*MTT 2003, Paris, 16–18 juin 2003*

# A Formalism for Dependency Grammar Based on Tree Adjoining Grammar

Aravind Joshi
Department of CIS, U. of Pennsylvania
Philadelphia, PA, USA
joshi@linc.cis.upenn.edu

Owen Rambow
Department of CS, Columbia U.
New York, NY, USA
rambow@cs.columbia.edu

## Mots-clefs – Keywords

Formalismes, Grammaire d'Arbres Adjoints
Formalisms, Tree Adjoining Grammar

## Abstract - Resumé

In this paper, we present a formalism for dependency grammar based on some key ideas from Tree-Adjoining Grammars. We represent a dependency grammar in terms of elementary dependency trees anchored on lexical items. These elementary trees correctly capture the dependencies associated with the lexical anchor. These trees may also include nodes that represent items on which the lexical anchor depends. These nodes are well motivated. We also describe operations that combine elementary or derived dependency trees, which are analogous to "substitution" and "adjoining" in TAG. This characterization of a dependency grammar allows one to transfer all the key insights from TAG to dependency grammars.

Dans cet article, nous présentons un formalisme pour les grammaires de dépendence qui est basé sur quelques idées-clef des grammaires d'arbres adjoints (TAG). Nous représentons une grammaire de dépendence par un ensemble d'arbres élémentaires qui sont anchrés par des lexèmes. Ces arbres élémentaires représentent les dépendences associés avec l'anchre lexical, y inclus (dans le cas des adjoints) les noeuds dont il dépend. Nous définissons aussi des opérations pour combiner des arbres (élémentaires ou dérivés) inspirés des opération de TAG: la substitution et l'adjonction. Cette définition formelle d'une grammaire de dépendence nous permet de dériver facilement certains ordres de mots non-projectifs.

## 1 Introduction

In (Rambow & Joshi, 1997), we compared TAGs and Dependency Grammars from the perspective of word order variation.[1] We observed that the derivation trees in TAG resemble dependency structures, but the derived trees are phrase-structure trees. In the present paper our goal is quite different: we will present a formalism for dependency grammar based on some key ideas from Tree-Adjoining Grammars, but which does not use phrase structure at all. We represent a dependency grammar in terms of elementary dependency trees anchored on lexical items. These elementary trees correctly capture the dependencies associated with the lexical anchor. These trees will also include nodes that represent items on which the lexical anchor depends. Thus each tree captures all and only the dependents of the lexical anchor plus ancestor nodes on which the lexical anchor itself depends. These nodes are well motivated. We also describe operations that combine elementary or derived dependency trees, which are analogous to "substitution" and "adjoining" in TAG. This characterization of a dependency grammar allows one to transfer all the key insights from TAG to dependency grammars. This work has some close connections to (Nasr, 1995; Nasr, 1996; Candito & Kahane, 1998; Kahane, 2002), which we will briefly discuss later. However, like our previous paper and unlike other formal models of MTT (in particular, (Kahane, 2002)), we depart from MTT significantly in that we do not maintain a separation between an representation of unordered surface syntax and a representation of ordered deep morphology; instead, we suggest that in the derivation (i.e., construction) of the syntactic representation, the word order is determined *at the same time*, since the structures we use use for this purpose fix word order, and the operations we use to combine the structures determine the word order of the combined structure, given the orders of the two structures to be combined. In this respect, our formalism follows TAG. We believe there are computational advantages to such a system (both for applications, and for modeling human processing), but we do not go into details in this paper.

The paper is organized as follows. First we will present a short introduction to TAGs including a short comparison with Dependency Grammars. Then we will describe our new approach, mainly by means of examples and then comment on this new formalism from the perspective of insights from TAGs. As in our previous paper, our focus is on non-projective structures, as we feel these pose a particular challenge to the formal description of dependency grammar. We conclude with a comparison to other approaches.

## 2 Introduction to TAGs

In a Tree Adjoining Grammar (TAG), the elementary structures are phrase-structure trees. A sample grammar is given in Figure 1. It consists of three trees, one of which is rooted in S, and two of which are rooted in NP. Note that we have defined TAG in such a way that a tree is now an elementary object of the grammar.

[1]For a general introduction to TAG, see (Abeill´e & Rambow, 2000); for a summary of mathematical and computational properties of TAGs and some related phrase-structure formalisms, see (Joshi *et al.*, 1991); for a discussion of the relation between TAG and categorial systems, see (Joshi & Kulick, 1995).

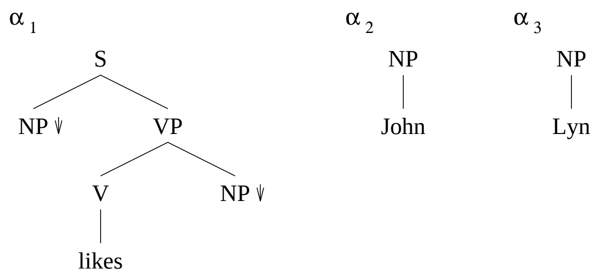

Figure 1: A sample Tree Adjoining Grammar

We combine elementary structures in a TAG by using two operations, *substitution* and *adjunction*. We can substitute tree $\beta$ into tree $\alpha$ if there is a nonterminal symbol on the frontier of $\alpha$ which has the same label as the root node of $\beta$. We can then simply append $\beta$ to $\alpha$ at that node. A derivation in our sample TAG is shown in Figure 2. The trees representing the two arguments of the verb *like*, *John* ($\alpha_2$) and *Lyn* ($\alpha_3$), are substituted into the tree associated with the verb ($\alpha_1$), yielding the well-formed tree $\alpha_4$, from which the sentence *John likes Lyn* can be read off.

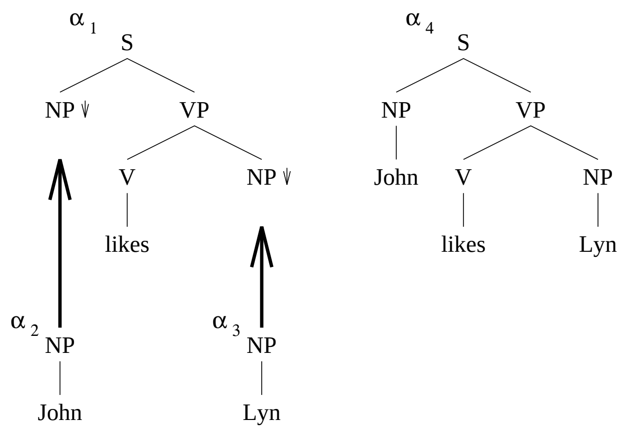

Figure 2: Substitution of arguments into initial tree of *likes*

In adjunction, a tree $\alpha$ (called an “initial tree”) contains a non-terminal node labeled $A$; the root node of tree $\beta$ (an “auxiliary tree”) is also labeled $A$, as is exactly one non-terminal node on its frontier (the “foot node”). All other frontier nodes are terminal nodes or substitution nodes. We take tree $\alpha$ and remove the subtree rooted at its node $A$, insert in its stead tree $\beta$, and then add at the footnode of $\beta$ the subtree of $\alpha$ that we removed earlier. Thus, adjunction can have the effect of inserting one tree into the center of another. Our linguistic example is continued in Figure 3. Tree $\beta_1$ containing the adverb is adjoined at the VP node into tree $\alpha_4$. The result is tree $\alpha_5$. Note that $\alpha_5$ is composed of trees $\alpha_1$, $\alpha_2$, $\alpha_3$ and $\beta_1$, each of which correspond to exactly one lexical item.

TAG elementary structures have an extended “domain of locality”. This increased domain of locality allows the linguist to associate each tree with one lexical head, and to state linguistic relationships of the lexical head (such as subcategorization, semantic roles of arguments, case assignment, agreement, and word order) locally in the elementary tree. We call a formalism which has one lexical head per tree a “lexicalized grammar”. As an example, take agreement between subject and verb in English. The linguist working in TAG can simply state (by using

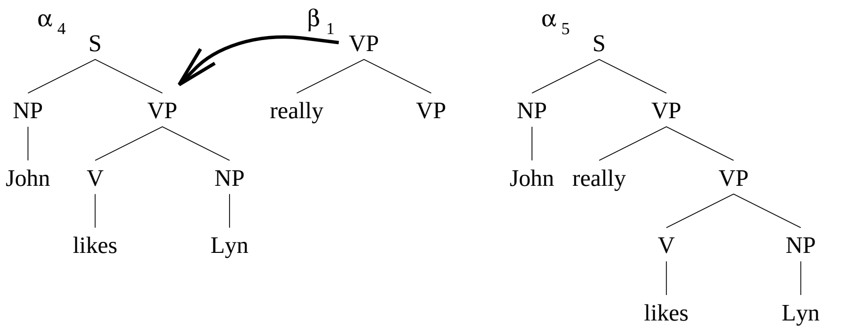

Figure 3: Adjunction of *really* into initial tree

some feature-based notation) that the verb and the NP in subject position in tree $\alpha_1$ of Figure 1 agree with respect to number.

## 3 A Formalism for Dependency Grammars

We now present a formalism which, unlike TAG, directly derives a dependency structure, but still allows us to handle the same cases of non-projectivity that TAG handles. The idea is to define a grammar with elementary dependency trees which encode both dependency and word-order, and that are combined using well-defined operations. This approach, like TAG, means that all word-order phenomena are expressed locally in the elementary trees; there are no global word-order rules. We do not define this formalism formally but give some examples. We start out with English in this section, then turn to German and Dutch in the next section.

In Figure 4 we show how the dependency tree for *John eats apples often* can be composed out of four elementary dependency trees, one each for *eats*, *John*, *apples*, and *often*. The tree for *eats* shows a black node associated with *eats*, the lexical anchor. The white nodes show the dependents of *eats*.[2] The white nodes represent variables in a sense. We also associate labels with nodes that indicate the lexical class, for example, $N$, and perhaps other feature content (such as [wh:+]) as needed. In brackets we add the lexical content – if there is lexical content, the node is black, white otherwise. Note that the trees (in their graphical representation) fully specify the ordering of all the nodes in the tree. The elementary trees of *John* and *apples* are atomic as they have no white nodes. The elementary tree for *often* has no dependents. However, the white node stands for an ancestor node on which *often* depends (representing its passive valency).

We initially use a single combination operation, which we will informally call *attachment*. An attachment consists in unifying a non-lexical white node with a lexical black node. Attachment is thus non-directional. Given the trees in Figure 4 on the left and the double-headed arrows indicating attachment, it is clear that we can derive the tree on the right.

For adjuncts such as *often*, the question immediately arises how we can state word order constraints. The ordering of the dependent (*often* in this case) and the governor is fixed in the

[2]The black node/white node notation is borrowed from (Nasr, 1996).

tree, as in all our elementary trees. However, the formalism so far does not specify how to fix the order of *often* with respect to its sisters after it is combined with the tree for *eat*.[3] Several options are available. We can simply leave the word order of adjuncts with respect to sisters underspecified, and overgenerate ungrammatical sentences (such as **John eats often apples*). We can let the dependent tree (the adjunct) specify restrictions on its immediate neighbors; for example, English adverbials could state that they may not immediately follow a verb. Or, finally (and this is probably the most appealing solution), we can let the governor specify restrictions on adjuncts, in this case that adverbials may not appear between the head and the direct object.[4]

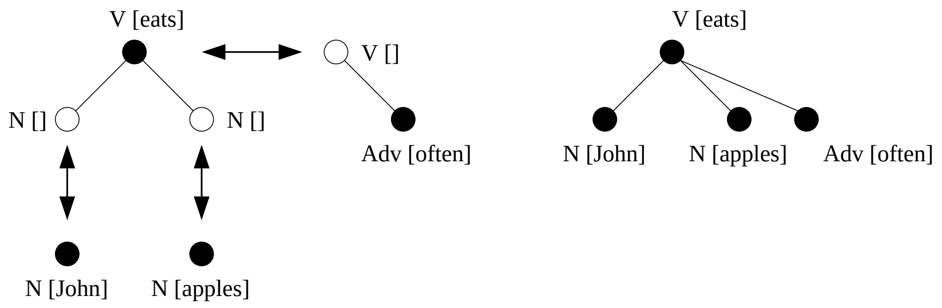

Figure 4: Some elementary tree (left) with attachments; final derived tree (right). Double-headed arrows show attachments.

In Figure 5 we show a slightly more complicated example. The elementary tree for *eat* has three white nodes. One of these will be replaced with the black node that corresponds to *John*, which is one of the dependents of *eat*. The other dependent of *eat* is the white node with label $wh+$. This dependent should have appeared below the node for *eat* but we show it in a displaced location. First, we have another white node above the black node associated with *eat*. This is an ancestor node on which the node for *eat* depends. This is predicted by the lexical anchor *eat* because it is in a non-finite form and therefore must in turn depend upon another node, which in the present case is unified with the black node for *does*. (Note that we assume that the category Aux is a subtype of V and therefore Aux and V unify to Aux.) This additional node in the elementary tree for *eat* provides a location for attaching the white node ($wh+$ node) and thus allows us to model the *wh*-movement in the elementary tree for *eat*. We could also add a white node under the black node for *eat* and coindex it with the $wh+$ white node (i.e., a trace). This additional white node will be an empty node, i.e., it will not be unified with any black node. This will make it analogous with TAG. However we have not followed this path as we want our representations close to dependency grammars.

Note that (even if we do not use traces) there are two cases of white nodes. In the first case, there is no lexical content at all (empty brackets). The white node merely expresses the fact that a node of a certain type must be present at the end of the derivation (active valency of the governor). It is like a substitution node in TAG, and is used for (obligatory) arguments, or, in the

[3] This is the same problem for many dependency formalisms; for example, (Mel'čuk & Pertsov, 1987) provide a very detailed list of English "syntagms", i.e. orderings of governor and dependent, but do not discuss the ordering of sisters.

[4] This approach may be similar in some respects to the approach of (Gerdes, 2002), though his machinery is much more powerful.

case of adjuncts, it is like the footnode in modifier auxiliary trees, indicating where the modifier can adjoin. In the second case, the white node has no explicit lexical content either but the node is closely associated with a black node of the same category in the same tree (brackets with an empty string *e*). In this case, the white node is predicted from the morphology or semantics of the black node. Additional words may be added (depending on the exact feature content), but these words will not be autosemantic. The distinction between the two types of white nodes reflects the distinction between the dependency between a verb and one of its arguments, and the dependency between a verb and an auxiliary: clearly, these are very different types of dependencies (as reflected in the fact that the former, but not the latter, appear in the DSyntR of MTT).

The elementary tree for *eat* is of depth greater than one. It is this feature, i.e., the possibility of having an ‘extended domain of locality’, that allows us to show the two dependents of *eat* at different levels and thus account for the correct valency as well as the ‘movement’. This is exactly what is achieved in a TAG by having elementary trees not just one level phrase structure trees but possibly larger tree–large enough to encapsulate all the arguments of the lexical anchor but also capture the relative positions of the anchor and its arguments.

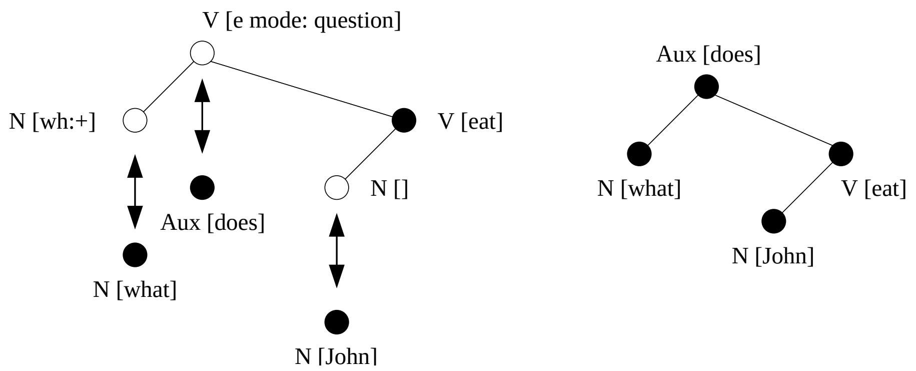

Figure 5: Trees for *eats* with *wh*-movement and other trees (left); derived tree (right).

Finally, we give a yet more complex example, which involves the composition of a complement-taking predicate and the complement itself. When combined with $wh$-movement, we get an example that brings out all the key aspects of TAG. In this example, in Figure 6, the elementary tree for *eat* is the same as in Figure 4. One of the dependents is dislocated, as before. The elementary tree for *think* has two white nodes, to one which will be unified with the black node for *Mary* (its nominal subject) and the other with the black node associated with *eat* (its clausal object). In this case, we cannot view the operation of attachment simply as one in which we equate (unify) a black node and a white node, because there is more structure associated with the white node (and we cannot have a structure in which a node has two distinct governors). Instead, we use a more complex notion of attachment: after unifying the black node for *eat* with the white node below *think*, we insert the whole *think* tree in the middle of the *eat* tree, in order to obtain a new tree. Note that, as in simple attachment, we have unified exactly one pair of nodes (one black, one white). It is clear that complex attachment mirrors adjunction of a predicative auxiliary tree (a tree representing a matrix verb) in TAG. For our dependency formalism (in analogy to TAG), we will require that when we adjoin a tree at a node, the adjoined tree must have a root node and a leaf node of the same lexical category (V in our case); furthermore, the leaf node must have no lexical content, not even an empty category. Put differently, it must be an argument of the head (V[]).

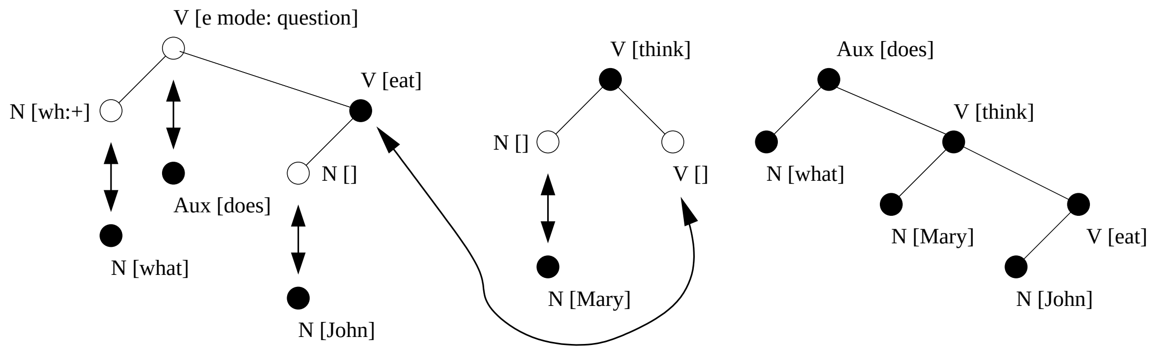

Figure 6: Tree for *eats* with long-distance *wh*-movement (left); derived tree (right).

## 4 Examples from German and Dutch

*Wh*-movement in English is one type of non-projectivity; it is characterized by the fact that only one phrase is "displaced" and causes non-projectivity. In cross-serial dependencies, we have another example of non-projectivity which involved an unbounded number of phrases, and is thus more difficult to account for. We start out with German nested dependencies. For simplicity, we present German syntax with English words. In the following, the intended meaning of the German and Dutch examples is '(because) John tried to teach Mary to feed the pigs'.

(1) (because) John Mary pigs to-feed to-teach tried

This kind of sentence is not a non-projective structure, but we present it for completeness. The trees are really the same as for the simple English case shown in Figure 4, except that they represent the underlying SOV word order in German (as opposed to SVO for English). The derivation proceeds by attaching the root node of each embedded tree at the white verbal node of its governor. This operation is exactly identical to substituting the embedded tree at the the V[] node of the matrix clause. The result is shown below in Figure 7. As can be seen, we obtain the desired word order, and this is the dependency tree one would expect.

We now turn to Dutch:

(2) (because) John Mary pigs tried to-teach to-feed

For Dutch, the elementary are as shown in Figure 8, at the top. We can see that they are identical to German ones — except that, as in the English *wh*-moved case, we have added an additional node, above the actual verb. However, compared to the English example of *wh*-movement, the non-finite verb and the empty node have switched places and the verb actually dominates an

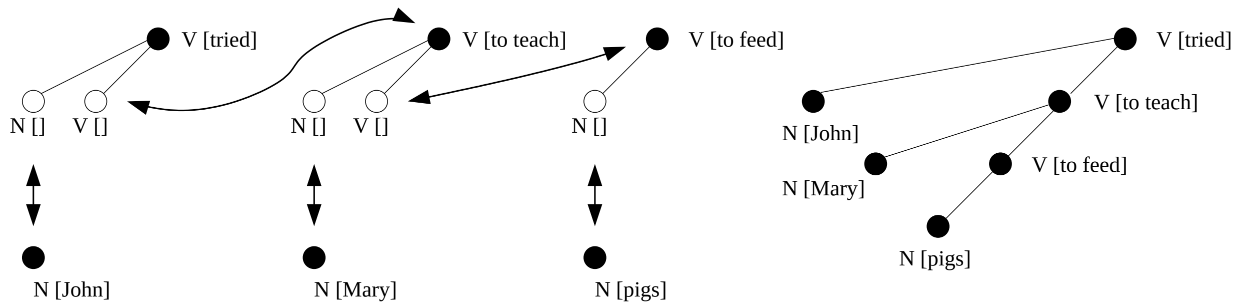

Figure 7: Trees for German nested dependencies

empty node! Why is that? Following much of the (transformational) literature on Germanic, we claim that verbs can "raise" to higher positions in certain languages such as Dutch (or in fact, must do so). For clarity, we have annotated the empty node with the verb of its elementary tree. We now attach each clause to its immediately embedded clause at the position "vacated" by the verb, and we obtain the structure shown in Figure 8 below. We can see that the word order is as desired. If we consider the empty nodes as well (and pay attention to the verbs in parentheses), we can also retrieve the original predicate-argument structure from the derived tree.

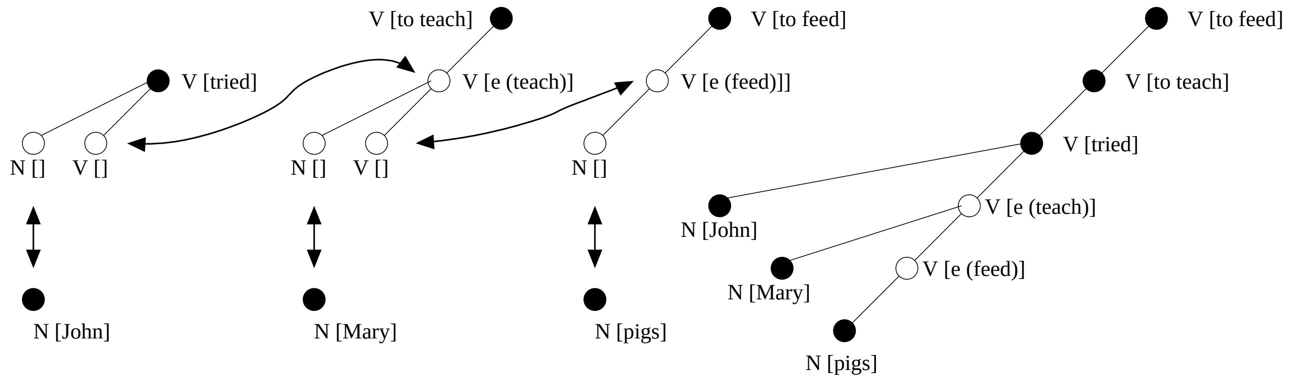

Figure 8: Trees for Dutch cross-serial dependencies

## 5 Comparison to Other Approaches

Our work is directly inspired by that of (Nasr, 1996), who used a similar tree formalism for dependency grammar. He was mainly inspired by the desire to co-locate more than one word in a structure, in order to represent multi-word lexemes and idioms. However, he did not use an operation analogous to adjunction.

While our previous paper (Rambow & Joshi, 1997) addressed the issue of non-projectivity, it did not present a formalism for dependency grammar; rather, it used the derivation tree of TAG as a dependency tree. We showed that the derivation tree could in fact be non-projective. (Candito

& Kahane, 1998) addressed certain issues in comparing the derivation tree to the dependency tree, and suggested ways of mapping directly to a more semantic level of representation.

Several researchers have addressed the issue of defining a formalism for dependency grammar that can directly derive non-projective structures (Lombardo & Lesmo, 1998; Kahane *et al.*, 1998; Kahane, 2000; Kahane, 2002). This work differs from ours in an important way: in these approaches, nodes in a dependency tree are allowed to be reassigned as dependents of an ancestor. In this manner, non-projective structures can be derived since nodes get new governors; however, in those versions that have been shown to have some of TAG's computationally benign properties (Lombardo & Lesmo, 1998; Kahane *et al.*, 1998), it is still impossible to derive Dutch cross-serial dependencies.

## 6 Conclusion

We have sketched a formalism for dependency grammars based on Tree Adjoining Grammar. Like TAG, this formalism can generate certain long-distance phenomena, which in dependency grammar lead to non-projective trees, by specifying all relevant word order facts in the local trees associated with each lexical head. No global word order rules are necessary: the formalism is simpler, and we obtain appealing computational properties.[5] Of course, nothing is free is linguistics, and the tradeoff is that we need to introduce additional nodes associated with the lexical head. Such empty nodes are not part of the standard dependency representation. They are related to the notion of "head movement" introduced in Chomskyan theory in the eighties, according to which lexical heads have several different positions in which they can be realized. The syntax of the arguments and adjuncts is specified with respect to all of these possible head positions, not with respect to the actual head position in a given construction. However, we think that, while the notion of multiple possible head positions may not, at first, be a natural one for dependency syntacticians, it should be considered more carefully. For example, (Gerdes, 2002) uses the notion of "topological field" to describe German syntax in the context of MTT,[6] and the German topological fields of *Vor-*, *Mittel-* and *Nachfeld* are in fact delimited by the possible positions of the verbal head (in second and final positions).

## References

ABEILLÉ A. & RAMBOW O. (2000). Tree Adjoining Grammar: An overview. In A. ABEILLÉ & O. RAMBOW, Eds., *Tree Adjoining Grammars: Formalisms, Linguistic Analyses and Processing*, p. 1–68. CSLI Publications.

CANDITO M.-H. & KAHANE S. (1998). Can the TAG derivation tree represent a semantic graph? An answer in the light of Meaning-Text Theory. In *Proceedings of the Fourth Inter-*

[5]We did not discuss formal properties in this paper, but the results for TAG transfer straightforwardly.

[6]The word order in the *Mittelfeld* is independent of whether the lexical verb is in second or fi nal position (i.e., immediately precedes or immediately follows the *Mittelfeld*), so describing its syntax in relation to the *actual* position of the lexical verb is unnecessarily complex.

*national Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, IRCS Report 98–12, p. 21–24: Institute for Research in Cognitive Science, University of Pennsylvania.

GERDES K. (2002). *Topologie et grammaires formelles de l'allemand*. PhD thesis, Université Paris 7.

JOSHI A. K. & KULICK S. (1995). Partial proof trees as building blocks for categorial grammars. Presented at the MOL4 Workshop, October 1995. A revised version has been submitted for publication to *Linguistics and Philosophy*.

JOSHI A. K., VIJAY-SHANKER K. & WEIR D. (1991). The convergence of mildly context-sensitive grammatical formalisms. In P. SELLS, S. SHIEBER & T. WASOW, Eds., *Foundational Issues in Natural Language Processing*, p. 31–81. Cambridge, Mass.: MIT Press.

KAHANE S. (2000). Extractions dans une grammaire de dépendence lexicalisée à bulles. *Traitement automatque des langues*, **41**(1), 211–243.

KAHANE S. (2002). *Grammaire d'Unification Sens-Texte : Vers un modèle mathématique articulé de la langue*. Document de synthèse de l'habilitation à diriger les recherches, Universiteé Paris 7.

KAHANE S., NASR A. & RAMBOW O. (1998). Pseudo-projectivity: A polynomially parsable non-projective dependency grammar. In *36th Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98)*, p. 646–652, Montréal, Canada.

LOMBARDO V. & LESMO L. (1998). Formal aspects and parsing issue of dependency theory. In *36th Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98)*, p. 787–793, Montréal, Canada.

MEL'ČUK I. A. & PERTSOV N. V. (1987). *Surface Syntax of English*. Amsterdam/Philadelphia: John Benjamins.

NASR A. (1995). A formalism and a parser for lexicalised dependency grammars. In *4th International Workshop on Parsing Technologies*, p. 186–195, Prague.

NASR A. (1996). *Un système de reformulation automatique de phrases fondé sur la Théorie Sens-Texte : application aux langues contrôlées*. PhD thesis, Université Paris 7.

RAMBOW O. & JOSHI A. (1997). A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In L. WANNER, Ed., *Recent Trends in Meaning-Text Theory*, p. 167–190. Amsterdam and Philadelphia: John Benjamins.