SYSTEM DEMONSTRATION
FLAUBERT: AN USER FRIENDLY SYSTEM FOR MULTILINGUAL TEXT GENERATION

FRÉDÉRIC MEUNIER
meunier@linguist.jussieu.fr

LAURENCE DANLOS
danlos@linguist.jussieu.fr

TALANA UFR Linguistique
Case 7003-2, Place Jussieu
75251 Paris
France

1 Introduction

FLAUBERT is an engine for text generation. Its first applications has been for instructional texts, both in French and in English, in software and aeronautics domains. It is an implementation of G-TAG, a formalism for generation inspired from TAG ([Danlos & Meunier 96], [Meunier 97]). This formalism is a lexicalized text generation system ([Danlos 98a], [Danlos 98b]).
All linguistic data are outside of the engine code program. They are maintained directly by linguists under a simple text editor. The syntactic TAG grammar we use for French is that written by ([Abeillé 91]). Moreover, the French families of elementary trees are automatically generated thanks to the hierarchical representation of LTAGS ([Candito 96]). The TAG grammar we use for English is home made.
This engine runs on Sun Solaris with 32 Mo RAM (generator and interface), and is written in Ada 95 (generator) and C (interface). It is compiled by the GNU compilers, and uses GNU scripts (bash, perl, sed, awk).

2 Description

As in DRAFTER ([Paris et al. 95]), FLAUBERT takes as input a conceptual representation provided by the user who fills a questionnaire through an interface that proposes cascading menus based on a domain model (see below). The emphasis is put on linguistic issues such as lexical choices (including choices of connectives), parallelism issues, stylistic issues (e.g. length and content of clauses and sentences), etc.
FLAUBERT uses three databases:

- A domain model describing an ontology of concepts in a typed feature formalism. In a standard way, the concepts include objects, actions, states and relations between them;
- A set of lexical data bases associated with concepts; the lexical database for a given concept describes its semantico-lexical realizations (lexical heads + argument structures) accompanied with tests of applicability for right semantics and well formdness;
- A TAG grammar whose syntactic informations allow a derived tree to be computed from a derivation tree (see the data flow below).

3 Data flow

The data flow of FLAUBERT is given in Figure 1. The system is sequential:

- compiling the input data;
- building a lexicalized tree structure called a "g-derivation tree";
- building a derived tree;

- post-processing.

The first step deals with concepts of the domain and their instances provided by the user. It leads to a conceptual representation. Afterwards, the system search in the lexical data bases to make lexical choices and builds a g-derivation tree. During this step, it uses also other linguistic resources (lexical entry as well as syntactic functions) to optimize lexical choices (parallelism, aggregation, etc.). Next the system builds a derived tree (syntactic representation), using standard algorithms ([Schabes & Shieber 94]) and an existing TAG grammar designed for syntactic analysis. Finally, the text is post-processed (flexion, word re-ordering, typographic considerations, etc.).

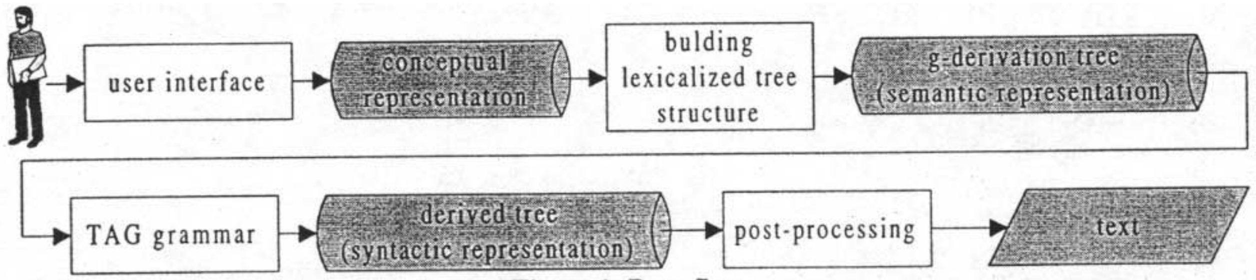
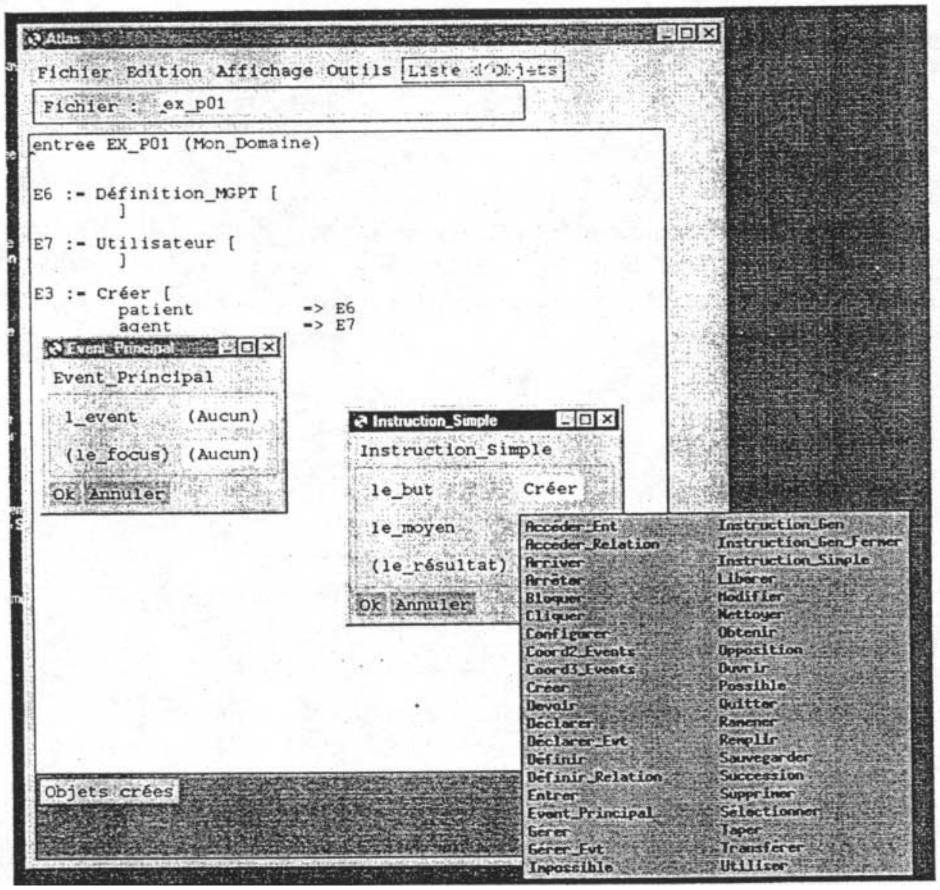

user interface
conceptual representation
bulding lexicalized tree structure
g-derivation tree (semantic representation)
TAG grammar
derived tree (syntactic representation)
post-processing
text

Figure 1: Data flow

3.1 User interface



```
Fichier : ex_p01

entree EX_P01 (Mon_Domaine)

E6 := Définition_MGPT [
          ]

E7 := Utilisateur [
          ]

E3 := Créer [
          patient              -> E6
          agent                -> E7
```

```
Event_Principal
  l_event       (Aucun)
  (le_focus)    (Aucun)
```

```
Instruction_Simple
  le_but          Créer
  le_moyen
  (le_résultat)
```

Since the user may encounter difficulties to give input to FLAUBERT, we have developed a friendly user interface which proposes him/her to instanciate concepts with cascading menus as it is shown in Figure 2. This interface is under X, and can be displayed on most X servers. It invokes the generator in a Xterm which is automatically opened.

3.2 Conceptual representation

Below an example of conceptual representation for an instructional text (in software application domain):

```
E0 := INSTRUCTION [                       E3 := OPEN [
      goal        => E1                         opened=> TOK4 ]
      body        => E2
      effect      => E3 ]
E1 := CREATE [                            H1 := USER [ ]
      creator     => H1
      created     => TOK1 ]               TOK1 := USER_ID [ ]
E2 := SUCCESSION [
      1st-event   => E4                   TOK2 := WINDOW [
      2nd-event   => E5 ]                       name   => "User ID" ]
E4 := OPEN [
      opener      => H1                   TOK3 := BUTTON [
      opened      => TOK2 ]               name   => "Add..."    ]
E5 := CLICK [
      clicker     => H1                   TOK4 := WINDOW [
      clicked     => TOK3 ]               name   => "User name" ]
```

3.3 Semantic representation

From E0, the system computes for French the g-derivation tree shown in Figure 3. In this tree, each node written in bold (possibly accompanied with a [T_Feature], e.g. [T_Réduc]) points to a TAG lexicalized elementary tree, except **newS**, a special tree which adds a new sentence to a text.

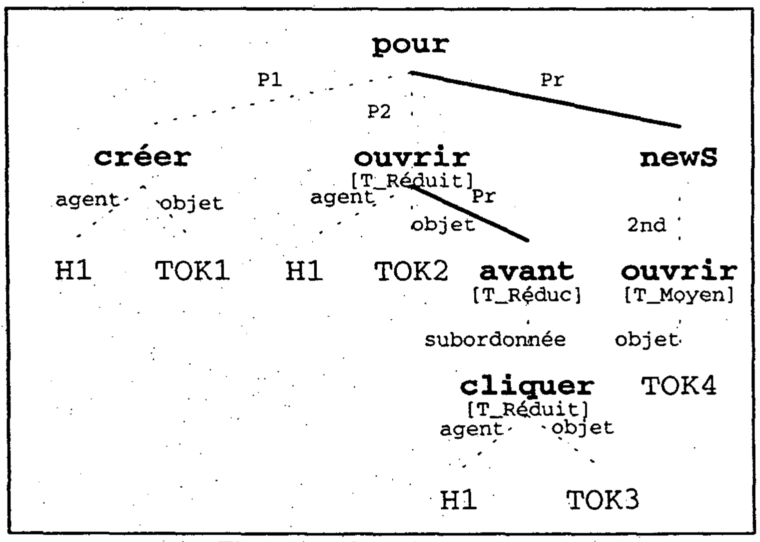


Figure 3 : G-derivation tree

3.4 Syntactic representation

From the g-derivation tree in Figure 3 and with a French TAG grammar, the derived tree schematically resumed in Figure 4 is composed.

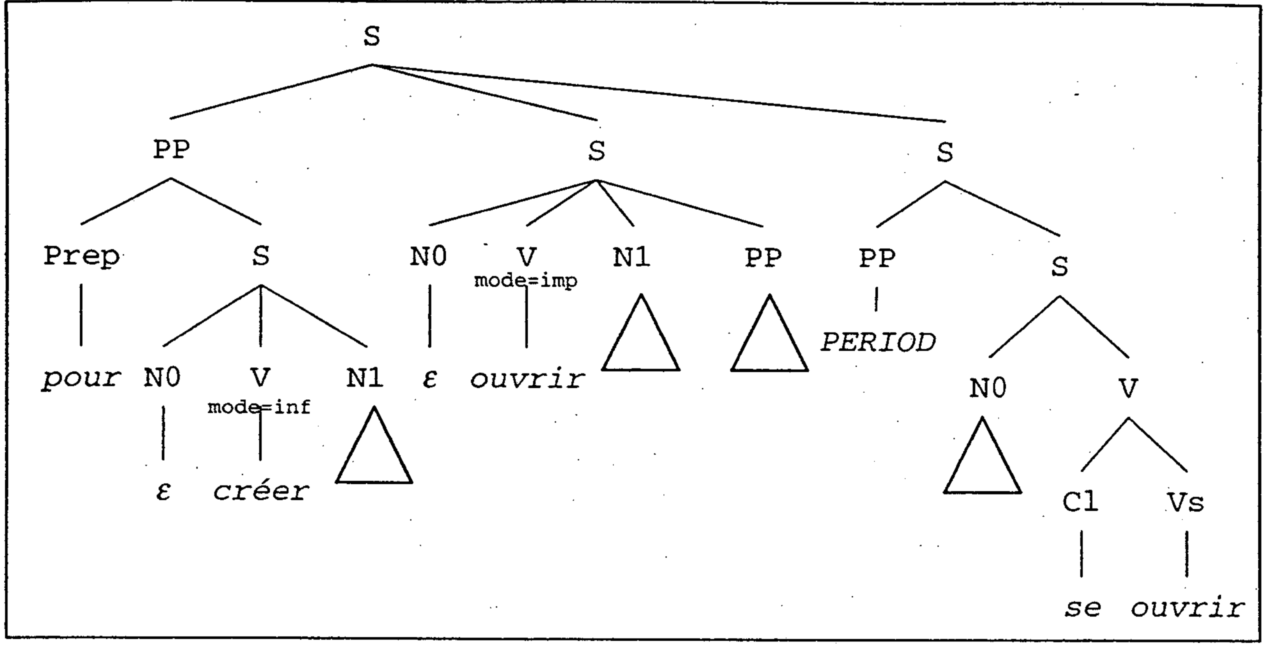


Figure 4: Derived tree

3.5 French and English Texts

French: *Pour créer un identificateur d'utilisateur, ouvrez la fenêtre "User ID" avant de cliquer sur le bouton "Add...". La fenêtre "User name" s'ouvre.*

English: *In order to create an user ID, open the "User ID" window. Afterwards, click on the "Add..." button. The "User name" window is opened.*

References

[Abeillé 91] Abeillé, A. 1991. *Une grammaire lexicalisée d'arbres adjoints pour le français*. Ph.D., Université de Paris 7.

[Candito 96] Candito, M.-H. 1996. A principle-based hierarchical representation of LTAGS. *Proceedings of COLING'96*, Copenhagen.

[Danlos&Meunier 96] Danlos, L., and F. Meunier. 1996. G-TAG, un formalisme pour la génération de textes : présentation et applications industrielles. *Actes de ILN'96*, Nantes.

[Danlos 98a] Danlos, L. 1998. Linguistic way for expressing a discourse relation in a lexicalized text generation system. *Proceedings of COLING-ACL'98*, Montréal.

[Danlos 98b] Danlos, L. 1998. G-TAG: A formalism for text generation inspired from TAG. In A. Abeillé and O. Rambow (eds). *Tree Adjoining Grammars*, CSLI, Stanford.

[Meunier 97] Meunier, F. 1997. *Implantation du formalisme de génération G-TAG*. Ph.D., Université de Paris 7.

[Paris et al. 95] Paris, C., K. Vander Linden, M. Fischer, A. Hartley, L. Pemberton, R. Power, and D. Scott. 1995. A support Tool for Writing Multilingual Instructions. *Proceedings of IJCAI-95*, 1398-1404, Montréal.

[Schabes&Shieber 94] Schabes, Y., and S. Shieber. 1994. An alternative Conception of Tree-Adjoining Derivation, *Computational Linguistics*, 20:1.