A Proof System for Tree Adjoining Grammars

Adi Palm
University of Passau

1. Introduction

Many TAG-based systems employ a particular tree adjoining grammar to generate the intended structures of the set of sentences they aim to describe. However, in most cases, the underlying set of elementary trees is more or less hand-made or maybe derived from a given tree data-base. We present a formal framework that allow to specify tree adjoining grammars by logical formulae. Based on this formalism we can check whether a given specification is TAG-consistent or whether a given TAG meets some particular properties. In addition, we sketch a method that generates a TAG from a given logical specification. As formal foundation, we employ a particular version of modal hybrid logic to specify the properties of T/D-trees. Such trees structurally combine a derived TAG-tree T and its associated derivation tree D. Finally, we sketch a labeled tableau calculus that constructs a set of tree automata representing the elementary trees of the specified TAG and a special tree automaton for the corresponding derivation trees.

In literature, we find some approaches specifying TAGs, or more generally, mildly context-sensitive grammar formalisms, that gradually vary in their underlying framework. Commonly, either starts with a logical description of recognizable sets of trees (Thatcher and Wright, 1968). However, they differ in their method of leaving the context-free paradigm. The approach mentioned in (Morawietz and Mönnich, 2001) and (Michaelis, Mönnich and Morawietz, 2000) uses a 'lifting' function that encodes a TAG into a regular tree grammar. In (Rogers, 1999) (and related works) we find a logical description of TAGs that is based on a 3-dimensional view of trees. The important issue of this approach is to combine the derived TAG-tree and its derivation tree to a single 3-dimensional structure.

Similarly, we also consider the derived TAG-tree and its derivation tree employ so-called T/D-trees. However we only associate the nodes of the derived tree with the corresponding node in the derivation tree. Consequently, all nodes of the same instance of an elementary tree refer to the same corresponding node in the derived tree. Therefore, we can specify structural properties of the derived TAG-tree and of the derivation tree at the same time. Using the links to the derivation tree, we can identify nodes in the TAG tree that belong to the same instance of some elementary tree. In contrast to the other approaches mentioned above which encode the TAG-tree into other kind of structures, we keep the original derived TAG tree as a structural unit. Consequently, we can directly access the nodes and the structural properties of the TAG tree without employing a particular projection function or any other special coding issues.

In essence, our formalism employs modal hybrid logic that combines the simplicity of modal logic and the expressivity of classical logic. The use of so-called nominals in hybrid logic offer explicit references to certain tree nodes which is (directly) possible in modal approaches. We introduce the hybrid language HL_{TAG} that specifies properties of the combined structure of derived TAG-trees and their derivation trees. Using this language we specify a number of TAG axioms which establish a notion of TAG-consistency. Further, we briefly illustrate a formalism that constructs a number of tree automata representing the underlying TAG for a given TAG-consistent HL_{TAG} formula.

2. A Hybrid Language for TAGs and their Derivations

Our formalism considers pairs of trees called T/D-trees as introduced in (Palm, 2000) where T represents a derived TAG-tree and D denotes the corresponding derivation tree. In general, a derived TAG tree $T = (t, V_t)$ is made up of a k_t-tree domain $t \subseteq \{1, \ldots, k_t\}^*$ for $k_t > 0$ and a labeling function $V_t \colon t \to \mathit{Pow}(\mathcal{P}_t)$ decorating tree nodes with a set of propositions of $\mathcal{P}_t$. The set of propositions $V_t(n)$ of some node n may be viewed as the label of n. Likewise, a derivation tree $D = (d, V_d)$ is made up of a k_d-tree domain $d \subseteq \{1, \ldots, k_d\}^*$ for some $k_d > 0$ and a labeling function $V_d \colon d \to \mathit{Pow}(\mathcal{P}_d)$. In addition, each T/D-tree includes the total linking function $\tau : t \to d$ that associates each node in the derived TAG tree T with the corresponding instance of its elementary tree in the derivation tree D.

* An extended version can be found at `http://www.phil.uni-passau.de/linguistik/palm/papers/`

 Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6), pp. 137–144. Universitá di Venezia.

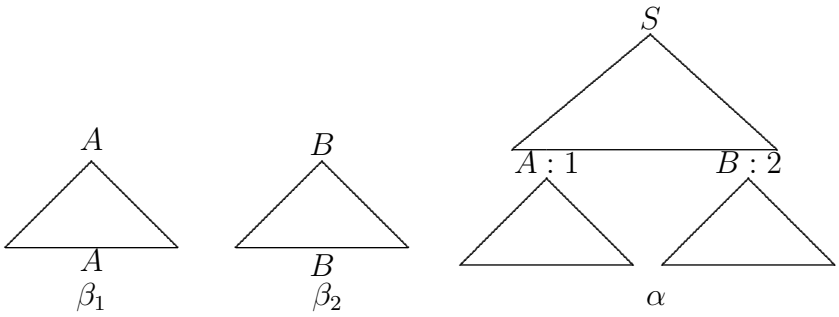

Figure 1: Sample TAG with the initial tree α and two auxiliary trees β_1 and β

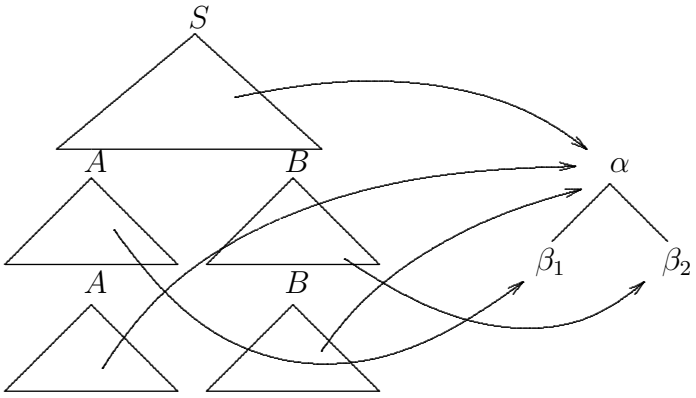

Figure 2: Resulting T/D-tree after adjoining β_1 and β_2 in α

The correspondence between either trees works straightforwardly. By the tree T we represented the derived TAG tree which results from an initial tree after adjoining and substituting auxiliary trees. By the derivation tree D we graphically represent these operations. Each children position of some node n in D represents a certain place of adjunction (or substitution) in the elementary tree represented by n. For instance, in Figure 1, the elementary tree α includes two nodes where we can adjoin another tree; we uniquely associate these nodes with the numbers 1 and 2, respectively . Now if we adjoin β_2 at the second node, this instance of β_2 in the derivation tree becomes the second child $n.2$ of the node n representing the corresponding instance α. Once we adjoined the tree β_1 at the first position, n obtains its first child $n.1$ representing this instance of β_1. Obviously, we associated each node of the corresponding instances of α, β_1 and β_2 in the derived TAG-tree with the nodes n, $n.1$ and $n.2$ in the derivation tree, respectively. Figure 2 shows the resulting T/D-tree. Note that for our formalism we assume that we can only adjoin at the inner nodes of an elementary tree, i.e. there is no adjunction at the root or at some leaves. This restriction ensures that the parent of the root and the children of the foot are nodes of the tree at which the adjunction took place.

For the formal foundation of our TAG-specification language we employ hybrid modal logic HL (Blackburn and Tzakova, 1998), (Blackburn and Tzakova, 1999), (Blackburn, 2000a; Blackburn, 2000b). This formalism extends modal (or temporal) logic with particular propositions called nominals which enable references to particular nodes (or terms) in a model. Further, there is an implemented tableau-based prover (Blackburn, Burchard and Walter, 2001) which is partially based on (Tzakova, 1999). Compared with classical logic we prefer modal and hybrid approaches since they allow more compact proofs and specifications.

In essence, we employ a modal logic on trees where the reflexive dominance relation denotes the modal reachability relation. We enhance this language by the next operator $\bigcirc_r$ referring to the r-th child of a node, by the link operator $\heartsuit$ referring to the associated node in the derivation tree. For the hybrid formulae we include the jump operator $i{:}\,\varphi$ and nominal propositions i with $i \in \mathcal{N}om$ where $\mathcal{N}om$ is an enumerable set of nominal symbols. Further, the language depends on the finite sets of constant propositions $\mathcal{P}_T$ and $\mathcal{P}_D$ and on the set of nominal

symbols $\mathcal{N}om$. Altogether, we obtain the hybrid language $HL_{TAG}(\mathcal{P}_t, \mathcal{P}_d, \mathcal{N}om)$ which is defined as:

$$\varphi \ ::= \ p \,|\, i \,|\, i{:}\varphi \,|\, \neg\varphi \,|\, \varphi \wedge \varphi \,|\, \bigcirc_r \varphi \,|\, \Diamond\varphi \,|\, \heartsuit\varphi$$

where $1 \leq r \leq k$ (with $k = max\{k_t, k_d\}$), $p \in \mathcal{P}_t \cup \mathcal{P}_d$ denotes a propositional constant and $i \in \mathcal{N}_{T/D}$ a nominal. Further, we can define the operators $\vee$, $\rightarrow$, $\leftrightarrow$ and $\Box$ in the standard way. In addition, we define the *next*-operator referring to some child by $\bigcirc\varphi \equiv \bigcirc_1\varphi \vee \ldots \vee \bigcirc_k\varphi$ and its dual universal counterpart by $\circledast\varphi \equiv \neg\bigcirc\neg\varphi$.

For the semantics of hybrid logic, we consider, in general, Kripke-structures which are, for the case of HL_{TAG}, T/D-trees. Besides the structural information a T/D-tree associates each tree node of either tree with sets of constant propositions from $\mathcal{P}_T$ and $\mathcal{P}_D$, respectively. In addition, we require a nominal denotation function $g{:}\mathcal{N}om \rightarrow (t \cup d)$ evaluating the nominals. We interpret a given $HL_{TAG}(\mathcal{P}_t, \mathcal{P}_d, \mathcal{N}om)$ formula φ at some node $n \in \cup d$ of a tree T/D for a nominal denotation function $g{:}\mathcal{N}om \rightarrow t \cup d$ where g is only necessary for formulae including nominals. For the node n we assume that we know whether it is a member of t or d.

$$\begin{array}{lll}
T/D, n \models p & \text{iff} & n \in V_t(p) \cup V_d(p), \text{ for } p \in \mathcal{P}_T \cup \mathcal{P}_D \\
T/D, n \models \neg\varphi & \text{iff} & \mathcal{T}/D, n \not\models \varphi \\
T/D, n \models \varphi \wedge \psi & \text{iff} & T/D, n \models \varphi \text{ and} \\
 & & T/D, n \models \psi \\
T/D, n \models \bigcirc_r\varphi & \text{iff} & T/D, n.r \models \varphi, 1 \leq r \leq k \text{ where } k = max\{k_t, k_d\} \\
T/D, n \models \Diamond\varphi & \text{iff} & T/D, n.a \models \varphi \text{ for some } a \in \{1, \ldots, k\}^* \text{ where } k = max\{k_t, k_d\} \\
T/D, n \models \heartsuit\varphi & \text{iff} & T/D, \tau(n) \models \varphi
\end{array}$$

A T/D-tree satisfies the formula φ if φ holds for the root of T. The link operator $\heartsuit$ is self-dual, i.e. $\heartsuit\varphi \equiv \neg\heartsuit\neg\varphi$. For the nominal expressions, we define the semantics as follows:

$$\begin{array}{lll}
T/D, n, g \models i & \text{iff} & g(n) = i \\
T/D, n, g \models i{:}\varphi & \text{iff} & T/D, n', g \models \varphi \\
 & & \text{and } g(n') = i
\end{array}$$

A nominal uniquely denotes a certain tree node where we do not explicitly distinguish the elements of T and D. The statement i is true if and only if the nominal i denotes the node under consideration. In contrast, in $i{:}\varphi$ we refer to the node denoted by i which does not depend on the node considered currently. We say a T/D-tree T/D satisfies the (nominal) formula φ at the node $n \in t \cup d$, written $T/D, n \models \varphi$, if there is a nominal denotation $g{:}\mathcal{N}om \rightarrow (t \cup d)$ such that $T/D, n, g \models \varphi$ is true. Similarly, T/D satisfies φ, written $T/D \models \varphi$ if there is a nominal denotation g such that $T/D, \mathit{root}_t, g \models \varphi$ where root_t denotes the root of the derived TAG tree T. Hence $T/D \models \varphi \wedge \Box\varphi$ states that φ must apply to all nodes of the derived TAG tree, $T/D \models \heartsuit\varphi$ states that φ applies to the root of the derivation tree and $T/D \models \heartsuit(\varphi \wedge \Box\varphi)$ states that φ applies to all nodes of the derivation tree. Finally, a HL_{TAG} formula φ is satisfiable if and only if there is a $T/D - tree$ and a nominal denotation g such that φ satisfies T/D by g.

Note that employing nominal propositions increases the expressivity of the former language. For instance, we can define the until-operator "until φ is true ψ must apply" or the unique existence operator $\Diamond_1\varphi$ which are not expressible in ordinary modal logic (Blackburn and Tzakova, 1999).

$$\begin{array}{rcl}
\mathit{until}(\varphi, \psi) & \equiv & \Diamond(\varphi \wedge i) \wedge \Box(\Diamond i \rightarrow \psi) \\
\Diamond_1\varphi & \equiv & \Diamond(i \wedge \varphi) \wedge \Box(\varphi \rightarrow i)
\end{array}$$

In the first case we search a descendant node that satisfies φ and mark this node by the nominal i. Then each descendant node that dominates i is an intermediate node that must satisfy ψ. Similarly, we specify the unique existence operator. Again we search a descendant node that satisfies φ and employ the nominal i in order to identify this node. Now all descendants that meet φ must also meet i. In general, by introducing nominal propositions, we can extend the expressivity of the underlying formalism. As shown in (Blackburn and Seligman, 1995; Blackburn and Seligman, 1997) hybrid logic is stronger than propositional modal logic. For instance, we can formulate the

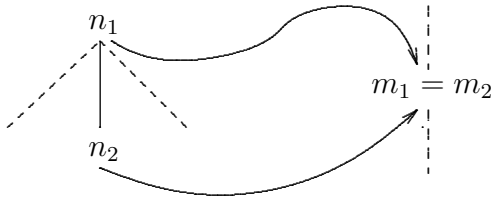

Figure 3: T/D-tree: n_1 and n_2 are internal nodes of the same elementary represented by m_1 and m_2.

until operator, or by $i \wedge \Box\neg i$ we can demand that the underlying modal reachability relation is irreflexive. Either of these properties fails to be expressible by means of propositional modal logic. On the other hand, we can specify the standard translation from hybrid logic to classical first-order logic. Therefore hybrid logic cannot be stronger than first-order logic. Moreover, as shown in (Schlingloff, 1992; Palm, 1997) the expressive power of the first-order logic for trees and the temporal logic for trees is identical. Since we can formulate the until-operator by means of hybrid logic, we obviously reach the expressivity of the temporal logic and the first logic on trees. However, the more crucial aspect of our formalism is the link operator $\heartsuit$ which allows to identify particular sets of tree nodes in the derived tree by referring to the same node in the derivation tree. Consequently, HL_{TAG} describes first-order definable sets of derivation trees; the expressivity for the derived tree obviously depends on the properties of linking function τ. Next we discuss some restrictions on τ leading to tree adjoining grammars.

3. TAG Axioms for HL_{TAG}

Obviously, by the language HL_{TAG} we can describe derived TAG trees and their corresponding derivation trees in an appropriate manner. However, so far it is unclear, what the necessary properties of a T/D tree are in order to describe valid TAG-trees and their derivations. Likewise, we want to know whether a given HL_{TAG} formula φ is TAG-satisfiable, i.e. whether the set of T/D satisfying φ represents a certain TAG. The answer to either question is the set of TAG axioms for the language HL_{TAG}. Hence, a T/D-tree would be TAG generated if and only if it meets these axioms, and a HL_{TAG}-formula φ is TAG-satisfiable if and only if it is consistent with these axioms, i.e. φ and the axioms are satisfiable.

Before we turn to the axioms in detail, we examine the construction and the structural properties of a T/D-tree by a given TAG derivation. For simplification purposes we put some restriction on the kind of TAGs considered here. At first, we restrict our formalism to the adjunction operation and ignore substitution. Nevertheless it is possible to simulate a substitution by an adjunction. Further, we assume that nodes, where adjunction is possible, are marked by the special auxiliary proposition *adj* and, correspondingly, all non-adjunction nodes must fail *adj*. Moreover, an adjunction node must be an inner node of an elementary tree, i.e. it cannot be the root or some leaf. As a consequence, we obtain only TAG trees where an adjoined tree is completely surrounded by the elementary tree it was adjoined to. This leads to the following lemma:

Lemma 3.1
Let $T/D = \langle (t, v_t), (d, V_d), \tau \rangle$ *be a TAG-generated T/D-tree and* $n_1, n_2 \in t$, $m_1, m_2 \in d$ *with* $m_1 = \tau(n_1)$, $m_2 = \tau(n_2)$ *and* $n_2 = n_1.r$ *for some* $1 \leq r \leq k_t$. *Then exactly one of the following cases must be true:*

1. $m_1 = m_2$
2. $m_1.s = m_2$, *for some* $1 \leq s \leq k_d$
3. $m_1 = m_2.s$, *for some* $1 \leq s \leq k_d$

□

This lemma considers the properties of a pair of immediately dominating nodes n_1 and n_2 in the derived TAG tree. In the first case, both nodes belong to the same instance of an elementary tree. Therefore, they are linked to the same node in the derivation tree, as illustrated in Figure 3. The second case n_2 is the root of an adjoined tree. By the assumption we made above, the parent of a root node must be a node of the tree where the adjunction took place. Therefore n_1 must be linked with the parent of the derivation tree node that is linked with n_2, see Figure 4.

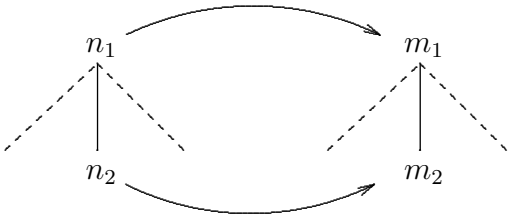

Figure 4: T/D-tree: n_2 is the root node of the elementary represented by m_2

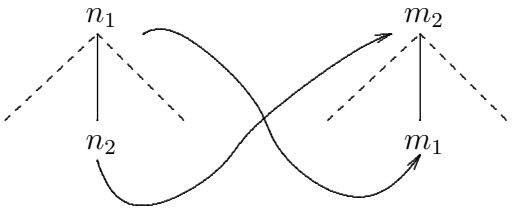

Figure 5: T/D-tree: n_1 is the foot node of the adjoined elementary tree represented by m_1

In the third case, n_1 is the foot node of an adjoined tree and, by assumption, each of its children must be a node of the tree where the adjunction took place. Consequently, m_1 must be a child of m_2, see Figure 5. Finally, due to above assumptions, no other case is possible.

Now we turn to the TAG axioms of HL_{TAG} which ensure that a given formula describes a TAG. For the general tree axioms we refer to the similar modal tree logic as presented for example in (Blackburn, Meyer-Viol and de Rijke, 1996). However, the more interesting issue are the TAG axioms. They should ensure that HL_{TAG} formulae only describe TAG-generated T/D-trees. For simplification, we introduce two auxiliary propositions *foot* and *root* that mark the corresponding nodes of an adjoined elementary tree. The TAG-axioms standing below assert the correct distribution of the auxiliary propositions *root* and *foot* and the correct linking between the derived and the derivation tree.

(D1)	$T_{root}\colon \mathit{root} \wedge \heartsuit D_{root}$	(associating the root nodes)
(D2)	$(i \wedge \mathit{root} \wedge \heartsuit k \wedge j\colon(\mathit{root} \wedge \heartsuit k)) \rightarrow j\colon i$	(unique root)
(D3)	$(i \wedge \mathit{foot} \wedge \heartsuit k \wedge j\colon(\mathit{foot} \wedge \heartsuit k)) \rightarrow j\colon i$	(unique foot)
(D4)	$(i \wedge \heartsuit k \wedge j\colon(\mathit{root} \wedge \heartsuit k)) \rightarrow j\colon(i \vee \diamond i)$	(root domination)
(D5)	$\bigcirc_r \mathit{root} \leftrightarrow \bigcirc_r \heartsuit i \wedge \heartsuit \bigcirc i$	(link properties of the root)
(D6)	$\mathit{foot} \leftrightarrow \heartsuit i \wedge \circledast \heartsuit \bigcirc i$	(link properties of the root)
(D7)	$\neg \mathit{foot} \wedge \bigcirc_r \neg \mathit{root} \leftrightarrow \heartsuit i \wedge \bigcirc_r \heartsuit i$	(link properties of the inner nodes)

The first axiom asserts that in a t/d-tree T/D the underlying initial tree of the derive tree T is linked with the root of the derivation tree D. Actually, it is sufficient that (D1) only links the root node of the derived tree *root* with root of the derivation tree. The correct linking of the remaining nodes of the initial tree follows from (D7). In order to access the root nodes of either tree, we assume two special nominal propositions T_{root} and D_{root} referring to the root nodes of T and D, respectively. The next two axioms (D2) and (D3) ensure that every instance of an elementary tree occurring in T/D has a unique root and a foot. We consider a root (or foot) node with the nominal i that is linked with derivation tree with the nominal k. Then every root (or foot) node that is linked with k must be identical to i. Moreover (D4) asserts that all nodes of the same instance of an elementary tree are dominated by the root node of this instance. Finally, the axiom (D5), (D6) and (D7) ensure the local structural properties mentioned in Lemma 3.1. By (D5), the r-th child of a node meets the proposition *root* if and only if the successor relationship also applies to the derivation tree nodes corresponding to them. By (D6), a node is a foot node if and only if it is linked to the node whose parent is associated with all children of the node considered. Finally, (D7) asserts that all pairs of immediately dominating nodes share the same instance of an elementary tree, if neither the upper one is its foot nor the lower one is its root.

Obviously, due to Lemma 3.1 and the properties of a TAG derivation, every T/D-tree that is generated by a given TAG must meet these axioms. Thus, these axioms are sound with respect to tree adjoining grammars. However, the opposite direction is less obvious. It states that every T/D-tree satisfying these axioms must be generated by a tree adjoining grammar. Next we describe a tree-extraction formalism that establishes this:

1. We arbitrarily select a leaf of the derivation tree with some nominal k and, further, we consider all nodes in the derived tree that are linked with k.

2. By the axioms (D2) and (D3) there must be a unique root and foot and by (D4) all nodes the are linked with k are weakly dominated by the the root. In addition, since k has no child, no other tree was adjoined. Therefore, due to (D5), (D6) and (D7) all nodes that are linked with k define a coherent tree section in the tree t.

3. We extract the tree section as identified previously, and we replace it by a single adjunction node that is linked with the parent of k.

4. We remove k in the derivation tree

5. We repeat the steps above until a single node in the derivation tree remains.

6. Due to (D1) the remaining structure defines the underlying initial tree of that TAG-tree. The trees we extracted above are the corresponding elementary trees.

This formalism illustrates how to construct a TAG for any given t/d-tree satisfying the axioms (D1) to (D7) such that the resulting TAG generates the given T/D-tree. In general, we obtain that a T/D-tree is TAG generated, if and only if it meets these axioms at every node of the derived tree.

Moreover this formalism can be extended t in the following way. So far we know that every HL_{TAG} formula that is consistent with the TAG axioms (D1) to (D7) specifies a set of trees where each member is generated by a certain TAG. We briefly sketch a method that constructs a corresponding TAG for a given TAG-consistent HL_{TAG} formula. Therefore, we combine the above extraction formalism with an ordinary method of constructing tree models, especially tree automata, from a given modal tree description. The desired result are two linked tree-automaton for the derived tree and the derivation tree. Instead of linking to certain tree nodes of the derivation tree, we employ links to the states of the corresponding tree automaton. Then we can apply a slightly modified version of the extraction method to these tree automata. Instead of extracting trees, this modified version considers subtree automata. The final result is a (finite) set of tree automata where each of them represents a set of initial trees. In addition, the resulting automaton for the derivation tree expresses possible adjunction operations for these automata.

In order to construct these automata, we can employ, for instance, well-known labeled tableau methods as described in (Goré, 1999), which were adopted for our purposes. The overall goal is to construct a set of simple tree automata representing the initial trees of the TAG described and another special tree automaton representing the corresponding derivation trees. Using labeled formulae in the tableau, we can indicate the node and the tree the formulae considered must apply to. A crucial part of the tableau system concerns the construction of the r^{th} successor of some node by the formula $\bigcirc_r\varphi$:

$$(\bigcirc_r)\ \frac{\alpha, \sigma :: \bigcirc_r\varphi}{\begin{array}{l|l|l} \alpha, \sigma :: \heartsuit i & \alpha, \sigma :: \heartsuit\bigcirc_s i & \alpha, \sigma :: \heartsuit i \\ \alpha, \sigma.r :: \varphi & \alpha.s, \sigma.r :: \varphi & \alpha', \sigma.r :: \varphi \\ \alpha, \sigma.r :: \heartsuit i & \alpha.s, \sigma.r :: \heartsuit i & \alpha', \sigma.r :: \heartsuit\bigcirc_s i \\ \alpha, \sigma :: \neg \mathit{foot} & \alpha, \sigma.r :: \mathit{root} & \alpha, \sigma :: \mathit{foot} \\ \alpha, \sigma.r :: \neg \mathit{root} & & \end{array}}$$

where $\alpha = \alpha'.s$. The premise of this rule considers the node σ of the derived tree that is associated with the node α of the derivation tree and $\bigcirc_r\varphi$ must hold at σ. Obviously this rule states again the situation described in Lemma 3.1. In the first case (see Figure 6), the new successor node $\sigma.r$ belongs to the same elementary tree as its parent, so both are associated with the same node α in the derivation tree. For the second case (see Figure 7), this rule generates a root node of an adjoined tree, so the new successor $\sigma.r$ is associated with $\alpha.s$, i.e. the corresponding successor. Note that we also construct the s-successor of α. Finally, in the third case (see Figure 8) σ denotes a foot node, so the successor $\sigma.r$ must be associated with the parent of α.

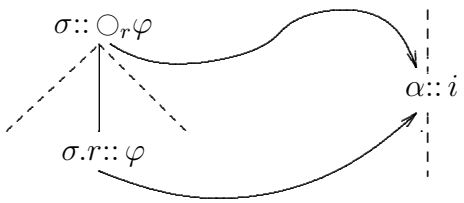

Figure 6: tableau rule for $\bigcirc_r \varphi$: inner nodes

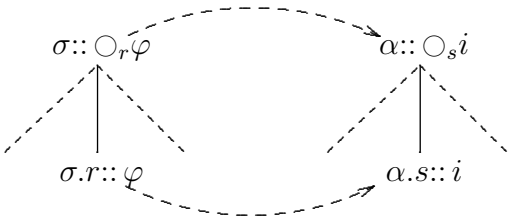

Figure 7: tableau rule for $\bigcirc_r \varphi$: root node

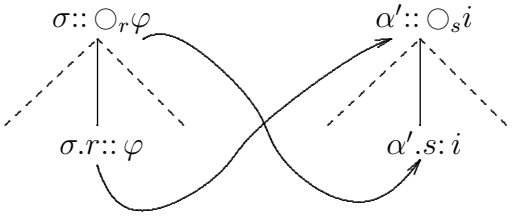

Figure 8: tableau rule for $\bigcirc_r \varphi$: foot node

Most other rules of this labeled tableau calculus are more or less straightforward or result from the requirements of the TAG axioms. To obtain tree automata from the tableau, we define classes of equivalent labels including the same set of formulae:

$$\alpha, \sigma \approx \alpha', \sigma' \text{ iff } \{\varphi \mid \alpha, \sigma :: \varphi\} = \{\varphi \mid \alpha', \sigma' :: \varphi\}$$

Obviously, the number of such equivalence classes is finite, since the number of occurring subformulae is finite as well. Every class defines some state of a tree automaton, and the immediate dominance relation leads to the state transition relation. Accordingly, we can extract the tree automaton for each elementary tree α by selecting the states including the label α. Then a gap in the immediate dominance relation indicates an adjunction node which must be handled correspondingly. Eventually, we also obtain a tree automaton for the derivation tree.

4. Conclusions

Specifying sets of trees beyond context-free grammars requires additional structural information. For the TAG-approach presented here, we combined the derived tree and the derivation tree leading to so-called T/D-trees. While the derived tree actually is the object of consideration, the derivation tree serves as a kind of storage for the required additional information. The linking function from the derived tree to the derivation tree combines the TAG tree nodes sharing the same instance of an elementary tree. Therefore we can access the underlying elementary trees and, consequently, we can decide whether a tree is TAG-generated.

For the formal description we employed hybrid logic which provides sufficient expressivity to specify TAG axioms and further constraints on TAGs. On the other hand, the modal foundation of hybrid logic, offers simple formulations that are easier to handle than classical logic. The result is a simple proof system for TAGs, which can be used to verify certain formal properties for a given TAG or, as we have sketched briefly, to construct a TAG from a given formal specification

An open question, so far, is the expressive power of our formalism. Obviously, it is possible to specify sets of T/D-trees that fall out of the scope of TAGs. For instance, it is possible specify complete binary trees. Nevertheless, such a specification would violate the TAG axioms.

References

Blackburn, P. 2000a. Internalizing Labelled Deduction. *Journal of Logic and Computation*, 10(1):137–168.

Blackburn, P. 2000b. Representation, Reasoning, and Relational Structures: a Hybrid Logic Manifesto. *Logic Journal of the IGPL*, 8(3):339–365.

Blackburn, P., A. Burchard and S. Walter. 2001. Hydra: a tableaux-based prover for basic hybrid logic. In C. Areces and M. de Rijke, editors, *Proceedings of Methods for Modalities 2*, Amsterdam, The Netherlands, November.

Blackburn, P., W. Meyer-Viol and M. de Rijke. 1996. A Proof System for Finite Trees. In H. Kleine Büning, editor, *Computer Science Logic*, LNCS, vol. 1092. Springer Verlag, pages 86–105.

Blackburn, P. and J. Seligman. 1995. Hybrid Languages. *Logic Journal of Logic, Language and Information*, 4(1):251–272.

Blackburn, P. and J. Seligman. 1997. What are Hybrid Languages. In M. Kracht, H. Wansing and M. Zakharyshev, editors, *Advances in Modal Logic '96*. CSLI Publications, Stanford University.

Blackburn, P. and M. Tzakova. 1998. Hybrid Completeness. *Logic Journal of the IGPL*, 6(4):625–650.

Blackburn, P. and M. Tzakova. 1999. Hybrid Languages and Temporal Logic. *Logic Journal of the IGPL*, 7(1):27–54.

Goré, R. 1999. Tableau Methods for Modal and Temporal Logic. In M. D'Augustino, D. Gabbay, R. Hähnle and J. Posegga, editors, *Handbook of Tableau Methods*. Kluwer, Dordrecht, pages 297–396.

Michaelis, J., U. Mönnich and F. Morawietz. 2000. Derivational Minimalism in Two Regular and Logical Steps. In *Proceedings of TAG+5, Paris*, pages 163–170.

Morawietz, F. and U. Mönnich. 2001. A Model-Theoretic Description of Tree Adjoining Grammars. In *Formal Grammar Conference/MOL Conference, Helsinki*, Electronical Notes in Theoretical Computer Science, vol. 53. Elsevier Science.

Palm, A. 1997. *Transforming Tree Constraints into Rules of Grammars*. DISKI, volume 173. St. Augustin: infix-Verlag.

Palm, A. 2000. Structure Sharing in Tree-Adjoining Grammars. In *Proceedings of TAG+5*.

Rogers, J. 1999. Generalized Tree-Adjoining Grammars. In *Proceedings of 6th Meeting on Mathemathics of Language (MOL6)*.

Schlingloff, B.-H. 1992. Expressive Completeness of Temporal Logic for Trees. *Journal of Applied Non-Classical Logics*, 2:157–180.

Thatcher, J.W. and J.B. Wright. 1968. Generalized Finite Automata Theory with an Application to Decision Porblems of Second-Order Logic. *Mathematical System Theory*, 2:57–81.

Tzakova, M. 1999. Tableaux calculi for hybrid logics. In N. Murray, editor, *Conference on Tableaux Calculi and Related Methods (TABLEAUX), Saratoga Springs, USA*, LNAI, vol. 1617. Springer Verlag, pages 278–292.