Enriching the TAG Derivation Tree for Semantics

Laura Kallmeyer
laura.kallmeyer@linguist.jussieu.fr
TALaNa-Lattice, University Paris 7

Abstract

Most of the proposals for semantics in the Tree Adjoining Grammar (TAG) framework suppose that the derivation tree serves as basis for semantics. However, sometimes the derivation tree does not provide the semantic links one needs. In this paper, some of these cases are inspected, in particular the analysis of quantifiers and of unbounded wh-movement in embedded interrogatives. The paper proposes to enrich the TAG derivation tree and use the resulting structure as basis for semantics. This allows to deal with the problematic cases.

1 TAG and the syntax-semantics interface

1.1 Lexicalized Tree Adjoining Grammars (LTAG)

A LTAG (Joshi and Schabes, 1997) consists of a finite set of trees (elementary trees) associated with lexical items and of composition operations of substitution (replacing a leaf with a new tree) and adjunction (replacing an internal node with a new tree). The elementary trees represent extended projections of lexical items and encapsulate all syntactic/semantic arguments of the lexical anchor. They are minimal in the sense that only the arguments of the anchor are encapsulated, all recursion is factored away.

LTAG derivations are represented by derivation trees that record the history of how the elementary trees are put together. A derived tree is the result of carrying out the substitutions and adjoinings. For a sample derivation see the TAG analysis of (1) in Fig. 1.

(1) John always loves Mary.

The numbers at the nodes in the derivation tree are the positions of the nodes where the trees are added: *John* is substituted for the node at position (1), *Mary* for the node at position (22) and *always* is adjoined to the node at position (2).

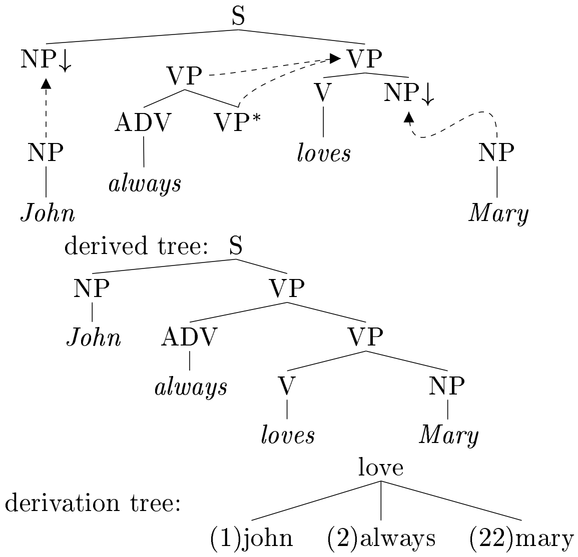

Figure 1: TAG derivation for (1)

1.2 Compositional semantics with LTAG

Because of the localization of the arguments of a lexical item within elementary trees TAG derivation trees express predicate argument dependencies. Therefore it is generally assumed that the proper way to define compositional semantics for LTAG is with respect to the derivation tree, rather than the derived tree (see, e.g., Shieber and Schabes, 1990; Candito and Kahane, 1998; Joshi and Vijay-Shanker, 1999; Kallmeyer and Joshi 1999, 2002).

The overall idea is as follows. Each elementary tree is linked to a semantic representation.

The way the semantic representations combine with each other depends on the derivation tree. Following Kallmeyer and Joshi (1999, 2002), in this paper, we will adopt 'flat' semantic representations as in, for example, Minimal Recursion Semantics (MRS, Copestake et al., 1999). (2) shows the elementary semantic representations for (1).

(2)

l_1 : $\mathsf{love}(x_1, x_2)$ $h_1 \geq l_1$
arg: $\langle x_1, (1)\rangle, \langle x_2, (22)\rangle$

$\mathsf{john}(x)$
arg: –

l_2 : $\mathsf{always}(h_2)$ $g_1 \geq l_2, h_2 \geq s_1$
arg: g_1, s_1

$\mathsf{mary}(y)$
arg: –

Roughly, a semantic representation consists of a conjunctively interpreted set of formulas (typed lambda-expressions), scope constraints and a set of argument variables. The formulas may contain labels and holes (metavariables for propositional labels). In the following, $l_1, l_2, \ldots$ are propositional labels, $h_1, h_2, \ldots$ are propositional holes, $s_1, s_2, \ldots$ are propositional and $x_1, x_2, \ldots$ individual argument variables (whose values must be propositional labels/free individual variables) and $g_1, g_2, \ldots$ are hole variables (special argument variables whose values must be holes). Argument variables may be linked to positions in the elementary tree, as it is the case for x_1 and x_2 in (2).

The use of holes is motivated by the desire to generate underspecified representations (as in, e.g., Bos, 1995) for scope ambiguities though this is not subject of this paper. In the end, after having constructed a semantic representation with holes and labels, disambiguation is done which consists of finding bijections from holes to labels that respect the scope constraints. The examples in this paper are not ambiguous, i.e., there will be just one possible disambiguation. In the semantic representation for *love*, there is for example a hole h_1 above the label l_1 (indicated by the constraint $h_1 \geq l_1$). Between h_1 and l_1, other labels and holes might come in (introduced for example by quantifiers or adverbs) or, if this is not the case, l_1 will be assigned to h_1 in the disambiguation(s).

When combining semantic representations, values are assigned to argument variables and the union of the semantic representations is built. (This corresponds more or less to a conjunction.) The values for the argument variables of a certain (elementary) semantic representation must come from semantic representations that are linked to it in the derivation tree.

The linking of argument variables and syntactic positions restricts the possible values as follows: In a substitution derivation step at a position p, only argument variables linked to p get values. In an adjunction step, only argument variables that are not linked to any positions can get values. In the case of a substitution, a new argument is inserted and therefore a value is assigned to an argument variable in the old semantic representation. However, in the case of an adjunction, a new modifier is applied and therefore a value is assigned to a variable in the semantic representation that is added.

The derivation tree in Fig. 1 indicates that the value of x_1 needs to come from the semantic representation of *John*, the one of x_2 from *Mary* and the values of g_1 and s_1 need to come from *love*. Consequently, $x_1 \to x, x_2 \to y, g_1 \to h_1$ and $s_1 \to l_1$. The result is (3).

(3)

l_1 : $\mathsf{love}(x, y)$, $\mathsf{john}(x)$, $\mathsf{mary}(y)$, l_2 : $\mathsf{always}(h_2)$ $h_1 \geq l_1, h_1 \geq l_2, h_2 \geq l_1$
arg: –

According to (3), $h_1 \geq l_2, l_2 > h_2$ (because h_2 appears inside a formula labelled l_2) and $h_2 \geq l_1$. Consequently $h_1 \neq l_1$ and therefore the only possible disambiguation is $h_1 \to l_2, h_2 \to l_1$. This leads to the semantics $\mathsf{john}(x) \wedge \mathsf{mary}(y) \wedge \mathsf{always}(\mathsf{love}(x, y))$.

1.3 Separating scope and predicate argument information

A central aspect of (Kallmeyer and Joshi, 1999, 2002) is the idea that the contribution of a quantifier is separated into a scope and a predicate argument part: Quantifiers have a set of two elementary trees and mulicomponent TAGs are used. An auxiliary tree consisting of a single node is linked to the scope part of the semantics, while an initial tree is linked to the predicate argument part. E.g., consider (4).

(4) every dog barks

Fig. 2 shows the syntactic analysis of (4) in this framework. The elementary semantic rep-

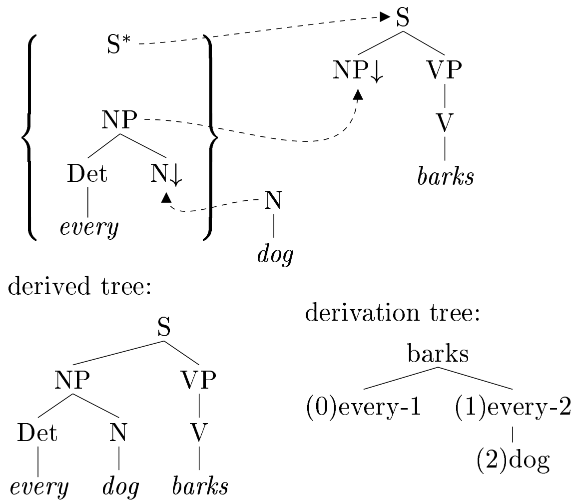

Figure 2: Syntactic analysis of (4)

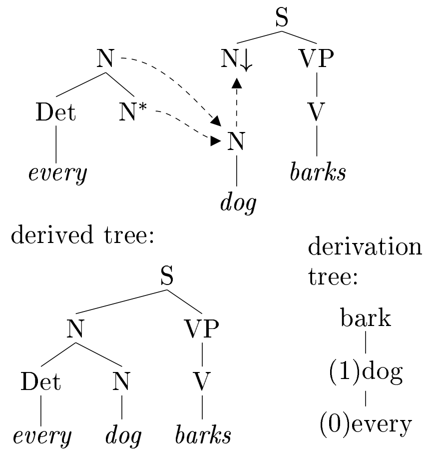

Figure 3: New syntactic analysis of (4)

resentations are shown in (5).

(5)

$l_1 : \mathsf{bark}(x_1)$ $l_1 \leq h_1$	$l_2 : \mathsf{every}(x, h_2, h_3)$ $s_1 \leq h_3$
arg: x_1	arg: s_1

$l_3 : p_1(x)$ $l_3 \leq h_2$	$q_1 : \mathsf{dog}$
arg: p_1	arg: –

The scope part of the quantifier (second representation in (5)) introduces a proposition containing the quantifier, its variable and two holes for its restrictive and nuclear scope. The proposition this semantic representation is applied to (variable s_1) is in the nuclear scope of the quantifier ($s_1 \leq h_3$). The predicate argument part (third representation in (5)) introduces a proposition $p_1(x)$ where p_1 will be the noun predicate dog. This proposition is in the restrictive scope of the quantifier ($l_3 \leq h_2$). The values for the argument variables are $x_1 \to x, s_1 \to l_1, p_1 \to q_1$ which leads to (6).

(6)

$l_1 : \mathsf{bark}(x)$, $l_2 : \mathsf{every}(x, h_2, h_3)$, $l_3 : \mathsf{dog}(x)$ $l_1 \leq h_1$, $l_1 \leq h_3$, $l_3 \leq h_2$
arg: –

The only disambiguation is $h_1 \to l_2, h_2 \to l_3, h_3 \to l_1$ which leads to the semantics $\mathsf{every}(x, \mathsf{dog}(x), \mathsf{bark}(x))$.

To account for cases with more than one quantifier, a restricted use of multiple adjunctions (for the scope parts) is necessary.

2 Quantifiers as adjuncts

2.1 The problem

The approach of Kallmeyer and Joshi is problematic in cases where quantifiers are treated as adjuncts. Such an analysis is proposed for English in (Hockey and Mateyak, 2000) and for French in (Abeillé, 1991) and it is even adopted in the XTAG grammar (XTAG Research Group, 1998) and in the French TAG implemented at Paris 7 (Abeillé et al., 2000).

According to this analysis, the noun in (4) is first added to the verb by substitution, and then the quantifier is adjoined to the noun (see Fig. 3). In the derivation tree, there is no link between the quantifier and the verb. However, the variable introduced by the quantifier is an argument of the verb, and, furthermore, the proposition introduced by the verb is part of the nuclear scope of the quantifier. (This is why l_1 was assigned to s_1 in (5).) Therefore, for semantics, a link between the quantifier and the verb is needed.

The use of a second elementary tree for the scope part that is adjoined to the whole sentence would require non-local MCTAG since the quantifier is not adjoined to the verb but to the noun. Non-local MCTAG is much more powerful than TAG and a solution using TAG or a mildly context-sensitive TAG variant is preferable. Therefore I will not adopt the idea of

(Kallmeyer and Joshi, 2002) to separate the contribution of a quantifier into two parts. An advantage of not doing so is that multiple adjunctions are not necessary. Multiple adjunctions are problematic because in combination with multicomponent derivations, they extend the generative power of the grammar, and therefore their use needs to be restricted.

2.2 Enriching the derivation tree

If one adopts the syntactic analysis in Fig. 3 and one wants to retain the semantic analysis given in (5) for quantifiers, the quantifier *every* must have access to both elementary semantic representations, the one of *barks* and the one of *dog*. This means that the derivation tree is too restrictive, it does not provide the links one needs for semantics. On the other hand, the way syntactic elements are put together in a derivation reflects predicate-argument relations, i.e., seems still to provide the dependencies semantics should be based on. Therefore I propose to keep the idea of using the derivation tree for semantics, but to enrich the derivation tree in order to obtain the semantic links one needs.

The basic intuition is as follows: In Fig. 3 for example, the quantifier is adjoined to the root of the initial tree for *dog*, and consequently in the derived tree it will be a direct neighbour not just of the elementary tree for *dog* but also of the one for *barks* (see Fig. 4 where the elementary tree of the quantifier is marked by a triangle). Therefore there is a syntactic link between the quantifier and the verb and this should enable the quantifier to have semantic access to the verb.[1] For this reason, in the case of an adjunction at a root node of some elementary γ, the adjoined tree is not only connected to γ but also to the tree to which γ was added in some previous derivation step. The enriched derivation structure used for semantics is called *e-derivation structure* for short. The e-derivation structure of (4) is shown in Fig. 4. The additional link, i.e., the link that is not part of the derivation tree, is depicted as a dotted edge.

As mentioned above, trees that are neighbours of each other in the derived tree, should be related in the e-derivation structure. However, it is not just neighbourhood in the derived tree that determines whether two elementary trees are linked in the e-derivation structure. If this was the case, such a link (e.g., the one between *dog* and *barks*) would be destroyed when adjoining at the root of the lower tree. Consequently, the e-derivation structure would not be monotonic with respect to derivation. Since semantics will be defined based on this structure, non-monotonicity is something one definitely wants to avoid.

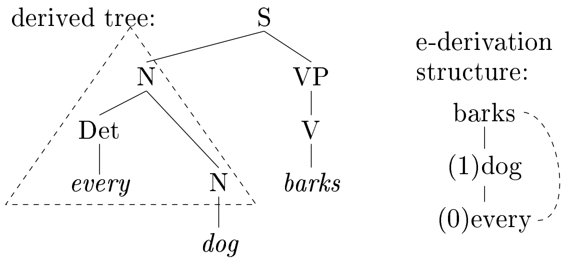

Figure 4: E-derivation structure for (4)

Therefore I define the e-derivation structure as follows: all edges occurring in the derivation tree are also e-derivation links. Furthermore, two nodes labelled γ and β are linked if in the derivation tree there are nodes $\gamma', \beta_1, \dots, \beta_n$ such that γ' is daughter of γ, β_1 daughter of γ' with position 0 (adjunction at the root), β_{i+1} is daughter of β_i with position 0 ($1 \leq i < n$) and $\beta_n = \beta$. (The definition applies to the derivation of (4) with γ being the elementary tree of *barks*, γ' being the elementary tree of *dog* and $\beta_1 = \beta_n = \beta$ being the one of *every*.)

The e-derivation structure is a graph, not necessarily a tree. This is one of the differences compared to the meta-level derivation structure proposed in (Dras et al., 2000) that is also intended to be a "more semantic" dependency structure than the original derivation tree.

It is important to emphasize that the e-derivation structure is a way of making information about shared nodes (or in an FTAG shared feature structures) explicit that is already present in the original derivation tree. In a way, it is even more appropriate as a derivation structure than the original TAG derivation tree since it reflects all feature structure unifications performed between elementary trees.

Since the feature structure unifications in the

[1]In a feature-structure based TAG (FTAG, Vijay-Shanker and Joshi 1988), this link is even more explicit, since a unification of feature structures from all three trees (here *barks*, *every* and *dog*) takes place.

derived tree seem to give the additional links one needs, one might ask whether it might be more appropriate to do semantics in the feature structures on the derived tree. However, it is probably not possible to do semantics using only the feature unification mechanism. One also needs to take into account the way two elementary trees combine, i.e., whether they combine by substitution or adjunction. In a substitution step, an argument is added, while in an adjunction step, a new function (e.g., a modifier or a complement clause taking verb) is applied to an already derived argument. Therefore, depending on the syntactic combination, different semantic operations would be performed on the semantic representations in the feature structures of the corresponding nodes. Such an approach is for example pursued in (Frank and van Genabith, 2001). It makes use not only of the information available in the derived tree but also of information about how the elementary trees were put together, i.e., of the derivation tree. Compared to this, an advantage of the approach proposed here is that semantics is based only on the enriched derivation tree and does not need to go back and to use both, the derived tree and the derivation tree. In particular, it can abstract away from the concrete shape of the elementary trees, i.e., it uses only links between elementary trees and not links between nodes. This seems appropriate considering that one of the guiding linguistic principles of LTAG is semantic minimality, i.e. that the semantics of elementary trees is non-decomposable. Furthermore, as already mentioned, the e-derivation structure is more than just a technical way to obtain the semantic links one needs. It is also a way to make explicit all links performed between elementary trees during derivation.

For the semantics of the quantifier adjunct analysis of (4), in contrast to (Kallmeyer and Joshi, 2002), the quantifier contribution is no longer separated into two parts:

(7)

$l_2 : \mathsf{every}(x, h_2, h_3)$, $l_3 : p_1(x)$ $s_1 \leq h_3$, $l_3 \leq h_2$
arg: s_1, p_1

Using the semantics in (5) for *dog* and *barks*, with the links in the e-derivation structure, the assignments are $x_1 \to x, s_1 \to l_1, p_1 \to q_1$ which gives (6), the semantics already derived in 1.3.

This analysis of quantifiers allows to generate underspecified semantic representations for quantifier scope ambiguities and it correctly allows quantifiers embedded into PPs to have wide scope (Kallmeyer, 2002).

Note that the problem discussed here is specific for the quantifier adjunct analysis. Though this analysis is sometimes preferred and it is even used in XTAG and the French TAG, the problem might be overcome by adopting an analysis where first the quantifier is substituted into the tree of the verb and then the noun substituted into the quantifier tree. The problems discussed in the following sections are of a more general nature, in their case, there are no alternative syntactic analyses that allow to avoid them.

3 Unbounded dependencies in embedded interrogatives

The second problem treated in this paper are unbounded dependencies in embedded interrogatives as in (8). An adequate semantic analysis of (8) should have the structure (9).

(8) Mary wondered who Peter thought John said Bill liked

(9) $\mathsf{wonder}(\mathsf{mary}, \mathsf{who}(x,$
$\quad\mathsf{think}(\mathsf{peter}, \mathsf{say}(\mathsf{john}, \mathsf{like}(\mathsf{bill}, x)))))$

The embedding of $\mathsf{think}(\ldots)$ into $\mathsf{who}(x, \ldots)$ is a scope relation while the other embeddings are predicate argument relations. Both should be part of an adequate semantic representation and I expect the structure underlying semantics to provide all the links necessary for scope and for the predicate argument structure. (In the case of scope ambiguities, scope can of course be partly unspecified.)

In order to obtain the relation between think and who, *think* must be connected either to *who* or to *like* (if the semantic representation of *like* contains a part that corresponds to its 'moved' wh-part). Fig. 5 shows the classical TAG analysis of (8), following (Kroch, 1987). The derivation tree does not contain the necessary links. The e-derivation structure however provides an additional link between *like* and *think*. In the following I will provide a semantics for (8) based on the e-derivation structure. The semantic representations are shown in (10).

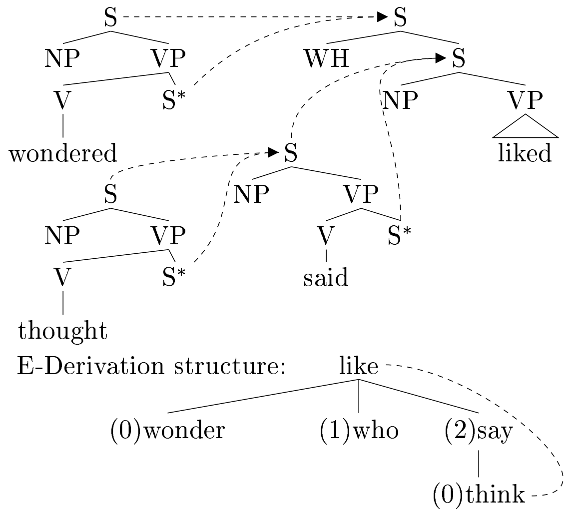

Figure 5: TAG derivation of (8)

(10)

$$\begin{array}{|l|}\hline \langle l_2,(1)\rangle : g_2,\ l_5 : \mathsf{like}(x_4,x_5) \\ \langle h_2,(1)\rangle \geq l_2, g_2 \geq h_5,\ h_5 \geq l_5 \\ \hline \text{arg: } \langle g_2,(2)\rangle, \langle x_4,(21)\rangle, \langle x_5,(1)\rangle \\ \hline \end{array}$$

$$\begin{array}{|l|}\hline l_1 : \mathsf{wonder}(x_1,g_1) \\ h_1 \geq l_1 \\ \hline \text{arg: } \langle x_1,(1)\rangle, g_1 \\ \hline \end{array} \quad \begin{array}{|l|}\hline l_6 : \mathsf{who}(x,h_6) \\ g_5 \geq l_6, h_6 \geq s_1 \\ \hline \text{arg: } g_5, s_1 \\ \hline \end{array}$$

$$\begin{array}{|l|}\hline l_3 : \mathsf{think}(x_2,g_3) \\ h_3 \geq l_3 \\ \hline \text{arg: } \langle x_2,(1)\rangle, g_3 \\ \hline \end{array} \quad \begin{array}{|l|}\hline l_4 : \mathsf{say}(x_3,g_4) \\ h_4 \geq l_4 \\ \hline \text{arg: } \langle x_3,(1)\rangle, g_4 \\ \hline \end{array}$$

The semantic representation of *like* contains a scope part that is responsible for scope of the wh-element and a predicate argument part. Complement clause taking verbs are treated as having a hole variable that will be filled by the top-most hole of the embedded clause.

So far, I have only linked argument variables to substitution nodes. Now I am adding two other possibilities to link nodes in the elementary trees to parts of the semantic representation: Firstly, I allow links between nodes and elements (holes, labels or free variables) that are potential values for argument variables of other semantic representations. When adding something at the node in question, the elements linked to it are used as values for the argument variables of the new semantic representation. This additional linking mechanism is needed to make sure that the wh-element has wide scope with respect to the propositions l_2 and l_5. Secondly, argument variables may be linked to internal nodes which means that they must be filled by anything (added to some elementary tree) adjoined to that node.

With these two additional links, one already knows that for g_5 and s_1, $g_5 \to h_2$ and $s_1 \to l_2$ must be chosen because these are linked to the position the wh-element is added to. Furthermore, the value of g_2 must be either h_3 or h_4 because these are the two holes introduced by *say* (adjoined at position (2) to *like*) and by *think* (adjoined to *say*).

I suppose that a hole cannot appear in more than one place inside the formulas of a semantic representation. This makes sense since holes will later become propositional labels and a propositional label is meant to point at a (sub-)formula. Two different (sub-)formulas can never have the same label. Consequently, the same hole being part of different formulas should be forbidden.

There is only one possible assignment for the propositional and hole variables: Possible values for g_1, g_2, g_3 and g_4 are: $g_1 : h_2, h_5$, $g_2 : h_3, h_4$, $g_3 : h_2, h_4, h_5$, $g_4 : h_2, h_5$. Since the values of these four hole variables must be different (because they all occur inside some formula), one can already conclude $g_2 \to h_3$ and $g_3 \to h_4$. Furthermore, $g_4 \to h_2$ leads to a contradiction, namely $h_2 > h_3 > h_4 > h_2$. Consequently $g_1 \to h_2$ and $g_4 \to h_5$.

(11)

$$\begin{array}{|l|}\hline l_1 : \mathsf{wonder}(x_1,h_2),\ l_2 : h_3, \\ l_3 : \mathsf{think}(x_2,h_4),\ l_4 : \mathsf{say}(x_3,h_5), \\ l_5 : \mathsf{like}(x_4,x),\ l_6 : \mathsf{who}(x,h_6), \\ h_1 \geq l_1,\ h_2 \geq l_2, l_1 \geq h_2,\ h_3 \geq l_3, \\ h_4 \geq l_4,\ h_5 \geq l_5, h_2 \geq l_6, h_6 \geq l_2 \\ \hline \text{arg: } x_1, x_2, x_3, x_4 \\ \hline \end{array}$$

As a result, we obtain (11) (with $x_5 \to x$), and the only disambiguation is $h_1 \to l_1, h_2 \to l_6, h_3 \to l_3, h_4 \to l_4, h_5 \to l_5, h_6 \to l_2$. This leads to the desired semantics $\mathsf{wonder}(x_1, \mathsf{who}(x, \mathsf{think}(x_2, \mathsf{say}(x_3, \mathsf{like}(x_4, x)))))$.

4 Other problematic cases

In this section, I consider two other problems for derivation tree based LTAG semantics that are often mentioned in the literature. I just sketch the way these problems might be solved without going into the details of the semantic analyses.

4.1 Multiple modifiers

The problem of multiple modifiers is often discussed as an example where the TAG derivation tree does not give the semantic dependencies one needs (see, e.g., Schabes and Shieber, 1994; Rogers, 2002).

(12) roasted red pepper

In a standard TAG analysis of examples as (12) only the modifier that directly precedes the noun is linked to the noun in the derivation tree. The correct semantic dependencies however, are links between *pepper* and each of the adjectives.

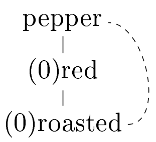

Figure 6: E-derivation structure for (12)

The e-derivation structure provides the missing link between the first adjective and the noun. In contrast to the solutions proposed so far to the problem of multiple modifiers, the e-derivation structure still contains the link between the two adjectives. In some cases this link is in fact needed, namely in those cases where the meaning of the left adjective is not purely intersective but depends on the semantics of the right adjective. This is the case in (13):

(13) a big newborn baby

The baby is neither big in some absolute sense nor big for a baby but big for a newborn baby.

4.2 Raising verbs in embedded clauses

Another problem often mentioned is the analysis of raising verbs. In LTAG, raising verbs are added by adjunction, similar to adverbs. Consider (14), taken from (Rambow et al., 1995):

(14) Paul Peter claims Mary seems to love

As pointed out by Rambow et al. (1995), the derivation tree for (14) (see Fig. 7 for the derivation) does not provide the dependencies one wants to obtain since *claim* and *seem* both adjoin to *love*. There is no link between *seem* and *claim*. In this case, the e-derivation structure does not add any links since none of the trees is adjoined to a root node.

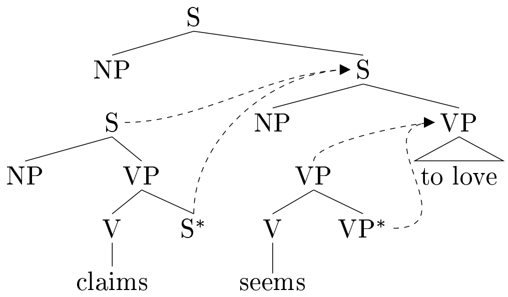

Figure 7: Derivation of (14)

This problem, however, is in fact not a problem, at least not for semantics. It can be overcome by treating raising verbs semantically as adverbs. Consider the example (15).

(15) Paul Peter claims Mary apparently loves

(15) shows semantic dependencies between *claim* and *love* and between *love* and *apparently* but not between *claim* and *apparently*. The similarity to (14) suggests that (14) might be analyzed treating *seems* as a kind of adverb modifying *love*.

5 Conclusion

I have shown in this paper that, in spite of some mismatches between TAG derivation trees and dependency structures, it is possible to build a semantics in the TAG framework based on the derivation trees. The crucial point is that, instead of using the derivation tree as basis for semantics, I am using an enriched structure called e-derivation structure that can be easily obtained from the derivation tree. On the basis of this structure it is possible to account for the semantics of quantifiers and of unbounded dependencies, phenomena that are problematic for the assumption that derivation trees provide the right dependency structure to use for semantics.

Acknowledgements

For valuable discussions of the work presented here I am grateful to Aravind Joshi, Chung-hye Han, Sylvain Kahane, James Rogers and Maribel Romero. Furthermore, I would like to thank

two anonymous reviewers for their very helpful comments.

References

Anne Abeillé, Marie-Hélène Candito, and Alexandra Kinyon. 2000. The current status of FTAG. In *Proceedings of TAG+5*, pages 11–18, Paris.

Anne Abeillé. 1991. *Une grammaire lexicalisée d'arbres adjoints pour le français: application à l'analyse automatique.* Ph.D. thesis, Université Paris 7.

Johan Bos. 1995. Predicate logic unplugged. In Paul Dekker and Martin Stokhof, editors, *Proceedings of the 10th Amsterdam Colloquium*, pages 133–142.

Marie-Hélène Candito and Sylvain Kahane. 1998. Can the TAG Derivation Tree represent a Semantic Graph? an Answer in the Light of Meaning-Text Theory. In *Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks, IRCS Report 98–12*, pages 25–28, University of Pennsylvania, Philadelphia.

Ann Copestake, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 1999. Minimal Recursion Semantics. An Introduction. Manuscript, Stanford University.

Mark Dras, David Chiang, and William Schuler. 2000. A Multi-Level TAG Approach to Dependency. In *Proceedings of the Workshop on Linguistic Theory and Grammar Implementation, ESSLLI 2000*, pages 33–46, Birmingham, August.

Anette Frank and Josef van Genabith. 2001. GlueTag. Linear Logic based Semantics for LTAG – and what it teaches us about LFG and LTAG. In Miriam Butt and Tracy Holloway King, editors, *Proceedings of the LFG01 Conference*, Hong Kong.

Beth Ann Hockey and Heather Mateyak. 2000. Determining Determiner Sequencing: A Syntactic Analysis for English. In Anne Abeillé and Owen Rambow, editors, *Tree Adjoining Grammars: Formalisms, Linguistic Analyses and Processing*, pages 221–249. CSLI.

Aravind K. Joshi and Yves Schabes. 1997. Tree-Adjoning Grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, pages 69–123. Springer, Berlin.

Aravind K. Joshi and K. Vijay-Shanker. 1999. Compositional Semantics with Lexicalized Tree-Adjoining Grammar (LTAG): How Much Underspecification is Necessary? In H. C. Blunt and E. G. C. Thijsse, editors, *Proceedings ot the Third International Workshop on Computational Semantics (IWCS-3)*, pages 131–145, Tilburg.

Laura Kallmeyer and Aravind K. Joshi. 1999. Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG. In Paul Dekker, editor, *12th Amsterdam Colloquium. Proceedings*, pages 169–174, Amsterdam, December.

Laura Kallmeyer and Aravind K. Joshi. 2002. Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG. *Journal of Language and Computation.* To appear.

Laura Kallmeyer. 2002. Using an Enriched TAG Derivation Structure as Basis for Semantics. In *Proceedings of TAG+6 Workshop*, pages 127 – 136, Venice, May.

Anthony S. Kroch. 1987. Unbounded dependencies and subjacency in a Tree Adjoining Grammar. In A. Manaster-Ramer, editor, *Mathematics of Language*, pages 143–172. John Benjamins, Amsterdam.

Owen Rambow, K. Vijay-Shanker, and David Weir. 1995. D-Tree Grammars. In *Proceedings of ACL*.

James Rogers. 2002. One More Perspective on Semantic Relations in TAG. In *Proceedings of TAG+6 Workshop*, pages 118 – 126, Venice, May.

Yves Schabes and Stuart M. Shieber. 1994. An Alternative Conception of Tree-Adjoining Derivation. *Computational Linguistics*, 20(1):91–124, March.

Stuart M. Shieber and Yves Schabes. 1990. Synchronous Tree-Adjoining Grammars. In *Proceedings of COLING*, pages 253–258.

K. Vijay-Shanker and Aravind K. Joshi. 1988. Feature structures based tree adjoining grammar. In *Proceedings of COLING*, pages 714–719, Budapest.

XTAG Research Group. 1998. A Lexicalized Tree Adjoining Grammar for English. Technical Report 98–18, Institute for Research in Cognitive Science, Philadelphia.