Semantic construction in Feature-Based TAG

Claire Gardent
CNRS, Nancy
BP 239 - Campus Scientifique
54506 Vandoeuvre-les-Nancy,France
gardent@loria.fr

Laura Kallmeyer
TALaNa / Lattice, Université Paris 7
2 place Jussieu
75251 Paris Cedex 05, France
laura.kallmeyer@linguist.jussieu.fr

Abstract

We propose a semantic construction method for Feature-Based Tree Adjoining Grammar which is based on the derived tree, compare it with related proposals and briefly discuss some implementation possibilities.

1 Introduction

Semantic construction is the process of constructing semantic representations for natural language expressions. Perhaps the most well-known proposal for semantic construction is that presented in (Montague, 1974) in which grammar rules are applied in tandem with semantic rules to construct not only a syntactic tree but also a lambda term representing the meaning of the described constituent.

Montague's approach gave rise to much further work aiming at determining the correct rules and representations needed to build a representation of natural language meaning. In particular, computational grammars were developed which by and large took on Montague's proposal, building semantic representations in tandem with syntactic structures. Thus for instance, (Copestake et al., 2001) shows how to specify a Head Driven Phrase Structure Grammar (HPSG) which supports the parallel construction of a phrase structure (or derived) tree and of a semantic representation, (Zeevat et al., 1987) shows it for Unification Categorial Grammar (UCG) and (Dalrymple, 1999) for Lexical Functional grammar (LFG).

One grammatical framework for which the idea of a Montague style approach to semantic construction has not been fully explored is Tree Adjoining Grammar (TAG, (Joshi and Schabes, 1997)). In that framework, the basic units are (elementary) trees and two operations are used to combine trees into bigger trees, namely, adjunction and substitution. Because the adjunction rule differs from standard phrase structure rules, two structures are associated with any given derivation: a derivation tree and a derived tree. While the derived tree is the standard phrase structure tree, the derivation tree records how the elementary trees used to build this derived tree are put together using adjunction and substitution. Furthermore, because TAG elementary trees localise predicate-argument dependencies, the TAG derivation tree is usually taken to provide an appropriate basis for semantic construction. And thus, the more traditional, "derived tree"-based approach is not usually pursued – An exception to this is (Frank and van Genabith, 2001) which presents a fairly extensive specification of a derived tree based semantic construction for TAG and with which we will compare our approach in section 5.

In this paper, we explore the idea of a semantic construction method which is based on the TAG derived tree and show how a Montague style (unification based) approach to semantic construction can be applied to Feature-Based Tree Adjoining Grammar (FTAG, (Vijay-Shanker and Joshi, 1988)). We relate our approach to existing proposals and discuss two possibilities for implementation.

2 Hole semantics

We start by introducing the semantic representation language we use. As mentioned above, Montague was using the lambda calculus. In computational linguistics, two new trends have emerged however on which our proposal is based.

On the one hand, there is a trend towards emulating beta reduction using term unification[1]. Instead of applying a function to its argument and reducing the resulting lambda term using beta reduction, functors are represented using terms whose arguments are unification variables. The syntax/semantics interface and the use of unification then ensures that these variables get assigned the appropriate values i.e., the values representing their given arguments.

On the other hand, flat semantics are being increasingly used to (i) underspecify the scope of scope bearing operators and (ii) prevent the combinatorial problems raised during generation and machine translation by the recursive structure of lambda term and first order formulae (Bos, 1995; Copestake et al., 2001).

Our proposal builds on these two trends. It mimicks beta reduction using unification and uses a flat semantics to underspecify scope and facilitate processing.

The language L_U (for "underspecified logic") is a unification based reformulation of the PLU logic presented in (Bos, 1995). We give here an informal presentation of its syntax and semantics and refer the reader for more details to (Bos, 1995).

L_U describes first order logic formulae. Because we introduce unification variables to support semantic construction, we distinguish two types of L_U formulae: the *unifying formulae*, which contain unification variables, and the *saturated formulae* which are free of unification variables.

First we define the set of unifying formulae. Let I_{var} be a set of individual unification variables and I_{con} be a set of individual constants. Let H be a set of "hole" constants, L_{con} be a set of "label" constants and L_{var} be a set of "label" unification variables. Let R be a set of n-ary relations over $I_{var} \cup I_{con} \cup H$. Finally let $\geq$ be a relation on $H \cup L_{con}$ called "has-scope-over". Then the **unifying formulae** (UF) of L_U are defined as follows:

Given $l \in L_{var} \cup L_{con}$, $h \in H$, $i_1, \ldots, i_n \in I_{var} \cup I_{con} \cup H$ and $R^n \in R$. Then:

1. $l : R^n(i_1, \ldots, i_n)$ is a UF of L_U
2. $h \geq l$ is a UF of L_U
3. ϕ, ψ is a UF of L_U if ψ is a UF of L_U and ϕ is a UF of L_U
4. Nothing else is a UF of L_U

That is, unifying formulae of L_U consist of labelled elementary predications, scoping constraints and conjunctions. The **saturated formulae** of L_U are unifying formulae which are devoid of unification variables. The models these saturated formulae describe are first order formulae and are defined by the set of possible "pluggings" i.e., injections from the holes of a formula to the labels of this formula. Given a saturated formula $\phi \in L_U$, **a plugging P is possible for** ϕ iff ϕ is consistent with respect to this plugging.

Let us define in detail what this means. First, we introduce the relation $\geq_\phi$ on $L_\phi \cup H_\phi$ for a given saturated formula ϕ: for all $k, k', k'' \in L_\phi \cup H_\phi$:

1. $k \geq_\phi k$
2. $k \geq_\phi k'$ if $k \geq k'$ is in ϕ
3. $k \geq_\phi k''$ if $k \geq_\phi k'$ and $k' \geq_\phi k''$
4. if there is a $k : \tau$ in ϕ with k' occurring in τ, then $k \geq_\phi k'$ and $k' \not\geq_\phi k$
5. if k and k' are different arguments of the same R^n in ϕ (i.e., there is a $R^n(..., k, ..., k', ...)$ in ϕ), then $k \not\geq_\phi k'$ and $k' \not\geq_\phi k$
6. nothing else is in $\geq_\phi$

Condition 5. is important to separate for instance, between scope and restriction of a quantifier as nothing can be part of both at the same time.

Let P be an injection from H_ϕ to L_ϕ and let ϕ' be the result of replacing in ϕ all $k \in H_\phi$ with $P(k)$. Then P is a possible plugging for ϕ iff for all $k, k' \in L_\phi$: if $k \geq_{\phi'} k'$, then either $k = k'$ or $k' \not\geq_{\phi'} k$.

Intuitively, the set of possible pluggings for a given L_U formula defines the set of first order logic formulae which are described by this formula. The following example illustrates this. Suppose the sentence in (1) is assigned the L_U formula (2).

(1) Every dog chases a cat

(2) $l_0 : \forall(x, h_1, h_2), h_1 \geq l_1, l_1 : D(x), h_2 \geq l_2, l_2 : Ch(x, y), l_3 : \exists(x, h_3, h_4), h_3 \geq l_4, l_4 : C(y), h_4 \geq l_2$

1. There are well known empirical problems with this approach such as an incorrect treatment of certain conjunction cases. Nonetheless the order independence supported by unification means that in practice, most large coverage grammars continue to do unification based semantic construction.

Only two pluggings are possible for this formula in (2) namely $\{h_1 \rightarrow l_1, h_2 \rightarrow l_3, h_3 \rightarrow l_4, h_4 \rightarrow l_2\}$ and $\{h_1 \rightarrow l_1, h_2 \rightarrow l_2, h_3 \rightarrow l_4, h_4 \rightarrow l_0\}$. They yield the following meaning representations for (1):

l_0:∀(x,l_1 ,l_3), l_1:D(x),l_2:Ch(x,y),l_3:∃(x,l_4 ,l_2), l_4:C(y)

l_0:∀(x,l_1 ,l_2), l_1:D(x),l_2:Ch(x,y),l_3:∃(x,l_4 ,l_0), l_4:C(y)

In what follows, we use the following notational conventions. We write $l_0, l_1, \ldots$ for label unification constants, $s_0, s_1, \ldots$ for label unification variables, $a, b, \ldots$ for individual unification constants and $x_0, x_1, \ldots$ for individual unification variables.

3 A unification based Syntax-Semantics interface for TAG

An FTAG consists of a set of (auxiliary or initial) elementary trees and two tree composition operations: substitution and adjunction. Substitution is the standard tree operation used in phrase structure grammars while adjunction – sketched in Fig. 1 – is an operation which inserts an auxiliary tree into a derived tree. To account for the effect of these insertions, two feature structures (called **top** and **bottom**) are associated with each tree node in FTAG. The top feature structure encodes information that needs to be percolated up the tree should an adjunction take place. In contrast, the bottom feature structure encodes information that remains local to the node at which adjunction takes place.

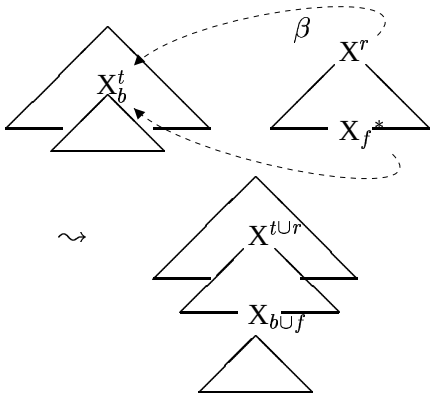

FIG. 1 – *Adjunction in FTAG*

To construct semantic representations on the basis of the derived tree, we proceed as follows.

First we associate each elementary tree with an L_U formula representing its meaning. Second we decorate some of the tree nodes with unification variables and constants occuring in the L_U formula. The idea behind this is that the association between tree nodes and unification variables encodes the syntax/semantics interface – it specifies which node in the tree provides the value for which variable in the final semantic representation.

As trees combine during derivation, two things happen: (i) variables are unified – both in the tree and in the associated semantic representation – and (ii) the semantics of the derived tree is constructed from the conjunction of the semantics of the combined trees. A simple example will illustrate this.

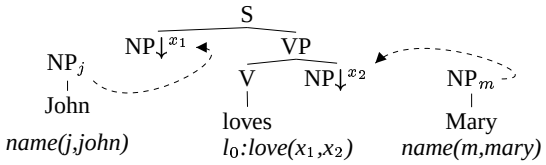

FIG. 2 – *"John loves Mary"*

Suppose the elementary trees for "John", "loves" and "Mary" are as in Fig. 2 where a downarrow (↓) indicates a substitution node and C^x/C_x abbreviate a node with category C and a top/bottom feature structure including the feature-value pair $\{$ **index :** $x\}$. On substitution, the root node of the tree being substituted in is unified with the node at which substitution takes place. Further, when derivation ends, the top and bottom feature structures of each node in the derived tree are unified. Thus in this case, x_1 is unified with j and x_2 with m. Hence, the resulting semantics is:

$$l_0 : love(j, m), name(j, john), name(m, mary)$$

4 Some further examples

For lack of space, we cannot here specify the general principles underlying the semantic labelling of lexical trees in a unification based TAG grammar. Instead, we focus on a number of linguistic phenomena which are known to be problematic for TAG based semantic construction and show how they can be dealt with in the proposed framework.

4.1 Quantification

In some TAG approaches (Hockey and Mateyak, 2000; Abeillé, 1991; Abeillé et al., 2000), and in

particular in Abeillé's grammar for French, quantifiers are treated as adjuncts. First, the noun is added to the verb by substitution then, the quantifying determiner is adjoined to the noun (see Fig.3).

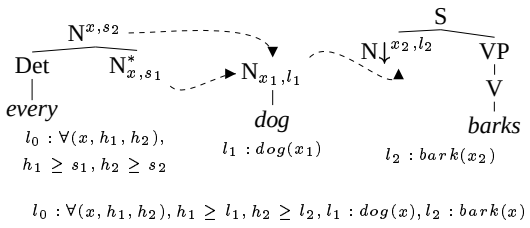

FIG. 3 – *Quantifiers*

Semantically, a quantifying determiner expresses a relation between the denotation of some external verbal argument (the quantifier *scope*) and that of its nominal argument (the quantifier *restriction*). In the flat semantics we are using, this is captured by associating with "every" the formula

$$l_0 : \forall(x, h_1, h_2), h_1 \geq s_1, h_2 \geq s_2$$

where the two label variables s_1, s_2 indicate the missing arguments. During semantic construction, these two variables must be unified with the appropriate values, namely with the labels of the restriction and of the scope respectively (e.g., in our example with the labels l_1 and l_2). Moreover the variable x bound by the quantifier must be unified with the variables x_1 and x_2 predicated of by the noun and the verb respectively.

To account for these various bindings, we proceed as follows. First, we associate with the relevant tree nodes not only an index but also a label so that C^{x,l}/C$_{x,l}$ now abbreviate a node with category C and a top/bottom feature structure including the feature-value pair $\{$ **index :** x, **label :** $l\}$. Second, we distribute these variables between top and bottom information so as to correctly capture the semantic dependencies between determiner, scope and restriction. More specifically, note that the restriction label variable (s_1) is part of the bottom feature structure of the foot node. In this way, s_1 remains local to the N* node and unifies with the bottom-label of the root node of the tree to which the determiner adjoins. By contrast, the scopal label variable s_2 (whose value is fixed by the verb) is included in the top feature structure of the root node of the determiner tree. It thereby can be percolated up to the NP argument node of the verb and thus unified with the label made available at that node i.e.,, with the verb label (l_2). Since the variable x bound by the quantifier is shared by both scope and restriction, it is included in both the top feature structure of the determiner root node and the bottom feature structure of the determiner foot node. As a result, x is unified with both x_1 and x_2.

As should be obvious, the approach straightforwardly extends to scope ambiguities: by a derivation process similar to that sketched in Figure 3, the semantic representation obtained for a sentence with two quantifiers such as (1) above will be (2) which, as seen in section 2 above, describes the two formulae representing the possible meanings of "every dog chases a cat".

4.2 Intersective Adjectives

In a Montague style semantics, an intersective adjective denotes a function taking two arguments (an individual and a property) and returning a proposition. Using a flat semantics, this intuition can be captured by having adjectives binding both an individual and a label variable. Thus in Fig. 4, the adjective "black" is associated with the semantic representation $s_3 : black(x_1)$ where s_3 is a label variable and x_1 an individual variable. Since the values of these variables are provided by the modified noun and since the combination of adjective and noun is mediated by adjunction, these variables label the bottom feature structure of the adjective tree foot node. On adjunction, this bottom feature structure is then unified with that of the argument noun (itself labelled with its own index and label) so that noun and adjective end up with identical index and label. Note that as the adjective "passes up" index and label information to the adjective tree root node, combination with a quantifier will further bind the index now shared by noun and adjective to the quantifier index.

Although we cannot present it here for lack of space, the approach can also be extended to deal with non subsective adjectives and account for cases such as "the former king" (and similarly for adverbs modifying adjectives "the potentially contro-

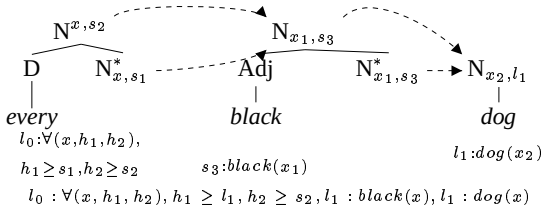

FIG. 4 – *Intersective adjectives*

versial plan") where the individual predicated of is actually not a king (or a controversial plan). In that case the predicate associated with the adjective must label the adjective node thereby providing a value for its modifier.

4.2.1 VP and S modifiers

Consider the following examples.

(3) a. Pat allegedly usually drives a Cadillac.
b. Intentionally, John knocked twice.
c. John intentionally knocked twice.

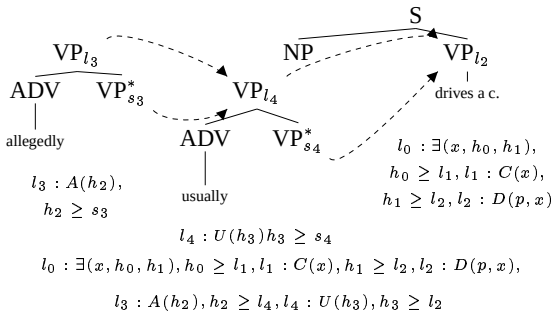

FIG. 5 – *VP opaque modifier*

The sentence in (3a) has three readings depending on the respective scope of "allegedly", "usually" and "a cadillac". However in all three cases, "allegedly" scopes over "usually". Further, there are two possible readings for both (3b) and (3c) depending on whether "intentionally" scopes over "twice" or the converse.

The first example can be captured as suggested in (Kallmeyer and Joshi, 2002) by ruling out multiple adjunctions (one VP modifier is adjoined to the other rather than both modifiers being applied to the verb) and treating "usually" as an "opaque" modifier i.e., one that does not pass up the verb label (cf. Fig. 5).

By contrast, "intentionally" (a so-called "subject adverb" with the associated scoping properties) and "twice" (a postposed VP adverb) are treated as non opaque in that they pass up the verb (rather than their own) label to the bottom feature structure of their root node. Thereby, scope bearing elements occurring further up in the derived tree bind the verb label. E.g., in (3b) and (3c), the two adverbs consume and pass on the verb label so that the following L_U formula is obtained:

$$l_1 : I(h_1), h_1 \geq l_0, l_2 : T(h_2), h_2 \geq l_0, l_0 : K(j)$$

4.3 Control verbs

In a subject control sentence, "controller" (the denotation of the subject of the control verb) and "controlee" (the denotation of the unexpressed subject of the complement) must be identified. This is clearest with ditransitive control verbs such as "promise". Given the sentence

(4) John promised Mary to leave

the meaning representation must make clear that the unexpressed subject of "leave" is "John".

Fig. 6 sketches the elementary trees associated in FTAG with a control verb and its complements. As the figure shows, it is easy to associate these trees with semantic information that yields the desired dependencies and in particular, the coreference between the implicit subject of the sentential complement and that of the control verb.

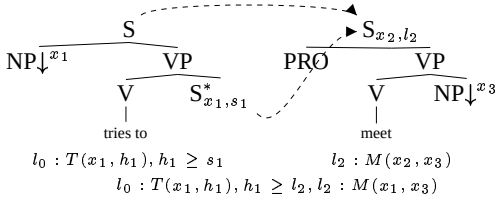

FIG. 6 – *Control verbs*

5 Related work

We now compare our approach with three related proposals: that of basing semantic construction on the TAG derivation tree as put forward in (Kallmeyer and Joshi, 2002); an extension of this proposal presented in (Kallmeyer, 2002b) and the

glue semantic approach proposed in (Frank and van Genabith, 2001).

5.1 Semantic construction and the derivation tree

The LTAG derivation tree records how elementary trees are combined during derivation. Hence the nodes of this tree stand for elementary trees and the arrows either for substitution or for adjunction. In what follows an upward pointing arrow indicates an adjunction, a downward one a substitution. As, e.g., (Kallmeyer and Joshi, 2002) shows, semantic construction can be based on the derivation tree as follows.

First elementary trees are associated with semantic representations. The derivation tree is then used to determine functor-argument dependencies: an (upwards or downwards going) arrow between n_1 and n_2 indicates that n_1 is a semantic functor and n_2 provides its argument(s).

Although the approach works well in general, it is known that derivation trees do not provide all the necessary functor-argument dependencies.

A first problem case is embodied by quantifiers. As we saw in section 4, quantifiers are semantic functors taking two arguments namely, a restriction and a scope. Further it has been argued mainly for French but also for English that syntactically a quantifier should be adjoined to its complement noun. As a result the derivation tree of a quantified intransitive sentence as in Fig. 3 is as given in Fig. 7. As observed in (Kallmeyer, 2002b), this is problematic for semantic construction because there is no arrow pointing from the determiner to its scope hence no base on which to determine the scope of the quantifier. This can be solved however by using multi-component TAG to represent a quantifier with two trees, one representing the relation between determiner and restriction, the other representing the relation between determiner and scope (Kallmeyer and Joshi, 2002).

A second problem is illustrated by wh-questions. In that case, an element (the wh-word) has a dual semantic function: on the one hand, it provides a verb argument and on the other, it takes scope over a (possibly complex) sentence. In Fig. 7, we give the derivation tree for the sentence

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

As can be seen there is no direct link between "who" and the verb introducing its scoping sentence, namely "think". Hence the scoping relation between "who" and "does Paul think John said Bill likes" cannot be captured.

A third type of problems occur when several trees are adjoined to distinct nodes of the same tree. This typically occurs when raising verbs interact with long distance dependencies e.g.,

(6) Mary Paul claims John seems to love.

As the derivation tree in Fig. 7 shows, the multiple adjunction of the trees for "claim" and "seems" to (respectively the S and the VP node of) "love" result in a derivation tree where no link occurs between "claim" and "seem". But obviously this is needed as the "seems" sentence provides the propositional argument expected by "claims".

None of these cases are problematic for the derived tree based approach. Quantifiers are treated as described in section 4 while examples (5) and (6) are treated as sketched in figures 8 and 9.

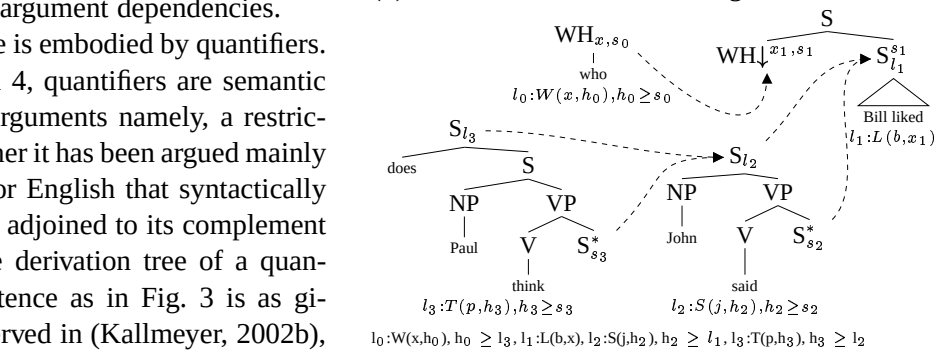

FIG. 8 – *Wh-questions*

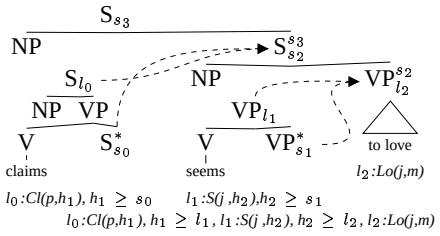

FIG. 9 – *Raising verbs*

5.2 Derivation trees with additional links

(Kallmeyer, 2002b; Kallmeyer, 2002a) shows that some of the problems just described can be solved once additional links are added to the derivation

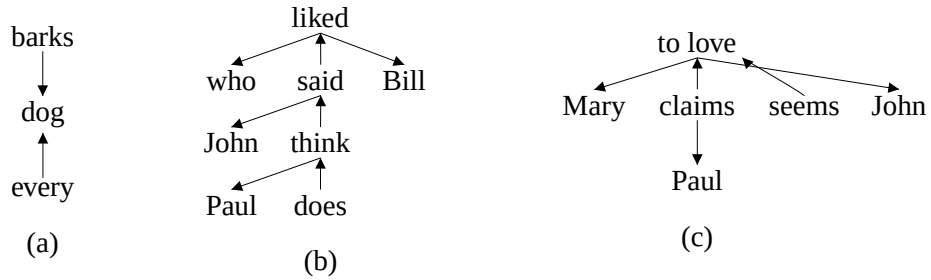

FIG. 7 – *Derivation trees*

tree. In particular, given three nodes n_1, n_2, n_3 such that n_1 is above n_2 and n_3 is above n_3, if n_3 is a tree *adjoined at the root of* n_2, then an additional link can be established between n_1 and n_3. In this way, adjoining quantifiers become unproblematic as an additional link is established between "barks" and "every" thereby supporting the semantic relation between the quantifier and its scope. (Kallmeyer, 2002a) further shows that the approach can deal with questions.

Nonetheless since additional links only are warranted when adjunction takes place at a root node, the approach does not straightforwardly extend to cases such as (6) where none of the two problematic adjunctions takes place at the root node of the "love" tree; or to derivations such as illustrated in Fig. 6 where "john" is substituted into the tree for "try" which itself is adjoined to the tree for "meet" ("john" does not adjoin to the root node of "try", hence no additional link is warranted between "john" and "meet").

5.3 Glue semantics

The present approach is closest to the glue semantics approach presented in (Frank and van Genabith, 2001). As in our proposal, meaning representations are associated with elementary trees, variables are shared by the nodes of the elementary trees and the meaning representations and semantic construction is based on the derived, rather than on the derivation tree.

There are two main differences though.

The first resides in the tools used to do semantic construction. In a traditional Montague type approach to semantic construction, the assumption that semantic composition follows surface constituent structure results in the stipulation of (sometimes extremely) complex lambda terms as lexical meaning representations. In a medium size grammar, the complexity induced by this assumption is non-trivial and adds to the complexity of the already difficult task of grammar writing. In effect, unification-based semantic construction and glue semantics provide two different ways of addressing this problem. Glue semantics uses linear logic and deduction to combine semantic meanings on the basis of a functional structure wheras the approach proposed here uses unification to do bracketting independent semantic construction on the basis of constituent structure.

The second difference lies in the way variables are assigned a value. In the (Frank and van Genabith, 2001)'s approach, the assignment of values to variables results from the additional stipulation of a series of variable equation principles: one for substitution, another for adjunction of a modifier auxiliary tree and a third one for the adjunction of a predicative auxiliary tree. By contrast, in the present approach, this process is mediated by unification and follows from the definition of the substitution and adjunction operation in FTAG. Since these definitions are already needed for morpho-syntax, it seems a priori better to use them rather than to add additional stipulations for semantics. Further, for the range of phenomena discussed in (Frank and van Genabith, 2001), such additional stipulations do not seem needed within the flat semantic framework we adopt. Finally, the chosen unification based semantic construction method together with the choice of a flat semantics means that the ideas developped within the wide coverage and freely available HPSG grammar ERG can be drawn upon when developing a large scale TAG with semantic information.

6 Implementation

There are at least two obvious ways to implement the above proposal. A first possibility is to keep elementary trees and associated semantic representations separate and to specify a parser which combines not just trees but pairs of trees and semantic representations. The second possibility is to integrate the semantic representations into the elementary trees under some priviledged feature say **sem** and to take the semantic representation of a derived tree to be the unioned values of this **sem** feature[2].

We are currently experimenting with the second possibility but within a parsing framework which uses the "polarities" presented in (Perrier, 2000) to drastically reduce the parsing search space. Preliminary results are encouraging as for the small but non trivial grammar fragment available, polarities can be shown to restrict the output to only exactly as many parses as there are possible syntactic and semantic representations for the input sentence.

7 Conclusion

We have shown how FTAG could be used to construct flat semantic representations during derivations and compared this approach with related proposals. Future work will concentrate on (i) implementing and extending the present fragment, (ii) integrating the present proposal within a metagrammar for FTAG so as to factorise semantic information and automatically produce FTAGs with a semantic dimension and (iii) investigating how semantic information could be used to prune parse forests and improve parsing performance.

Acknowledgments

The cooperation between the authors leading to this paper was made possible by the INRIA ARC GENI (Generation and Inference). We are grateful to Anette Frank, Josef van Genabith, Aravind Joshi, Maribel Romero and three anonymous reviewers for their comments on this paper.

2. Because we use a flat semantics, the feature structures needed to represent a tree meaning need not be recursive and given some arbitrary but reasonable bound on the set of labels, individuals and holes used in a derivation, it might still be possible in that case to have an FTAG that has the same generative capacity as a TAG.

References

A. Abeillé, M.H. Candito, and A. Kinyon. 2000. The current status of FTAG. In *Proceedings of TAG+5*, pages 11–18, Paris.

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

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

A. Copestake, A. Lascarides, and D. Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France.

M. Dalrymple. 1999. *Semantics and syntax in lexical functional grammar*. MIT Press.

A. Frank and J. van Genabith. 2001. GlueTag. Linear Logic based Semantics for LTAG. In M. Butt and T. Holloway King, editors, *Proceedings of the LFG01 Conference*, Hong Kong.

B. A. Hockey and H. 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.

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

L. Kallmeyer and A. K. Joshi. 2002. Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG. *Research on Language and Computation*. To appear.

L. Kallmeyer. 2002a. Enriching the TAG Derivation Tree for Semantics. In *Proceedings of KONVENS 2002*, pages 67 – 74, Saarbrücken, October.

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

R. Montague. 1974. The Proper Treatment of Quantification in Ordinary English. In Richmond H. Thomason, editor, *Formal Philosophy: Selected Papers of Richard Montague*, pages 247–270. Yale University Press, New Haven.

G. Perrier. 2000. Interaction grammars. In *Proceedings of 18th International Conference on Computational Linguistics (CoLing 2000)*, Saarbrücken.

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

H. Zeevat, E. Klein, and J. Calder. 1987. An introduction to unification categorial grammar. In *Categorial Grammer, Unification grammar and parsing*. University of Edinburgh.