Defining DTG derivations to get semantic graphs
Marie-Hélène Candito & Sylvain Kahane
TALANA, Université Paris 7, 2, place Jussieu, case 7003, 75251 Paris Cedex 05
marie-helene.candito@linguist.jussieu.fr, sk@ccr.jussieu.fr

Introduction
The aim of this paper is to find a formalism of the TAG family, where the derivation controller can be interpreted as a semantic dependency graph, in the sense of Meaning-Text Theory (ZM67; M88).

In a previous paper (CK98), we study this interpretation of the derivation tree (DT) in the case of standard TAG. We prove that, in the general case, if the predicate-argument cooccurrence principle1 [\(\Rightarrow\) PACP] holds and if elementary trees correspond to a semantic unit (A91), substitution arcs can be read as semantic dependencies where the dependent is the anchor of the substituted tree, and adjunction arcs — of any type— can be read as semantic dependencies in the opposite direction.

Yet we also characterized cases where the DT shows wrong (semantic) dependencies (cf also (RVW95)). A problem may occur when, in the same sentence, clausal complementation is handled both with substitution of an embedded clause and with adjunction of a main verb.2

Further, there are well-known cases that TAG cannot handle if the PACP holds (e.g. clitic climbing in Romance (B98), Kashmiri wh-extraction (RVW95), extraction out of NP in French (A98)). Finally, in some cases, the argumental positions in a tree are not filled by the right arguments, and thus the derivation tree does not show the right semantic dependencies (pied-piping (CK98)).

(RVW95) have defined D-tree Grammars (DTG) by ruling out predicative adjunction (e.g. adjunction of bridge verbs). Thus, DTG seems a good candidate for our goal.3 In Section 1, we recall DTG operations and study the case of relative clause interacting with a bridge verb, which has proved to be correctly handled by TAG, as far as semantic dependencies are concerned (CK98). This leads us to propose an extension of DTG, called GAG for Graph-driven Adjunction Grammar, whose derivation controllers are graphs (Section 2). Finally, in Section 3, we develop an original analysis of wh-words in GAG.4

1. Generalized substitution and generalized adjunction
DTG (RVW95) handles both clausal and nominal complementation with the same operation, a generalized substitution, called subsertion, and thus avoids the use of predicative adjunction. In order to cover the long-distance dependency data (including cases not handled in TAG), this operation allows pieces of the substituted element to come in between elements of the tree receiving substitution.

---

1 A tree anchored by a predicate must contain positions for all and only its arguments.
2 For a sentence such as That Paul wanted to stay surprised Mary, the DT shows the wrong dependencies if the tree for surprise has a substitution node for its subject, and the one for want has a foot node for its embedded clause (CK98). Another problem occurs with a raising verb that serves as semantic argument to a bridge verb as in Paul claims Mary seems to adore hotdogs (adapted from (RVW95)). To get the correct semantic dependencies, the trees for claims and seems should combine together (either via substitution of seems or adjunction of claims) but this is impossible in TAG since seems is represented by a VP-rooted tree.
3 In (RVW95), one motivation was to get (deep) syntactic dependencies. Though in most cases semantic and deep syntactic dependency structures induce the same non-oriented graph, we will study a case of mismatch in Section 1 and 3.
4 We are thankful to Owen Rambow and David Weir for valuable discussions about this work.
substituted node or placed above the root node. Fig. 1 shows an example of subsertion.  

Now, ruling out predicative adjunctions implies to reconsider cases that were correctly handled by TAG as far as semantic dependencies are concerned. For example, in order to handle extraction out of a modifier (e.g. preposition stranding) and « extraction of a modifier », we define a parallel generalization of the adjunction operation. We will thus refer to generalized substitution and generalized adjunction. Fig. 2 shows the generalized adjunction of in [this bed] for the sentence:  

(1) In this bed, I think I have slept twice.

To get the semantic dependency between in and slept, we want βin to adjoin in αslept, still allowing a piece of the modifier (here the whole modifier) to be inserted higher.

![Figure 2: Generalized adjunction](image)

We thus have to specify what happens in the case of node unification during substitution or adjunction. In case of substitution, a rope unifies with a substitution originates. The resulting DG ψ' is the DG ψ plus a to book (creating the syntactic attachment) and wrote subserted into thinks with the relative pronoun being inserted into the right place and receiving co-reference with books through features (thus creating the semantic attachment).

2. GAG: a multi-rope DTG

GAG is an extension of DTG that uses the same elementary structures, namely d-trees. But in GAG, some nodes of an elementary d-tree must be marked as being ropes (underlined in the figures). Only roots of components can be ropes. Any component containing a foot node has a root which is a rope. A component without foot node is substitutable if and only if its root is a rope. In (RVW95), all the components of an elementary d-tree are considered substitutable, namely each component’s root is a rope, but a d-tree can be subserted only once, namely all ropes are mutually exclusive. In our extension, d-trees with n mutually exclusive ropes are expanded in n d-trees with a single rope. Further, a d-tree may have several ropes which are not mutually exclusive, that is, that can each be combined with a separate d-tree. To sum up, GAG is a multi-rope DTG.

From the linguistic point of view, we foresee the use of one-rope and two-rope d-trees only. Examples of two-rope d-trees will be given in Section 3.

Let us now define the derivation graph (DG), which is a structure that partially encodes a GAG derivation (and that we will interpret as a semantic graph). If a two-rope d-tree substitutes in a one-rope d-tree, we obtain a two rope derived d-tree and nothing in the DG tells us from which elementary d-tree each rope comes from. Thus in GAG, the original elementary d-tree for each node of a derived d-tree is memorized.

We thus have to specify what happens in the case of node unification during substitution or adjunction. In case of substitution, a rope unifies with a substitution site. We then consider that the resulting node comes from the elementary d-tree that is substituted. In case of generalized adjunction, the node receiving adjunction is replaced by a component of the adjoined tree. In the derived tree, we consider that the root of that adjoined component belongs to the tree receiving adjunction.

The DG can now be defined as follows: let γ be an elementary d-tree. Let ϕ be a derived tree, and ψ the corresponding derivation graph (DG). If ϕ substitutes (resp. adjoins) in γ, one of its ropes is used up. Let α be the name of the d-tree from which this rope originates. The resulting DG ψ' is the DG ψ plus a
substitution (resp. adjunction) arc between \( \alpha \) and \( \gamma \) (\( \gamma \) being the mother node).\(^{10}\) Consequently to this definition, a d-tree has as many mother nodes in the final DG as it has used ropes.

As in DTG, the derivation succeeds if the d-edges of the derived tree can be collapsed (forgetting the fact that some nodes can be rope nodes). From the computational point of view it can be noted that the ropes of a multi-rope d-tree can combine in whatever order with other d-trees.

3. Taking advantage of GAG to analyse extraction

As we said, the main motivation for GAG is to have a formalism inspired by TAG whose derivation controllers induces semantic dependency graphs. In order to achieve that, we have relaxed the constraint that these controllers be trees.

As linguistic constraints for elementary structures, in addition to the PACP, we type the argumental positions as foot nodes and substitution nodes on purely linguistic grounds.\(^ {11}\) Generalized substitution is used for elements that are subcategorized and to which a thematic role is assigned, while generalized adjunction is used for modifiers.

In the following, we concentrate on examples of GAG analysis involving wh-words. Consider:

(3a) Children books Peter thinks Mary wrote.
(3b) I bought the books which Peter thinks Mary wrote.
(3c) I wonder which books Peter thinks Mary wrote.
(3d) Which books does Peter think Mary wrote?

In these four examples we have a clause of the form Peter thinks Mary wrote [book]. The distribution of this clause depends on the extracted element: for example, in (3b) that clause is an NP modifier because of the relative wh-word which and in (3c) that clause can be the syntactic argument of wonder because of the interrogative wh-word which. The (T59) analysis of relative and (indirect) interrogative clauses is that the wh-word plays two roles: on one hand, it fills a position in the clause as pronoun and on the other hand it controls the distribution of the clause and is thus its syntactic head.

We claim that it is possible (though not mandatory) to have an analysis where the particular distribution of wh-clauses is completely assumed by the wh-word. To do this, we represent wh-words with two-rope elementary d-trees: the first rope will be linked to the main clause and the second one will be linked to the phrase showing extraction. So for a relative wh-word, the first rope is a foot node which adjoins on the antecedent and the second rope substitutes or adjoins in the phrase showing extraction, depending on the complement/modifier nature of the extracted element.

For an interrogative wh-word, the first rope substitutes in the verb which subcategorizes for the interrogative element. We give example of relative clauses only:

(4a) I know the books which Mary wrote.
(4b) I know the bed in which Peter slept.
(4c) I know the books whose authors are famous.
(4d) I know the man whose car Peter borrowed.
(4e) I know the place where Peter was born.

Fig. 3 shows the two-rope d-trees for the wh-words involved (the ropes are underlined).

\[\text{Fig. 3: some two-rope elementary trees} \]
\[\text{(for relative wh-words)}\]

The analysis for (3b), (4a) and (4b) use the same d-tree for which, \( \beta \alpha \), which substitutes respectively in the d-trees for wrote and in (Fig. 4).

\[\text{Fig. 4: Substitution of a two-rope d-tree} \]

\[\text{(with extraposition of the object)}\]

10 It can be noted that because we remember the origin of each node of a derived tree, a derivation need not be bottom-up.

11 This is possible because we allow the derivation controller to be a graph and use the generalized substitution.
This last operation arises typically for some relative pronouns. In the βα d-trees of Fig. 3, the foot node does not represent a semantic argument of the anchor, but a duplication of the anchor itself (the antecedent in syntax). Thus the correspondent adjunction arc in the derivation controller (eg. the adjunction arc in Fig. 5) has to be collapsed in order to get the semantic graph (Fig. 6).

\[ \text{In (4c), whose is an argument of authors, thus } \beta \alpha \text{ whose substitutes in the authors tree. In (4d), we consider that whose is a lexicalization of the two-place semanteme 'own'. Its d-tree } \beta \beta \text{ whose}\textsuperscript{12} \text{ adjoins twice, on the trees for both its arguments (here lexicalized by man and car). Fig 7 shows the DG for (4d).} \]

\[
\begin{array}{c}
\alpha \text{ know} \\
\alpha \text{ I} \quad \alpha \text{ books} \\
\beta \alpha \text{ whose} \quad \alpha \text{ Peter} \\
\end{array}
\]

**Figure 5: GAG derivation graph**

*I know the books which Peter thinks Mary wrote*

To get the semantic graph, the two adjunction arcs of \( \beta \beta \text{ whose} \) are interpreted as semantic dependencies from the adjoined tree to the tree receiving adjunction (Fig. 8). Similarly, \( \beta \beta \) where corresponds to a semanteme ‘location’ (= ‘is located in’) with two arguments.

\[
\begin{array}{c}
\alpha \text{ know} \\
\alpha \text{ I} \quad \alpha \text{ car} \\
\beta \beta \text{ whose} \quad \alpha \text{ Peter} \\
\end{array}
\]

**Figure 7: GAG derivation graph**

*I know the man whose car Peter borrowed*

In the \( \beta \alpha \) d-trees of Fig. 3, the foot node does not represent a semantic argument of the anchor, but a duplication of the anchor itself (the antecedent in syntax). Thus the correspondent adjunction arc in the derivation controller (eg. the adjunction arc in Fig. 5) has to be collapsed in order to get the semantic graph (Fig. 6).

\[
\begin{array}{c}
\alpha \text{ know} \\
\alpha \text{ I} \quad \alpha \text{ books} \\
\beta \alpha \text{ whose} \quad \alpha \text{ Peter} \\
\end{array}
\]

**Figure 6: MTT semantic graph**

*I know the books which Peter thinks Mary wrote*

\[
\begin{array}{c}
'\text{know}' \\
'\text{write}' \\
'I' \\
'\text{book}' \\
'Mary' \\
\end{array}
\]

**Figure 8: MTT semantic graph**

*I know the man whose car Peter borrowed*

DTG). They are needed for instance to block extraction in the case of non-bridge verbs or to express constraints on double extractions and topicalization.

**Conclusion**

Building on DTG and TAG, we have defined a formalism, GAG, where the derivation controller can be seen as a semantic dependency graph, with the reading defined in (CK98). This allows us to propose an analysis in which the distribution of clauses containing wh-words is totally controlled by the d-trees associated with the wh-words themselves. Thus topicalization, relativization, (direct or indirect) interrogation and cleft clauses can be handled with the same elementary d-trees for verbs. Computational properties of GAG need a further study.

**References**


