The CLEF semi-recursive generation algorithm

Rodrigo Reyes
Thomson-CSF/LCR
Talana, Univ. Paris 7
February 6th ,2000

Abstract

We will discuss the semi-recursive algorithm for text generation, as defined for the GTAG formalism, and its implementation in the CLEF project. We will show how to use lexical choice constraints and properties of the LTAG grammar to minimize the backtracking of the semi-recursive algorithm.

1 Introduction

GTAG is a multilingual text generation formalism derived from the Tree Adjoining Grammar model ((Joshi and al.,1975), (Shabes and Shieber,1994)). This formalism, which is still evolving, uses slightly enriched TAG grammars as its syntactic back-end.

GTAG only deals with the « How to say it ?» aspect of the generation process. It takes as input a partly computed symbolic representation of the discourse, and defines the processing and the data necessary to produce the final text.GTAG is widely described in (Danlos, 1996), (Danlos, 1998), and (Danlos1999).

Flaubert was the first GTAG implementation, conducted by Cora SA ((Meunier and Danlos, 1998), (Meunier 1997)). This implementation validated the first versions of the formalism, and yielded a stable version of GTAG. A new implementation has been initiated by Thomson-CSF LCR ((Meunier and Reyes, 1999), (Meunier, 1999)), using the Java language, with a strong emphasis on research and applications.

We will give firstly a short insight of the GTAG formalism, then introduce the semi-recursive algorithm in comparison with the recursive approach. Then, we will give a presentation of the CLEF generation algorithm that yields a nearly-surfacic syntactic representation from the conceptual representation (a post-processing phase takes care of the final output).

2 Presentation of GTAG

The GTAG formalism describes the domain model used to specify the input of the generator, as well as the linguistics data and processing necessary to generate texts. GTAG uses a first-order logic formalism for its domain model, and a lexicalized TAG grammar as its syntactic model. We introduce hereafter both sub-formalisms, and the manner in which GTAG links them.

2.1 Domain model

The Login (Logic and inheritance) formalism ((Aït-Kaci and Nasr, 1986), (Meunier, 1997)) has been used to model the domain knowledge. It takes type constraints into account, thus allowing to validate, to a certain extent, the input structures' coherence.

GTAG specifies an additional constraint on the model : the existence of three generic concepts used to divide the conceptual domain, as follows.

- Entities, representing objects (individuals) of the world.
- 1st-order relations, representing simple events, between entities, or between entities and relations.
- 2nd-order relations, representing relations between relations.

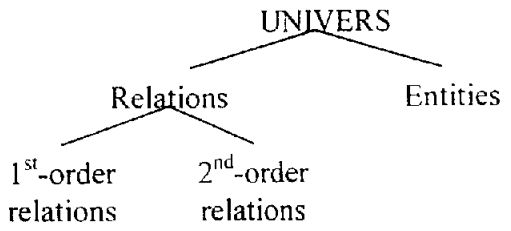

Figure 1. A typical model for the domain.

Thus, a typical model will have the following form, as shown in figure 1.

2.2 The grammar

The syntactic back-end is a Lexicalized Tree Adjoining Grammar, which complies with the Predicate-Argument Cooccurrence Principle (Abeillé, 1993), enriched with the following :

- T-Features, that tag and identify the differences between the syntactic structures (namely the trees). Those atomic tags, such as T_CANONICAL, T_PASSIVE, T_WITHOUT_ARG1, T_WITHOUT_ARG2, etc., are used as a compositional selection mechanism of syntactic structures. For instance, the « passive without agent » structure is determined by the following array of features : T_CANONICAL+T_PASSIVE+T_WITHOU T_ARG1.
- Forms features, associated with each of the grammar trees, indicating their textual level. The possible values for the form features are as follows :
  - -T-P : for noun phrases.
  - -T+P : for a single sentence.
  - +T+P : for a text, i.e. various sentences.
  - +P : for either a single sentence or a text.
- G-derivation structures, which correspond to underspecified derivation trees. The nodes can be constant (each node is equivalent to an elementary tree, as in standard derivation trees), or variable. In the latter case, the node can be associated with either a concept attribute name, or with an instantiated concept.
- Syntactic functions, which are encoded in the grammar.

2.3 The semantic-conceptual interface

The semantic-conceptual interface is provided by the lexical bases (LB), which relate each instantiated concept from the generator input structure with an array of g-derivation trees. This interface associates arguments (from the grammar) with concept attributes (from the conceptual domain).
A lexical base must be able to give, at any time of the process, a valid lexical choice. When it is created, a lexical base refers to a g-derivation tree, selected by default by the domain creator. This lexical choice can be modified by constraints introduced by the generation process.

The GTAG global generation strategy is to modify the lexical base so that the lexical choices be coherent (selected trees can be combined), and that the text be as good as possible.

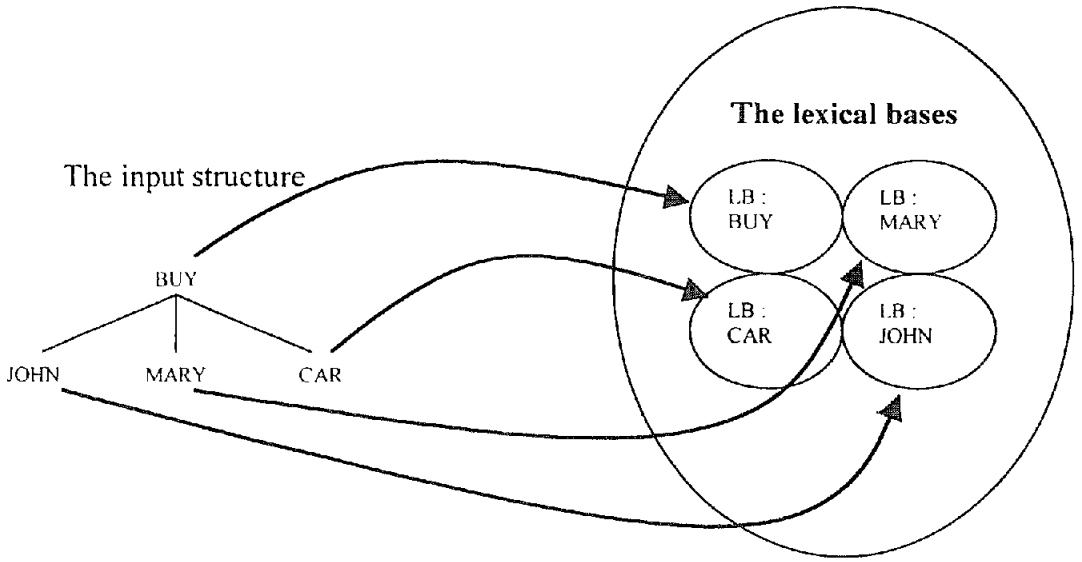

Figure 2. The input structure associated to lexical bases

3 The generation algorithm

Tactical generation algorithms use various strategies which are all input-dependent. Moreover, defining the bounds between the "how to say" and the "what to say" is still an open question. RAGS (rags, 1999) proposes a standard architecture for the data, but leaves the processing details underspecified.

However two main approaches can be noticed : the recursive one and the deductive one. The recursive approach is basically a depth-first backtracking search (for example (Nicolov, 1998)), while the deductive one uses inference mechanisms such as those used in expert systems or specialized languages such as PROLOG (Panaget, 1997). As deductive systems are often used as opaque ways of resolving problems, we will focus on the recursive algorithm, that can easily be used as a base for the customizing of algorithms.

3.1 The input structure

The input of the CLEF generation system is a hierarchical representation (i.e. a tree structure) of the conceptual structure. Therefore, a crucial choice is done before the proper linguistic generation : selecting the theme and the rheme of the utterance[1].

The main advantage from a technical point of view is the processing linearization : such a structure is not ambiguous regarding to the mapping between the elements of the input structures and the elements of the grammar. The input structure is therefore always considered as a single tree, and the text generation algorithm is basically a tree walk on this structure, with a lexical choice processing for each node.

3.2 Lexical choice constraints

The lexical choices made by the lexical bases are modified by constraints that are related to different aspects of the selection : either on the T-Feature, or on the linear order of the elements, or on the realisation of the arguments, etc.

A list of constraints (initially empty) is associated to each lexical base, and the generation process adds new constraints while parsing the input structure. There are two types of constraints additions :

- implicit constraints. Such constraints are added by a parent LB that added a constraint due to its own internal lexical choice. For example, the lexical selection of "S1 before S2" (for a succession concept) imposes grammatical constraints on the argument "after" (related to S2) so that the selection of the argument be grammatically compatible (i.e. add a constraint that imposes the use of an infinitive sentence).
- explicit contraints. Those constraints are added by stylistic rules that will carry on the lexical choice in order to avoid poor style, or to prevent dead-ends in the generation process. For example, the parallelism rule (Meunier 97) should impose that two verbal predicates use the same syntactic function for an argument they have in common.

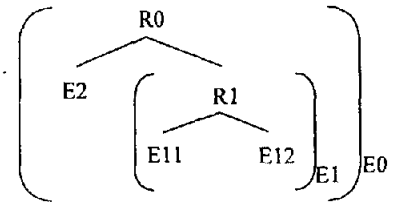

Figure 3 : R0 and R1 are two 2nd-order relations. E2, E11, and E12 are two 1st-order relation. E0 and E1 are two global schemas.

Every constraint addition is associated with a position in the input structure walk, so that it can be removed whenever the backtracking is used. We will also discuss how the backtracking can be partly avoided taking into account some properties of the algorithm, and using a minimum constraint propagation technique.

[1] This choice is clearly arbitrary, because it is equally relevant to the "what to say" and the "how to say it". Such a choice for CLEF was mainly guided by technical considerations.

3.3 The semi-recursive algorithm

(Danlos, 1996) emphasizes on the problems tied to the use of a recursive depth-first algorithm in the area of text generation. More specifically she discusses the impossibility of preventing poor stylistic choices, even when they can be easily predicted. In fact, the problem holds in that stylistic or grammatical rules use information that are computed later in the generation stage by the recursive algorithm.

Thus, in the examples given by (Danlos, 1996) (see figure 3), the two 2nd-order relation choices are obviously linked to each other. Nevertheless, the computation of the selection of R1 is not done until other selections are done (at least E2, in this example). In this way, if no lexical selection satisfies the syntactic or stylistic constraints, the generation process will backtrack on the whole array of previous selections.

Some techniques can be used to partially make up for the problem, for example the memoization ((Nicolov, 1998), (Becker, 1998)), but it does not solve the problem. The fact is that depth-first recursive approaches are not adapted to text generation, where lexical choices must be done in a global, holistic perspective (Danlos, 1998) and (Busemann, 1993).

In this perspective, (Danlos, 1996) proposes a different algorithm, called "semi-recursive" algorithm, in that it remedies to the main drawbacks of the recursive algorithm. The latter is caracterized by the following features :

- The lexical choices of the different levels of relations are carried out in parallel.
  The combinations of the trees and the stylistic choices are carried out separately for each level of concept. Thus, the consistency of all the lexical choices for a particular level is ensured, and each level of concept is considered globally.
- The compatibility tests between the selections (i.e. the three levels of concepts) are carried out. If the combination is valid, it is accepted, otherwise some new selections are done until the compatibility tests succeed.

The approach of this algorithm is particularly relevant, as the consistency is not ensured merely for the array of previous lexical choices (which is not enough, as we discussed), but for the whole set of lexical choices on the same level. This provides a realistic implementation of the global approach.

3.4 The CLEF algorithm

The CLEF algorithm is a variant of the semi-recursive algorithm. In fact, the main idea of the semi-recursive algorithm is the separated processing of the different levels (entity, 1st-order relations, 2nd-order relations).

One problem remains : although the context is taken into account, it is only used in the same level of concepts. Thus, both the 2nd-order relations and the 1st-order relations remain

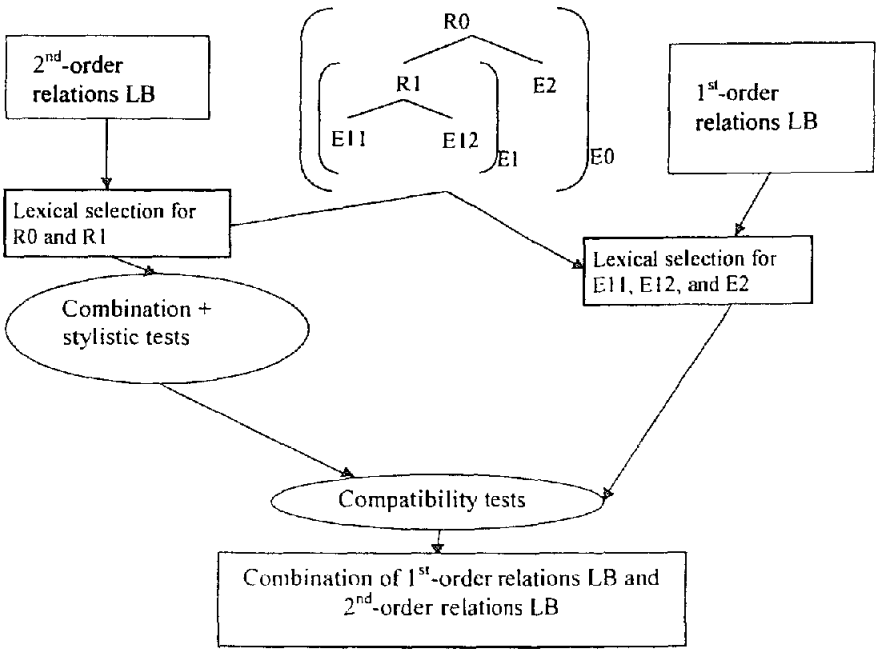

Figure 4. The semi-recursive algorithm.

independent from each other, and in case of failure of the compatibility test, incompatible selections must be computed again. This is due to the fact that choices are carried out in parallel. In order to solve this problem easily, computation should be carried out sequentially and the different levels should be computed in a predefined order. In this case, the question arises : in which order should the different conceptual levels be computed ?

Several evidences indicate that higher level elements should be selected first, then the lower levels (i.e. the 2nd-order relations first, then the 1st-order relations, and then the entities). In fact, on the rhetorical point of view, the higher level elements (in GTAG, the 2nd-order relations) determine the text argumentative structure, thus providing stylistic consistency on the whole generated text. Were they not selected first, they would be constrained by the lexical choices of the other types of concepts. In other words, they would yield to constraints other than purely stylistic, which is not suitable for elements which first criterion of choice is, precisely, stylistics.

Moreover, it seems that in numerous cases, it is preferable to select the simpler elements according to more complex ones. This corresponds to the approach developped by (Rastier and al., 1994), that shows that an element is only relevant in its surrounding context. Such an approach is relevant in our framework, since a particular lexical selection can only be done with full knowledge of the facts if its context is known. By context, we mean the conceptual-semantic context (eg. a reference to an entity that already exists in the discourse), the lexical context (eg. some lexical selection that has already been used for an entity), and the syntactic one (eg. a previous lexical selection imposes some syntactic constraint). Many essential information, for example to decide whether a noun phrase must be pronominalized or not, whether a verb can be elided or not, are available only if the surrounding context does exist and is known.

The "principle of the determination of the local context by the global one" (called "hermeneutics principle" in (Rastier and al., 1994)) can therefore be applied only if the global context is already computed, then the local one, according to the global context. In order for the generation process to be compliant with this principle, elements should be computed in the following order : 2nd-order relations first, then 1st-order relations, and then entities.

Proceeding otherwise would be inconsistent : it is not possible to determine the lexical-syntactic selection of an entity without knowing if it is bound to a noun or a verb. The two possibilities are not necessarily available for a given concept, and carrying on without this piece of information could be considered a last resort.

Besides surrounding context, the local context is also necessary, as shows the *perspective* notion which can be found in (Busemann, 1993), and also supported by (Rastier and al., 1994). It is therefore necessary to know the dependents (the children in the input tree structure) in order to make a lexical-syntactic choice.

These elements were crucial for the design of the CLEF generation algorithm, which we will described hereafter.

3.4.1 Linearization of the algorithm

The CLEF generation algorithm considers the three conceptual levels one by one, carrying out the lexical selection at first on the 2nd-order relations, then on the 1st-order relations, and finally on the entities.

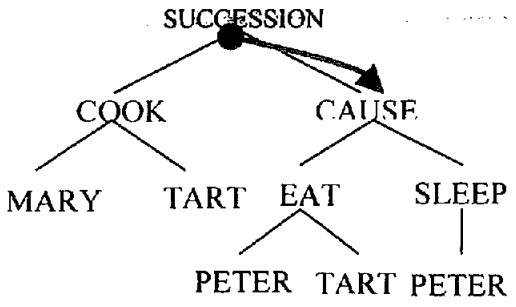

Figure 5. The 1st phase

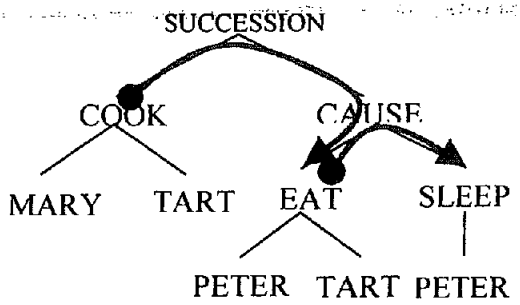

Figure 6. The 2nd phase

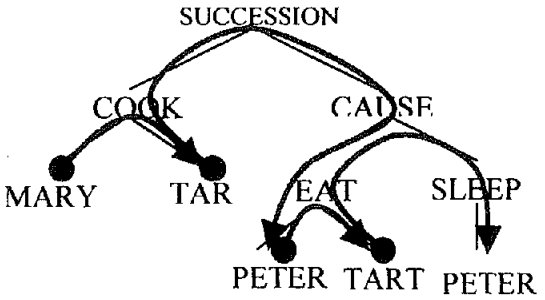

Figure 7. The 3rd phase

The graph walk has the following properties :

- It is carried out in **three phases**, one for each conceptual level.
- The walk is done depth-first, but **following the surfacic linear order** of the elements. Thus, the walk order for the 1st-order elements depends on the lexical choices for R0 and R1 (see fig.3) : if the lexicalized tree selected for R1 situates E2 before R1, then E2 should be computed before E11 and E12. The same way, for E11 and E12, the order of the processing will depend on the tree selected by R1. On our example (fig. 5 to 7), this means that the order of walk of the second phase (regarding the 1st-order relations) will depend on the lexical choices took during the first phase. For instance, if the SUCCESSION concept is lexicalized using a tree anchoring « S1 then S2 » (like in « Mary cooked a tart, then ... »), then COOK will be computed before EAT and SLEEP. On the contrary, if a tree anchoring « S1 after S2 » is used (like in « Peter ate the tart and fell asleep after... »), EAT and SLEEP will be computed first, then COOK. This important property of the algorithm allows to make lexical choices according to previous ones.
- A stack is added, and allows the storage of every lexical choice according to their linear order. This stack stores the history of the choices carried out, and thus allows to backtrack when needed.
- During the tree walk, some constraints are propagated towards the lower elements of the tree. So, a lexical-syntactic selection of a concept would be able to add constraints over the lexical choices of the lexical bases lower-dependent. For example, such a concept could select a particular Form feature, or a particular set of T-Features, for one of its dependent.

Unlike the semi-recursive algorithm, the global context choice also carries out according to local choices. The stylistic rules can not only use the

information given by the concepts of the same level, but also the information given by the dependent nodes, which allows the retrieval of some predictable information. This information is, of course, limited as the lexical-syntactic choice of the dependent lexical bases if not performed at this point.
For example, if E2 and E12 refers to the same concept, some constraints could be computed as soon as the lexical selection for R0 is done, and added to the lexical bases of the dependent nodes of R0, in particular R1.

3.4.2 Minimum constraint propagation : controlling the backtracking

The backtrack, when the processing comes to an impossibility for a lexical base to get a satisfactory lexical choice, is not excluded, although it is inherently limited by the nature of the graph walk.
The handling of the backtracking can also take advantage of both the walk mechanism and the data structures used.
For example, if the algorithm fails to find a valid choice for the E12 element (see figure 3), the backtracking can be performed directly on R1, to find an alternative choice compatible with all the dependent nodes (that is E11 and R0, in our example). If a modification is both available and compatible with the related lexical bases, it will be validated. For a lexical base, a modification is considered compatible if it does not imply any modification in the set of implicit constraints added to the related lexical bases, except the lexical base from which the processing backtracks.
In our example, the backtracking process can ask the LB for R1 to find an alternative choice that modifies the constraints on E12, without modifying any other, that is letting the implicit constraints for E11 et R0 untouched. In other words, it consists in adding a new constraint on R1 that will imply the selection of a new lexical-syntactic structure so that the implicit constraints on E11 and R0 remain identical, and so that the implicit constraints on E12 impose a new, compatible, choice.
The algorithm can therefore carry out a minimum propagation of the constraints, without necessarily doing a full backtracking. In that case, it attempts to find the best compromise with the related lexical bases, before falling back to the normal backtracking mechanism.

4 Conclusion

We discussed how to implement and improve the semi-recursive algorithm described in (Danlos1996), so that the global (holistic) approach ((Danlos, 1996) and (Rastier and al.,1994)) be realistically implemented. Because of its nature, the algorithm is intented rather for best-first generation, and several improvements are still being studied, regarding the paraphrase and the dynamic construction and updating of the lexical bases.

5 References

(Abeille, 1993) Abeillé A. (1993) : « Les nouvelles syntaxes : grammaires d'unification et analyse du français », Armand Colin, Paris.
(Aït-Kaci and Nasr, 1986) Aït-Kaci H, Nasr R. (1986) : « Login : A logic-Programming Language with Built-In Inheritance », in Journal of Logic Programming, 3.
(Becker, 1998) Becker T. (1998) : « Fully Lexicalized head-driven syntactic generation », INLG'98, Niagara-on-the-Lake, Ontario, Canada.
(Busemann, 1993) Busemann S. (1993) : « A holistic view of lexical choice », in Helmut Horacek (ed.), New Concepts in Natural Language Generation : Planning, Realization, and Systems, Frances Pinter, London, New-York.
(Danlos, 1996) Danlos L. (1996) : « Présentation de G-TAG, un formalisme pour la génération de textes inspiré des grammaires d'arbres adjoints », in *Actes TALN-96*, Marseille.
(Danlos, 1998) Danlos L. (1998), « G-TAG: a Formalism for Text Generation inspired from Tree Adjoining Grammar: TAG issues », in Abeillé A., Rambow, O. (eds), *Tree Adjoining Grammars*, CSLI, Standford.
(Danlos, 1999) Danlos L. (1999), « G-TAG : un formalisme lexicalisé pour la génération de textes inspiré de TAG », *TAL*, Vol. 39.2.
(Danlos and Meunier, 1996) Danlos L., Meunier F. (1996), « La génération multilingue : applications industrielles et réalisation scientifique », *Langues situées, technologie communications*, 1996

(Joshi and al., 1975) Joshi A., Levy L., Takahashi M. (1975) : « Tree Adjunct Grammars », in Journal of the Computer and System Sciences, 10 :1.

(Meunier, 1997) Meunier F. (1997), *Implantation du formalisme de génération G-TAG*, Thèse de doctorat, Université de Paris 7

(Meunier, 1999) Meunier F. (1999), Modélisation des ressources linguistiques d'une application industrielle, *Actes TALN'99*, Cargèse.

(Meunier and Reyes, 1999) Meunier F., Reyes R. (1999), Plate-forme de développement de générateurs multilingues, *Actes GAT'99*, Grenoble, France.

(Meunier and Danlos, 1998) Danlos L., Meunier F. (1998), « FLAUBERT : User-friendly multilingual NLG », INLG 1998, Niagara-on-the-Lake, Ontario, Canada.

(Nicolov, 1998) Nicolov N. (1998) :
« Memoization in Sentence Generation with Lexicalized Grammars ». Proceedings of the 4th International Workshop on Tree-Adjoining Grammars and Related Frameworks (TAG+4), Philadelphia.

(Rags, 1999) The RAGS Project (1999) :
« Toward a reference architecture for natural language generation systems ».
http://www.itri.brighton.ac.uk/projects/rags

(Rastier and al., 1994) Rastier F., Cavazza M., and Abeillé A. (1994) : « Sémantique pour l'analyse », Masson, Paris.

(Schabes and Shieber, 1994) Schabes Y., and Shieber S. (1994) : « An alternative conception of the Tree-Adjoining Grammars », in Actes 13rd COLING-90, Helsinki.