The Complexity of Structure Sharing in Unification-Based Grammars

Marten Trautwein[1]
University of Amsterdam
Department of Mathematics and Computer Science
Plantage Muidergracht 24
1018 TV Amsterdam

Abstract

We apply complexity theory to grammatical theories. This enables us to compare these grammatical theories by an objective measure. In this paper, we concentrate on restricted versions of Lexical Functional Grammar (LFG) and Head-driven Phrase Structure Grammar (HPSG).
We show that the 'fixed grammar' recognition problems of both restricted versions are intractable. In analogy to the intractability result for LFG in (Barton Jr. et al. 1987), we present an *NP*-hard lower bound on the complexity of both recognition problems. In addition, we also present an *NP* upper bound on the complexity. Thus we show that the recognition problems are *NP*-complete.
The presented versions avoid theory specific constructions, because of the imposed restrictions. Thus, we conjecture that the complexity results hold for other unification grammars, e.g., Functional Unification Grammar and PATR-II.
The intractability proofs show that excessive structure sharing and recursive attributes cause the complexity. We conclude that the grammatical theories should force restrictions on these phenomena.

1 Introduction

In this paper we consider the complexity of the recognition problem of grammatical theories. Especially, we consider the recognition problem of fixed grammars. The recognition problem of a grammar is the problem whether a string, w, is a string of the language, $L(G)$, generated by a grammar, G.

[1]The author was supported by the Foundation for language, speech and logic (TSL), which is funded by the Netherlands organization for scientific research (NWO)

There are multiple versions of the question whether a grammar generates a string. In the general case, the question concerns an arbitrary string and an arbitrary grammar. This version of the recognition problem is called the 'universal recognition problem' (URP). A more specific version of this question results when one considers a fixed grammar instead of an arbitrary one. We will call this version the 'fixed recognition problem' (FRP).

The universal recognition problem (URP) is defined as follows.

Definition 10: Given a pair (w, G), where w a string and G a grammar. Question: is the string w in the language generated by G $(w \in L(G))$?

The fixed recognition problem (FRP) is defined as follows.

Definition 11: Let there be a fixed grammar G. Given a string w. Question: is the string w in the language generated by G $(w \in L(G))$?

The universal recognition problem is at least as difficult as the fixed recognition problem. The universal recognition problem may be more difficult. For instance, an algorithm that solves the universal recognition problem efficiently also solves the fixed recognition problem efficiently. On the other hand, an efficient algorithm for the fixed recognition problem may take an amount of time that is exponential in the size of the grammar. Clearly, such an algorithm does not solve the universal recognition problem efficiently.

In this paper, we will present an *NP*-hard lower bound for the fixed recognition problems of restricted fragments of Lexical Functional Grammar (LFG) and Head-driven Phrase Structure Grammar (HPSG). Moreover, we will also present an *NP* upper bound for the universal recognition problems of these fragments of Lexical Functional Grammar and Head-driven Phrase Structure Grammar. As a consequence, both versions of the recognition problem of these restricted fragments of LFG and HPSG are *NP*-complete.

This paper is organized as follows. Section 2 contains the descriptions of the restricted fragments of LFG and HPSG. In Section 3 and Section 4, respectively, we show that the FRPs of LFG and HPSG are *NP*-hard. In Section 5 we show that the URPs of LFG and HPSG are included in *NP*. Finally, Section 6 contains the conclusions that we draw from the *NP*-completeness proofs and results.

2 The Restricted Fragments

Lexical Functional Grammar (LFG) and Head-driven Phrase Structure Grammar (HPSG) are both grammatical theories that use feature structures to encode linguistic information. The versions of LFG and HPSG that we present below have in common that they use primitive descriptions for feature structures. That is, the descriptions will contain conjunctions and structure sharing, but will not contain negations, disjunctions, sets, or more exotic things.

2.1 The Restricted Fragment of LFG

In this section we will describe LFG only briefly. In LFG a mixture of equations and attribute-value matrices is used to describe feature structures. For a more complete description of LFG, we refer to (Kaplan and Bresnan 1982). We can think of an LFG-grammar as a set of combinatory rules plus a lexicon. These combinatory rules are context-free rewrite rules that are annotated with equations. The context-free rewrite part of the combinatory rules describe a conventional constituent structure tree. The equations typically express how the information in a feature structure is passed from a mother nonterminal in the constituent structure tree to a daughter nonterminal.

The following simplified rule serves to illustrate the combinatory rules. The annotation of the rule consists of equations, indicated by $\doteq$, that are linked to some right-hand side nonterminal. The equations contain metavariables in the shape of arrows. The upward arrow refers to the feature structure of the mother nonterminal, the *S*. The downward arrow refers to the feature structure of the nonterminal to which the equation is linked, the *NP*, or *VP*. The equations state that the feature structure of the *S* contains an attribute SUBJ with as value the feature structure of the *NP*, and the feature structure of the *S* is equal to the feature structure of the *VP*.

$$\begin{array}{lcc} S \rightarrow & NP & VP \\ & (\uparrow \textsc{subj}) \doteq \downarrow & \uparrow \doteq \downarrow \end{array}$$

The lexicon can be viewed as an annotated unary context-free rewrite rule, whose right-hand side is a terminal. For example, the lexicon may contain the following entry, which denotes that the terminal 'John' is a third-person singular nounphrase:

$$\text{John} : \ NP \ \ \{(\uparrow \textsc{person}) \doteq \mathit{3rd}, (\uparrow \textsc{number}) \doteq \mathit{singular}\}$$

When we use the attribute-value matrix notation for feature structures, this entry specifies that the feature structure of the nonterminal *NP* is described by

$$\begin{bmatrix} \textsc{person} & \mathit{3rd} \\ \textsc{number} & \mathit{singular} \end{bmatrix}$$

We define the combinatory rules and the lexicon of LFG as follows.

Definition 12: Given a set of attributes, L, and atomic values, A, to construct equations, and a set of nonterminals, N, and terminals, Σ, to construct context-free rewrite rules.

- The set of combinatory rules is a subset of $\bigcup_{i>0} N \times (N, 2^{E(A,L)})^i$.
- The lexicon is a subset of $\Sigma \times N \times 2^{E(A,L)}$.
- The set of equations is defined as $E(A,L) = \{\uparrow p \doteq \uparrow q, \uparrow p \doteq \downarrow q, \downarrow p \doteq \uparrow q, \downarrow p \doteq \downarrow q, \uparrow p \doteq a, \downarrow p \doteq a \mid p, q \in L^*, a \in A\}$

2.2 The Restricted Fragment of HPSG

In this section we will describe HPSG only briefly. In HPSG attribute-value matrices describe feature structures. For a more complete description of HPSG, we refer to (Pollard and Sag 1994). We can think of an HPSG-grammar as a lexicon, a set of principles and a set of rules. In the lexicon the basic feature structures of the grammar are described. The principles state how the information in the feature structures is passed on in a constituent structure. The rules state which constituent structures can be formed.

We assume that the entries in lexicon are attribute-value matrices that contain the following four attributes

- the attribute PHON, whose value is a string;
- the attribute CONTENT, which denotes the semantic feature information;
- the attribute HEAD, which denotes the syntactic category of this feature structure;
- the attribute SUBCAT, whose value is a list of partial descriptions of feature structures that this feature structure still has to combine with.

The three principles that we consider in this paper are the 'Head Feature Principle' (HFP), the 'Subcategorization Principle' (SP), and the 'Semantics Principle' (SemP). The Head Feature Principle states that in a constituent the mother and the head-daughter share the same syntactic category, which is specified by the attribute HEAD. The Semantics Principle states that in a constituent which contains an adjunct-daughter, the mother and the adjunct-daughter share the same semantic feature information, which is specified by the attribute CONTENT; in the other constituents the mother and the head-daughter share the same semantic feature information. The Subcategorization Principle states that a constituent subcategorizes for the list of constituents that its head-daughter subcategorizes for, minus the constituent's complement-daughters.

The rules of the grammar are specified by Immediate Dominance schemas (ID-schemas). In this paper, we consider the 'head-subject' ID-schema, the 'head-complement' ID-schema, and 'head-adjunct' ID-schema. The head-subject ID-schema states that a constituent consists of a mother with an empty subcategorization-list, a head-daughter and one complement-daughter. The head-complement ID-schema states that a constituent consists of a mother with a subcategorization-list that contains one element, one head-daughter, and at least one complement-daughter. The head-adjunct ID-schema states that a constituent consists of an adjunct-daughter that modifies the head-daughter in such a way that the adjunct-daughter's attribute HEAD contains an attribute MOD that is shared with the value of the head-daughter.

When we combine these principles and rules, we obtain the partial constituent structure trees that are given in Figure 1 through Figure 3.

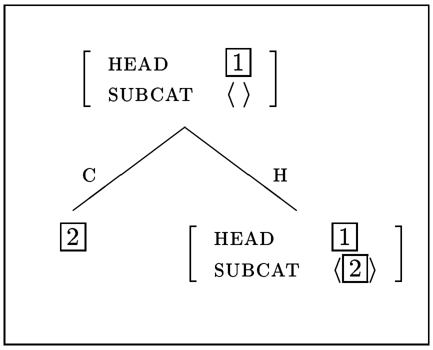

Figure 1: Combination of HFP, SP, SemP, and head-subject ID-schema

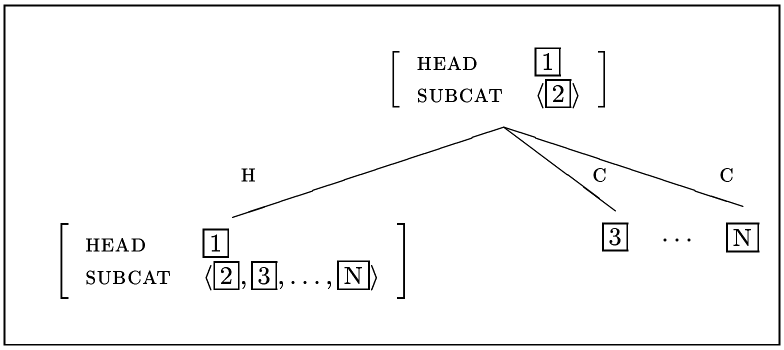

Figure 2: Combination of HFP, SP, SemP, and head-complement ID-schema

3 A Lower Bound for LFG

In this section we will show that the recognition problems of LFG are *NP*-hard. First we will repeat the *NP*-hardness proof for the universal recognition problem of LFG, given in (Barton Jr. et al. 1987). Then, we will show that the recognition problem of the LFG-grammar G_F is *NP*-hard by a reduction from the URP of LFG-grammars.

3.1 The Universal Recognition Problem

In (Barton Jr. et al. 1987) a polynomial time, many-one reduction from the known *NP*-complete problem 3-SATISFIABILITY proves that the URP of LFG is *NP*-hard. Such a reduction from some problem A to some problem B maps instances of problem A onto instances of problem B. The polynomial time, many-one reductions are subject to two conditions: (1) the reductions are easy to compute, and (2) the reductions preserve the answers. A reduction from A to B is easy to compute, if the mapping takes an amount of steps that is polynomial in the size of A, i.e., takes polynomial time. A reduction preserves answers if the answer to the instance of A is the same as the answer to the

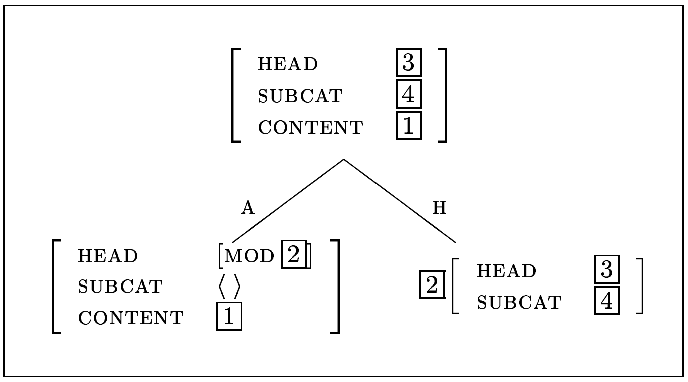

Figure 3: Combination of HFP, SP, SemP, and head-adjunct ID-schema

instance of B. That is, the answer to the instance of A is 'Yes' if, and only if, the answer to the instance of B is also 'Yes'.

A reduction is an elegant way to classify a problem as *NP*-hard. Suppose problem B is a problem with unknown complexity. Let there be a reduction f from an *NP*-hard problem A to problem B. Furthermore, let f conform to the two conditions above. By an indirect proof, it follows from this reduction that B is at least as hard as A. Hence B is also an *NP*-hard problem. If we also prove that we can guess a solution for B and check that guessed solution in polynomial time, then B is an *NP*-complete problem.

The well-known *NP*-complete problem 3-SATISFIABILITY (3SAT) is defined as follows.

Definition 13: 3-SATISFIABILITY
INSTANCE: A formula ϕ, from propositional logic, in 3-conjunctive normalform.
QUESTION: Is there an assignment of truth-values to the propositional variables of ϕ, such that ϕ is true?

The instances of 3SAT are formulas in 3-conjunctive normalform, i.e., the formulas are conjunctions of clauses. The clauses are disjunctions of three literals, and the literals are positive and negative occurrences of propositional variables. Without lose of generality we may assume that the variables are of the form p_i, where the index i is a natural number in binary notation. We call formula ϕ a satisfiable formula iff an assignment exists that makes formula ϕ true.

An assignment assigns either the value true or the value false to each propositional variable. Given such an assignment, we can determine the truth-value of a formula. The formula $\phi = (\gamma_1 \wedge \ldots \wedge \gamma_m)$ is true iff each clause, γ_i, is true. A clause $\gamma = (l_1 \vee l_2 \vee l_3)$ is true iff at least one literal, l_i, is true. A positive (negative) literal, $l_i = p_j$ ($l_i = \overline{p_j}$), is true iff the variable p_j is assigned the value true (false).

As an example consider the formula $\phi = (p_1 \vee \overline{p_{10}} \vee p_{11}) \wedge (\overline{p_1} \vee p_{10} \vee \overline{p_{11}})$. The assignment of the value true to variable p_{10} and false to variable $\overline{p_{11}}$ makes this formula true.

The reduction from 3SAT to the URP of LFG presented in (Barton Jr. et al. 1987) maps formulas onto a string and a grammar in the following way. Given a formula ϕ in 3CNF, with the n propositional variables $p_1 \ldots p_n$ and m clauses. The reduction maps ϕ onto the string w_ϕ, by omitting the parentheses, disjunction symbols, and conjunction symbols. For instance, the formula ϕ given above, would be mapped onto the string $p_1\,\overline{p_{10}}\,p_{11}\,\overline{p_1}\,p_{10}\,\overline{p_{11}}$. The reduction maps ϕ onto a grammar G_ϕ with eight combinatory rules and $4n$ entries in the lexicon (see Tables 1 and 2).

$S \rightarrow \underset{\uparrow \doteq \downarrow}{S} \quad \underset{\uparrow \doteq \downarrow}{S}$	$S \rightarrow \underset{\uparrow \doteq \downarrow}{T} \quad \underset{\uparrow \doteq \downarrow}{F} \quad \underset{\uparrow \doteq \downarrow}{F}$
$S \rightarrow \underset{\uparrow \doteq \downarrow}{T} \quad \underset{\uparrow \doteq \downarrow}{T} \quad \underset{\uparrow \doteq \downarrow}{T}$	$S \rightarrow \underset{\uparrow \doteq \downarrow}{F} \quad \underset{\uparrow \doteq \downarrow}{T} \quad \underset{\uparrow \doteq \downarrow}{T}$
$S \rightarrow \underset{\uparrow \doteq \downarrow}{T} \quad \underset{\uparrow \doteq \downarrow}{T} \quad \underset{\uparrow \doteq \downarrow}{F}$	$S \rightarrow \underset{\uparrow \doteq \downarrow}{F} \quad \underset{\uparrow \doteq \downarrow}{T} \quad \underset{\uparrow \doteq \downarrow}{F}$
$S \rightarrow \underset{\uparrow \doteq \downarrow}{T} \quad \underset{\uparrow \doteq \downarrow}{F} \quad \underset{\uparrow \doteq \downarrow}{T}$	$S \rightarrow \underset{\uparrow \doteq \downarrow}{F} \quad \underset{\uparrow \doteq \downarrow}{F} \quad \underset{\uparrow \doteq \downarrow}{T}$

Table 1: Combinatory rules of G_ϕ

$p_i\ : T$	$p_i\ : F$	$\overline{p_i}\ : T$	$\overline{p_i}\ : F$
$\uparrow i \doteq +$	$\uparrow i \doteq -$	$\uparrow i \doteq -$	$\uparrow i \doteq +$

Table 2: Lexicon entries of G_ϕ $(1 \leq i \leq n)$

It is not difficult to show that the reduction is computable in polynomial time and preserves the answers, i.e., maps all and only all satisfiable formulas onto pairs (w_ϕ, G_ϕ) such that G_ϕ generates w_ϕ. We will call the LFG-grammars resulting from the reduction BBR-grammars.

Lemma 3.1 (Barton Jr. et al. 1987) *The universal recognition problem of LFG is NP-hard.*

Proof. See (Barton Jr. et al. 1987, Chapter 4). □

3.2 The Fixed Recognition Problem

Now, we will prove the *NP*-hardness of the FRP of LFG by a reduction from the URP of BBR-grammars. The reduction will map a string w_ϕ and a grammar

$T \rightarrow$	A	T	$T \rightarrow$	B	T
		$(\uparrow 1) \doteq \downarrow$			$(\uparrow 0) \doteq \downarrow$
$F \rightarrow$	A	F	$F \rightarrow$	B	F
		$(\uparrow 1) \doteq \downarrow$			$(\uparrow 0) \doteq \downarrow$

Table 3: The four extra rules of G_F

$p : T$	$p : F$	$\overline{p} : T$	$\overline{p} : F$	$0 : A$	$1 : B$
$\uparrow v \doteq +$	$\uparrow v \doteq -$	$\uparrow v \doteq -$	$\uparrow v \doteq +$		

Table 4: The six lexicon entries of G_F

G_ϕ onto a string w'_ϕ such that the LFG-grammar G_F generates w'_ϕ iff the BBR-grammar G_ϕ generates w_ϕ. The LFG-grammar G_F contains the combinatory rules of the BBR-grammars plus the four extra rules that are given in Table 3. The lexicon of G_F consists of the six entries given in Table 4.

Now the reduction from (w_ϕ, G_ϕ) to w'_ϕ is computed as follows.

1. Check that w_ϕ is of the form $(p_i \cup \overline{p_i})^{3m}$, where i, m are natural numbers, i in binary notation.
2. Check that G_ϕ is a BBR-grammar.
3. Check that the lexicon of G_ϕ contains an entry for all words p_i and $\overline{p_i}$ in the string w_ϕ.
4. If any of the checks fail, the reduction outputs the empty string, which is not generated by G_F.
5. Otherwise, the reduction outputs the string w'_ϕ, which is obtained from w_ϕ by substituting each word p_i by the string $i\,p$ and each word $\overline{p_i}$ by the string $i\,\overline{p}$.

Clearly, the reduction is computable in polynomial time. In order to prove that the reduction preserves the answers we have to show that nonterminal T with the feature structure described by the attribute-value matrix $[i\,+]$ derives word p_i in G_ϕ in one step iff $i = i_1 \dots i_{\log(n)}$ and T with $[i_1\,[\dots[i_{\log(n)}[v\,+]]\dots]]$ derives string $i\,p$ in G_F in $\log(n)$ steps, and also for word $\overline{p_i}$ and the same for nonterminal F.

Lemma 3.2 *The fixed recognition problem of LFG is NP-hard.*

Proof. There is a polynomial time many-one reduction from the URP of BBR-grammars to the FRP of the LFG-grammar given in the Tables 1, 3 and 4. Details are left to the reader. □

4 A Lower Bound for HPSG

In this section we will show that the recognition problems of HPSG are *NP*-hard. First we will show that the universal recognition problem of HPSG is *NP*-hard, by analogy with the *NP*-hardness proof for LFG. Then, we will show that the recognition problem of the HPSG-grammar H_F is *NP*-hard by a reduction from the URP of HPSG-grammars.

4.1 The Universal Recognition Problem

The following reduction from 3SAT to the URP of HPSG maps formulas onto a string and a grammar. We will leave it to the reader to show that the reduction is computable in polynomial time and preserves answers. Given a formula ϕ in 3CNF, with the n propositional variables $p_1 \dots p_n$ and m clauses. The reduction maps ϕ onto the string v_ϕ, by omitting the parentheses, disjunction symbols, and conjunction symbols, and suffixing it with a \$. The reduction maps ϕ onto a grammar H_ϕ with the Head Feature, Subcategorization and Semantics Principle, the head-subject and head-complement ID-schemas, and the $4n+1$ entries in the lexicon given in Table 5. See Figures 1 and 2, with $N = 4$, for the combined action of the principles and rules.

$$\begin{bmatrix} \text{PHON} & p_i \\ \text{HEAD} & TF \\ \text{SUBCAT} & \langle\,\rangle \end{bmatrix} \;\Bigg|\; \begin{bmatrix} \text{PHON} & \overline{p_i} \\ \text{HEAD} & TF \\ \text{SUBCAT} & \langle\,\rangle \end{bmatrix} \;\Bigg|\; \begin{bmatrix} \text{PHON} & \$ \\ \text{HEAD} & S \\ \text{SUBCAT} & \langle\,\rangle \end{bmatrix}$$

$$\begin{bmatrix} \text{PHON} & p_i \\ \text{HEAD} & S \\ \text{SUBCAT} & \langle S : \boxed{1}, TF, TF \rangle \\ \text{CONTENT} & \boxed{1}\,[i\,+] \end{bmatrix} \;\Bigg|\; \begin{bmatrix} \text{PHON} & \overline{p_i} \\ \text{HEAD} & S \\ \text{SUBCAT} & \langle S : \boxed{1}, TF, TF \rangle \\ \text{CONTENT} & \boxed{1}\,[i\,-] \end{bmatrix}$$

Table 5: Lexicon entries of H_ϕ $(1 \leq i \leq n)$

As an example consider the formula $\phi = (p_1 \vee \overline{p_{10}} \vee p_{11}) \wedge (\overline{p_1} \vee p_{10} \vee \overline{p_{11}})$. The reduction will map this formula onto to the string $v_\phi = p_1\,\overline{p_{10}}\,p_{11}\,\overline{p_1}\,p_{10}\,\overline{p_{11}}\,\$$. A possible derivation for the substring $p_1\,\overline{p_{10}}\,p_{11}$ is given in Figure 4. The example shows that this reduction and the reduction provided by (Barton Jr. et al. 1987) differ slightly. In this reduction only the 'truth-value' of the head-daughter is passed on to the mother, whereas in the former reduction the 'truth-values' of all daughters are passed on to the mother. This difference, however, is not essential, because it does not affect the satisfiability of the encode 3CNF formula.

We leave it to the reader to show that the reduction is computable in polynomial time and preserves the answers, i.e., maps all and only all satisfiable formulas onto pairs (v_ϕ, H_ϕ) such that H_ϕ generates v_ϕ. We will call the HPSG-grammars resulting from the reduction 3SAT-HPSG-grammars.

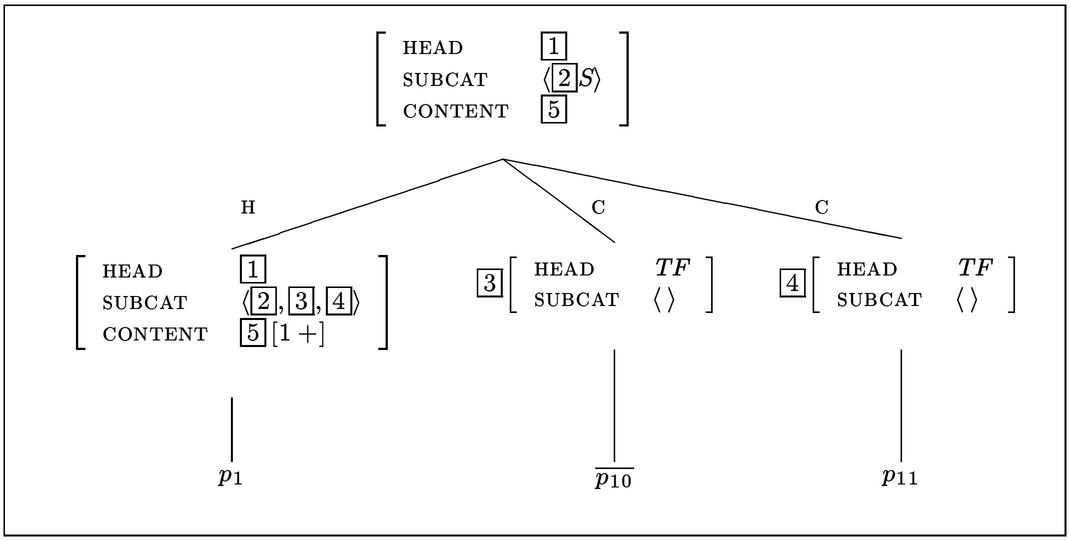

Figure 4: A possible derivation for the string $p_1\,\overline{p_{10}}\,p_{11}$

Lemma 4.1 *The universal recognition problem of HPSG is NP-hard.*

Proof. Left to the reader. □

4.2 The Fixed Recognition Problem

Now, we will prove the *NP*-hardness of the FRP of HPSG by a reduction from the URP of 3SAT-HPSG-grammars. This proof is similar to the proof given for the *NP*-hardness of the FRP of LFG. The reduction will map a string v_ϕ and a grammar H_ϕ onto a string v'_ϕ such that the LFG-grammar H_F generates v'_ϕ iff the 3SAT-HPSG-grammar H_ϕ generates v_ϕ. The HPSG-grammar H_F contains the three principles and two rules of the 3SAT-HPSG-grammars plus one extra rule: the head-adjunct ID-schema. See Figure 3 for the combined action of the principles and head-adjunct ID-schema. The lexicon of H_F consists of ten entries, which are given in Table 6.

Now a similar reduction from (v_ϕ, H_ϕ) to v'_ϕ is taken, as the reduction from the URP of BBR-grammar to the FRP of LFG-grammars. The only difference is that the string v'_ϕ is prefixed with an extra \$-marker. We leave it to the reader to show that the reduction is computable in polynomial time and preserves the answers. The computation steps in the reduction are as follows.

1. Check that v_ϕ is of the form $(p_i \cup \overline{p_i})^{3m}$, where i is a binary number, and m is some natural number.

2. Check that H_ϕ is a 3SAT-HPSG-grammar.

$$\begin{bmatrix} \textsc{phon} & p \\ \textsc{head} & S \\ \textsc{subcat} & \left\langle S : \begin{bmatrix} \textsc{v} & \boxed{1} \\ \textsc{w} & \boxed{1} \end{bmatrix}, TF, TF \right\rangle \\ \textsc{content} & \begin{bmatrix} \textsc{w} & \boxed{1} \\ \textsc{v} & + \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} \textsc{phon} & \overline{p} \\ \textsc{head} & S \\ \textsc{subcat} & \left\langle S : \begin{bmatrix} \textsc{v} & \boxed{1} \\ \textsc{w} & \boxed{1} \end{bmatrix}, TF, TF \right\rangle \\ \textsc{content} & \begin{bmatrix} \textsc{w} & \boxed{1} \\ \textsc{v} & - \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \textsc{phon} & p \\ \textsc{head} & TF \\ \textsc{subcat} & \langle\rangle \end{bmatrix} \quad \begin{bmatrix} \textsc{phon} & \overline{p} \\ \textsc{head} & TF \\ \textsc{subcat} & \langle\rangle \end{bmatrix}$$

$$\begin{bmatrix} \textsc{phon} & 0 \\ \textsc{head}\vert\textsc{mod} & \begin{bmatrix} \textsc{head} & S \\ \textsc{subcat} & \langle S, TF, TF \rangle \\ \textsc{content} & \begin{bmatrix} \textsc{w} & \boxed{1} \\ \textsc{v} & \boxed{2} \end{bmatrix} \end{bmatrix} \\ \textsc{subcat} & \langle\rangle \\ \textsc{content} & \begin{bmatrix} \textsc{w} & \boxed{1} \\ \textsc{v} & [0\,\boxed{2}] \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} \textsc{phon} & 1 \\ \textsc{head}\vert\textsc{mod} & \begin{bmatrix} \textsc{head} & S \\ \textsc{subcat} & \langle S, TF, TF \rangle \\ \textsc{content} & \begin{bmatrix} \textsc{w} & \boxed{1} \\ \textsc{v} & \boxed{2} \end{bmatrix} \end{bmatrix} \\ \textsc{subcat} & \langle\rangle \\ \textsc{content} & \begin{bmatrix} \textsc{w} & \boxed{1} \\ \textsc{v} & [1\,\boxed{2}] \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \textsc{phon} & 0 \\ \textsc{head}\vert\textsc{mod} & \begin{bmatrix} \textsc{head} & TF \\ \textsc{subcat} & \langle\rangle \\ \textsc{content} & \begin{bmatrix} \textsc{v} & \boxed{1} \end{bmatrix} \end{bmatrix} \\ \textsc{subcat} & \langle\rangle \\ \textsc{content} & \begin{bmatrix} \textsc{v} & [0\,\boxed{1}] \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} \textsc{phon} & 1 \\ \textsc{head}\vert\textsc{mod} & \begin{bmatrix} \textsc{head} & TF \\ \textsc{subcat} & \langle\rangle \\ \textsc{content} & \begin{bmatrix} \textsc{v} & \boxed{1} \end{bmatrix} \end{bmatrix} \\ \textsc{subcat} & \langle\rangle \\ \textsc{content} & \begin{bmatrix} \textsc{v} & [1\,\boxed{1}] \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \textsc{phon} & \$ \\ \textsc{head} & S \\ \textsc{subcat} & \left\langle S : \begin{bmatrix} \textsc{v} & \boxed{1} \\ \textsc{w} & \boxed{1} \end{bmatrix} \right\rangle \\ \textsc{content} & \begin{bmatrix} \textsc{w} & \boxed{1} \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} \textsc{phon} & \$ \\ \textsc{head} & S \\ \textsc{subcat} & \langle\rangle \end{bmatrix}$$

Table 6: The ten lexicon entries of H_F

3. Check that the lexicon of H_ϕ contains an entry for all words p_i and $\overline{p_i}$ in the string v_ϕ.
4. If any of the checks fail, the reduction outputs the empty string, which is not generated by H_F.
5. Otherwise, the reduction outputs the string v'_ϕ, which is obtained from v_ϕ by substituting each word p_i by the string $i\,p$, substituting $\overline{p_i}$ by $i\,\overline{p}$, and prefixing v'_ϕ with a \$.

As an example consider the string $v_\phi = p_1\,\overline{p_{10}}\,p_{11}\,\overline{p_1}\,p_{10}\,\overline{p_{11}}\,\$$. The reduction will map this string to the string $v'_\phi = \$\,1p\,10\overline{p}\,11p\,1\overline{p}\,10p\,11\overline{p}\,\$$.

Lemma 4.2 *The fixed recognition problem of HPSG is NP-hard.*

Proof. There is a polynomial time many-one reduction from the URP of 3SAT-HPSG-grammars to the FRP of the HPSG-grammar H_F, whose lexicon is given in the Table 6, and principles and rules are represented in the Figures 1, 2 and 3. Further details are left to the reader. □

5 An Upper Bound for LFG and HPSG

In the preceding sections we have shown that the recognition problems of restricted fragments of LFG and HPSG are *NP*-hard. The liberal use of structure sharing and recursive attributes seems to have caused the complexity. The *NP*-hardness results prove that the recognition problems are at least as hard an any problem in the complexity class *NP*. That is, less than '*NP*-power' is insufficient to solve the recognition problems. A natural question that arises is: 'Is *NP*-power sufficient to solve the recognition problems'? Stated differently, can these recognition problems be harder than *NP*; for instance, if the grammars contain more structure sharing, or a more excessive use of recursion, than the grammars presented thus far? The answer to this question is 'No'. The recognition problems of the restricted fragments of LFG and HPSG are included in *NP*.

In this section we will show that the universal recognition problems of the restricted fragments of LFG and HPSG are in *NP*. We will prove this upper bound by providing an *NP*-algorithm that solves the universal recognition problem. That is, we provide an algorithm that guesses a solution for this problem, and then checks the guessed solution. This check, however, may not take more than polynomial time.

In the *NP*-algorithm that we will provide for the universal recognition problem, we take the 'off-line' view on derivations. In general there are two views on derivations in unification-based grammatical theories. One is sometimes called 'off-line', the other 'on-line', c.f. (Pereira and Warren 1983). In the off-line view, a derivation consists of two phases. In the first phase, a total constituent structure tree is constructed. Each node in the constituent structure tree is provided with its own partial description of a feature structure. That is, no unification has been computed yet. Only in the second phase, the descriptions at the nodes are combined, and the complete descriptions of the feature structures are computed. Contrast this off-line view with the on-line view, which consist of only one phase. In the on-line view the descriptions of feature structures and the constituent structure tree are constructed at the same time.

The first phase of the off-line view on derivations shows in the second step of the next *NP*-algorithm for the universal recognition problem. The second phase shows in the third step. In the third step, a check for consistency is sufficient, because we are interested *whether* the grammar generates the string, not *how* the grammar generates the string. Given a string w and LFG- or HPSG-grammar G, the *NP*-algorithm consists of the following three steps.

- Guess a derivation for w.
- Check that the steps in the derivation conform to the combinatory rules or ID-schemas of the grammar G.
- Check that the descriptions of feature structures are consistent.

Now, we have to show that both checks take an amount of steps that is polynomial in the size of the string w and the size of the grammar G.

Because the LFG- and HPSG-grammars are restricted, the derivations for a string take an amount of steps that is at most polynomial in the length of the string. To be more precise, the LFG-derivations have length $O(|w|^2|G|)$, because LFG-derivations conform to the off-line parsability constraint. The HPSG-derivations have length $O(|w|)$, because the ID-schemas require that after each step two words are combined. So the derivations are short and the first check can be performed in polynomial time.

Not only are the derivations short, but also only a small amount of feature information is involved in each derivation step. Each derivation step add at most a polynomial number of partial descriptions for feature structures. More precise, the number of partial descriptions added at each step is linear in the size of the grammar, $O(|G|)$. Hence the total number of partial descriptions in a derivation is linear in the size of the derivation and the grammar; therefore, polynomial in the size of the grammar and the string. Because we consider restricted fragments, the feature information only contains conjunctions and structure sharing. In (Smolka 1992) a quadratic time algorithm is presented that checks the consistency of a collection of partial descriptions for feature structures. So the second check of the *NP*-algorithm can also be performed in polynomial time, because the derivation contains a polynomial number of partial descriptions.

From the above and the previous sections we conclude that the following holds.

Theorem 5.1 *The recognition problems of LFG and HPSG are NP-complete.*

6 Conclusions

We have shown that the recognition problems of restricted fragments of LFG and HPSG are *NP*-complete. Moreover the *NP*-completeness does not depend on theory specific constructions. Therefore, we conjecture that the complexity results hold for other unification grammars, e.g., Functional Unification Grammar and PATR-II. For a description of Functional Unification Grammar and PATR-II, we refer to (Kay 1985) and (Shieber 1986).

To conclude this paper, we will consider what the complexity results further contribute to computation linguistics. The way in which humans process natural language suggests that natural language processing is a deterministic polynomial time computable problem. Now if natural language processing is a deterministic polynomial time computable problem, then probably natural language utterances contain enough information to enable efficient processing. The high complexity of the recognition problems of the fragments presented in this paper therefore seems to indicate that insufficient information is available in these fragments. So either the fragments are not properly extracted from the grammatical theories, or the grammatical theories lack some essential information.

The intractability proofs also seem to show where this information is missing. One point is the liberal use of structure sharing, for instance, as represented by the following rule and substructure from Table 1 and Table 6, respectively,

$$S \to \underset{\uparrow \doteq \downarrow}{S} \quad \underset{\uparrow \doteq \downarrow}{S} \qquad \text{and} \qquad \begin{bmatrix} \text{SUBCAT} & \langle S: \begin{bmatrix} \text{V} & \boxed{1} \\ \text{W} & \boxed{1} \end{bmatrix}, \ldots \rangle \\ \text{CONTENT} & \begin{bmatrix} \text{W} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

However, argumentation is given that case marking in German requires the kind of structure sharing that we presented, c.f., (Nerbonne et al. 1994). Another point is the recursive use of attributes, for instance, as represented by the following rule and substructure from Table 3 and Table 6, respectively,

$$T \to 1 \quad \underset{(\uparrow 1)\ \doteq \downarrow}{T} \qquad \text{and} \qquad \begin{bmatrix} \ldots\text{CONTENT} & \begin{bmatrix} \text{V} & \boxed{1} \end{bmatrix} \\ \text{CONTENT} & \begin{bmatrix} \text{V} & [1\ \boxed{1}] \end{bmatrix} \end{bmatrix}$$

From a formal point of view, the fragments that we presented in this paper can hardly be restricted more. Nevertheless, the high complexity of these fragments seems to indicate that the fragments do not describe natural language grammars. So probably there is a specific linguistic reason that explains why the above constructions do not appear in natural language. In our opinion computational linguists should be concerned about finding this reason.

References

Barton Jr., G., Berwick, R., and Ristad, E. (1987). *Computational Complexity and Natural Language.* Cambridge, MA: MIT Press.

Kaplan, R., and Bresnan, J. (1982). Lexical-functional grammar: a formal system for grammatical representation. In Bresnan, J., editor, *The Mental Representation of Grammatical Relations*, 173–281. Cambridge, MA: MIT Press.

Kay, M. (1985). Parsing in functional unification grammar. In Dowty, D., Karttunen, L., and Zwicky, A., editors, *Natural Language Parsing*, 251–278. New York, NY: Cambridge University Press.

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

Pereira, F., and Warren, D. (1983). Parsing as deduction. In *Proceedings 21th Annual Meeting of ACL*, 137–144.

Pollard, C., and Sag, I. (1994). *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. Stanford & Chicago: CSLI & University of Chicago Press.

Shieber, S. (1986). *An Introduction to Unification-Based Approaches to Grammar*, vol. 4 of *CSLI Lecture Notes*. Stanford: CSLI.

Smolka, G. (1992). Feature-constraint logics for unification grammars. *Journal of Logic Programming*, 12(1):51–87.