UDRT-based Semantics Construction for LTAG

– and what it tells us about the role of adjunction in LTAG –

Philipp Cimiano
AIFB Karlsruhe
University of Karlsruhe
Karlsruhe, Germany

Anette Frank
Language Technology Lab
DFKI GmbH
Saarbrücken, Germany

Uwe Reyle
Inst. of Natural Language Processing
University of Stuttgart
Stuttgart, Germany

1 Introduction

Lexicalized Tree Adjoining Grammars (LTAGs) [6] are tree rewriting systems consisting of a finite set of trees associated with lexical items, so-called elementary trees (etrees). Elementary trees represent extended projections of lexical items that encapsulate all their syntactic/semantic arguments. Due to this property, semantics construction in LTAG can be based on elementary trees as basic units, and is typically performed on the basis of the *derivation tree*, which records the history of how elementary trees are combined by substitution and adjunction operations.

However, it has been argued by Frank and van Genabith [3] that the derivation tree is not an appropriate structure for principle-based semantics construction in that the derivation tree neither mirrors the phrase structure nor the dependency structure of the sentence. It is essentially due to the adjunction operation that dependencies that are crucial for principle-based semantics construction procedure are not available from the derivation tree. Instead, Frank and van Genabith propose to define semantics construction on the basis of the *derived tree*.

In this paper we will have a closer look at the role that adjunction plays for semantics composition in LTAG and examine whether the semantics construction approach based on derivation trees can be rescued.

The contributions of the paper are manifold. First, we show that the semantics construction approach for LDGs presented by Cimiano and Reyle [2] can in principle be applied to semantics construction for LTAG. We borrow from Cimiano and Reyle the idea of talking explicitly about labels and scope, which will turn out as a crucial component for principle-based semantics construction on the basis of the derivation tree. Second, we show that an adequate account of quantifier scope within LTAG is possible without necessarily leading to an extension of LTAG as

proposed by Kallmeyer [9], who introduces multicomponent elementary trees and allows for restricted multiple adjoining in the case of quantifiers. Third, we show that it is possible to define semantics from derivation trees in a principled way, thus overcoming the problems arising from the non-isomorphism of adjunction and syntactic dependencies discussed in [3]. We show that by respecting the depth of embedding of constituents as well as defining semantic composition as an update function iteratively combining the semantic representations of a mother node and each child, the above non-isomorphism can be solved in a principled way. Further, we show that by resorting to an underspecified semantic representation language – UDRT in our case [11] – we can handle handle scope phenomena without any complication of LTAG as well as provide a semantic counterpart for the (syntactic) operation of adjunction. Finally, from a more general perspective, our work helps to clarify the role of adjunction in LTAG.

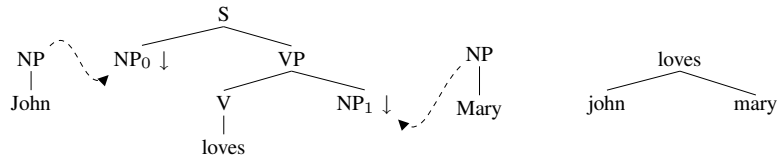
The paper is structured as follows: in Section 2 we briefly introduce the syntactic operations of substitution and adjunction, and the standard approach to semantics construction in LTAG. We then review an example discussed by Frank and van Genabith, in which the derivation tree does not contain certain dependencies that are necessary to define semantics compositionally, using standard methods. In Section 3 we present our UDRT-based approach to semantics construction based on the derivation tree. In Section 4 we show how the approach elegantly accounts for two problems which have been observed for semantics construction based on the derivation tree. In Section 5 we discuss related work. Section 6 concludes.

2 Semantics Construction for LTAG

The two main operations in LTAG are substitution and adjunction. Substitution is a local operation for the insertion of subcategorized arguments, while adjunction typically folds one tree into another, thereby introducing modifiers or recursively embedding structures, such as clausal arguments.

Substitution is illustrated by the quantifier-free example (1), with elementary trees for *John, loves, Mary* and the ensuing derivation tree, defined by the two substitution operations that build the sentence *John loves Mary.*[1]

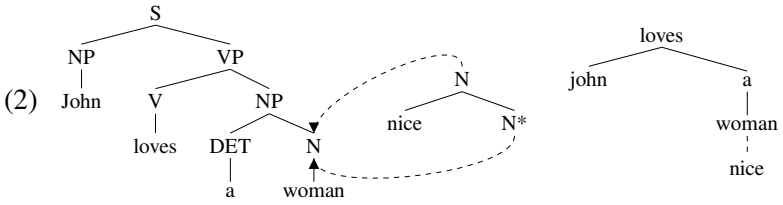
(1)



[1]Throughout this paper, ↓ stands for a substitution node and ∗ marks the footnote of an auxiliary tree. In derivation trees, direct edges stand for substitution edges, dashed lines for adjoining.

Adjunction is illustrated by adjective modification in example (2) below. We display adjunction of the etree for *nice*, along with the resulting derivation tree.

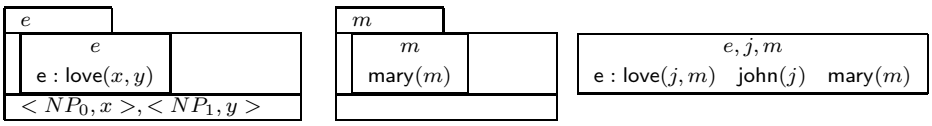
(2)



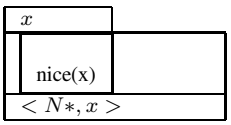
For semantics construction based on the derivation tree (as in [9], [7], etc.), it is essential that all arguments in the semantic form of an elementary tree are associated with their corresponding substitution nodes in the elementary tree, so that the main variable of the semantic representation of an elementary tree that is substituted in can be bound to the correct argument. Similarly, for adjunction the main variable of the semantic form of an adjoining tree has to bind the variable associated with the tree or node it adjoins to.

We illustrate this standard mechanism by way of two examples. The semantic forms given below, phrased in a DRT-style [10] notation, contain simple lexical DRSs in their middle part. The lower part records the correspondence of argument variables with syntactic substitution or foot nodes, while the top part marks the main variable that is externalised for binding.

Semantics construction for the substitution edges in (1) is performed by binding the externalised variables of the etrees for *John* and *Mary* to the variables of the respective substitution nodes in the etree for *loves*. The entries for the verb *love*, proper nouns like *Mary* as well as the result after substitution are given below:



In the semantic form for the auxiliary tree *nice*, the predicated variable is externalised for binding and associated with the the foot node of the elementary tree. As a result of adjoining, the variable x will be unified with the main variable in the semantic representation of the noun the adjective tree adjoins to.



As noted by Frank and van Genabith, semantics construction from the derivation tree is not always as straightforward as in the examples above. In certain con-

figurations, the special nature of the adjunction operation yields a derivation tree that does not specify certain syntactic dependencies that are needed for a principle-based semantics construction procedure. The example they discuss – *Spicy hotdogs Peter claims Mary seems to adore* – involves an interaction of topicalization, raising and control verbs. The relevant elementary trees are given below, with an indication of the adjunction operations necessary to build the derived tree. As can be seen, the semantic dependency between claim and seem does not have any syntactic counterpart in the structure of the derivation tree. This is because *seem* and *claim* are directly and independently adjoined to the etree of *adore* (cf. [3]).

This causes a serious problem for principle-based semantics construction, as there is a missing syntactic dependency between *claim* and *seem* which needs to be made explicit in the final semantic representation.

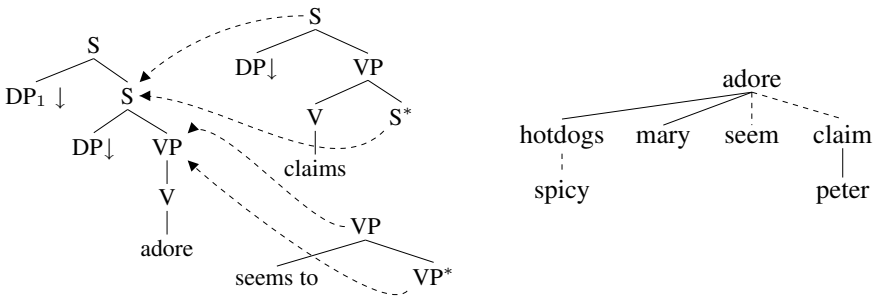


Figure 1: *Spicy hotdogs Peter claims Mary seems to adore.*

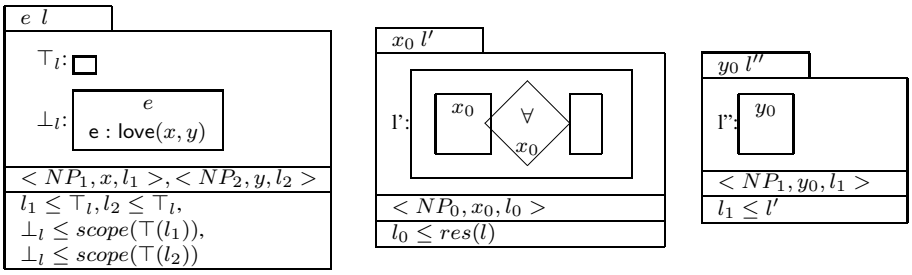
Different solutions have been proposed for this problem. Frank and van Genabith [3] as well as Gardent and Kallmeyer [4] propose to construct semantics from the derived tree, while the proposal of Kallmeyer [8] would suggest introducing an additional syntactic link between the control and the raising verb. In our proposal we will try to rescue the standard LTAG approach where semantics construction is based on the derivation tree, while avoiding to introduce additional syntactic links as suggested by Kallmeyer. We develop an UDRT-based approach which, by using a labeled and underspecified semantic representation language, allows us to 'talk' about scope relations explicitly. This enables us to define a genuine semantic counterpart of the adjunction operation that solves the problem in a principled way.

3 UDRT-based Semantics Construction for LTAG

The main ingredients of our UDRT-based approach to semantics construction for LTAG are: (i) a labeled and underspecified semantic representation language allowing to talk about scope (UDRT in our case), (ii) a universal semantic composition

operation which is independent of the specific syntactic operation applied, (iii) a recursively defined semantics construction method which results in iterative updates of the semantics of the mother node en par children are processed in a certain order and their semantic representation combined with it. operates on partial underspecified semantic representations and (iv) an ordered traversal of child nodes in semantic composition, which mirrors the dominance relations in the derived tree.

We illustrate points (i) and (ii) by way of an example. Consider the following extended entries for the verb form *loves* and the determiners *every* and *a*:



The above semantic representations consist of four boxes. The upper left box contains the *distinguished referent (dr)* and the *distinguished label (dl)* of the semantic representation. For verbal entries and determiners, the distinguished referent is the logically bound referent, and for nouns and adjectives it is the free variable. The distinguished label is the label that provides an index for the whole set of labelled *UDRS-components* given in the second box. We assume that the $\bot$- and $\top$-components of verbs (and sentential UDRSs in general) are given as values of functions $\bot$ and $\top$ applied to the distinguished label. For determiners the functions *res* and *scope* map the distinguished label to the labels of the restrictor and scope components, respectively.[2] The third box consists of *argument triples* representing local substitution/foot nodes X of the elementary tree in question together with their distinguished referents and labels. They implement the syntax-semantics interface in the following way: substitution or adjunction of a tree γ at X will trigger unification of the referent variable and label associated with X in the 2nd and 2rd component of the triple, i.e. the distinguished referent and label of the semantics of the node combined with X, respectively. The fourth box contains conditions that partially order the labels (and thus the DRSs) introduced in the second box and in the argument triples.

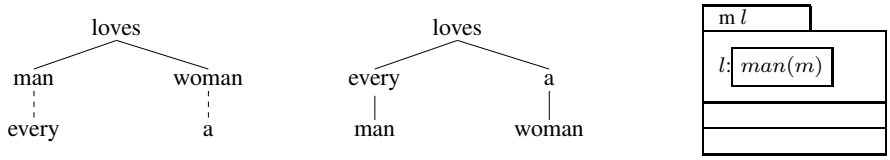
Summing up, our semantic representations are 5-tuples $\sigma = \langle dr, dl, C, A, S \rangle$, where dr is the distinguished referent, dl is the distinguished label, C is a set of

[2]For genuine quantifiers we have $res(l) < l$ and $scope(l) < l$, and for indefinite determiners and proper names $res(l) = scope(l) = l$. As these are general conditions on the representation language they are not made explicit in the above representations.

labelled DRSs, A is a set of argument triples consisting of a node identifier, a referent variable and a label and S is a set of subordination relations on labels.

Let us now turn to point (ii) and consider the semantics composition involving a main verb, a determiner and a noun. When nouns are combined with determiners, nouns may either be substituted into the etree of the determiners, or the tree of the determiner is considered as an auxiliary tree that is adjoined to the noun[3] (see [8]).

(2)



Each derivation tree given in (2) should yield the same UDRS, i.e. the one given in Figure 2. To achieve this result, we need (i) to form the set-theoretic union of the second and forth boxes of the semantics of determiner and noun, (ii) unify the distinguished referent of the determiner with the variable in the noun and the corresponding argument variable of the verb, and (iii) unify the distinguished label of the noun with the label of the restrictor of the determiner and the distinguished label of the verb with its scope. The following definition of the semantic composition operator, $\oplus$, makes this more precise.

Definition 1 (Semantics composition $\oplus$)
Let (γ_1, γ_2) be an edge in a derivation tree labeled with p, where p is a node in γ_1, and $\sigma_1 = \langle dr_1, dl_1, C_1, A_1, S_1\rangle$ and $\sigma_2 = \langle dr_2, dl_2, C_2, A_2, S_2\rangle$ are the semantic representations of γ_1 and γ_2 respectively. Then the result of combining σ_1 and σ_2 is an update of σ_1, i.e. $\sigma_1 := \sigma_1 \oplus \sigma_2 = \langle dr', dl', C', A', S'\rangle$, where one of two configurations holds, depending on whether the edge was created by (i) a substitution or (ii) an adjunction operation:

(i) if $(p_1, x_1, l_1) \in A_1$, for $p_1 = p$, then $dr' = dr_1$, $dl' = dl_1$,
$C' = C_1 \cup C_2$ where x_1 is unified with dr_2
$S' = S_1 \cup S_2$ where l_1 is unified with dl_2
$A' = A_1 \backslash \{(p_1, x_1, l_1)\} \cup A_2$
(ii) if $(p_2, x_2, l_2) \in A_2$ for some $p_2 \neq p$, then $dr' = dr_2$, $dl' = dl_2$,
$C' = C_1 \cup C_2$ where x_2 is unified with dr_1
$S' = S_1 \cup S_2$ where l_2 is unified with dl_1
$A' := A_1 \cup (A_2 \backslash \{(p_2, x_2, l_2)\})$[3]

[3]This is actually the way determiners are modeled in the XTAG Grammar [5].

[3]For the cases we have investigated it actually suffices to have $A' := A_1 \backslash \{(p_1, x_1, l_1)\}$ in case (i) and $A' := A_1$ in case (ii).

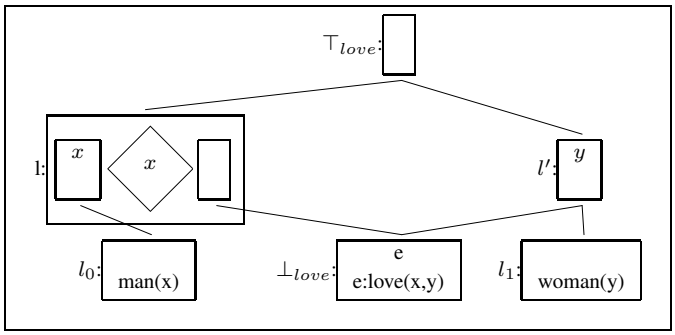


Figure 2: UDRS for *Every man loves a woman*

While we mention the correspondence with the distinct syntactic operations, the above definition presupposes that semantics composition is completely specified by the arguments made explicit in the third box of the semantic representation. It thus represents a universal operation for semantic composition in LTAG. For this we have required that adjectives and adverbial modifiers specify nouns and a verb phrases as arguments, respectively. Further, the above definition rests on the standard assumption in LTAG requiring exactly one foot node per auxiliary tree (see [6]). This implies that there is at most one foot node p_2 that can be involved in adjunction, such that for the semantic representation σ_2 above, all arguments identified by substitution nodes have been already inserted, while only one argument identified with the unique foot node remains, which makes case (ii) deterministic.[4] In essence, semantic composition could thus also be performed relying on the lambda calculus, which is an ideal formalism to make required arguments explicit and model argument insertion through functional application. The bottleneck of the lambda calculus, however, is that it assumes a specific order in which the insertions need to be carried out, while in the above formalism this is not the case.

Taking stock, we have handled scope in an appropriate way which differs from the way it is treated in [9] but also in [3]. In Kallmeyer [9], scope relations are accounted for by separating syntactically the restrictor and scope part of a quantifier and extending the syntactic representation by a 'degenerate' auxiliary tree. This presupposes to allow adjoining of multiple auxiliary trees to one tree and thus the extension to multiple component TAGs (MCTAGs). We have shown that scope relations can be cleanly accounted for at the semantic representation level with-

[4] As an alternative, we could revise LTAG's standard labelling scheme for derivation trees. Edges of the derivation tree are standardly labelled with the *target* node of substitution and adjunction operations. For the adjunction case (ii), though, we need to refer to the source node p_2 in the adjoining tree γ_2. By revising the labelling scheme such that edges are labelled with the target node for substitution and the source node in case of adjunction, conditions (i) and (ii) could be collapsed to uniformly refer to the edge label p in order to pick the correct nodes for the unification of referents.

out extending LTAG. Further, we have sketched a universal semantic composition operation which is independent of the syntactic operation performed. Finally, we have introduced the idea of building up partial semantic structures which represent the merge of all the semantic representations applied so far. This idea is elaborated in the next section and is shown to overcome the problems sketched in Section 2.

4 The derivation tree approach revisited

Different configurations have been shown to pose problems for a derivation-based approach for principle-based semantics construction, due to missing syntactic links. We have mentioned, on the one hand, the treatment of a determiner as an auxiliary tree and have shown in the last section that our approach does not face this problem. On the other hand, we have illustrated problems arising from configurations involving topicalization, control and raising verbs by means of the *spicy hotdog* example in Section 2. Adding an additional syntactic link between *claim* and *seem* might seem a solution to the problem, but would lead to a rather 'ad-hoc' complication of LTAG. In our approach, these problems are circumvented by defining semantic composition recursively in such a way that the semantic representation of a mother node is not only combined with the semantic representation of a single child (as in standard derivation-based approaches), but with the merge (as specified by Def.1) of the semantic representations built up so far. The recursive definition is as follows:

Definition 2 (Semantic Composition on the Basis of the Derivation Tree)
For a leaf node t the semantics $\sigma(t)$ of t is σ_t (termination condition).
For a node f with n (unordered) children $c_1, ...c_n$, the semantics of f is defined as

$$\sigma(f) = (...((\sigma_f \oplus \sigma(c_{m_1})) \oplus \sigma(c_{m_2}))... \oplus \sigma(c_{m_n}))$$

where the order of composition is defined such that if an auxiliary tree c_i adjoins at a lower node in the elementary tree than another auxiliary tree c_j, then c_i is processed before c_j, i.e. it holds for their indexes that $i < j$.

The above order specifies that adjunctions are carried out in a way that mirrors the way semantic composition would be applied in a derived tree approach. For our *spicy hotdog* example this means that *seem* will be processed before *claim*. However, it is important to mention that we do not need to construct the derived tree as such, as the nodes in derivation trees are typically numbered and adjoining edges typically specify the node where adjunction is performed.

The crucial move is thus to ensure that, after *seem* has been adjoined to *adore*, *claim* is not only combined with the semantic representation of *adore*, but with the

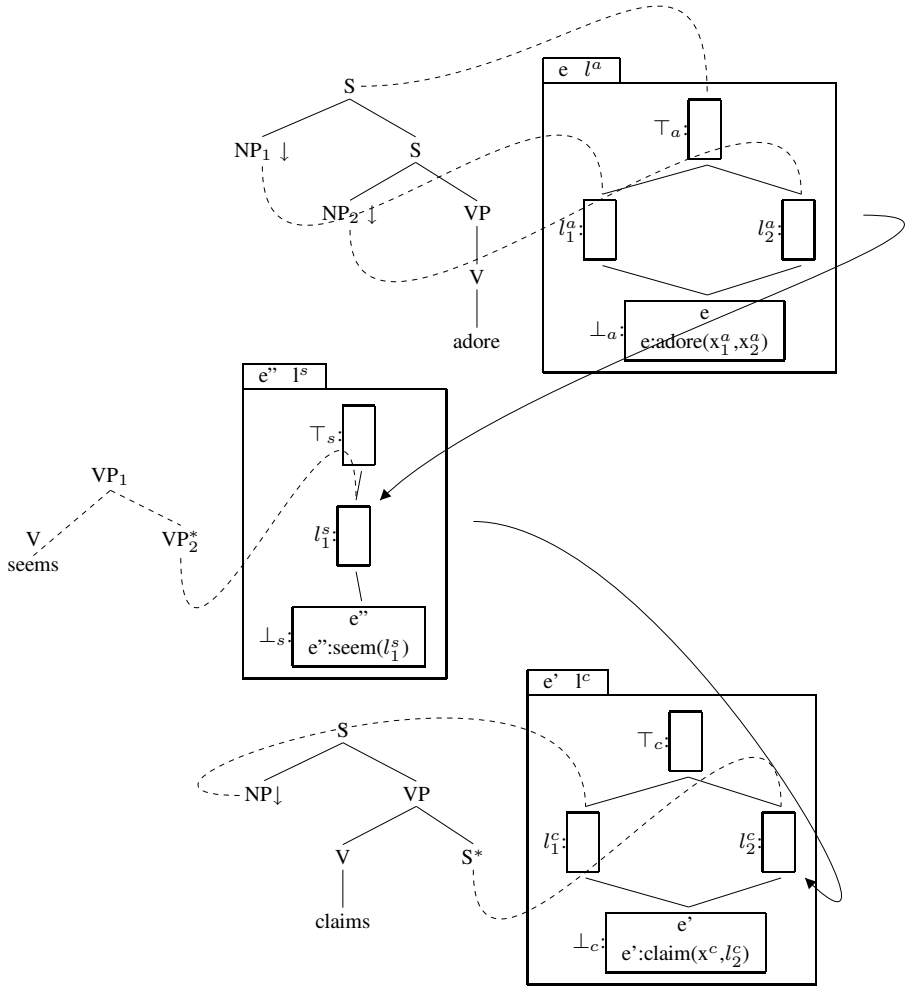
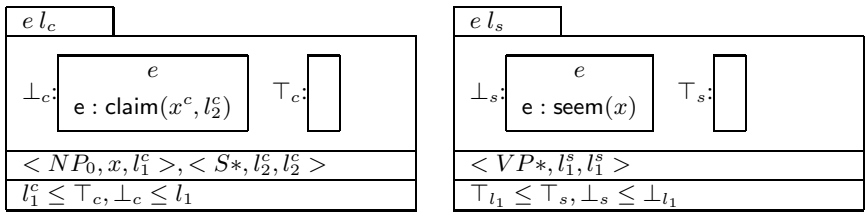


Figure 3: Semantic representation and interaction between *adore*, *claim* and *seem*.

result of combining *adore* and *claims*. The sentential complement of *claim* is thus not *adore*, but *seem to adore*. We show below that we also yield the right scope relation in this way. The entries for *seems* and *claims* look as follows:[5]



[5]For the sake of simplicity the propositional arguments of both verbs are referred to by means of their labels l_1^s and l_2^c.

Now, when *seem* adjoins to *adore*, it wraps the whole semantic representation of *adore*, which gets inserted into the $\top_{seem}$ and $\bot_{seem}$ components. The same happens then again when *claim* is now adjoined, not only to the semantic representation of *adore*, but to the whole semantic representation built up so far. Claims thus wraps around the whole semantic representation built up so far with the result that $\top_{seem}$ gets subordinated by $\top_{claim}$ and $\bot_{seem}$ subordinates $\bot_{claim}$, thus leading to the correct semantic representation given in Figure 3. Thus, *claim* and *seem* introduce semantic structures which 'wrap around' the semantic structure they adjoin to (compare Figure 3). Such a perspective is indeed very interesting as it provides a semantic counterpart to the syntactic operation of adjoining, which can also be defined as an operation in which the auxiliary tree wraps around the elementary tree it adjoins to (compare [9]).

Note here that the surface-oriented order for traversal of the derivation tree is indeed crucial as processing *claim* before *seem* would have yielded incorrect semantic dependencies. The order defined in Definition 2 in some sense mirrors the derived tree, but we have argued that the construction of the derived tree itself is not necessary as each adjoining child specifies the node and hence the position of the elementary tree it adjoins to. A similar solution to this problem was discussed in [7], but their approach was unfortunately not formalized.

5 Related Work

Joshi and Vijay-Shanker [7] proposed a compositional semantics approach based on the derivation tree. They assume flat semantic representations and define appropriate semantic construction operations for substitution and adjoining. Their approach is essentially restricted to the predicate-argument domain. Most importantly, they propose a similar solution to ours, which accounts for the interaction between control and raising verbs, but is not formalized, however. Further, they also suggest a treatment of quantifiers by (syntactically) factoring the scope and restrictor part of quantifiers and representing them via 'degenerate' auxiliary trees and extending LTAG to MCTAG (Multi-Component-Tag). This idea is further elaborated by Kallmeyer and applied to hole semantics [9], showing that MCTAG accounts for scope underspecification by introducing the above mentioned 'degenerate' auxiliary trees which can multiply adjoin to the top S node of the main clause verb. Actually, in Kallmeyer's proposal, the expressiveness of MCTAG is restricted to merely represent the degenerate trees representing quantifiers. Our proposal, which avoids any extension of LTAG by using an underspecified representation language, thus being able to talk about labels and scope relations explicitly, has shown that the use of MCTAG is unnecessary if one resorts to the

semantic level to handle scope[6].

The principled problems arising from the non-isomorphism between the operation of adjunction and the underlying syntactic-semantic dependencies discussed in Section 2, have led some authors to abandon the derivation tree for principled semantics construction (see Frank and van Genabith's proposal based on glue logic [3] or the FTAG proposal of Gardent and Kallmeyer [4], based on minimal recursion semantics). Other proposals have tried to circumvent the problem by introducing additional syntactic links (e.g. [8]). We have shown that the problems are indeed not principled ones if we respect the two key characteristics of semantics construction based on the derivation tree: the order of constituent embedding and the fact that the semantics composition is defined as an update function which iteratively merges the semantic representation of each child node with the semantic representation of the mother node.

6 Conclusion

From a more general perspective, our paper offers to take a fresh look at a long debated issue, namely the special role of adjunction in LTAG and its proper treatment in principle-based semantics construction. Classical approaches to semantics construction are based on the phrase structure or dependency structure of the sentence. Since adjunction in LTAG factors recursion differently from phrase structure grammars, this difference is reflected in the structure of the derivation tree.

The main contribution of this paper can thus be seen in three aspects. First, we show that it is crucial for a proper treatment of LTAG adjunction in semantic composition, to respect the order of syntactic embedding that is usually reflected in the phrase structure (here, derived) tree. This embedding structure can be read off the derivation tree, provided the node position of adjunction sites are recorded.

Second, we have shown that it is essential for an approach based on the derivation tree to define semantics compositionally in such a way that the semantics of each child node is not only combined with the semantic representation of the mother node, but with the iteratively updated representation resulting from the merge of the semantic representation of the children processed so far with the mother node's semantics.

Summarizing, it is the iterative definition of semantics construction as well as the fact that syntactic embedding is considered, both key features of standard approaches to semantics construction based on phrase structure, that help to overcome the problems resulting from missing dependencies described in Section 2.

[6]This only holds with respect to the treatment of quantifiers. There are very good reasons to adopt an MCTAG analysis to account for German scrambling for example [1].

Finally, by resorting to a semantic formalism that defines scope on the basis of labeled partial semantic structures, it is possible to account for scope underspecification without any additional syntactic extension of LTAG as well as to define semantic composition for adjunction in a way that mirrors its "wrapping" behavior to model syntactic recursion at the level of semantics composition

References

[1] T. Becker, A. Joshi, and O. Rambow. Long distance scrambling and tree adjoining grammars. In *Proceedings of the 5th Conference of the European Chapter of the ACL*, 1991.

[2] P. Cimiano and U. Reyle. Talking about trees, scope and concepts. In H. Bunt, J. Geertzen, and E. Thijse, editors, *Proceedings of the 6th International Workshop on Computational Semantics (IWCS-6)*, pages 90–102, 2005.

[3] A. Frank and J. van Genabith. Ll-based semantics for LTAG - and what it teaches us about LFG and LTAG. In M. Butt and T. Holloway King, editors, *Proceedings of the International Conference on Lexical Function Grammar (LFG)*. CSLI Online Publications, 2001.

[4] C. Gardent and L. Kallmeyer. Semantic construction in feature-based TAG. In *Proceedings of the 10th Meeting of the European Chapter of the Association for Computational Linguistics*, 2003.

[5] XTAG Research Group. A lexicalized tree adjoining grammar for english. Technical Report IRCS-01-03, IRCS, University of Pennsylvania, 2001.

[6] A.K. Joshi and Y. Schabes. Tree-adjoining grammars. In *Handbook of Formal Languages*, volume 3, pages 69–124. Springer, 1997.

[7] A.K. Joshi and K. Vijay-Shanker. Compositional semantics with lexicalized tree-adjoining grammar (LTAG): How much underspecification is necessary? In *Proc. of the 3ed International Workshop on Computational Semantics (IWCS-3)*, pages 131–145, 1999.

[8] L. Kallmeyer. Enriching the tag derivation tree for semantics. In *Proceedings of KONVENS (Konferenz zur Verarbeitung natürlicher Sprache)*, pages 67–74, 2002.

[9] L. Kallmeyer and A.K. Joshi. Factoring predicate argument and scope semantics: Underspecified semantics with LTAG. *Research on Language and Computation*, 1(1-2):3–58, 2003.

[10] H. Kamp and U. Reyle. *From Discourse to Logic*. Kluwer, 1993.

[11] Uwe Reyle. Dealing with ambiguities by underspecification: Construction, representation and deduction. *Journal of Semantics*, 10(2):123–179, 1993.