Arbeitspapiere des SFB 340
Bericht Nr. 134, April 1997

A PROPOSED LINEAR SPECIFICATION LANGUAGE

THILO GOETZ AND GERALD PENN

ABSTRACT. We propose a relaxation of context-free grammar rules to handle languages with weaker linear precedence constraints. This proposal will serve as the basis of the design and implementation of a specialised computational formalism for parsing with linearisation domains in these languages. Some issues pertaining to the off-line compilation of specifications are also discussed.

CONTENTS

1. MOTIVATION

In the last phase of the SFB 340/B4 project, we developed the ConTroll system (Goetz and Meurers 1995; Goetz and Meurers 1997), a constraint-resolution system for processing with typed feature-based grammars. This system, implemented in Prolog, compiles grammars into definite clauses, which are then transformed to optimise performance relative to a mode of processing (parsing or generation) and a mode of interpretation. It was developed in tandem with the HPSG German syntax fragment of project SFB 340/B8, and has been tested extensively by the implementors of that HPSG grammar.

Within any large-scale implementation task, of course, the goal is to recognise background knowledge in the empirical domain which can be brought to bear early enough to refine the search for a solution. As a

ISSN 0947-6954/97
© 1997 Thilo Goetz and Gerald Penn

grammar fragment grows larger, this goal will become increasingly more vital for continuing to ensure efficient processing.

To date, however, ConTroll only uses two specifically linguistic sources of knowledge in its compilation. The first is (partial) knowledge about the logic of typed feature structures. Using the restrictions inherent to the logic of King (1989), we can partially pre-compile common operations such as unification and feature access more than would otherwise be possible. Given that these are the basic operations with which processing at every level of the system occurs, even slight improvements to their efficiency yield enormous benefits to linguist users.

The second is knowledge about lexical rules (Meurers and Minnen 1997). The linguist user can define a set of lexical rules, which generalise over some set of lexical entries. This is particularly important as the size of the lexicon grows large. The meaning of these lexical rules is defined in terms of the underlying relational language of ConTroll in a straightforward way, transparent to the user. The lexical rule compiler, however, makes use of the special properties of lexical rules and compiles them into a more efficient internal representation, using finite state techniques to compute ahead of time the possible lexical rule interactions.

In the next phase of our project, we will extend our use of domain knowledge to linear precedence and linearisation domains. We will achieve this by designing and implementing a Linear Specification Language (LSL), a language just for immediate dominance and those word-order constraints. Specifically, LSL will be a conservative extension of phrase structure rules by a set of simple, near-atomic statements about linear precedence and domain formation, in order to be adequate for freer word-order languages in general, and the B8 fragment grammar in particular. At the same time, these near-atomic statements will be capable of identifying sources of complexity and sources of deterministic information to a compiler which can translate these rules into efficient code for parsing. LSL specifications, along with the rest of the grammar, will be compiled into Prolog code. The result will be a feature-based grammar development system which is capable of efficient parsing with the B8 fragment's linearisation-intensive grammar.

Linearisation grammars have recently emerged as a successor to ID/LP grammars in constraint based analyses of (semi-) free word order languages. In contrast to ID/LP grammars, there is no phrase structure skeleton on which the LP component operates. Instead, so-called linearisation domains form the basic units for determining word order, and it is these that LP constraints are defined on. Local phrase structure trees and linearisation domains do not necessarily coincide. Meurers et al. (1997) for example propose a grammar of German where the Mittelfeld has a binary branching phrase structure, yet the whole Mittelfeld is one linearisation domain. Word order in the Mittelfeld is thus totally independent from phrase structure.

Linearisation in the B8 fragment grammar was, in fact, a significant source of complexity for parsing in the last phase; and it is about to become much worse. The next phase of the SFB 340 (projects A3 and B8) will together consider phenomena which will present two sources, in addition to the Mittelfeld, of linearised freedom. The first is parentheticals. To date, there is no known analysis of parentheticals in German which reduces the problem to a context-free grammar; and it is unlikely that such an analysis will emerge, particularly one where the phrase structure rules have any natural correspondence to the assemblage of a semantic interpretation. Parentheticals are, at once, very free in their position, but not totally unconstrained, so that their position cannot simply be ignored.

The second source is data often analysed as cases of extraposition to the Nachfeld. These are found both in the verbal complex (e.g. incoherent verbal complements, non-verbal elements of the verbal complex), and at the clausal level of structure (e.g. verb-second complement clauses, free and complement daß clauses). Not only does a proper treatment of extraposition require a linearisation-based analysis, but arguably requires the extra notion of *liberation*, the ability for some constituents to break away from their otherwise cohesive domain objects.

2. Background: Linguistic

In Generalised Phrase Structure Grammar (GPSG, (Gazdar et al. 1985)), the notion of phrase structure grammar as a set of phrase structure rules was generalised to a set of Immediate Dominance (ID) and Linear Precedence (LP) rules. The ID/LP format allows the succinct description of constituent order variations of sister nodes in a local tree.

The ID/LP approach was in turn generalised by Zwicky (1986). Zwicky had the idea of *liberating* nodes from a local tree to allow them to be ordered at a higher level. His approach worked on a rule by rule basis. Rules were either ordinary ID-rules, or they were marked as liberating their daughter constituents. Dowty (1989) took up Zwicky's ideas, but made a careful distinction between *tectogrammar* (the rules that license linguistic structures) and *phenogrammar* (how the strings of substructures are combined when a rule is applied). This is not a useful distinction for a phrase structure or ID/LP grammar, since the only string combination operation for those kinds of grammars is concatenation. This distinction has also been successfully applied to a linearisation-based Serbo-Croatian HPSG by Penn (1997).

Dowty also added the notion of *bounding categories* to his system. Bounding categories do not allow their strings to merge with outside string material. Such restrictions can not be expressed in Zwicky's framework, since liberation there works on a rule-to-rule basis. Finally, Dowty introduced the distinction between *precedence* and *attachment*. Precedence constraints just require one node to precede the other, but other material may intervene. Attachment requires the yield of some node to *immediately* precede some other, so that no intervening material is allowed.

Within the HPSG community, linearisation grammars have recently emerged as a successor to ID/LP grammars for the analysis of (semi-) free word order languages. The ideas of this line of research are based on the work of Reape (1989, 1992), who replaced the phrase structure based notions of ID/LP grammars by so-called word-order domains. The extension to ID/LP grammars consists mainly in the fact that word-order domains are a non-hierarchical level of linguistic structure that allow non-sister constituents to be ordered. (Reape 1994) showed how word-order domains can be applied to an HPSG analysis of German clause structure, the verb cluster, and scrambling in the Mittelfeld.

(Pollard et al. 1993) build on the notions developed by Zwicky, Dowty and Reape. They propose to build a HPSG theory of linearisation and apply it to a wide range of data, including German extraposition phenomena. We will cooperate with the research group at the Ohio State University to ensure that our linearisation specification language is powerful enough to express the necessary constraints for the analyses developed by that group.

(Kasper et al. 1995) showed that the formal integration of word-order domains into HPSG theory was a non-trivial exercise. It is particularly the presence of structure sharing information and calls to relational goals in linear precedence constraints that pose difficult formal (and thus, computational) problems.

Decisive progress in terms of formal specification and empirical adequacy of HPSG linearisation grammars was made in the context of the B8 project. (Richter and Sailer 1995) shows how a theory of linearisation can be correctly formalised in SRL (Speciate Re-entrant Logic, (King 1994)), a logic specifically designed to be the formal basis of HPSG, which is also the basis of ConTroll. This theoretical proposal was then implemented in the linearisation component of the fragment grammar.

A different approach was taken by Manandhar (1995b), who showed that the feature logic of set descriptions (Manandhar 1993; Manandhar 1995a) can be used to encode linear precedence constraints over linearisation domains. The basic idea is for the user to provide a relational "feature," *prec*, which holds over two nodes, *a prec b*, provided the yield of *a* precedes the yield of *b*. With set descriptions, it provides two basic additional constraints: *domain inclusion*, in which one includes one feature structure's yield into another's, and *domain precedence*, in which two set-valued feature values stand in the *prec* relation. Notice that this work does not provide a theory of linearisation, but just the means in which to express some theories.

3. Background: Computational

The first computational generalisation of ID/LP grammars was an implementation which incorporates LP constraints into lexical entries (Engelkamp et al. 1992). The approach is interesting in that it allows the ordering of non-sister constituents. Engelkamp et al. (1992) give an example of a binary branching analysis of the German Mittelfeld, where it is still possible to express LP constraints on the complements and adjuncts. This is achieved by extending the ordering domains from local trees to so-called head-domains (the head and its complements and adjuncts).

In the first phase of the B4 project, Morawietz (1994) showed how standard ID/LP grammars could be implemented efficiently by using delaying techniques for evaluating LP statements. The implementation was integrated with the Troll system (Gerdemann et al. 1995).

The only implementation of a linearisation grammar other than the B8 Fragment grammar that we are aware of was written in the context of the RGR (Reusabiliy of Grammatical Resources, (Erbach et al. 1995)) project at the Universities of Saarbrücken and Edinburgh. The implementation is a combination of a constraint solver for (Manandhar 1995b) and the ProFIT system (Erbach 1994). Preliminary results with an experimental linearisation grammar, however, indicated slow run-time performance.

Our approach is distinguished from this in that linearisation statements from our specification language operate over the yields of phrase-structure-like trees, not over sets within the feature logic. The goal of Manandhar (1995b) was to provide a generalisation of feature logic itself to allow a declarative expression of linear precedence constraints. We are providing a means for *processing* linear precedence at the level of derivation trees to ensure efficiency, at the price of departing from the logical framework of HPSG itself. In this respect, our language falls between

Manandhar (1995b) and the rules used as input to the "head-corner" parser of van Noord (1991). This parser uses a head-driven, rather than string-position-driven approach to parsing, combined with a more general means of phonological combination, in order to capture discontinuous constructions. Unlike these rules, ours abstract away from data structures and string operations, and will handle a much wider range of linearisation-constrained theories.

Processing linear precedence with a set-augmented feature logic presents some challenges which we can easily avoid in our approach to linearisation and feature-based parsing. First, representing linear precedence as a feature, and order-underspecified linear domains as set values within the feature logic, involves linear order and domains in the operation of feature structure unification. In the case of feature-based linear precedence, one can only encode non-immediate precedence by adding functional uncertainty over feature paths to the description language. Although no proof yet exists, it is generally believed that this significantly extends the expressive power of the description language (e.g. it is possible to describe all and only lists of finite length), as well as the computational cost. In the case of sets, the only known implementation of this logic requires an all-pairs check over the number of sets in order to decide when to enforce set unification. Even then, there is an additional non-deterministic cost in how to combine two sets' values consistently. In our approach, domains are manipulated outside the feature structures themselves; and the unification of two domains is never required.

Second, as noted by Manandhar (1995b), immediate precedence can be difficult to enforce in a system which requires linearised models. For example, if $<<$ represents immediate precedence, then the expression:

$$(a << b) \wedge (c << d) \wedge (a < d) \wedge (c < b)$$

actually has a solution, in which a is unified with c and b is unified with d. In general, the most natural interpretation of feature-based set logic without arity constraints entails that a set with two elements, a, b, subsumes a set with one element, $a \wedge b$. The phonological yields we manipulate in our approach exist in a logic over derivation trees in which there is no unification of domain elements. In this respect, we are incorporating the linguistic knowledge that words in a string can automatically be taken as distinct. To do this in a feature logic would require the explicit addition of pairwise inequational constraints, such as ALE provides (Carpenter and Penn 1994).

By using linearisation constraints as annotations on phrase-structure-like rules, we are, at the same time, delineating a family of explicit control strategies, namely generalisations of the familiar parsing algorithms for context-free languages (e.g. Earley deduction, chart parsing, "head-corner" parsing), which, at any rate, bound search in that they specify linear precedence relative to specific statements about immediate dominance. Previous approaches have either required the user to use genuine phrase structure rules, or provided no control strategy whatsoever, simply a declarative specification. An example of the former would be the familiar parsing algorithms themselves; and, of the latter, feature-based precedence with set descriptions. In the latter case, of course, such declarations can be used inside phrase structure rules whose categories themselves are feature structures; but in this case, linearisation information is merely one undistinguished source among many at the level of feature structure unification, and does not figure in the selection of control strategy for its rules or their compilation. These two must be interleaved to ensure efficient processing.

This annotation method can also be distinguished from the context-free extraction and annotation method of Briscoe and Carroll (1993), which has been implemented in their Alvey Natural Language Tools. There, an atomic-category context-free backbone is extracted from a feature-based phrase structure grammar. The categories are drawn from the largest maximally specific set of disjoint descriptions which cover the descriptions used in the feature-based grammar. The remaining unifications required by the feature-based grammar are "attached" to this atomic backbone. These unifications are postponed even if they are completely deterministic, and cheap ways of constraining search, the reason simply being that LR(1) compilation, the technique they use on these atomic rules, has never been extended to the case of feature-based rules. In our approach, all feature unifications are still part of the categories in the phrase-structure-like rules, and are treated as part of them during compilation. The attachments consist only of linearisation constraints and possibly procedural attachments.

4. Background: Grammars and Systems

To the best of our knowledge, the Fragment grammar of B8 implemented in ConTroll is the only large, working linearisation grammar. The reader is referred to the Ergebnisbericht 1995–1997 of the SFB 340 for more details.

ALE (Carpenter and Penn 1994) is a parsing and logic programming system based on Carpenter's logic of typed feature structures (Carpenter 1992b). This system showed that compiling type signatures and appropriateness declarations off-line could yield very efficient Prolog code for unification of feature structures (which is then compiled further by a Prolog compiler). In the same manner, ALE also compiles phrase structure rules whose categories are feature structure descriptions, and expressions from a feature structure-based procedural attachment language into Prolog code. It also has a limited constraint resolver, and supports inequational descriptions and extensional typing.

The first phase of the B4 project developed Troll (Gerdemann and King 1994; Gerdemann, Goetz, Griffith, Kepser, and Morawietz 1995),[1] which has essentially the same functionality as ALE, only based on King's logic for HPSG, as formalised by (King 1989). Griffith (forthcoming) also provides a means for compiling named disjunctions in Troll.

The more recent phase of the B4 project concentrated on HPSG constraint resolution. The ConTroll system (Goetz 1997) permits the user to specify grammars directly with feature-based implicational constraints and functional attachments, augmented with delay statements. It transforms these constraints into expressions of a feature-based logic programming language, which are then further transformed into efficient parsing code. ConTroll also has a graphical development and debugging environment, and has been extensively used in the development of the B8 fragment grammar.[2]

The present proposal represents a novel compromise of these approaches. A phrase-structure-like backbone will be provided, but with generalised linear precedence, provided by a special declarative constraint language. All other principles can be provided either as implicational constraints or relational attachments. In this way, we hope to capture the efficiency which more rule-based approaches provide, while still retaining the convenience and modularity of a constraint-based system.

5. A Proposed LSL

An LSL should support efficient parsing algorithms, incorporate the basic insights of Zwicky (1986), Dowty (1989), Engelkamp et al. (1992), Pollard et al. (1993), Reape (1994), and Richter and Sailer (1995). An

[1]The reader is referred to the Ergebnisbericht 1992–1994 of the SFB 340 for a complete description.

[2]The reader is referred to the Ergebnisbericht 1995–1997 of the SFB 340 for a complete description.

LSL should be an extended phrase-structure-based formalism for describing the composition of constituents with linearisation. We envisage this formalism as having the form of a sequence of phrase-structure-like rules, augmented with an inventory of basic expressions for describing linear precedence constraints in natural language, composable with conjunction and disjunction.

In general, the essential tension that will shape the final structure of this language is between the need for sufficient generality to express realistic linearisation grammars on the one hand, and the need for computational tractability on the other. Although phrase-structure-based, our rule system will provide powerful linearisation constraints, thus making the transition from purely relational and constraint-based descriptions into this language relatively easy. It will thus subsume ID/LP and conventional phrase structure grammars as well. The specification of linearisation constraints relative to particular rules, however, provides a backbone along which linearisation constraints can be propagated for efficiency, and also significantly restrictions the range over which the constraints need to be considered. In particular, although constraints can, to a certain extent, be propagated up and down a derivation tree, they cannot be propagated across to sister trees. This small amount of extra structure can have large consequences for constraining the search for heads of domains or constituents, for example.

Figure 1 shows a preliminary proposal for how such a language could look. While the details of this language are, of course, very tentative at this point, it crucially supports the notions of liberation (Zwicky 1986), isolation (called *bounding* by Dowty (1989)), linear precedence and immediate precedence (called *attachment* by Dowty (1989)), and constraints such "nth from the left (or right)" (Dowty 1989).

Notice that this LSL (hereafter, called simply *LSL*) does not explicitly allow universal LP statements such as $A \prec B$. These statements actually have a number of possible interpretations; but the common ones do have correlates in LSL, and these can be placed on every rule in the grammar to achieve the same effect.

A few examples will illustrate the expressive power of LSL. Suppose we would like to encode the context-free rule $A \rightarrow B\ C\ B$. We could express this as (without explicit quantification)

$$
\begin{aligned}
& y_0 \rightarrow y_1\ y_2\ y_3. \\
& \qquad y_0 \sim A\ \wedge\ y_1 \sim B\ \wedge\ y_2 \sim C\ \wedge\ y_3 \sim B\ \wedge \\
& \qquad y_1 \ll y_2\ \wedge\ y_2 \ll y3
\end{aligned}
$$

$Yield:$		subtree yield
	$v,\ v \in V_y$	variable yield
	$w,\ w \in Phon \cup \{\epsilon\}$	phonological yield
$Desc:$		feature term
	$v,\ v \in V_d$	description variable
	$\sigma,\ \sigma \in Type$	type
	$f : d,\ d \in Desc$	feature selection
	$d_1 \wedge d_2,\ d_1, d_2 \in Desc$	conjunction
	$\neg\, d,\ d \in Desc$	negation
$Rule:$		phrase structure rule
	$\exists y_0, \dots, y_n \in Yield : y_0 \rightarrow y_1, \dots, y_n.c,\ c \in Cons$	
$Cons:$		constraint
	$\top$	truth
	$y_1 < y_2,\ y_1, y_2 \in Yield$	precedence
	$y_1 \ll y_2,\ y_1, y_2 \in Yield$	immediate precedence
	$y \sim d,\ y \in Yield,\ d \in Desc$	root description
	$\langle y_1 \rangle\, (: y_2),\ y_1, y_2 \in Yield$	isolation (and liberation)
	$y_1 = y_2,\ y_1, y_2 \in Yield$	equality
	$y_1 \neq y_2,\ y_1, y_2 \in Yield$	inequality
	$\stackrel{n}{\rightarrow} y,\ y \in Yield, n \in \mathcal{N}$	nth from left
	$\stackrel{n}{\leftarrow} y,\ y \in Yield, n \in \mathcal{N}$	nth from right
	$\forall y_1 \in y_2.c,\ y_2 \in Yield, c \in Cons$	universal quantification
	$\exists y_1 \in y_2.c,\ y_2 \in Yield, c \in Cons$	membership
	$\exists y_1, \dots, y_n \in Yield :$	decomposition
	$y_0 \asymp \{y_1 \dots y_n\}.c,\ y_0 \in Yield,\ c \in Cons$	
	$c_1 \wedge c_2,\ c_1, c_2 \in Cons$	conjunction
	$c_1 \vee c_2,\ c_1, c_2 \in Cons$	disjunction

FIGURE 1. Syntax of LSL (preliminary)

This rule licenses a local tree with 3 daughters, where the mother is an A, the daughters are a B, a C and a B, respectively, the y_1 daughter immediately precedes the y_2 daughter, and the y_2 daughter immediately precedes the y_3 daughter. The first thing to note is that our precedence rules operate on tree nodes, and do not express constraints like "all As must precede all Bs". Our next example will show how this can be done.

The other important thing to note is that this is a rule that cannot be expressed in an ID/LP grammar, since it violates the Exhaustive Constant Partial Ordering (ECPO) property (Gazdar, Klein, Pullum, and Sag 1985). We reject the idea of making the ECPO property part of the grammar formalism. The grammar writers should be free to enforce it if they want.

A more linguistic example shows a (much simplified) treatment of Dutch cross serial dependencies. Consider the following rules.

$$
\begin{aligned}
& y_0 \rightarrow y_1\ y_2\ y_3. \\
& \qquad y_0 \sim S \ \wedge\ y_3 \sim S \ \wedge\ y_1 \sim NP \ \wedge\ y_2 \sim V \ \wedge \\
& \qquad y_1 < y_2 \ \wedge\ y_1 < y_3 \ \wedge \\
& \qquad \forall x \in y_3.\ ((x \sim NP) \rightarrow (x < y_2)) \ \wedge \\
& \qquad \forall x \in y_3.\ ((x \sim V) \rightarrow (y_2 < x))
\end{aligned}
$$

$$
\begin{aligned}
& y_0 \rightarrow y_1. \\
& \qquad y_0 \sim S \ \wedge\ y_1 = \epsilon
\end{aligned}
$$

The first rule describes the recursive construction of cross serial dependencies, where a NP y_1 and a verb y_2 are added to a series of NPs and verbs y_3. The third line says that the new NP must precede all other material. The fourth line says that the new verb must follow all NPs in y_3, and the final line requires the new verb to precede all other verbs. Notice that this is much stronger than standard ID/LP format, where one could require all NPs to precede all verbs, but not in which order. The second rule just provides the base case to recursive definition.

As a final example, consider the following set of rules. It encodes the optional right extraposition of $\bar{S}$ complements of nouns in German.

$$
\begin{aligned}
& 1)\ y_0 \rightarrow y_1\ y_2. \\
& \qquad y_0 \sim NP \ \wedge\ y_1 \sim \bar{N} \ \wedge\ y_2 \sim \bar{S} \ \wedge \\
& \qquad \stackrel{1}{\leftarrow} y_2 \ \wedge \\
& \qquad (\langle y_0 \rangle \ \vee\ \langle y_0 \rangle\ : y_2)
\end{aligned}
$$

$$
\begin{aligned}
& 2)\ y_0 \rightarrow y_1\ y_2. \\
& \qquad y_0 \sim S \ \wedge\ y_1 \sim NP \ \wedge\ y_2 \sim VP
\end{aligned}
$$

$$
\begin{aligned}
& 3)\ y_0 \rightarrow y_1\ y_2. \\
& \qquad y_0 \sim VP \ \wedge\ y_1 \sim V \ \wedge\ y_2 \sim NP
\end{aligned}
$$

$$
\begin{aligned}
& 4)\ y_0 \rightarrow y_1\ y_2. \\
& \qquad y_0 \sim \bar{S} \ \wedge\ y_1 \sim Comp \ \wedge\ y_2 \sim S \ \wedge\ y_1 \ll y_2 \ \wedge
\end{aligned}
$$

$$\exists x \in y_2 .\ x \sim V \ \wedge \ (\ (\exists y \in y_2 .\ y \sim \bar{S} \wedge \stackrel{2}{\leftarrow} x) \vee$$
$$(\forall y \in y_2 .\ y \sim \neg \bar{S} \wedge \stackrel{1}{\leftarrow} x)\ )$$

5) $y_0 \rightarrow y_1$.
$$y_0 \sim \bar{S} \ \wedge \ y_1 \sim S \wedge$$
$$\exists y \in y_1 .\ y \sim V \wedge \stackrel{2}{\rightarrow} y$$

Rule 1) combines an $\bar{N}$ with its $\bar{S}$ complement. The $\bar{S}$ must be the rightmost element in its domain. However, what its domain is remains underspecified at this point. If the whole NP is isolated ($\langle y_0 \rangle$), then the $\bar{S}$ must be adjacent to the $\bar{N}$. Otherwise, only the $\bar{N}$ is isolated, and the complement sentence liberated. This has the effect that whichever domain the $\bar{S}$ finally ends up in, it will be realised as the rightmost constituent. Rule 2) combines a VP with its subject, and rule 3) a verb with its object (we only treat transitive verbs for the purposes of this example). Notice that neither rule specifies anything about linear precedence. Rule 4) combines a S with a complementiser to give an $\bar{S}$. The complementiser must immediately precede the S. The rest of the rule says that if there is a free $\bar{S}$ floating around, then the verb takes second but last position. The $\bar{S}$ comes last. Otherwise, there is no $\bar{S}$, and the verb comes last. Rule 4) finally licenses V2 clauses.

It is interesting to note how our approach to linearisation is different from the one of Manandhar (1995b). The main difference is that Manandhar (1995b) includes domain objects in the feature logic, rather than at a separate level. This is also the approach we took in the last phase, which turned out to be simply too slow for grammars of the size we processed. Instead of providing a completely general linear precedence module, we propose to integrate linearisation constraints with a phrase structure backbone, thus providing a basis for efficient processing. In fact, the language we propose is designed for parsing; although, because of its declarative definition, we expect it to be easily adaptable to generation.

On the other hand, the feature logic is still available for other non-linearised dependencies such as a SLASH-passing mechanism, quantifier storage, argument raising etc. Each node in a derivation tree produced by these rules is a typed feature structure, over which principles both inside and outside the logic operate.

The Manandhar (1995b) approach also provides no primitive for liberation, a notion built into our language. In both cases (linear precedence and liberation), we provide special primitives for these outside the logic of typed feature structures. This will allow us to provide special run-time support for these computationally difficult notions.

By providing linearisation constraints on tree nodes, we solve another problem of (Manandhar 1995b): we have no problem with linearisable models and thus, immediate precedence. This is simply a consequence of the fact that we impose a phrase structure backbone, thus enforcing linearised models.

The interpretation of the phrase structure-like rules to which the linearisation constraints are attached is one of otherwise unconstrained immediate dominance schemata. In order to encode a normal set of phrase structure rules, one would need to place appropriately strong linear precedence constraints on the daughters of each of them. It is important to remember, with expressions from this language, that more constraints do not necessarily mean more work. Some of these linear precedence constraints, such as isolation and immediate precedence in particular, *constrain* search; and, in general, more of these actually lead to greater efficiency in a traditional chart parser. As a result, there will naturally be rule systems from this augmented language which are inherently intractable to parse. The object of defining this language is to make explicit those constraints which can be used to improve the situation.

Previous formal approaches to semi-free word order languages have taken either of two tracks. One has been to ignore the free word order generalisations, use only phrase structure rules, and simply write a multitude of them to achieve descriptive adequacy. The disadvantage of this approach is clearly that linguistic generalisations are lost, grammar writing is more difficult, and the grammar explodes in size. The other approach has been to abandon positional encoding (e.g., "the verb comes second") and only allow relative ordering statements (e.g., "all As precede all Bs"). We believe that for both linguistic and computational purposes, a combination of these approaches promises the best results. If a rule naturally makes a statement about a fixed position, then the formalism should be able to express this directly. If, on the other hand, one needs to talk about relative order, then one needs to be able to do so.

6. Compilation issues

LSL specifications, together with any attached relational or implicational constraints, can be compiled into Prolog code. The compilation takes place relative to a control strategy, which for the purposes of this project, will only be parsing.

Our goal is ultimately to provide portable compiled code for parsing with large-scale grammars which is as efficient as those grammars' linear constraints allow them to be. In particular, we expect the constraints from natural languages to admit efficient algorithms. The auxiliary attachments on the rules from LSL are simple, near-atomic statements about linear precedence. These provide as conservative of an extension to phrase structure as possible to be adequate for freer word-order languages, while identifying to the next compilation step the sources of deterministic information to be used to constrain search, and the sources of complexity to be postponed.

Techniques from constraint logic programming and head-driven, or so-called "head-corner," parsing can be applied to the compilation of LSL specifications. LSL does not actually have a bias towards a parsing strategy. Head-driven approaches are particularly attractive in the context of HPSG, whose heads contain the necessary subcategorisation frames.

At compile-time, the first task is to determine what mother-daughter combinations of phrase structure rules are consistent with the constraints given. This is very similar to the pre-compilation of lexical rules devised for ConTroll,[3] and also to Torisawa (1996). Having found those combinations, the next step is to *propagate* constraints, given a particular combination of mother and daughter, in order to maximise efficiency. Finding a language-preserving propogation strategy which improves efficiency will be one of the major tasks of the proposed phase of this project. Here the object is to push the more deterministic and computationally cheaper constraints *down* the derivation tree (i.e. towards the terminal nodes, which correspond to lexical items), and the more non-deterministic or computationally expensive constraints *up* the derivation tree, as much as possible. It is also important to perform this propogation efficiently at compile-time (naïvely considering all possible local subtrees, for example, could be quite expensive if the rules have large branching factors).

Which constraints are expensive or cheap will be determined, of course, by the final structure of LSL. A brief glance at our preliminary syntax on page 10, however, already gives some indication. Root descriptions, which are always non-disjunctive, play a very important role in refining search, and must be pushed down. That there is any flexibility to make such a push should be evident from the existence of constraints such as membership or decomposition, which allow reference to yields further down than the immediate daughters of a given rule.

[3]The reader is referred to the Ergebnisbericht 1995–1997 of the SFB 340 for a complete discussion.

Isolation must also be pushed down. Isolation corresponds to "bounding" or "compaction" in linearisation-based theories — it at least partially prevents scrambling, and therefore refines search. Immediate precedence, and to a lesser extent, general linear precedence, should also be pushed down. Membership, on the other hand, is inherently non-deterministic, and should generally be pushed up towards the root of the derivation, i.e. postponed. Universal quantification is not necessarily non-deterministic, but can be quite expensive, especially over large, unisolated domains, and must also be postponed.

Once these constraints have been properly distributed, the compiler must then generate code that enforces them inside a roughly head-driven parsing strategy. At run-time, this code must seek out potential heads, identify the smallest possible isolated domains in which their arguments can be guaranteed to be found, and find those arguments, while collecting linear precedence information throughout to ensure consistency with the grammar. Argument and adjunct combination rules may also need to be distinguished by the user in order to avoid problems with potentially unbounded adjunction. It may be possible to identify certain cases automatically.

It is, of course, not unreasonable to expect that, in those cases where the linearisation constraints are strong enough to render a rule equivalent to a traditional phrase structure rule, that the system treat the rule as such. This is also one of our goals — to avail ourselves of true phrase structure, where it exists, even if the entire grammar may not be a phrase structure grammar. In the case of the B8 fragment grammar, for instance, the structures of determiner phrases and the verbal complex are analysed without recourse to linearisation or scrambling. In these cases, once a DP or VC head is detected, parsing of those constituents should proceed as in a normal head-driven parser.

With respect to both our linear specification language (versus Manandhar (1995b)), and our procedural attachment language (versus other theoretical appeals to relations), the object is to remove information from the feature logic itself, where it is expressed declaratively, and explicitly relocate that information to a derivation rule, where it can be enforced more efficiently, using linear precedence and immediate dominance as guides.

References

Briscoe, T. and J. Carroll (1993). Gernalized probabilistic lr parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics 19*(1).

Carpenter, B. (1992b). *The logic of typed feature structures*, Volume 32 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

Carpenter, B. and G. Penn (1994). ALE – The Attribute Logic Engine, User's Guide, Version 2.0.1, December 1994. Technical report, Carnegie Mellon University.

Dowty, D. (1989). Toward a minimalist theory of syntactic structure. Paper prepared for the Tilburg Conference on Discontinous Constituency. Corrected version February 24, 1992.

Engelkamp, J., G. Erbach, and H. Uszkoreit (1992). Handling linear precedence constraints by unification. In *30th annual meeting of the ACL*, pp. 201–208.

Erbach, G. (1994). Profit: Prolog with features, inheritance and templates. Technical report, CLAUS.

Erbach, G., M. van der Kraan, S. Manandhar, H. Ruessink, C. Thiersch, and W. Skut (1995). Extending unification formalisms. In *Proceedings of the 2nd Language Engineering Convention*, London.

Gazdar, G., E. Klein, G. Pullum, and I. Sag (1985). *Generalized Phrase Structure Grammar*. Harvard University Press.

Gerdemann, D., T. Goetz, J. Griffith, S. Kepser, and F. Morawietz (1995). *Troll manual* (draft ed.). Seminar für Sprachwissenschaft, Universität Tübingen.

Gerdemann, D. and P. J. King (1994). The correct and efficient implementation of appropriateness specifications for typed feature structures. In *Proceedings of COLING-94*, Kyoto, Japan.

Goetz, T. (1997). *The ConTroll manual and user's guide* (Draft ed.). Seminar für Sprachwissenschaft, University of Tübingen.

Goetz, T. and W. D. Meurers (1995). Compiling HPSG type constraints into definite clause programs. In *Proceedings of the Thrirty-Third Annual Meeting of the ACL*, Boston. Association for Computational Linguistics.

Goetz, T. and W. D. Meurers (1997). Interleaving universal principles and relational constraints over typed feature logic. In *Proceedings of the 35th Annual Meeting of the ACL and the 8th Conference of the EACL*, Madrid, Spain.

Griffith, J. (forthcoming). *DUSt: a disjunctive approach to unification*. Ph. D. thesis, Universität Tübingen.

Kasper, R., A. Kathol, and C. Pollard (1995). A relational interpretation of linear precedence constraints. In *Proceedings of the workshop on the mathematics of language*.

King, P. J. (1989). *A logical formalism for head-driven phrase structure grammar.* Ph. D. thesis, University of Manchester.

King, P. J. (1994). An expanded logical formalism for head-driven phrase structure grammar. Arbeitspapiere des SFB 340 Nr. 59, Universität Tübingen.

Manandhar, S. (1993). *Relational Extensions to Feature Logic: Application to Constraint Based Grammars.* Ph. D. thesis, Ph.D. thesis, Department of Artificial Intelligence, University of Edinburgh.

Manandhar, S. (1995a). An attributive logic of set descriptions and set operations: Extended report. *Edinburgh Working Papers in Cognitive Science Nonclassical Feature Systems*(10), 39–60.

Manandhar, S. (1995b). Deterministic consistency checking of LP constraints. In *proceedings of the 7th conference of the EACL*, Dublin, Ireland.

Meurers, D., and G. Minnen (1997). A computational treatment of lexical rules in HPSG as Covariation in Lexical Entries. *Computational Linguistics 23*(4).

Meurers, D., F. Richter, M. Sailer, and H. Winhart (1997). Ein HPSG-Fragment des Deutschen, Teil 1: Theorie. Arbeitspapiere des SFB 340, Universität Tübingen.

Morawietz, F. (1994). Formalization and Parsing of Typed Unification-Based ID/LP Grammars. Master's thesis, Universität Tübingen.

Nerbonne, J., K. Netter, and C. Pollard (Eds.) (1994). *German in Head-Driven Phrase Structure Grammar.* CSLI Publications. Lecture Notes 46.

Penn, G. (1997). Multiple wh-extraction and clitic placement in serbocroatian. Linguistics Qualifying Paper, Carnegie Mellon University.

Pollard, C., R. Kasper, and R. Levine (1993). Studies in Constituent Ordering. Toward a Theory of Linearization in Head-Driven Phrase Structure Grammar. Research Proposal to the National Science Foundation, Ms. Ohio State University.

Reape, M. (1989). A logical treatment of semi-free word order and bounded discontinuous constituency. In *Proceedings of the 4th Conference of the European Chapter of the ACL*, pp. 103–110.

Reape, M. (1992). *A Formal Theory of Word Order: A Case Study of West Germanic.* Ph. D. thesis, University of Edinburgh.

Reape, M. (1994). Domain Union and Word Order Variation in German. In (Nerbonne, Netter, and Pollard 1994), pp. 151–197.

Richter, F. and M. Sailer (1995). Remarks on Linearization. Reflections on the Treatment of LP-Rules in HPSG in a Typed Feature

Logic. Magisterarbeit, Seminar für Sprachwissenschaft, Universität Tübingen.

Torisawa, K. (1996). Off-line raising, dependency analysis, and partial unification. In *Third Annual Conference on Natural Language Processing*.

van Noord, G. (1991). Head corner parsing for discontinuous constituency. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pp. 114–121.

Zwicky, A. (1986). Concatenation and liberation. In *Proceedings of the Twenty-Second Regional Meeting of the Chicago Linguistic Society*, pp. 65–74.