Using an Enriched TAG Derivation Structure as Basis for Semantics

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

Abstract

Most of the proposals for semantics in the Tree Adjoining Grammar (TAG) framework suppose that the derivation tree serves as basis for semantics. However, in some cases the derivation tree does not provide the semantic links one needs. This paper concentrates on one of these cases, namely the analysis of quantifiers as adjuncts. The paper proposes to enrich the TAG derivation tree and use the resulting structure as basis for semantics. This allows to deal with quantifiers, even in PPs embedded into NPs, such that an adequate semantics with appropriate scope orders is obtained. The enriched derivation structure allows also to treat other cases that are problematic for the assumption that a TAG semantics can be based on the derivation tree.

1. TAG and the syntax-semantics interface

1.1. Lexicalized Tree Adjoining Grammars (LTAG)

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

LTAG derivations are represented by derivation trees that record the history of how the elementary trees are put together. A derived tree is the result of carrying out the substitutions and adjoinings. For a sample derivation see the TAG analysis of (1) in Fig. 1. The numbers at the nodes in the derivation tree are the positions of the nodes where the trees are added: *John* is substituted for the node at position (1), *Mary* for the node at position (22) and *always* is adjoined to the node at position (2).

(1) John always loves Mary.

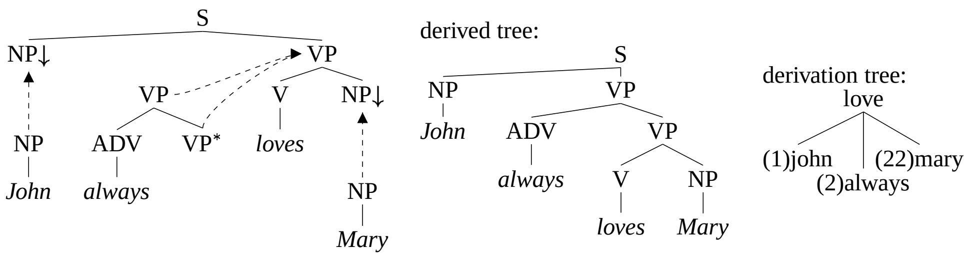


Figure 1: TAG derivation for (1)

1.2. Compositional semantics with LTAG

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

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

(2)	l_1 : *love'*(x_1, x_2) $h_1 \geq l_1$	*john'*(x)	l_2 : *always'*(h_2) $g_1 \geq l_2, h_2 \geq s_1$	*mary'*(y)
	arg: $\langle x_1, (1)\rangle, \langle x_2, (22)\rangle$	arg: –	arg: g_1, s_1	arg: –

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

The use of holes is motivated by the desire to generate underspecified representations (as in, e.g., Bos, 1995) for scope ambiguities. In the end, after having constructed a semantic representation with holes and labels, disambiguation is done which consists of finding bijections from holes to labels that respect the scope constraints and that are such that no label is below two labels that are siblings (e.g., this ensures that nothing can be in the restriction and the body of a quantifier at the same time). In the semantic representation for *love*, there is for example a hole h_1 above the label l_1 (indicated by the constraint $h_1 \geq l_1$). Between h_1 and l_1, other labels and holes might come in (introduced for example by quantifiers or adverbs) or, if this is not the case, l_1 will be assigned to h_1 in the disambiguation(s).

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

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

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

(3)	l_1 : *love'*(x, y), *john'*(x), *mary'*(y), l_2 : *always'*(h_2) $h_1 \geq l_1, h_1 \geq l_2, h_2 \geq l_1$
	arg: –

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

1.3. Separating scope and predicate argument information

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

(4) every dog barks

Fig. 2 shows the syntactic analysis of (4) in this framework. The semantic representations corresponding to the four elementary trees are shown in (5).

(5)	l_1 : *bark'*(x_1) $l_1 \leq h_1$	l_2 : *every'*(x, h_2, h_3) $s_1 \leq h_3$	$l_3 : p_1(x)$ $l_3 \leq h_2$	q_1 : *dog'*
	arg: x_1	arg: s_1	arg: p_1	arg: –

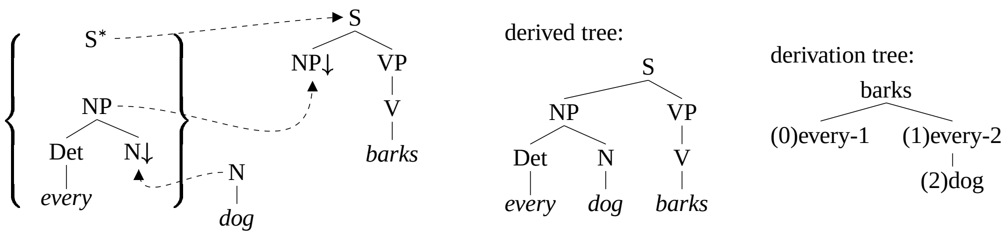


Figure 2: Syntactic analysis of (1)

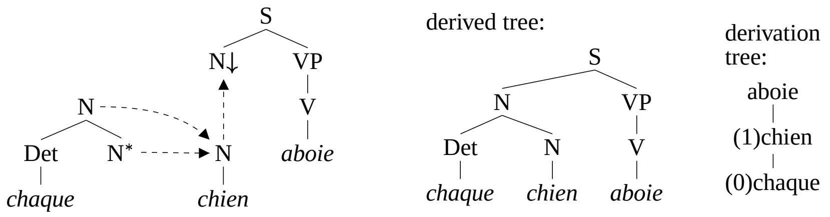


Figure 3: Syntactic analysis of (7)

The scope part of the quantifier (second representation in (5)) introduces a proposition containing the quantifier, its variable and two holes for its restrictive and nuclear scope. The proposition this semantic representation is applied to (variable s_1) is in the nuclear scope of the quantifier ($s_1 \leq h_3$). The predicate argument part (third representation in (5)) introduces a proposition $p_1(x)$ where p_1 will be the noun predicate *dog'*. This proposition is in the restrictive scope of the quantifier ($l_3 \leq h_2$). The values for the argument variables are $x_1 \to x, s_1 \to l_1, p_1 \to q_1$ which leads to (6). The only disambiguation is $h_1 \to l_2, h_2 \to l_3, h_3 \to l_1$ which leads to *every'*$(x,$ *dog'*$(x),$ *bark'*$(x))$.

(6)

l_1 : *bark'*(x), l_2 : *every'*(x, h_2, h_3), l_3 : *dog'*(x) $l_1 \leq h_1, l_1 \leq h_3, l_3 \leq h_2$
arg: –

To account for cases with more than one quantifier, a restricted use of multiple adjunctions is necessary.

2. Quantifiers as adjuncts

2.1. The problem

The approach of Kallmeyer and Joshi is problematic in cases where quantifiers are analyzed as adjuncts. Such an analysis however is proposed for English in (Hockey and Mateyak, 2000) and for French in (Abeillé, 1991). Abeillé's proposal for French is even adopted in the French TAG implemented at the University Paris 7 (Abeillé, Candito and Kinyon, 2000). In the following we will sketch the French quantifier analysis.

(7) chaque chien aboie

According to Abeillé (1991), in (7), the French translation of (4), the noun is first added to the verb by substitution, and then the quantifier is adjoined to the noun (see Fig. 3). In the derivation tree, there is no link (i.e., no edge) between the quantifier and the verb. However, the variable introduced by the quantifier is an argument of the verb, and, furthermore, the proposition introduced by the verb is part of the nuclear scope of the quantifier. (This is why l_1 was assigned to the argument variable s_1 in (5).) Therefore, for semantics, a link between the quantifier and the verb is needed.

The use of a second elementary tree for the scope part that is adjoined to the whole sentence would require non-local MCTAG since the quantifier is not adjoined to the verb but to the noun. Non-local MCTAG is much more powerful than TAG and a solution using TAG or a mildly context-sensitive TAG variant is preferable. Therefore I will not adopt the idea of (Kallmeyer and Joshi, 2002) to separate the contribution of a quantifier into two parts.

An advantage of not doing so is that multiple adjunctions are not necessary. Multiple adjunctions are problematic because in combination with multicomponent derivations, they extend the generative power of the grammar, and therefore their use needs to be restricted.

2.2. Enriching the derivation tree

The syntactic analysis of (7) shows that, if one wants to retain the semantic analysis given in (5) for quantifiers, the quantifier *chaque* in Fig. 3 needs to have access to both elementary semantic representations, the one of *aboie* and the one of *chien*. This means that the derivation tree is too restrictive, it does not provide the dependency structure one needs for semantics. On the other hand, the way syntactic elements are put together in a derivation reflects predicate-argument relations, i.e., seems still to provide the dependencies semantics should be based on. Therefore this paper proposes to keep the idea of using the derivation tree for semantics, but to enrich the derivation tree in order to obtain the semantic links one needs.

The basic intuition is as follows: In Fig. 3 for example, the quantifier is adjoined to the root of the initial tree for *chien*, and consequently in the derived tree it will be a direct neighbour not just of the elementary tree for *chien* but also of the one for *aboie* (see Fig. 4 where the elementary tree of the quantifier is marked by a triangle). Therefore there is a syntactic link between the quantifier and the verb and this should enable the quantifier to have semantic access to the verb. For this reason, in the case of an adjunction at a root node of some elementary γ, the adjoined tree is not only connected to γ but also to the tree to which γ was added in some previous derivation step. The enriched derivation structure used for semantics is called *e-derivation structure* for short. The e-derivation structure of (7) is shown in Fig. 4, the additional link, i.e., the one that does not be part of the derivtaion tree, is depicted as a dotted edge.

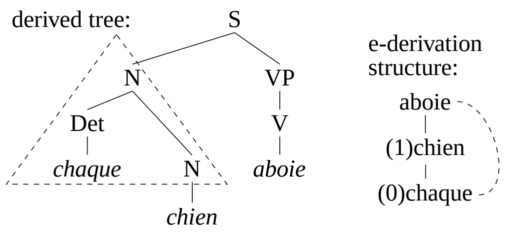


Figure 4: Derived tree and e-derivation structure of (7)

In a feature-structure based TAG (FTAG, (Vijay-Shanker and Joshi, 1988)), the additional connection between the qunatifier and the verb is even more obvious, since a unification of feature structures from all three trees (here *aboie*, *chaque* and *chien*) takes place. See Fig. 5 for a general FTAG derivation sequence of substitution and adjunction at the root of the tree that was added by substitution. In the derived tree, the root of the adjoined tree carries a top feature structure that results from unifying feature structures from all three trees involved in this derivation. In this sense, the links in a e-derivation structure reflect unifications of feature structures between elementary trees.

This feature-structure view is related to the question whether in the derived tree in Fig. 4, it is appropriate to consider the upper N node as being part of just the elementary tree of the quantifier, as done in standard TAG, instead of considering it as a node shared between the three elementary trees. I will not further pursue this question in this paper.

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

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

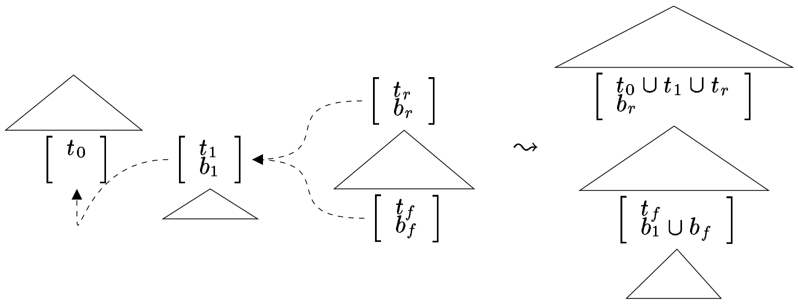


Figure 5: E-derivation structure reflects unifications of feature structures between elementary trees

of (7) with γ being the elementary tree of *aboie*, γ' being the elementary tree of *chien* and $\beta_1 = \beta_n = \beta$ being the one of *chaque*.) Fig. 6 sketches the general definition of the e-derivation structure.

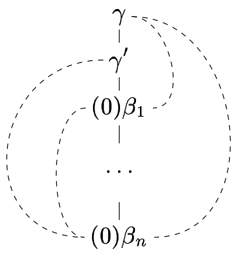


Figure 6: E-derivation structure

Two kinds of edges are distinguished in e-derivation structures: edges belonging also to the derivation tree are called *primary edges* whereas edges that do not belong to the derivation tree are called *secondary edges*. We will see that this is a useful distinction, since primary edges seem to be stronger with respect to semantics.

The e-derivation structure is a graph, not necessarily a tree. This is one of the differences compared to the meta-level derivation structure proposed in (Dras, Chiang and Schuler, 2000) that is also intended to represent a “more semantic” dependency structure than the original derivation tree.

It is important to emphasize that the e-derivation structure is a way of making information about shared nodes (or in an FTAG shared feature structures) explicit that is already present in the original derivation tree. In this sense a semantics based on the e-derivation structure is still a semantics based only on the derivation tree.

For the semantics of (7), the representations in (8) are used. These are more or less the same as in (Kallmeyer and Joshi, 2002), except that the quantifier contribution is not separated into two parts.

(8)

$l_1 : \textit{aboire'}(x_1)$ $l_1 \le h_1$
arg: $\langle x_1, (1)\rangle$

$l_2 : \textit{chaque'}(x, h_2, h_3), l_3 : p_1(x)$ $s_1 \le h_3, l_3 \le h_2$
arg: s_1, p_1

$q_1 : \textit{chien'}$
arg: –

I will keep the idea that in case of a substitution, the variable assignment is downwards while in case of an adjunction it is upwards. An argument variable linked to a substitution position p receives its value from an elementary tree below or equal to the tree substituted at position p. A variable that is not linked to a substitution position and that belongs to a tree that is added by adjunction receives its value from some tree above the adjoined tree. Here, ‘below’ and ‘above’ refer to the derivation tree, i.e., the primary edges in the e-derivation structure.

According to the e-derivation structure of (7), the only possible assignment for the argument variables is $x_1 \to x, s_1 \to l_1, p_1 \to q_1$ which leads to (9).

(9)

$l_1 : \textit{aboire'}(x), l_2 : \textit{chaque'}(x, h_2, h_3), l_3 : \textit{chien'}(x)$ $l_1 \le h_1, l_1 \le h_3, l_3 \le h_2$
arg: –

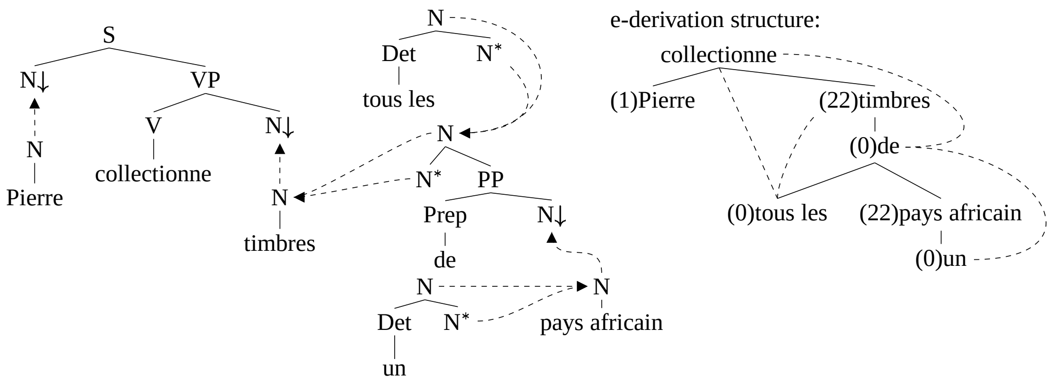


Figure 7: Syntactic analysis and e-derivation structure of (10)

3. Quantifiers embedded into NPs

One of the first questions to be considered is whether the analysis proposed above allows quantifiers embedded into NPs to have wide scope. Consider (10) and (11).

(10) Pierre collectionne tous les timbres d'un pays africain
'Pierre collects every stamp of an African country'

(11) Pierre connaît tous les détenteurs d'un prix
'Pierre knows every winner of a price'

Both sentences are ambiguous with respect to quantifier scope: the two meanings of (10) are that Pierre collects either 1) all the stamps coming from African countries (wide scope of *tous les timbres*) or 2) the stamps of a single specific African country (wide scope of *un pays africain*). The two meanings of (11) are that either 1) Pierre knows everybody who obtained some price (wide scope of *tous les détenteurs*) or 2) there is a specific price such that Pierre knows everybody who obtained that price (wide scope of *un prix*). An adequate semantic analysis should allow for both scope orders, if possible they should be represented in one underspecified semantic representation.

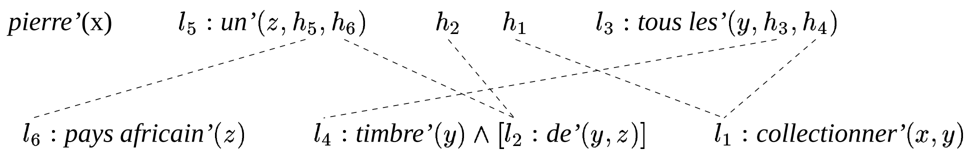
There is a crucial difference between (10) and (11). In (10) the embedded PP is not an argument and therefore, in the FTAG analysis, it is adjoined to the noun of the higher NP, whereas in (11) the PP *d'un prix* is an argument of the noun *détenteur* and therefore it is added by substitution.[1] In the following, I will consider in detail the two syntactic and semantic analyses.

3.1. PPs as noun adjuncts

For (10), the elementary trees and the way they are put together is shown in Fig. 7 (leaving aside the decomposition of *pays africain*). As it is traditionally done in TAG, I suppose NA conditions for foot nodes. The elementary trees shown in Fig. 7 allow a second analysis, namely adjoining *tous les* at the root of *timbres* and then adjoining *de* at the root of *tous les*. This would lead to another derived and another derivation tree but the e-derivation structure would be the same, except for the distinction between primary and secondary edges. However, the use of adequate features can block this second derivation. This is for example done in the French TAG Grammar (Abeillé, Candito and Kinyon, 2000). Therefore, in the following, I will only consider the analysis in Fig. 7.

The semantic analysis that I propose in the following is such that the scope of a quantifier is not restricted by something higher in the scope order and therefore, in the case of (10), *un* can rise and get scope over *tous les*. The semantic representations for the elementary trees in Fig. 7 are shown in Fig. 12. The ones for *collectionne, Pierre, timbres, tous les, pays africain* and *un* follow the proposals made in Section 2.2. The semantic representation

1. Interestingly, quantifiers in adjunct PPs seem to have a stronger preference for wide scope than quantifiers in argument PPs. I would like to pursue the issue of scope preferences in future work, but in this paper this is left aside.



pierre'(x)
l_5 : *un'*(z, h_5, h_6)
h_2
h_1
l_3 : *tous les'*(y, h_3, h_4)
l_6 : *pays africain'*(z)
l_4 : *timbre'*$(y) \wedge [l_2$: *de'*$(y, z)]$
l_1 : *collectionner'*(x, y)

Figure 8: Graphical representation of the scope constraints for (10)

of *de* is such that it takes a predicate p_1, in this case *timbre'*, and modifies it such that *timbre'* is replaced by $\lambda u[\textit{timbre'}(u) \wedge [l_2 : \textit{de'}(u, x_3)]]$. The propositional label l_2 is needed in order to make *de'*(u, x_3) accessable for quantifiers. When adding *un*, l_2 is assigned to s_2.

(12)
$$\boxed{\begin{array}{l} l_1 : \textit{collectionner'}(x_1, x_2) \\ l_1 \leq h_1 \\ \hline \text{arg: } \langle x_1, 1\rangle, \langle x_2, 22\rangle \end{array}} \quad \boxed{\begin{array}{l} \textit{pierre'}(x) \\ \hline \text{arg: } - \end{array}} \quad \boxed{\begin{array}{l} q_1 : \textit{timbre'} \\ \hline \text{arg: } - \end{array}} \quad \boxed{\begin{array}{l} q_2 : \lambda u[p_1(u) \wedge [l_2 : \textit{de'}(u, x_3)]] \\ l_2 \leq h_2 \\ \hline \text{arg: } p_1, \langle x_3, 22\rangle \end{array}}$$
$$\boxed{\begin{array}{l} l_3 : \textit{tous les'}(y, h_3, h_4), l_4 : p_2(y) \\ s_1 \leq h_4, l_4 \leq h_3 \\ \hline \text{arg: } s_1, p_2 \end{array}} \quad \boxed{\begin{array}{l} q_3 : \textit{pays-africain'} \\ \hline \text{arg: } - \end{array}} \quad \boxed{\begin{array}{l} l_5 : \textit{un'}(z, h_5, h_6), l_6 : p_3(z) \\ s_2 \leq h_6, l_6 \leq h_5 \\ \hline \text{arg: } s_2, p_3 \end{array}}$$

Taking the semantic representations from (12) and the e-derivation structure from Fig. 7, the semantic assignments when building a semantic representation of (10) are as follows:

For x_1, x_2 and x_3 free individual variables that are not argument variables must be found. The value of x_1 has to come from something (below the tree) substituted at position (1), therefore $x_1 \rightarrow x$. The value of x_2 has to come from some tree that is (below the tree) substituted at position (22) and that is linked to *collectionne*. Consequently, it has to come from *timbres*, *de* or *tous les*. The only possibility is $x_2 \rightarrow y$. For x_3, the value needs to come from one of the elementary trees added below the position (22) to the elementary tree of *de*, i.e. the value is taken from *pays africain* or *un*. Consequently, $x_3 \rightarrow z$.

For p_1, since it is not linked to a substitution position, one needs to find a unary predicate label or free variable in one of the elementary representations already present when adjoining the elementary tree of *de*, i.e., a value coming from *timbres* or *collectionne*. The only possibility is $p_1 \rightarrow q_1$. The values of s_2 and p_3 come from *de* or *pays africain*. Consequently $s_2 \rightarrow l_2$ and $p_3 \rightarrow q_2$ or $p_3 \rightarrow q_3$. With $s_2 \rightarrow l_2, p_3 \rightarrow q_2$ is not possible. Otherwise, l_2 would be in the restriction of *un'* ($h_5 \geq l_6 > l_2$) and in its body ($l_2 \leq h_6$). This is a contradiction to the definition of well-formed semantic representations. Therefore $p_3 \rightarrow q_3$.

The values for s_1 and p_2 must be taken from *de*, *timbres* or *collectionne*. For s_1 we get two possibilities, l_1 or l_2 and for p_2 possible values are q_1 or q_2.

For p_2, $p_2 \rightarrow q_2$ is preferable because $p_2 \rightarrow q_1$ would mean that we first have to do this assignment, i.e. to produce $l_4 : (q_1 : \textit{timbre'})(y)$ and then to perform $p_1 \rightarrow q_1$, i.e., to produce $l_4 : \textit{timbre'}(y) \wedge [l_2 : \textit{de'}(y, z)]$ because this last makes q_1 disappear (see the definition of semantic composition in (Kallmeyer and Joshi, 2002)). On the other hand, with $p_2 \rightarrow q_2$, the order in which the semantic representations are put together, does not matter. Therefore the last assignment is preferable and I propose to adopt the rule that, in case of two possibilities where just one corresponds to a primary edge in the e-derivation structure (in this case the one between *tous les* and *de*, i.e., $p_2 \rightarrow q_2$), this last one is chosen.

For s_1, $s_1 \rightarrow l_2$ is excluded because it would lead to a contradiction: it would lead to the constraints $l_2 \leq h_4$ and $l_4 \leq h_3$. With $l_2 \leq l_4$ (because the proposition labeled l_2 is embedded in the one labeled l_4, this gives $l_2 \leq h_4$ and $l_2 \leq h_3$, i.e. l_2 is in the restriction and the body of *tous les'*. Therefore $s_1 \rightarrow l_1$.

The semantic representation one obtains is (13):

(13)
$$\boxed{\begin{array}{l} l_1 : \textit{collectionner'}(x, y), \textit{pierre'}(x), l_3 : \textit{tous les'}(y, h_3, h_4), \\ l_4 : (q_2 : \lambda u[\textit{timbre'}(u) \wedge [l_2 : \textit{de'}(u, z)]])(y), l_5 : \textit{un'}(z, h_5, h_6), l_6 : (q_3 : \textit{pays africain'})(z) \\ l_1 \leq h_1, l_2 \leq h_2\ l_1 \leq h_4, l_4 \leq h_3\ l_2 \leq h_6, l_6 \leq h_5 \\ \hline \text{arg: } - \end{array}}$$

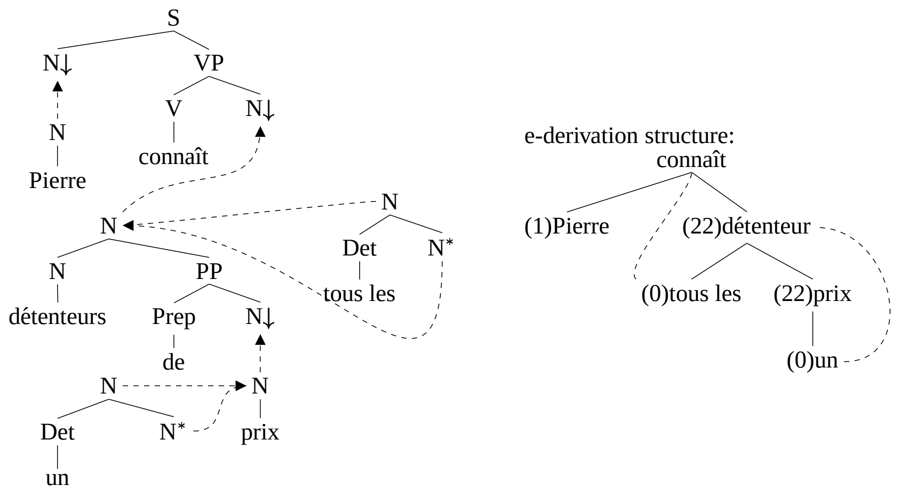


Figure 9: Syntactic analysis of (11)

The scope constraints in (13) are depicted in Fig. 8. As one can see, there is no constraint restricting the order of l_5 and l_3. The only restrictions we have are that, if *tous les* is in the scope of *un*, it must be in its body, and if *un* is in the scope of *tous les*, it must be in its restriction. This follows from the constraints involving l_2 and l_4.

(14) $\delta_1 : \left\{ \begin{array}{lll} h_1 \to l_3 & h_2 \to l_2 & h_3 \to l_5 \\ h_4 \to l_1 & h_5 \to l_6 & h_6 \to l_4 \end{array} \right. \quad , \quad \delta_2 : \left\{ \begin{array}{lll} h_1 \to l_5 & h_2 \to l_2 & h_3 \to l_4 \\ h_4 \to l_1 & h_5 \to l_6 & h_6 \to l_3 \end{array} \right.$

The two disambiguations corresponding to the two scope orders are shown in (14). δ_1 corresponds to wide scope of *tous les'* and δ_2 to wide scope of *un'*. These are the only two disambiguations for (13).

3.2. PPs as arguments of NPs

The case of the PP being an argument of the noun, as in (11), is actually the simpler case of the two constructions with PPs embedded into NPs. There is no extra elementary tree for the preposition. Instead the preposition is treated as semantically void and it is part of the elementary tree of the noun that selects for the PP, in this case *détenteur*. Furthermore, this elementary tree contains a substitution node for the embedded noun, its argument. The syntactic analysis and the e-derivation structure for (11) is shown in Fig. 9.

The semantic representations for (11), shown in (15), are more or less the same as for (10), except for the one for *détenteur*. This semantic representation gives the predicate used as argument of *tous les* and, at the same time, contains the proposition that is argument of *un*.

(15)

$\boxed{\begin{array}{l} l_1 : \textit{connaître'}(x_1, x_2) \\ l_1 \leq h_1 \\ \hline \text{arg: } \langle x_1, 1\rangle, \langle x_2, 22\rangle \end{array}}$ $\boxed{\begin{array}{l} \textit{pierre'}(x) \\ \hline \text{arg: } - \end{array}}$ $\boxed{\begin{array}{l} q_1 : \lambda u[l_2 : \textit{détenteur-de'}(u, x_3)] \\ l_2 \leq h_2 \\ \hline \text{arg: } \langle x_3, 22\rangle \end{array}}$

$\boxed{\begin{array}{l} l_3 : \textit{tous les'}(y, h_3, h_4), l_4 : p_1(y) \\ s_1 \leq h_4, l_4 \leq h_3 \\ \hline \text{arg: } s_1, p_1 \end{array}}$ $\boxed{\begin{array}{l} q_2 : \textit{prix'} \\ \hline \text{arg: } - \end{array}}$ $\boxed{\begin{array}{l} l_5 : \textit{un'}(z, h_5, h_6), l_6 : p_2(z) \\ s_2 \leq h_6, l_6 \leq h_5 \\ \hline \text{arg: } s_2, p_2 \end{array}}$

For the same reasons as in (10), we get the following assignments: $x_1 \to x$, $x_2 \to y$, $x_3 \to z$, $p_1 \to q_1$ and $s_1 \to l_1$. For s_2, $s_2 \to l_2$ is the only possibility, and for p_2, $p_2 \to q_2$ is chosen, it follows not only the primary edge but it is even the only possibility because $p_2 \to q_1$ would lead to $l_2 \leq h_6$ and $l_2 \leq h_5$ which contradicts the separation of restriction and body of a quantifier. The semantic representation in (16) is obtained for (11).

(16)

l_1 : *connaître'*(x, y), *pierre'*(x), l_3 : *tous les'*(y, h_3, h_4), $l_4 : (q_1 : \lambda u[l_2 :$ *détenteur-de'*$(u, z)]])(y)$ l_5 : *un'*(z, h_5, h_6), $l_6 : (q_2 :$ *prix'*$)(z)$ $l_1 \leq h_1, l_2 \leq h_2$ $l_1 \leq h_4, l_4 \leq h_3$ $l_2 \leq h_6, l_6 \leq h_5$
arg: $-$

(16) corresponds to the semantic representation of (10), in particular the constraints for quantifier scope are the same. Consequently, as in the case of (10), the two scope orders of the quantifiers are both possible.

As we have seen, the approach proposed in this paper correctly allows quantifiers in embedded PPs to take wide scope. Furthermore, for cases of scope ambiguities, it even allows to generate appropriate underspecified representations.

4. Unbounded dependencies in embedded interrogatives

The problem this paper concentrates on are quantifiers and their analysis in TAG. However, the enriching of the derivation tree proposed above is also useful for other problems one encounters when doing semantics with TAG. One often mentioned problem are unbounded dependencies in embedded interrogatives as in (17).

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

An adequate semantics for (17) should have the following structure:

(18) *wonder'*(*mary'*, *who'*$(x$, *think'*(*peter'*, *say'*(*john'*, *like'*(*bill'*, x))))).

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

In order to obtain the relation between *think'* and *who'*, *think* must be connected either to *who* or to *like* (if the semantic representation of *like* contains a part that corresponds to its 'moved' wh-part). Fig. 10 shows the classical TAG analysis of (17), following (Kroch, 1987). The derivation tree does not contain the necessary links. The e-derivation structure however provides an additional link between *like* and *say*. Based on this structure it is possible to build an appropriate semantics for such cases.

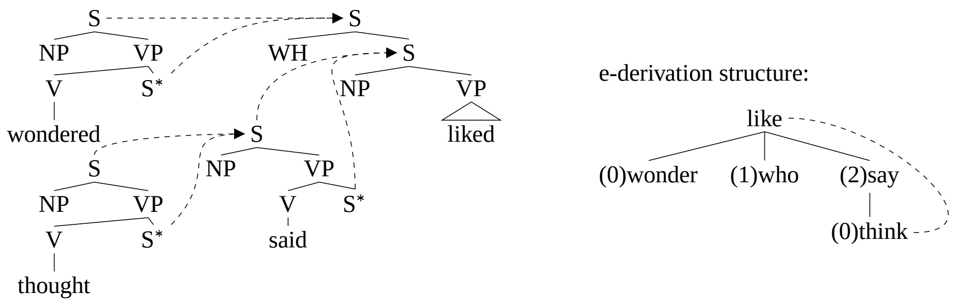


Figure 10: TAG derivation and e-derivation structure for (17)

5. Related work

An approach that also defines an additional structure related to the derivation tree in order to solve the problems one accounts with a semantics directly based on the derivation tree is (Dras, Chiang and Schuler, 2000). Dras et al. view the derivation tree as a tree derived by a meta-level TAG and the derivation trees provided by this second TAG are the structures they use for semantics. An obvious advantage of our approach is that it is less complex. Starting from the derivation tree it is easy to obtain the e-derivation structure. Furthermore, the e-derivation structure is a natural extension in the sense that it just makes things explicit that are already present in the derivation tree.

Frank and van Genabith (2001) propose to define TAG semantics based on the derived tree in order to solve the problems mentioned in this paper. However, they make use not only of the information available in the derived tree but also of information about how the elementary trees were put together, i.e., of the derivation tree. Compared to this, the advantage of the approach proposed here is that semantics is based only on the enriched derivation tree and does not need to go back and to use both, the derived tree and the derivation tree.

6. Conclusion

I have shown in this paper that, in spite of some mismatches between TAG derivation trees and dependency structures, it is possible to build a semantics in the TAG framework based on the derivation trees. The key idea is that I am enriching the derivation tree by making links between elementary trees explicit that are already present in the derivation and that can be read off the derivation tree. This enriched structure called e-derivation structure is used as basis for semantics. This approach allows to account for the semantics of quantifiers even if a syntactic analysis is assumed that treats quantifiers as noun adjuncts. I have shown that the semantics proposed here correctly allows quantifiers embedded into NPs to take wide scope. Furthermore, the e-derivation structure also allows to deal with other phenomena that are problematic for the assumption that derivation trees provide the right dependency structure to use for semantics, such as unbounded dependencies in embedded interrogatives.

References

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

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

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

Candito, Marie-Hélène 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.

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

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

Frank, Anette 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 Fracy Holloway King, editors, *Proceedings of the LFG01 Conference*, Hong Kong.

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

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

Joshi, Aravind K. 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.

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

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

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

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

Vijay-Shanker, K. and Aravind K. Joshi. 1988. Feature Structures Based Tree Adjoining Grammar. In *Proceedings of COLING*, pages 714–719, Budapest.