Unification-based grammars and complexity classes

Anders Søgaard
Center for Language Technology
Njalsgade 80
DK-2300 Copenhagen
anders@cst.dk

Abstract

A simple unification-based formalism is identified, and it is demonstrated how small changes effect its computational complexity. Versions are defined for polynomial time (PTIME), non-deterministic polynomial time (NPTIME), polynomial space (PSPACE), exponential time (EXPTIME), and undecidability (UNDEC). If the grammar is unambiguous or deterministically copying, its recognition problem is solvable in PTIME; the same goes for the polysized and k-ambiguous fragment; if the grammar has no unary cycles, it is solvable in NPTIME; in the abscence of *true* path equations and global implication, if functional uncertainty is impossible, and no weak precedence is employed (no quantification over strings), the grammar is solvable in PSPACE; if true path equations are not employed, a positive result can be obtained for EXPTIME; and finally, if the grammar is unrestricted and thus subsumed by Kasper-Rounds logic with global implication, it is undecidable. In fact, when functional uncertainty is also employed, this is high undecidability.

1 Introduction

In Figure 1 below some known complexity results for unification-based formalisms are listed. Of course the list is not exhaustive, but the references there are prominent, widely cited, and relevant for our purposes. This literature, in other words, is the context in which the present paper is written. In this introductory section, some of these results are explained, and our investigations are motivated.[1]

Three grammar formalisms are mentioned in the above: categorial unification grammar (CUG), lexical-functional grammar (LFG) and head-driven phrase structure grammar (HPSG). If no grammar formalism is mentioned, the result is obtained on an abstract formalism, which is somehow meant to denote the intersection of a number of unification-based formalisms, much like the one defined in the next section. The formalisms here differ in a number of respects, some of which concern operators and specific properties, but one difference deserves special notice, since it is absolutely fundamental: the definition of the extension of a grammar. The difference between LFG and CUG, in this respect, is minor, but they both differ radically from HPSG. In recent literature, the two perspectives on the extensions of grammars have been labelled, respectively, the *proof-theoretic* and the *model-theoretic* perspective. The difference is spelled out in Definition 1.1.

The results obtained in (Blackburn and Spaan, 1993) are important for our investigations. The results were obtained by reductions to modal languages of varying complexity that are defined on deterministic frames.[2] The NP-completeness result was obtained when the modal language contained no global quantification.[3] The modal language that was employed here was adopted from (Kasper and Rounds, 1986). The second result was obtained by adding global quantification to a modal logic with nominals, i.e propositions that denote singleton sets. Nominals are weaker than path equations (out of context), but serve the same purpose, namely to implement reentrancies. The alternative is to have true path equations, as in Kasper-Rounds logic (Kasper and Rounds, 1986). In the context of global quantification, this leads to undecidability, the third result of (Blackburn and Spaan, 1993). The results of (Blackburn and Spaan, 1993) refer to an abstract unification-based formalism, and nothing is said about parsing. The results are of relevance both under the proof-theoretic and the model-theoretic perspective. However, since nothing is said about parsing, the results do not address the universal recognition problem. Consequently, similarly to (Johnson, 1991), the abstract formalism discussed in (Blackburn and Spaan, 1993) is not a stand-alone formalism; rather it has to be integrated with a proof-theoretic or model-theoretic backbone that drives the actual parsing.

In Johnson's earlier work (Johnson, 1988), he adopts a proof-theoretic perspective in the investigation of the

[1] **A note on terminology:** A problem is *complete* for a complexity class, if it is in it and, at the same, as hard as any other problem in that class. For instance, a problem which is in NPTIME and hard for this class (NP-hard), is said to be NP-complete.

[2] In modal logic, this amounts to adding $\Diamond\phi \rightarrow \Box\phi$ to the axioms. The axiom says that $\forall x, z\ R(x, z) \wedge \exists y\ R(x, y) \wedge Q(y) \rightarrow Q(z)$. The deterministic restriction only effects complexity in the context of the NP-completeness result. The other results can also be obtained on arbitrary frames.

[3] Global quantification effects the expressivity. Invariance under disjoint unions, for instance, is lost when global quantification is imported.

Author(s)	**Formalism**	**Result**	**Comments**
(Blackburn and Spaan, 1993)		NP-complete	All features deterministic, and no global quantification.
(Blackburn and Spaan, 1993)		EXPTIME	No path equations.
(Blackburn and Spaan, 1993)		UNDEC	
(Johnson, 1988)	LFG	UNDEC	Proof-theoretic perspective.
(Johnson, 1988)	LFG	NP-complete	Off-line parsable.
(Johnson, 1991)	LFG	NP-complete	Not stand-alone.
(Kasper and Rounds, 1986)		NP-complete	Same as (Blackburn and Spaan, 1993).
(Kepser and Mönnich, 2003)	HPSG	UNDEC	Indirect result.
(Seki et al., 1993)	LFG	PTIME	Deterministically copying.
(Trautwein, 1995)	HPSG/LFG	NP-complete	Various restrictions.
(Trautwein, 1995)	CUG	NP-complete	

Figure 1: Some known complexity results

computational complexity of LFG. He establishes an undecidability result for the full formalism, in the abscence of functional uncertainty, but he also shows that under an off-line parsability restriction, the recognition problem can be solved in NPTIME. (Kepser and Mönnich, 2003) considers HPSG. Their undecidability result is rather indirect; in fact, what they prove is that there is no reduction of HPSG to monadic second order logic which is decidable. Other reduction arguments in the literature are of similar nature. For instance, (Søgaard, 2006a) proves that even simple versions of HPSG are undefinable in decidable prefix-vocabulary classes of first order logic. In this paper, general results are sought that are neutral to the logical design choices, so it is important not to confuse these two kinds of results.

The PTIME result in (Seki et al., 1993) is obtained under a proof-theoretic perspective too by syntactic restrictions on the productions of LFG. The productions are first restricted to be either of the form $(\uparrow \texttt{attr}) = \texttt{val}$ (an immediate value schema) or $(\uparrow \texttt{attr}) =\downarrow$ (a structure synthesizing schema). It is then said that for each pair of rules $r_1 : A \rightarrow \alpha_1$ and $r_2 : A \rightarrow \alpha_2$ whose left-hand sides are the same, is inconsistent in the sense that there exists no f-structure that locally satisfies both of the functional schemata of r_1 and r_2. Other polynomial fragments have been identified in the literature. For instance, (Søgaard and Haugereid, 2006) show that common versions of HPSG are solvable in PTIME, if the lexicon is rigid, and no ambiguous derivations exist. The result transfers to LFG. Finally, (Trautwein, 1995) obtains a number of NP-completeness results for various unification-based formalisms. Some of these rely on some rather amputated versions of the formalisms. The results are mentioned here for comparison; in particular, our NPTIME fragment is more expressive than the fragments presented in (Trautwein, 1995). See below for details. The result on CUG is mentioned separately, since this result is fully valid, and of particular interest to us.

The definition of the extension of grammars under the proof-theoretic and model-theoretic perspectives was promised you.

Definition 1.1 (Grammar extensions)**.** The extension of a grammar $\mathcal{G}$ is denoted by $\mathcal{L}(\mathcal{G})$. Consider the definition of $\mathcal{L}(\mathcal{G})_p$ and $\mathcal{L}(\mathcal{G})_m$.

$$\begin{array}{rcl} \mathcal{L}(\mathcal{G})_p & = & \{x \in \mathcal{V}^* | \exists M \in \mathcal{M}\ \texttt{start}' \sqsubseteq M \wedge M{:}x\} \\ \mathcal{L}(\mathcal{G})_m & = & \{x \in \mathcal{V}^* | \exists M \in \mathcal{M}\ M, w \models (\mathsf{Axioms} \wedge x')\} \end{array}$$

$\mathcal{L}(\mathcal{G})_p$ is the extension of a language on the proof-theoretic perspective, i.e the extension of a grammar is the set of permutations over the vocabulary that can be derived from relational (feature) structures that are subsumed by the structure associated with the start node. Structures are derived by (extensions of) standard algorithms. The model-theoretic perspective, represented by $\mathcal{L}(\mathcal{G})_m$, is different. The extension of a model-theoretic grammar is the set of permuations over the vocabulary whose descriptions (x' is the logical description of the linear string x) are satisfiable in conjunction with the axioms of the grammar.[4] The axioms correspond roughly to derivation rules, but the relation from one perspective to the other is a bit more complex than that.

In the sections to come, we often adopt an entirely model-theoretic view on grammars. Several advantages of this view can be listed: (i) It is fully declarative; (ii) it integrates nicely with automated reasoning techniques; (iii) it opens the door for an open-ended lexicon and a more flexible notion of grammaticality, and, finally, (iv) it is agnostic on certain philosophical questions, incl. the cardinality of natural languages. See (Pullum and Scholz, 2005) for details. Since we seek general results, an abstract formalism is identified which to some extent captures a common core of the majority of unification-based formalisms. Model-theoretic algorithms have a natural lower complexity bound in NPTIME, however, since model-theoretic parsing is a search problem, so for our PTIME results we return to a proof-theoretic or hybrid perspective.

[4] In fact, some researchers have proposed another and perhaps more radical model-theoretic view, on which the language is the set of strings that satisfy the axioms of the grammar directly. This perspective has no obvious applications for unification-based formalisms.

What is extremely important here is that we are interested in the universal recognition problem, not just the satisfiability of some logic that happens to be suitable to describe linguistic structures. Consequently, we will have to provide all the information necessary for the grammar to distinguish grammatical strings from non-grammatical ones, and in the context of unification-based formalisms, every parse of a grammatical string should result in a relational structure that tells us about the syntactic properties of the string.

2 Unification-based grammar

On the face of it, the formalism presented here will look much like HPSG with a reduced feature geometry. In this early version, however, it is closer to CUG. It is shown how it embeds the basics of other formalisms too.

$$\left[\begin{array}{l} \text{sign} \\ \textsc{phon}\left\langle !\ \text{lp-on-string},\ldots ! \right\rangle \\ \textsc{cat}\ \text{cat} \\ \textsc{order}\ \text{lp} \\ \textsc{sem}\left\langle !\ \left[\begin{array}{l} \text{relation} \\ \textsc{pred}\ \text{predicate} \\ \textsc{arg0} \left[\begin{array}{l} \text{index} \\ \textsc{case} \left[\begin{array}{l} \textsc{theta}\ \text{predicate} \\ \textsc{case-type}\ \text{case-type} \end{array}\right] \end{array}\right] \end{array}\right],\ldots ! \right\rangle \end{array}\right]$$

The intuition behind this formalism is that weak precedence constraints, e.g "α precedes β, but other words may intervene them," should be easily expressible. The simple move is to let the ORDER attribute, which is known from CUG, take four values, rather than just left and right application; namely, left and right weak precedence and left and right immediate precedence. The CAT attribute points to information about syntactic categories. The semantic relations relate arguments which contain information about case (to ensure agreement), but also information about Θ-roles. The idea is that the relation between the main argument of, say, a sentence (an event) and the argument can be inferred on the basis of case information and information about Θ-roles. The inferred value is the value of the PRED attribute. In other words, a neo-Davidsonian-style semantics is assumed, and in a simple sentence, each argument is linked to the event argument by a numbered argument relation.

The feature geometry is augmented by HPSG-style attributes HEAD-DTR and NON-HEAD-DTR to represent functor-argument structure; since weak precedence constraints are employed, this is not the same as constituent structure. These are the only recursive attributes. The PHON and SEM diff-lists are in principal unbounded,[5] but it is easy to ensure that only lexical items introduce phonological material and semantic relations. The reader may, under these circumstancies, already see how an NPTIME result can be obtained. The trick is to establish the *polysize model property*, i.e that any satisfiable input formula is satisfied in a model polynomial to the length of the input formula. It holds for our toy formalism that every model (feature structure) that satisfies a formula (a string) *is* polynomial in the length of the input string, if unary extensions are acyclic, i.e no unary rule applies twice in the same unary extension. In particular, if a string σ is satisfiable, there exists a model M of size less than or equal to $(2|\sigma|-1)(u+1)\times\mathbf{paths}$ where u is the number of unary rules or phrases in the grammar, $\mathbf{paths} = |\{\pi \in \mathbf{Lbls}^*|\text{no label occurs twice in } \pi\}|$, and $\mathbf{Lbls}$ is the set of labels that denote attributes. In CUG, $u = 0$. This is in fact an alternative proof of the result obtained in (Trautwein, 1995).

On the other hand, our grammar formalism is NP-hard, even if unary extensions are acyclic. Consider the following sketch of a proof: Since the satisfiability problem of propositional logic with exactly three literals per conjoined clause (3SAT) is as hard as any problem in this class, it follows that if our grammar formalism encodes 3SAT, then its universal recognition problem is hard (and thus complete) for NPTIME. The intuition is to let each propositional variable correspond to a lexical item with category values $true$ or $false$, and independently, the item is either transitive or takes no complements at all. It is ensured that at least one of the three constituents that make up a phrase (since no adjuncts exist) has the category value $true$. A conjunction rule is added. The consistency of the assignments is ensured in the semantics features. The trick is simple: For each sign that corresponds to a propositional variable, a novel attribute is introduced whose value is unified with the category value. The semantic values of all the constituents are unified, when phrases are built. Consequently, the truth assignment has to be consistent. Our grammar now encodes 3SAT in the sense that if a propositional formula is valid, it translates into a string that is accepted.

Theorem 2.1. *Our grammar formalism is NP-complete if unary extensions are acyclic.*

As already mentioned, Trautwein defines an NP-completeness result for HPSG, but a rather amputated version of this formalism. Let us consider the restrictions he places on HPSG. One is the same acyclicity requirement that we adopted, but there are many more. The most important restriction is that he does not allow for weak linear precedence constraints (and domain union and more), the obvious reason being that he employs a generative perspective on derivation. His reconstruction is also vulnerable to the introduction of sets. Ours is not, if (non-wellfounded) sets are introduced by polyadic

[5] Diff-lists are lists, where the last element on the list has been extracted. It is, in other words, possible to remove elements from both ends.

modalities (Reape, 1994).[6] We can also (freely) add feature cooccurence restrictions (which seems to be an obvious way to implement parameter-based learning), for instance.

Say unary extensions can be cyclic. What is the complexity of our grammar formalism then? One way to estimate this is to translate our grammar in specific logical languages. Such translation exist, but some of them are very partial, in that they do not encode the full recognition problem. The simplest logic of (Blackburn and Spaan, 1993), for instance, does not encode the recognition problem. It is possible to do so in the most complex language, whereas the intermediate one gives us a specification language that does the job, but only with some assistance from the outside. In the true sense of the word, it is not *stand-alone* either. The intermediate language is hybrid logic with global implication; the stronger one is Kasper-Rounds logic with global implication. None of these languages encode weak linear precedence, since they don't have any non-deterministic operators, such as the Kleene star operator. This is remedied here. Two languages are defined of the same complexity as the languages in (Blackburn and Spaan, 1993) which encode weak linear precedence: $\mathsf{HDL}^{\Diamond}$ and $\mathsf{PDL}^{\Diamond,\cap}$, respectively the dynamic extension of hybrid logic with global implication and propositional dynamic logic with global implication and intersection (or path equations). $\mathsf{HDL}^{\Diamond}$ is decidable in EXPTIME (Areces et al., 1999), and $\mathsf{PDL}^{\Diamond,\cap}$ is undecidable (Søgaard, 2006a).

The paper is too short to represent a full specification of a grammar in one of the logics, but the idea is this: Our logics are interpreted over Kripke models. If the reader is willing to excuse us some mild abuse of notation, the following is said to hold on our specification in $\mathsf{PDL}^{\Diamond,\cap}$, for instance:

$$\begin{bmatrix} \text{FUNCT} \begin{bmatrix} \text{NUM } \boxed{1} \end{bmatrix} \\ \text{ARG} \begin{bmatrix} \text{NUM } \boxed{1} \end{bmatrix} \end{bmatrix}$$
$$\models \langle \text{FUNCT}; \text{NUM} \cap \text{ARG}; \text{NUM} \rangle \top$$

Or in $\mathsf{HDL}^{\Diamond}$, though the two formulas are not quite equivalent:

$$\langle \text{FUNCT}; \text{NUM} \rangle i \wedge \langle \text{ARG}; \text{NUM} \rangle i$$

Some notes on the difference between the two formalizations follow. This question is namely of some importance to us, since the gain in expressivity (of $\mathsf{PDL}^{\Diamond,\cap}$) can then be weighed to the loss of decidability.

Hybrid logics extend the language of Boolean connectives with the modalities, as they appear in basic modal logic, i.e $\langle \alpha \rangle \phi$ means that there is an α-transition to a state in ϕ, and nominals. A nominal is a propositional variable which is interpreted as a singleton subset; if $M, w \models i$, it follows that the valuation of w is $\{i\}$. The satisfaction operator $@_i \phi$ says "go to state i and evaluate ϕ", intuitively. The non-deterministic Kleene star operator is adopted, and the master modality is defined with its dual and $+$-varieties, e.g $\langle + \rangle \phi$ is true if there is state in ϕ one or more transitions down. The $\Diamond$ is a global modality, and $\Diamond \phi$ means that ϕ is true "somewhere" in the model. Its dual ($\Box$) means something like "everywhere".

The difference between the two formulas in the above now amounts to this: A nominal names a state, and everywhere the nominal is used again, it receives the same interpretation. This is important when it comes to rules. If a rule employs nominals, and it is used multiple times in a derivation, it will enforce undesired reentrancies. This is unfortunate, and the only way to deal with it is to somehow dynamically reset the nominals for each application. In some sense, this is cheating; in addition, resetting is finitely bound.

The logics must now be shown to encode our grammar. It was shown in the small example above how to *describe* feature structures, so we can easily describe lexical entries and thus give lexical input to the parsing procedure. The parsing procedure will be seen as connecting lexical input and a root node (roughly, what corresponds to `start` on the proof-theoretic perspective) as a directed acyclic graph. It is then necessary to enforce the Kripke models to be acyclic, rooted, and connected. The relevant constraints are listed in Figure 2, where $|[\![root]\!]| = 1$. It is trivial to encode phrases, and a type hierarchy is encoded in the Boolean fragment of the languages.

Theorem 2.2. *Our grammar formalism is in EXPTIME if dynamic resetting of nominals is ensured, but undecidable if existential quantification over nominals or true path equations are allowed.*

Of course we have presented no actual undecidability result for the full version of our grammar (though such a proof can easily be constructed; see (Johnson, 1988), for instance). It is just the case that the full fragments seem to be specified only in undecidable languages. Just a comment here on the PSPACE-fragment, and then the last paragraphs are devoted to PTIME-fragments. Hybrid logic and modal logic are both decidable in PSPACE. What does this correspond to in linguistic terms? In a sense, this is an unprincipled grammar, unrestricted with regard to the licensed extensions. It is, at least, unprinci-

[6] In fact, this is a bit more tricky than it seems at first sight. Most monadic (but polymodal) modal logics have PTIME model checking problems, and this property is silently assumed in the NPTIME proof above. In other words, it is necessary for obtaining NPTIME that grammars can be encoded in a logic with PTIME model checking. In the unrestricted polyadic case, this may not always hold, as pointed out to me by Martin Lange. If the upper bound on the arity of polyadic modalities is fixed, both PTIME model checking and the polysize model property can be established. If that upper bound is polynomial in the length of the input, it seems that only the polysize model property can be established, and the NPTIME proof thus fails to apply. If the relations can be represented in polynomial space, however, i.e a d-ary relation only has n^d members, things may be fine anyway. I will study this question in more detail in future work.

Acyclicity	$\neg @_i . \langle + \rangle i$	$\neg \langle \epsilon \cap + \rangle \top$
Connectivity/rootedness	$\Diamond(\neg \langle + \rangle^{-1} root) \wedge \Box(\langle * \rangle root)$	do.

Figure 2: Some properties of feature structures

pled in the sense that no universal rules apply unless they are limitations on the class of Kripke models considered. (In fact, this may not be too silly. Universal grammar is then a lattice of modal logics. In standard modal logic, two nodes can be said to be information-sharing, though not identical, if $\Diamond_1 \phi \rightarrow \Diamond_2 \phi$. This is fine as long as reentrant information is bound (Kracht, 1995).)

2.1 Polynomial time

The possible natural languages may reasonably be identified as the efficiently communicative and learnable ones, but what does efficiency mean here? Certain researchers rely on heuristics, remain agnostic, and some have argued for the adequacy of NPTIME algorithms. The more traditional answer is tractability, however; that is, recognizability in polynomial (deterministic) time. The key to PTIME, in the context of unification-based grammar as well as other logical problems, is ambiguity reduction. The whole trick is to find sound and linguistically adequate restrictions on ambiguity. This is by no means trivial. Some results have already been mentioned. They are not exhaustive, but it suffices to say here that there is no consensus about which fragments are more adequate than others.

One important difference between PTIME fragments is *how* much ambiguity reduction is going on, and how directly ambiguity is constrained. (Søgaard, 2006b) presents a PTIME fragment of HPSG which is really quite expressive. The only constraints are that

- recursive feature geometry is polynomially bound in the length of the string, and
- ambiguity is bound by some constant, i.e at some point in the derivation, signs begin to combine unambiguously.

Our use of PHON is a nice example of a recursively defined feature which is polynomially bound in the length of the string. The second constraint amounts to k-ambiguity. Our two constraints may sound odd to most linguists. Ambiguity is a pervasive phenomenon, and it seems to pop up everywhere. It is important to remember that HPSG is based on typed feature structures, however. In brief this means that grammatical information may be underspecified with respect to a disjunct set of specific subtypes. In the context of other information, further specification can be inferred (when a greatest lower bound of the old and new information is found in the hierarchy of types). Consider lexical ambiguity, for instance. Underspecification here amounts to replacing two lexical entries of, say, *sleep*, a nominal and a verbal one, with a single lexical entry associated with a lexical type whose lexical types are limited to the nominal and the verbal one.

The major advantage of the formalism introduced in (Søgaard, 2006b) is that it allows for unordered rules and weak linear precedence. In terms of generative capacity, it really cross-cuts the Chomsky Hierarchy, i.e. its denotation includes langauges that are not even mildly context-sensitive, but not all regular ones. Linguistic motivation for this is presented.

3 Conclusions

We have identified fragments of unification-based grammar for the complexity time classes PTIME, NPTIME, PSPACE, EXPTIME and UNDEC, and briefly discussed their potential in the linguistic sciences. Some researchers argue against complexity studies of this kind, saying that linguistics, as a field, is too premature for worrying about computational complexity. Since some linguistic constraints are (probably) yet to be identified, it is also yet uncertain what exactly makes natural languages efficiently communicative. On my view, complexity studies are important guides for theoretical linguistics, the associated field of knowledge is a ressource for empirical linguistics and computer science, and it is certainly fundamental in language technology. For instance, (Johnson, 1988) had impact on both the (empirical) LFG community and on the design of actual grammar engineering platforms. The idea is, of course, that if feasible fragments can be implemented, the grammar writer no longer has to worry about ensuring reasonable runtimes by introducing *ad hoc* constraints. Such constraints are instead hardwired into the system.

References

Carlos Areces, Patrick Blackburn, and Maarten Marx. 1999. The computational complexity of hybrid temporal logics. *The Logic Journal of the IGPL*, 8(5):653–679.

Patrick Blackburn and Edith Spaan. 1993. A modal perspective on the computational complexity of attribute value grammar. *Journal of Logic, Language and Information*, 2(2):129–169.

Mark Johnson. 1988. *Attribute-value logic and the theory of grammar*. CSLI Publications, Stanford, California.

Mark Johnson. 1991. Features and formulae. *Computational LInguistics*, 17:131–152.

Robert Kasper and William Rounds. 1986. A logical semantics for feature structures. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, pages 257–265, New York, NY.

Stephan Kepser and Uwe Mönnich. 2003. Undecidability results for HPSG. In Anton Nijholt and Giuseppe Scollo, editors, *Algebraic Methods in Language Processing*, pages 1–12.

Marcus Kracht. 1995. Is there a genuine modal perspective on feature structures? *Linguistics & Philosophy*, 18:401–458.

Geoffrey Pullum and Barbara Scholz. 2005. Contrasting applications of logic in natural language syntactic description. In Petr Hájek, Luis Valdés-Villanueva, and Dag Westerståhl, editors, *Logic, methodology and philosophy of science*, pages 481–503. King's College Publications, London, England.

Mike Reape. 1994. A feature value logic with intensionality, nonwellfoundedness and functional and relational dependencies. In C. J. Rupp, M. Rosner, and R. Johnson, editors, *Constraints, language and computation*, pages 77–110. Academic Press, San Fransisco, California.

Hiroyuki Seki, Ryuichi Nakanishi, Yuichi Kaji, and Sachiko Andoand Tadao Kasami. 1993. Parallel multiple context-free grammars, finite-state translation systems, and polynomial-time recognizable subclasses of lexical-functional grammars. In *Proceedings of the 31st Annual Meeting on Association for Computational Linguistics*, pages 130–139, Columbus, Ohio.

Anders Søgaard and Petter Haugereid. 2006. Functionality in grammar design. In Stefan Werner, editor, *Proceedings of the 15th NODALIDA*, pages 180–189, Joensuu, Finland.

Anders Søgaard. 2006a. *HPSG and model generation*, volume 7 of *Center for Language Technology Working Papers*. University of Copenhagen Press, Copenhagen, Denmark. In press.

Anders Søgaard. 2006b. Tractable typed attribute-value languages and freer word order that cross-cut the chomsky hierarchy. *Submitted recently*.

Marten Trautwein. 1995. *Computational pitfalls in tractable grammar formalisms*. Ph.D. thesis, University of Amsterdam, Amsterdam, the Netherlands. ILLC Dissertation Series DS-1995-15.