TAG Parsing as Model Enumeration

Ralph Debusmann
Programming Systems Lab
Saarland University
Postfach 15 11 50
66041 Saarbrücken
Germany
rade@ps.uni-sb.de

Denys Duchier
Équipe Calligramme
LORIA – UMR 7503
Campus Scientifique, B. P. 239
54506 Vandœuvre lès Nancy CEDEX
France
duchier@loria.fr

Marco Kuhlmann
Programming Systems Lab
Saarland University
Postfach 15 11 50
66041 Saarbrücken
Germany
kuhlmann@ps.uni-sb.de

Stefan Thater
Computational Linguistics
Saarland University
Postfach 15 11 50
66041 Saarbrücken
Germany
stth@coli.uni-sb.de

Abstract

This paper introduces *well-ordered* derivation trees and makes use of this concept in a novel axiomatization of the TAG parsing problem as a constraint satisfaction problem. Contrary to prior approaches, our axiomatization focuses on the derivation trees rather than the derived trees. Well-ordered derivation trees are our primary models, whereas the derived trees serve solely to determine word order.

1 Introduction

Tree Adjoining Grammar (TAG) relates strings with two kinds of structures: derivation trees and corresponding derived trees. Derivation trees are more informative than their corresponding derived trees in the sense that the derived trees can be reconstructed from them. However, derivation trees are usually interpreted as unordered trees; they then cannot be used to formulate the TAG parsing problem directly, as they do not encode word order information.

This paper suggests to interpret derivation trees as *ordered* trees. It introduces the notion of *well-ordered* derivation trees: A derivation tree is called well-ordered if its nodes stand in the same precedence relation as the anchors in the corresponding derived tree. Because TAG can generate non-context-free languages, well-ordered derivation trees can be non-projective, i.e., they can contain "crossing" edges. The main contribution of this paper is an axiomatization of the exact form of non-projectivity licensed by TAG operations. It thereby provides a novel model-theoretic interpretation of the LTAG parsing problem.

The axiomatization of well-ordered derivation trees is put into practice in a description-based approach to TAG parsing, in which the parsing problem of strongly lexicalized TAGs[1] is interpreted as a *model enumeration problem*: given a description (a logical formula) ϕ of the input string, enumerate all and only those well-ordered derivation trees that are licensed by ϕ. Based on earlier work by Koller and Striegnitz (2002), we show that the solutions to this problem can in turn be characterised as the solutions of a constraint-satisfaction problem (CSP) on finite set integer variables, which can be solved by state-of-the-art constraint technology.

[1] A TAG is called strongly lexicalized, if each of its elementary trees contains exactly one anchor.

Our approach offers at least two interesting perspectives. First, it enables the encoding of LTAG grammars as certain *dependency grammars*, thereby illuminating the exact relation between the two formalisms. Second, the formulation of the LTAG parsing problem as a CSP opens up a large quantity of existing data to evaluate the constraint-based approach to parsing more thoroughly than what could be done before.

Plan of the paper. In §2, we show how the relation between TAG derivation trees and elementary trees can be formulated as the relation between models and logical descriptions, and introduce the notion of a TAG satisfiability problem. In §3, we extend satisfiability problems to parsing problems; we formalize the notion of well-ordered derivation trees as the structures under investigation in these problems, and show how their solutions can be obtained by solving a CSP. We illustrate this approach by means of an example in §4. §5 discusses the two perspectives of our approach mentioned above. Finally, §6 concludes the paper and presents ideas for future work.

2 TAG Satisfiability Problem

There are two major and constrasting formal perspectives on parsing: proof-theoretic and model-theoretic. The former emphasizes the construction of logical derivations, while the latter more directly states how models satisfy descriptions. The model-theoretic perspective (Cornell and Rogers, 1998) applied to a description-based specification of parsing problems is often more readily amenable to constraint-based processing. This is the view which we adopt for the remainder of this paper.

As a first step in our description-based approach to TAG parsing, we formulate the TAG *satisfiability problem*:

Given a multiset of elementary trees, can they be combined using operations of adjunction and substitution to form a complete derived tree?

To formally express TAG satisfiability problems, we introduce the following description language:

$$\phi ::= w : \tau \mid \phi \wedge \phi', \tag{1}$$

where w is taken from a set of *tree variables*, and τ from a set of TAG elementary trees of some grammar. We call $w : \tau$ a *tree literal*. We say that ϕ is *normal* if every tree variable in ϕ appears precisely in one tree literal.

It is well-known that the satisfiability problem is equivalent to the existence of a *derivation tree*, hence the idea to use derivation trees as models of normal ϕ descriptions. In order to make this more precise, we need some definitions:

Let Π be the set of finite paths, i.e. the finite sequences of positive integers. For simplicity in the following, we shall identify a node of a tree with the path $\pi \in \Pi$ that leads to it starting from the root of said tree.

A *derivation tree* is a tree (V, E) formed from vertices V and labeled edges $E \subseteq V \times V \times \Pi$. A model of a normal description $w_1 : \tau_1 \wedge \ldots \wedge w_n : \tau_n$ is a derivation tree where $V = \{w_1, \ldots, w_n\}$[2] and such that the following conditions are satisfied, where we write $w_1 - \pi \rightarrow w_2$ for an edge labeled with path π and representing either an adjunction or a substitution of τ_2 at node π of τ_1:

- If w_k is the root of the tree, then τ_k must be an initial tree.
- For each w_i, its outgoing edges must all be labeled with distinct paths, and for each substitution (resp. adjunction) node π in τ_i there must be exactly (resp. at most) one π-labeled edge.[3]
- For each edge $w_1 - \pi \rightarrow w_2$, if π is a substitution (resp. adjunction) node in τ_1, then τ_2 must be an initial (resp. auxiliary) tree.

In order to model lexical ambiguity, the description language can be extended with a limited form of disjunction, using for example the following extended language:

$$\phi ::= w_k : \{\tau_1^k, \ldots, \tau_{n_k}^k\} \mid \phi \wedge \phi',$$

where the set is to be interpreted disjunctively.

[2]Thus setting the interpretation of tree variables to be the identity substantially simplifies the presentation.

[3]For expositional simplicity, we do not cover adjunction constraints here. If an adjunction node is labeled with an adjunction constraint, then the exact well-formedness condition depends on that particular constraint.

The notion of TAG satisfiability problem as outlined above is implicit in (Koller and Striegnitz, 2002), who formulate the surface realization problem of natural language *generation* as the configuration problem of (unordered) dependency trees. A natural question is whether this treatment can be extended to parsing problems.

Given our formalization of TAG satisfiability problems, parsing problems cannot be expressed directly, as the models under consideration—derivation trees—are unordered trees. In order to express word order, a more natural class of models are derived trees, as these encode word order information in a direct way. However, the problem in using derived trees is that the formalization of the satisfaction relation becomes non-trivial, as the adjunction operation now requires a more complicated interpretation of elementary trees—not as atomic entities, but as groups of nodes that may get separated by material being "glued in between" by adjunction. If not conditioned carefully, this might lead to a formalism that is more expressive than TAG (Muskens, 2001; Rogers, 2003).

We suggest to solve the problem by considering derivation trees as being ordered. In the next section, we will introduce the notion of *well-ordered* derivation trees, which are possibly non-projective, ordered derivation trees whose precedence relation agrees with the precedence relation in the corresponding derived tree. This allows for an extension of the description language from (1) with precedence literals, which can be interpreted on (well-ordered) derivation trees in a straightforward way.

3 Well-ordered Derivation Trees

Our ambition is to tackle the parsing problem where word-order is part of the input specification. To this end, we formulate the TAG *parsing problem* analogously to our earlier definition of the TAG *satisfiability problem*:

Given an ***ordered*** *multiset of elementary trees, can they be combined using operations of adjunction and substitution to form a complete derived tree where the respective anchors are realized in the same order as stipulated by the input specification?*

To formally express the parsing problem, we extend our description language with precedence literals $w \prec w'$:

$$\phi \quad ::= \quad w : \tau \mid w \prec w' \mid \phi \wedge \phi' \tag{2}$$

where $w \prec w'$ means that w's anchor must precede w''s. For the same reasons as before, the approach that we will develop for this language will trivially extend to one with lexical ambiguity.

For the language of (1), the models where valid derivation trees. Now, however, we must additionally interpret the precedence literals of (2), which means that we need an order on the interpretations of the tree variables.

A derivation tree uniquely determines a derived tree and moreover uniquely determines a mapping I from each node π of each elementary tree w to its interpretation $I(w,\pi)$ as a node of the derived tree. The order that we are interested in is the one induced by the precedence between the interpretations of the anchors. More formally, writing $w_\diamond$ for the anchor node in w, we are interested in the order defined by:

$$w \prec w' \quad \equiv \quad I(w,w_\diamond) \prec I(w',w'_\diamond) \tag{3}$$

Thus, we arrive at the notion of a *well-ordered derivation tree*: a pair of a derivation tree and of the total order it induces on the elementary trees.

Unfortunately, we no longer have a simple way to enumerate these more complex models, unless we also construct the derived trees. Our contribution in this section is to show how the total order that we seek can also be obtained as the solution of a CSP.

3.1 Principles

To talk about order, we will need to talk about the set of anchors that are interpreted by nodes in the subtree (of the derived tree) rooted at $I(w,\pi)$. We write $\mathsf{yield}(w,\pi)$ for this set.

Assuming for the moment that we can freely use the notion of yield, we now show that the well-ordered derivation trees are precisely the valid derivation trees that satisfy the two principles of *convexity* and *order*:

Principle of convexity. The yield of the root of an elementary tree is convex. A set S is said to be convex with respect to a total order $\prec$ if for any $x \notin S$, x either precedes all elements of S or follows them all.

$$\begin{aligned} &\forall w,w' \in V: w' \notin \mathsf{yield}(w,\varepsilon) \Rightarrow \\ &\qquad w' \prec \mathsf{yield}(w,\varepsilon) \vee w' \succ \mathsf{yield}(w,\varepsilon) \end{aligned} \tag{4}$$

where we write $x \prec S$ as a shorthand for $\forall y \in S.\ x \prec y$.

Principle of order. If π_1 and π_2 are leaves in elementary tree w and $\pi_1 \prec \pi_2$, then also $\mathsf{yield}(w,\pi_1) \prec \mathsf{yield}(w,\pi_2)$, i.e. all anchors below π_1 precede all anchors below π_2.

$$\begin{aligned} &\forall w \in V: \forall \pi_1,\pi_2 \in \Pi: \\ &\quad \pi_1 \prec \pi_2 \wedge \pi_1 \in \mathsf{leaves}(w) \wedge \pi_2 \in \mathsf{leaves}(w) \Rightarrow \\ &\qquad \mathsf{yield}(w,\pi_1) \prec \mathsf{yield}(w,\pi_2) \end{aligned} \tag{5}$$

It is easy to show that these principles hold at every point of a TAG derivation. We now show that they suffice to completely determine the order among anchors. Consider the adjunction example of Figure 1. For brevity, we omit to say “the yield of”: by (5), we know that $\alpha_1 \prec \alpha_2 \prec \alpha_3$ and $\beta_1 \prec \beta_2$. The adjunction places α_2 in the yield of the foot node of β. Therefore, again by (5), we have $\beta_1 \prec \alpha_2 \prec \beta_2$. Now by (4) $\beta_1 \cup \alpha_2 \cup \beta_2$ is convex, therefore α_1 must either precede or follow it. Since $\alpha_1 \prec \alpha_2$, we must have $\alpha_1 \prec \beta_1$. Similarly for α_3.

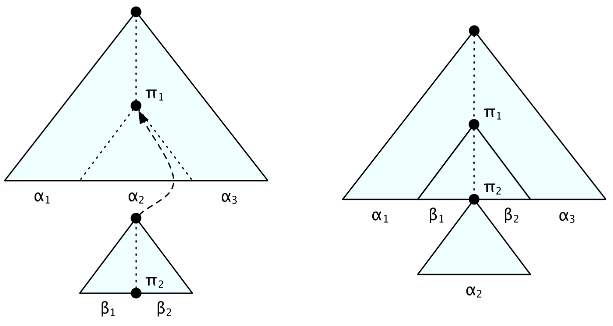


Figure 1: Adjunction

3.2 Axiomatization of Yield

Since we assume a strongly lexicalized version of TAG, each elementary tree has precisely one anchor. Therefore, for simplicity, we shall identify an anchor with the tree variable of the tree literal in which it appears. Thus $\mathsf{yield}(w,\pi)$ is a set of tree variables (standing for their respective anchors). We are going to show that $\mathsf{yield}(w,\pi)$ can also be obtained as the solution of a CSP. In order to do this, we will need to introduce additional functions.

$\mathsf{anchors}(w,\pi)$ is the set of anchors whose interpretations coincide with $I(w,\pi)$. Clearly:

$$\mathsf{anchors}(w,\pi) = \begin{cases} \{w\} & \text{if } \pi \text{ is anchor in } w \\ \emptyset & \text{otherwise} \end{cases} \tag{6}$$

$\mathsf{below}(w,\pi)$ is the set of anchors whose interpretations lie in the subtrees of the derived tree rooted at the interpretations of those nodes in w which are strictly dominated by the node π:

$$\mathsf{below}(w,\pi) = \{\mathsf{yield}(w,\pi') \mid \pi \lhd^+ \pi' \text{ in } w\} \tag{7}$$

$\mathsf{inserted}(w,\pi)$ concerns nodes where a substitution or adjunction has taken place. What is *inserted* is the yield of the tree which is being substituted or adjoined at node π of elementary tree w. We write $w - \pi \rightarrow w'$ for an edge in the derivation tree representing a substitution or an adjunction of w' at node π of w:

$$\mathsf{inserted}(w,\pi) = \begin{cases} \mathsf{yield}(w',\varepsilon) & \text{if } \exists w - \pi \rightarrow w' \\ \emptyset & \text{otherwise} \end{cases} \tag{8}$$

Finally, $\mathsf{pasted}(w,\pi)$ concerns foot nodes. When w is adjoined into w' at π', the subtrees hanging off π' in w' are cut out and pasted back under the foot node of w. Thus:

$$\mathsf{pasted}(w,\pi) = \begin{cases} \mathsf{below}(w',\pi') & \text{if } \pi \text{ is foot of } w, \\ & \text{and } \exists w' - \pi' \rightarrow w \\ \emptyset & \text{otherwise} \end{cases} \tag{9}$$

The yield can be obtained by taking the union of these quantities:

$$\mathsf{yield}(w,\pi) = \mathsf{anchors}(w,\pi) \cup \mathsf{inserted}(w,\pi) \cup \mathsf{pasted}(w,\pi) \cup \mathsf{below}(w,\pi) \quad (10)$$

The intuition for why this is correct can be outlined with an analysis by cases:

1. If π is anchor in w, then $\mathsf{anchors}(w,\pi) = \{w\}$ and all other quantities are empty sets.

2. If π is the foot node of w, then there must be an adjunction $w' - \pi' \rightarrow w$ and the anchors reachable from $I(w,\pi)$ are precisely those reachable from the material pasted at π as a result of the adjunction. anchors, inserted and below are all empty.

3. If a substitution has taken place at node π of w, then π is at least a leaf of w. The anchors reachable from $I(w,\pi)$ are precisely those reachable from the material that was inserted at (w,π). All other quantities are empty.

4. If an adjunction has taken place at node π of w, then at least π is not an anchor. The anchors reachable from $I(w,\pi)$ are now precisely those reachable from the material that was inserted at (w,π). Since $\mathsf{below}(w,\pi)$ is pasted at the foot node of the material that is being inserted, it ends up being included in $\mathsf{inserted}(w,\pi)$. anchors and pasted are empty.

5. If none of the above applies, then the anchors reachable from $I(w,\pi)$ are precisely those reachable from the children of π in w, i.e. from $\mathsf{below}(w,\pi)$. All other quantities are empty.

The definitions of (6,8,9) each contain a case analysis. In the definition of $\mathsf{anchors}(w,\pi)$ (6), the case condition is static: *is π the anchor of w or not?* Thus the satisfaction relation can be stated statically, and (6) can be interpreted as a constraint.

However, (8) and (9) both have conditions which dynamically depend on the existence of a substitution or adjunction dependency in the derivation tree. In order to arrive at a simple CSP, we need to slightly refine the formulation to avoid the case analysis. We take advantage of the fact that there is at most one adjunction (or substitution) at a given node:

$$\mathsf{inserted}(w,\pi) = \cup\{\mathsf{yield}(w',\varepsilon) \mid w - \pi \rightarrow w'\} \quad (11)$$

Let $\mathsf{children}(w,\pi) = \{w' \mid w - \pi \rightarrow w'\}$. (Duchier, 2003) showed how this equation could be reduced to a constraint and included in the CSP. Thus we obtain:

$$\mathsf{inserted}(w,\pi) = \cup\{\mathsf{yield}(w',\varepsilon) \mid w' \in \mathsf{children}(w,\pi)\} \quad (12)$$

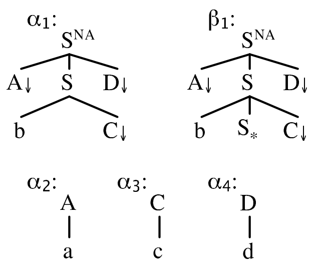
Again, as shown in (Duchier, 2003), this equation has precisely the form required for implementation by the *selection union constraint*. Similarly for pasted we obtain:

$$\mathsf{pasted}(w,\pi) = \begin{cases} \cup\{\mathsf{below}(w',\pi') \mid w \in \mathsf{parents}(w',\pi')\} & \\ & \text{if } \pi \text{ is foot in } w \\ \emptyset & \text{otherwise} \end{cases} \quad (13)$$

Given a (normal) TAG parsing problem, we are now able enumerate its models (the well-ordered derivation trees) as solutions of a CSP. First, the part of the CSP which enumerates the derivation trees remains as described by Koller and Striegnitz (2002). Second, for each node π of a tree w, we add the constraints (6,7,10,12,13): this allows us to obtain $\mathsf{yield}(w,\pi)$ as the solution of a CSP. Finally, we add the constraints corresponding to the principles of convexity and order, and the ordering constraints from the specification of the parsing problem. In this manner, we obtain a CSP which allows us to enumerate all and only the *well-ordered derivation trees*.

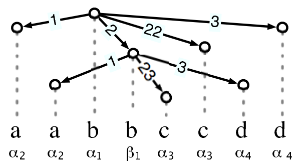
4 Example

We now show how our axiomatization of yield and the axiomatic principles derive the correct precedence constraints for a sample LTAG grammar. The grammar G that we are considering is the following:



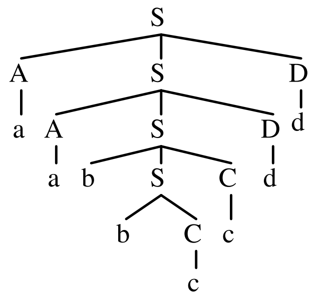
G produces the language $L = \{a^n b^n c^n d^n \mid n \geq 1\}$, a language not contained in the set of context-free languages.

Given G, the derivation tree for the string *aabbccdd* can be drawn as follows:



(Recall that a label π on an edge $w_1 - \pi \rightarrow w_2$ denotes the path address in w_1 at which the substitution or adjunction of w_2 has taken place.) In this tree, all edges except one correspond to substitutions; the edge from the left b to the right b corresponds to the adjunction of β_1 into α_1.

The given drawing of the derivation tree is well-ordered: The order of the anchors in it (connected to the nodes by vertical dashed lines) corresponds precisely to the order of the anchors in the corresponding derived tree:



To illustrate the axiomatization of yield, we give the yields of the elementary trees participating in the derivation of the string *aabbccdd* in Figure 2. Each table row pertains to a pair (w,π) of an elementary tree w (identified by its anchor) and a path address π in w, and shows the sets $\mathsf{anchors}(w,\pi)$, $\mathsf{inserted}(w,\pi)$, $\mathsf{pasted}(w,\pi)$ and $\mathsf{below}(w,\pi)$, whose union equals $\mathsf{yield}(w,\pi)$. With respect to the case analysis in the preceding section, each pair (row) corresponds to one of the cases:[4]

1. $(a_1,1)$, $(a_2,1)$, $(c_1,1)$ $(c_2,1)$, $(d_1,1)$ and $(d_2,1)$ correspond to the first case (anchor) since for all these elementary trees, 1 is the address of the anchor. The same holds for $(b_1,21)$ and $(b_2,22)$.

2. $(b_2,22)$ corresponds to the second case (foot node). This is the most interesting case, where the anchors below the adjunction site $(b_1,2)$ (i.e. b_1 and c_2) are “pasted” at the foot node $(b_2,22)$ of b_2.

3. $(b_1,1)$, $(b_1,22)$, $(b_1,3)$, $(b_2,1)$, $(b_2,23)$ and $(b_2,3)$ correspond to the third case (substitution), where $\mathsf{insert}(w,\pi)$ is the only non-empty set, containing the yields of the substituted trees.

4. $(b_1,2)$ corresponds to the fourth case (adjunction). This works like substitution.

5. The pairs (w,ε) correspond to the fifth case (else case) in the case analysis in the preceding section, where only the $\mathsf{below}(w,\pi)$ set is non-empty, containing the yields of the nodes below. The same holds for $(b_2,2)$.

5 Perspectives

The notion of well-ordered derivation trees offers some interesting perspectives. First, it allows us to encode each LTAG grammar into an equivalent dependency grammar. Second, the axiomatization of well-ordered derivation trees can be transformed into a constraint-based parser for LTAG in a straightforward way.

[4]In order to distinguish several occurrences of letters from each other, we have indexed them.

5.1 TAG and Dependency Grammar

If we ignore word order, derivation trees have a natural reading as dependency trees: anchors of elementary trees correspond to lexical entries, substitution and adjunction edges mirror the lexical entry’s valency.

Koller and Striegnitz (2002) develop this insight and formulate the surface realization problem of natural language generation as a parsing problem in a dependency grammar with free word order. In their approach, the dependency grammar lexicon is induced by “reading off” the valencies of elementary trees: substitution sites are encoded as obligatory valencies, adjunction sites as valencies that can be filled arbitrarily often.[5] This encoding embeds TAG into dependency grammar in that well-formed dependency trees directly correspond to TAG derivation trees and, indirectly, derived trees. However, the embedding is weak in the sense that its correctness of the encoding relies upon the fact that word order cannot be specified in the grammar; thus, the encoding cannot be applied to parsing problems.

The notion of well-ordered derivation trees allows us to extend the encoding to directly formulate parsing problems. To this end, we need to (i) specify the local (linear) order of the substitution valencies of each lexical entry, (ii) specify the local dominance relation among valencies and (iii) restrict the class of models to well-ordered derivation trees. Both the local order and the local dominance relations can be read off the LTAG elementary trees. The restriction of the class of models to well-ordered derivation trees then guarantees that the locally specified orderings are consistent with the global order in the dependency tree.

From other work on the interpretation of TAG as dependency grammar (Joshi and Rambow, 2003), this encoding is distinguished by three features:

- It does not stipulate any non-traditional rules to combine dependency structures, but only uses the standard “plugging” operation to fill valencies.
- It does not assume nodes in the dependency tree except for the nodes introduced by the (anchors of the) elementary trees.
- It is able to maintain the dependencies associated to a lexical anchor throughout the derivation. This is achieved by “hiding” the structure of the (partially) derived tree in the axiomatization of well-ordered derivation trees.

5.2 Constraint-based parsing of LTAGs

To solve the problem of surface realization as dependency parsing, Koller and Striegnitz (2002) successfully

[5]The encoding is based on a notion of derivation trees where different auxiliary trees can be adjoined at the same node.

(w,π)	anchors	inserted	pasted	below
$(a_1,1)$	$\{a_1\}$	$\emptyset$	$\emptyset$	$\emptyset$
$(a_2,1)$	$\{a_2\}$	$\emptyset$	$\emptyset$	$\emptyset$
$(c_1,1)$	$\{c_1\}$	$\emptyset$	$\emptyset$	$\emptyset$
$(c_2,1)$	$\{c_2\}$	$\emptyset$	$\emptyset$	$\emptyset$
$(d_1,1)$	$\{d_1\}$	$\emptyset$	$\emptyset$	$\emptyset$
$(d_2,1)$	$\{d_2\}$	$\emptyset$	$\emptyset$	$\emptyset$
$(b_1,21)$	$\{b_1\}$	$\emptyset$	$\emptyset$	$\emptyset$
$(b_2,21)$	$\{b_2\}$	$\emptyset$	$\emptyset$	$\emptyset$
$(b_2,22)$	$\emptyset$	$\emptyset$	$\{b_1,c_2\}$	$\emptyset$
$(b_1,1)$	$\emptyset$	$\{a_1\}$	$\emptyset$	$\emptyset$
$(b_1,22)$	$\emptyset$	$\{c_2\}$	$\emptyset$	$\emptyset$
$(b_1,3)$	$\emptyset$	$\{d_2\}$	$\emptyset$	$\emptyset$
$(b_2,1)$	$\emptyset$	$\{a_2\}$	$\emptyset$	$\emptyset$
$(b_2,23)$	$\emptyset$	$\{c_1\}$	$\emptyset$	$\emptyset$
$(b_2,3)$	$\emptyset$	$\{d_1\}$	$\emptyset$	$\emptyset$
$(b_1,2)$	$\emptyset$	$\{a_2,b_2,c_1,c_2,d_1\}$	$\emptyset$	$\emptyset$
(a_1,ε)	$\emptyset$	$\emptyset$	$\emptyset$	$\{a_1\}$
(a_2,ε)	$\emptyset$	$\emptyset$	$\emptyset$	$\{a_2\}$
(c_1,ε)	$\emptyset$	$\emptyset$	$\emptyset$	$\{c_1\}$
(c_2,ε)	$\emptyset$	$\emptyset$	$\emptyset$	$\{c_2\}$
(d_1,ε)	$\emptyset$	$\emptyset$	$\emptyset$	$\{d_1\}$
(d_2,ε)	$\emptyset$	$\emptyset$	$\emptyset$	$\{d_2\}$
(b_1,ε)	$\emptyset$	$\emptyset$	$\emptyset$	$\{a_1,a_2,b_1,b_2,c_1,c_2,d_1,d_2\}$
(b_2,ε)	$\emptyset$	$\emptyset$	$\emptyset$	$\{a_2,b_2,c_1,c_2,d_1\}$
$(b_2,2)$	$\emptyset$	$\emptyset$	$\emptyset$	$\{b_1,b_2,c_1,c_2\}$

Figure 2: Yields in the analysis of string *aabbccdd*

employ an existing constraint-based parser for Topological Dependency Grammar (Duchier and Debusmann, 2001). In light of the fact that surface realization is an NP-complete problem, the efficiency of this parser is quite remarkable. One of the major questions for a description-based approach to LTAG *parsing* is, whether the benign computational properties of existing, derivation-based parsers for LTAG[6] can be exploited even in the constraint framework.

We have started work into this direction by implementing a prototypical constraint parser for LTAG, and investigating its properties. The implementation can be done in a straightforward way by transforming the axiomatization of well-ordered derivation trees that was given in Section 3 into a constraint satisfaction problem along the lines of Duchier (2003). The resulting parser is available as a module for the XDG system (Debusmann, 2003).

Preliminary evaluation of the parser using the XTAG grammar shows that it is not competitive with state-of-the-art TAG parsers (Sarkar, 2000) in terms of run-time; however, this measure is not the most significant one for an evaluation of the constraint-based approach anyway. More importantly, a closer look on the search spaces explored by the parser indicates that the inferences drawn from the axiomatic principles are not strong enough to rule out branches of the search that lead to only inconsistent assignments of the problem variables. Future work needs to closely investigate this issue; ideally, we would arrive at an implementation that enumerates all well-ordered derivation trees for a given input without failure.

[6]The parsing problem of LTAG can be decided in time $O(n^6)$.

One of the benefits of the constraint formulation of dependency parsing given in Duchier (2003) is that it provides a means of effectively dealing with disjunctive information, e.g. information introduced by lexical ambiguity. The idea is to make the common information in a set of possible lexical entries available to the constraint solver as soon as possible, without waiting for one entire lexical entry from the set to be selected. If e.g. all elementary trees still possible for a given word are of different shape, but have the same number of substitution and adjunction sites labeled with the same categories—i.e., have the same valencies—, the constraint solver can configure the derivation tree before it would need to commit to any specific candidate tree. The question of whether this technique can be applied to widen the bottleneck that lexical ambiguity constitutes for TAG parsing awaits further exploration. With the encoding presented here, and the large grammatical resources for LTAG that it makes available to the application of constraint parsing, we are at least in the position now to properly evaluate the effectiveness of the constraint-based treatment of constructive disjunction.

6 Conclusion

In this paper, we introduced the notion of well-ordered derivation trees. Using this notion, we presented an axiomatization of the TAG parsing problem with a natural interpretation as a constraint satisfaction problem. The main burden lay in the axiomatization of the yield, which captures the dynamic aspects of a TAG derivation in terms of declarative constraints.

Contrary to previous approaches, we have shifted the emphasis away from the derived trees to the derivation trees. From this perspective, the derivation tree is the crucial ingredient of a TAG analysis, whereas the derived tree serves solely to constrain word order. This focusing on the derivation tree brings our approach in closer vicinity to dependency grammar.

Our approach yields two new avenues for future research. The first is to encode LTAG grammars into equivalent dependency grammars, and to intensify research on the relationship between TAG and DG. Second, the axiomatization of well-ordered derivation trees can be straightforwardly transformed into a constraint-based parser for LTAG. Koller and Striegnitz (2002) have shown that a similar approach can yield interesting results for generation, but we have not yet been able to reproduce them for parsing. To this end, we are moving towards the abstract notion of a *configuration problem* encompassing the constraint-based processing of both TAG, DG, and related frameworks, even semantic ones. We think that this abstraction eases the search for efficiency criteria for solving particular configuration problems, and can thus help us to pin down ways how to do efficient constraint-based TAG parsing in particular.

References

Thomas Cornell and James Rogers. 1998. Model theoretic syntax.

Ralph Debusmann. 2003. A parser system for extensible dependency grammar. In Denys Duchier, editor, *Prospects and Advances in the Syntax/Semantics Interface*, Lorraine-Saarland Workshop Series, pages 103–106. LORIA, Nancy/FRA.

Denys Duchier and Ralph Debusmann. 2001. Topological dependency trees: A constraint-based account of linear precedence. In *Proceedings of the 39th ACL*, Toulouse, France.

Denys Duchier. 2003. Configuration of labeled trees under lexicalized constraints and principles. *Journal of Research on Language and Computation*, 1(3/4).

Aravind Joshi and Owen Rambow. 2003. A formalism for dependency grammar based on tree adjoining grammar. In *Proc. of the First International Conference on Meaning-Text Theory*, pages 207–216, Paris, June.

Alexander Koller and Kristina Striegnitz. 2002. Generation as dependency parsing. In *Proceedings of the 40th ACL*, Philadelphia.

Reinhard Muskens. 2001. Talking about Trees and Truth-conditions. *Journal of Logic, Language and Information*, 10(4):417–455.

James Rogers. 2003. Syntactic structures as multi-dimensional trees. *Journal of Research on Language and Computation*, 1(3/4).

Anoop Sarkar. 2000. Practical experiments in parsing using tree adjoining grammars. In *Proceedings of the Fifth Workshop on Tree Adjoining Grammars, TAG+ 5*, Paris.