Multi-Component TAG and Notions of Formal Power

William Schuler, David Chiang
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
{schuler,dchiang}@linc.cis.upenn.edu

Mark Dras
Inst. for Research in Cognitive Science
University of Pennsylvania
Suite 400A, 3401 Walnut Street
Philadelphia, PA 19104-6228
madras@linc.cis.upenn.edu

Abstract

This paper presents a restricted version of Set-Local Multi-Component TAGs (Weir, 1988) which retains the strong generative capacity of Tree-Local Multi-Component TAG (i.e. produces the same derived structures) but has a greater derivational generative capacity (i.e. can derive those structures in more ways). This formalism is then applied as a framework for integrating dependency and constituency based linguistic representations.

1 Introduction

An aim of one strand of research in generative grammar is to find a formalism that has a restricted descriptive capacity sufficient to describe natural language, but no more powerful than necessary, so that the reasons some constructions are not legal in any natural language is explained by the formalism rather than stipulations in the linguistic theory. Several mildly context-sensitive grammar formalisms, all characterizing the same string languages, are currently possible candidates for adequately describing natural language; however, they differ in their capacities to assign appropriate linguistic structural descriptions to these string languages. The work in this paper is in the vein of other work (Joshi, 2000) in extracting as much structural descriptive power given a fixed ability to describe strings, and uses this to model dependency as well as constituency correctly.

One way to characterize a formalism's descriptive power is by the the set of string languages it can generate, called its *weak generative capacity*. For example, Tree Adjoining Grammars (TAGs) (Joshi et al., 1975) can generate the language $a^n b^n c^n d^n$ and Context-Free Grammars (CFGs) cannot (Joshi, 1985).

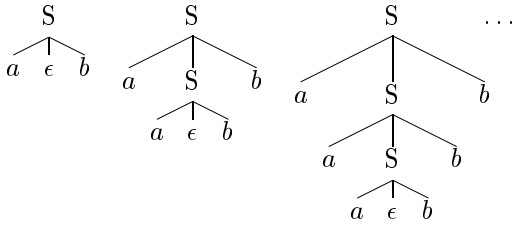

Figure 1: CFG-generable tree set for $a^n b^n$.

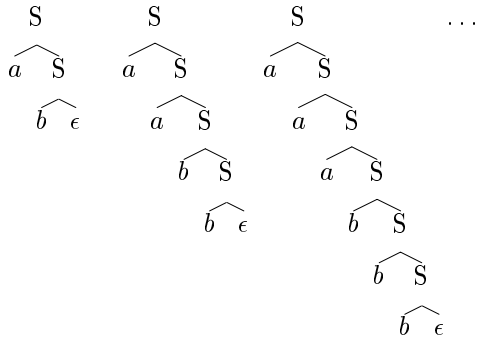

Figure 2: TAG-generable tree set for $a^n b^n$.

However, weak generative capacity ignores the capacity of a grammar formalism to generate derived trees. This is known as its *strong generative capacity*. For example, CFGs and TAGs can both generate the language $a^n b^n$, but CFGs can only associate the a's and b's by making them siblings in the derived tree, as shown in Figure 1, whereas a TAG can generate the infinite set of trees for the language $a^n b^n$ that have a's and b's as siblings, as well as the infinite set of trees where the a's dominate the b's in each tree, shown in Figure 2 (Joshi, 1985); thus TAGs have more strong generative capacity than CFGs.

In addition to the tree sets and string languages a formalism can generate, there may

also be linguistic reasons to care about how these structures are derived. For this reason, multi-component TAGs (MCTAGs) (Weir, 1988) have been adopted to model some linguistic phenomena. In multi-component TAG, elementary trees are grouped into *tree sets*, and at each step of the derivation all the trees of a set adjoin simultaneously. In tree-local MCTAG (TL-MCTAG) all the trees of a set are required to adjoin into the same elementary tree; in set-local MCTAG (SL-MCTAG) all the trees of a set are required to adjoin into the same elementary tree set. TL-MCTAGs can generate the same string languages and derived tree sets as ordinary TAGs, so they have the same weak and strong generative capacities, but TL-MCTAGs can derive these same strings and trees in more than TAGs can. One motivation for TL-MCTAG as a linguistic formalism (Frank, 1992) is that it can generate a functional head (such as *does*) in the same derivational step as the lexical head with which it is associated (see Figure 3) without violating any assumptions about the derived phrase structure tree – something TAGs cannot do in every case.

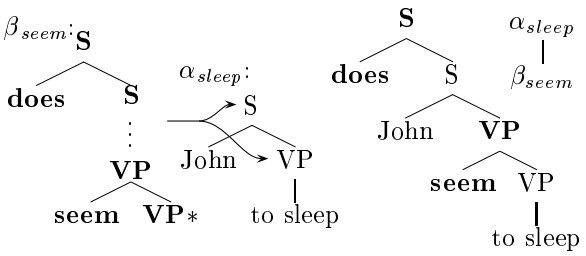

Figure 3: TL-MCTAG generable derivation

This notion of the derivations of a grammar formalism as they relate to the structures they derive has been called the *derivational generative capacity* (1992). Somewhat more formally (for a precise definition, see Becker et al. (1992)): we annotate each element of a derived structure with a code indicating which step of the derivation produced that element. This code is simply the address of the corresponding node in the derivation tree.[1] Then a formalism's derivational generative capacity is the sets of derived structures, thus annotated, that it can generate.

[1] In Becker et al. (1992) the derived structures were always strings, and the codes were not addresses but unordered identifiers. We trust that our definition is in the spirit of theirs.

The derivational generative capacity of a formalism also describes what parts of a derived structure combine with each other. Thus if we consider each derivation step to correspond to a semantic dependency, then derivational generative capacity describes what other elements a semantic element may depend on. That is, if we interpret the derivation trees of TAG as dependency structures and the derived trees as phrase structures, then the derivational generative capacity of TAG limits the possible dependency structures that can be assigned to a given phrase structure.

1.1 Dependency and Constituency

We have seen that TL-MCTAGs can generate some derivations for "Does John seem to sleep" that TAG cannot, but even TL-MCTAG cannot generate the string, "Does John seem likely to sleep" with a derived tree that matches some linguistic notion of correct constituency and a derivation that matches some notion of correct dependency. This is because the components for 'does' and 'seem' would have to adjoin into different components of the elementary tree set for 'likely' (see Figure 4), which would require a set-local multi-component TAG instead of tree-local.

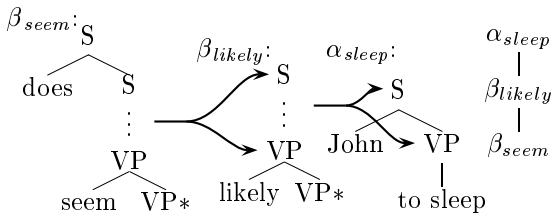

Figure 4: SL-MCTAG generable derivation

Unfortunately, unrestricted set-local multi-component TAGs not only have more derivational generative capacity than TAGs, but they also have more weak generative capacity: SL-MCTAGs can generate the quadruple copy language $wwww$, for example, which does not correspond to any known linguistic phenomenon. Other formalisms aiming to model dependency correctly similarly expand weak generative capacity, notably D-tree Substitution Grammar (Rambow et al., 1995), and consequently end up with much greater parsing complexity.

The work in this paper follows another

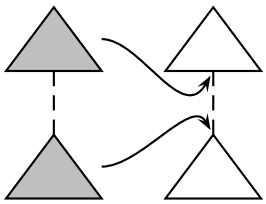

Figure 5: Set-local adjunction.

line of research which has focused on squeezing as much strong generative capacity as possible out of weakly TAG-equivalent formalisms. Tree-local multicomponent TAG (Weir, 1988), nondirectional composition (Joshi and Vijay-Shanker, 1999), and segmented adjunction (Kulick, 2000) are examples of this approach, wherein the constraint on weak generative capacity naturally limits the expressivity of these systems. We discuss the relation of the formalism of this paper, Restricted MCTAG (R-MCTAG) with some of these in Section 5.

2 Formalism

2.1 Restricting set-local MCTAG

The way we propose to deal with multicomponent adjunction is first to limit the number of components to two, and then, roughly speaking, to treat two-component adjunction as one-component adjunction by temporarily *removing* the material between the two adjunction sites. The reasons behind this scheme will be explained in subsequent sections, but we mention it now because it motivates the somewhat complicated restrictions on possible adjunction sites:

- One adjunction site must dominate the other. If the two sites are η_h and η_l, call the set of nodes dominated by one node but not strictly dominated by the other the *site-segment* $\langle \eta_h, \eta_l \rangle$.
- Removing a site-segment must not deprive a tree of its foot node. That is, no site-segment $\langle \eta_h, \eta_l \rangle$ may contain a foot node unless η_l is itself the foot node.
- If two tree sets adjoin into the same tree, the two site-segments must be simultaneously removable. That is, the two site-segments must be disjoint, or one must contain the other.

Because of the first restriction, we depict tree sets with the components connected by a dominance link (dotted line), in the manner of (Becker et al., 1991). As written, the above rules only allow tree-local adjunction; we can generalize them to allow set-local adjunction by treating this dominance link like an ordinary arc. But this would increase the weak generative capacity of the system. For present purposes it is sufficient just to allow one type of set-local adjunction: adjoin the upper tree to the upper foot, and the lower tree to the lower root (see Figure 5).

This does not increase the weak generative capacity, as will be shown in Section 2.3. Observe that the set-local TAG given in Figure 5 obeys the above restrictions.

2.2 2LTAG

For the following section, it is useful to think of TAG in a manner other than the usual. Instead of it being a tree-rewriting system whose derivation history is recorded in a derivation tree, it can be thought of as a set of trees (the 'derivation' trees) with a *yield function* (here, reading off the node labels of derivation trees, and composing corresponding elementary trees by adjunction or substitution as appropriate) applied to get the TAG trees. Weir (1988) observed that several TAGs could be daisy-chained into a *multi-level TAG* whose yield function is the composition of the individual yield functions.

More precisely: a 2LTAG is a pair of TAGs $\langle G, G' \rangle = \langle \langle \Sigma, NT, I, A, S \rangle, \langle I \cup A, I \cup A, I', A', S' \rangle \rangle$.

We call G the *object-level* grammar, and G' the *meta-level* grammar. The object-level grammar is a standard TAG: Σ and NT are its terminal and nonterminal alphabets, I and A are its initial and auxiliary trees, and $S \in I$ contains the trees which derivations may start with.

The meta-level grammar G' is defined so that it derives trees that look like derivation trees of G:

- Nodes are labeled with (the names of) elementary trees of G.
- Foot nodes have no labels.
- Arcs are labeled with Gorn addresses.[2]

[2]The Gorn address of a root node is ϵ; if a node has Gorn address η, then its ith child has Gorn address

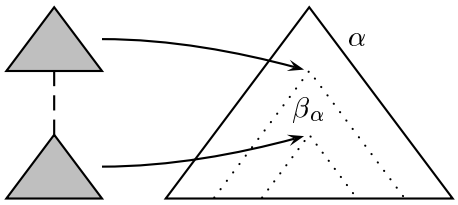

Figure 6: Adjoining into α by removing β_α.

- An auxiliary tree may adjoin anywhere.
- When a tree β is adjoined at a node η, η is rewritten as β, and the foot of β inherits the label of η.

The *tree set* of $\langle G, G' \rangle$, $\mathcal{T}(\langle G, G' \rangle)$, is $f_G[\mathcal{T}(G')]$, where f_G is the yield function of G and $\mathcal{T}(G')$ is the tree set of G'. Thus, the elementary trees of G' are combined to form a derived tree, which is then interpreted as a derivation tree for G, which gives instructions for combining elementary trees of G into the final derived tree.

It was shown in Dras (1999) that when the meta-level grammar is in the regular form of Rogers (1994) the formalism is weakly equivalent to TAG.

2.3 Reducing restricted R-MCTAG to RF-2LTAG

Consider the case of a multicomponent tree set $\{\beta_1, \beta_2\}$ adjoining into an initial tree α (Figure 6). Recall that we defined a site-segment of a pair of adjunction sites to be all the nodes which are dominated by the upper site but not the lower site. Imagine that the site-segment β_α is excised from α, and that β_1 and β_2 are fused into a single elementary tree. Now we can simulate the multi-component adjunction by ordinary adjunction: adjoin the fused β_1 and β_2 into what is left of α; then replace β_α by adjoining it between β_1 and β_2.

The replacement of β_α can be postponed indefinitely: some other (fused) tree set $\{\beta_1', \beta_2'\}$ can adjoin between β_1 and β_2, and so on, and then β_α adjoins between the last pair of trees. This will produce the same result as a series of set-local adjunctions.

More formally:

1. Fuse all the elementary tree sets of the grammar by identifying the upper foot with the lower root. Designate this fused node the *meta-foot*.
2. For each tree, and for every possible combination of site-segments, excise all the site-segments and add all the trees thus produced (the excised auxiliary trees and the remainders) to the grammar.

Now that our grammar has been smashed to pieces, we must make sure that the right pieces go back in the right places. We could do this using features, but the resulting grammar would only be strongly equivalent, not derivationally equivalent, to the original. Therefore we use a meta-level grammar instead:

1. For each initial tree, and for every possible combination of site-segments, construct the derivation tree that will reassemble the pieces created in step (2) above and add it to the meta-level grammar.
2. For each auxiliary tree, and for every possible combination of site-segments, construct a derivation tree as above, and for the node which corresponds to the piece containing the meta-foot, add a child, label its arc with the meta-foot's address (within the piece), and mark it a foot node. Add the resulting (meta-level) auxiliary tree to the meta-level grammar.

Observe that set-local adjunction corresponds to meta-level adjunction along the (meta-level) spine. Recall that we restricted set-local adjunction so that a tree set can only adjoin at the foot of the upper tree and the root of the lower tree. Since this pair of nodes corresponds to the meta-foot, we can restate our restriction in terms of the converted grammar: no meta-level adjunction is allowed along the spine of a (meta-level) auxiliary tree except at the (meta-level) foot.

Then all meta-level adjunction is regular adjunction in the sense of (Rogers, 1994). Therefore this converted 2LTAG produces derivation tree sets which are recognizable, and therefore our formalism is strongly equivalent to TAG.

Note that this restriction is much stronger than Rogers' regular form restriction. This was done for two reasons. First, the definition of our restriction would have been more complicated otherwise; second, this restric-

$\eta \cdot i$.

tion overcomes some computational difficulties with RF-TAG which we discuss below.

3 Linguistic Applications

In cases where TAG models dependencies correctly, the use of R-MCTAG is straightforward: when an auxiliary tree adjoins at a site pair which is just a single node, it looks just like conventional adjunction. However, in problematic cases we can use the extra expressive power of R-MCTAG to model dependencies correctly. Two such cases are discussed below.

3.1 Bridge and Raising Verbs

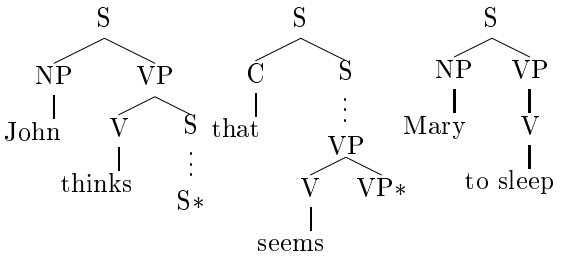

Figure 7: Trees for (1)

Consider the case of sentences which contain both bridge and raising verbs, noted by Rambow et al. (1995). In most TAG-based analyses, bridge verbs adjoin at S (or C′), and raising verbs adjoin at VP (or I′). Thus the derivation for a sentence like

(1) John thinks that Mary seems to sleep.

will have the trees for *thinks* and *seems* simultaneously adjoining into the tree for *like*, which, when interpreted, gives an incorrect dependency structure.

But under the present view we can analyze sentences like (1) with derivations mirroring dependencies. The desired trees for (1) are shown in Figure 7. Since the tree for *that seems* can meta-adjoin around the subject, the tree for *thinks* correctly adjoins into the tree for *seems* rather than *eat*.

Also, although the above analysis produces the correct dependency links, the directions are inverted in some cases. This is a disadvantage compared to, for example, DSG; but since the directions are consistently inverted, for applications like translation or statistical modeling, the particular choice of direction is usually immaterial.

3.2 More on Raising Verbs

Tree-local MCTAG is able to derive (2a), but unable to derive (2b) except by adjoining the auxiliary tree for *to be likely* at the foot of the auxiliary tree for *seem* (Frank et al., 1999).

(2) a. Does John seem to sleep?
b. Does John seem to be likely to sleep?

The derivation structure of this analysis does not match the dependencies, however—*seem* adjoins into *to sleep*.

DSG can derive this sentence with a derivation matching the dependencies, but it loses some of the advantage of TAG in that, for example, cases of super-raising (where the verb is raised out of two clauses) must be explicitly ruled out by subsertion-insertion constraints. Frank et al. (1999) and Kulick (2000) give analyses of raising which assign the desired derivation structures without running into this problem. It turns out that the analysis of raising from the previous section, designed for a translation problem, has both of these properties as well. The grammar is shown back in Figure 4.

4 A Parser

Figure 8 shows a CKY-style parser for our restriction of MCTAG as a system of inference rules. It is limited to grammars whose trees are at most binary-branching.

The parser consists of rules over items of one of the following forms, where $w_1 \cdots w_n$ is the input; η, η_h, and η_l specify nodes of the grammar; i, j, k, and l are integers between 0 and n inclusive; and *code* is either + or −:

- $[\eta, code, i, -, -, l, -, -]$ and $[\eta, code, i, j, k, l, -, -]$ function as in a CKY-style parser for standard TAG (Vijay-Shanker, 1987): the subtree rooted by $\eta \in T$ derives a tree whose fringe is $w_i \cdots w_l$ if T is initial, or $w_i \cdots w_j F w_k \cdots w_l$ if T is the lower auxiliary tree of a set and F is the label of its foot node. In all four item forms, $code = +$ iff adjunction has taken place at η.

- $[\eta, code, i, j, k, l, -, \eta_l]$ specifies that the segment $\langle \eta, \eta_l \rangle$ derives a tree whose fringe is $w_i \cdots w_j L w_k \cdots w_l$, where L is the label of η_l. Intuitively, it means that a potential site-segment has been recognized.
- $[\eta, code, i, j, k, l, \eta_h, \eta_l]$ specifies, if η belongs to the upper tree of a set, that the subtree rooted by η, the segment $\langle \eta_h, \eta_l \rangle$, and the lower tree concatenated together derive a tree whose fringe is $w_i \cdots w_j F w_k \cdots w_l$, where F is the label of the lower foot node. Intuitively, it means that a tree set has been partially recognized, with a site-segment inserted between the two components.

The rules which require differ from a TAG parser and hence explanation are Pseudopod, Push, Pop, and Pop-push. Pseudopod applies to any potential lower adjunction site and is so called because the parser essentially views every potential site-segment as an auxiliary tree (see Section 2.3), and the Pseudopod axiom recognizes the feet of these false auxiliary trees.

The Push rule performs the adjunction of one of these false auxiliary trees—that is, it places a site-segment between the two trees of an elementary tree set. It is so called because the site-segment is saved in a "stack" so that the rest of its elementary tree can be recognized later. Of course, in our case the "stack" has at most one element.

The Pop rule does the reverse: every completed elementary tree set must contain a site-segment, and the Pop rule places it back where the site-segment came from, emptying the "stack." The Pop-push rule performs set-local adjunction: a completed elementary tree set is placed between the two trees of yet another elementary tree set, and the "stack" is unchanged.

Pop-push is computationally the most expensive rule; since it involves six indices and three different elementary trees, its running time is $\mathcal{O}(n^6 G^3)$.

It was noted in (Chiang et al., 2000) that for synchronous RF-2LTAG, parse forests could not be transferred in time $\mathcal{O}(n^6)$. This fact turns out to be connected to several properties of RF-TAG (Rogers, 1994).[3]

[3]Thanks to Anoop Sarkar for pointing out the first such connection.

The CKY-style parser for regular form TAG described in (Rogers, 1994) essentially keeps track of adjunctions using stacks, and the regular form constraint ensures that the stack depth is bounded. The only kinds of adjunction that can occur to arbitrary depth are root and foot adjunction, which are treated similarly to substitution and do not affect the stacks. The reader will note that our parser works in exactly the same way.

A problem arises if we allow both root and foot adjunction, however. It is well-known that allowing both types of adjunction creates derivational ambiguity (Vijay-Shanker, 1987): adjoining β_1 at the foot of β_2 produces the same derived tree that adjoining β_1 at the root of β_2 would. The problem is not the ambiguity *per se*, but that the regular form TAG parser, unlike a standard TAG parser, does not always distinguish these multiple derivations, because root and foot adjunction are both performed by the same rule (analogous to our Pop-push). Thus for a given application of this rule, it is not possible to say which tree is adjoining into which without examining the rest of the derivation.

But this knowledge is necessary to perform certain tasks online: for example, enforcing adjoining constraints, computing probabilities (and pruning based on them), or performing synchronous mappings. Therefore we arbitrarily forbid one of the two possibilities.[4] The parser given in Section 4 already takes this into account.

5 Discussion

Our version of MCTAG follows other work in incorporating dependency into a constituency-based approach to modeling natural language. One such early integration involved work by Gaifman (1965), which showed that projective dependency grammars could be represented by CFGs. However, it is known that there are common phenomena which require non-projective dependency grammars, so looking only at projective de-

[4]Against tradition, we forbid root adjunction, because adjunction at the foot ensures that a bottom-up traversal of the derived tree will encounter elementary trees in the same order as they appear in a bottom-up traversal of the derivation tree, simplifying the calculation of derivations.

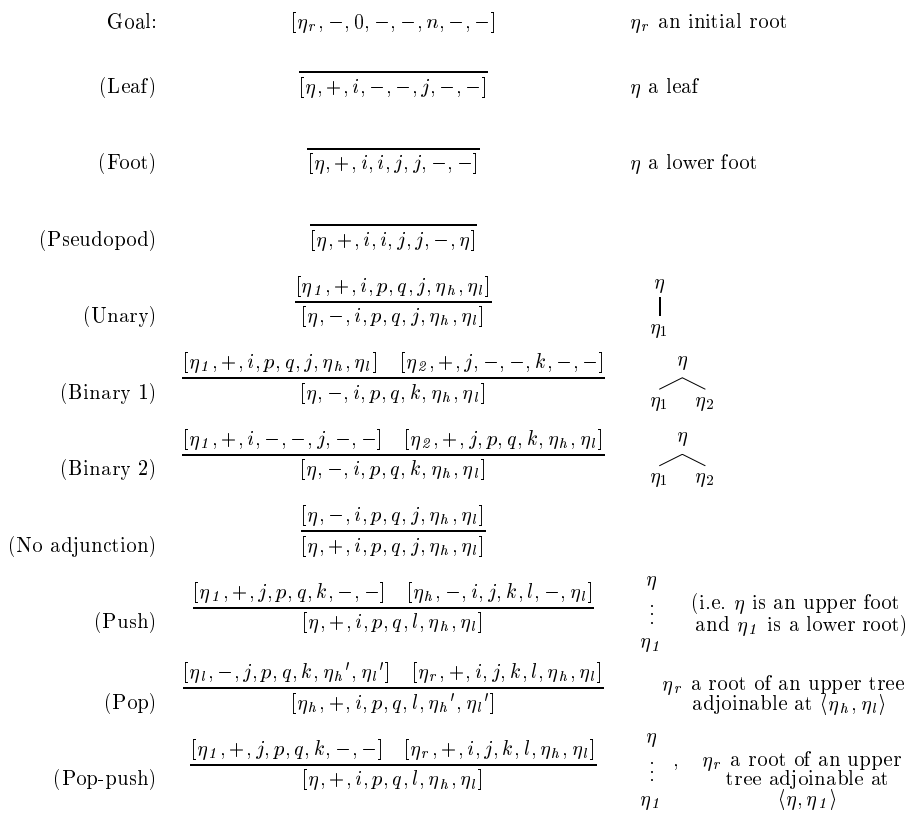

Figure 8: Parser

pendency grammars is inadequate. Following the observation of TAG derivations' similarity to dependency relations, other formalisms have also looked at relating dependency and constituency approaches to grammar formalisms.

A more recent instance is D-Tree Substitution Grammars (DSG) (Rambow et al., 1995), where the derivations are also interpreted as dependency relations. Thought of in the terms of this paper, there is a clear parallel with R-MCTAG, with a local set ultimately representing dependencies having some yield function applied to it; the idea of non-immediate dominance also appears in both formalisms. The difference between the two is in the kinds of languages that they are able to describe: DSG is both less and more restrictive than R-MCTAG. DSG can generate the language COUNT-k for some arbitrary k (that is, $\{{a_1}^n {a_2}^n \ldots {a_k}^n\}$), which makes it extremely powerful, whereas R-MCTAG can only generate COUNT-4. However, DSG cannot generate the copy language (that is, $\{ww \mid w \in \Sigma^*\}$ with Σ some terminal alphabet), whereas R-MCTAG can; this may be problematic for a formalism modeling natural language, given the key role of the copy language in demonstrating that natural language is not context-free (Shieber, 1985). R-MCTAG is thus a more constrained relaxation of the notion of immediate dominance in favor of non-immediate dominance than is the case for DSG.

Another formalism of particular interest here is the Segmented Adjoining Grammar of (Kulick, 2000). This generalization of TAG is characterized by an extension of the adjoining operation, motivated by evidence in scrambling, clitic climbing and subject-to-subject raising. Most interestingly, this extension to TAG, proposed on empirical grounds, is defined by a composition operation with constrained non-immediate dominance links that looks quite similar to the formalism described in this paper, which began from formal considerations and was then applied to data. This confluence suggests that the ideas described here concerning combining dependency and constituency might be reaching towards some

deeper connection.

6 Conclusion

From a theoretical perspective, extracting more derivational generative capacity and thereby integrating dependency and constituency into a common framework is an interesting exercise. It also, however, proves to be useful in modeling otherwise problematic constructions, such as subject-auxiliary inversion and bridge and raising verb interleaving. Moreover, the formalism developed from theoretical considerations, presented in this paper, has similar properties to work developed on empirical grounds, suggesting that this is worth further exploration.

References

Tilman Becker, Aravind Joshi, and Owen Rambow. 1991. Long distance scrambling and tree adjoining grammars. In *Fifth Conference of the European Chapter of the Association for Computational Linguistics (EACL'91)*, pages 21–26.

Tilman Becker, Owen Rambow, and Michael Niv. 1992. The derivational generative power of formal systems, or, Scrambling is beyond LCFRS. Technical Report IRCS-92-38, Institute for Research in Cognitive Science, University of Pennsylvania.

David Chiang, William Schuler, and Mark Dras. 2000. Some Remarks on an Extension of Synchronous TAG. In *Proceedings of TAG+5*, Paris, France.

Mark Dras. 1999. A meta-level grammar: redefining synchronous TAG for translation and paraphrase. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL '99)*.

Robert Frank, Seth Kulick, and K. Vijay-Shanker. 1999. C-command and extraction in tree-adjoining grammar. *Proceedings of the Sixth Meeting on the Mathematics of Language (MOL6)*.

Robert Frank. 1992. *Syntactic locality and tree adjoining grammar: grammatical acquisition and processing perspectives.* Ph.D. thesis, Computer Science Department, University of Pennsylvania.

Haim Gaifman. 1965. Dependency Systems and Phrase-Structure Systems. *Information and Control*, 8:304–337.

Gerald Gazdar. 1988. Applicability of indexed grammars to natural languages. In Uwe Reyle and Christian Rohrer, editors, *Natural Language Parsin and Linguistic Theories.* D. Reidel Publishing Company, Dordrecht, Holland.

Aravind Joshi and K. Vijay-Shanker. 1999. Compositional Semantics with Lexicalized Tree-Adjoining Grammar (LTAG): How Much Underspecification is Necessary? In *Proceedings of the 2nd International Workshop on Computational Semantics.*

Aravind K. Joshi, Leon S. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal of computer and system sciences*, 10:136–163.

Aravind K. Joshi. 1985. How much context sensitivity is necessary for characterizing structural descriptions: Tree adjoining grammars. In L. Karttunen D. Dowty and A. Zwicky, editors, *Natural language parsing: Psychological, computational and theoretical perspectives*, pages 206–250. Cambridge University Press, Cambridge, U.K.

Aravind Joshi. 2000. Relationship between strong and weak generative power of formal systems. In *Proceedings of TAG+5*, pages 107–114, Paris, France.

Seth Kulick. 2000. A uniform account of locality constraints for clitic climbing and long scrambling. In *Proceedings of the Penn Linguistics Colloquium.*

Owen Rambow, David Weir, and K. Vijay-Shanker. 1995. D-tree grammars. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL '95).*

James Rogers. 1994. Capturing CFLs with tree adjoining grammars. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL '94).*

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

K. Vijay-Shanker. 1987. *A study of tree adjoining grammars.* Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

David Weir. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms.* Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.