Ergebnisbericht

Teilprojekt	B4
Thema	Constraints on Grammar for Efficient Generation
Leiter	Prof. Dr. Erhard W. Hinrichs Dr. Dale Gerdemann
Mitarbeiter	Dr. Paul John King Guido Minnen Thilo Götz

Inhalt

1 Einleitung

The aim of our project, as given in the Finanzierungsantrag and supplement, was to investigate the problem of efficient generation using a unification grammar. More specifically, the HPSG (head-driven phrase structure grammar) of Pollard and Sag (1987) had emerged as a de facto standard grammatical framework for computational linguistics whose uniform representation of syntax, semantics and phonology offered an ideal platform for the comparatively neglected study of generation. We proposed to investigate the consequences for computational efficiency of applying head-driven and other generation techniques to a HPSG account of those German constructions that are known to be problematic for unification-based generation algorithms. Such constructions include those in which the head is either displaced or empty, such as the V2 position of finite verbs and the topicalisation of nonfinite verbs (see Gerdemann and Hinrichs (in press)).

In outline, our project strategy was as follows: choose a mathematical formalism to underlie HPSG, choose a basic processor that is correct with respect to this formalism, incorporate various generation techniques into the basic processor to yield specialised processors configured for efficient generation, construct HPSG grammars of linguistic phenomena known to be problematic for generation, implement these grammars on the specialised processors, and evaluate the performance of the specialised processors with respect to the problematic phenomena. Many sections of our report reflect our strategy in that they comprise four subsections, where the first two subsections concern the development of the mathematical formalism and the basic processor, the third concerns the incorporation of various generation techniques into the basic processor, and the fourth concerns the formulation of HPSG analyses of linguistic phenomena that traditionally present problems for generation.

2 Kenntnisstand bei der letzten Antragstellung und Ausgangsfragestellung

2.1 A HPSG Formalism

As part of our project strategy, we sought an underlying mathematical formalism for HPSG with which to evaluate candidate HPSG processors. Unfortunately, a schism over HPSG formalisms significantly complicated our search. Pollard & Sag (1987) originally provided HPSG with a formalism based upon a Heyting algebra of feature structures that model partial information about linguistic entities, but King (1989) proved that this formalism does not always yield exactly the implication and negation of partial information that HPSG wants in order to express conditional linguistic constraints like the head-feature principle. Subsequently, HPSG was furnished with formalisms based upon the KRL (Kasper-Rounds Logic) of Rounds & Kasper (1986) and the AVL (attribute-value logic) of Johnson (1988). Despite apparent similarities, KRL and AVL are incompatible logics that yield equally incompatible HPSG formalisms. To illustrate this incompatibility, we here sketch both kinds of formalism.

KRL provides a set of feature structures, a set of formulae, and a satisfaction relation between the feature structures and the formulae. The set of formulae is closed under both conjunction and disjunction, and a feature structure satisfies the conjunction (disjunction) of two formulae iff the feature structure satisfies the first formula and (or) the second. KRL also provides a mechanism for drawing from the class of feature structures that satisfy a formula a finite set of feature structures that represents the formula inasmuch as a feature structure satisfies a formula iff some member of the representation of the formula subsumes the feature structure. KRL tacitly considers a feature structure to be a model of partial information, and the representation of a formula models some partial information iff some member of the representation models the partial information. KRL holds that the representation of the conjunction (disjunction) of two formulae models the conjunction (disjunction) of the partial information that the representations of the individual formulae model.

The KRL of Rounds & Kasper (1986) falls short of a HPSG formalism on two counts. Firstly, a feature structure models no type information. Secondly, the logic enables no implication or negation of partial information. Moshier (1988) created a typed KRL that overcomes the first shortfall, but a conflict between the semantics assumed by HPSG and the ontology assumed by KRL precludes a full remedy of the second. On the one hand, HPSG wants partial information to behave classically: the negation of some partial information is about an entity iff the partial information is not about the entity. On the other, every formula has a representation only if KRL meets a condition that we call subsumption monotonicity: if a first feature structure both satisfies a formula and subsumes a second feature structure then the second feature structure satisfies the formula. One can certainly envisage a classical extension of KRL in which the formulae are closed under negation, and satisfaction is extended so that a feature structure satisfies the negation of a formula iff the feature structure does not satisfy the formula. Unfortunately, though Rounds & Kasper (1986) showed that KRL is subsumption monotonic, Moshier & Rounds (1987) showed that the classical extension of KRL is not. Fortunately, by modifying the feature structures and their subsumption, and limiting negation to certain kinds of formulae, Carpenter (1992) constructed a classical (albeit restricted) yet subsumption-monotonic extension of Moshier's typed KRL.

The assumption that a feature structure models partial information underlies KRL. However, AVL assumes that the only aspect of this partial information that a linguist uses is whether the partial information is true or false of an entity. Consequently, AVL abandons feature structures that model partial information in favour of formulae that are interpretable as true or false of entities. AVL provides a set of entities, a set of formulae, and a description relation between the formulae and the entities. The set of formulae is closed under negation, conjunction and disjunction, the negation of a formula describes an entity iff the formula does not describe the entity, and the conjunction (disjunction) of two formulae describes an entity iff the first formula and (or) the second formula describe the entity.

KRL and AVL seem very alike: KRL and AVL formulae look similar, and AVL description appears to be the dual of KRL satisfaction. However, this apparent similitude belies a fundamental and irreconcilable difference between KRL and AVL. An AVL formula describes an entity, whereas a KRL formula can be said to describe an entity iff some feature structure both satisfies the formula and models true partial information about the entity. This difference is no mere logical nicety. The use of feature structures to model partial information precludes an unrestricted classical extension of KRL. By dispensing with feature structures and interpreting formulae directly, AVL provides exactly the classical semantics that HPSG seeks, without restrictions. The AVL of Johnson (1988) had no types, but Smolka (1988) extended AVL to include types, and King (1989) tailored this extension to include the HPSG appropriateness conditions.

At the time our project began, the KRL and the AVL based HPSG formalisms exhibited complementary strengths and weaknesses. The KRL-based formalisms promised ready access to a large stock of computational techniques to represent, test and process feature structures, whereas the AVL-based formalisms offered a pellucid relationship between the formulae and the entities. While the lucidity of the AVL-based formalisms had great appeal for us, we were not blind to the computational advantages of the KRL-based formalisms. Ultimately, however, the fidelity of a formalism to HPSG was to be our arbiter. We would choose one of the plethora of formalisms as the project formalism on the basis of the answer to the question:

Which formalism is most faithful to HPSG?

2.2 A HPSG Processor

Two basic approaches to implementing unification grammars had been followed prior to the time our project started. The first, exemplified by the PATR-II (parse and translate-II) of Shieber (1985), and the Unicorn processor of Gerdemann & Hinrichs (1988) and Gerdemann (1991), used rule-based techniques. The second, exemplified by the KBL (knowledge base language) of Aït-Kaci (1985), the parser of Franz (1990), and the TFS (typed feature system) of Zajac (1992), employed general constraint-solving techniques. Each approach had advantages and disadvantages for us. The advantage of the constraint-solving approach was its proximity to the declarative formalism used by HPSG. Also, as a declarative processor, this approach is very much direction neutral: both parsing and generation can be seen as solving the same set of constraints. The advantage of the rule-based approach was that it allows the use of standard parsing algorithms, and is thus much more efficient than general constraint solving.

Since we were concerned with issues of efficient processing, and since we already had considerable expertise with rule-based processors via the work of Gerdemann and Hinrichs on Unicorn, we preferred a rule-based basic processor. Unicorn was based on the extensions to Earley's algorithm described in Shieber (1985), though with several additional extensions which make it quite efficient for both parsing and generation. A major source of inefficiency in PATR-II and other chart-parsing approaches is the subsumption check that must be performed before adding new edges to a chart. Unicorn minimises this expense by performing the subsumption test on the restricted feature structures used in the prediction step as described in Gerdemann (1989). Unicorn also maintains efficiency by employing an efficient Prolog representation of feature structures in which features are kept in lexicographic order and re-entrancies are marked with Prolog variables. The advantages of this approach are that destructive binding of variables can be avoided, resulting in less structure copying, and that cyclic structures can be easily represented.

At the time our project started, constraint- and rule-based processors differed in another respect: constraint-based processors such as TFS made more use of types than did rule-based processors. The rule-based CUF (comprehensive unification formalism) processor of Eisele & Dörre (1990) admittedly had a notion of typing, but it was not specifically designed for HPSG. In fact, at the beginning of the project, no rule-based processor employing HPSG types existed. The foregoing raised the question:

> Can we exploit the ideas embodied in Unicorn and other untyped rule-based processors to provide a typed rule-based processor that is correct with respect to our chosen HPSG formalism?

In answering this question, we were subsequently greatly aided by the ALE (attribute logic engine) of Carpenter (1993), a rule-based processor that both implements the formalism of Carpenter (1992) and emphasises very efficient rule-based processing with typed feature structures.

2.3 Generation

A central problem for natural language processing, especially tactical generation, is the *control problem*: the restricting information available when evaluating particular subparts of a construction might be insufficient to prevent massive nondeterminacy, leading to inefficiency or even nontermination. The approaches to generation known at the start of our project were divided between those that attempt to overcome the control problem dynamically by on-line amendment of the evaluation strategy, and those that attempt to overcome it statically during compilation. The on-line approaches try to achieve efficient parsing and generation with a single (possibly procedurally annotated) representation of linguistic knowledge, whereas the off-line approaches try to compile a declarative grammar specification into a representation that is suited to processing in a particular direction (thus, for generation, an input grammar is compiled into a semantically equivalent grammar that can be efficiently used if the logical form of the natural language expression to be generated is known.)

Both of the on-line generation algorithms that were predominant when our project began are modifications of traditional parsing algorithms. One is the head-corner algorithm of Shieber et al. (1990), a generalisation of the Prolog style left-corner parser of Matsumoto et al. (1983), and the other is Earley's algorithm configured for generation. The novelty of the head-corner approach is that it starts bottom-up from the head. Using the head as the pivot for generation is an important idea, but it also restricts the form of a grammar, since it works best if most of the grammar rules are *chain-rules*, rules in which the semantics of the head is reentrant with the semantics of one of the daughters. Fortunately, this is a major feature of lexically based grammars like HPSG. Shieber (1988) gave the first use of Earley's algorithm for generation, but this algorithm does not use the prediction step to restrict feature instantiations on the predicted phrases, and thus lacks goal directedness. Though Gerdemann (1991) showed how to modify the restriction function to make top-down information available for the bottom-up completion step, Earley generation with top-down prediction still has an efficiency problem in that generating the subparts of a construction in the wrong order might make insufficient information available to restrict generation. Gerdemann (1991) partly overcame this problem by incorporating a head-driven strategy into Earley's algorithm.

Both head-corner and Earley generation encounter efficiency problems if the head of a construction provides insufficient information to restrict the evaluation of the dependent subparts of the construction, and this latter is generally the case when the head of a construction is either missing, displaced or underspecified. Furthermore, Martinovic & Strzalkowski (1992) and others have observed that a simple head-first reordering of the grammar rules may still lead to massive nondeterminacy unless the form of the grammar is restricted to unary or binary rules.

The off-line approach is represented by the EAA (essential arguments approach) of Strzalkowski (1990a, 1990b, 1991, 1993). This is a top-down approach to parsing and generation that uses off-line compilation. Strzalkowski presents the EAA as a means to invert logic grammars already oriented to a particular processing direction. The inversion process consists of both the automatic static reordering of nodes in the grammar, and the interchanging of arguments in recursive procedures. It is based on the notion of *essential arguments*, arguments which must be instantiated to ensure the efficient and terminating execution of a node.

The EAA inverts a logic grammar in two stages. Firstly, virtually all permutations of the grammar (possible orderings of the nodes in the grammar) are investigated in a bottom-up fashion via a precalculation of the MSEAs (minimal sets of essential arguments) of their nodes. This precalculation is based on a dataflow analysis of the grammar. Secondly, the optimal order of evaluation is determined by ensuring that only those nodes which have at least one of their MSEAs instantiated are called. This is achieved by *intraclausal* and *interclausal* reordering of the nodes in the grammar, where an intraclausal reordering is a reordering of the nodes within a rule of the grammar, and an interclausal reordering is a reordering of nodes between clauses, akin to the fold/unfold transformation of Tamaki & Sato (1984) and the partial execution in Pereira & Shieber (1987). Strzalkowski suggests that, in certain recursive procedures, an *argument series* can be identified, such as a list value that grows with each recursive call. By the use of interclausal reordering to reverse this argument series, certain nontermination problems can be solved. This is particularly effective for so-called *deadlock-prone rules*, rules in which one right-hand-side category that is supposed to provide information to restrict the evaluation of another right-hand-side category is recursively defined. Unfortunately, the actual inversion proposed by Strzalkowski was incorrect since it changed program semantics. A different inversion was needed. This was developed in Minnen & Gerdemann (1993) and Minnen et al. (1993).

The EAA is interesting in that using dataflow analysis to determine the suitability of a given evaluation order enables the EAA to determine a full reordering of the rules in a grammar independent of any head. However, the EAA is a generate-and-test approach, since the precalculation of the MSEAs demands the investigation of almost all possible permutations of a grammar. This is computationally infeasible. Moreover, intraclausal reordering together with interchanging arguments in recursive procedures fails to guarantee that input and output grammars are semantically equivalent.

Strzalkowski presented the EAA as an approach for inverting parsers into generators. However, the EAA can just as well take a declarative grammar specification as input. Dymetman & Isabelle (1988) took this theoretically more interesting perspective as the basis for their approach. Though closely resembling the EAA, this approach accomplishes the static reordering of the rules in the grammar not automatically, but by manually annotating the grammar rules. Dymetman et al. (1990) and Dymetman (1992a, 1992b) adopted the same theoretical perspective. However, Dymetman et al. (1990) tackled the nontermination part of the control problem by compiling a DCG (definite clause grammar) into generalised Greibach normal form, a non-left-recursive and declaratively-equivalent form of the input grammar.

Eliminating left recursion solves the nontermination problem for top-down parsing of off-line parsable grammars. However, Gerdemann (1991) observed that general recursion can be just as problematic as left recursion for top-down generation. Such observations led Dymetman et al. (1990) to employ *guides* to ensure termination in a manner similar to that of the EAA. Severe restrictions on the form of grammars are used to circumvent efficiency problems.

Our project began with the following questions concerning generation:

> Can we develop automatic generation techniques for typed feature structure-based processors that reduce or avoid the difficulties that empty or displaced heads have hitherto posed for head-driven generation techniques?
>
> Can we optimise these generation techniques to support true bidirectionality, with comparable efficiency in both directions?
>
> Can we incorporate the optimised techniques into the project processor?
>
> How well do the incorporated techniques perform when generating from the project grammars?

2.4 Linguistic Analyses

In order to provide a testbed for the generation algorithms to be developed in our computational linguistics research, the development of a HPSG grammar fragment for German was an important goal for the linguistic research to be carried out during the project. Accordingly, the coverage of this fragment was intended to emphasise those constructions which are of theoretical interest for automatic natural language generation. In keeping with our project proposal, we principally focussed upon those German constructions – such as the V2 position of finite verbs and the topicalisation of nonfinite verbs – that have been analysed in generative grammar by means of empty heads.

Several HPSG analyses of pertinent German constructions provided a natural starting point for further investigation: the work of Pollard (1990) and Nerbonne (1991) on the V2 position and on topicalisation, and our previous work in Hinrichs & Nakazawa (1989a, 1989b, 1991) on the verb complex. However, these analyses differ significantly over fundamental assumptions about the constituent structure of German. Our work assumed a hierarchical structure for the German Mittelfeld, with NP complements being added one at a time, whereas Pollard (1990) and Nerbonne (1991) proposed a flat-structure. Moreover, in our work finite auxiliaries in subordinate clauses were generated within the verbal complex, whereas Nerbonne and Pollard treated finite auxiliaries as sisters of S. Moreover, Pollard offered an analysis for all relevant clause types, whereas our work concentrated on verb-final structures and particularly on the auxiliary-flip construction, of which neither of the other two analyses offered an account. On the other hand, the proposals of Nerbonne and Pollard were able to account for a wider range of topicalisation data, albeit at the expense of having to introduce spurious ambiguity (Pollard) into the grammar or at the expense of introducing a proliferation of lexical specifications for auxiliaries (Nerbonne).

Though the scope and detail of these analyses differ considerably, the basic techniques they employ overlap substantially. Long-distance dependencies such as topicalisation constructions are treated by the percolation of slash features, familiar from categorial grammar and imported into HPSG from the treatment of unbounded dependencies in the GPSG (generalised phrase structure grammar) of Gazdar et al. (1985). For the treatment of the verbal complex in German, all three analyses employed the notion of argument raising first proposed by Hinrichs & Nakazawa (1989b). This allows the raising of NP complements and the construction of complex verbal constituents without modifying the subcategorisation principle of Pollard & Sag (1987).

On the basis of the foregoing, four questions arose:

> How can our previous work be generalised to analyse the constructions that are relevant to our project? More specifically:
>
> > How can our account of subordinate clauses be generalised to a treatment of verb-initial and verb-second constructions?
> >
> > How can our account of topicalisation be extended to account for a larger range of topicalisation data, especially the topicalisation of partial verb phrases?
>
> To what extent can we integrate the HPSG analyses of Pollard and Nerbonne into our account of the topicalisation and verb-second constructions?
>
> To what extent can we incorporate the HPSG analyses of other German linguistic phenomena into our fragment? The work of Uszkoreit (1987) on word order and Netter (1991) on the structure of noun phrases were particularly relevant in this regard.
>
> To what extent can we assimilate the insights of analyses of German within the GB (government and binding) of Chomsky (1981), such as Thiersch (1978) and Haider (1990), which treat the topicalisation and verb-second constructions in terms of empty heads? (This assimilation would await the results of project A6.)

3 Angewandte Methoden

3.1 A HPSG Formalism

Our initial problem as regards a HPSG formalism was to choose between the many HPSG formalisms available at the beginning of the project on the basis of their fidelity to HPSG. Since the formalisms of King (1989) and Carpenter (1992) were arguably the foremost AVL- and KRL-based HPSG formalisms available at the time our project started, our problem was reduced to choosing between the King and the Carpenter formalisms. In making this choice, we were greatly aided by a visit to the SFS (Seminar für Sprachwissenschaft) by Carl Pollard, and the drafts of Pollard & Sag (1994) that he kindly made available to us.

At first sight, the choice seemed obvious. Pollard & Sag (1994) not only used feature structures, but also asserted that

> [Carpenter (1992)] in particular sets forth a logic very close to the one that we assume will underlie a fully formalized version of our theory.
>
> Pollard & Sag (1994), page 8

However, there was good reason to question this assertion. It was not that HPSG does not meet the feature introduction condition of Carpenter (1992), since King & Götz (1993) not only identified this shortcoming, but also provided a means to overcome it. It was rather that there is a fundamental and insurmountable incompatibility between the formalism that Pollard & Sag (1994) desired for HPSG and the formalism that Carpenter (1992) provided for it.

Pollard & Sag (1994) distinguished between concrete particulars called *tokens* and abstract universals called (confusingly) *types*,[1] and contended that

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

Pollard & Sag (1994) used feature structures to model linguistic types.

> In HPSG, the modelling domain... is a system of *sorted feature structures* that are intended to stand in a one-to-one relation with types of natural language expressions and their subparts. Pollard & Sag (1994), page 8

Linguistic types are not partial information, and Pollard & Sag (1994) realised that feature structures used to model one cannot be used to model the other. Indeed, in order to use feature structures as models of linguistic types, Pollard & Sag (1994) required that feature structures be both *totally well-typed* and *sort resolved*, and recognised that this requirement conflicts with the use of feature structures as models of partial information.

> A common source of confusion is that feature structures themselves can be used as descriptions of other feature structures. Since feature structures that are not totally well-typed and sort-resolved can be arranged into a partial ordering relation (called the *subsumption* relation), any feature structure can be thought of as partially describing any of the feature structures that it subsumes.... We choose to eliminate this possible source of confusion by using only totally well-typed, sort resolved feature structures as (total) models of linguistic entities and [attribute value matrix] diagrams (not feature structures) as descriptions.
>
> Pollard & Sag (1994), page 21

Thus, the Pollard & Sag (1994) feature structures do not play the role that feature structures play in KRL. Rather, Pollard & Sag (1994) envisaged a formal language in

[1] The 'types' of Pollard & Sag (1987) are the 'sorts' of Pollard & Sag (1994).

which formulae are interpreted as true or false of models of linguistic entities.

> The role of the linguistic theory is to give a precise specification of which feature structures are to be considered admissible; the types of linguistic entities that correspond to the admissible feature structures constitute the predictions of the theory. Pollard & Sag (1994), page 8

Thus, Pollard & Sag (1994) envisaged a HPSG formalism based on AVL, not KRL. Thus, despite the claim to the contrary, the Carpenter formalism is inherently unsuited to the HPSG of Pollard & Sag (1994).

Of course, the foregoing does not imply that the King formalism suits HPSG any better than the Carpenter formalism. For example, Pollard & Sag (1994) wanted to interpret each formula as true or false of a linguistic entity indirectly via a feature structure that models the entity, whereas King (1989) interpreted each formula as true or false of a linguistic entity directly. Also, Pollard & Sag (1994) explicitly assumed that each linguistic entity is an abstract type, whereas King (1989) implicitly assumed that each linguistic entity is a concrete token. However, in King (1994b) we reconstructed within the King formalism precisely the HPSG formalism that Pollard & Sag (1994) envisaged. We first said that two tokens are indiscernible iff there is no formula that is true of one token and false of the other. Clearly, indiscernibility is an equivalence relation. We then defined a type (in the sense of Pollard & Sag (1994)) to be an indiscernibility class of tokens, and defined what it is for a set of formulae to admit a type. Next, we constructed a mathematical entity called a morph, defined what it is for a set of formulae to admit a morph, and demonstrated that our types stand in a one-to-one correspondence with the morphs such that a set of formulae admits a type iff the set admits the corresponding morph. Finally, we proved that the morphs stand in a one-to-one correspondence with the Pollard & Sag (1994) sorted feature structures such that a set of formulae admits a morph iff the set admits the corresponding sorted feature structure. Thus, our types stand in a one-to-one correspondence with the Pollard & Sag (1994) types such that a set of formulae admits one of our types iff the set predicts the corresponding Pollard & Sag (1994) type. Pollard and King worked together to guarantee that the King (1994b) expansion of the King (1989) formalism was entirely in accord with the intuitions of Pollard with respect to a formalism for HPSG.

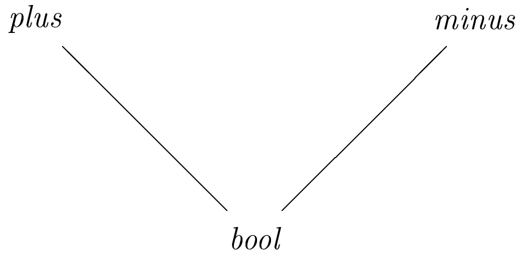
In King (1994b) we also showed how to adapt the Carpenter formalism to the HPSG of Pollard & Sag (1994). This adaptation reconfigures the Carpenter (1992) feature structures to function as models of linguistic tokens, and modifies the satisfaction relation accordingly. The adaptation yields a fragment of the King formalism, and essentially creates an AVL counterpart of the Carpenter formalism. The adaptation frees the Carpenter formalism from the expressive limitations of KRL concerning classical negation and implication, but also ignores the avowed primary concern of Carpenter (1992) with the representation and the processing of partial information. Consequently, in keeping with the principal concerns of King (1989) and Carpenter (1992), we adopted the King formalism as our project formalism.

3.2 A HPSG Processor

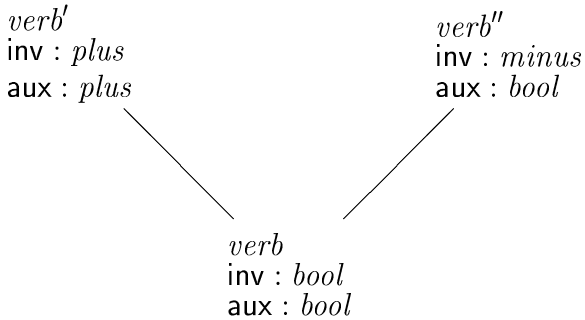
Our formalism provides a class of formal languages, where each formal language comprises a set of *descriptions* and a class of *interpretations*, and each interpretation comprises a set of *objects* and a *denotation function* that assigns each description the set of objects in the universe of which the description is true. A description is satisfiable iff the description is true of some object under some interpretation. We deemed a processor to be correct with respect to our formalism iff the processor provides an effective algorithm to decide if a description is satisfiable.

Our formalism's treatment of types sufficiently influenced the search for a basic HPSG processor to merit special mention here. Each HPSG type is a symbol that denotes a set of objects, and Pollard & Sag (1987) introduced a subsumption partial order over the types such that if a first type subsumes a second type then the denotation of the first contains the denotation of the second. We call a type that subsumes only itself a *species*. Each HPSG attribute is a symbol that denotes a partial function from objects to objects, and Pollard & Sag (1987) introduced an appropriateness relation over the attributes and the types such that if an attribute is appropriate for a type then the denotation of the attribute is defined on each member of the denotation of the type. In this way, Pollard & Sag (1987) explicitly assumed several conditions on the denotations of the types and the attributes. However, these conditions are relatively weak, and King (1989) argued that Pollard & Sag (1987) implicitly assumed two much stronger conditions on the denotations of the species, the types and the attributes (Pollard & Sag (1994) explicitly assumes both conditions): firstly, that the denotation of a type is partitioned by the denotations of the species subsumed by the type; secondly, that the denotation of an attribute is defined either on all or none of the members of the denotation of a species. We call these respectively the *partition condition* and the *all-or-nothing condition*. If both conditions hold then the entirety of the information held by the types with subsumption can be held by the species without subsumption. The project formalism explicitly assumes both conditions, and considers a type to be simply a disjunction of species. In fact, the partition and the all-or-nothing conditions are of interest beyond HPSG research. In Gerdemann & King (1993, 1994) we used them to encode certain of the FCRs (feature co-occurrence restrictions) of GPSG using types and subsumption, and showed that a processor that does not meet them both cannot correctly implement these FCRs.

We could find no processor that was correct with respect to our formalism, since we could find no processor that obeyed the partition and the all-or-nothing conditions. However, ALE was almost correct. A Carpenter (1992) typed feature structure is a finite and rooted Moore automaton whose input alphabet comprises the attributes and whose output alphabet comprises the types. Following Kasper (1987), one can easily take a finite set of typed feature structures to be a normal form representation of a positive description, where a description is positive iff the description contains no implication or negation. Every positive formula has such a representation. Carpenter (1992) defined the notion of a *totally well-typable* feature structure, and ALE has an effective algorithm to decide if a typed feature structure is totally well-typable. It is straightforward to prove that if a positive description is satisfiable then some member of its normal form representation is totally well-typable, thus ALE provides an effective and sound algorithm to decide if a positive description is satisfiable. Unfortunately, ALE does not provide a complete algorithm, since there are totally well-typable feature structures that represent unsatisfiable positive descriptions. An example best illustrates this. Under the partition condition, the type hierarchy



indicates that an object of type *bool* must be an object either of species *plus* or of species *minus*, and the type hierachy



indicates that an object of type *verb* must be an object either of species *verb′* or of species *verb″*. Thus, there can be no object of type *verb* with an inv attribute of type *plus* and an aux attribute of type *minus*. In effect, the above type hierachies encode an FCR that says if a verb is inverted then it is an auxiliary (a simplified version of FCR 1 of Gazdar et al. (1985)). The typed feature structure

$$\begin{bmatrix} \textit{verb} & \\ \mathsf{inv} & \textit{plus} \\ \mathsf{aux} & \textit{minus} \end{bmatrix}$$

represents a positive description that violates this FCR, and is thus unsatisfiable. However, the feature structure is totally well-typable.

ALE fails us only because it does not analyse types in terms of the species that they subsume. This prompted us to examine the use of typed feature structures as normal form representations of positive descriptions, in conjunction with the partition and the all-or-nothing conditions, to very fruitful effect. We call a well-typed (see

Carpenter (1992)) feature structure which outputs a species at each state a *resolved* feature structure, and say that a resolved feature structure is a *resolvant* of a typed feature structure iff both feature structures have the same underlying automaton (that is, they differ only in their outputs) and the typed feature structure subsumes the resolved feature structure. Viewing a typed feature structure as a normal form description, and interpreting it accordingly, we proved in King (1994a) that a typed feature structure is true of an object under an interpretation iff some resolvant of the typed feature structure is true of the object under the interpretation, and that each resolved feature structure is satisfiable. Thus, a typed feature structure is satisfiable iff it has a resolvant. We also gave an effective algorithm, that we call *resolution*, to enumerate the resolvants of a typed feature structure. Since a positive description is satisfiable iff some member of its normal form representation is satisfiable, resolution provides an effective, sound and complete algorithm to decide if a positive description is satisfiable. Unable to find a typed rule-based processor that was correct with respect to our full formalism, we decided to implement resolution in order to provide a basic processor that was at least correct with respect to the positive part of our formalism. We called this processor Troll.

In principle, an implementation could use the set of resolvants of a typed feature structure without additional type inferencing to maintain satisfiability. In practice, however, it is inefficient to work with large sets of resolved feature structures, and resolution itself is inefficient, since it considers species and can thus be shown to be NP-complete. Consequently, Troll uses three techniques – disjunction naming, unresolving and unfilling – that we collectively call *compacting*, in order to both minimise the space used to represent the resolvants of a feature structure, and avoid where possible the use of species. Compaction is additionally useful in a processor intended to implement HPSG grammars which very often generate feature structures of almost overwhelming size, since keeping feature structures as small as possible aids not only efficiency but also comprehension of the grammar by the user. We here outline the three components of compaction.

The resolvants of a typed feature structure have the same underlying automaton, and Troll inherited from Unicorn a relatively simple version of named disjunction that exploits this fact in order to more efficiently represent the set of resolvants of a typed feature structure as a single untyped feature structure plus a set of dependent state outputs. For example, under the type hierarchy given earlier, the feature structure

$$\begin{bmatrix} \textit{verb} & \\ \mathsf{inv} & \textit{bool} \\ \mathsf{aux} & \textit{bool} \end{bmatrix}$$

has the set of resolvants

$$\left\{ \begin{bmatrix} \textit{verb}' & \\ \mathsf{inv} & \textit{plus} \\ \mathsf{aux} & \textit{plus} \end{bmatrix}, \begin{bmatrix} \textit{verb}'' & \\ \mathsf{inv} & \textit{minus} \\ \mathsf{aux} & \textit{plus} \end{bmatrix}, \begin{bmatrix} \textit{verb}'' & \\ \mathsf{inv} & \textit{minus} \\ \mathsf{aux} & \textit{minus} \end{bmatrix} \right\},$$

which can be collapsed into the feature structure with named disjunction

$$\begin{bmatrix} \langle 1\ verb'\ verb''\ verb'' \rangle \\ \mathsf{inv}\ \ \langle 1\ plus\ minus\ minus \rangle \\ \mathsf{aux}\ \langle 1\ plus\ plus\ minus \rangle \end{bmatrix}.$$

In this simple example, using named disjunctions does not provide much benefit, since all the disjunctions interact. In general, however, using named disjunctions as in Maxwell & Kaplan (1989), Dörre & Eisele (1990) or Gerdemann (1991) can result in an exponential decrease in data-structure size (see Maxwell & Kaplan (1989)), so named disjunction is a quite significant compaction technique.

Requiring all states to output species results in needless complexity in cases where there are no co-occurrence restrictions between these species. Unresolving allows the reintroduction of nonspecies outputs into a feature structure in just such a case. For example, the previous feature structure can be unresolved to

$$\begin{bmatrix} \langle 1\ verb'\ verb'' \rangle \\ \mathsf{inv}\ \ \langle 1\ plus\ minus \rangle \\ \mathsf{aux}\ \langle 1\ plus\ bool \rangle \end{bmatrix}.$$

Though unresolving can be used, as in this example, within named disjunctions, the technique is generally most applicable to ordinary, noninteracting disjunctions. For example, an ordinary, noninteracting, disjunctive value of $\{plus, minus\}$ can always be reduced to the simple value *bool*.

In Götz (1993) we called *redundant* any transition in a typed feature structure to a nonre-entrant, terminal state which outputs a type inferable from the appropriateness specification, and we proved that removing a redundant transition from a typed feature structure viewed as a normal form description does not change the denotation of the feature structure. Unfilling is the process of removing redundant transitions from a typed feature structure. For example, the previous feature structure can be unfilled to

$$\{verb', verb''\},$$

which can then be further unresolved to

$$verb.$$

For maximum efficiency, Troll performs all resolution and as much compaction as possible off line. Thus, no further type resolution is needed during on-line unification. Though the system avoids the on-line type inferencing of Carpenter (1992, 1993), one could think of the on-line resolving of named disjunctions as the Troll analog of type inferencing. This analog of type inferencing is still quite efficient, since Troll in effect determines at compile time just those places where this inferencing must be performed during run time. Since a typical HPSG type hierarchy only needs such resolution of disjunctions to be performed at very few places, most instances of unifying two feature structures employ no resolution of named disjunctions at all. As a result, the parsing times of Troll are comparable to those of ALE, despite the complexity of named disjunctions.

Finally, we must emphasise that although Troll is a practical system for grammar development, it was designed much more as a vehicle for testing theoretical ideas about HPSG processing. Resolution, named disjunction, unresolving and unfilling are very general techniques that can be readily divorced from their actual implementation in Troll itself and incorporated into other HPSG processors.

3.3 Generation

We pursued two approaches to tactical generation. We optimised a typed extension of the Earley-style generator of Gerdemann (1991), and we adapted off-line compilation techniques to work with typed feature structure grammars. We took both approaches in parallel, with a view to their comparison and integration, and our current system in fact combines the two: off-line compilation is used to produce grammars for the Earley-style generator. The transformation, as described below, overcomes problems with empty or displaced heads.

By the end of the project, a different system will be finished, which uses off-line transformation for the purpose of bottom-up evaluation. The transformation for this purpose, so-called *Magic compilation* (see, among others, Beeri & Ramakrishnan (1991) and Ramakrishnan (1991)), is different because top-down guidance must be incorporated into the transformed grammar. The use of Magic compilation represents a significant advance over previous approaches to generation, since the problems of non-terminating recursion are handled in a uniform manner.

3.3.1 On-line Head-driven Generation

As Shieber (1988) noted, the main shortcoming of Earley generation is a lack of goal directedness that results in a proliferation of edges. Gerdemann (1991) overcame this shortcoming by modifying the restriction function to make top-down information available for the bottom-up completion step. Gerdemann's generator follows a head-driven strategy in order to avoid inefficient evaluation orders. More specifically, the head of the right-hand-side of each grammar rule is distinguished, and distinguished categories are scanned or predicted upon first. The resulting evaluation strategy is similar to that of the head-corner approach: prediction follows the main flow of semantic information until a lexical pivot is reached, and only then are the head-dependent subparts of the construction built up in a bottom-up fashion.

We further improved the Earley generator with a number of techniques, some of which were already mentioned in Gerdemann (1991), that reduce the number of edges created during generation. Three optimisations were especially helpful. The first supplies each edge in the chart with two indices, a *backward index* pointing to the state in the chart that the edge is predicted from, and a *forward index* pointing to the states that are predicted from the edge. By matching forward and backward indices, the states that must be combined for completion can be located faster. The second creates a table of the categories which have been used to make predictions from. As discussed in Gerdemann (1989, 1991), such a table can be used to avoid redundant predictions without a full and expensive subsumption test. The third indexes the lexical entries by their main semantic functor. This is the generation counterpart of phonological indexing in parsing. Such indexing is necessary to obtain constant-time lexical access. The work in this project has shown that these optimisations can lead to very significant gains in efficiency.

However, despite these heuristic improvements, the problem of goal-directedness still remained. As stated in our proposal, empty or displaced heads present the principle goal-directedness problem for any head-driven generation approach, where empty head refers not just to a construction in which the head has an empty phonology, but to any construction in which the head is partially unspecified. (Indeed, since phonology does not guide generation, the phonological realisation of the head of a construction plays no part in the generation of that construction. A surprising feature of the project grammar is that it contains examples of phonologically nonempty heads which, like classical examples of empty-head constructions, are nonetheless semantically and syntactically underspecified.) To better illustrate the problem that underspecified heads pose, consider the sentence

Hat Karl Marie geküßt?

The project grammar follows the argument composition analysis of verbal complexes presented in Hinrichs & Nakazawa (1989b): the subcategorisation list of the auxiliary verb is partially instantiated in the lexicon and only becomes fully instantiated upon its combination with its verbal complement, the main verb. The phrase structure rule that describes this construction is

$$\begin{bmatrix} \mathrm{s} \\ \textsf{comps}\ \langle\rangle \end{bmatrix} \longrightarrow \begin{bmatrix} \mathrm{V} & \\ \textsf{aux} & + \\ \textsf{comps} & \langle \boxed{\mathrm{x}} \mid \boxed{\mathrm{y}} \rangle \end{bmatrix}, \boxed{1}, \ldots, \boxed{\mathrm{n}}, \boxed{\mathrm{x}} \begin{bmatrix} \mathrm{V} \\ \textsf{comps}\ \boxed{\mathrm{y}} \langle \boxed{1}, \ \ldots\ , \boxed{\mathrm{n}} \rangle \end{bmatrix}.$$

Though a head-driven generator must generate first the head of the rule, nothing prescribes the order of generation of the complements of the head. If the generator generates second the main verb then the comps list of the main verb instantiates the comps list of the head, and generation becomes a deterministic procedure in which complements are generated in sequence. However, if the generator generates second some complement other than the main verb then the comps list of the head contains no restricting information to guide deterministic generation, and generation becomes a generate-and-test procedure in which complements are generated at random, only to be eliminated by further unifications. Clearly then, the order of evaluation of the complements in a rule can profoundly influence the efficiency of generation, and an efficient head-driven generator must order the evaluation of the complements in a rule accordingly.

Dynamic, on-line reordering can solve the complement ordering problem, but is rather unattractive: interpreting grammar rules at run time creates much overhead, and locally determining the optimal complement evaluation order is often impossible. Goal-freezing can also overcome the complement ordering problem, but is equally unappealing: goal-freezing is computationally expensive, it demands the procedural annotation of an otherwise declarative grammar specification, and it presupposes that a grammar writer possesses substantial computational processing expertise. We chose instead to tackle the complement ordering problem by using off-line compilation to automatically optimise the project grammar in such a way that the optimised grammar can be used for generation, without additional provision for dealing with the complement evaluation order, by our Earley generator.

3.3.2 Off-line Compilation Techniques

Our investigations of the compilation techniques described in section 2.3 led to the development of an off-line compilation approach for logic grammar optimisation. Part of this approach was then generalised to typed feature structure grammars and used, with our optimised Earley generator, for generation with the project grammar

The DIA (direct inversion approach) of Minnen & Gerdemann (1993) and Minnen et al. (1993) compiles a logic grammar into a new executable grammar in which some grammar rule nodes may have been reordered and some recursive procedures may have been reformulated. Dataflow analysis determines the relative efficiencies of certain grammar permutations, and the DIA determines the most efficient grammar permutation for a given input specification. The DIA uses a generalisation of the argument series of Strzalkowski (1991, 1993) to reformulate recursive procedures. Informally speaking, reformulation achieves a top-down encoding of the bottom-up evaluation of recursively defined heads. Unlike the EAA, the DIA reorders nodes without precalculating their MSEAs. This, and the use of memoisation, makes the DIA a direct, goal directed and, consequently, computationally feasible off-line compilation approach.

The DIA surpasses the top-down strategy of Dymetman & Isabelle (1988) by being fully automatic. The DIA also betters the EAA not only in being computationally tractable, but also with respect to the class of grammars it can invert. Firstly, due to its goal-directedness, the DIA can successfully invert certain recursive procedures that are deemed uninvertable by the EAA. Secondly, the reformulation of recursive procedures enables the DIA to deal successfully with the inversion of logic grammars containing deadlock-prone rules. This is an important result, since it was hitherto thought impossible to deal with deadlock-prone rules in such a way that top-down evaluation is guaranteed to terminate (see Shieber et al. (1990)). Finally, for a specific class of grammars, the new executable grammar produced by the DIA is more efficient.

However, the main advantage of the DIA for us (shared with the EAA) is that it optimises a logic grammar independently of the notion of a head. This renders the DIA an interesting approach to parsing and generation with logic grammars, since it circumvents the ordering problem.

Using DIA techniques to optimise typed feature structure grammars is complicated by the fact that the dataflow analysis upon which DIA node reordering is based can only deal with fully instantiated arguments, whereas feature structure grammar categories are usually not fully instantiated. This led us to generalise DIA dataflow analysis to a dataflow analysis for typed feature structure grammars. Our generalised dataflow analysis uses structure sharing in (or unification with) the defining rules of a category or definite goal, together with a specification of the fully instantiated features of the category/goal, in order to compute how the evaluation of the category/goal leads to the instantiation of features in other categories or goals. By comparing the defining rules as regards the information that is deemed to be available, the maximal degree of nondeterminacy that is introduced by the evaluation of a category is computed. This enables us to determine the relative efficiency of a particular evaluation order.

As an illustration, reconsider the argument raising example discussed earlier. We find the optimal evaluation order of the complements of the auxiliary verb through an investigation of the maximal degree of nondeterminacy introduced by the evaluation of the complements in various permutations. Dataflow analysis determines that the logical form of a sentence can only be used to restrict the evaluation of the nonverbal complements after the evaluation of the verbal complement. The relative efficiency of this evaluation could lead our dataflow analysis to choose

$$\begin{bmatrix} \text{s} \\ \mathsf{comps}\ \langle\rangle \end{bmatrix} \longrightarrow \begin{bmatrix} \text{v} \\ \mathsf{aux} \quad + \\ \mathsf{comps}\ \langle\, \boxed{x} \mid \boxed{y} \,\rangle \end{bmatrix}, \boxed{x} \begin{bmatrix} \text{v} \\ \mathsf{comps}\ \boxed{y} \langle\, \boxed{1}, \ldots, \boxed{n} \,\rangle \end{bmatrix}, \boxed{1}, \ldots, \boxed{n}$$

as the optimal evaluation order of our phrase structure rule for verbal complexes. Notice however, that this need not be the case. Our dataflow analysis treats heads and complements alike, and includes the head in the calculation of the optimal evaluation order of a rule. If the evaluation of the head of a rule introduces much nondeterminacy or provides insufficient restricting information for the evaluation of its complements, our dataflow analysis might not select the head as the first category to be evaluated, and choose instead

$$\begin{bmatrix} \text{s} \\ \mathsf{comps}\ \langle\rangle \end{bmatrix} \longrightarrow \boxed{x} \begin{bmatrix} \text{v} \\ \mathsf{comps}\ \boxed{y} \langle\, \boxed{1}, \ldots, \boxed{n} \,\rangle \end{bmatrix}, \begin{bmatrix} \text{v} \\ \mathsf{aux} \quad + \\ \mathsf{comps}\ \langle\, \boxed{x} \mid \boxed{y} \,\rangle \end{bmatrix}, \boxed{1}, \ldots, \boxed{n}$$

as the optimal evaluation order. This clearly demonstrates an extremely important consequence of using our dataflow analysis to compile a declarative grammar into a grammar optimised for generation. Empty or displaced heads pose us no problem, since the optimal evaluation order of the right-hand-side of a rule is determined regardless of the head. Our dataflow analysis ignores the grammatical head, but identifies instead the 'processing head', and (no less importantly) the 'first processing complement', the 'second processing complement', and so on. We currently use this off-line reordering approach to produce input for our Earley-style generator. Our results show that such transformations significantly improve the efficiency of our system.

It is important to note that the use of dataflow analysis for grammar optimisation depends crucially on the monostratal, uniform treatment of syntax, semantics and phonology in HPSG. In addition, the use of typed feature structures, as in HPSG implementations such as Troll, allows a more accurate analysis of the dataflow in a grammar, since it allows the determination of whether the value of a particular path is partially or fully instantiated. A value is said to be partially instantiated if it is more specific than the value required by the appropriateness specification. A value is fully instantiated if it is a species and the values for all of its features are also fully instantiated. Such a determination would, of course, be impossible in an untyped approach such as PATR-II. A dataflow analysis program designed for typed feature structures is currently being used as part of the transformation program providing input to our Earley-style generator.

The current system, then, uses off-line transformed grammars as input to the Earley generator. However, by the end of the project we will also finish development of a compiler to transform grammars for efficient bottom-up evaluation. This compiler extends the off-line compilation techniques for typed feature structure grammars of the previous paragraphs by combining it with a variant of Magic compilation, a technique developed within the deductive database community for goal-directed bottom-up evaluation.

Magic compilation is a general technique, based on dataflow analysis, to reformulate a logic program by adding new, so-called magic predicates, and changing the body of the clauses of the predicates in the program in such a way that the program can be evaluated bottom-up in a fashion that is both

goal-directed: the evaluation of literals in the program is identically restricted as in the case of top-down evaluation, and

safe: in case of nontermination, all possible solutions to a query are derived before the bottom-up evaluation gets into a recursive loop.

Henceforth, bottom-up evaluation of Magic-compiled programs is referred to as Magic evaluation.

D. S. Warren (1992) and Minnen (in preparation) observed a close correspondence between the Earley deduction of Pereira and D. H. D. Warren (1983) and Magic evaluation. Earley deduction is essentially a bottom-up approach, which nonetheless requires a significant amount of overhead for the top-down filtering produced by the prediction step. However, the top-down filtering in magic compiled programs is achieved by so-called *magic-predicates*. Thus, the complexity of the edges in the chart is reduced, because the top-down filtering does not result in rule instantiations (active edges) which have to be stored in the chart, but rather in derived magic facts (passive edges) representing argument bindings. A similar effect is accomplished in the bottom-up chart parser of ALE: by constructing the chart in reverse order, no active edges need to be stored. However, the ALE approach crucially depends on the absence of top-down prediction, which is clearly undesirable in case of generation, where a great deal of restricting information is made available in the initial start category. Magic compilation of logic grammars employs the same dataflow analysis as the one underlying the DIA. Moreover, the generalisation of this dataflow analysis that we developed for the optimisation of typed feature structure grammars through node-reordering is also used in Magic compilation of typed feature structure grammars.

3.4 Linguistic Analyses

3.4.1 Linguistic Analyses in HPSG

As a starting point for our linguistic research, we made a detailed investigation of the grammatical analyses in Pollard (1990) and Nerbonne (1991) of the verb-second and topicalisation constructions, and assessed the possibility of incorporating these analyses into our own previous work on the German verbal complex in Hinrichs & Nakazawa (1989a, 1989b, 1991). Pollard's visit to the SFS was extremely helpful in this regard. We adopted Pollard's head nonmovement analysis of V2 sentences and Nerbonne's lexical analysis of topicalisation as our central hypotheses. At the same time, however, the problem of spurious ambiguity in Pollard's account and serious formal flaws in the statement of Nerbonne's lexical rule for topicalisation were recognised as crucial shortcomings to be overcome by our subsequent work.

We also identified split-NP topicalisation as another phenomenon of German syntax which is challenging for head-driven approaches to generation. As illustrated in (1), this construction arguably involves a remnant noun phrase in the Mittelfeld that contains an empty nominal head that is linked semantically to a topicalised NP.

1 [Frische Tomaten]$_1$ haben wir keine e_1.
fresh tomatoes have we none
'We don't have any fresh tomatoes.'

Although split-np topicalisation has been extensively discussed in the GB literature (for example, see van Riemsdijk (1987) and Fanselow (1988)), no HPSG analysis of this construction was available at the beginning of our project.

3.4.2 Implementation of the Grammar Fragment

In order to empirically test our linguistic analyses and provide sample grammars for the development of Troll, we implemented our grammar fragment in parallel with our linguistic research. During Troll's infancy, we implemented the grammar fragment of Pollard (1990) using CUF and TFS (Meurers and Götz (1993) reported our experiences using these systems), but later ported this fragment to ALE and

Troll. We then implemented the HPSG grammar described in Hinrichs & Nakazawa (1994) using ALE and Troll. The resulting fragment covers the phenomena of auxiliary flip, partial VP topicalisation, and split-NP topicalisation in German, as well as the positioning of the finite verb in all relevant clause types of German.

Converting a linguistic analysis formulated in HPSG into a specification that a system like ALE or Troll can recognise is by no means trivial, and Meurers (1994) described in detail some of the intricacies involved. The main issues that needed to be resolved concerned the encoding of dominance and precedence relations, the expressiveness of the constraint language, and the encoding of the lexicon and the lexical rules.

HPSG grammars are expressed in terms of a type hierarchy and an accompanying appropriateness specification which introduces the set of appropriate features for each type. This specification serves to classify linguistic objects described by the linguistic theory. The set of well-formed linguistic objects are further constrained by a set of universal or language-specific principles, which are usually expressed as implications. As mentioned in section 2.2, computational systems that only employ constraint resolution can in principle compute whether a given description is consistent with the constraints in a HPSG grammar. However, such systems have serious problems with the control structure, since they supply no ordering of constraints with respect to their processing importance. To avoid this problem, ALE and Troll demand that the user specifies the dominance and precedence relations separately. The system can then use the resulting phrase structure rules as a backbone for constraint resolution. Thus, in order to implement the Hinrichs & Nakazawa (1994) grammar fragment, we encoded dominance schemata and precedence rules in phrase structure rules, and attached the HPSG universal principles to each phrase structure rule via a system of macros and definite clause attachments.

While the principles of HPSG can take the form of arbitrarily complex implications, current implementational systems for HPSG cannot accomodate such a rich constraint language. In our current implementation of the grammar fragment, two methods are employed to overcome this problem. Firstly, if the antecedent of an implicational constraint contains no path equations then it is possible to introduce a new type for the antecedent. However, it should be noted that this significantly enlarges the type hierarchy, since the descriptions making up the constraint have to be mediated via the appropriateness conditions. Secondly, a macro can be used to specify a class of lexical entries. The definition of a lexical entry is thus reduced to a call to the macro of its lexical class and the specification of the entry specific information, resulting in a modular and maintainable lexicon.

To capture further generalisations in the lexicon, lexical rules as relations between classes of lexical entries are an integral part of the HPSG architecture, and the Hinrichs & Nakazawa (1994) grammar fragment makes extensive use of this mechanism in its treatment of topicalisation. However, at present there is no consensus regarding the exact properties of the lexical rules mechanism, nor how to make algorithmic use of the linguistic generalisations expressed in lexical rules. The current implementation of our HPSG grammar fragment encodes lexical rules as modification automata which are attached to lexical entries. The construction of such automata could be automated in a later step.

4 Ergebnisse und ihre Bedeutung

4.1 A HPSG Formalism

Pollard and Sag (1987) provided HPSG with a formalism built upon a Heyting algebra of feature structures seen as models of partial information about entities, and assumed that the meet of two feature structures is partial information about an entity iff the first and the second feature structure are partial information about the entity, the join of two feature structures is partial information about an entity iff the first or the second feature structure are partial information about the entity, and the pseudocomplement of a feature structure is partial information about an entity iff the feature structure is not partial information about the entity. Unfortunately, King (1989) showed that while the first assumption is true under certain circumstances, and the second assumption is true generally, the third assumption is false. More positively, King (1989) argued that HPSG uses feature structures to *describe* entities, replaced the troublesome algebra of feature structures with a well behaved syntax of descriptions, and thus created an AVL formalism for HPSG.

Though Pollard and Sag (1994) abandoned the HPSG formalism of Pollard and Sag (1987), and envisaged instead a formalism akin to that provided by King (1989), the envisaged and the provided formalisms differ in three respects. Firstly, Pollard and Sag (1994) introduced a type/token distinction, and contended that types are the true subject of linguistic enquiry, whereas King (1989) had no notion of a type in this sense. Secondly, Pollard and Sag (1994) envisaged the feature structures playing a mediating role in the interpretation of a theory, whereas King (1989) interpreted a theory directly, without mediation. Thirdly, and most seriously, Pollard and Sag (1994) envisaged a KRL-based HPSG formalism, whereas King (1989) provided an AVL-based HPSG formalism. In King (1994b) we reconciled the first two differences by defining the tokens and the types within the King (1989) formalism, and constructing a class of abstract graph theoretic entities called morphs that stand in a one-to-one correspondence with the types. We then argued that Pollard and Sag (1994) was in error to envisage a KRL-based HPSG formalism, since KRL cannot provide the envisaged formalism without rejecting the KRL tenet that feature structures model partial information, and thus rendering irrelevant such KRL concerns as subsumption, unification and subsumption monotonicity. Rather, the HPSG formalism that Pollard and Sag (1994) envisaged not only *can* but also *must* be AVL-based. In effect, Pollard and Sag (1994) converted HPSG from a unification grammar to a constraint grammar, and King (1994b) expanded King (1989) to provide the appropriate formalism for this grammar.

4.2 A HPSG Processor

The partition and the all-or-nothing conditions profoundly influenced our search for a HPSG processor. Pollard and Sag (1987, 1994) assumed (the former implicitly, the latter explicitly) that HPSG meets both conditions, and Gerdemann and King (1993, 1994) showed that a typed feature structure processor can encode certain simple GPSG FCRs iff the processor exploits both conditions. Thus, any HPSG (or GPSG) processor must meet both conditions. The fact that we could find no such processor lead us to develop our own HPSG processor. The additional fact that ALE is almost correct with respect to the positive part of our formalism lead us to build our processor upon the notion of a typed feature structure as a description. King (1994a) showed that resolution is an effective, sound and complete algorithm to decide the satisfiability of a positive description. Our Troll processor is an efficient implementation of this algorithm.

Troll is a fairly conventional unification-based processor. For example, the Troll parser is a standard chart parser, coupled with a definite-clause interpreter which includes cuts and a freeze mechanism along the lines of the delay statements of CUF. Troll nonetheless incorporates several novel features which could be readily transferred to other processors. One such feature is type inferencing using named disjunctions. Troll demonstrates that this is not only a theoretic possibility, but also a practical and efficient form of type inferencing. Another such feature is the unfilling detailed in Götz (1993). Unfilling not only minimises the size of the untyped feature structure underlying a disjunctive feature structure, but also removes from a disjunctive feature structure each named disjunction that merely records the set of possibilities allowed by the appropriateness specification, thus reducing the number of disjunctions. Both size reductions significantly contribute to the efficiency of Troll. Indeed, despite the disjunction inherent to resolution, Troll is a surprisingly efficient parser. Parsing times are generally comparable to those of ALE, even though Troll does the extra work of verifying satisfiability during unification. This efficiency can be attributed to both the typing that permits the unfilling of feature structures, and the off-line predetermination of the on-line performance of type inferencing.

Troll has proved to be a useful processor which has applications quite independent of the generation component required for our project. For one example, the fact that Troll can encode FCRs in the type hierarchy is certainly a point of great interest to many linguists (for example, see Copestake et al. (1993)). For another, the correctness of Troll with respect to a formalism that bears striking similarities to members of the KL-ONE (knowledge language-one) family of knowledge representation languages (see Brachman and Schmolze (1985) for an overview of the seminal KL-ONE language) has aroused the interest of knowledge representation researchers (see section 5).

4.3 Generation

We investigated techniques for both on-line generation and off-line compilation for generation. Our on-line generator is an Earley-style generator which includes, for efficiency, indexing both on edges in the chart and on lexical entries. We found that these indexing techniques make the generator an order of magnitude faster. The generator also uses a modified version of restriction, which allows all semantic information to be passed downward through the tree and thus allows this information to become available for the top-down filtering needed for goal directedness.

Modified restriction alone, however, is not sufficient for goal-directed generation. One minimally needs a transformation of the grammar to achieve a head-driven strategy. For our project grammar, however, such a minimal transformation was clearly not enough since in cases of, say, verb raising, the head is significantly underspecified. In fact, the head auxiliary verb only obtains an instantiated comps list when it combines with the main verb. To handle such grammars, we required a grammar transformation program that would go significantly beyond the simple head-first strategy.

We developed the DIA, a new compilation technique for logic grammars. The DIA produces a new executable grammar that is suited to efficient top-down evaluation. The DIA improves upon both the approach described in Dymetman & Isabelle (1988) and Strzalkowski's EAA. The intraclausal reordering technique developed within the context of the DIA was then generalised so that it could be employed to optimise the project grammar for use with the Earley generator. The main component of this generalisation was a dataflow-analysis program for typed feature structures. The reordering technique solves the ordering problem by producing a new grammar in which the right-hand-sides of the grammar rules are optimally reordered for efficient generation.

The off-line compilation techniques we developed make crucial use of the fundamental properties of the HPSG formalism. The monostratal, uniform treatment of syntax, semantics and phonology supports dataflow analysis, which is used extensively to provide the information upon which off-line compilation is based. In addition, the notion of typing, as in HPSG implementations like Troll, proved to be invaluable, since it enabled a more accurate analysis of the dataflow in a grammar. In particular, an appropriateness specification could be used to determine when the value of a path was to any extent more specified than its default value.

Our current system inputs off-line compiled grammars to our Earley generator. By the end of the project, however, we will finish developing a new compiler for typed feature structures that produces executable parsing and generation grammars that support strict bottom-up evaluation. Safe, goal-directed bottom-up evaluation is achieved by means of the application of a variant of Magic compilation. Integrating this with a complete reordering of the nodes in the rules of a grammar (similar to the DIA node reordering) ensures that the evaluation of categories proceeds in the most efficient order.

On-line head-driven generation supports reversibility, an attractive property for a natural language processing system, but our experience with the project grammar showed that it is necessary to employ some form of freeze mechanism to achieve efficient on-line processing without specialised grammars for parsing and generation. This is unattractive, since such mechanisms require considerable processing overhead and the addition of processing directives by the grammar writer/implementer. To achieve efficiency, we therefore combined on-line generation with off-line compilation techniques for typed feature structure grammars. This combination not only elegantly solves the problems of on-line head-driven generation, but also has the attraction of using an independent grammar specification. It simultaneously provides efficiency and true bidirectionality, in that a grammar can be optimised for any given processing direction.

4.4 Linguistic Analyses

Hinrichs & Nakazawa (1994) presented our HPSG analyses of the constructions that were most relevant to our project. These analyses cover several German constructions which are of theoretical interest for automatic natural language generation. Major portions of these analyses were developed during the visit of Tsuneko Nakazawa to the SFS, and the visit of Hinrichs to the NTT Message Processing Laboratory in Yokosuka, Japan during his sabbatical in 1994.

4.4.1 Topicalisation

Our HPSG analysis covers a wide range of topicalised constituents, including the topicalisation of NPs as in (2a), of saturated VPs as in (2b), and of partial VPs as in (2c–f).

2 a **Ein Märchen** wird er seiner Tochter erzählen.
a fairy tale will he his daughter tell
'He will tell his daughter a fairy tale.'

b **Seiner Tochter ein Märchen erzählen** wird er.
his daughter a fairy tale tell will he
'He will tell his daughter a fairy tale.'

c **Ein Märchen erzählen** wird er seiner Tochter.
a fairy tale tell will he his daughter
'He will tell his daughter a fairy tale.'

d **Ein Märchen erzählen** wird er seiner Tochter müssen.
a fairy tale tell will he his daughter must
'He will have to tell his daughter a fairy tale.'

e **Erzählen** wird er seiner Tochter ein Märchen.
tell will he his daughter a fairy tale
'He will tell his daughter a fairy tale.'

f **Erzählen müssen** wird er seiner Tochter ein Märchen.
tell must will he his daughter a fairy tale
'He will have to tell his daughter a fairy tale.'

In addition, our analysis covers so-called NP split as in (3a), and the interaction of NP split and partial VP topicalisation as in (3b).

3 a Giftige Schlangen gibt es hier keine.
poisonous snakes gives it here no
'There are no poisonous snakes here.'

b Briefe geschrieben hat er keine.
letters written has he no
'He has written no letters.'

Hinrichs & Nakazawa (1994) showed in detail that the material of a split NP which is located in the Vorfeld and the Mittelfeld (*giftige Schlangen* and *keine* in (2a) for example) consists syntactically and morphosyntactically of two independent NPs which are combined semantically to form a single nominal argument. The material in the Mittelfeld is analysed as an NP with an empty nominal head, as Nerbonne et al. (1989) have suggested for the treatment of elliptical NPs in English.

We formally treat topicalisation by means of lexical rules which change the valency of verbs in such a way that certain NP complements are realised not in the Mittelfeld, but in the Vorfeld. This means that the complements appear not on the comps list of the subcategorising verb, but on its slash list. Nerbonne (1991) already stated the basic idea of the suggested lexical analysis, but his suggested lexical rule proved to be unsound and had to be fundamentally revised.

The results of our research are important in several respects. Firstly, Hinrichs & Nakazawa (1994) encompassed a broader empirical domain than did earlier analyses of German topicalisation in HPSG and other monostratal frameworks, because it presented the first analysis of NP split in such a framework. Secondly, our analysis covers a broader analytical domain than do studies of NP split in the GB tradition (see van Riemsdijk (1987) and Fanselow (1988) for example), since, unlike the GB analyses, it can treat the interaction of VP topicalisation and NP split adequately. Finally, the analysis of German topicalisation suggested by Pollard (1990) suffers from the drawback that it arrives at the whole range of topicalisable constituents only by introducing massive structural ambiguity in the Mittelfeld for which there is no independent empirical motivation. The Hinrichs & Nakazawa (1994) analysis successfully eliminated these unmotivated ambiguities, yet treated all the data which Pollard covered. The elimination of the unwelcome ambiguities is important from a computational-linguistic perspective, since it significantly reduces the number of derivation steps which are necessary for parsing or generating such a construction.

4.4.2 Word Order

The HPSG fragment provides LP (linear precedence) rules for the placement of verbal constituents and NP complements for all major clause types of German. As in Pollard (1990), the placement of the finite verb is accomplished via LP rules alone, without recourse to head movement. Thus, for V2 sentences, the finite verb appears as the head daughter of the saturated V projection, which is treated as an inverted clause in the same way as the HPSG analysis handles AUX inversion in English. The sentence-initial topicalised constituent is treated as the filler constituent in a head-filler structure and is introduced via lexical rules, as described above.

5 Vergleiche mit anderen Arbeiten

5.1 A HPSG Formalism

The most obvious comparison for us was between the Carpenter formalism and our formalism, and this comparison gained importance with the creation of Troll. We have already outlined the differences between these formalisms, and given our reasons for choosing the King formalism as our project formalism. More positively, however, the project had developed two different AVL counterparts of the Carpenter formalism. The first (see King (1994b)) reconfigured the typed feature structures to function as models of linguistic tokens. The second (see King (1994a)) viewed the typed feature structures as normal form positive descriptions. Both counterparts were discussed at length during a visit to the SFS by Bob Carpenter, who found the second particularly appealing, and cited the numerous 'most general satisfier' theorems of Carpenter (1992) in support of it. Carpenter and King have begun to develop this counterpart, and hope to extend resolution to provide both an effective algorithm to decide the satisfiability of an inequated feature structure (see Carpenter (1992)), and an effective algorithm to decide if the denotation of a first inequated feature structure always contains the denotation of a second.

A more surprising comparison for us concerned knowledge representation languages. Nebel and Smolka (1988) noted strong similarities between AVL and the KL-ONE family of knowledge representation languages. Nevelin Boynov and Kiril Simov of the Bulgarian Academy of Sciences hoped to combine aspects of KL-ONE languages with aspects of our formalism to yield a knowledge representation language suitable for systems able to support the incremental acquisition of knowledge. Boynov and Simov visited the SFS under the Tempus program in order to meet the project members, and the project extended their visit in order to begin collaborative research with King. Though not directly connected to our project, this collaboration reveals potential uses for our work on formalism beyond natural language processing.

5.2 A HPSG Processor

Troll is most directly comparable with ALE. Though we prefer the semantics of Troll, since the semantics of ALE simply does not suit our formalism, other aspects of ALE interest us greatly. Troll performs unfilling to reduce data-structure size, whereas ALE performs total filling for the seemingly contradictory purpose of also reducing data-structure size. The intuition behind ALE is that total filling can be used to allow positional encoding of features in a term representation as opposed to the less efficient list of feature-value pair representation used in Troll. However, as seen earlier, total filling of uninformative features in Troll can lead to the introduction of very many disjunctions, and can also lead to infinite feature structures in the worst case. Nevertheless, we have experimented with positional encoding of unfilled feature structures by introducing dummy slots for unfilled features. The idea needs more development and testing, but appears promising. ALE also incorporates many ideas for Prolog compilation from the WAM (Warren abstract machine) of D. H. D. Warren (1983). A further line of development for Troll will be to see to what extent these compilation techniques are applicable to resolved feature structures.

Adapting ideas from CUF may also yield additional efficiency for Troll. Whereas Troll uses named disjunctions to maintain the satisfiability of feature structures during unification, CUF uses constraints on feature structures (as described in Dorna (1992)) to obtain the same effect. Comparing these two approaches in future work would certainly be worthwhile.

Another worthwhile comparison is between Troll and the LKB (lexical knowledge base) processor which has been designed as part of the ACQUILEX project (see Briscoe (1991)) in order to allow the representation of syntactic and semantic information semi-automatically extracted from machine readable dictionaries on a large scale. LKB uses the typed feature structures of Carpenter (1992), but extends the notion of an appropriateness specification to allow path equations, and calls a typed feature structure that is well-typed in the sense of Carpenter (1992) but with respect to this extended appropriateness specification a *well-formed* feature structure. LKB is an implementation of the algebra of well-formed feature structures presented by de Paiva (1993).

However, the creators of LKB found shortcomings in well-formedness similar to those that we found in total well-typedness.

> One disadvantage of the system as described is that it is not possible in general to enforce co-occurrence restrictions, even of a quite limited sort.
>
> Copestake et al. (1993), page 153

Independently of us, Copestake et al. (1993) ascribed this shortcoming to the fact that well-formedness does not necessarily consider species, and defined the notion of an *ultimately well-formed* feature structure, where an ultimately well-formed feature structure is a feature structure that has at least one well-formed resolvant. Copestake et al. (1993) conjectured that ultimate well-formedness is a more correct notion than well-formedness, and claimed that though checking for ultimate well-formedness is computationally expensive, this expense is insignificant since the check need only be performed when automatically acquiring a lexical entry from a machine readable dictionary. However, any unification based processor that subsequently uses an LKB acquired lexicon must check for ultimate well-formedness after each unification, since the unification of two ultimately well-formed feature structures need not be ultimately well-formed.

We feel that the ACQUILEX project and our project have much to offer each other. The LKB appropriateness specification interests us greatly, and we could offer in return both a rigorous semantic justification for the intuitions of Copestake et al. (1993) regarding ultimate well-formedness, and the various compaction techniques of Troll for making any unification based processor that used an LKB acquired lexicon as efficient as possible. Unfortunately, we learned of ACQUILEX very recently, and have only just begun communicating with them. Nonetheless, both projects earnestly hope for mutually beneficial collaboration in the future.

TFS allows even more complex appropriateness specifications than LKB. Nonetheless, resolution may yet be applicable to TFS. Although TFS has extremely general type definitions, Martin Emele (personal communication) has informed us that it is still possible to extract from these definitions an appropriateness specification for the purpose of performing some off-line type inferencing. Thus, Troll could benefit TFS. Conversely, TFS could benefit Troll. Specifically, a facility to specify complex type definitions as in TFS and compile these definitions into a simple appropriateness specification that a rule based system such as Troll could use is extremely attractive. This compromise between the flexibility of general type definitions and the efficiency of rule-based processing will, in fact, be a major focus of the next phase of the B4 project.

Troll was an aspect of our project that especially attracted Boynov and Simov to our formalism (see subsection 5.1). As well as formulating a knowledge representation language suitable for systems able to support incremental knowledge acquisition, Boynov and Simov hoped to construct such a system using some of the techniques employed by Troll. During their visit to the SFS, Boynov and Simov helped debug, rationalise and comment Troll, and ported Troll to run on LPA Mac Prolog, ready for use at their home institution. We believe that this amply demonstrates both the utility and the portability of the techniques that we have implemented in Troll.

5.3 Generation

Our approach to HPSG generation is highly innovative in that we use an off-line compiler to automatically prime a declarative grammar for generation (or any other given processing direction), and input the primed grammar to an on-line generator. Unlike van Noord (1993) for example, we do not use on-line generation exclusively. Our off-line compiler extends techniques proposed by Dymetman and Strzalkowski.

However, unlike Dymetman, our compiler is fully automatic; unlike Dymetman and Strzalkowski, our compiler compiles typed feature structure grammars, rather than simple logic grammars. Combining on-line and off-line techniques provides both an elegant solution to the problems that empty or displaced heads pose conventional head-driven generation, and efficient bidirectional processing.

Several of the problems we encountered during our research were not specific to our project. This led SFB 340 projects B4 and B5, and the Verbmobil 'Syntax/Lexicon' group to form a working group to investigate the control problem for (typed) feature structure grammars. The working group meets regularly to exchange experiences and study relevant current literature.

5.4 Linguistic Analyses

The computational linguistic research projects within and without Germany that use HPSG are so numerous that it is impossible to compare the grammatical analyses of all these projects in depth, all the more since most projects in Europe and North America focus on English grammar. As regards research in the Federal Republic of Germany, the BMfT supported Verbundprojekt Verbmobil, and the BiLD and DISCO projects headed by Hans Uszkoreit at the University of the Saarland, Saarbrücken relate most closely to the research reported here. However, to our knowledge, none of these projects have hitherto analysed the interplay between topicalisation, the V2 position and the verbal complex in German to the extent that it became necessary in the current phase of the B4 project for the evaluation of algorithms for natural language generation.

6 Offene Fragen

A number of open questions arise from our work on a HPSG formalism, but two are especially relevant to our project: the mathematical formalisation of both LP rules and lexical rules. Each presents a serious problem for a HPSG formalism, not least because the intuitions they are intended to capture are far from clear. Our primary concern is to avoid ad hoc remedies, and develop instead rational accounts of both phenomena. Though some instances of LP and lexical rules have been expressed using our formalism, these expressions lack elegance and perspicuity, and they require far too extensive logical preparation to be considered convenient for use by linguists. Elegant and comfortable tools with which a linguist can easily and accurately express LP and lexical rules are a pressing need. We believe that our formalism provides an ideal foundation upon which to build such tools, and we hope to undertake their construction during the next phase of the B4 project.

In addition to formalism specific problems, LP and lexical rules present problems for processing. For example, Seiffert (1987) noted that the nonlocal flow of information together with the use of a restriction function can lead to undetected LP violations in feature structure grammars which factor LP information. Certainly, solutions to such problems have been suggested. For example, Seiffert (1987) gave a generate-and-test solution to the above mentioned problem that double-checks the satisfaction of LP statements, Morawietz (in preparation) presented a better solution based upon the direct parsing of context-free ID/LP (immediate dominance/linear precedence) grammars of Shieber (1984), and Engelkamp et al. (1992) used off-line compilation to incorporate LP rules into the lexical entries. Nonetheless, choosing between various solutions and relating them to our compilation techniques are open questions that we intend to address in the next phase of the B4 project.

Though Troll is correct with respect to the positive part of our formalism, the problem of choosing a processor that is correct with respect to the entirety of our formalism remains. Such a processor exists in principle, since Wedekind (1991) gave an effective satisfiability algorithm for the untyped AVL of Johnson (1988), and in Kepser (in preparation) we made a far from trivial extension of this algorithm in order to yield an effective satisfiability algorithm for our full formalism. We believe that the various techniques that we have developed in building Troll can be used to implement this new, more general algorithm efficiently. Needless to say, such an implementation would be a priority during the next phase of the B4 project.

The generation work of this phase of the B4 project has been innovative in at least two respects. First, we have extended generation techniques to cover problems of empty or underspecified heads. Such extensions involve off-line transformations which are guided by dataflow analysis of the grammar. We have applied these techniques to actual examples of underspecified-head constructions from a linguistically motivated grammar to produce transformed grammars that are then used by our Earley-style generator. An open question, to be investigated in the next phase of project B4, is how far these transformation techniques can be used to transform completely declarative grammars that are specified without phrase-structure rules. This further work will allow pure constraint-based grammars to be used in both parsing and generation.

As a second innovation in this phase of B4, we have extended generation techniques to handle HPSG grammars with typed feature structures. The generalisation to typed feature structures has shown how typing information may be used to develop dataflow analyses similar to the use of dataflow analysis for Prolog programs or grammars. A Prolog term has arguments which are either instantiated or uninstantiated, whereas a feature structure has paths which are in a sense always instantiated. We have followed the strategy of considering a path to be instantiated when its value is more specific than the value required by the appropriateness specification. Instantiation of a path, however, is not a yes-or-no question. In some cases the value of a path may be fully instantiated, but in general there will be degrees of instantiation. Here, it is clear that dataflow analysis must be combined with heuristics that determine which instantiations are more important than others. Needless to say, a considerable amount of work is required in this area.

The linguistic analyses of the syntax of German that we have completed during the present phase of the B4 project have lead to a HPSG fragment that provides a detailed analysis of topicalisation, the syntax of coherent constructions (in the sense of Bech (1955/57)), and the placement of verbal constituents in the major clause types of German. We believe that this work provides a good starting point for the development of an even more substantial HPSG fragment of the syntax and semantics of German, of the kind proposed for the next phase of SFB 340 by projects B8 and B9. The following topics would be particularly interesting in this regard:

The current HPSG fragment developed by the B4 project concentrates on matters of syntax. Questions of semantic representation are considered only to the extent that they are relevant to particular aspects of the syntactic analysis. Future work on a HPSG fragment would benefit significantly from a close interaction with the on-going research in discourse representation structures.

The current treatment of the Vorfeld is restricted to topicalisation phenomena. A natural next step would be a more general account of various kinds of movement constructions, particularly wh-movement and scrambling.

The current treatment of complex verbal constituents is restricted to an analysis of coherent infinitival constructions. A natural and much needed extension would involve a treatment of incoherent infinitival constructions first described by Bech (1955/57) and most recently studied in the HPSG framework by Kiss (1993).

In all of the three areas mentioned for future research, the Sonderforschungsbereich would provide a stimulating environment in which to explore in much greater detail than was possible in the current phase of the B4 project how current work in the GB framework could be incorporated into a HPSG fragment for German.

7 Gastwissenschaftler, Vorträge und Veröffentlichungen

7.1 Gastwissenschaftler

Carl Pollard (pollard@ling.ohio-state.edu). Linguistics Department, 222 Oxley Hall, 1712 Neil Avenue, Ohio State University, Columbus, Ohio 43210, USA. From 15th June to 15th July, 1992.

Tsuneko Nakazawa (tsuneko@nttnly.ntt.jp). NTT Message Processing Systems Laboratory, Yokosuka Research and Development Center, Yokosuka, Japan. From 1st February to 31st October, 1993.

Bob Carpenter (carp@lcl.cmu.edu). Computational Linguistics Program, Philosophy Department, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, USA. From 24th May to 3rd June, 1993.

Nevelin Boynov (lml@bgcict.bitnet) and Kiril Simov (kivs@bgearn.bitnet). Linguistics Modelling Laboratory, Bulgarian Academy of Sciences, 25A Acad. G. Bonchev Str., 1113 Sofia, Bulgaria. From 3rd March to 15th May, 1994.

Raghu Ramakrishnan (raghu@cs.wisc.edu). Department of Computer Science, University of Wisconsin, Madison, Wisconsin 53711, Canada. Invited for summer/autumn, 1994.

7.2 Vorträge und sonstige Aktivitäten

26th June, 1992. Guido Minnen. Paper "Earley-type Parsing of TAG and Factoring Linear Precedence". To the *2nd TAG+ Workshop*. Philadelphia, Pennsylvania, USA.

17th–28th August, 1992. Dale Gerdemann & Erhard W. Hinrichs. Advanced lecture course 'Natural Language Generation'. At the *4th European Summer School in Logic, Language and Information*. Colchester, England.

19th March 1993. Dale Gerdemann. Invited paper "A Minimalist Approach to Feature Structures" (coauthored by Thilo Götz). To the *Groningen Workshop on HPSG and Categorial Grammar*. Groningen, The Netherlands.

25th July, 1993. Erhard W. Hinrichs & Tsuneko Nakazawa. Invited paper "Topicalization in German: An HPSG Analysis". To the *1st International Workshop on HPSG*. Columbus, Ohio, USA.

4th August, 1993. Detmar Meurers. Paper "An Alternative Approach to Linearization: SUP Grammars" (coauthored by Frank Morawietz). To the *1st International Workshop on HPSG*. Columbus, Ohio, USA.

23rd–27th August, 1993. Bob Carpenter & Erhard W. Hinrichs. Workshop 'Implementations of Attribute-Value Logics for Grammar Formalisms'. At the *5th European Summer School in Logic, Language and Information*. Lisbon, Portugal.

23rd August, 1993. Dale Gerdemann. Paper "Type Resolution in the Troll System: The Implementation". To the 'Implementations of Attribute-Value Logics for Grammar Formalisms' worskshop at the *5th European Summer School in Logic, Language and Information*. Lisbon, Portugal.

23rd August, 1993. Paul J. King. Paper "Type Resolution in the Troll System: The Mathematics". To the 'Implementations of Attribute-Value Logics for Grammar Formalisms' worskshop at the *5th European Summer School in Logic, Language and Information*. Lisbon, Portugal.

25th August, 1993. Erhard W. Hinrichs & Tsuneko Nakazawa. Invited paper "Split NP Topicalization: A Challenge for Categorial Grammar". To the 'Categorial Grammar' workshop at the *5th European Summer School in Logic, Language and Information*. Lisbon, Portugal.

26th August, 1993. John Griffith. Paper "DUSt: A Disjunctive Unification System". To the 'Implementations of Attribute-Value Logics for Grammar Formalisms' worskshop at the *5th European Summer School in Logic, Language and Information*. Lisbon, Portugal.

26th August, 1993. Detmar Meurers. Paper " Linearization with LP statements and the SUP Grammar Alternative" (coauthored by Frank Morawietz). To the 'Implementations of Attribute-Value Logics for Grammar Formalisms' workshop at the *5th European Summer School in Logic, Language and Information*. Lisbon, Portugal.

29th August, 1993. John Griffith. Invited paper "The Use of Disjunction". To the *Expert Advisory Group on Linguistic Engineering Standards Working Group on Linguistic Adequacy of Grammar Formalisms*. Lisbon, Portugal.

28th August 1993. Detmar Meurers. Invited paper "Some Choicepoints for Linguistic Formalisms". To the *Expert Advisory Group on Linguistic Engineering Standards Working Group on Linguistic Adequacy of Grammar Formalisms*. Lisbon, Portugal.

29th September 1993. Guido Minnen. Paper "Direct Automated Inversion of Logic Grammars". To the *4th International Workshop on Natural Language Understanding and Logic Programming*. Nara, Japan.

5th October, 1994. Tsuneko Nakazawa & Erhard W. Hinrichs. "Topicalization in German: An HPSG Analysis". At *Between Syntax, Semantics and Logic*. Stuttgart, Germany.

7th October, 1993. Erhard W. Hinrichs & Paul J. King. Workshop 'What Feature Structures Do'. At *Between Syntax, Semantics and Logic*. Stuttgart, Germany.

7th October, 1993. Paul J. King. Paper "The Type/Token Distinction: Escaping the Escher Staircase". To the 'What Feature Structures Do' workshop at *Between Syntax, Semantics and Logic*. Stuttgart, Germany.

29th October, 1993. Erhard W. Hinrichs. Invited paper "NP Aufspaltung: eine HPSG Analyse". To the *Linguistic Colloquium of the* IBM *Wissenschaftliches Zentrum*. Heidelberg, Germany.

19th November, 1993. Dale Gerdemann. Paper "Typed Feature Structures for Expressing and Computationally Implementing Feature Co-occurrence Restrictions" (coauthored by Paul J. King). To the *4. Fachtagung der Sektion Computerlinguistik der Deutschen Gessellschaft für Sprachwissenschaft*. Hamburg, Germany.

9th January, 1994. Erhard W. Hinrichs & Tsuneko Nakazawa. Paper "Split NP Topicalization in German: An HPSG Analysis". To the *Annual Meeting of the Linguistic Society of America*. Boston, Massachusetts, USA.

10th January, 1994. Erhard W. Hinrichs. Invited Paper "Quantifier Scope Ambiguities". To the NTT *Basic Research Laboratory*. Tokyo, Japan.

28th January, 1994. Erhard W. Hinrichs. Invited paper "Implementation of HPSG Grammars for German in the Troll System". To the NTT *Yokosuka Research and Development Center*. Yokosuka, Japan.

9th January, 1994. Erhard W. Hinrichs & Tsuneko Nakazawa. Paper "Split NP Topicalization in German: An HPSG Analysis". To *The Annual Meeting of the Linguistic Society of America*. Boston, Massachusetts, USA.

19th May, 1994. Erhard W. Hinrichs. Invited paper "HPSG for German". To the *Kyoto Workshop for German-Japanese Machine Translation*. Kyoto, Japan.

15th–20th August, 1994. Tsuneko Nakazawa & Erhard W. Hinrichs. Worskshop 'Grammar Formalisms for NLP'. At the *6th European Summer School in Logic, Language and Information*. Copenhagen, Denmark.

August, 1994. Dale Gerdemann. Paper "Parsing as Tree Traversal". To COLING-94. Kyoto, Japan.

August, 1994. Dale Gerdemann. Paper "The Correct and Efficient Implementation of Appropriateness Specifications for Typed Feature Structures" (coauthored by Paul J. King). To COLING-94. Kyoto, Japan.

August, 1994. Paul J. King. Reserve paper "Typed Feature Structures as Descriptions". To COLING-94. Kyoto, Japan.

September 1994. Paul J. King. Invited paper. To the *Logic, Structures and Syntactic Theorising Workshop*. Utrecht, The Netherlands.

7.3 Veröffentlichungen und Arbeitspapiere

Dale Gerdemann (1994a). "Complement Inheritance as Subcategorization Inheritance". In *HPSG for German*. CSLI Lecture Notes. CSLI, Stanford, California, USA.

Dale Gerdemann (1994b). "Parsing as Tree Traversal". In *Proceedings of* COLING-94. Kyoto, Japan.

Dale Gerdemann & Erhard W. Hinrichs (in press). "Some Open Problems in Head-driven Generation". In *Linguistics and Computation*. CSLI Lecture Notes. CSLI, Stanford, California, USA.

Dale Gerdemann & Paul J. King (1993). "Typed Feature Structures for Expressing and Computationally Implementing Feature Co-occurrence Restrictions". In *Proceedings der 4. Fachtagung der Sektion Computerlinguistik der Deutschen Gessellschaft für Sprachwissenschaft*, pages 33–39. Hamburg, Germany.

Dale Gerdemann & Paul J. King (1994). "The Correct and Efficient Implementation of Appropriateness Specifications for Typed Feature Structures". In *Proceedings of* COLING-94. Kyoto, Japan.

Thilo Götz (1993). *A Normal Form for Typed Feature Structures*. Magisterarbeit. Eberhard-Karls-Universität, Tübingen, Germany. Published as SFB 340 technical report, number 40. IBM Deutschland GmbH, Heidelberg, Germany.

John Griffith (in preparation). *Disjunction and Efficient Processing of Feature Descriptions*. Doctoral dissertation.

Erhard W. Hinrichs & Tsuneko Nakazawa (1993). “Aspects of German VP Structure: An HPSG Account”. Technical report, SFS-Report-01-93. Seminar für Sprachwissenschaft, Eberhard-Karls-Universität, Tübingen, Germany.

Erhard W. Hinrichs & Tsuneko Nakazawa (1994). “Partial-VP and Split-NP Topicalization in German: An HPSG Analysis”. In Erhard W. Hinrichs, Detmar Meurers and Tsuneko Nakazawa, *Partial-VP and Split-NP Topicalization in German: An HPSG Analysis and its Implementation*. SFB 340 technical report. IBM Deutschland GmbH, Heidelberg, Germany.

Erhard W. Hinrichs & Tsuneko Nakazawa (in press). “Linearizing AUXs in German Verbal Complexes”. In John Nerbonne, Klaus Netter and Carl Pollard (editors), *German Grammar in HPSG*. CSLI Lecture Notes. CSLI, Stanford, California, USA.

Stephan Kepser (in preparation). *A Satisfiability Algorithm for a Logic for Typed Feature Structures*. Magisterarbeit. Eberhard-Karls-Universität, Tübingen, Germany.

Paul J. King (1994a). “Typed Feature Structures as Descriptions”. In *Proceedings of* COLING-94. Kyoto, Japan.

Paul J. King (1994b). “An Expanded Logical Formalism for Head-driven Phrase Structure Grammar”. SFB 340 technical report. IBM Deutschland GmbH, Heidelberg, Germany.

Paul J. King & Thilo Götz (1993). “Eliminating the Feature Introduction Condition by Modifying Type Inference”. SFB 340 technical report, number 31. IBM Deutschland GmbH, Heidelberg, Germany.

Dieter Martini (1994). *Elimination von Linksrekursion bei Definite Clause Grammatiken*. Magisterarbeit. Eberhard-Karls-Universität, Tübingen, Germany.

Detmar Meurers (1994). “On Implementing an HPSG Theory”. In Erhard W. Hinrichs, Detmar Meurers and Tsuneko Nakazawa, *Partial-VP and Split-NP Topicalization in German: An HPSG Analysis and its Implementation*. SFB 340 technical report. IBM Deutschland GmbH, Heidelberg, Germany.

Detmar Meurers (in preparation). *Aspects of the German Verbal Complex: Empirical Overview and HPSG Analysis*. Magisterarbeit. Eberhard-Karls-Universität, Tübingen, Germany.

Detmar Meurers & Thilo Götz (1993). “Some Notes on Using TFS and CUF”. Unpublished manuscript.

Detmar Meurers & Frank Morawietz (1993). “An Alternative Approach to ID/LP-SUP Grammars”. Unpublished manuscript.

Guido Minnen (1992). "Earley-type Parsing of TAG and Factoring Linear Precedence". In *Proceedings of the 2nd* TAG+ *Workshop*. Philadelphia, Pennsylvania, USA.

Guido Minnen (1994). "Predictive Left to Right Parsing of a Restricted Variant of TAG(ID/LP)". SFB 340 technical report. IBM Deutschland GmbH, Heidelberg, Germany.

Guido Minnen (in preparation). *Bidirectionality: Compilation Techniques for Analysis and Synthesis of Natural Language*. Doctoral dissertation.

Guido Minnen & Dale Gerdemann (1993). "Direct Automated Inversion of Logic Grammars". SFB 340 technical report, number 35. IBM Deutschland GmbH, Heidelberg, Germany.

Guido Minnen, Dale Gerdemann & Erhard W. Hinrichs (1993) "Direct Automated Inversion of Logic Grammars". In *Proceedings of the 4th Workshop on Natural Language Understanding and Logic Programming*. Nara, Japan.

Frank Morawietz (in preparation). *Direct Parsing of Typed* ID/LP *Unification Grammars*. Magisterarbeit. Eberhard-Karls-Universität, Tübingen, Germany.

Kiril I. Simov & Paul J. King (in preparation). "Speciate Re-entrant Logic as a KL-ONE Knowledge Representation Language."

8 Literatur

Hassan Aït-Kaci (1985). *A New Model of Computation Based on a Calculus of Type Subsumption*. Doctoral dissertation. University of Pennsylvania, Philadelphia, Pennsylvania, USA.

Gunnar Bech (1955/57). *Studien über das deutsceh verbum infinitum*. Historisk-filologiske Meddelelser udgivet at Det Kongelige Danske Videnskabernes Selskab. Copenhagen, Denmark.

Catriel Beeri & Raghu Ramakrishnan (1991). "On the Power of Magic". In *Journal of Logic Programming*, volume 10, number 2, pages 255–299.

Ronald J. Brachman & James G. Schmolze (1985). "An Overview of the KL-ONE Knowledge Representation System". In *Cognitive Science*, volume 9, number 2, pages 171–216.

Ted J. Briscoe (1991). "Lexical Issues in Natural Language Processing". In Ewan Klein and Frank Veltman (editors), *Natural Language and Speech*, pages 39–68. Springer-Verlag, Berlin, Germany.

Bob Carpenter (1992). *The Logic of Typed Feature Structures*. Cambridge Tracts in Theoretical Computer Science, number 32. Cambridge University Press, Cambridge, England.

Bob Carpenter (1993). ALE*: the Attribute Logic Engine*. User's guide. Carnegie Mellon University, Pittsburg, Pennsylvania, USA.

Noam Chomsky (1981). *Lectures on Government and Binding.* Foris, Dordrecht, The Netherlands.

Ann Copestake, Antonio Sanfilippo, Ted J. Briscoe & Valeria de Paiva (1993). "The ACQUILEX LKB: An Introduction". In Ted J. Briscoe, Ann Copestake & Valeria de Paiva (editors), *Inheritance, Defaults and the Lexicon*, pages 164–189. Cambridge University Press, Cambridge, England.

Michael Dorna (1992). "Erweiterung der Constraint-Logiksprache CUF um ein Typsystem". Diplomarbeit, number 896. Universität Stuttgart, Stuttgart, Germany.

Jochen Dörre & Andreas Eisele (1990). "Feature Logic with Disjunctive Unification". In *Proceedings of* COLING-90, volume 2, pages 100–105. Helsinki, Finland.

Marc Dymetman (1992a). "A Generalized Greibach Normal Form for Definite Clause Grammars". In *Proceedings of* COLING-92, pages 366–372. Nantes, France.

Marc Dymetman (1992b). *Transformations de Grammaires Logiques: Applications au Problème de la Reversibilité an Traduction Automatique.* Thèse d'état. Université de Grenoble, Grenoble, France.

Marc Dymetman & Pierre Isabelle (1988). "Reversible Logic Grammars for Machine Translation". In *Proceedings of the 2nd International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages.* Pittsburgh, Pennsylvania, USA.

Marc Dymetman, Pierre Isabelle & François Perrault (1990). "A Symmetrical Approach to Parsing and Generation". In *Proceedings of* COLING-90, pages 90–96. Helsinki, Finland.

Andreas Eisele & Jochen Dörre (1990). "A Comprehensive Unification Formalism". DYANA-2, Deliverable R1.2.A.

Judith Engelkamp, Gregor Erbach & Hans Uszkoreit (1992). "Handling Linear Precedence Constraints by Unification". In *Proceedings of the 30th Annual Meeting of the* ACL, pages 201–208. Newark, Delaware, USA.

Alex Franz (1990). "A Parser for HPSG". Laboratory for Computational Linguistics technical report, number LCL-90-3. Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

Gerald Gazdar, Ewan Klein, Geoffrey Pullum & Ivan Sag (1985). *Generalized Phrase Structure Grammar.* Harvard University Press, Cambridge, Massachusetts, USA.

Dale Gerdemann (1989). "Using Restriction to Optimize Parsing". In *Proceedings of the International Workshop on Parsing Technologies*, pages 8–17. Pittsburg, Pennsylvania, USA.

Dale Gerdemann (1991). *Parsing and Generation of Unification Grammars.* Doctoral dissertation. University of Illinois, Urbana-Champaign, Illinois, USA. Published as Beckman Institute Cognitive Science technical report, number CS-91-06. The Beckman Institute, Urbana-Champaign, Illinois, USA.

Dale Gerdemann & Erhard W. Hinrichs (1988). "Unicorn: a Unification Parser for Attribute-Value Grammars". In *Studies in the Linguistic Sciences*, volume 18, number 2, pages 41–86.

Gisbert Fanselow (1988). "Aufspaltung von NPs und das Problem der 'freien' Wortstellung". In *Linguistische Berichte*, volume 114, pages 91–111.

Hubert Haider (1990). "Topicalization and Other Puzzles of German Syntax". In Günther Grewendorf and Wolfgang Sternefeld (editors), *Scrambling and Barriers*. John Benjamins Publisher, Amsterdam, The Netherlands.

Erhard W. Hinrichs & Tsuneko Nakazawa (1989a). "Flipped out: AUX in German". In *Proceedings of the 25th Annual Meeting of the Chicago Linguistic Society*. Chicago, Illinois, USA.

Erhard W. Hinrichs & Tsuneko Nakazawa (1989b). "Subcategorization and VP Structure in German". Paper presented to the '3rd Symposium on Germanic Linguistics' at Purdue University. To appear in Shaun Hughes (editor), *Proceedings of the 3rd Symposium on Germanic Linguistics*. John Benjamins Publisher, Amsterdam, The Netherlands. Also published in Hinrichs & Nakazawa (1993).

Erhard W. Hinrichs & Tsuneko Nakazawa (1991). "Linearizing AUXs in German Verbal Complexes". Paper presented to the 'HPSG in German' conference at the University of the Saarland, Saarbrücken, Germany. To appear in John Nerbonne, Klaus Netter and Carl Pollard (editors), *German Grammar in HPSG*. CSLI Lecture Notes. CSLI, Stanford, California, USA.

Mark Johnson (1988). *Attribute-value Logic and the Theory of Grammar*. CSLI Lecture Notes, number 16. CSLI, Stanford, California, USA.

Robert T. Kasper (1987). "A Unification Method for Disjunctive Feature Structures". In *Proceedings of the 25th Annual Meeting of the ACL*, pages 235–242. Stanford, California, USA.

Paul J. King (1989). *A Logical Formalism for Head-driven Phrase Structure Grammar*. Doctoral dissertation. Manchester University, Manchester, England.

Tibor Kiss (1993). "Infinite Komplementation. Neue Studien zum deutschen Verbum infinitum". SFB 282 technical report, number 42. Heinrich Heine Universität, Düsseldorf, Germany.

Miroslav Martinovic & Tomek Strzalkowski (1992). "Comparing Two Grammar-based Generation Algorithms: A Case Study". In *Proceedings of the 30th Annual Meeting of the ACL*, pages 81–88. Newark, Delaware, USA.

Yuji Matsumoto, Hozumi Tanaka, Hideki Hirakawa, Hideo Miyoshi & Hideki Yasukawa (1983). "BUP: A Bottom-up Parser Embedded in Prolog". In *New Generation Computing*, volume 1, number 2, pages 145–158.

John T. Maxwell (III) & Ronald M. Kaplan (1989). "An Overview of Disjunctive Constraint Satisfaction". In *Proceedings of the International Workshop on Parsing Technologies*, pages 18–27. Pittsburg, Pennsylvania, USA.

M. Andrew Moshier (1988). *Extensions to Unification Grammar for the Description of Programming Languages*. Doctoral dissertation. University of Michigan, Ann Arbor, Michigan, USA.

M. Andrew Moshier & William C. Rounds (1987). "A Logic for Partially Specified Data Structures". In *Proceedings of the 14th* ACM *Symposium on Principles of Programming Languages*, pages 156–167. München, Germany.

Bernhard Nebel & Gert Smolka (1988). *Representation and Reasoning with Attributive Descriptions*. IWBS technical report, number 81. IBM Deutschland GmbH, Stuttgart, Germany.

John Nerbonne (1991). "Partial verb Phrases and Spurious Ambiguities". Paper presented to the 'HPSG in German' conference at the University of the Saarland, Saarbrücken. To appear in John Nerbonne, Klaus Netter and Carl Pollard (editors), *German Grammar in* HPSG. CSLI Lecture Notes. CSLI, Stanford, California, USA.

John Nerbonne, Masayo Iida & William Ladusaw (1989). "Running on Empty: Null Heads in Head-driven Grammar". In *Proceedings of* WCCFL-8, pages 276–288. Vancouver, British Columbia, Canada.

Klaus Netter (1991). "Towards a Theory of Functional Heads: German Nominal Phrases". Paper presented to the 'HPSG in German' conference at the University of the Saarland, Saarbrücken. To appear in John Nerbonne, Klaus Netter and Carl Pollard (editors), *German Grammar in* HPSG. CSLI Lecture Notes. CSLI, Stanford, California, USA.

Gertjan van Noord (1993). *Reversibility in Natural Language Processing*. Doctoral dissertation. University of Utrecht, Utrecht, The Netherlands.

Valeria de Paiva (1993). "Types and Constraints in the LKB". In Ted J. Briscoe, Ann Copestake & Valeria de Paiva (editors), *Inheritance, Defaults and the Lexicon*, pages 164–189. Cambridge University Press, Cambridge, England.

Fernando C. N. Pereira & Stuart Shieber (1987). *Prolog and Natural Language Analysis*. CSLI Lecture Notes, number 10. CSLI, Stanford, California, USA.

Fernando C. N. Pereira & David H. D. Warren (1983). "Parsing as Deduction". In *Proceedings of the 21st Annual Meeting of the* ACL, pages 137–144. Cambridge, Massachusetts, USA.

Carl J. Pollard (1990). "On head non-movement". In *The Tilburg Symposium on Discontinuous Constituents*. Tilburg, The Netherlands.

Carl J. Pollard & Ivan A. Sag (1987). *Information-based Syntax and Semantics*. CSLI Lecture Notes, number 13. CSLI, Stanford, California, USA.

Carl J. Pollard & Ivan A. Sag (1994). *Head-driven Phrase Structure Grammar*. University of Chicago Press, Chicago, Illinois, USA.

Raghu Ramakrishnan (1991). "Magic Templates: a Spellbinding Approach to Logic Programs". In Robert A. Kowalski & Kenneth A. Bowen (editors.) *Logic Programming: Proceedings of the 5th International Conference and Symposium*, volume 1, pages 140–159. MIT Press in Logic Programming. MIT Press, Cambridge, Massachusetts, USA.

Henk van Riemsdijk (1987). "Movement and regeneration". In Paola Neninca (editor), *Linguistic Theory and Dialectology*. Foris, Dordrecht, The Netherlands.

William C. Rounds & Robert T. Kasper (1986). "A Complete Logical Calculus for Record Structures Representing Linguistic Information". In *Proceedings of the 15th Annual* IEEE *Symposium on Logic in Computer Science*, pages 38–43. Cambridge, Massachusetts, USA.

Roland Seiffert (1987). "Chart-Parsing of Unification-based Grammars with ID/LP-rules." In Ewan Klein and Johan van Benthem (editors), *Categories, Polymorphism and Unification*, pages 335–354. Centre for Cognitive Science, University of Edinburgh, Edinburgh, England; Institute for Language, Logic and Information, University of Amsterdam, Amsterdam, The Netherlands.

Stuart Shieber (1984). "Direct Parsing of ID/LP Grammars". In *Linguistics and Philosophy*, volume 7, number 2, pages 135–154.

Stuart Shieber (1985). "Using Restriction to Extend Parsing Algorithms for Complex-Feature-Based Formalisms". In *Proceedings of the 23rd Annual Meeting of the* ACL, pages 145–152. Chicago, Illinois, USA.

Stuart Shieber (1988). "A Uniform Architecture for Parsing and Generation". In *Proceedings of* COLING-88, pages 614–619. Budapest, Hungary.

Stuart Shieber, Gertjan van Noord, Fernando C. N. Pereira & Robert Moore (1990). "Semantic Head-Driven Generation". In *Computational Linguistics*, volume 16, number 1, pages 30–42.

Gert Smolka (1988). "A Feature Logic with Subsorts". LILOG technical report, number 33. IBM Deutschland GmbH, Stuttgart, Germany.

Tomek Strzalkowski (1990a). "How to Invert a Parser into an Efficient Generator: an Algorithm for Logic Grammars". In *Proceedings of* COLING-90, volume 2, pages 347–352. Helsinki, Finland.

Tomek Strzalkowski (1990b). "Reversible Logic Grammars for Natural Language Parsing and Generation". In *Canadian Computational Intelligence Journal*, volume 6, number 3, pages 145–171.

Tomek Strzalkowski (1991). "A General Computational Method for Grammar Inversion". In *Proceedings of the Reversible Grammar Workshop*, pages 91–99. Berkeley, California, USA.

Tomek Strzalkowski (1993). "A General Computational Method for Grammar Inversion". In Tomek Strzalkowski (editor), *Reversible Grammar in Natural Language Processing*. Kluwer Academic Publishers, Dordrecht, The Netherlands. This is an article based on the workshop paper Strzalkowski (1991).

Hisao Tamaki & Taisuke Sato (1984). "Unfold/Fold Transformation of Logic Programs". In *Proceedings of the 2nd International Logic Programming Conference*, pages 127–138. Uppsala, Sweden.

Craig Thiersch (1978). *Topics in German Syntax*. Doctoral dissertation. MIT, Cambridge, Massachusetts, USA.

Hans Uszkoreit (1987). "Linear Precedence in Discontinuous Constituents: Complex Fronting in German". In *Syntax and Semantics*, volume 20, pages 406-424. Academic Press, San Diego, California, USA.

David H. D. Warren (1983). "An Abstract Prolog Instruction Set". Technical note 309. SRI International, Menlo Park, California, USA.

David S. Warren (1992). "Memoing for Logic Programs with Applications to Abstract Interpretation and Partial Deduction". In *Communications of the* ACM, volume 35, number 3.

Jürgen Wedekind (1991). "Classical Logics for Attribute-value Language". In *Proceedings of the* EACL, pages 204–209.

Rémi Zajac (1992). "Inheritance and Constraint-Based Grammar Formalisms". In *Computational Linguistics*, volume 18, number 2, pages 181–203.