type **variable** or of type **formula**.

Obviously, further information has to be given about what constitute legal structures of type **formula**. As was mentioned above, semantic formulae in InL are typically tripartite:

(2)
$$\begin{bmatrix} \textbf{formula} \\ \textit{index} : \textbf{variable} \\ \textit{predicate} : \textbf{basic} \sqcup \textbf{list} \\ \textit{arglist} : \textbf{basic} \sqcup \textbf{sem\_args} \end{bmatrix}$$

For present purposes, it suffices to know that the first element is the index, a privileged variable representing the ontological type and the identity of the semantic structure. Next, there is the predicate. This may be basic or a list of atoms. The type **basic** is the only type provided as a primitive in the system, and indicates that only instantiations to an atomic value (in the PROLOG sense of atomic) are legal. In the case where the predicate is a list, it represents a disjunction over adjunct functions, as will be discussed below.

Further discussion of (1) and (2) is not possible within the limited space here. The examples are only intended to illustrate how at each level in a UCG sign, type specifications can be given that indicate restrictions on the value any given feature may take on. However, one point deserves further

[1] Smolka uses the term 'sort' in place of 'type'; however, as already mentioned, we reserve the former for talking about InL expressions.

comment. It will be recalled that earlier we said the structure (1) was "defined for exactly three features". It follows from this that, for example, (1*l*) would not be a legal instantiation of this type:

(1*l*)
$$\begin{bmatrix} \textbf{sign} \\ \textit{phonology}: \textbf{value\_a} \\ \textit{category}: \textbf{value\_b} \\ \textit{semantics}: \textbf{value\_c} \\ \textit{arglist}: \textbf{value\_d} \end{bmatrix}$$

Thus, types in UCG are *closed*: all features which are not explicitly stated as defined in a particular type declaration are held to be undefined for that type (i.e. they can only be specified as $\perp$). Consequently, closed types offer a form of universal quantification over features. This device offers a way of characterizing the well-formedness of different dimensions of a sign that is stronger than systems based on open types, such as HPSG.[2]

The UCG compiler uses declarations like those in (1) and (2) to check variables for consistent typing. This involves keeping track of all variables introduced by a particular UCG expression as well as of the possible types that a variable may be assigned. The compiler proves that, for multiple occurrences of the same variable, the intersection of the sets of possible types induced for each occurrence of the variable is non-empty. If the set is empty, the compilation process fails and an error is reported.

This technique has the advantage that one may partition the set of variables employed by the system. Thus in UCG, the set of PROLOG variables that is used to represent variables in an InL formula is disjoint from the set used to represent the predicate introduced by a sign: the type of variables of the first set is stated to be **variable**, while the type of those of the second set is **predicate**. This property is crucial if we wish to check for correctness of highly underspecified structures.

3 The sort system

The ontological types of InL indices are formalized by dividing the set of InL variables into sorts. Taking results from work in automated theorem proving (Cohn 1984, Walther 1985), the use of sorted variables in InL was first presented in Calder *et al* (1986). Similar proposals have also been made in the SRI Core Language Engine (Alshawi *et al* 1988) and in recent HPSG work on referential parameters (Pollard & Sag 1988).

As a first approximation, InL sorts can be identified with bundles of feature-value pairs, such as (3):

(3) [-Temporal, +Human, +Singular]

However, the standard linguistic notation for feature bundles is too restricted, since it only allows conjunction and negation of atoms. We find it useful to use a full propositional language $\mathcal{L}_{sor}$ for expressing sortal information, where each feature specification of the form $+F$ is translated into $\mathcal{L}_{sort}$ as an atomic proposition F, and each specification $-F$ is translated as a negated atom $\neg F$. Thus, in place of (3) we write the following:

(4) $\neg$Temporal $\wedge$ Human $\wedge$ Singular

This is construed as a partial description of elements in the semantic domain used to interpret InL. In order to calculate the unification of two sorted variables, we conjoin the associated sort formulae and check for consistency.

The design of the sort structure as a theory of propositional logic also allows the incorporation of background constraints or axioms with which every possible description in the structure is consistent. Let's call the theory $\mathcal{T}_{sort}$. A few examples of these background axioms in $\mathcal{T}_{sort}$ are given in (5) to (9):

(5) Temporal $\rightarrow$ Neuter $\vee$ Plural
(6) Neuter $\rightarrow$ Singular $\wedge$ $\neg$Human
(7) Singular $\rightarrow$ Objectual
(8) Measure $\rightarrow$ Objectual $\wedge$ (Tmeasure $\vee$ Lmeasure)
(9) Stative $\rightarrow$ Eventual

From (5), (6) and (7) it follows that the unification of an index of sort *Temporal* and an index of sort *Neuter* should give us an index of sort

(10) Objectual $\wedge$ Singular $\wedge$ $\neg$Human

And from (8) it follows that anything that is *Tmeasure* is also *Objectual*. This implicit deductive capacity is useful in specifying concisely and accurately the sort of an index.

A few examples will help clarify these distinctions. Below are listed the lexical definitions for some of the nouns in the current lexicon. In these definitions, the items preceded by the symbol "@" are templates, in the sense of PATR-II. Template whose names are the unabbreviated form of sort names instantiate the *index* of the *semantics* of a sign to the corresponding sort. For example "@Extended" specifies the sort of the InL variables as *Extended*, "@Neuter" as *Neuter*, etc.

tomato: [@Noun, @Neuter, @Extended,
:pred = tomato].

[2] See Uszkoreit (1987) and Bouma *et al* (1988) for a system that allows the flexible combination of open and closed types.

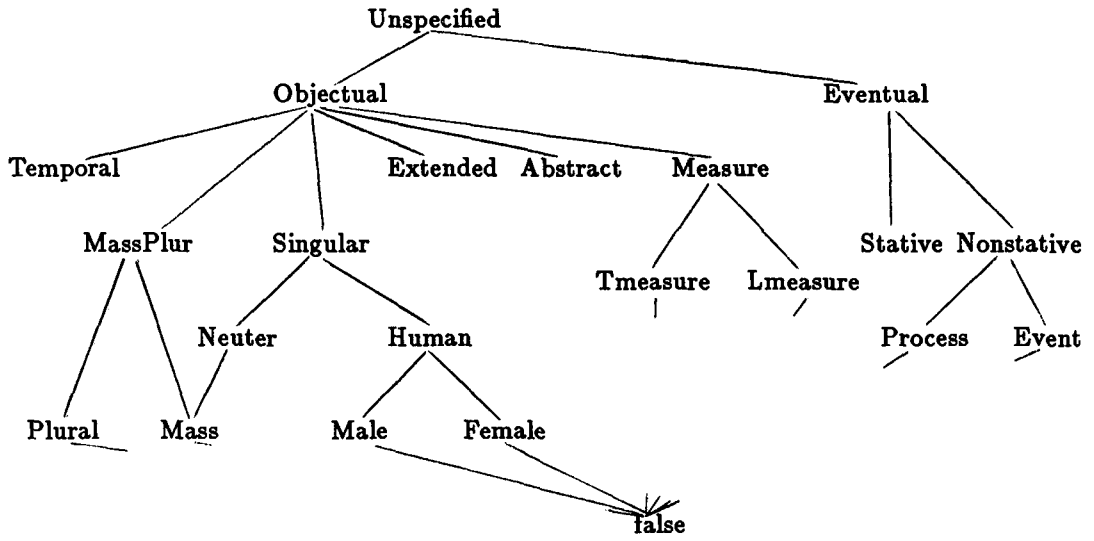

Figure 1: Sort lattice (overview)

inquiry: [@Noun, @Temporal, @Neuter, :pred = inquiry].

organisation: [@Noun, @Neuter, @Abstract, :pred = organisation].

miles: [@Noun, @Lmeasure, @Plural, :pred = mile].

night: [@Noun, @Neuter, @Tmeasure, :pred = night].

A tomato is obviously an object with spatial extent. It is also *Neuter*, which implies—given the axiom in (6) above—that it is also *Singular*, and not *Human*. An inquiry is also *Neuter*, but it has a temporal dimension; a time span can be predicated of it. An organisation is an abstract entity; it is, moreover, *Neuter* (implying it is a singular object). Finally, miles has the index *Lmeasure* since it can be used in measure phrases to express the length of something; and night is *Tmeasure* which means it can be used to express the temporal duration of something.

The standard consequence relation over these partial descriptions (i.e. the formulae of $\mathcal{L}_{sort}$) induces a lattice (cf. Mellish 1988). Moreover, the sets of models associated with these partial descriptions (i.e. the truth assignments to the formulae) also form a lattice, ordered by the set inclusion relation. This lattice is isomorphic to the lattice of descriptions. The model sets can be encoded as binary bit strings where a zero bit indicates that the corresponding model is not a member of the model set and a one bit indicates the opposite. Model set intersection is equivalent to bitwise conjunction and model set union to bitwise disjunction. Testing for the satisfiability of the conjunction of two descriptions can consequently be performed in two machine instructions, viz. taking the bitwise conjunction of two model set encodings and testing for zero (cf. Proudian & Pollard 1985).

Such a model set encoding is obviously linear in the number of models it generates; in the worst case, the number of models is exponential in the number of propositional constants mentioned in $\mathcal{T}_{sort}$, but typically it is much less. This means that the exponential complexity involved in testing for satisfiability can be compiled away offline; the resulting model set encoding can be used with equal computational efficiency.

As illustrated above, the statements that define the lattice of sorts can be arbitrary statements in classical propositional logic. This is in distinction to systems discussed by Mellish (1988) and Alshawi *et al* (1988), in which the set of logical connectives is restricted to those for which an encoding exists using PROLOG terms without repeated variables and for which PROLOG unification provides an immediate test of the compatibility of two descriptions. The resulting sort definition language is therefore more expressive. The major drawback of such an approach is that the encoding

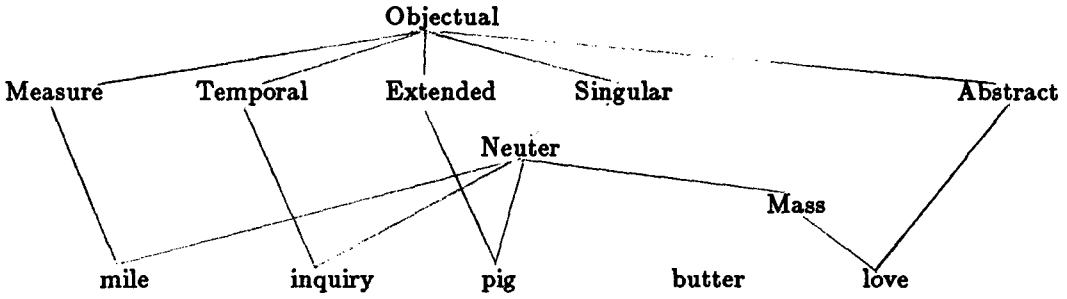

Figure 2: Sort lattice plus examples (detail)

in terms of sets of satisfying models prevents the statement of reentrant dependencies between features in the sort system and features in the rest of the grammar. A more general, but computationally less efficient approach would use general disjunction and negation over feature structures, as discussed by Smolka (1988), and so give a uniform encoding of sortal and general grammatical information.

Figure 1 depicts part of our current lattice of sorts. It is not complete in that not all the sorts we currently use are represented in Figure 1, nor are all the meets of the sorts in Figure 1 represented. Figure 2 gives an enlarged fragment of Figure 1, showing a more complete picture of the sorts related to *Neuter*, as well as some instantiations of these sorts in English.

The fact that the lattice soon becomes rather complicated isn't particularly worrisome: the grammar writer need only write simple background axioms in $\mathcal{T}_{sort}$, like the ones in (5) to (9), to extend or otherwise change the sort lattice. To check for plausibility, the grammar writer can also ask for the models or truth assignments to the properties of the sort system.

In UCG, sortal restrictions have been used to capture certain agreement phenomena. Collective nouns like *committee*, for example, are lexically marked as being either *Neuter* or *Plural* (for which, of course, the term *Collective* can be introduced). In British English, this allows anaphoric reference by means of a singular as well as a plural pronoun:

(11) The committee met yesterday. It/They rejected the proposal.

Proper binding of the pronoun in (11) requires the index associated with *it* or *they* to be identical with that introduced by *committee*. Since *committee* is marked as either *Neuter* or *Plural*, both bindings are possible.

However, once the choice has been made (as in (12a) and (b)) the referential index for committe has become specified more fully (as being eithe singular or plural) and further pronominal refer ence in the discourse is restricted (as illustrate in (c) and (d)) (cf. Klein & Sag 1982, and mor recently Pollard & Sag 1988 on this issue):

(12a) The committee has rejected its own pro posal.

(12b) The committee have rejected their ow proposal.

(12c) *The committee has rejected their ow proposal.

(12d) *The committee have rejected its own pro posal.

Note that sorts like *Plural* or *Neuter* are not syn tactic features, but are part of the internal struc ture of referential indices introduced through th usage of certain expressions. These indices are ab stract objects whose function in a discourse repre sentation it is, amongst other things, to keep trac of the entities talked about in the discourse.

Of course, sorts like *Plural* or *Human* also hav a *semantic* import in that they permit real-worl non-linguistic objects to be distinguished fron one another (cf. Hoeksema (1983) and Chierchi (1988) on a similar use of indices in theories o agreement and binding). Nevertheless, the aim o the sort system is not to reflect the characteris tics of real world objects and events referred to b linguistic expressions, but rather to systematiz the ontological structure evidenced by linguisti expressions.

The usefulness of being able to express globa constraints over the sort lattice can best be illus trated by considering the treatment of adjunct res olution in UCG. It is to a brief account of this tha we turn next.

4 Adjunct resolution

Ambiguity in the attachment of prepositional phrases is a longstanding problem in the area of natural language processing. We suggest that this ambiguity has two basic causes. First, there is structural ambiguity in that prepositional phrases may modify at least nouns and verb phrases. This structural ambiguity is a cause of inefficiency in processing. Second, prepositions may have several distinct, if related, meanings. (This problem becomes even more acute in a multilingual setting with a common semantic representation language). Such ambiguity then represents an indeterminacy for theorem provers and knowledge bases that deal with the output of a natural language component.

The mechanisms we have introduced above allow us to address both these problems simultaneously. We use the term adjunct resolution to describe the situation in which the possible meanings of a preposition, perhaps drawn from a universal set of possible prepositional meanings, and the possible attachments of a prepositional phrase are mutually constraining.

To consider the problem from the multilingual point of view, the way in which a particular language uses its prepositions to decompose the set of spatial and temporal relations that obtain between objects and events may well be inconsistent with the decomposition shown in other languages. For example, the French preposition *dans* can express spatial location (*il est* dans *la chambre* – *he is* in *the room*), spatial inclusion (dans *un rayon de 15 kilomètres* – within *a radius of 10 miles*), spatial path (*il passerait* dans *le feu pour elle* – *he'd go* through *fire for her sake*), spatial source (*copier quelque chose* dans *un livre* – *copy something* from *a book*), and several other relations.

In the semantic representation language InL, the meaning of a preposition is a relation between two InL indices. Thus the translation of a sentence like

(14) John walked to the store

would be

(15) [e][walk(e,john) & store(x)
& direction(e,x)]

where "direction(e,x)" represents a relation between the going event and the store. However, as noted above, a preposition will typically introduce a disjunction over relations. The French preposition *dans*, for example, will have as its translation a disjunction of spatial location, spatial inclusion, spatial source and spatial path. Some of these it will share with the English preposition *in*; others will be shared with *within*, *through* and the other prepositions mentioned above.

Let us look at an English example in some more detail. An adjunct phrase introduced by *with* can express (without aiming to be exhaustive) an *accompaniment* relation (as in 18a), the *manner* in which an act was carried out (18b), the *instrument* with which it was carried out (illustrated in 18c), or something which is part of something or owned by someone (as in 18d).

Sortal restrictions on the arguments of these relations are expressed by means of the three-place predicate *sort_restriction*:

(16) sort_restriction(RELATION,
HEAD_INDEX,
MODIFIER_INDEX).

In (16), RELATION is a possible adjunct relation (or a list of adjunct relations, interpreted disjunctively), HEAD_INDEX represents the conditions on the index of the expression modified by the adjunct, and MODIFIER_INDEX likewise states restrictions on the index of the object that is part of the modifier phrase.

An instance of this schema is (17):

(17) sort_restriction(instrument,
¬Stative ∧ Eventual,
Extended ∧ ¬Human)

The declaration in (17) restricts instruments to be non-human, extended objects. They can, moreover, only be combined with nonstative or event expressions. This rules out an instrumental reading for the *with*-phrases in (18a) and (b) (since *teacher* will be marked in the lexicon as *Human*, and *effort* is *Abstract*), and for (18d) (since *the man* is not *Eventual*), but allows it for (c):

(18a) Lisa went to Rome with her teacher.
(18b) He ran with great effort.
(18c) He broke the window with a hammer.
(18d) There's the man with the funny nose.

The restrictions on accompaniment, manner and possession are given as follows:

(19) sort_restriction(accompaniment,
Eventual,
Extended)
(20) sort_restriction(manner,
¬Stative ∧ Eventual,
Abstract)
(21) sort_restriction(possession,
Objectual,
Extended ∧ ¬Human)

It is easy to verify that (19) rules out an accompaniment reading for (18b) (since *effort* is not *Extended*) and for (18d) (since *man* is not *Even-*

tual). (20) renders a manner reading impossible for (18a), (c) and (d), since neither *teacher*, *hammer* or *nose* are *Abstract*. Finally, (21) rules out a possession relation for (18a) and (b).

In some cases the sortal restrictions will reduce the disjunction of possible readings to a single one, although this is obviously not a goal that is always obtainable or even necessary for the semantics component of a natural language system.

As the discussion of the *with*-clauses shows, in some cases PP attachment ambiguity may be reduced by restrictions associated with particular adjunct prepositions. A standard example of such an ambiguity is

(22) John saw the man with a telescope.

There are two readings to this sentence, represented by these two bracketings:

(23a) [$_{VP}$saw [$_{NP}$the man [$_{PP}$with a telescope]]]

(23b) [$_{VP}$ [$_{VP}$saw the man][$_{PP}$with a telescope]]

Due to the restrictions given above, only the *possession* relation may hold between *man* and *telescope* in (23a), while in (b) only the relations *accompaniment* or *instrument* may hold between the telescope and the event of seeing.

In some cases, the sortal restrictions may actually remove prepositional attachment ambiguities altogether. Examples (24) are predicted by most theories to be ambiguous:

(24a) John will eat the tomato in two hours.

(24b) John will eat the tomato in his office.

The ambiguity arises because the prepositional phrase may attach low, to the noun phrase, or high, modifying the verb phrase. In the system described here, the first sentence is not ambiguous. The preposition *in* introduces a disjunction between (amongst other things) spatial location and duration. The former can relate an object with any other object or event. The latter relation can only hold of expressions involving some temporality; as was illustrated above, *tomato* has no temporal extent, therefore does not allow this kind of temporal time-span to be predicated of it. As a result, the prepositional phrase in (24a) can only get high attachment.

Although the discussion has been limited to the use of sortal information in adjunct resolution and the treatment of certain agreement phenomena, it should be clear that exactly the same mechanism may be used to indicate sortal restrictions associated with any other predicates of the system. Thus we have one way of expressing the linguistic concept of selectional restrictions. We realize that care has to be taken here, since there is no well-defined point at which statements about sortal correctness become clearly inappropriate. For instance, we might be tempted to treat the ambiguity associated with the verb *bank* as in *Ronnie banked the cheque* and *Maggie banked the* MIG by invoking a feature *monetary* for the first example and a feature *manoeuvrable* for the second. If we had a clear picture of precisely those properties that might be invoked for lexical disambiguation, this approach might be tenable. It seems more likely to be the case that the features and axioms about those features used in a particular case are *ad hoc* and domain-specific, as their creation and definition would be governed by just those lexical items one wanted to distinguish. Also they are language-specific, as patterns of homography presumably do not hold cross-linguistically. It is, nevertheless, plausible (following Kaplan 1987) to assume that the techniques we have introduced could be employed in the automatic projection of non-lexical knowledge into the lexicon.

The notation we have presented above for the definition of sorts and the relations between sorts that prepositions represent may appear somewhat removed from the notation introduced in section 2 in our discussion of typed grammatical objects. It is however worth noting that the use of "order-sorted algebras" (Meseguer *et al* 1987) as the mathematical basis of feature structures allows not only the statement of such restrictions on the structure of grammatical and semantic objects, but also the definition of relations, like our prepositional relations above, whose interpretation is dependent on the interpretation of the structures they relate. Such formalisms may well provide a useful foundation for a more general theory of prepositional meaning and its relation to syntactic structure.

References

Alshawi, H., Carter, D. M., van Eijck, J., Moore, R. C., Moran, D. B., Pereira, F. C. N., Smith, A. G. and Pulman, S. G. [1988] Interim Report on the SRI Core Language Engine. Technical Report No. CCSRC-005. Cambridge Computer Science Research Centre, Cambridge, UK. July 1988.

Bouma, G., Koenig, E. and Uszkoreit, H. [1988] A Flexible Graph-Unification Formalism and its Application to Natural Language Processing. IBM *Journal of Research and Development*, **32**, 170-184.

Calder, J. [1987] Typed unification for nat-

ural language processing. In Klein, E. and van Benthem, J. (eds.) *Categories, Polymorphism and Unification*, pp65-72. Centre for Cognitive Science, University of Edinburgh, and Institute for Language, Logic and Information, University of Amsterdam.

Calder, J., Klein, E., Moens, M. and Zeevat, H. [1986] Problems of Dialogue Parsing. ACORD Deliverable T2.1, Centre for Cognitive Science, University of Edinburgh. December 1986.

Calder, J., Klein, E., Moens, M. and Reape, M. [1988a] Global Constraints in Unification Grammar. ACORD Deliverable T1.6, Centre for Cognitive Science, University of Edinburgh. February 1988.

Calder, J., Klein, E. and Zeevat, H. [1988b] Unification Categorial Grammar: A Concise, Extendable Grammar for Natural Language Processing. In *Proceedings of the 12th International Conference on Computational Linguistics and the 24th Annual Meeting of the Association for Computational Linguistics*, Budapest. August 1988, pp. 83-86.

Chierchia, G. [1988] Aspects of a Categorial Theory of Binding. In Oehrle, R., Bach, E. and Wheeler, D. (eds.) *Categorial Grammars and Natural Language Structures*, pp125-151. Dordrecht: D. Reidel.

Cohn, A. G. [1984] On the Solution of Schubert's Steamroller in Many Sorted Logic. Unpublished paper, Department of Computer Science, University of Warwick.

Hoeksema, J. [1983] Plurality and conjunction. In ter Meulen, A. (ed.) *Studies in Modeltheoretic Semantics*, pp63-84. Dordrecht: Foris Publications.

Kaplan, R. M. [1987] Three Seductions of Computational Psycholinguistics. In Whitelock, P., Wood, M. M., Somers, H. L., Johnson, R. and Bennett, P. (eds.) *Linguistic Theory and Computer Applications*, pp149-188. London: Academic Press.

Kay, M. [1979] Functional Grammar. In *Proceedings of the Fifth Annual Meeting of the Berkeley Linguistic Society*, 1979, pp142-158.

Klein, E. and Sag, I. A. [1982] Semantic type and control. In Barlow, M., Flickinger, D. and Sag, I. A. (eds.) *Developments in Generalized Phrase Structure Grammar: Stanford Working Papers in Grammatical Theory*, pp1-25. Bloomington, Indiana: Indiana University Linguistics Club.

Mellish, C. S. [1988] Implementing Systemic Classification by Unification. *Computational Linguistics*, **14**, 40-51.

Meseguer, J., Goguen, J. A. and Smolka, G. [1987] Order-Sorted Unification. Technical Report No. CSLI-87-86, Center for the Study of Language and Information, Stanford, Ca. March 1987.

Pollard, C. and Sag, I. [1987] *An Information-Based Approach to Syntax and Semantics. Volume 1: Fundamentals.* Stanford, Ca.: Center for the Study of Language and Information.

Pollard, C. and Sag, I. A. [1988] An Information-Based Theory of Agreement. Report No. CSLI-88-132, Center for the Study of Language and Information, Stanford, Ca. September 1988.

Proudian, D. and Pollard, C. J. [1985] Parsing Head-driven Phrase Structure Grammar. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, University of Chicago, Chicago, Illinois. July 1985, pp167-171.

Shieber, S., Uszkoreit, H., Pereira, F. C. N., Robinson, J. J. and Tyson, M. [1983] The Formalism and Implementation of PATR-II. In Grosz, B. and Stickel, M. E. (eds.) *Research on Interactive Acquisition and Use of Knowledge*, SRI International, Menlo Park, 1983, pp39-79.

Smolka, G. [1988] A Feature Logic with Subsorts. LILOG-Report No. 33, IBM Deutschland GmbH, Stuttgart. May 1988.

Uszkoreit, H. [1987] A Flexible Type-Unification-Based Representation Formalism. In *Alvey/SERC Workshop on Natural Language Processing, Unification and Grammatical Formalisms*, University of Edinburgh. June 1987, pp1-2.

Walther, C. [1985] A mechanical solution of Schubert's steamroller by many-sorted resolution. *Artificial Intelligence*, **26**, 217-224.

Zeevat, H. [1988] Combining categorial grammar and unification. In Reyle, U. and Rohrer, C. (eds.) *Natural Language Parsing and Linguistic Theories*, pp202-229. Dordrecht: D. Reidel.