LTAG Semantics with Semantic Unification

Laura Kallmeyer
TALaNa-Lattice
UFRL, University Paris 7
2 place Jussieu, Case 7003,
75251 Paris Cedex 05
France
lkallmey@linguist.jussieu.fr

Maribel Romero
Department of Linguistics
610, Williams Hall
University of Pennsylvania
Philadelphia, PA, 19104-6305
U.S.A.
romero@ling.upenn.edu

Abstract

This paper sets up a framework for LTAG (Lexicalized Tree Adjoining Grammar) semantics that brings together ideas from different recent approaches addressing some shortcomings of TAG semantics based on the derivation tree. Within this framework, several sample analyses are proposed, and it is shown that the framework allows to analyze data that have been claimed to be problematic for derivation tree based LTAG semantics approaches.

1 Introduction

An LTAG (Joshi and Schabes, 1997) consists of a finite set of *elementary* trees associated with lexical items. From these trees, larger trees are derived by substitution (replacing a leaf with a new tree, a so-called *initial tree*) and adjunction (replacing an internal node with a new tree, a so-called *auxiliary tree*).

The elementary trees of an LTAG 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. These linguistic properties of elementary trees are formulated in the *Condition on Elementary Tree Minimality (CETM)* from (Frank, 1992).

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 adjunctions. Each edge in the derivation tree stands for an adjunction or a substitution. The edges are equipped with Gorn addresses of the nodes where the substitutions/adjunctions take place.[1] See for example the derivation of (1) in Fig. 1.

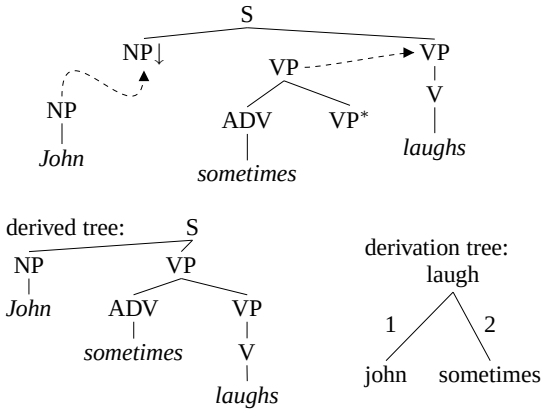

Figure 1: TAG derivation for (1)

(1) John sometimes laughs

Taking into account the minimality of elementary trees and the fact that derivation steps in TAG correspond to predicate-argument applications, it seems appropriate to base LTAG semantics on the derivation tree (Candito and Kahane, 1998; Joshi and Vijay-Shanker, 1999; Kallmeyer and Joshi, 2003). However, it has been observed that in some cases this is problematic since the derivation tree does not provide enough information to correctly construct the desired semantic dependencies.

The goal of this paper is to bring together ideas from several recent approaches in order to develop a general framework for LTAG semantics that allows us to compute semantic representations on the derivation tree, overcoming some otherwise problematic cases. Within this framework we then sketch several sample analyses.

2 Previous approaches to LTAG semantics

The data that are claimed to be the most problematic for derivation tree based LTAG semantics (see (Rambow et

[1] The root has the address 0, the jth child of the root has address j and for all other nodes: the jth child of the node with address p has address $p \cdot j$.

TAG+7: Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms.
May 20-22, 2004, Vancouver, BC, CA.
Pages 155-162.

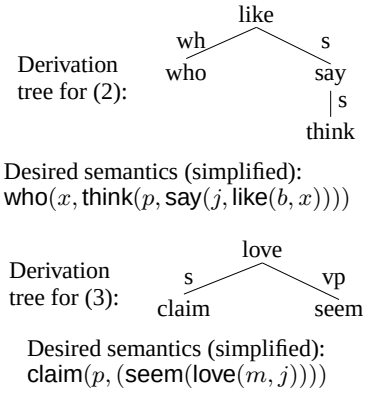

Figure 2: Problematic derivation trees for semantics

al., 1995; Dras et al., 2004; Frank and van Genabith, 2001; Gardent and Kallmeyer, 2003)) are long-distance wh-movements as in (2) and interactions of attitude verbs and raising verbs or adverbs as in (3).

(2) Who does Paul think John said Bill liked?

(3) a. Mary, Paul claims John seems to love
b. Paul claims Mary apparently loves John

The problem of (2) is that in the LTAG analysis, *who* is substituted into the wh-NP node of *like*, *say* is adjoined to the lower S node of *like* and *think* adjoins to *say*. Consequently, in the derivation tree (see Fig. 2), there is neither a link between *who* and *think* nor a link between *like* and *think*.[2] But in the semantics, we want the *think* proposition to be the scopal argument of the wh-operator, i.e., a link between *who* and *think* must be established. This can be done via the semantics of *like* but at least some possibility to link *like* to *think* is necessary. In (3), *claim* and *seem* (or *apparently* resp.) adjoin to different nodes in the *love* tree, i.e., they are not linked in the derivation tree. But the propositional argument of *claim* is the *seems* (*apparently* resp.) proposition. This case however is less hard than (2) since one can choose the semantics of *like* in such a way that the desired scope orders are obtained without a direct link between the embedding attitude verb and the embedded raising verb (adverb resp.). A semantics in the (Kallmeyer and Joshi, 2003) framework is possible here. Example (2) however poses a serious problem for derivation tree based approaches.

Several proposals have been made to avoid the problems that arise when doing semantics based on the derivation tree:

Instead of using the derivation tree for semantics, one could try to compute semantics based on the derived tree. Such an approach is pursued in (Frank and van Genabith, 2001). However, their approach 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 information available in the derivation tree. Therefore, in a sense, their semantics is based on both, derived and derivation tree. Considering that one of the guiding linguistic principles of LTAG is semantic minimality of elementary tree, i.e. that the semantics of elementary trees is non-decomposable, it is more appropriate to link semantic representations to whole elementary trees and to abstract away (at least to a certain degree) from the concrete shape of the elementary trees. This amounts to linking semantic representations to nodes in the derivation tree.

An alternative proposal for computing semantics only on the derivation tree is to enrich the derivation tree with additional links as in (Kallmeyer, 2002a; Kallmeyer, 2002b). In this approach, the derived tree needs not be considered for computing semantics. The problem with this proposal is that sometimes it is not clear which link one has to follow in order to find the value for some semantic variable. Therefore additional rules for ordering the links for semantic computation are needed. The result is a rather complex machinery in order to obtain the dependencies needed for semantics.

More recently, (Gardent and Kallmeyer, 2003) propose to use the feature unification mechanism in the syntax, i.e., in the derived tree, in order to determine the values of semantic arguments. The underlying observation is that whenever a semantic link in the derivation tree is missing, it is either a) a link between trees attaching to different nodes in the same tree (see(3)), i.e., attaching to nodes that can share features inside an elementary tree, or b) a link between trees γ_1 and γ_2 such that γ_2 adjoins to the root of a tree that (adjoins to the root of a tree that ...) attaches to some node μ in γ_1 (see (2)). In this case, indirectly, the top of μ and the top of the root of γ_2 unify and thereby features can be shared. This approach works in the problematic cases and it has the advantage of using a well-defined operation, unification, for semantic computation. But it has the disadvantage of using the derived tree for semantics even though semantic representations are assigned to whole elementary trees (i.e., to nodes in the derivation tree) and not to nodes in the derived tree. Furthermore, the feature structures needed for semantics are slightly different form those used for syntax since they contain semantic variables and labels as possible feature values. Consequently, the number of feature structures is no longer finite (in contrast to feature-based TAG (FTAG) as defined in (Vijay-Shanker and Joshi, 1988)) and therefore the generative capacity of the formalism is extended. In other words, a more powerful formalism is used for syntax just because it is needed for the specific

[2]For the sake of readability, we use names np, vp, r for *root*, f for *foot*, ... for the node positions instead of the usual Gorn adresses.

semantic features.[3]

In order to separate more neatly between syntax with feature structures linked to nodes in the derived tree and semantics where semantic representations are linked to nodes in the derivation tree, we propose in the following to incorporate semantic feature structures in the derivation tree. Formally, this means just extracting the semantic features used in (Gardent and Kallmeyer, 2003) from the derived trees and putting them in a semantic feature structure linked to the semantic representation of the tree in question. Of course one still has to link semantic features to specific node positions in the elementary tree, e.g., in order to make sure that syntactic argument positions get correctly linked to the corresponding semantic arguments.

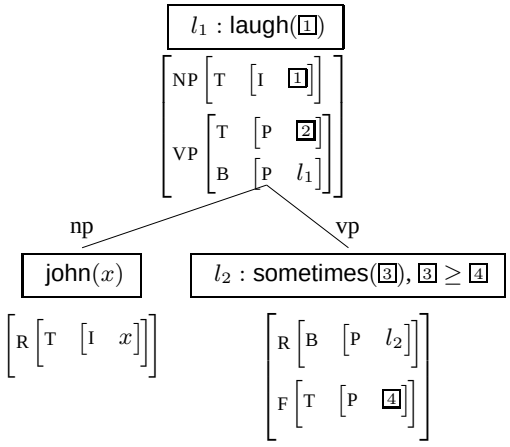

Figure 3: Semantic representations for (1) *John sometimes laughs*

3 LTAG semantics with semantic unification

3.1 Semantic feature structures

Semantic representations are as defined in (Kallmeyer and Joshi, 2003) except that they do not have argument variables: they consist of a set of formulas (typed λ-expressions with labels) and a set of scope constraints. A scope constraint is an expression $x \geq y$ where x and y are propositional labels or propositional variables (these last correspond to the holes in (Kallmeyer and Joshi, 2003)). Each semantic representation is linked to a semantic feature structure. Semantic feature structures are typed feature structures, the type of the whole feature structure is sem. The definition of the feature structures is as follows:

- a feature structure of type sem consists of features 0 (the root position), 1, 2, ..., 11, 12, ... for all node positions that can occur in elementary trees (finite for each TAG), the values of these features are of type tb (for 'top-bottom')
- a feature structure of type tb consists of a T and a B feature (top and bottom) whose values are feature structures of type $bindings$
- a feature structure of type $bindings$ consists of a feature I whose values are individual variables, a feature P whose values are propositional labels, etc.

3.2 Semantic unification

Semantic composition consists only of feature unification. It corresponds to the feature unifications in the syntax that are performed during substitutions and adjunctions and the final top-bottom unifications in the derived tree. In the derivation tree, elementary trees are replaced by their semantic representations plus the corresponding semantic feature structures. Then, for each edge in the derivation tree from γ_1 to γ_2 with position p:

- The top feature of position p in γ_1 and the top feature of the root position in γ_2, i.e., the feature structures $\gamma_1.p.\text{T}$ and $\gamma_2.0.\text{T}$ are identified,
- and if γ_2 is an auxiliary tree, then the bottom feature of the foot node of γ_2 and the bottom feature of position p in γ_1, i.e., (if f is the position of the foot node in γ_2) the feature structures $\gamma_1.p.\text{B}$ and $\gamma_2.f.\text{B}$ are identified.

Furthermore, for all γ in the derivation tree and for all positions p in γ such that there is no edge from γ to some other tree with position p: the T and B features of $\gamma.p$ are identified.

By these unifications, some of the variables in the semantic representations get values. In the end, after having performed these unifications, the union of all semantic representations is built. The result is an underspecified representation.[4]

3.3 A sample derivation

As an example consider the analysis of (1): Fig. 3 shows the semantic representations and the semantic feature structures of the three elementary trees involved in the derivation.

[3] A similar approach is (Stone and Doran, 1997) where, as in (Gardent and Kallmeyer, 2003), each elementary tree has a flat semantic representation, the semantic representations are conjoined when combining them and variable assignments are done by unification in the feature structures on the derived tree. But there is no underspecification, and the approach is less explicit than (Gardent and Kallmeyer, 2003).

[4] For combining feature structure, we adopt an operational way in this paper because this is general practice in LTAG. I.e., unification is an operation on actual structures. Viewing feature structures as descriptions and thinking of unification as finding a consistent model (see, e.g., (Johnson, 1994)), is of course possible as well. But then one needs additional constraints that reflect the identifications performed during substitution, adjunction and top-bottom feature structure unification.

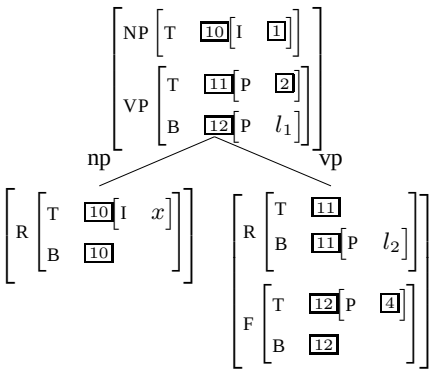

Figure 4: Semantic unification for (1)

The different unifications lead to the feature value identities in Fig. 4. This gives the identities $\boxed{1} = x$, $\boxed{2} = l_2$, and $\boxed{4} = l_1$, which results in the following semantic representation:

(4) $l_1 : \mathsf{laugh}(x), \mathsf{john}(x), l_2 : \mathsf{sometimes}(\boxed{3}), \boxed{3} \geq l_1$

In the end, appropriate disambiguations must be found. These are assignments for the remaining variables, i.e., functions that assign propositional labels to propositional variables, respecting the scope constraints (Kallmeyer and Joshi, 2003). The disambiguated representation is then interpreted conjunctively. (4) has only one disambiguation, namely $\boxed{3} \rightarrow l_1$. This leads to $\mathsf{john}(x) \wedge \mathsf{sometimes}(\mathsf{laugh}(x))$.

4 Alternative ways of obtaining scope constraints

Instead of stating explicit scope constraints of the form $x \geq y$, one could imagine two other possibilities: either i) not using any scope constraints at all and obtaining scope by identifying propositional variables and propositional labels by unification during the derivation, or ii) obtaining scope constraints from the final top-bottom unification instead of stating them explicitely, i.e., not doing real top-bottom unification but adding instead a constraint top $\geq$ bottom whenever a node has not been used for attaching other elementary trees. These alternatives are illustrated in Fig. 5 and 6.

Possibility i) has the obvious problem that is does not allow for underspecified representations, which means that in cases of scope ambiguities the number of representations one would have to generate would explode. Possibility ii) looks more interesting. In Fig. 5 for example, the B feature of position 2 in *laugh* is unified with the (empty) B feature of position 2 in *sometimes* so that in the result, there is a node with T [P $\boxed{3}$] and B [P l_1]. From this node, the desired scope constraint $\boxed{3} \geq l_1$ can

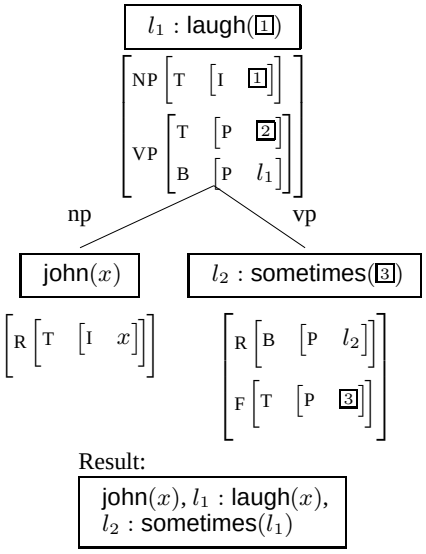

Figure 5: Alternative i): Analysis of (1) without scope constraints

be obtained. One problem with ii) is that in some cases one might need the original final top-bottom unification, so one would have to distinguish between cases where a scope constraint has to be added (these are perhaps the cases of P features) and cases where usual unification is done. But even more problematic is that in some cases, it is not possible to obtain all scope constraints one needs by the final top-bottom mechanism. Examples are cases where two quantifier scope parts attach to the same node as in (5).

(5) someone likes everybody

Following (Kallmeyer and Joshi, 2003), we suppose that the contribution of a quantifier consists of an NP initial tree (the predicate argument part) and a separate auxiliary tree with just one S node (the scope part). The analysis of (5) with possibility ii) is sketched in Fig. 7. The scope constraints one wants to obtain are 1) those that place the proposition coming with the noun in the restriction of the quantifiers, i.e., $\boxed{4} \geq l_3$ and $\boxed{8} \geq l_5$, 2) those that place the *like* proposition in the nuclear scope of the quantifiers, i.e., $\boxed{5} \geq l_1$ and $\boxed{9} \geq l_1$, and 3) those that limit the scope of the quantifier inside the sentence the quantifier attaches to, i.e., $\boxed{1} \geq l_2$ and $\boxed{1} \geq l_4$.[5] For 1) and 2), corresponding top and bottom feature have to be put on some node, in Fig. 7 they are on positions N and L (for the lexical anchor) of the NP tree. However, this is very arbitrary, they are not really related to these nodes. Therefore, it is much more appropriate to state the constraints in a general way

[5]The last constraints are important to make sure that in examples as *Mary thinks John likes everybody* the embedded quantifier cannot take scope over *thinks*.

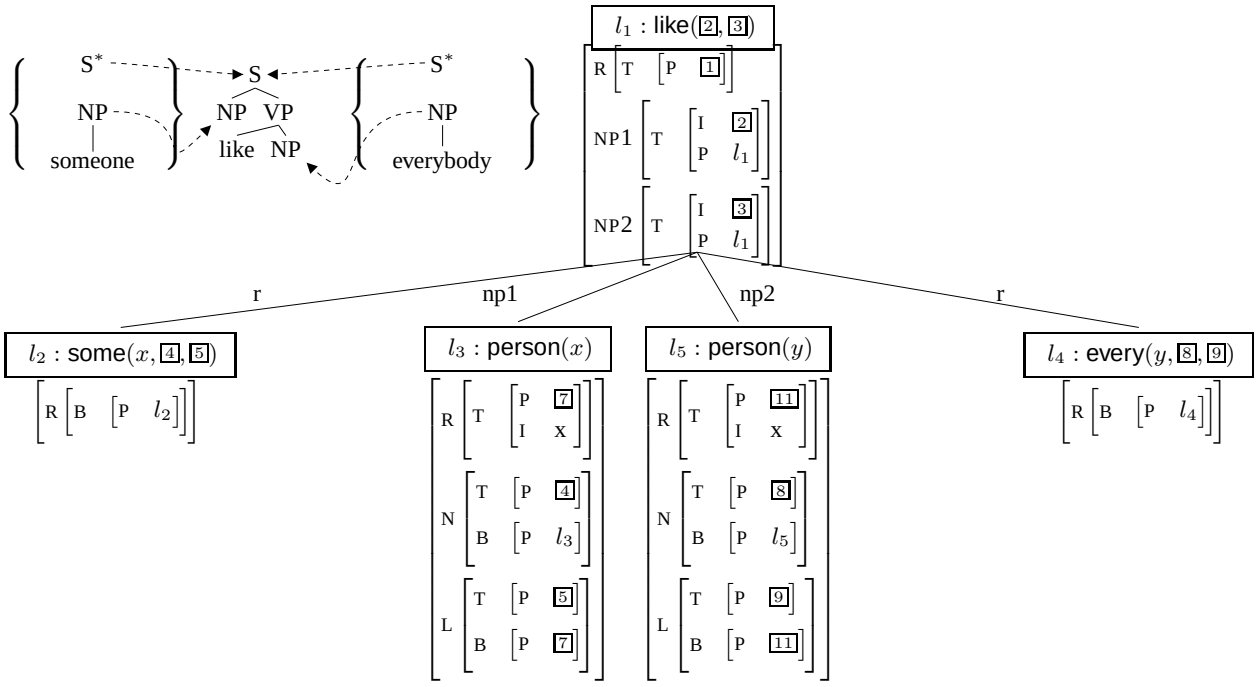

Figure 7: Problematic case for possibility ii): Analysis of (5)

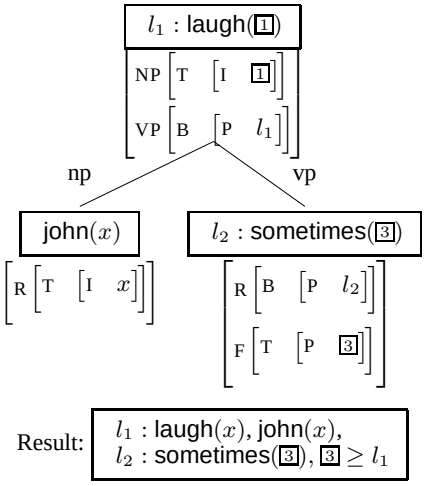

Figure 6: Alternative ii): scope constraints from top-bottom unification

and to link them only to the semantic representation without linking them to any node. The constraints 3) have to come from the scope tree, i.e., they have to be linked to its root since this is the only node in these trees. But this is not possible since in the course of the derivation, the bottom parts of all scope parts attaching to the same node unify because of the unifications done during adjunction. In Fig. 6 for example this means that [P l_2] and [P l_4] are unified, which leads to a failure.

Because of these considerations we decided not to choose possibilities i) or ii) but to state scope constraints explicitely in the semantic representations and use semantic unification with final top-bottom unification as described above.

5 Comparison to Gardent & Kallmeyer

Among the approaches to LTAG semantics mentioned in the beginning of this section, (Gardent and Kallmeyer, 2003) is the closest to our framework.

Obviously, everything one can do in the approach proposed in (Gardent and Kallmeyer, 2003) can be directly transformed into the approach presented here. An advantage of our approach is that semantic feature structures are linked to whole elementary trees and therefore they offer the possibility to define global features for elemenary trees. So far we have not exploited this in this paper but it obviously might be useful, for example for the MAXS and MINP features in section 6.

A problem of (Gardent and Kallmeyer, 2003) is that, as already mentioned, in order to do semantics using the feature structures in the syntax, an arbitrary number of possible feature values needs to be allowed, since the number of labels and individual variables occurring in a sentence cannot be limited in a general way. Consequently the number of possible feature structures is no longer finite and therefore, in contrast to standard FTAG (Vijay-Shanker and Joshi, 1988), the formalism is no longer equivalent to TAG. This means that semantic features are slightly different from those needed for syntax in terms of formal properties. Therefore, it is more appropriate to separate them from syntactic features and to

link them to whole semantic representations (i.e., to link them to whole elementary trees). This is what the approach described above does: instead of increasing the formal power of the syntactic formalism, the extra power needed for semantics is added to the semantic representations, i.e., to nodes in the derivation tree.

A further important difference is that we do not use explicit holes $h_1, h_2, \ldots$ besides propositional variables. Instead, the propositional variables that remain after having performed all unifications are understood as being holes in the sense of previous LTAG semantics approaches. This simplifies the formal framework considerably.

6 Sample analyses

6.1 Quantifiers

(6) everybody laughs

For quantificational NPs as in (6) we propose the analysis shown in Fig. 8. This allows us to obtain the scope constraints mentioned above: the NP proposition is in the restriction of the quantifier because of $\boxed{4} \geq l_3$. Furthermore, the following must be guaranteed: 1. the proposition to which a quantifier attaches must be in its nuclear scope and 2. a quantifier cannot scope higher than the next finite clause. The first constraint must result from the combination of the lower part of the quantifier (the NP tree) and the tree to which it attaches.[6] We introduce a feature MINP to pass the proposition of a tree to an embedded quantifier. The second constraint must result from the adjunction of the scope part of the quantifier. We use a feature MAXS ('maximal scope') that passes an upper limit for scope from a verb tree to an adjoining scope tree. E.g., see Fig. 8 for the analysis of (6). It leads to the following unifications: $\boxed{6} = \boxed{2}$ (adjunction of the scope part), $\boxed{1} = x$ and $\boxed{7} = l_1$ (substitution of the predicate-argument part, and $\boxed{3} = l_1$ (final top-bottom unification). The result is (7) which has just one disambiguation: $\boxed{2} \rightarrow l_2, \boxed{4} \rightarrow l_3, \boxed{5} \rightarrow l_1$.

(7)
$$\begin{array}{l} l_1 : \mathsf{laugh}(x), \\ l_2 : \mathsf{every}(x, \boxed{4}, \boxed{5}), l_3 : \mathsf{person}(x) \\ \boxed{2} \geq l_1, \boxed{2} \geq l_2, \boxed{4} \geq l_3, \boxed{5} \geq l_1 \end{array}$$

Note that this analysis of quantifiers differs crucially form what is proposed in (Gardent and Kallmeyer, 2003) where quantifiers do not have a separate scope part. This separate scope part allows us to account for various constraints for quantifier scope.[7]

[6] This is particularly clear in examples with quantificational NPs that are embedded in other quantificational NPs as considered in (Joshi et al., 2003). Here, the minimal nuclear scope of the embedded NP depends on the embedding NP and not on the verb tree.

[7] (Joshi et al., 2003) derive for example constraints for relative quantifier scope in so-called inverse linking configurations from the way the scope parts combine.

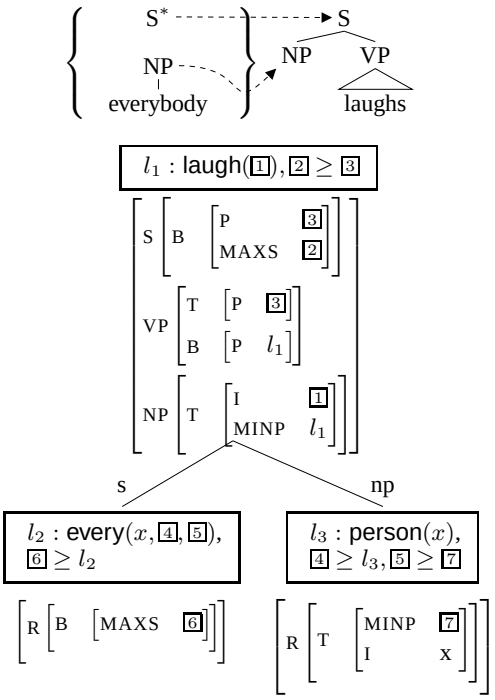

Figure 8: Analysis of (6)

6.2 Attitude verbs

(8) Mary thinks John laughs

The analysis of attitude verbs such as *thinks* in (8) is shown in Fig. 9. The propositional argument of *think* (variable $\boxed{3}$) is the MAXS value of the embedded verb (MAXS of the top of the foot node). This means that quantifiers or adverbs attaching to the lower verb cannot scope over *thinks*.[8] The adjunction leads to $\boxed{3} = \boxed{1}$.

(9) Mary thinks John likes everybody

In (9), wide scope of *everybody* over *thinks* should be disallowed. If its scope part attaches to the S node of *likes*, then the scope is blocked by the MAXS value of *likes*. Consequently, *everybody* cannot have scope over *thinks* because *thinks* takes the MAXS proposition of *likes* as its argument. However, we have to make sure that the scope part of *everybody* cannot attach higher, i.e., to *thinks*.

In general, we allow scope parts to adjoin higher. But, following (Joshi et al., 2003), the compositions must be such that one or more already derived trees or tree sets attach (by substitution or adjunction) to one single elementary tree. If only the NP tree of *everybody* attaches to *like*, there are only two possible continuations and both lead to an incorrect derivation for (9). The first possible

[8] Some counterexamples to finite clause boundness are analyzed nowadays as cases of illusive scope (Fox and Sauerland, 1996).

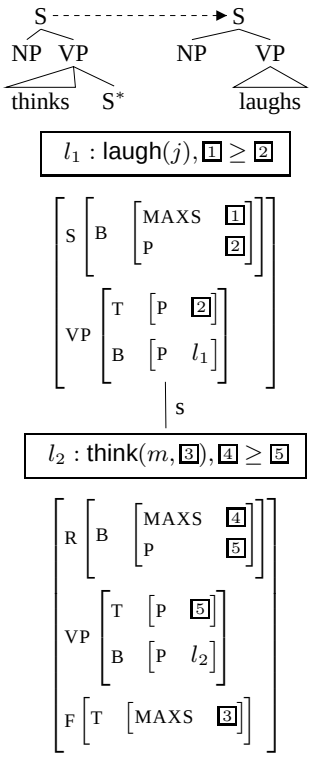

Figure 9: Analysis of (8)

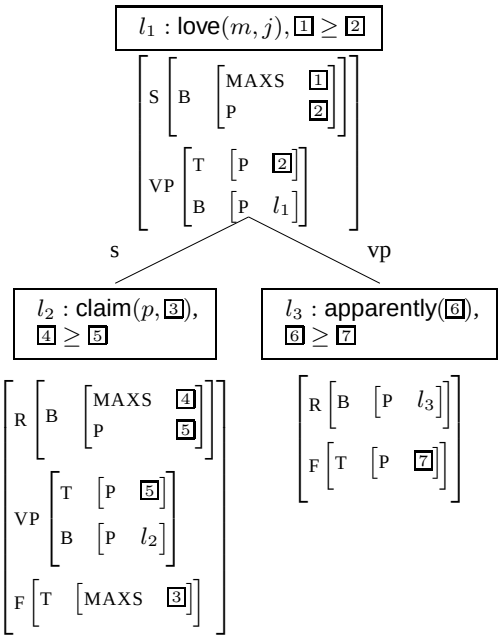

Figure 10: Analysis of (11b)

continuation is to adjoin the scope part of *everybody* to *thinks*. Then the derived *like* tree also must be added to *thinks* since it is part of the same derived tree set, i.e., *thinks* would have a substitution node instead of a foot node. This however is problematic for the analysis of long-distance dependencies in LTAG. The second possible continuation is that *thinks* attaches to *like* simultaneously with the lower *everybody* part. But then the scope part has to find some other node than the S node of *thinks* in order to attach to it. There is no other S node besides those coming from *thinks* or *like*, so this possibility does not work either. Consequently, one has to adjoin the scope part to the *like* S node.

6.3 Problems for derivation based semantics

Now let us come back to the examples (2) and (3) mentioned in the beginning, repeated here as (10) and (11):

(10) Who does Paul think John said Bill liked?

(11) a. Mary, Paul claims John seems to love
b. Paul claims Mary apparently loves John

For an analysis of (10) we refer to (Romero et al., 2004) in this volume. An analysis of (11b) is shown in Fig. 10, (11a) is analyzed in the same way. We analyze raising verbs similar to adverbs (see *sometimes* in Fig. 3). They are in a sense inserted between the top and bottom P values of the node to which they adjoin. They scope over the lower proposition. By unification, the proposition introduced by the topmost adverb/raising verb is the P value of the root of the verb tree which is below the MAXS proposition. Therefore, in (11b), the attitude verb *claim* takes scope over the adverb.

Furthermore, the problem of multiple modifiers as in (12) is also 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)). These cases are difficult for a derivation tree based semantics because only the adjective that is closest to the modified noun attaches to the noun, all adjectives that are further to the left attach to the adjective on their right. However, all adjectives equally take the variable provided by the noun as their argument.

(12) roasted red pepper

As shown in Fig. 11, in our approach the arguments of the three predicates, *pepper*, *red* and *roasted* can be easily unified such that they all refer to the same individual.

7 Conclusion

In this paper we introduced an LTAG semantics framework based on the derivation tree. We use feature structure unification on the derivation tree as semantic composition operation, similar to the syntactic features on the derived tree that are used in TAG. Within this framework, we proposed an account of quantificational NPs, adverbs, raising verbs and attitude verbs, and we have shown that

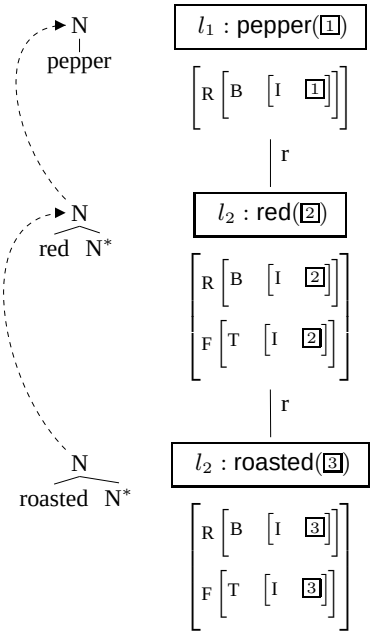

Figure 11: Analysis of (12)

we can analyze the examples considered in the literature as problematic for derivation tree based LTAG semantics approaches.

Acknowledgments

For many fruitful discussions of the framework and the analyses presented here we would like to thank Aravind K. Joshi and all members of the XTAG Group at the University of Pennsylvania. Furthermore, we are grateful to two anonymous reviewers for their useful comments.

References

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.

Mark Dras, David Chiang, and William Schuler. 2004. On Relations of Constituency and Dependency Grammars. *Journal of Language and Computation*, 2(2):281–305.

Danny Fox and Uli Sauerland. 1996. Illusive Scope of Universal Quantifiers. In *Proceedings of NELS 26*, Amherst.

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.

Robert Frank. 1992. *Syntactic Locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perspectives*. Ph.D. thesis, University of Pennsylvania.

Claire Gardent and Laura Kallmeyer. 2003. Semantic Construction in FTAG. In *Proceedings of EACL 2003*, Budapest.

Mark Johnson. 1994. Computing with Features and Formulae. *Computational Linguistics*, 20(1).

Aravind K. Joshi and Yves Schabes. 1997. Tree-Adjoining 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.

Aravind K. Joshi, Laura Kallmeyer, and Maribel Romero. 2003. Flexible Composition in LTAG: Quantifier Scope and Inverse Linking. In Harry Bunt, Ielka van der Sluis, and Roser Morante, editors, *Proceedings of the Fifth International Workshop on Computational Semantics IWCS-5*, pages 179–194, Tilburg.

Laura Kallmeyer and Aravind K. Joshi. 2003. Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG. *Research on Language and Computation*, 1(1–2):3–58.

Laura Kallmeyer. 2002a. Enriching the tag derivation tree for semantics. In *Proceedings of KONVENS 2002*, pages 67 – 74, Saarbrücken, October.

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

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.

Maribel Romero, Laura Kallmeyer, and Olga Babko-Malaya. 2004. LTAG Semantics for Questions. In *Proceedings of TAG+7*, Vancouver.

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

Matthew Stone and Christine Doran. 1997. Sentence Planning as Description Using Tree-Adjoining Grammar. In *Proceedings of ACL*, pages 192–205.

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