G-TAG: A Lexicalized Formalism for Text Generation inspired by Tree Adjoining Grammar

LAURENCE DANLOS
TALANA, Université Paris 7 & LORIA

Introduction

G-TAG is a formalism to generate texts from their conceptual representation. It is inspired from the framework of lexicalized tree adjoining grammar (noted as TAG). It is designed to use the syntactic and lexical information of a TAG grammar. We extended this TAG grammar to handle multi-sentential texts and not only isolated sentences. We also added a conceptual-semantic interface. This conceptual-semantic interface is lexicalized, as it is the case for the semantic-syntax interface, i.e. the TAG grammar. Therefore, G-TAG is thus a lexicalized formalism for text generation. This innovative approach shows that lexicalization can also be used for texts and not only for sentences as is the case for most other generation systems.

G-TAG transforms a conceptual representation into a text. This representation should be language independent and enriched with pragmatic information. It can come from two sources:

- a *What to say* module which selects the information to convey from an intended communicative act and which establishes conceptual links between them;

- a user providing the information by answering questions through a cascading menu, as in DRAFTER (Paris et al. 1995).

The structure of the conceptual input is not committed to any particular linguistic realization. G-TAG thus deals with the *How to say it?* issue, understood as covering all and only linguistic decisions: segmentation into sentences, ordering of sentences, choice of connectives, choice of lexical items and syntactic constructions within a sentence, etc.

As shown in Figure 1, the basic idea underlying G-TAG is to use a kind of derivation tree, called a "g-derivation tree", as a semantic level intermediary between a conceptual representation and a text. From the parsing point of view, the derivation tree in TAG is seen as the "history" of the derivation, but also as a linguistic representation, closer to semantics, which can be the basis for a further analysis. A g-derivation tree in G-TAG is closer to semantics than a derivation tree in TAG: it is a semantic dependency tree (annotated with syntactic information).

A g-derivation tree specifies a unique "g-derived tree", in the same way as a derivation tree specifies a unique derived tree. A g-derived tree is a syntactic tree annotated with morphological information. From a g-derived tree, a post-processing module computes a text by performing morphological computations and formatting operations. This module can also produce *surface variants* of the text specified by the g-derived tree.

The conceptual-semantic interface is made up of concepts each associated with a lexical data base. A lexical data base for a given concept records the lexemes lexicalizing it with their argument structure, and the mappings between the conceptual and semantic arguments (semantic arguments are pseudo thematic roles, i.e. arg1, arg2, arg3). The conceptual-semantic interface is thus similar to the semantic-syntactic interface based on a TAG grammar which is made up of lexical data bases. A data base for a given lexical entry records the syntactic structures realizing it with their syntactic arguments. I assume moreover that the TAG grammar records the mappings between the semantic and syntactic arguments.

With such a lexicalized conceptual-semantic interface, the process for computing a g-derivation tree relies upon a single type of operation: lexicalization, i.e. the choice of a lexeme and its syntactic realization to convey an instance of a concept. Since all the main decisions are made during this process, G-TAG can be considered as a "lexicalized formalism for text generation". The architecture of G-TAG and its data bases are outlined in Figure 1.

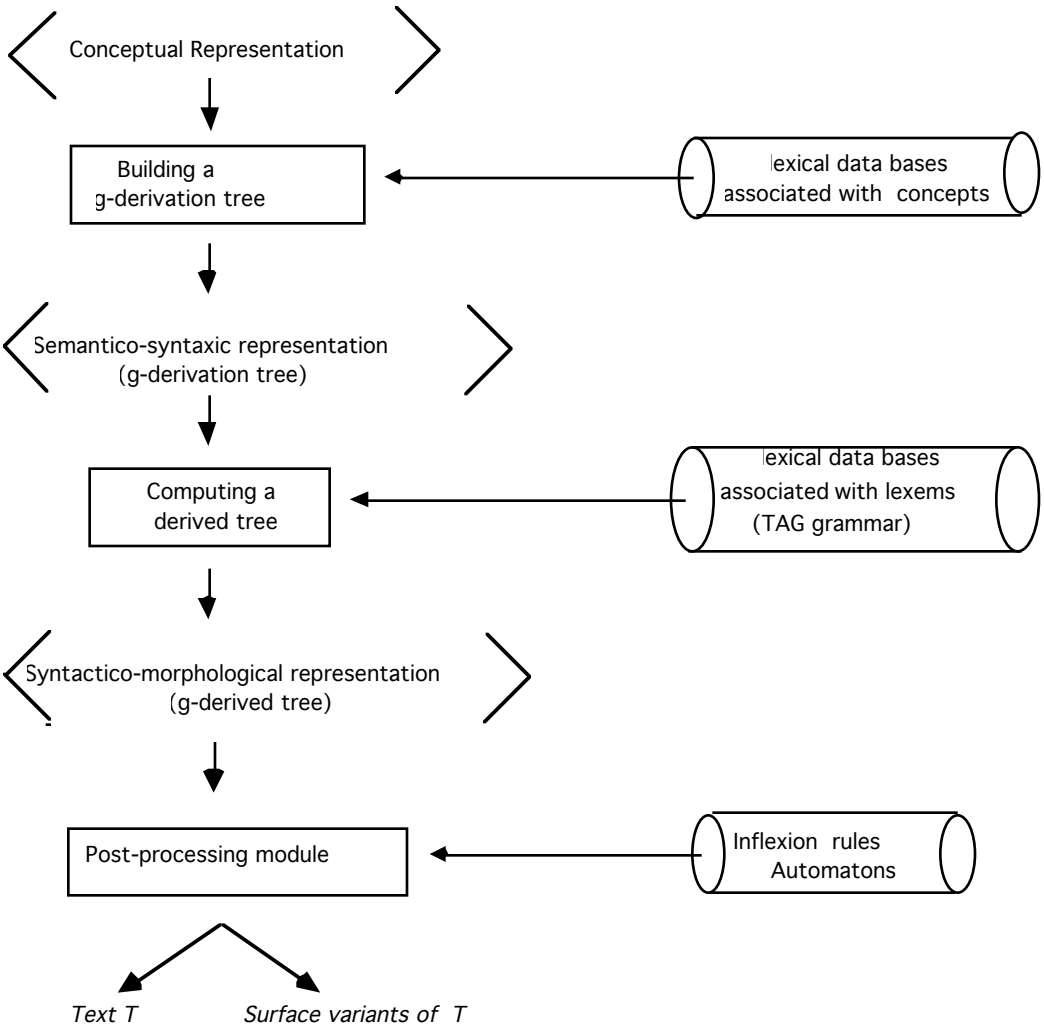


Figure 1. Architecture and data bases of G-TAG

This paper is organized as follows:

- Section 1 describes briefly the conceptual level, input to G-TAG;
- Section 2 presents the semantico-syntactic level (i.e. g-derivation trees both for sentences and texts), the syntactico-morphological level (g-derived trees) and the post-processing module;
- Section 3 presents the lexical data bases that constitute the conceptual-semantic interface;
- Section 4 describes how to compute a g-derivation tree;

- Section 5 compares G-TAG with other related work;
- Section 6 presents the implementations and applications of G-TAG and ends on future research.

In all these sections, the same reference example will be used: the different levels of representation to generate the text in (1) will be presented.

(1) Jean a passé l'aspirateur pour être récompensé par Marie. Ensuite, il a fait la sieste pendant deux heures.

(John vacuumed in order to be rewarded by Mary. Afterwards, he took a nap for two hours.)

1 Conceptual level

The domain model is a hierarchically organized collection of concepts. The universe is dichotomized between THING and RELATION (names of concepts are written in upper cases):
- THING comprises "things" such as HUMAN, CONCRETE, etc.;
- RELATION is divided into 1ST-ORDER-RELATION (i.e. mainly relations between things, e.g. REWARDING, VACUUMING, NAPPING) and 2ND-ORDER-RELATION (i.e. relations between relations, e.g. SUCCESSION, GOAL). 2ND-ORDER-RELATIONs correspond roughly to "discourse relations", while I will explain in Section 5 why I want to avoid the term "discourse relation".

A concept is associated with a structure, namely a set of arguments which are also written in upper cases (RWDER and RWDEE for RWDIND[1]). The value of each argument is conceptually restricted (the RWDER of RWDING must refer to an HUMAN). A 2ND-ORDER-RELATION has two arguments[2] each of which have to refer to a RELATION. I use the following representations for RWDING and SUCCESSION.

```
RWDIND < 1ST-ORDER-RELATION [RWDER => HUMAN, RWDEE => HUMAN]
SUCCESSION < 2ND-ORDER-RELATION [1ST-EVENT => RELATION, 2ND-EVENT => RELATION]
```

A token identifies an instance of a concept and it specifies the values of the arguments which are instances of concepts or constants. Figure 2 gives the conceptual representation of our reference example (1), without pragmatic nor temporal information.

```
E0 =: SUCCESSION [1st-EVENT => E1, 2ND-EVENT => E2]
E1 =: GOAL [action => E11, PURPOSE => E12]
E2 =: NAPPING [NAPPER => H1], with [DURATION => D1]3
E11 =: VACUUMING [VACUUMER => H1]
```

---

[1] RWDING (= REWARDING) could include a third argument, i.e. the reward as *baiser* in (i), but I will leave this issue aside here.
(i) Marie a récompensé Jean d'un baiser. (Mary rewarded John with a kiss.)

[2] An n-ary relation, e.g. SUCCESSION, is turned into a cascade of binary relations in a classic way.

[3] This notation means that DURATION is not an argument of NAPPING but is a modifier.

```
E12 =: RWDING [RWDER => H2, RWDEE => H1]
H1 =: HUMAN [NAME => "Jean", Sex => masc]
H2 =: HUMAN [NAME => "Marie", Sex => fem]
D1 =: DURATION [UNITY => hour, QUANTITY => 2]
```

Figure 2: Conceptual representation of (1)

G-TAG takes as input an instance of RELATION (most often an instance of 2ND-ORDER-RELATION) enriched with pragmatic information. It produces as output a text of one or more sentences.

2 G-derivation trees, g-derived trees and post-processing module

We will first summarize the discussions on how and to what extent a TAG derivation tree can be considered as a semantic dependency tree. Afterwards, we will present how a g-derivation tree and a derivation tree differ. Next, we will show how to extend a TAG grammar to handle texts and not only isolated sentences. Finally, we will show how to compute a text from a g-derivation tree.

2.1 TAG derivation trees / semantic dependency trees

I assume that the TAG grammar embedded in G-TAG is made up of elementary trees sharing the following properties: an elementary tree corresponds to exactly one semantic unit[4] and respects the predicate argument co-occurrence principle (predicates anchor trees with positions for all and only their semantic arguments). With these properties, a derivation tree in the sense of (Shieber & Schabes 1994) can be considered as a linguistic representation close to semantics.

Yet, even with these properties, it has been argued that there exist cases where a derivation tree shows incorrect dependencies either at the semantic or deep-syntactic level. These incorrect dependencies arise mainly because bridge verbs are generally represented as auxiliary trees in TAG in order to account for unbounded dependencies. However, unbounded dependencies almost never occur in technical texts. Since technical texts are the only kind of texts for which automatic generation can be contemplated, this phenomena giving rise to derivation trees with incorrect dependencies can be put aside. G-TAG thus handles only (g)-derivation trees with correct semantic dependencies. Moreover, the notion of a g-derivation tree used in G-TAG is closer to semantics than the one of a derivation tree in TAG, as explained below.

---

[4] An elementary tree can thus have several lexical anchors, either because some are semantically empty (functional words), or because the several anchors form an idiom, whose semantic is not compositional.

2.2 G-derivation trees

Let us first present lexical entries. In G-TAG, a lexical entry e (a lexical entry is underscored) corresponds to a lemma and points to a set of elementary trees via its family as in TAG: e -> {e0, e1, …, en}. e0 is considered as the canonical representative, the other elementary trees ej (with j > 0) being identified by one or several "T-features", noted as [Tk]. The values of Tk are + and -. [Tk] is equivalent to [Tk = +]. For example, in the family of transitive verbs (with two arguments arg1 and arg2):

- the elementary tree for the construction in the active is the canonical representative,
- the tree for the construction in the passive is identified with the T-feature [T-passive],
- the tree for the construction in the absolute is identified with [T-without-arg2],
- the tree for the construction in the passive without agent is identified with [T-passive] and [T-without-arg1].

In the French applications of G-TAG (Section 6), the elementary trees identified by T-feature(s) have been automatically generated out of the hierarchical representation of (Candito 1996, 1998).

Let us now present g-derivation trees. The nodes in a g-derivation tree are names of lexical entries. They can receive two kinds of features: T-features to select one of the elementary trees pointed to by the lexical entry while computing the g-derived tree (Section 2.4), and morphological features to compute the inflected forms in the post-processing module (Section 2.4). Like in a derivation tree, there are two kinds of arcs in a g-derivation tree: substitution arcs (which are not ordered and represented by simple dashes) and adjunction arcs (which are ordered for adjunctions at the same address, see (Shieber & Schabes 1994), and represented by thick dashes). The addresses for substitution arcs are thematic roles, which stay invariant regardless of the features that are added to the nodes. Let us say again that the TAG grammar is supposed to record (one way or another) the mappings between the thematic roles and the syntactic arguments (in this paper, these mappings are recorded in the elementary trees[5]).

The g-derivation trees for (2a), (2b) and (2c) are respectively shown in (3a), (3b) and (3c) (il is the French referential subject pronoun which is realized as *il, elle, ils* or *elles*).

(2) a Marie a récompensé Jean . (Mary rewarded John.)

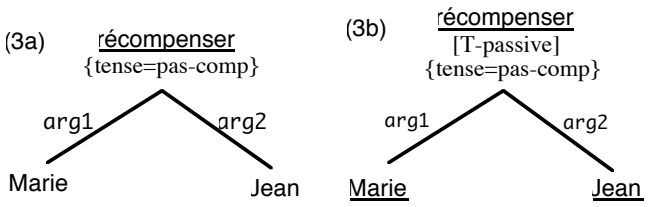
b Jean a été récompensé par Marie. (John was rewarded by Mary.)[6]

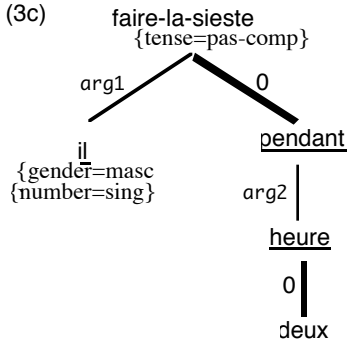
---

[5] However, they can also be recorded in the lexical entries if the TAG grammar is written in such a way that the syntactic arguments are semantic invariants (a choice made in the French TAG grammar described in (Abeillé 1991, Abeillé & Candito this volume)).

[6] The g-derivation tree for the infinitival clause *être récompensé par Marie (be awarded by Mary)* will be shown in Section 4.

b Il a fait la sieste pendant deux heures. (He took a nap for two hours.)





In comparison with the "classic" derivation trees used in TAG, we can highlight the following differences:

- naming of nodes: tree sketch name + inflected anchor in TAG versus name of a lexical entry + syntactic and morphological features in G-TAG,
- addresses for substitution arcs: Gorn numeric addresses in TAG versus thematic roles in G-TAG,
- auxiliary verbs: in analysis, they are typically handled by adjunction and so appear as nodes in derivation trees, while temporal and aspectual information is recorded as features in g-derivation trees.

There exists another crucial difference between a g-derivation tree and a derivation tree: a g-derivation tree corresponds to a set of surface variants (with respect to word order, for example), while a derivation tree represents a unique surface form. This will be explained in Section 2.4. Beforehand, let us present how to extend a TAG grammar to handle texts consisting of several sentences[7].

---

[7] Recently, (Webber & Joshi 1998) have proposed also a TAG grammar for text. Their approach will be compared with mine in Section 6.

2.3 TAG grammar for texts

There are two ways to link two sentences to build a text: either with an adverbial phrase as in (1) or (1a) (the position of the adverbial phrase within the second sentence will be discussed in the next section), or without any adverbial as in (4a) and (4b).

(1) Jean a passé l'aspirateur … . Ensuite, il a fait la sieste pendant deux heures.

(1a) Jean a poussé Marie. Donc, elle est tombée. (John pushed Mary. Therefore, she fell.)

(4) a Jean a poussé Marie. Elle est tombée. (John pushed Mary. She fell.)

b Marie est tombée. Jean l'a poussée. (Mary fell. John pushed her.)

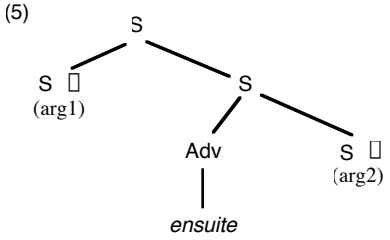
Let us first examine adverbials such as *ensuite (afterwards)* or *donc (therefore)*. At the semantic level, they are predicates with two sentential arguments (Danlos 1998). One evidence for this claim is that a sentence (clause) which comprises a discourse cue (e.g. *Ensuite, il a fait la sieste*) cannot be understood when the left context is empty. Moreover, the two arguments of a discourse cue have the same importance: the claim that the second sentence is the "satellite" (modifier) of the first one which is the "nucleus" (modifee) (in RST terms (Mann & Thomson 1988)) seems unjustified. As a proof, *S1. Ensuite S2.* is paraphrased by *D'abord S1. Ensuite S2.* (*First S1. Afterwards S2.*) and *D'abord S1.* cannot be understood when the right context is empty. Therefore, in G-TAG, the canonical elementary tree whose anchor is *ensuite* is an initial tree with two sentential arguments, (5)[8]. The same kind of initial tree is used for every discourse cue (whatever its rhetorical versus descriptive nature). It corresponds to a unique semantic unit and it respects the predicate argument co-occurrence principle. However, it is not the kind of tree used in TAG: at the syntactic level, a discourse cue (adverbial phrase) anchors an auxiliary tree with one sentential (or verbal) argument. This discrepancy between the argumentarity of discourse cues at the semantic and syntactic level, which is also outlined in Meaning to Text Theory (Iordanskaja & Mel'cuk 1999), means that the transition from the syntactic sentential level to the semantic textual level cannot follow a totally compositional path.

With (5) as elementary tree for *ensuite*, the g-derivation tree underlying (1) is (6) in which `GDT1` and `GDT2` represent respectively the g-derivation trees for the first and second sentences.

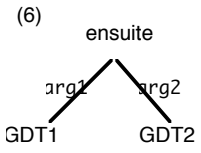
---

[8] This tree could have two lexical anchors: *d'abord* in the first sentence marked as optional, and *ensuite* in the second sentence. For *Adv1 S1. Adv2 S2.* texts (e.g. *D'une part S1. D'autre part S2.* (*On the one hand S1. On the other hand S2.*)) elementary trees with two lexical anchors (*adv1* and *adv2*) are also needed.

(5)



(6)



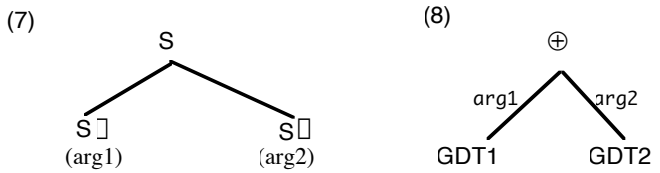
As shown in (5), a text is represented with the category S, which represents either a text or a sentence. This allows to build a text consisting of more than two sentences. However, a text and a sentence are distinguished through a "form feature" which will be explained in Section 3.

Let us now examine *S1.S2.* texts such as (4) without a connective to link the two sentences. In most of the cases, a *S1. S2.* text can be seen as the result of an "adverbial ellipsis" from a *S1. Adv S2.* text, e.g. (4a) is an elliptical form of (1a)[9]. This adverbial ellipsis does not follow from the ellipsis of an element occurring in the left context, as it is the case in VP ellipsis. Let us say that a *S1. S2.* text is a "pure elliptical form". Such a pure elliptical form requires extra-linguistic knowledge to be understood like the "Push Causal Law" (Lascarides & Asher 1991) for (4)[10]. The question arises on how to represent pure elliptical forms. The only possible way seems to be by means of a special predicate, noted as ⊕, which refers to an elementary (initial) tree similar to that in (5) but without a lexical head, (7). In TAG, it is postulated that each elementary tree must be anchored by a (non empty) lexical head and that the treatment of elliptical forms such as VP ellipsis should not make use of elementary trees without a lexical head. However, for a pure elliptical form, one is driven to postulate an elementary tree without a lexical head. The g-derivation tree for a *S1.S2.* text is therefore (8), where ⊕ points to the elementary tree without a lexical head given in (7), and `GDT1` and `GDT2` represent respectively S1 and S2. The similarity between (8) for a *S1. S2.* text and (6) for a *S1. Adv S2.* text is satisfactory: it reflects the analysis of a *S1.S2.* text as an elliptical form of a *S1. Adv S2.* text.

---

[9] However, some *S1. S2.* texts expressing an elaboration (e.g. *Ted bought a painting. It was painted by K. Beurrier.*) are better seen as the result of the ellipsis of the coordination conjunction *and (Ted bought a painting and it/this painting was painted by K. Beurrier.).* A variant of this analysis by ellipsis of *S1.S2.* texts is proposed in (Harris 1982): the period between *S1* and *S2* is considered as a "degenerated" discourse cue.

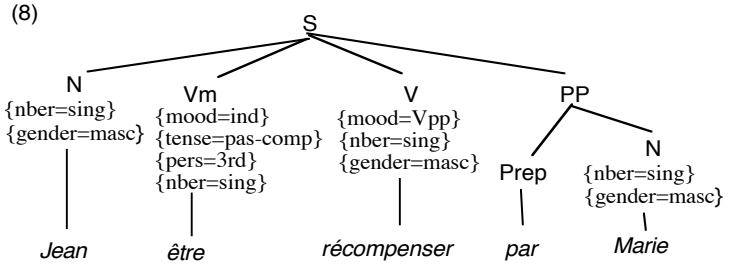
[10] The use of these elliptical forms depends on the target language. For example, in Arabic or Korean, the equivalent of (4a) is excluded: there exists only the equivalent of (1a) with a connective to link the two sentences.



2.4 Computing a text from a g-derivation tree

A g-derivation tree specifies a unique g-derived tree, in the same way as a derivation tree specifies a unique derived tree. In a g-derived tree, the leaves are lemmas, their father node bearing morphological features. These features come either from the conceptual level if they are meaningful (e.g. number for an N) or from equations in the tree sketches (e.g. number for a V). The g-derived tree computed from (3a) is shown in (9).

(9)



A post-processing module linearizes a g-derived tree: it computes the inflected forms of the leaves, concatenates and formats them. The linearization of (9) yields naturally (2a) (i.e. *Jean a été récompensé par Marie*.).

However, the post-processing module performs more operations than the ones given before: it may synthesize surface variants of the text produced by linearization of a g-derived tree. Consider again the predicate *ensuite (afterwards)*. First, at the lexical level, it seems that *ensuite* and *puis (next)* are pure variants: there seems to be no pragmatic, conceptual, semantic or syntactic criterion which would allow a generation system to choose between (1) and (1')[11].

(1) Jean a passé l'aspirateur … . **Ensuite**, il a fait une sieste pendant deux heures.

(1') Jean a passé l'aspirateur … . **Puis**, il a fait une sieste pendant deux heures.

Therefore, only the g-derivation tree of (1) is computed from the conceptual representation `E0` given in Section 1 (Section 4 will explain how). However, (1') can be produced by the post-processing module. This module can either

[11] The only possible criterion to distinguish *ensuite* from *puis* (*afterwards* from *next*) may be a register question.

randomly activate the rule in (10a) or contextually activate the rule in (10b) or (10c).

(10)a *ensuite --> puis*

b *ensuite … ensuite --> ensuite … puis*

c *ensuite … ensuite… ensuite --> ensuite … puis… finalement*

Such lexical operations performed in the post-processing module simplify the lexical choice module (see Section 4), while offering some lexical variations.

Secondly, at the word order level, *ensuite* can be either at the beginning of its second sentential argument, (1), or within it, (1").

(1) Jean a passé l'aspirateur … . **Ensuite**, il a fait une sieste pendant deux heures.

(1") Jean a passé l'aspirateur … . Il a **ensuite** fait une sieste pendant deux heures.

Again, no criterion would allow a generation system to choose between (1) and (1"). Therefore, the generation process produces only (1) which is considered as the canonical form. (1") is produced randomly by the post-processing module when it activates the (simplified) rule given in (11).

(11) *ensuite NP (ne)* V_{aux} *(pas) --> NP (ne)* V_{aux} *(pas) ensuite*

More generally, the following methodological principle is applied in the case of several surface variants of a text: the canonical variant is produced by the generation process , the other ones are generated by the post-processing module.

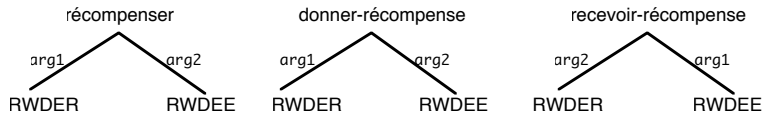
To sum up, a g-derivation tree (or the g-derived tree it specifies) corresponds to a set of surface variants. One is considered as canonical and produced by linearization of the g-derived tree, the other ones are produced from the canonical one in the post-processing module. This approach seems sound from the generation perspective (see the methodological principle above) and it avoids the difficulties encountered in TAG with word order variants. For example, analyzing or generating (1") in which *ensuite* occurs within the VP requires a formalism more powerful than TAG (e.g. a D-Tree grammar) if one wants to maintain that *ensuite* has two sentential arguments, see (Rambow et al. 1995) or (Nicolov et al. 1997).

3 Conceptual-semantic interface

A concept is lexicalized in a target language as one or several lexemes. For example, `RWDING` is lexicalized in French as *récompenser (reward), donner une récompense (give a reward)* or *recevoir une récompense (receive a reward).* The mappings between the arguments of a concept and the arguments of a lexeme lexicalizing it have to be recorded. For example, the `RWDER` of `RWDING` corresponds to arg1 of *récompenser* and to arg2 of *recevoir une récompense*.

In G-TAG, these data are recorded in lexical data bases, noted as LBs, made up of "underspecified g-derivation trees". Let us give an example. The French LB associated with RWDING, noted as LB(RWDING), comprises the three underspecified g-derivation trees given in (15).

(15)



An underspecified g-derivation tree differs from a g-derivation tree to the extent that it comprises two kinds of nodes: constant and variable nodes. A constant node is the name of a lexical entry, e.g. récompenser. A variable node is a conceptual argument, e.g. RWDER. The variable nodes are specified during the generation process: they are first instantiated (e.g. in E12, instance of RWDING, RWDER is instantiated as H2); next their instantiated values are replaced by g-derivation trees. The full process will be explained in the next section.

A lexical data base made up of underspecified g-derivation trees is associated with any kind of concept, be it a sub-type of 2ND-ORDER-RELATION, 1ST-ORDER-RELATION or THING. Let us illustrate an LB for a 2ND-ORDER-RELATION, i.e. SUCCESSION < 2ND-ORDER-RELATION[1ST-EVENT => RELATION, 2ND-EVENT => RELATION]. Different lexicalizations are illustrated in (16)[12].

(16) a Jean a passé l'aspirateur. Ensuite, il a fait une sieste. (John vacuumed . Afterwards, he took a nap.)

b Jean a fait une sieste. Auparavant, il avait passé l'aspirateur. (John took a nap. Beforehand, he had vacuumed.)

c Jean a passé l'aspirateur avant de faire une sieste. (John vacuumed before taking a nap.)

d Jean a fait une sieste après avoir passé l'aspirateur. (John took a nap after vacuuming.)

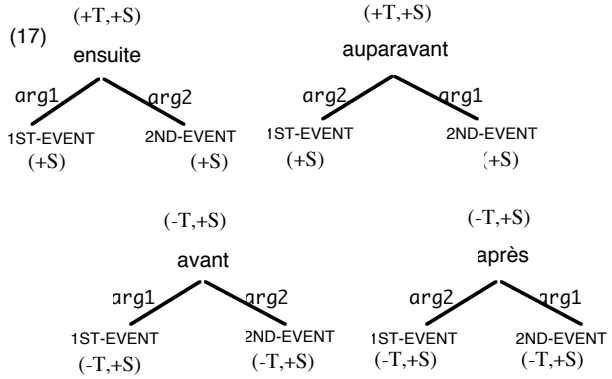
The adverbials *ensuite (afterwards)* and *auparavant (beforehand)* are used to build a text, while the subordinating conjunctions *avant (before)* and *après (after)* are used to build a sentence. These data must be recorded, for example to avoid incorrect embeddings such as embedding a text in a matrix clause. The categories of the arguments of these connectives must also be recorded so as to avoid incorrect embeddings, such as embedding a text in a subordinate clause. For this purpose, a "form" feature is added to an underspecified g-derivation as a whole and to each variable node. The form

[12] Other lexicalizations, e.g. an NP whose head is *succession* (Danlos 1998), are omitted here for the sake of simplicity.

feature (+T, +S) is used for a text, (-T, +S) for a sentence, (+S) for a text or a sentence, (-T, -S) for an NP, (-T) for a sentence or an NP. These features are illustrated in LB(SUCCESSION) shown in (17). Two points need to be emphasized:

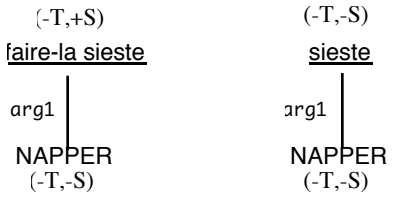
- an underspecified g-derivation tree whose constant node is a subordinating conjunction looks like a classic semantic dependency tree (a conjunction is a predicate with two sentential arguments) even if a conjunction anchors an auxiliary tree in the TAG grammar. This characteristic of G-TAG will be explained in Section 6.
- 1ST-EVENT corresponds to arg1 of *ensuite* and to arg2 of *auparavant*. The right ordering of sentences is thus obtained, since, in the elementary trees anchored by adverbial connectives (e.g. (5) or (13) for *ensuite*), arg1 corresponds to the first sentence and arg2 to the second one.

(17)



The form features are also used in the LBs associated with concepts which are subtypes of 1ST-ORDER-RELATION. Consider NAPPING. It can be lexicalized in French either with the verbal predicate *faire la sieste (take a nap)*, or as the nominal predicate *sieste (nap)*. So LB(NAPPING) is made of the two underspecified g-derivation trees in (21), where the left tree is marked as building a sentence - form feature (-T, +S), the right one as forming an NP - form feature (-T, -S).

(21)



Finally, let us give an example of a LB associated with a concept which is a subtype of THING. LB(BIKE) is made of the two underspecified g-derivation

trees bicyclette and vélo, each one with a nominal constant node and without variable nodes.

To sum up, the conceptual-semantic interface is a set of LBs. LBs are made of underspecified g-derivation trees and are designed along the same principles, whatever the type of the concept involved (subtype of 2ND-ORDER-RELATION, 1ST-ORDER-RELATION or THING) and whatever the maximal projection (text, sentence or NP) of constant nodes[13]. This makes the process of computing a g-derivation tree homogeneous. It relies upon a unique operation: choice of an underspecified g-derivation tree in a LB, as explained in the next section.

4 Building a g-derivation tree

The basic principle used to build a g-derivation tree from a conceptual representation identified by a token is recursivity. The first step consists of lexicalizing the concept C0 of which the token is an instance, that is select an underspecified g-derivation tree in LB(C0)[14]. For this selection, the underspecified g-derivation trees are equipped with tests and these tests may result in adding one or several T-features to the constant nodes, as illustrated below. When an underspecified g-derivation tree has been selected, it is instantiated, i.e. the variable nodes are replaced by tokens. The tokens are recursively lexicalized. This algorithm is similar to the semantic head-driven algorithm described in (Shieber et al. 1990). It is however more complex. This will be illustrated by building a g-derivation tree from the conceptual representation E0 of our reference example (Section 1).

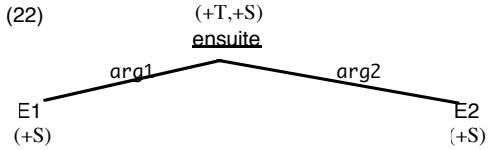
E0 is an instance of SUCCESSION whose LB was presented in (17). Recall that this LB is structured (thanks to the form features) in underspecified g-derivation trees which give rise to a text (those with ensuite and auparavant) and trees which give rise to a sentence (those with avant and après). When synthesizing the first token, a stylistic heuristics says to select an underspecified g-derivation tree which gives rise to a text, so as to avoid producing a long sentence with a lot of subordinate clauses. Moreover, each element in LB(SUCCESSION) is annotated with a feature (not represented in (17)) which indicates whether the chronological order is respected or not: this is the case for ensuite, but not for auparavant. The chronological order is chosen by default. Therefore, the underspecified g-derivation tree for ensuite is selected and instantiated, which leads to the tree in (22).

---

[13] The elementary trees for adjectives in their predicative use include a verbal anchor (the copula). The sentences *Jean est amoureux de Marie (John is fond of Mary)* et *Jean aime Marie (John loves Mary)* comprise respectively the predicates *être amoureux* and *aimer*. When they are modifiers, adjectives are handled by adjunction as usual.

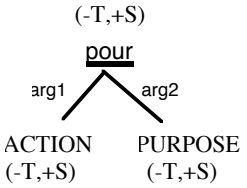
[14] In fact, the lexicalization process leads to a list of underspecified g-derivation trees in order of preference, so as to reduce backtracking in case of incompatibility with other decisions. However, the complete data flow is not within the scope of this paper (see Meunier 1997).

(22)

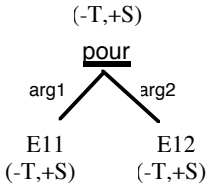


The next step consists of lexicalizing E1 whose class is GOAL. LB(GOAL) comprises only the underspecified g-derivation tree for the subordinating conjunction *pour (que)* [15]*(in order that / to)*. So LB(GOAL) is (23) which is instantiated in (24) for E1.

(23)

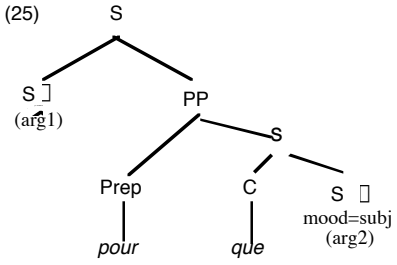


(24)

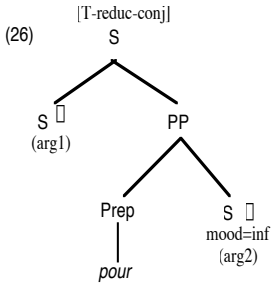


The lexical entry pour points to the elementary trees shown in (25) and (26)[16] (putting aside the trees marked with the T-feature [T-Anteposition] for anteposed subordinate clauses). The tree in (25) is the canonical tree where the conjunction introduces a finite clause; the tree in (26), marked with the T-feature [T-reduc-cunj], is used when the conjunction introduces an infinitive clause.

(25)



(26)



Since the subordinating conjunction *pour (que)* can introduce an infinitival clause, a special mechanism is used to lexicalize E11 and E12 in (24) (they have to be lexicalized as sentences because of their feature (-T, +S)). This mechanism comes from a French stylistic heuristics: an infinitival clause is

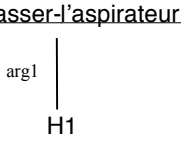
---

[15] The conjunction *pour (que)* has lexical variants (Section 2.4) like *dans le but (que)*.

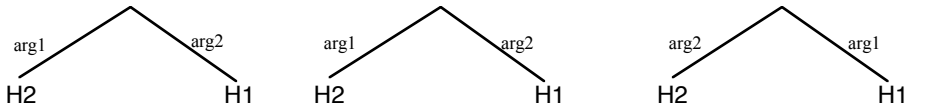
[16] For the sake of simplicity, these trees are initial trees. However, they could be auxiliary trees (with the node S corresponding to arg1 as foot node) as well, see Section 6.

better than a finite one. Therefore, the lexicalization operations for E11 and E12 in (24) are not carried out independently, but in such a way that they yield (if possible) sentences with the same subject, so that pour introduces an infinitival clause. This could apply to E11 and E12 since they share a token, i.e. H1. So the goal is to find sentential lexicalizations for E11 and E12 so that H1 is subject in both sentences. The only possible lexicalization for E11 is passer-l'aspirateur (ran-the-vacuum-cleaner) whose arg1 is H1, see (27a). For E12, instance of RWDING, there are the three underspecified g-derivation trees given in (15) which are instantiated in (27b).

(27a)



(27b)



The syntactic functions of the NPs synthesizing H1 in these trees have to be known to select the trees in which H1 will be synthesized in the subject position. For that purpose, data bases are extracted from the TAG grammar which record for each family the functions of the syntactic arguments according to the syntactic constructions. For example, in the transitive verb family:

- arg1 is the subject and arg2 is the object in the canonical construction (without T-feature),
- arg1 is the by-object and arg2 is the subject in the passive construction (identified with the T-feature [T-passive]),
- etc.

Therefore it is known that:

- in (27b) for E12:
  - arg2 of récompenser (i.e. H1) is the subject in the passive construction,
  - arg2 of donner-récompense (i.e. H1) can never be subject[17],
  - arg1 of recevoir-récompense (i.e. H1) is the subject in the canonical construction;
- in (27a) for E11:
  - arg1 of passer-l'aspirateur (i.e. H1) is the subject (in the canonical construction).

[17] The English passive form *John was given a reward by Mary* does not exist in French.

As the goal is to build sentences whose subjects refer to H1, the tree with donner-récompense is eliminated for E12 (the tree with passer-l'aspirateur is selected right away for E11). So two possibilities are left for E12: either récompenser marked with [T-passive] or recevoir-récompense without T-feature. In order to illustrate an alternation, let us suppose that the generator chooses the former, i.e. récompenser [T-passive]. The lexicalization of E1 is therefore (28), which is obtained from (24) by substituting the tree in (27a) for E11 and the leftmost tree in (27b) with the T-feature [T-passive] for E12. Moreover, as it is known that the subordinating conjunction will introduce an infinitival clause, [T-reduc-conj] is added to pour to indicate that it will introduce an infinitival clause, [T-reduc] is added to récompenser [T-passive] to indicate that the sentence will have an empty subject, and the symbol □ (which represents the empty sequence) is added to arg2 of récompenser to indicate that it will be synthesized as an empty sequence[18]. The elementary tree defined by récompenser [T-passive] [T-reduc] is shown in (29) in which the node N which dominates the empty sequence allows to transmit the agreement features to the past participle thanks to morphological features[19].

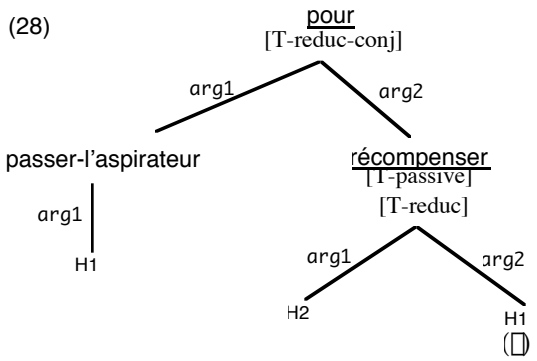
[18] This node is not deleted because it is not empty at the semantic level (although it is empty at the phonological level). On the other hand, a node which corresponds to a conceptual argument which is not specified is deleted. For example, from an instance of EAT in which EATEE is not specified, the node corresponding to the arg2 of manger (eat) is deleted and the T-feature [T-without-arg2] is added to manger.

[19] In French, these morphological features are needed for agreement rules in the infinitival clause: see (i) versus (ii).

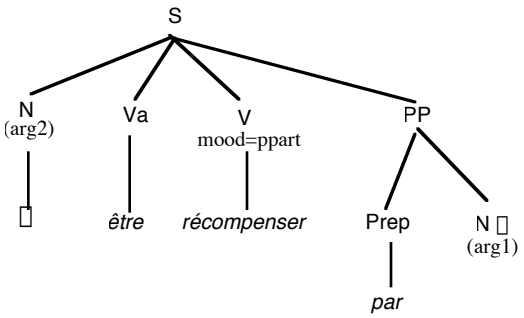
(i) **Jean** a passé l'aspirateur pour être récompens**é** par Marie.

(ii) **Sue** a passé l'aspirateur pour être récompens**ée** par Marie.

(28)

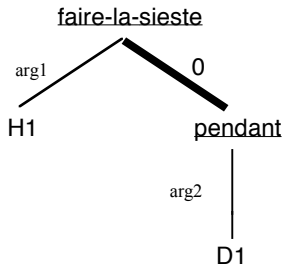


(29)



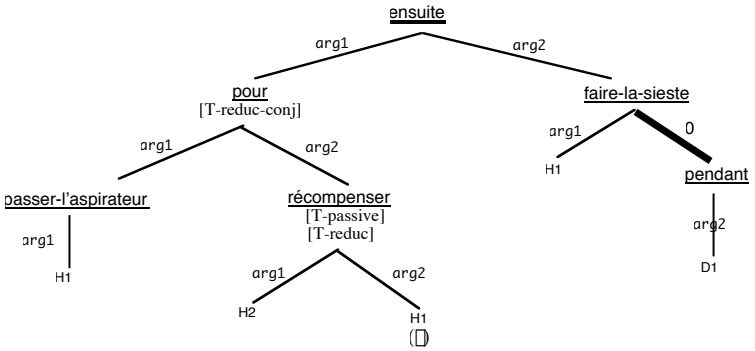
Since all the instances of RELATION should be lexicalized before the instances of THING (for pronominalization issues, see below), the lexicalization of E2, instance of NAPPING, in (22) is performed after that of E1. As E2 includes the modifier DURATION, the lexicalization of this token is its lexicalization without a modifier to which is adjoined the lexicalization of its modifier. Recall that the underspecified g-derivation trees in LB(NAPPING) shown in (21) have respectively the constant nodes faire-la-sieste which builds a sentence - form feature (-T, +S), and sieste which builds an NP - form feature (-T, -S). As E2 is marked in (22) with the feature (+S), the g-derivation tree for faire-la sieste is selected. The lexicalization of E2 is therefore (30).

(30)



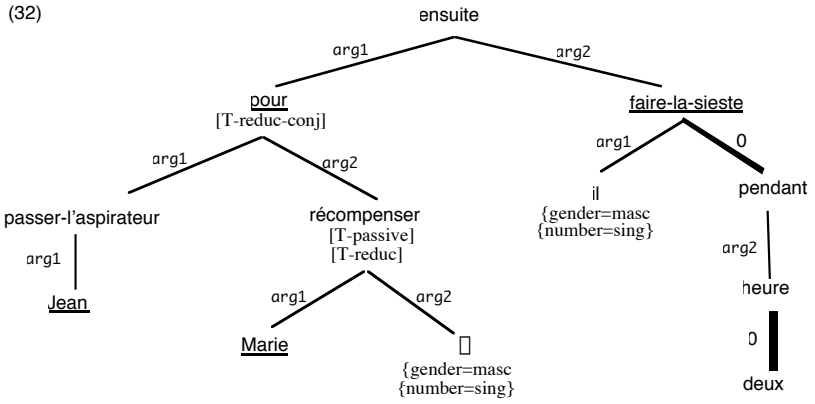
At this stage, all the instances of RELATION have been lexicalized and the result is shown in (31), in which the trees in (28) and (30) have been substituted respectively for E1 and E2 in (22).

(31)



In (31), three tokens which are instances of THING have to be lexicalized:
- H2 and D1 which occur only once. Therefore, their lexicalization is straightfoward.
- H1 which occurs three times, one occurrence being marked with (□). There is no room to describe the pronominalization module, which is quite complex in French because of pronominal clitics (Namer 1990, Danlos 1992). However, one can assume that, in this simple case for pronominalization, this module computes that the occurrence of H1 which is arg1 of faire-la-sieste is pronominalized as the subject pronoun il. So finally, the g-derivation tree computed from E0 is shown in (32), in which the morphological features added to □ and il come from Jean.

(32)



One type of information is missing in (32) to compute a g-derived tree (Section 2.4), namely the temporal and aspectual information for verbal items. I have not studied this topic and for the time being use a default value, e.g. "passé composé" (roughly perfect). So (1) is produced from (32).

To sum up, the algorithm for building a g-derivation tree from an instance of RELATION goes as follows:

1. lexicalization of the instances of 2ND-ORDER-RELATION. At the end of this step, an evaluation of the global text structure is carried out (although it has not been illustrated here). If this evaluation returns a bad result, backtracking is used to make other choices.
2. lexicalization of the instances of 1ST-ORDER-RELATION. This step may use special procedures to deal with parallelism issues, as illustrated above for the subordinating conjunctions which can introduce infinitival clauses (which is a special case of parallelism).
3. lexicalization of the instances of THING. This step calls upon a pronominalization module.

5 Related works

I first compare the architecture of G-TAG with other architectures proposed for text generation. Next, I compare G-TAG with other generation systems inspired from TAG.

G-TAG is a system in which lexical choice is made after the content specification, and before the surface realization, but very importantly, lexical choice is interleaved with syntactic realization. A g-derivation tree specifies a "thematic structure" (in the terms of Elhadad et al. 1997) with information for syntactic realization encoded in T-features. Therefore, the surface realizer (which builds a g-derived tree from a g-derivation tree and transmits it to the post-processes module) does not make significant linguistic decisions (except to produce surface variants). It just uses of the syntactic information encoded in a lexicalized TAG grammar (extended to handle texts) and to the

rules of the post-processing module. G-TAG is therefore a "lexicalized" approach (in the terms of Elhadad et al. 1997).

The lexicalized approach used in G-TAG, where the lexicon drives the generation process, is also advocated, among others, by (Beale et al. 1998) and (Stede 1996), but only to generate sentences. The originality of G-TAG in comparison with other lexicalized approaches is that the lexicon drives the generation process not only for sentences but also for texts. No difference is made between texts, sentences and NPs. This position is linguistically justified because a given 2ND-ORDER-RELATION can be expressed either in a text or in a sentence or even in an NP (see note 12). This is why I use the term "2ND-ORDER-RELATION" instead of "discourse relation". Similarly a given 1ST-ORDER-RELATION can be expressed in a sentence or an NP (so it is not a "sentence relation"). This position also explains why G-TAG is not modularized into a "text planner" and a "sentence planner" as generally admitted (Reiter 1994). As explained in Section 4, G-TAG is modularized between a component for instances of 2ND-ORDER-RELATION, a component for instances of 1ST-ORDER-RELATION and a component for instances of THING. This modularization is naturally based on the input (i.e. the conceptual structure) and not on the output, i.e. the generated text with its segmentation into sentences, as it is the case for most systems with a text planner. However, I should say that G-TAG is designed for a relatively simple conceptual representation which leads to a paragraph-length text. A more complex conceptual representation has to be pre-segmented into smaller clusters. However, this pre-segmentation is not a segmentation into sentences. Let us underline again that the segmentation of a text (paragraph) into sentences is a "surface matter" which depends (among other things) on the lexical resources of the target language. For example, GOAL can only be expressed in a (complex) sentence in French (Section 4), while SUCCESSION can be expressed in a text, a sentence or an NP (Section 3). Another example: RESULT can be expressed in English in a resultative construction with only one verb *(Ted hammered the metal flat)*, while two clauses (with two verbs) are needed in French like in *Ted hammered the metal. He flattened it*.

Let us now move to works in text generation inspired from TAG. Since it has been put forward that TAG is an especially well suited grammatical theory for text generation - see (Joshi 1987), (McDonald & Pustejosky 1985), (McDonald 1993) - adapting TAG for generation has already been explored, mainly by (McDonald & Meteer 1990), (Harbusch & al. 1991), (Shieber & Schabes 1991), (McDonald 1993), (Stone & Doran 1997), and (Becker 1998). Most of these authors address issues related to the generation of a single sentence from its logical or deep syntactic representation. G-TAG differs from these approaches, as it is a completely integrated text generation formalism: it is designed to take a conceptual representation as its input, to handle textual phenomena and to produce texts of good quality. However, the system proposed in (Stone & Doran 1997) takes as its input the conceptual representation of a sentence, so let us compare their system with

G-TAG. In their system, there is no semantic level. There is a direct mapping between a conceptual structure (enriched with pragmatic information) and the syntactic level, the TAG grammar. Consequently, the elementary trees are annotated both with conceptual information (e.g. in the canonical elementary tree for *reward*, the subject is marked as being the RWDER) and with pragmatic information (in the topicalized construction illustrated in *This book, the library has*., the entity referred to by *this book* is roughly salient). There seems to be no major drawback with this direct conceptual-syntax interface, except for the organization of the lexical information. The conceptual-syntax data base for a given concept has to record not only all the predicates lexicalizing it but also, for each of those lexemes, all its syntactic realizations. The size of such a lexical data base can be unacceptable (it could have 350 elements if there are 10 lexemes per concept and 35 syntactic realizations per lexeme). In G-TAG, this problem does not occur: in the conceptual-semantic interface, a data base for a given concept records all the lexemes lexicalizing it, and in the semantic-syntax interface, a data base for a given lexeme records all the syntactic constructions realizing it. Each data base is relatively small. Moreover, in the system of Stone & Doran, if a lexeme lexicalizes several concepts, i.e. has several meanings, while keeping the same syntactic properties, a set of elementary trees has to be created for each meaning (since elementary trees record conceptual information). This is useless. In G-TAG such a lexeme appears in several LBs in the conceptual-semantic interface, but it appears only once in the semantic-syntax interface. To sum up, the system of Stone & Doran may run smoothly to generate a sentence from its conceptual representation, but only in a toy implementation where it is assumed that each concept is lexicalized by a unique lexeme and that each lexeme has a unique meaning.

6 Implementation and applications of G-TAG, future work

The idea of using a g-derivation tree as a semantic level intermediary between a conceptual structure and a text is satisfactory and yields a lexicalized formalism which has been implemented and used in several applications. G-TAG, which I started to design at the beginning of the 90's - see (Danlos 1993, 1995, Danlos & Meunier 1996) - has been first implemented in Ada (Meunier 1997). The platform so realized, called Flaubert, has been used for three applications in technical domains: software, chemicals and aeronautics (respectively for Sanofi, CEA and Aérospatiale companies), see (Delaunay 1995, 1996), (Lux 1998), (Meunier & Danlos 1998). The texts produced are both in French and in English. The French TAG grammar embedded in G-TAG is the one written by (Abeillé 1991) and revised in (Abeillé & Candito this volume); the English grammar was especially designed for G-TAG.

G-TAG has been re-implemented in Java in a multi-agent structure, in collaboration with Thomson-CSF company, (Meunier 1999, Meunier & Reyes 1999). The platform so realized is called CLEF (Computed Lexical-

Choice Extended Formalism). Contrary to Flaubert, CLEF has the following advantage: the linguist in charge of writing the conceptual-semantic interface does not need to know the TAG grammar in detail. For example, he/she can ignore whether a subordinating conjunction (e.g. *pour (que)*) anchors an initial or auxiliary tree: the semantic dependency tree in (23) can be used in the conceptual-semantic interface even if *pour (que)* anchors an auxiliary tree (as it is the case in the French TAG grammar embedded into G-TAG). The system computes the corresponding g-derivation trees according to the elementary trees of the TAG grammar. Let us underline however that the automatic computation of g-derivation trees from semantic dependency trees is possible only if the target TAG grammar respects the predicate argument co-occurrence principle (predicates anchor elementary trees with positions for all and only their semantic arguments). In other words, this computation is possible for a lexeme which has the same number of arguments at the semantic and syntax level, whatever the nature of the syntactic arguments (substitution or foot node). This computation is thus possible for a subordinating conjunction. It would not be possible for a discourse cue such as *ensuite* with two arguments at the semantic level and only one argument at the syntactic level (Section 2.3). This is why elementary trees for adverbials such as *ensuite* with two sentential arguments have been created in G-TAG: the classic elementary trees used in TAG with a single argument would not work.

I will conclude by comparing my work with that of (Webber & Joshi 1998) who present a TAG grammar for discourse (not in the perspective of generation, but this is not a relevant difference). Without going into details, in G-TAG the four ways to link two sentences together, i.e. *S1 cunj S2*., *S1*. *Adv S2*., *Adv1 S1*. *Adv2 S2*., and *S1*.*S2*., all involve a predicate with two sentential arguments (respectively *cunj*, *adv*, *adv1 … adv2*, and ⊕). Thus, they get the same kind of semantic representation, but this is not the case in Webber & Joshi's work. As an illustration, even the *S1 but S2*. and *S1 and S2*. texts do not involve the same kind of elementary trees and therefore yield different derivation trees. Webber & Joshi want the distinction between presuppositional versus compositional meaning of discourse cues to be stated in the elementary trees. I argue that this distinction should not be taken into account at the level of elementary trees, which is basically designed for morpho-syntactic phenomena. This point of view is supported by the following fact: as far as I know, nobody has never designed two elementary trees for the definite article, one for its presuppositional meaning (as in *The king of France is bald*), the other one for its anaphoric meaning (as in *A king entered*. *The king kissed me*.). More generally, a TAG grammar seems not to be the right tool to handle presuppositions.

On the other hand, Webber & Joshi can easily handle a *S1*. *Adv1 Adv2 S2*. text, i.e. a text with two discourses cues in the second sentence. This

kind of text is not yet handled in G-TAG but should be dealt with in some future work.

Acknowledgments

I want to thank Anne Abeillé, Marie Hélène Candito, Sylvain Kahane, Frédéric Meunier and Owen Rambow for fruitful discussions on this paper.

References

Abeillé A., 1991. Une grammaire lexicalisée d'arbres adjoints pour le français□ application à l'analyse automatique, Thèse de Doctorat en Linguistique, Université Paris 7, Paris.

Abeillé A. Candito M.H., 2000 (this volume). "F-TAG : A Lexicalized French Tree adjoining Grammar", in Abeillé et Rambow (eds), *Tree Adjoining Grammars*, CSLI, Stanford

Beale, S., Nirenburg S., Viegas E., Wanner L., 1998, "De-Constraining Text Generation", in Proceedings of the 9th International Workshop on Natural Language Generation (INLG'98), Niagara-on-the-Lake.

Becker T., 1998 "Fully Lexicalized Head-Driven Syntactic Generation", in Proceedings of the 9th International Workshop on Natural Language Generation (INLG'98), Niagara-on-the-Lake.

Candito M.H., 1998. Représentation hiérarchique des grammaires TAG lexicalisées : application au français et à l'italien, Thèse de Doctorat en Linguistique, Université Paris 7, Paris.

Candito M.H.,1996. "A principle-based hierarchical representation of LTAGs", in Proceedings of the 16th International Conference on Computational Linguistics (COLING'96), Copenhagen.

Danlos L., 1992. "Contraintes syntaxiques de pronominalisation en génération de textes", Langages, 106, Larousse, Paris.

Danlos L., 1993. "Génération de textes à partir d'un graphe conceptuel dans un formalisme inspiré de TAG", in Actes de l'école d'été Informatique linguistique du CNET, Lannion.

Danlos L., 1995. "Présentation de G-TAG, un formalisme pour la génération de textes", in Actes de la conférence Traitement Automatique du Langage Naturel (TALN-95), Marseille.

Danlos L., 1998. "Linguistic ways for expressing a discourse relation in a lexicalized text generation system", in Proceedings of Discourse Relations and Discourse markers Workshop, ACL-COLING'98, Montréal.

Danlos L., Meunier, F., 1996. "G-TAG : présentation et applications industrielles", in Actes du colloque Informatique et Langue Naturelle (ILN'96), Nantes.

de Smedt K., Horacek H., Zock M., 1996. "Architectures for Natural Language Generation: Problems and Perspectives", in G. Adorni and M. Zock (eds) Trends in Natural Language Generation, Springer-Verlag.

Delaunay, M. P., 1995, "G-TAG : les bases lexicales dans les domaines de la chimie et du logiciel", Rapport FLAUBERT, n°3, TALANA-CORA, Paris.

Delaunay, M. P., 1996, Ecriture d'une grammaire TAG pour la génération : comparaison avec la grammaire d'analyse, Mémoire de DEA, Université de Paris 7, Paris.

Elhadad M., McKeown K., Robin J., 1997, "Floating Constraints in Lexical Choice, Computational Linguistics vol 23 n° 2.

Harbusch, K., Finkler, W., Schauder, A., 1991, "Incremental Syntax Generation with Tree Adjoining Grammars" in Proceedings 4.Int. GI-Kongress Wissensbasierte Systeme, GWAI.

Harris Z., 1982, *A Grammar of English on Mathematical Principles*, Wiley-Interscience, New York.

Iordanskaja, L., Mel'cuk, I., 1999. "Traitement lexicographique de deux connecteurs textuels du français contemporain□ *en fait vs en réalité*", in Mel'cuk I. et al. eds ,Dictionnaire explicatif et combinatoire du français contemporain, Volume IV., Presses de l'Université de Montréal, Montréal.

Joshi, A., 1987. "The relevance of Tree Adjoining Grammar to Generation", in G. Kempen, ed, Natural Language Generation: New Results in Artificial Intelligence, Psychology and Linguistics, Martinus Nijhoff Publishers.

Lascarides A., Asher N., 1991. "Discourse Relations and Defeasible Knowledge", in Proceedings of the 29th American Conference on Computational Linguistics (ACL'91), Berkeley.

Lux V., 1998, Elaboration d'un Français Rationalisé Modulaire (FREM). Test du FREM en génération automatique., Thèse de Doctorat en Linguistique Informatique, Université Paris 7, Paris.

Mann, W., Thompson S., 1988, "Rhetorical Structure Theory: Toward a functional theory of text organization", Text, 8 (3), pp 223-281.

McDonald, D., 1993, "Reversible NLP by Linking the Grammar to the Knowledge Base", in Strzalkowski (ed.) Reversible Grammar in Natural Language Processing, Kluwer Academic Press.

McDonald, D., Meteer M., 1990, "The implications of Tree Adjoining Grammar to Generation", in G. Kempen, ed, Natural Language Generation, Dordrecht.

McDonald, D., Pustewosky, J., 1985, "TAG's as a Grammatical Formalism for Generation", in Proceedings of the 23th Annual Meeting of the Association for Computational Linguistics (ACL'85), Chicago.

Meunier F., 1997, Implémentation de G-TAG, formalisme pour la génération inspirée des grammaires d'arbres adjoints, Thèse de Doctorat en Informatique, Université de Paris 7, Paris.

Meunier F., Danlos L., 1998. "FLAUBERT: an User friendly system for Multilingual Text Generation", in Proceedings of the 9th International Workshop on Natural Language Generation (INLG'98), Niagara-on-the-Lake.

Meunier F. 1999. "Modélisation des ressources linguistiques d'une application industrielle", TALN'99,Cargèse, Corse.

Meunier F., Reyes R., 1999. "Plate-forme de développement de générateurs multilingues", Actes GAT'99 pp. 145-156, Grenoble, France.

Namer F., 1990, Pronominalisation et effacement du sujet en génération automatique de textes en langues romanes, Thèse de Doctorat en Informatique, Université Paris 7, Paris.

Nicolov N., Mellish, C., Ritchie G.,1997 "Approximate Chart Generation from Non-Hierarchical Representations", in R. Mitkov et N. Nicolov eds, Recent Advances in Natural Language Processing, Benjamin, Amsterdam.

Paris C., Vander Linden K., Fischer M., Hartley A., Pemberton L., Power,R., Scott D., 1995. "A support Tool for Writing Multilingaul Instructions", in in Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCCAI'95), Montréal.

Rambow O., Vijay-Shanker K., Weir D., 1995, "D-Tree Grammars", in Proceedings of the 33th Annual Meeting of the Association for Computational Linguistics (ACL'95).

Reiter E., 1994. "Has a consensus NL generation architecture appeared, is it psycholinguistically plausible?", Proceedings of the 7th International Workshop on Natural Language Generation (INLG'94), Kennebunkport.

Shieber S., Schabes S., 1991, "Generation and Synchronous Tree Adjoining Grammars", Computational Intelligence, vol 4 n°7.

Shieber S., Schabes S., 1994, "An Alternative Conception of Tree Adjoining Derivation", Computational Linguistics. vol 20 n°1.

Shieber, S., Noord G., Moore R., Pereira F., 1990, "A Semantic Head-Driven Generation Algorithm for Unification-Based Formalisms", Computational Linguistics 16:1. 30-42.

Stede, M., 1996, "Lexical paraphrases in Multilingual Sentence Generation", Machine Translation 11.

Stone M., Doran, C., 1997. "Sentence Planning Using TAG", in Proceedings of the Joint conference ACL/EACL '97, Madrid.

Webber B., Joshi A.,1998. "Anchoring a Lexicalized Tree-Adjoining Grammar for Discourse", in *Proceedings of Discourse Relations and Discourse Markers, ACL-COLING'98 Workshop,* Montréal, pp 86-92.