Efficiently Parsable Extensions to Tree-Local Multicomponent TAG

The Harvard community has made this article openly available. Please share how this access benefits you. Your story matters.

Citation

Published Version
doi:10.3115/1620754.1620768

Citable link
http://nrs.harvard.edu/urn-3:HUL.InstRepos:4133285

Terms of Use
This article was downloaded from Harvard University’s DASH repository, and is made available under the terms and conditions applicable to Open Access Policy Articles, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#OAP
Efficiently Parsable Extensions to Tree-Local Multicomponent TAG

Rebecca Nesson
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA
nesson@seas.harvard.edu

Stuart M. Shieber
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA
shieber@seas.harvard.edu

Abstract

Recent applications of Tree-Adjoining Grammar (TAG) to the domain of semantics as well as new attention to syntactic phenomena have given rise to increased interest in more expressive and complex multicomponent TAG formalisms (MCTAG). Although many constructions can be modeled using tree-local MCTAG (TL-MCTAG), certain applications require even more flexibility. In this paper we suggest a shift in focus from constraining locality and complexity through tree- and set-locality to constraining locality and complexity through restrictions on the derivational distance between trees in the same tree set in a valid derivation. We examine three formalisms, restricted NS-MCTAG, restricted Vector-TAG and delayed TL-MCTAG, that use notions of derivational distance to constrain locality and demonstrate how they permit additional expressivity beyond TL-MCTAG without increasing complexity to the level of set local MCTAG.

1 Introduction

Tree-Adjoining Grammar (TAG) has long been popular for natural language applications because of its ability to naturally capture syntactic relationships while also remaining efficient to process. More recent applications of TAG to the domain of semantics as well as new attention to syntactic phenomena such as scrambling have given rise to increased interest in multicomponent TAG formalisms (MCTAG), which extend the flexibility, and in some cases generative capacity of the formalism but also have substantial costs in terms of efficient processing. Much work in TAG semantics makes use of tree-local MCTAG (TL-MCTAG) to model phenomena such as quantifier scoping, Wh-question formation, and many other constructions (Kallmeyer and Romero, 2004; Romero et al., 2004). Certain applications, however, appear to require even more flexibility than is provided by TL-MCTAG. Scrambling is one well-known example (Rambow, 1994).

In addition, in the semantics domain, the use of a new TAG operation, flexible composition, is used to perform certain semantic operations that seemingly cannot be modeled with TL-MCTAG alone (Chiang and Scheffler, 2008) and in work in synchronous TAG semantics, constructions such as nested quantifiers require a set-local MCTAG (SL-MCTAG) analysis (Nesson and Shieber, 2006).

In this paper we suggest a shift in focus from constraining locality and complexity through restrictions that all trees in a tree set must adjoin within a single tree or tree set to constraining locality and complexity through restrictions on the derivational distance between trees in the same tree set in a valid derivation. We examine three formalisms, two of them introduced in this work for the first time, that use derivational distance to constrain locality and demonstrate by construction of parsers their relationship to TL-MCTAG in both expressivity and complexity. In Section 2 we give a very brief introduction to TAG. In Section 3 we elaborate further the distinction between these two types of locality restrictions using TAG derivation trees. Section 4 briefly addresses the simultaneity requirement present in MCTAG formalisms but not in Vector-
trees by substitution are called a base for derivations and may combine with other
minal node in an elementary tree also labeled with non-terv
notion entails splicing an auxiliary tree in at an internal
node which the root and a frontier node called the
stitution are combined by operations of
ementary tree structures of arbitrary depth which
vadjoining grammar consists of a set of elv
9 Background
beyond TL/T,(w
malisms and show that they permit analyses of some
we discuss the linguistic applications of these for
we recall the delayed TL/T,( formalism introv
duced by /hiang and Scheffler qøyy4r and define a
rvyvstyle parser for it as wellw In Section 4 we
duced by /hiang and Scheffler qøyy4r and define a
vstyle parser for themw In Section 3
novel formalismsu restricted nonvsimultaneous M/v
quirementw In Sections 1 and 2 we introduce two
T,( formalisms and argues for dropping the rev
weiru,
first introduced for use with contextvfree grammars
apply the /ockevKasamivYounger q/KYr algorithm
in /homsky normal form qKasamiu z5217 Youngeru
given in —igure zw —or further backgroundu refer to
ples of the adjunction and substitution operations are
Figure 1: An example of the TAG operations substitution and adjunction.

TAG formalisms and argues for dropping the re-
requirement. In Sections 5 and 6 we introduce two
ovel formalisms, restricted non-simultaneous MC-
TAG and restricted Vector-TAG, respectively, and
define CKY-style parsers for them. In Section 7
we recall the delayed TL-MCTAG formalism intro-
duced by Chiang and Scheffler (2008) and define a
CKY-style parser for it as well. In Section 8 we
explore the complexity of all three parsers and the
relationship between the formalisms. In Section 9
we discuss the linguistic applications of these for-
malisms and show that they permit analyses of some
of the hard cases that have led researchers to look
beyond TL-MCTAG.

2 Background
A tree-adjoining grammar consists of a set of el-
lementary tree structures of arbitrary depth, which
are combined by operations of adjunction and sub-
titution. Auxiliary trees are elementary trees in
which the root and a frontier node, called the foot
node and distinguished by the diacritic *, are labeled
with the same nonterminal A. The adjunction oper-
entails splicing an auxiliary tree in at an internal
node in an elementary tree also labeled with nonter-
inal A. Trees without a foot node, which serve as
a base for derivations and may combine with other
trees by substitution, are called initial trees. Exam-
les of the adjunction and substitution operations are
given in Figure 1. For further background, refer to
the survey by (Joshi and Schabes, 1997).

Shieber et al. (1995) and Vijay-Sanker (1987)
apply the Cocke-Kasami-Younger (CKY) algorithm
first introduced for use with context-free grammars
in Chomsky normal form (Kasami, 1965; Younger,
1967) to the TAG parsing problem to generate
parsers with a time complexity of $O(n^8|G|^2)$. In
order to clarify the presentation of our extended TL-
MCTAG parsers below, we briefly review the algo-
rithm of Shieber et al. (1995) using the inference
rule notation from that paper. As shown in Figure 2,
items in CKY-style TAG parsing consist of a node
in an elementary tree and the indices that mark the
edges of the span dominated by that node. Nodes,
notated $\alpha@\gamma^a$, are specified by three pieces of in-
formation: the identifier $\alpha$ of the elementary tree the
node is in, the Gorn address $\gamma$ of the node in that
tree\footnote{A Gorn address uniquely identifies a node within a tree. The Gorn address of the root node is $\varepsilon$. The $j$th child of the node with address $i$ has address $i-j$.}, and a diacritic, $\circ$, indicating that an adjunc-
tion or substitution is still available at that node or $\bullet$, indicating that one has already taken place.

Each item has four indices, indicating the left and
right edges of the span covered by the node as well
as any gap in the node that may be the result of a
foot node dominated by the node. Nodes that do
not dominate a foot node will have no gap in them,
which we indicate by the use of underscores in place
of the indices for the gap. To limit the number of in-
ference rules needed, we define the following func-
tion $i \cup j$ for combining indices:

\[
i \cup j = \begin{cases}
i & j = - \\
j & i = - \\
i & i = j \\
\text{undefined otherwise}
\end{cases}
\]

The side conditions $\text{Init}(\alpha)$ and $\text{Aux}(\alpha)$ hold if $\alpha$
is an initial tree or an auxiliary tree, respectively.
$\text{Label}(\alpha@\gamma^a)$ specifies the label of the node in tree
$\alpha$ at address $\gamma$. $\text{Flt}(\alpha)$ specifies the address of the
foot node of tree $\alpha$. $\text{Adj}(\alpha@\gamma^a, \beta)$ holds if tree $\beta$
may adjoin into tree $\alpha$ at address $\gamma$. $\text{Subst}(\alpha@\gamma^a, \beta)$
holds if tree $\beta$ may substitute into tree $\alpha$ at address
$\gamma$. These conditions fail if the adjunction or sub-
stitution is prevented by constraints such as mismatched
node labels.

Multi-component TAG (MCTAG) generalizes
TAG by allowing the elementary items to be sets
trees rather than single trees (Joshi and Schabes,
1997). The basic operations are the same but all
trees in a set must adjoin (or substitute) into another
tree or tree set in a single step in the derivation.

An MCTAG is tree-local if tree sets are required
to adjoin within a single elementary tree (Weir,
Goal Item: \( (\alpha @ \varepsilon^\bullet , 0, \ldots, n) \)  
Init(\( \alpha \)) 
Label(\( \alpha @ \varepsilon \)) = \( S \)

Terminal Axiom: \( (\alpha @ a^\bullet , i - 1, \ldots, \iota) \)  
Label(\( \alpha @ a \)) = \( w_i \)

Empty Axiom: \( (\alpha @ a^\bullet , i, \ldots, \iota) \)  
Label(\( \alpha @ a \)) = \( \varepsilon \)

Foot Axiom: \( (\alpha @ \text{Ft}(\alpha)^\circ , p, p, q, q) \)  
Aux(\( \alpha \))

Unary Complete: \( \frac{(\alpha @ (a \cdot 1)^\bullet , i, j, k, l)}{(\alpha @ a^\circ , i, j, k, l)} \)  
\( \alpha @ (a \cdot 2) \) undefined

Binary Complete: \( \left\{ \frac{(\alpha @ (a \cdot 1)^\bullet , i, j, k, l), (\alpha @ (a \cdot 2)^\bullet , l, j', k', m)}{(\alpha @ a^\circ , i, j \cup j', k \cup k', m)} \right\} \)

Adjoin: \( \frac{\beta @ \varepsilon^\bullet , i, p, q, l, (\alpha @ a^\circ , p, j, k, q)}{(\alpha @ a^\bullet , i, j, k, l)} \)  
Adj(\( \alpha @ a , \beta \))

No Adjoin: \( (\alpha @ a^\circ , i, j, k, l) \)

Substitute: \( \frac{\beta @ \varepsilon^\bullet , i, \ldots, \iota, l}{(\alpha @ a^\bullet , i, \ldots, \iota, l)} \)  
Subst(\( \alpha @ a , \beta \))

Figure 2: The CKY algorithm for TAG

1988). Although tree-local MCTAG (TL-MCTAG) has the same generative capacity as TAG (Weir, 1988), the conversion to TAG is exponential and the TL-MCTAG formalism is NP-hard to recognize (Søgaard et al., 2007). An MCTAG is set-local if tree sets required to adjoin within a single elementary tree set (Weir, 1988). Set-local MCTAG (SL-MCTAG) has equivalent expressivity to linear context-free rewriting systems and recognition is provably PSPACE complete (Nesson et al., 2008).

3 Domains of Locality and Derivation Trees

The domains of locality of TL-MCTAG and SL-MCTAG (and trivially, TAG) can be thought of as lexically defined. That is, all locations at which the adunction of one tree set into another may occur must be present within a single lexical item. However, we can also think of locality derivationally. In a derivationally local system the constraint is on the relationships allowed between members of the same tree set in the derivation tree.

TAG derivation trees provide the information about how the elementary structures of the grammar combine that is necessary to construct the derived tree. Nodes in a TAG derivation tree are labeled with identifiers of elementary structures. One elementary structure is the child of another in the derivation tree if it adjoins or substitutes into it in the derivation. Arcs in the derivation tree are labeled with the address in the target elementary structure at which the operation takes place.

In MCTAG the derivation trees are often drawn with identifiers of entire tree sets as the nodes of the tree because the lexical locality constraints require that each elementary tree set be the derivational child of only one other tree set. However, if we elaborate the derivation tree to include a node for each tree in the grammar rather than only for each tree set we can see a stark contrast in the derivational
locality of these two formalisms. In TL-MCTAG all trees in a set must adjoin simultaneously. In terms of well-formed derivation trees, this amounts to disallowing derivations in which a tree from a given set is the ancestor of a tree from the same tree set. For most linguistic applications of TAG, this requirement seems natural and is strictly obeyed. There are a few applications, including flexible composition and scrambling in free-word order languages that benefit from TAG-based grammars that drop the simultaneity requirement (Chiang and Scheffler, 2008; Rambow, 1994). From a complexity perspective, however, checking the simultaneity requirement is expensive (Kallmeyer, 2007). As a result, it can be advantageous to select a base formalism that does not require simultaneity even if the grammars implemented with it do not make use of that additional freedom.

5 Restricted Non-simultaneous MCTAG

The simplest version of a derivationally local TAG-based formalism is most similar to non-local MCTAG. There is no lexical locality requirement at all. In addition, we drop the simultaneity requirement. Thus the only constraint on elementary tree sets is the limit, \(d\), on the derivational distance between the trees in a given set and their nearest common ancestor. We call this formalism restricted non-simultaneous MCTAG. Note that if we constrain \(d\) to be one, this happens to enforce both the derivational delay limit and the lexical locality requirement of TL-MCTAG.

A CKY-style parser for restricted NS-MCTAG with a restriction of \(d\) is given in Figure 4. The items of this parser contain \(d\) lists, \(\Lambda^1, \ldots, \Lambda^d\), called histories that record the identities of the trees that have already adjoined in the derivation in order to enforce the locality constraints. The identities of the trees in a tree set that have adjoined in a given derivation are maintained in the histories until all the trees from that set have adjoined. Once the locality constraint is checked for a tree set, the Filter side condition expunges those trees from the histories. A tree is recorded in this history list with superscript \(i\), where \(i\) is the derivational distance between the location where the recorded tree adjoined and the location of the current item. The locality constraint is enforced at the point of adjunction or substitution where the
Goal Item
\[ \langle \alpha_0 @ \varepsilon^*, 0, \ldots, \emptyset \rangle \]

Terminal Axiom
\[ \langle \alpha_x @ a^*, i - 1, \ldots, i, \emptyset, \ldots, \emptyset \rangle \]

Empty Axiom
\[ \langle \alpha_x @ a^*, i, \ldots, i, \emptyset, \ldots, \emptyset \rangle \]

Foot Axiom
\[ \langle \alpha_x @ \text{ft}(\alpha_x)^0, p, q, \emptyset, \ldots, \emptyset \rangle \]

Unary Complete
\[ \frac{\langle \alpha_x @ (a \cdot 1)^*, i, j, k, l, \Lambda_1, \ldots, \Lambda^d \rangle}{\langle \alpha_x @ a^*, i, j, k, l, \Lambda_1, \ldots, \Lambda^d \rangle} \]

Binary Complete
\[ \frac{\langle \alpha_x @ (a \cdot 1)^*, i, j, k, l, \Lambda_1, \ldots, \Lambda^d \rangle}{\langle \alpha_x @ a^*, i, j, k, l, \Lambda_1, \ldots, \Lambda^d \rangle} \]

Adjoin:
\[ \frac{\langle \beta_y @ \varepsilon^*, i, p, q, l, \Lambda_1, \ldots, \Lambda_1^{d-1}, \emptyset \rangle}{\langle \alpha_x @ a^*, i, j, k, q, l, \Lambda_1, \ldots, \Lambda^d \rangle} \]

Substitute:
\[ \frac{\langle \beta_y @ \varepsilon^*, i, \ldots, l, \Lambda_1, \ldots, \Lambda_1^{d-1}, \emptyset \rangle}{\langle \alpha_x @ a^*, i, \ldots, l, \Lambda_1, \ldots, \Lambda^d \rangle} \]

No Adjoin:
\[ \frac{\langle \alpha_x @ a^*, i, j, k, l, \Lambda_1, \ldots, \Lambda^d \rangle}{\langle \alpha_x @ a^*, i, j, k, l, \Lambda_1, \ldots, \Lambda^d \rangle} \]

\[ \text{Init}(\alpha_1) \]
\[ \text{Label}(\alpha_0 @ \varepsilon) = S \]
\[ |\alpha| = 1 \]
\[ \text{Label}(\alpha_x @ a) = w_i \]
\[ \text{Label}(\alpha_x @ a) = \varepsilon \]
\[ \text{Aux}(\alpha_x) \]
\[ \alpha_x @ (a \cdot 2) \] undefined
\[ \text{Filter}(\Lambda_1^1 \cup \Lambda_2^1, \ldots, \Lambda_1^d = \Lambda_1^1, \ldots, \Lambda^d) \]
\[ \text{Adj}(\alpha_x @ a, \beta_y) \]
\[ \text{Filter}(\Lambda_2^1 \cup \{\beta_y\}, \Lambda_1^1, \ldots, \Lambda_1^1, \ldots, \Lambda_1^d = \Lambda_1^1, \ldots, \Lambda^d) \]
\[ \text{Subst}(\alpha_x @ a, \beta_y) \]
\[ \text{Filter}(|\beta_y|, \Lambda_1^1, \ldots, \Lambda_1^{d-1}) = \Lambda_1^1, \ldots, \Lambda^d \]

Figure 4: Axioms and inference rules for the CKY algorithm for restricted NS-MCTAG with a restriction of \( d \).

history at the limit of the permissible delay must be empty for the operation to succeed.

6 Restricted V-TAG

A Vector-TAG (V-TAG) (Rambow, 1994) is similar to an MCTAG in that the elementary structures are sets (or vectors) of TAG trees. A derivation in a V-TAG is defined as in TAG. There is no locality requirement or other restriction on adjunction except that if one tree from a vector is used in a derivation, all trees from that vector must be used in the derivation. The trees in a vector may be connected by dominance links between the foot nodes of auxiliary trees and any node in other trees in the vector. All adjunctions must respect the dominance relations in that a node \( \eta_1 \) that dominates a node \( \eta_2 \) must appear on the path from \( \eta_2 \) to the root of the derived tree. The definition of V-TAG is very similar to that of non-local MCTAG as defined by Weir (1988) except that in non-local MCTAG all trees from a tree set are required to adjoin simultaneously.

Restricted V-TAG constrains V-TAG in several ways. First, the dominance chain in each elementary tree vector is required to define a total order over the trees in the vector. This means there is a single base tree in each vector. Note also that all trees other than the base tree must be auxiliary trees in order to dominate other trees in the vector. The base tree may be either an initial tree or an auxiliary tree. Second, a restricted V-TAG has a restriction level, \( d \), that determines the largest derivational distance that may exists between the base tree and the highest tree in a tree vector in a derivation. Restricted V-TAG differs from restricted NS-MCTAG in one important respect: the dominance requirements of restricted V-TAG require that trees from the same
set must appear along a single path in the derived tree, whereas in restricted NS-MCTAG trees from the same set need not adhere to any dominance relationship in the derived tree.

A CKY-style parser for restricted V-TAG with restriction level \( d \) is given in Figure 5. Parsing is similar to delayed TL-MCTAG in that we have a set of histories for each restriction level. However, because of the total order over trees in a vector, the parser only needs to maintain the identity of the highest tree from a vector that has been used in the derivation along with its distance from the base tree from that vector. The Filter side condition accordingly expunges trees that are the top tree in the dominance chain of their tree vector. The side conditions for the Adjoin non-base rule enforce that the dominance constraints are satisfied and that the derivational distance from the base of a tree vector to its currently highest adjoined tree is maintained accurately. We note that in order to allow a non-total ordering of the trees in a vector we would simply have to record all trees in a tree vector in the histories as is done in the delayed TL-MCTAG parser.

7 Delayed TL-MCTAG

Chiang and Scheffler (2008) introduce the delayed TL-MCTAG formalism which makes use of a derivational distance restriction in a somewhat different way. Rather than restricting the absolute distance between the trees of a set and their nearest common ancestor, given a node \( \alpha \) in a derivation tree, delayed TL-MCTAG restricts the number of tree sets that are not fully dominated by \( \alpha \). Borrowing directly from Chiang and Scheffler (2008), Figure 7 gives two examples.

Parsing for delayed TL-MCTAG is not discussed by Chiang and Scheffler (2008) but can be accomplished using a similar CKY-style strategy as in the two parsers above. We present a parser in Figure 6. Rather than keeping histories that record derivational distance, we keep an active delay list for each item that records the delays that are active (by recording the identities of the trees that have joined) for the tree of which the current node is a part. At the root of each tree the active delay list is filtered using the Filter side condition to remove all tree sets that are fully dominated and the resulting list is checked using the Size to ensure that it contains no more than \( d \) distinct tree sets where \( d \) is the specified delay for the grammar. The active delays for a given tree are passed to its derivational parent when it adjoins or substitutes.

Delayed TL-MCTAG differs from both of the previous formalisms in that it places no constraint on the length of a delay. On the other hand while the previous formalisms allow unlimited short delays to be pending at the same time, in delayed TL-MCTAG, only a restricted number of delays may be active at once. Similar to restricted V-TAG, there is no simultaneity requirement, so a tree may have another tree from the same set as an ancestor.

8 Complexity

The complexity of the restricted NS-MCTAG and restricted V-TAG parsers presented above depends on the number of possible histories that may appear in an item. For each step of derivational distance permitted between trees of the same set, the corresponding history permits many more entries. History \( \Lambda^1 \) may contain trees that have adjoined into the same tree as the node of the current item. The number of entries is therefore limited by the number of adjunction sites in that tree, which is in turn limited by the number of nodes in that tree. We will call the maximum number of nodes in a tree in the grammar \( t \). Theoretically, any tree in the grammar could adjoin at any of these adjunction sites, meaning that the number of possible values for each entry in the history is bounded by the size of the grammar \( |G| \). Thus the size of \( \Lambda^1 \) is \( O(|G|^t) \). For \( \Lambda^2 \) the en-
tries correspond to tree that have adjoined into a tree
that has adjoined into the tree of the current item.
Thus, for each of the \( t \) trees that may have adjoined
at a derivational distance of one, there are \( t \) more
trees that may have adjoined at a derivational distance
of two. The size of \( \Lambda^2 \) is therefore \( |G|^t \). The
combined size of the histories for a grammar with a
delay or restriction of \( d \) is therefore \( O(|G|^{\sum_{i=1}^d t^i}) \).
Replacing the sum with its closed form solution, we
have \( O(|G|^{\frac{d+1}{i-1} - 1}) \) histories.

Using the reasoning about the size of the histories
given above, the restricted NS-MCTAG parser
presented here has a complexity of \( O(n^6 |G|^{\frac{d+1}{i-1} - 1}) \),
where \( t \) is as defined above and \( d \) is the limit on de-
lay of adjunction. For a tree-local MCTAG, the com-
plexity reduces to \( O(n^6 |G|^{2+i}) \). For the linguistic
applications that motivate this chapter no delay
greater than two is needed, resulting in a complexity
of \( O(n^6 |G|^{2+i}) \).

The complexity of the delayed TL-MCTAG
parser depends on the number of possible active de-
lay lists. As above, each delay list may have a maxi-
mum of \( t \) entries for trees that adjoin directly into it.
The restriction on the number of active delays means
that the active delay lists passed up from these child
nodes at the point of adjunction or substitution can
have size no more than \( d \). This results in an addi-
tional \( td(f-1) \) possible entries in the active de-

Figure 5: Inference rules for the CKY algorithm for restricted V-TAG with a restriction of \( d \). Item form, goal item and
axioms are omitted because they are identical to those in restricted NS-MCTAG parser.
lay list, giving a total number of active delay lists of $O(|G|^{1+d(f-1)})$. Thus the complexity of the parser is $O(n^6 |G|^{2+\epsilon(1+d(f-1))})$.

9 Conclusion

Each of the formalisms presented above extends the flexibility of MCTAG beyond that of TL-MCTAG while maintaining, as we have shown herein, complexity much less than that of SL-MCTAG. All three formalisms permit modeling of flexible composition (because they permit one member of a tree set to be a derivational ancestor of another tree in the same set), at least restricted NS-MCTAG and restricted V-TAG permit analyses of scrambling, and all three permit analyses of the various challenging semantic constructions mentioned in the introduction. We conclude that extending locality by constraining derivational distance may be an effective way to add flexibility to MCTAG without losing computational tractability.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. BCS-0827979.

References


