Distributed Parsing With HPSG Grammars

Abdel Kader Diagne, Walter Kasper, Hans-Ulrich Krieger

RR-95-19
Research Repo

**Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH**

**Research
Report**

RR-95-19

Distributed Parsing With HPSG Grammars

**Abdel Kader Diagne, Walter Kasper,
Hans-Ulrich Krieger**

December 1995

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, Sema Group, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry of Education, Science, Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct systems with technical knowledge and common sense which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Computer Linguistics
- Programming Systems
- Deduction and Multiagent Systems
- Document Analysis and Office Automation.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Dr. Dr. D. Ruland
Director

Distributed Parsing With HPSG Grammars

Abdel Kader Diagne, Walter Kasper, Hans-Ulrich Krieger

DFKI-RR-95-19

A version of this paper has been published in: Proceedings of the Fourth International Workshop on Parsing Technologies, IWPT'95, September 20–24, 1995, Prague and Karlovy Vary, Czech Republic.

This work has been supported by a grant from The Federal Ministry of Education, Science, Research and Technology (FKZ ITWM-Verbmobil 01 IV 101 K/1).

ISSN 0946-008X

Distributed Parsing With HPSG Grammars

Abdel Kader Diagne, Walter Kasper, Hans-Ulrich Krieger
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
{diagne,kasper,krieger}@dfki.uni-sb.de

Abstract

Unification-based theories of grammar allow for an integration of different levels of linguistic descriptions in the common framework of typed feature structures. Dependencies among the levels are expressed by coreferences. Though highly attractive theoretically, using such codescriptions for analysis create problems of efficiency. We present an approach to a modular use of codescriptions on the syntactic and semantic level. Grammatical analysis is performed by tightly coupled parsers running in tandem, each using only designated parts of the grammatical description. In the paper we describe the partitioning of grammatical information for the parsers and present results about the performance.

Acknowledgements. We are grateful to three anonymous IWPT referees and the audience at the 4th IWPT.

Contents

1 Introduction

A fundamental concept of Head-Driven Phrase Structure Grammar (HPSG) is the notion of a SIGN (Pollard and Sag 1987, Pollard and Sag 1994). A SIGN is a structure integrating information from all levels of linguistic analysis, such as phonology, syntax, and semantics. This structure also specifies interactions between these levels by means of coreferences which indicate the sharing of information and how the levels constrain each other mutually. Such a concept of linguistic description is attractive for several reasons:

1. it supports the use of common formalisms and data structures on all levels of linguistic descriptions,
2. it provides declarative and reversible interface specifications between the levels,
3. all information is simultaneously available,
4. no procedural interaction between linguistic modules needs to be set up.

Similar approaches, especially for the syntax-semantics interface, have been suggested for all major unification-based theories of grammar, such as LFG or CUG. For these theories and their underlying formalisms it was shown how to provide at least partial and underspecified semantic descriptions in parallel to syntax. Halvorsen and Kaplan 1988 call such approaches *codescriptive* in contrast to the approach of *description by analysis* which is closely related to sequential architectures where linguistic levels correspond to components which operate on the basis of the (complete) analysis results of lower levels.

Unification-based theories of grammar are expressed in feature-structure formalisms by equational constraints. Semantic descriptions are expressed there by additional constraints.

Though theoretically very attractive, codescription has its price:

- difficult to modularize
- computational overhead when parsers use the complete descriptions

Problems of these kinds which were already noted by Shieber 1985 motivated the research described here. The goal is to develop more flexible ways of using codescriptive grammars than having them applied by a parser with full informational power. The underlying observation is that constraints in such grammars can play different roles:

- **Genuine constraints** which take effect as filters on the input. These relate directly to the grammaticality (wellformedness) of the input. Typically, these are the syntactic constraints.

- **Spurious constraints** which basically build representational structures. These are less concerned with wellformedness of the input but rather of the output for other components in the overall system. Much of semantic descriptions is of this kind.

If the parser generated from such a grammar specification treats all constraints on a par, it cannot distinguish between the structure-building and the analytical constraints. Since unification-based formalisms are monotonic, large structures are built up and have to undergo all the steps of unification, copying, and undoing in the processor. The cost of these operations (in time and space) increase exponentially with the size of the structures.

In the VERBMOBIL project (Wahlster 1993, Kay et al. 1994) the grammar parser is used in the context of a speech translation system. The parser input consists of word lattices of hypotheses from speech recognition. The parser has to identify those paths in the lattice which represent a grammatically acceptable utterance. Parser and recognizer are incremental and interactively running in parallel. Even for short utterances, the lattices can contain several hundred of word hypotheses and paths, most of which are not acceptable grammatically.

The basic idea presented here is to distribute the labour of evaluating the constraints in the grammar on several processes. Important considerations in the design of the system[1] were:

- increase in performance
- maintenance of an incremental and interactive architecture of the system
- minimize the overhead in communication between the processors

Several problems must be solved for such a system:

- the ability to work with partial (incomplete) analysis
- synchronizing the processors: especially the parsers have to exchange information about success and failure in analysis

In the following sections, we will discuss these constraints in more detail. After that, we will describe the communication protocol between the parsing processes. Then several options for creating subgrammars from the complete grammar will be discussed. The subgrammars represent the distribution of information across the parsers. Finally, some experimental results will be reported.

We used a mid-size German grammar written in the typed feature-based formalism $\mathcal{TDL}$ (Krieger and Schäfer 1994) which covers dialogs collected in

[1] The *system* alluded to here and below which provided the context of this work, was the INTARC-II prototype of VERBMOBIL which was officially presented in April 1995.

VERBMOBIL. In this system, principles in the sense of HPSG are defined as types which are inherited by the grammar rules. The grammar cospecifies syntax and semantics in the attributes SYN and SEM. To facilitate experimentation with distributed processing a slightly unconventional SIGN structure was chosen. Additionally to the SYN and SEM attributes for syntactic/semantic descriptions, some features were singled out as control structures for the derivation processes which are shared by the subgrammars. These "controller" features include the SUBCAT list required to define completeness and coherence, and features which control discontinuous constituents in German, such as verb prefixes. These features normally would be included in the SYN structure despite their impact on semantics. Furthermore, unique identifiers for grammar rules and lexical entries had to be provided for the communication between the parsers, as will be explained below.

2 The Architecture

The most important aspect for the distribution of analysis and for defining modes of interaction between the analysis processes (parsers) is that one of the processes has to work as a filter on the word lattices, reducing the search space. The other component, however, only work with successful analysis results of the other one. This means that the two parsers do not really run in parallel on the input word lattices. Rather, one parser is in control over the second, which is not directly exposed to the word lattices. For reasons which will become obvious below, we will call the first of these parsers the SYN-*parser*, the second one controlled by the SYN-parser, the SEM-*parser*.

Another consideration to be taken into account is that the analysis should be *incremental* and *time-synchronous*. For the interaction of the parsers, this implies that the SYN-parser must not send its results only when it completely finished its analysis, forcing the SEM-parser to wait.[2]

Interactivity is another aspect we had to consider. The SEM-parser must be able to report back to the SYN-parser, at least when its hypotheses failed. This would not be possible when the SEM-parser has to wait till the SYN-parser is finished. This requirement also constrains the exchange of messages.

Incrementality and interactivity imply a regular exchange of messages between the parsers. An important consideration then is that the overhead for this communication should not outweigh the gains of distributed processing. This consideration rules out that the parsers directly communicate by exchanging their analysis results in terms of resulting feature structures. There are no good ways of communicating feature structures across distinct processes except as (large) strings. This means that the parsers would need the possibility to build

[2]Another problem in incremental processing is, that it is not known in advance when the utterance is finished or a new utterance starts. To deal with this, prosodic information is taken into account. This will not discussed here.

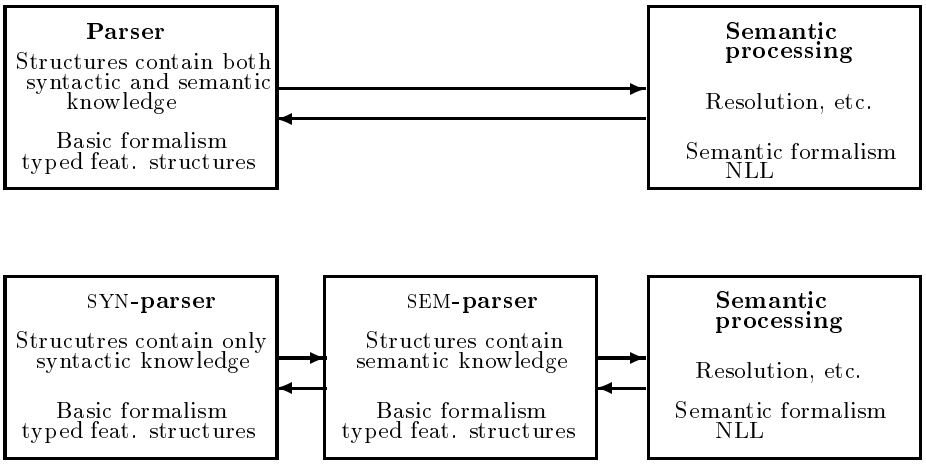

Figure 1: *Syntax/semantics interaction with centralized (upper picture) and distributed parsing.*

strings from feature structures and parse strings into feature structures. Also, on each communication event the parsers would have to analyze the structures to detect changes, whether a structure is part of other already known structures, etc. It is hard to see how this kind of communication can be interleaved with normal parsing activity in efficient ways.

In contrast to this, our approach allows to exploit the fact that the grammars employed by the parsers are derived from the same grammar and thereby similar in structure. This makes it possible to restrict the communication between the parsers to information about what rules were successfully or unsuccessfully applied. Each parser then can reconstruct on his side the state the other parser is in—how its chart or analysis tree looks like. Both parsers try to maintain or arrive at isomorphic charts.

The approach allows that the parsers never need to exchange analysis results in terms of structures as the parsers should always be able to reconstruct these, if necessary. On the other hand, this reconstructibility poses constraints on how the codescriptive grammar can be split up into subgrammars.

The requirements of incrementality, interactivity, and efficient communication show that our approach does not emulate the description-by-analysis methodology in syntax-semantics interfaces on the basis of codescriptive grammars.

The modified processing model is presented in Figure 1.

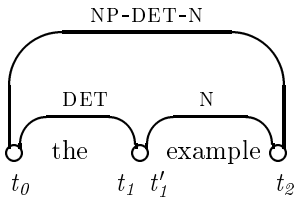

Figure 2: *A complete subtree.*

3 The Parsers

The SYN-parser and the SEM-parser are agenda driven chart parsers. For speech parsing, the nodes represent points in time and edges represent word hypotheses and paths in the word lattice.

The parsers communicate by exchanging *hypotheses*, bottom-up hypotheses from syntax to semantics and top-down hypotheses from semantics to syntax.

- **Bottom-up hypotheses** are emitted by the SYN-parser and sent to the SEM-parser. They undergo verification at the semantic level. A bottom-up hypothesis describes a passive edge (complete subtree) constructed by the syntax parser and consists of the identifier of the rule instantiation that represents the edge and the *completion history* of the constructed passive edge. Having passive status is a necessary but not sufficient condition for an edge to be sent as hypothesis. Whether a hypothesis is sent also depends on other criteria, such as its score. In the actual system the SYN-parser is a probabilistic chart parser, using a statistic language model as an additional knowledge source (Hauenstein and Weber 1994).
- **Top-Down hypotheses** result from activities of SEM-parser trying to verify bottom-up-hypotheses. To keep the communication efforts low, only failures are reported back to the SYN-parser by sending simply the hypothesis' identifier. This should start a chart revision process on the SYN-parser's side.[3]

The central data structure by which synchronization and communication between the parsers is achieved is that of a *completion history*, containing a record on how a subtree was completed. Basically it tells us for each edge in the chart which other edges are spanned. For the chart in Figure 2 the completion history is as follows:

$$c\text{-}hist(\text{NP-DET-N}) := ((\text{DET}\ t_0 t_1)\ (\text{N}\ t'_1\ t_2))$$

[3]This revision process is currently under investigation.

$c\text{-}hist(det) := ((\text{“}the\text{”}\ t_0\ t_1))$

$c\text{-}hist(\mathrm{N}) := ((\text{“}example\text{”}\ t'_1\ t_2))$

t_0, t_1, t'_1 and t_2 are points in time in milliseconds. *Gaps* may occur between two consecutive edges, i.e., it might be that $t'_1 > t_1$ in the example in Figure 2. This is in contrast to standard chart parsing where it is required that end point and starting point of consecutive edges are identical. The relaxation of this constraint helps to reduce the number of bottom-up hypotheses.

The completion history of a given edge should not contain references to other edges than those representing either lexical entries or previously completed subtrees. Depending on the *eagerness* of the underlying parsing and communication strategy, a completion history may be a sequence of lists of edges R $e_1 ... e_n$, where $e_1, \ldots, e_n$ are complete subtrees' labels and R is a rule's identifier, such that applying R to $e_1, \ldots, e_n$ returns a complete subtree described by the underlying completion history. Generally, completion histories are expressions as described by the following EBNF:

edge_id ::= "E" INTEGER
rule_id ::= "R"INTEGER
node_id ::= "N" INTEGER
edge_list ::= rule_id {rule_id {node_id}$^+$ | edge_id}*
completion_history ::= {edge_list M}*

E, R, N and M are delimiters and INTEGER an integer used as identifier. E marks an identifier for a complete subtree which has already been sent as a hypothesis (hypothesis' identifier) to SEM-parser, R an identifier for the rule used to build a complete subtree which has not been sent yet, N the position of the subtree's nodes and M delimits a list of edges representing the completion history of a completed part of the current edge. The following example illustrates a complex completion history:

R 1 R 10052 N 0 N 1 R 10032 N 1 N 2 M
R 4 R 10275 N 2 N 3 R 10311 N 3 N 4 M
R 2 R 1 N 0 N 2 R 4 N 2 N 4 M
R 0 R 2 N 0 N 4 E 1 M

This protocol allows the parsers efficiently to exchange information about the structure of their chart without having to deal with explicit analysis results as feature structures.

Since the SEM-parser does not work directly on linguistic input but is fed by the SYN-parser, there are some differences to ordinary chart parsing. The SEM-parser uses a *two-level agenda mechanism*. The low-level agenda manages the bottom-up hypotheses from the syntax. It is currently a queue, that is, hypotheses are treated in a FIFO manner. The high-level agenda is an agenda

repeat

1. *get next Bottom-Up-Hypothesis (bu-hypo) from low-level agenda.*
2. **if** *got bu-hypo?*

   then goto 3.
   else **repeat** *perform (quasi-)autonomous parsing*
   until *bu-hypo available* ;
   goto 1.

3. *perform non-autonomous parsing based on bu-hypo.*
4. **if** *successful verification*?

   then *confirm bu-hypo.*
   else *reject bu-hypo.*

until SYN-*parser sends a 'no-more-hypothesis' flag* **and**
low-level agenda is empty.

Figure 3: *Control in the* SEM*-parser*

as known from chart parsers with scanning, prediction, and combination tasks. What is special here is that the structure of the high-level-agenda is guided mainly by the low-level-agenda. Since the SEM-parser is controlled by the SYN-parser, there are two possible parsing modes:

- **Non-autonomous parsing.** The parsing process consists mainly of constructing the tree described via the completion history by using the semantic counterparts of the rules which led to the syntactic hypotheses. If this fails (due to semantic constraints), this is reported back to the SYN-parser.
- **Quasi-autonomous parsing.** If no syntactic hypotheses are present, the parser extends the chart on its own, using its rules by prediction and completion steps. Obviously, this is only possible after some initial information by the SYN-parser, since the SEM-parser is not directly connected to the input utterance.

We ignore here that SYN-parser and SEM-parser also receive hypotheses from prosody about *phrase boundaries* and *utterance mood*, also influencing the parsing process. The algorithm for the SEM-parser is shown slightly simplified in Figure 3.

4 Compilation of Subgrammars

In the following sections, we discuss possible options and problems for the distribution of information in a cospecifying grammar. Clearly, in our approach we have to specify which of the parsers uses what information. This set of information is what we call a *subgrammar*. These subgrammars are generated from a common source grammar.

4.1 Reducing Representational Overhead by Separation of Syntax and Semantics

An obvious choice for splitting up the grammar was to separate the linguistic levels (strata), such as syntax and semantics. This choice was also motivated by the observation that typically the most important constraints on grammaticality of the input are in the syntactic part, while most of the semantics was purely representational.[4] A straightforward way to achieve this is by destructively manipulating grammar rules and lexicon entries: in case of the SYN-parser, we delete the information under the SEM attribute and similarly clear the SYN attribute to obtain the subgrammar for the SEM-parser. Notice that such attributes not only occur on top of a feature structure but also inside it, e.g., within the subcategorization list through subcategorized elements (see feature SC in Fig. 4). We abbreviate these subgrammars by G_{syn} and G_{sem} and the original grammar by G.

Let us give an example to see the outcome of such a compilation. Consider the German first person singular form *komme* (*come*). A nearly complete lexicon entry for *komme* is depicted in Fig. 4. Dropping the SYN attribute yields the feature structure in Fig. 5. Notice the size of this structure—it is about 30–40% of the original entry for *komme*. Notice too that shared information between syntax and semantics is duplicated in that such structures will occur in both subgrammars.[5]

What syntax and semantics have in common is actually very little in our example (we call this the *coref skeleton*). This structure is depicted in Fig. 6.

Exactly these coreference constraints are eliminated through our method. This might lead to several problems which we address in Section 4.2. Section 4.3 then discusses possible solutions.

Another, more sophisticated way to keep the structures small is due to the type expansion mechanism in $\mathcal{TDL}$ (Krieger and Schäfer 1995). Instead of de-

[4] This must be taken *cum grano salis* as it depends on how a specific grammar draws the line between syntax and semantics: selectional constraints, e.g., for verb arguments, are typically part of semantics and are true constraints. Also, semantic constraints would have a much larger impact if, for instance, agreement constraints would be considered as semantic too, as Pollard and Sag 1994 suggest.

[5] As mentioned in the beginning, some attributes such as the subcategorization list SC have been taken out of the SYN structure to the top-level. It does not affect the argument.

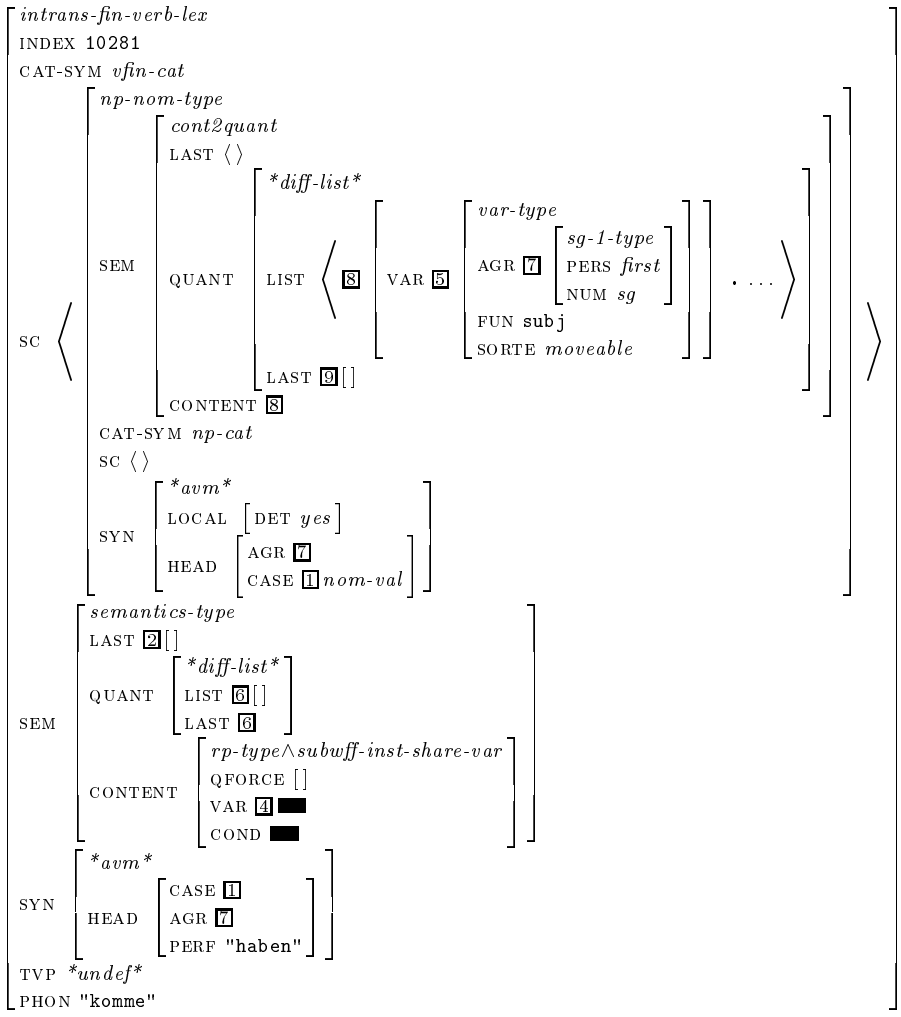

Figure 4: *The lexicon entry for komme. Notice the coreference tag* 7 *under path* SYN|HEAD|AGR *which can also be found in the subcategorization list under feature* SC *(both under* SYN *and* SEM*). Note further that we have omitted some unimportant details of the feature structure, indicated by* ■.

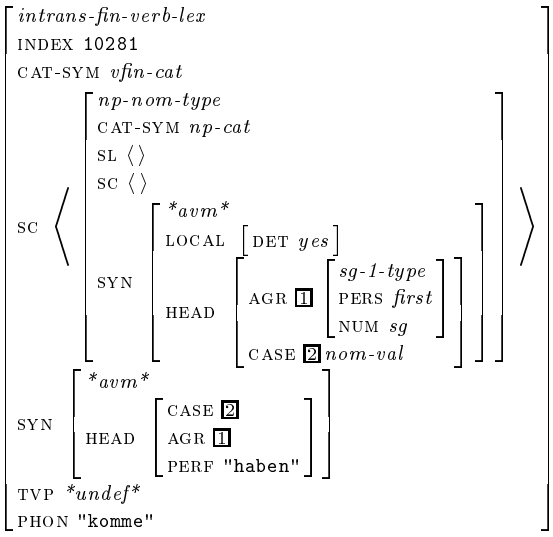

Figure 5: *The syntactic properties of komme. Notice that the coreference tag* 1 *(formerly* 7*) into the semantics of the subcategorized element is no longer established, since there is no semantics.*

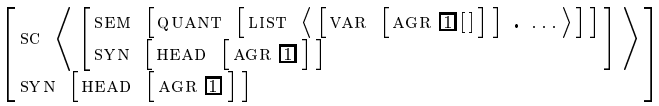

Figure 6: *The coref skeleton of komme.*

structively modifying the feature structures beforehand, we can employ type expansion to let SYN or SEM unexpanded. This has the desired effect that we do not lose the coreference constraints and furthermore are free to expand parts of the feature structure afterwards. We will discuss this feature in Section 4.4. In the actual system this option was not available as its SYN-parser (Hauenstein and Weber 1994) employed a simpler formalism which does not provide type expansion. Therefore the $\mathcal{TDL}$ (sub-)grammar had to be expanded beforehand in order to transform the structures to that formalism.

4.2 Problems

Obviously, the biggest advantage of our method is that unification and copying becomes faster during processing, due to smaller structures. We can even estimate the speed-up in the best case, viz., quasi-linear w.r.t. input structure if only conjunctive structures are used. Clearly, if many disjunctions are involved, the speed-up might even be exponential.

However, the most important disadvantage of the compilation method is that it no longer guarantees *soundness*, that is, the subgrammar(s) might accept utterances which are ruled out by the full grammar. This is due to the simple fact that certain constraints are neglected in the subgrammars. If at least one such constraint is a filtering constraint, we automatically enlarge the language accepted by this subgrammar w.r.t. full grammar. Clearly, *completeness* is not affected, since we do not add further constraints to the subgrammars.

At this point, let us focus on the estimation above, since it is only a best-case forecast. Sure, the structures become smaller (see Fig. 5); however, due to the possible decrease of filter constraints, we must expect an increase of hypotheses in the parser. And in fact, our experimental results in Section 5 show that our approach has a different impact on the SYN-parser and the SEM-parser (see Figure 7). Our hope here, however, is that the increase of non-determinism inside the parser is compensated by the processing of smaller structures (see Maxwell III and Kaplan 1991 for more arguments on this theme).

In general, even the intersection of the languages accepted by G_{syn} and G_{sem} does not yield the language accepted by G:

$$\mathcal{L}(G) \subset \mathcal{L}(G_{syn}) \cap \mathcal{L}(G_{sem})$$

This behaviour is an outcome of our compilation schema, namely, cutting reentrancy points. Thus, even if an utterance S is accepted by G with analysis fs (feature structure), we can be sure that the unification of the corresponding results for G_{syn} and G_{syn} will subsume fs:

$$fs \preceq fs_{syn} \wedge fs_{sem}$$

Let us mention further problems. First, *termination properties* might change in case of the subgrammars. Consider a subgrammar which contains empty

productions or unary (coercion) rules. Assume that such rules were previously "controlled" by constraints that are no longer present. Obviously, if a parser is not restricted through additional, non-grammatical constraints, the iterated application of these rules could lead to an infinite computation, i.e., a loop, inside the parser. This was sometimes the case during our experiments.

Second, recursive rules could introduce infinitely many solutions for a given utterance. Theoretically, this might not pose a problem, since the intersection of two infinite sets of parse trees might be finite.[6] However in practice this problem is hard to avoid.

4.3 Solutions

In this section, we will discuss three solution to the problems mentioned in the last section.

Feedback Loop. Although semantics construction is driven by the speech parser, the use of different subgrammars suggest that the speech parser should also be guided by the SEM-parser. This can be achieved by sending back *falsified* hypotheses. Because hypotheses are uniquely identified in our framework, we must only send the integer that identifies the falsified chart edge. However, such an approach presumes that the SYN-parser is able to revise its chart (cf. Wirén 1992). This idea is currently under implementation.

Coref Skeleton. In order to guarantee overall correctness of analysis, we might unify the results of both parsers with the corresponding coref skeletons at the end. This strategy will not be pursued here since it introduces an additional processing step during parsing. It would be better to employ type expansion here in order to let SYN or SEM unexpanded so that coreferences can be preserved. Exactly this treatment will be investigated in the next section.

Full-Size Grammar. The most straightforward way to guarantee correctness is simply by employing the full-size grammar in one of the two parsers. This might sound strange, but recall that we process speech input so that even a small grammar constrains possible word hypotheses. We suggest that the SEM-parser should operate on the full-size grammar since the speech parser directly communicates with the word recognizer and must process an order of magnitude more hypotheses than the SEM-parser. Because the SEM-parser passes its analysis on to the semantic evaluation module, it makes further sense to guarantee correctness here. This has been the final set-up during our experiments.

[6] To be more precise, intersection means here to take only those annotated derivation trees which have the same context-free skeleton.

4.4 Improvements

This section investigates several improvements of our compilation approach which solve the problems mentioned before.

Identifying Functional Strata Manually. Usually, the grammar writer "knows" which information needs to be made explicit. Hence, instead of differentiating between the linguistic strata SYN and SEM, we let the linguist identify which constraints filter and which only serve as a means for representation. In contrast to the separation along linguistic levels, this approach adopts a functional view cutting across linguistic strata. On this view, the syntactic constraints together with, e.g., semantic selection constraints would constitute a subgrammar.

We can use $\mathcal{TDL}$'s type expansion mechanism to make only the relevant information explicit. This can be achieved by specifying paths (absolute or relative) which participate in a "true" unification. Clearly, there are several disadvantages: (i) the separation is grammar-dependent, (ii) it might depend on the feature geometry, and (iii) it heavily depends on the intuition of the grammarian. Here is an example of control information in the concrete syntax of $\mathcal{TDL}$, telling the expansion mechanism which information should be expanded:

```
defcontrol :global
           ((:expand (*.SYN)
                     (SEM.CONTENT.VAR.SORTE))).
```

Bookkeeping Unifications. A semi-automatic way to determine true constraints w.r.t. a *training corpus* is simply by bookkeeping feature unification. Features that occur only once on top of the input feature structures do not specialize the information in the resulting structure (actually the values of these features). Furthermore, unrestricted features (i.e., typed to $\top$) do not constrain the result. For instance,

$$\begin{bmatrix} \textsc{a}\ s \\ \textsc{b}\ \begin{bmatrix} \textsc{d}\ w \end{bmatrix} \\ \textsc{c}\ u \end{bmatrix} \wedge \begin{bmatrix} \textsc{a}\ v \\ \textsc{b}\ \top \end{bmatrix} = \begin{bmatrix} \textsc{a}\ s \wedge v \\ \textsc{b}\ \begin{bmatrix} \textsc{d}\ w \end{bmatrix} \\ \textsc{c}\ u \end{bmatrix}$$

indicates that only the path A needs to be made explicit, since its value is more specific than the corresponding input values: $s \wedge v \preceq s$ and $s \wedge v \preceq v$.

Clearly, such a strategy presumes that we are able to bookkeep unifications during processing and afterwards evaluating the statistics to tell the type expansion mechanism which paths inside a feature structure have to be made explicit. Such a mechanism is currently under implementation (Weyers 1996). Ideally, we get a table like the following after a training run:

Feature		Failure		Success		Expec-
Name	Occur.	#	øDepth	#	øDepth	tation
CAT	3	2585	27	1168	48	34
SYN	12	2092	28	1795	37	32
HEAD	14	1223	6	2667	13	11
LOCAL	11	1132	28	2116	15	20
MAJ	12	642	3	2363	4	4
FIRST	21	528	8	2090	7	8
N	12	465	1	1998	1	1
LIST	12	460	8	5731	2	3
MORPH	11	427	10	2605	3	4
V-SUBCAT	10	394	45	1679	5	13
SEM	13	363	8	2730	11	11
SUBCAT	11	302	8	2567	3	4
FIN	6	240	3	765	4	4
CONTENT	13	212	5	2721	5	5

Again, several disadvantages are inherent to this method: (i) the separation is grammar-dependent, (ii) it is corpus-dependent, (iii) it might depend on the feature geometry, and (iv) it is a semi-automatic method.

Partial Evaluation. Partial evaluation, as known from logic programming (Warren 1992), is a method of carrying out parts of computation at compile time that would otherwise be done at run time, hence improving run time performance of logic programs. Analogous to partial evaluation of definite clauses, we can partially evaluate annotated grammar rules, since they drive the derivation. Take for instance the following grammar rule in $\mathcal{TDL}$ (this is a type definition!)

```
np-det-rule :=
  max-head-1-rule & [ CAT-SYM #cat ]
  -->
  < det-cat-type,
    n-bar-cat-type & [ SYN.LOCAL.DET no,
                       CAT-SYM #cat ] >.
```

which might be written à la GPSG somewhat simplified as

$$\text{NP} \longrightarrow \text{D } \overline{\text{N}}$$

Notice that `max-head-1-rule`, `det-cat-type`, and `n-bar-cat-type` are types which participate in type inheritance and abbreviate complex typed feature structure. Partial evaluation means here to substitute type symbols through their expanded definitions.

Because a grammar contains finitely many rules of the above form and because the daughters (the right hand side of the rule) are type symbols (and

there are only finitely many of them), this partial evaluation process can be performed off-line. Now, only those features are made explicit which actively participate in unification during partial evaluation. In contrast to the previous method, partial evaluation is corpus-independent.

Clearly, partial evaluation must not halt, due to recursive rules and coreference constraints. However, *lazy type expansion* as implemented in $\mathcal{TDL}$, is a way to cope with recursive types, such that partial evaluation often terminates in practice. Metaconstraints, like the "offline parsability" or "resolved typed feature structures", are another way to enforce termination here.

Type Expansion. We have indicated earlier that type expansion can be fruitfully employed to preserve the coref skeleton (let SYN or SEM unexpanded, resp.). Type expansion can also be chosen to expand parts of a feature structure on the fly at run time.

The general idea here is as follows. Guaranteeing that the lexicon entries and the rules are consistent, we let everything unexpanded unless we are enforced to make structure explicit. As was the case for the previous two strategies, this is only necessary if a path is introduced in the resulting structure which value is more specific than the value(s) in the input structure(s).

The biggest advantage of this approach is obvious—only those constraints must be "touched" which are involved in restricting the set of possible solutions, i.e., this method is optimal w.r.t. the size of a feature structure and w.r.t. a specific utterance. Clearly, such a test should be done every time the chart is extended. The cost of such tests and the on-line type expansions need further investigation.

5 Experimental Results

This section presents experimental results (Fig. 7) of our compilation method which indicate that the simple SYN/SEM separation does not perfectly match the distinction between true and spurious constraints. The measurements have been obtained w.r.t. 56 sentences from 4 dialogs of the VERBMOBIL corpus. During the measurements, we used as SYN-parser a simple bottom-up chart-parser. Also, no language model was used, nor other information from acoustics.

The column **Syn** shows that parsing with syntax only takes 50% of the time of parsing with the complete grammar (**SynSem**). The number of readings, hypotheses, and chart edges only slightly increase here. Recall that grammar rules and lexicon entries for G_{syn} are about 30–40% of size of the corresponding structures for G. Surprisingly, however, by employing G_{sem} only, run time efficiency decreases massively (**Sem**: 150%), due to the increase in the number of hypotheses (153%). This indicates that most of the filtering constraints are specified in the syntax. Consequently, the incremental version of semantics construction (**Sem-I**) is *in total* even more worse, due to the incremental behaviour

number of sentences: 56									
average length: 7.6									
	SynSem	**Syn**		**Sem**		**Sem-I**		**SynSem-I**	
			%		%		%		%
run time:	30.6	15.2	50	45.8	150	51.8	169	32.1	105
#readings:	1.7	2.1	123	1.8	105	4.2	247	3.8	224
#hypotheses:	53.0	58.1	110	81.3	153	*4.2		*4.6	
#chart edges:	192.0	215.0	112	301.1	156	361.7	188	227.7	119

Figure 7: *Experimental results of* SYN/SEM *separation. Run time of the* SYN*-parser without* SEM *is reduced, despite of overgeneration (see hypotheses). The efficiency of parsing without the* SYN *attribute is shown in the* **Sem** *column. It performs considerably worse w.r.t. integrated approach. The two last columns indicate the performance in incremental mode. * indicates the number of top-down hypotheses which are sent back by the feedback loop to the* SYN*-parser (see Section 4.3). The percentage specifications are relative to* **SynSem**. *During these measurements, the* SEM*-parser runs in the quasi-autonomous parsing mode.*

of the SEM-parser, namely, to create readings as early as possible and to pass these results immediately further to semantics evaluation (actually, parts of the analysis). **SynSem-I** depicts the measurements for the incremental version of the SEM-parser with the full-size grammar G.

The apparent increase in the number of readings in the incremental mode is a bit misleading. Since in absence of information about the length of the utterance in incremental mode, the parser used as sole criterion for a reading that there is an edge from the start to the current point of time which represents a maximal, saturated sign. This means for instance that it will always emit the sentence topic in verb-second sentences as a separate reading. Also, if a sentence ends with a sequence of free adjuncts, the parser will assume as many readings as there are adjuncts. So, a sentence like *I have a meeting on Monday with John* in incremental mode will get the following "pseudo-readings":

I

I have a meeting

I have a meeting on Monday

I have a meeting on Monday with John

In general, there is only a small overhead in incremental mode because we are operating over a chart which allows to reuse parts, already analyzed.

In the context of the INTARC-II system with word lattice parsing, the combination of running the SYN-parser with the syntactic part of the grammar on

the word lattices and running the SEM-parser with the full grammar (**Synsem-I**) proved to be quite efficient. That the SEM-parser has to deal with larger analysis structures in this setup was by far made up for by the fact that it had much less hypotheses to evaluate than the SYN-parser on the word lattice.

6 Conclusions

Linguistic theories like HPSG provide an integrated view on linguistic objects by providing a framework with a uniform formalism for all levels of linguistic analysis. All information is integrated in a single information structure, the SIGN. Though attractive from a theoretical point of view, it raises questions of computational tractability, especially when grammars become larger and more and more information is pulled into that structure.

We subscribe to that integrated view on the level of linguistic descriptions and specifications. On the other hand, from a computational view, we think that for special tasks not all that information is useful or required, at least not all at the same time.

In this paper we described first attempts to make a more flexible use of integrated linguistic descriptions by splitting it up into subpackages that are handled by special processors. We also devised an efficient protocol for communication between such processors. First results have been encouraging. On the other hand, we addressed a number of problems and possible solutions. Only further research can show which one to prefer.

References

Halvorsen, P.-K., and R. M. Kaplan. 1988. Projections and Semantic Description in Lexical-Functional Grammar. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, 1116–1122. Tokyo.

Hauenstein, A., and H. Weber. 1994. An Investigation of Tightly Coupled Time Synchronous Speech Language Interfaces Using a Unification Grammar. Verbmobil-Report 9, Universität Erlangen/Universität Hamburg.

Kay, M., J. M. Gawron, and P. Norvig. 1994. *Verbmobil. A Translation System for Face-to-Face Dialog.* Vol. 33 of CSLI Lecture Notes. Chicago University Press.

Krieger, H.-U., and U. Schäfer. 1994. *TDL*—A Type Description Language for Constraint-Based Grammars. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94, Kyoto, Japan*, 893–899.

Krieger, H.-U., and U. Schäfer. 1995. Efficient Parameterizable Type Expansion for Typed Feature Formalisms. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI-95, Montreal, Canada.* To appear.

Maxwell III, J. T., and R. M. Kaplan. 1991. The Interface between Phrasal and Functional Constraints. In *Proceedings of the Workshop on Constraint Propagation, Linguistic Description, and Computation*, ed. M. Rosner, C. Rupp, and R. Johnson, 105–120. Instituto Dalle Molle IDSIA, Lugano. Also in Computational Linguistics, Vol. 19, No. 4, 571–590, 1993.

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

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

Shieber, S. M. 1985. Using Restriction to Extend Parsing Algorithms for Complex-Feature-Based Formalisms. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics, ACL-85*, 145–152.

Wahlster, W. 1993. Verbmobil: Übersetzung von Verhandlungsdialogen. Verbmobil-Report 1, DFKI, Saarbrücken.

Warren, D. S. 1992. Memoing for Logic Programs. *Communications of the ACM* 35(3):93–111.

Weyers, C. 1996. Präferenzgesteuerte Unifikation von Merkmalsstrukturen. Master's thesis, Universität des Saarlandes, Department of Computer Science. Forthcoming. In German.

Wirén, M. 1992. *Studies in Incremental Natural-Language Analysis.* PhD thesis, Department of Computer and Information Science, Linköping University.