The complexity of linguistically motivated extensions of tree-adjoining grammar

Anders Søgaard
University of Copenhagen
anders@cst.dk

Timm Lichte
University of Tübingen
timm.lichte@uni-tuebingen.de

Wolfgang Maier
University of Tübingen
wo.maier@uni-tuebingen.de

Abstract

This paper proves the NP-hardness of three extensions of tree-adjoining grammar (TAG): FO-TAG [2], RSN-MCTAG [6] and TT-MCTAG [7]. The complexities of these extensions have all been presented as open problems in the literature. The extensions have been proposed to model scrambling and free word order phenomena in languages such as German, Korean and Japanese. It is shown that one of them also generates the MIX language. Finally, some polynomial time fragments are defined.

1 Introduction

Natural language has been shown to contain constructions which can not be adequately represented using context-free grammar (CFG), such as cross-serial dependencies. While first shown to exist in Swiss German [12], they can also be found in Tagalog [8]. Several formalisms have been introduced that provide more expressive power than CFG while staying computationally tractable, i.e. while retaining polynomial recognition. One of these formalisms is tree-adjoining grammar (TAG). Some knowledge of TAG is assumed in this paper. Consult [5] otherwise for a nice introduction. Informally, a TAG consists of finite sets of terminals and nonterminals and finite sets of initial and auxiliary trees. Larger trees are derived by substituting ↓-marked frontier nodes with trees or by adjoining trees (that have a root node and a $*$-marked frontier node with the same nonterminal) to interior nodes. The language of a TAG is the set of strings that are in the yield of trees that can be derived from an initial tree.

[2] showed that TAG does not have the expressive power necessary to capture scrambling and free word order phenomena in languages such as German, Korean and Japanese. Here's an example from Korean:

(1)

jatongcha-lul	*keu-ka*	*surihakess-tako*
car.DEF.ACC	PRO.3SG.NOM	repair.INF
yakosokhaessta		
promise.FIN		

'He promises to repair the car.'

In Korean, adjuncts and arguments scramble. In this case, an argument of the lower clause even appears in the upper clause. The structure is thus discontinuous. Scrambling over clause boundaries is sometimes referred to as long-distance scrambling. Such long-distance scrambling is beyond the expressive power of TAG, but even scrambling phenomena within the clause receive rather unelegant analyses in TAG.

ID/LP grammar was proposed by [11] for scrambling and free word order within the clause. In ID/LP grammar, the productions of CFG are split into immediate dominance (ID) and linear precedence (LP). For instance, the production $S \rightarrow \alpha\beta$ is split into the (unordered) ID $S \rightarrow \alpha\beta$ and the LP $\alpha \prec \beta$. The LPs can be relaxed or removed. In other words, free word order and scrambling do not complicate grammars, but simplify them. ID/LP grammar still only generates context-free languages, but in fact the universal recognition problem becomes NP-hard. This was proven for the fragment of ID/LP grammar with no LPs (UCFG) in [1] by an interesting application of the vertex cover problem; this problem is also used here to establish the NP-hardness of FO-TAG, RSN-MCTAG and TT-MCTAG.

The vertex cover problem involves finding the smallest set V' of vertices in a graph $D = \langle V, E\rangle$ such that every edge has at least one endpoint in the set. Formally, $V' \subseteq V : \forall\{a, b\} \in E : a \in V' \vee b \in V'$. The problem is thus an optimization problem, formulated as a decision problem:

INSTANCE:	A graph $D = \langle V, E\rangle$ and a positive integer k.
QUESTION:	Is there a vertex cover of size k or less for G?

Say $k = 2, V = \{a, b, c, d\}, E = \{\langle a, c\rangle, \langle b, c\rangle, \langle b, d\rangle, \langle c, d\rangle\}$, for instance. One way to obtain a vertex cover is to go through the edges and underline one endpoint of each edge. If you can do that and only underline two vertex symbols, a vertex cover has been found. Since $|V| = 4$, this is equivalent to leaving two vertex symbols untouched. Consequently, the vertex cover problem for this specific instance is encoded by this UCFG, where δ is a bookkeeping dummy symbol:

$$\begin{array}{rcl}
S & \rightarrow & \rho_1\rho_2\rho_3\rho_4 UU\delta\delta\delta\delta \\
\rho_1 & \rightarrow & a|c \\
\rho_2 & \rightarrow & b|c \\
\rho_3 & \rightarrow & b|d \\
\rho_4 & \rightarrow & c|d \\
U & \rightarrow & aaaa|bbbb|cccc|dddd \\
\delta & \rightarrow & a|b|c|d
\end{array}$$

ρ_i captures the ith edge in E. The input string $\omega = aaaabbbbccccdddd$. One derivation tree in our example will have the form:

$$[[aaaa]_U[bbbb]_U[c]_{(b,c)}[c]_{(a,c)}[c]_{(c,d)}[c]_\delta \\ [d]_{(b,d)}[d]_\delta[d]_\delta[d]_\delta]_S.$$

Generally, the first production has as many ρ_i's as there are edges in the graph, $|V| - k$ many U's and $|E| \times |V| - |E| - |E| \times (|V| - k)$ many δ's, i.e. the length of the string minus the number of edges and the extension of $|V| - k$ many U's. The ρ_i productions are simple, U extends into $|E|$ many a's or b's and so on, and δ extends into all possible vertices. Since the grammar and input string can be constructed in polynomial time from an underlying vertex cover problem $\langle k, V, E\rangle$, universal recognition of UCFG must be at least as hard as solving the vertex cover problem. Since the vertex cover problem is NP-hard [4], the universal recognition problem for totally unordered type 2 grammars is therefore NP-hard. It is easy to see that it is also in NP. Simply guess a derivation, linear in the size of the string, and evaluate it in polynomial time.

Several extensions, as already mentioned, have been proposed for long-distance scrambling. See [6] for a partial survey. Most proposals in one way or another relax the notion of immediate dominance between mothers and daughters in trees. Note that immediate dominance is already relaxed in TAG, since new trees can be adjoined to daughter nodes.

FO-TAG is probably the simplest proposal, as it is very similar to ID/LP grammar. In variants of multicomponent TAG (MCTAG), the relaxation of immediate dominance is obtained by splitting auxiliary trees into smaller trees and dominance (not immediate dominance) links. The grammar then consists of sets of trees rather than just elementary trees, except for the initial trees. It is usually a restriction that every auxiliary tree in a set must be applied to the derived tree, in a single derivation step. In the absence of any other restrictions, the fixed recognition problem of (even lexicalized) MCTAG can be shown to be NP-hard [10, 3]. [3] proves the NP-hardness of the fixed recognition problem of a particular grammar that solves all instances of the three-partition problem by accepting only some input.

Several variants of MCTAG have appeared in the last years, and their complexities have been presented as open problems. In this paper, our concern is with FO-TAG [2], RSN-MCTAG [6] and TT-MCTAG [7]. It seems that none of these extensions of TAG are able to easily reconstruct the three-partition problem, but they all have the power to solve the vertex cover problem. Or more accurately, for every instance of the vertex cover problem, there is a polynomial (and linear) translation into a grammar of one of these kinds and a string such that the string is only recognized by the grammar iff the source problem instance has a solution. Since the vertex cover problem is known to be NP-complete, the universal recognition problems of the three extensions are thus NP-hard. It is trivial to show that the extensions are also NP-complete.

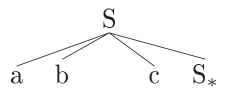
2 FO-TAG

Free order TAG (FO-TAG) was introduced in [2]. The definition is presented in Definition 2.1.

Definition 2.1. G is a free order tree-adjoining grammar iff $G = \langle N, T, I, A, S\rangle$ such that $G' = \langle N, T, I, A, S\rangle$ is a tree-adjoining grammar [5], except that initial and auxiliary trees are now tuples of unordered trees and LPs of the form $\alpha \prec \beta$ where $\alpha, \beta \in N$.

The language of a FO-TAG is, as in the case of TAG, the set of strings that are in the yield of the trees that can be derived from an S-rooted initial tree by adjunction and substitution.

Example 2.2. FO-TAG does not seem to generate the MIX language, but the totally unordered extension of it does, i.e. if a language L is generated by a FO-TAG, the language generated by its totally unordered extension is the set of all permutations of strings in L. See [13] for the notion of total unordering. The MIX language is conjectured not to be mildly context sensitive. It consists of all strings in $\{abc\}^*$; that is, every string that consists of the same number of a's, b's and c's. To see this, consider the auxiliary tree:

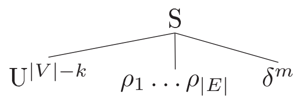


This generates the language whose permutations is the MIX language with an appropriate initial tree to begin the derivation and an appropriate auxiliary tree to end it.

For our NP-hardness proof, it is shown how to reconstruct the vertex cover problem in FO-TAG. The theorem is a trivial consequence of the result in [1], since every UCFG is also a FO-TAG.

Theorem 2.3. *The recognition problem of FO-TAG is NP-hard.*

Proof sketch 2.4. For each problem instance D, k we construct a FO-TAG $G = \langle N, T, I, A, S\rangle$ and a string σ such that σ is in the language of G iff D, k has a solution. First σ is defined as the concatenation of $|E|$ many v_i's for each $v_i \in V$. So for the instance $k = 2, V = \{a, b, c, d\}, E = \{\langle a, c\rangle, \langle b, c\rangle, \langle b, d\rangle, \langle c, d\rangle\}$, for example, a possible string is $aaaabbbbccccdddd$. It is then defined that the tuple of the elementary tree



with $m = |E| \times |V| - |E| - |E| \times (|V| - k)$. The S-initial tree has $|V| - k$ many U's as daughters. Consequently, there are only k vertices left to cover the graph. It should not be difficult to see how the proof proceeds; it is, in all respects, analogous to the proof for UCFG.

3 RSN-MCTAG

Restricted multicomponent TAG with shared nodes (RSN-MCTAG) was introduced in [6]. Its formal definition is presented in Definition 3.1.

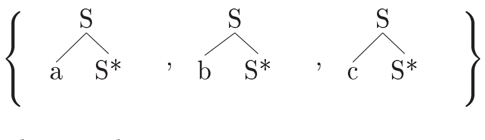
Definition 3.1. G is a restricted multicomponent tree-adjoining grammar with shared nodes iff $G = \langle N, T, I, A, \mathcal{A}, S \rangle$ such that $G' = \langle N, T, I, A, S \rangle$ is a tree-adjoining grammar [5], and where $\mathcal{A} \subseteq 2^{I \cup A}$.

The next step is to define a relation on the derivation tree R_s for node-sharing. A derivation tree is a tuple $D = \langle \mathsf{Trees}, \mathsf{Drvs} \rangle$, where $\mathsf{Trees} \subseteq (I \cup A)$, and $\mathsf{Drvs} \subseteq \mathsf{Trees} \times \mathsf{Trees} \times \mathsf{GornAddrs}$, where $\mathsf{GornAddrs}$ is the set of Gorn addresses. R_s is defined:

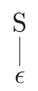
$$\begin{aligned} R_s \quad = \quad & \{\langle n_1, n_2 \rangle \mid n_1, n_2 \in \mathsf{Trees}, n_2 \\ & \text{is immediately dominated by } n_1 \\ & \text{or there are } t_1, \ldots, t_k \in \mathsf{Trees} \\ & \text{such that } t_1 \\ & \text{is immediately dominated by} \\ & n_1, n_2 = t_k \text{ and for all } i, \\ & 1 \leq i \leq (k-1) : \langle t_i, t_{i+1}, p' \rangle \text{ with } t_i \\ & \text{being an auxiliary tree with root} \\ & \text{note address } p' \} \end{aligned}$$

The language of a RSN-MCTAG is the set of strings that are in the yield of the trees that can be derived from an S-rooted initial tree by simultaneous adjunction and substitution of all elements of sets, with derivation tree $D = \langle \mathsf{Trees}, \mathsf{Drvs} \rangle$ such that for every $\{\beta_1, \ldots, \beta_n\} \in \mathcal{A}$, $\beta_i \in \mathsf{Trees}$, and there is a γ such that β_i is immediately dominated by (is the daughter of) γ in the derived tree or linked to γ by a chain of root adjunctions. In other words, $R_s(\gamma, \beta_i)$. In addition, at least one β_j must be immediately dominated by γ in the derivation tree. Due to the simultaneity constraint, no two β_i, β_j can dominate each other .

Example 3.2. Neither RSN-MCTAG or SN-MCTAG [6], that is, RSN-MCTAG without the immediate dominance restriction on set application, generate the MIX language. This is possible, however, if we give up the constraint that no two β_i, β_j can dominate each other. To see this, consider the set:



and the initial tree:

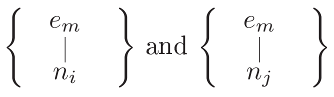


It should be relatively easy to see that this generates the MIX language if the auxiliary trees in a set can dominate each other.

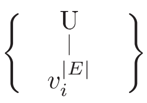
For our NP-hardness proof, it is shown how to reconstruct the vertex cover problem in RSN-MCTAG:

Theorem 3.3. *The recognition problem of RSN-MCTAG is NP-hard.*

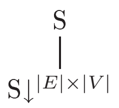
Proof sketch 3.4. For each problem instance D, k we construct a RSN-MCTAG $G = \langle N, T, I, A, \mathcal{A}, S \rangle$ and a string σ such that σ is in the language of G iff D, k has a solution. First σ is defined as the concatenation of $|E|$ many v_i's for each $v_i \in V$. So for the instance $k = 2, V = \{a, b, c, d\}, E = \{\langle a, c \rangle, \langle b, c \rangle, \langle b, d \rangle, \langle c, d \rangle\}$, for example, a possible string is *aaaabbbbccccdddd*. It is then defined that $N = \{D, U, S, e_1, \ldots e_{|E|}, \delta\}$. For each $e_m = \langle n_i, n_j \rangle \in E$, singleton sets are introduced:



For each $v_i = V$, singleton sets are introduced:



The set in Figure 1 is then introduced.
Finally, the S-initial tree is added:



The set in Figure 1 needs to saturate $|V| - k$ many U's. Consequently, there are only k vertices left to cover the graph. Consequently, only the trees that relate edge nonterminal symbols e_i with the terminals that are not used to build U's can be build. In our example, this will be those with terminals c, d. The δ-trees are just there for technical reasons.

The reconstruction shows that the recognition problem of RSN-MCTAG is NP-hard.

The proof also applies to SN-MCTAG [6], of course. Moreover the proof is independent on constraints such as tree-locality and set-locality, since the elementary trees of each set all apply, simultaneously, to the same tree and, therefore, the same set.

4 TT-MCTAG

Multicomponent TAG with tree tuples (TT-MCTAG) is introduced in [7]. Its formal definition is presented in Definition 4.1.

Definition 4.1. G is a multicomponent tree-adjoining grammar with tree tuples iff $G = \langle N, T, I, A, \mathcal{T}, S \rangle$ such that $G' = \langle N, T, I, A, S \rangle$ is a tree-adjoining grammar [5], and where $\mathcal{T} \subseteq (I \cup A) \times 2^A$ such that for each $\langle \gamma, \{\beta_1, \ldots, \beta_n\} \rangle \in \mathcal{T}$ the frontier nodes of the destination tree γ include at least one terminal symbol.

The next step is to define a relation on the derivation tree R_s for node-sharing. This is just as in RSN-MCTAG. A derivation tree is a tuple $D = \langle \mathsf{Trees}, \mathsf{Drvs} \rangle$, where $\mathsf{Trees} \subseteq (I \cup A)$, and $\mathsf{Drvs} \subseteq \mathsf{Trees} \times \mathsf{Trees} \times \mathsf{GornAddrs}$, where $\mathsf{GornAddrs}$ is the set of Gorn addresses.

The language of a TT-MCTAG is the set of strings that are in the yield of the trees that can be derived

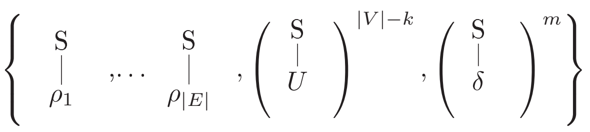
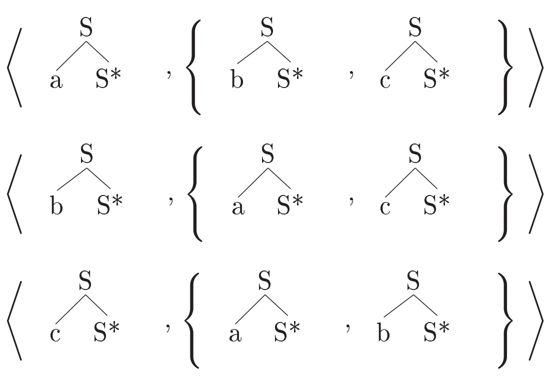


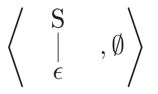
Fig. 1: *A set in the RSN-MCTAG reconstruction of the vertex cover problem.* $m = |E| \times |V| - |E| - |E| \times (|V| - k)$.

from an S-rooted initial tree by adjunction and substitution with derivation tree $D = \langle \mathsf{Trees}, \mathsf{Drvs} \rangle$ such that for every $\langle \gamma, \{\beta_1, \ldots, \beta_n\} \rangle \in \mathcal{T}$, $\beta_i \in \mathsf{Trees}$, and either β_i is substituted for a frontier node in γ, or adjoined to an interior node of γ or $R_s(\gamma, \beta_i)$.

Example 4.2. TT-MCTAG also generates the MIX language. To see this, consider the tree tuples:



and the tree tuple:

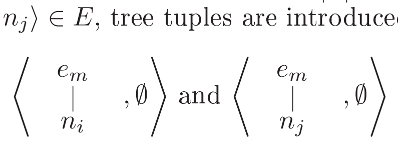


It should be relatively easy to see that this generates the MIX language. The saturation requirement in TT-MCTAG ensures that you use up all the trees in the tuples, whenever destination trees are introduced.

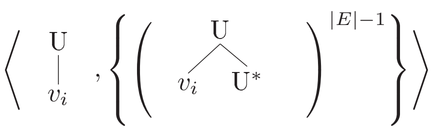
For our NP-hardness proof, it is shown how to reconstruct the vertex cover problem in TT-MCTAG:

Theorem 4.3. *The recognition problem of TT-MCTAG is NP-hard.*

Proof sketch 4.4. For each problem instance D, k we construct a TT-MCTAG $G = \langle N, T, I, A, \mathcal{T}, S \rangle$ and a string σ such that σ is in the language of G iff D, k has a solution. First σ is defined as the concatenation of $|E|$ many v_i's for each $v_i \in V$, prefixed by the symbol †. So for the instance $k = 2, V = \{a, b, c, d\}, E = \{\langle a, c\rangle, \langle b, c\rangle, \langle b, d\rangle, \langle c, d\rangle\}$, for example, a possible string is $\dagger aaaabbbbccccdddd$. It is then defined that $N = \{U, S, e_1, \ldots e_{|E|}, \delta\}$. For each $e_m = \langle n_i, n_j \rangle \in E$, tree tuples are introduced:



For each $v_i = V$, tree tuples are introduced:



Finally, the tree tuple in Figure 2 is introduced.

The S-initial tree needs to saturate $|V| - k$ many U's. Consequently, there are only k vertices left to cover the graph. Consequently, only the edge-tuples – that is, the ones with destination trees with interior nodes ρ_i – with the terminals that are not used to build U's can be build. In our example, this will be those with terminals c, d. The δ-trees are just there for technical reasons.

The reconstruction shows that the recognition problem of TT-MCTAG is NP-hard. It is, as already said, trivial to show NP-inclusion, since derived trees are clearly polynomial in the length of the input. Consequently, it is possible to guess a model and evaluate it in polynomial time.

5 Polynomial time fragments

This section defines some fragments of the above extensions whose recognition problems can be solved in polynomial time. Our first fragment is FO-TAG(k). An FO-TAG is a FO-TAG(k) iff a discontinuous tuple, that is, a tuple in which the linearization of the tree is not fully specified by the LPs, has a yield of at most k terminals.

Theorem 5.1. *The recognition problem of FO-TAG(k) is in P.*

Proof sketch 5.2. Consider the simpler case of UCFG(k), defined in an analogous fashion. Our first step is to define a chart. See [13] for a similar construction. If you have a UCFG $G = \langle N, T, P, S \rangle$ and some string $\omega_1 \ldots \omega_n$. Construct $G_\omega = \langle N_\omega, T_\omega, P_\omega, \{_1S_n\} \rangle$ such that

$$T_\omega \quad = \quad \{\omega_1, \ldots, \omega_n\}$$

and, recursively

(a) $(\omega_i \in T_\omega$ and $A \rightarrow \omega_i \in P) \implies ({}_iA_i \in N_\omega$ and ${}_iA_i \rightarrow \omega_i \in P_\omega)$

(b) $({}_iB_j, {}_{j+1}C_k, \ldots, {}_{m-1}X_m \in N_\omega$ and $A \rightarrow \{B, C, \ldots, X\}) \implies ({}_iA_m \in N_\omega \wedge {}_iA_m \rightarrow \{{}_iB_j {}_{j+1}C_k, \ldots, {}_{m-1}X_m \in P_\omega)$

The upper bound on $|P_\omega|$ is roughly:

$$\sum_{i<n}^{0\leq i}(|N| \times (n-i) \times \sum_{j<(n-i)}^{0\leq j} (|N|^{n-j} \times (n-i) \times (n-j)))$$

To see this, note that there are $\frac{n^2+n}{2}$ spans to assign one of $|N|$ nonterminals, and that each nonterminal in the chart with span of length $n-i$ may correspond to, roughly,

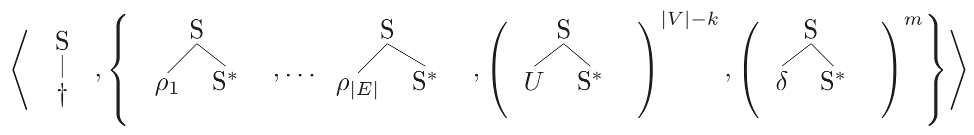


Fig. 2: *The S-initial tree in the TT-MCTAG reconstruction of the vertex cover problem.* $m = |E| \times |V| - |E| - |E| \times (|V| - k)$.

$$\sum_{j<(n-i)}^{0\le j} (|N|^{n-(i+j)} \times (n-i) \times (n-(i+j)))$$

productions, since you can partition the span in $(n-i) \times (n-(i+j))$ and assign $N^{(n-(i+j))}$ many combinations of nonterminals for each partitioning.

In UCFG(k), this number is much lower, namely

$$\sum_{i<k}^{0\le i}(|N| \times (n-i) \times \sum_{j<(k-i)}^{0\le j} (|N|^{k-(i+j)} \times (k-i) \times (k-(i+j))) + \sum_{i<n}^{k<i}(|N|^3 \times (n-i))$$

if it is assumed that any rule that spans more than k positions is binary. The bound implies parsing in $\mathcal{O}(n^3)$. This reflects that a k bound on yield means a k' bound on arity, and if there is such a bound, the underlying CFG can be constructed in $\mathcal{O}(k'!)$ time. In FO-TAG(k), the same effect is seen on charts. See [14] for a chart-based parsing algorithm for ordinary TAG. It is easy to see that since charts are polynomial in the length of the string, so is the time complexity of the recognition problem.

Similarly, a k bound on the number of nodes in unordered elementary trees will allow you to generate the underlying TAG in $\mathcal{O}((k-1)!)$. [6] defines a polynomial fragment of RSN-MCTAG (RSN-MCTAG(k)) by adding a restriction that roughly means you can only have k elementary trees between elements of tree sets.

A similar constraint can be imposed on TT-MCTAG. Or we can impose a k-gap degree constraint, as in non-projective dependency parsing [9]. In fact, the two constraints are intimately related. If there is a k bound on the elementary trees that can occur between elements of tree sets, there is also a k' bound, linear in k, on the gap degree.

FO-TAG(k) is weakly equivalent to TAG and FO-TAG, but the k-constraint is not harmless, from a linguistic point of view, since intra-clausal unordering is relevant for arbitrary yields. The second proposal, to restrict the number of nodes in unordered elementary trees, seems more realistic; most grammars have reasonable bounds on the number of nodes. The same applies to RSN-MCTAG(k). [9] shows that a low gap degree is realistic for a number of languages.

6 Conclusion

It was shown that FO-TAG [2], RSN-MCTAG [6] and TT-MCTAG [7], three extensions of tree-adjoining grammar that are suited for analyzing scrambling and free word order phenomena in languages such as German, Korean and Japanese, have NP-hard universal recognition problems. All three extensions are also NP-complete, but only one of them, TT-MCTAG, generates the MIX language. Some polynomial time fragments were defined. The NP-hardness proofs imply that tree-local MCTAG is NP-hard, while weakly equivalent to TAG. Consequently, any translation from tree-local MCTAG into TAG is exponential. This mirrors the relation between ID/LP grammars and CFG.

References

[1] E. Barton. The computational difficulty of ID/LP parsing. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 76–81, Chicago, Illinois, 1985.

[2] T. Becker, A. K. Joshi, and O. Rambow. Long-distance scrambling and tree adjoining grammars. In *Proceedings of the 5th Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–26, Berlin, Germany, 1991.

[3] L. Champollion. Lexicalized non-local MCTAG with dominance links is NP-complete. In *Proceedings of Mathematics of Language 10*, Los Angeles, California, 2007. To appear.

[4] M. Garey and D. Johnson. *Computers and intractability.* W. H. Freeman & Co., New York, New York, 1979.

[5] A. K. Joshi and Y. Schabes. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69–124. Springer, Berlin, Germany, 1997.

[6] L. Kallmeyer. Tree-local multicomponent tree-adjoining grammars with shared nodes. *Computational Linguistics*, 31(2):187–225, 2005.

[7] T. Lichte. An MCTAG with tuples for coherent constructions in German. In *Proceedings of the 12th Conference on Formal Grammar*, Dublin, Ireland, 2007. To appear.

[8] A. Maclachlan and O. Rambow. Cross-serial dependencies in tagalog. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6)*, pages 100–104, Venice, Italy, 2002.

[9] J. Nivre. Constraints on non-projective dependency parsing. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 73–80, Trento, Italy, 2006.

[10] O. Rambow and G. Satta. Formal properties of non-locality. In *Proceedings of the Second International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+2)*, Philadelphia, Pennsylvania, 1992.

[11] S. Shieber. Direct parsing of ID/LP grammars. *Linguistics and Philosophy*, 7:135–154, 1984.

[12] S. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343, 1985.

[13] A. Søgaard. Polynomial charts for totally unordered languages. In *Proceedings of the 16th Nordic Conference of Computational Linguistics*, pages 183–190, Tartu, Estonia, 2007.

[14] K. Vijay-Shanker and D. Weir. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636, 1993.