# Interface entre les modules *Quoi dire ?* et *Comment le dire ?* dans un système de génération de textes

Laurence Danlos
TALANA
Université Paris 7

Guy Lapalme
SCRIPTUM
Université de Montréal

## INTRODUCTION

Le projet GRABIG mené par l'équipe québécoise SCRIPTUM et l'équipe française TALANA concerne la génération automatique de textes. Il est classique de décomposer le processus de génération en un module qui détermine les informations à transmettre *(Quoi dire ?)* et un module qui traduit ces informations en un texte *(Comment le dire ?)*. Cette décomposition ne va pas sans problèmes mais elle avait déjà été adoptée par les deux équipes de recherche dans des projets antérieurs: SCRIPTUM, dans le cadre du projet SPIN, avait concentré ses efforts sur le module *Quoi dire?* et TALANA, dans le cadre du projet Flaubert, sur *Comment le dire?* L'objectif du projet de collaboration GRABIG est donc de combiner les compétences complémentaires des deux équipes afin d'obtenir un meilleur générateur de texte.

Nous présenterons d'abord les deux systèmes SPIN et FLAUBERT, puis nous exposerons la solution choisie pour les interfacer et les problèmes que nous avons rencontrés. Cette expérience nous a permis de bien faire ressortir la fluidité de la frontière entre les deux modules, *Quoi dire?* et *Comment le dire?* d'un système de génération. Cet article s'appuie sur les rapports effectués en fin d'échanges franco-québecois, en particulier sur (Meunier 1996), (Bouayad-Agha 1996) et (Delaunay 1996b).

## 1 SPIN

SPIN (**S**ystème de **p**lanification d'**i**nstructions en langage **n**aturel), développé par SCRIPTUM (Kosseim 1995, Kosseim et Lapalme 1995) et implémenté en Prolog, génère des instructions, qui se décomposent en une suite d'opérations. En entrée, l'utilisateur décrit un état désiré et l'état courant, en sortie SPIN produit des instructions en langue naturelle pour atteindre cet état désiré. Pour cela, SPIN utilise un certain nombre de bases de données, entre autres, une représentation conceptuelle du domaine et un modèle du lecteur.

Le processus de génération de SPIN est linéaire ; il se décompose en plusieurs modules récapitulés dans la figure 1. La planification de tâche étant le premier module, elle prend en entrée l'état désiré et l'état courant. Par exemple, l'état courant d'un magnétoscope peut être *la cassette vidéo est dans le magnétoscope, il est 10:05 PM,* et l'état désiré : *un enregistrement d'une heure et demie est programmé par une touche incrémentielle*. La planification se fait par unification et expansion des prédicats correspondant à ces deux états, connaissant l'ensemble des prédicats liés aux schémas d'opérations et à la nature de la tâche (de la représentation conceptuelle du domaine et du modèle du lecteur).

À la fin de la planification de tâche, on obtient en sortie la liste des instances d' opérations (un agent donné, un instant donné, un type d' opération) primitives qu' il faut effectuer. Elles sont ordonnées dans un plan non linéaire, et pour chacune d' elles on trouve (comme dans le schéma d' opération) :

- le nom de l' opération ;
- les préconditions pour que l' opération soit effectuée ;
- le corps, soit la suite des sous opérations à effectuer ;
- les postconditions de succès qui donnent le nouvel état du monde courant si l' opération est effectuée ;
- les postconditions d' échec qui donnent le nouvel état du monde si l' opération est mal effectuée.

Ainsi, en reprenant l' exemple de l' enregistrement, on peut obtenir un plan comme

```
régler sélecteur vitesse
  précond : meilleur qualité
  succès : sélecteur vitesse réglé

appuyer sur touche de canal
  succès : canal changé
  echec : non (canal changé)

enfoncer touche OTR 1 fois
  succès : PM 10:35 (30 min)
  echec : PM 10:05

enfoncer touche OTR 2 fois
  succès : PM 11:05 (1 h)
  echec : PM 10:35 (30 min)

enfoncer touche OTR 3 fois
  succès : PM 11:35 (1 h 30 min)
  echec : PM 11:05 (1 h)

appuyer touche TIMER dans 9 sec
  succès : enregistrement activé
  echec : non (enregistrement activé)
```

Une fois la planification de texte faite, on procède aux choix des éléments sémantiques. Pour cela on utilise cinq heuristiques de sélection :

- le contenu et la structure de la représentation conceptuelle (i.e. la sortie de la planification de tâche) ;
- les cooccurrences d' éléments sémantiques ;
- les connaissances et intentions du lecteur ;
- la nature de la tâche ;
- le but communicatif.

Toujours avec le même exemple, on obtient :

```
Structure Sémantique
element semantique: titre
contenu: programmer(obj:enregistrement,man:par une touche
  incrémentielle)

element semantique: option
contenu: meilleure qualité d'image
         régler(obj:sélecteur de vitesse de bande,dest:SP)

element semantique: op_seq
```

```
contenu: régler(obj:sélecteur de vitesse de bande,dest:SP)

element semantique: op_seq
contenu: sélectionner(obj:canal 4)

element semantique: op_seq
contenu: appuyer(dest:touche OTR)

element semantique: op_seq
contenu: appuyer(dest:touche TIMER,man:dans un délai de 9 secondes)
```

Les éléments sémantiques une fois structurés sont traduits en relations rhétoriques selon des heuristiques complexes. Ces heuristiques sont fondées sur :

- le contenu et la structure de la représentation sémantique ;
- les cooccurrences des relations rhétoriques ;
- les connaissances et intention du lecteur ;
- la nature de la tâche.

À chaque élément sémantique est associé un ensemble de relations rhétoriques possibles. Par exemple, pour un EFFET ( O2 est un EFFET de O1 si O2 se produit systématiquement dès que O1 est effectué), on peut choisir parmi les relations rhétoriques dites de but, de manière et de résultat. Selon le niveau du lecteur et/ou la nature d' une opération, on préférera certaines relations rhétoriques à d' autres, tout en prenant en compte des incompatibilités entre relations rhétoriques.

Une fois les choix des relations rhétoriques faits, il faut, comme pour les éléments sémantiques, réordonner les relations rhétoriques selon trois critères :

- les contraintes de cooccurrences ;
- position des satellites par rapport à leur noyau ;
- ajouter la ponctuation.

Toujours avec le même exemple on obtient :

```
Structure rhétorique
titre,
relation rhetorique: but
satellite: programmer(obj:enregistrement,man:par une touche
  incrémentielle)
noyau: _85839,_85840

paragraphe,
relation rhetorique: but
satellite: meilleure qualité d'image

virgule,
relation rhetorique: action
satellite: []
noyau: régler(obj:sélecteur de vitesse de bande,dest:SP)

point,
relation rhetorique: action
satellite: []
noyau: sélectionner(obj:canal 4)

point,
relation rhetorique: action
satellite: []
noyau: appuyer(dest:touche OTR)

point,
```

```
relation rhetorique: action
satellite: []
noyau: appuyer(dest:touche TIMER,man:dans un délai de 9 secondes)

point.
```

La structure rhétorique est finalement transformée en une représentation grammaticale et lexicale. C' est à cette étape que les choix lexicaux sont effectués. Une relation rhétorique ne peut en général se lexicaliser que par l' intermédiaire d' un seul connecteur, par exemple la relation de but se traduit toujours par le connecteur *pour*. Cependant certaines relations rhétoriques laissent le choix entre deux connecteurs, par exemple la "c-condition" se traduit par *si* ou *lorsque* selon le moment où la condition devient vraie. Après quelques ajustements syntaxiques, morphologiques et typographiques (ajustements qui ne relèvent pas d' une vraie grammaire), le texte final est produit. Pour notre exemple, on obtient :

*Programmation d'un enregistrement par une touche incrémentielle*

*- Pour une meilleure qualité d'image, régler le sélecteur de vitesse de bande sur SP.*
*- Sélectionner le canal 4.*
*- Appuyer sur la touche OTR.*
*- Appuyer sur la touche TIMER dans un délai de 9 secondes.*

## 2. Flaubert

Flaubert prend en entrée un graphe événementiel fourni par l'utilisateur grâce à une interface qui lui propose des listes de concepts en cascade. Le système ne comporte donc pas de module de planification ni du modèle du lecteur. L'accent est mis sur les questions linguistiques. Celles-ci sont traitées dans le formalisme GTAG (Danlos 1996, Danlos et Meunier 1996), inspiré des grammaires d'arbres adjoints lexicalisés LTAG développées pour l'analyse (Joshi 1987, Abeillé 1991).

Flaubert utilise trois bases de données :

- modélisation du domaine ;
- bases linguistiques relatives aux concepts du domaine ;
- bases syntaxiques.

La modélisation du domaine en décrit les connaissances (concepts) dans un formalisme d'attributs typés. C'est une taxonomie (c.-à-d. un magnétoscope est un appareil électrique, un appareil électrique est une entité, etc.), qui divise le monde en un treillis (ordre partiel) dont chaque nœud peut être assimilé à une classe d' objets selon la nomenclature utilisée dans les langages de programmation objet. Ces noeuds sont appelés concepts, et regroupent aussi bien les entités du domaine, que les événements simples (les actions comme enregistrer sur une cassette vidéo, les changements d' état comme allumer le magnétoscope, etc.), que les relations binaires entre événements (succession, causalité, etc.). Les concepts sont accompagnés d' attributs (le type d' un magnétoscope par exemple), qui pointent eux-mêmes vers d' autres concepts. L' entrée du générateur est une instance d'un concept événementiel qui se représente graphiquement sous forme de graphe événementiel.

À chaque concept, on associe un certain nombre de réalisations " sémantico-lexicales ", appellées structures de g-dérivation. Pour les concepts décrivant une entité du domaine (magnétoscope), les structures de g-dérivation associées donneront un groupe nominal (" *un magnétoscope VHS* "). Pour les concepts décrivant un événement simple (par exemple enregistrer), on aura des structures de g-dérivation qui donneront une phrase simple

articulée autour d' un verbe (“*enregistrer une émission*”) ou un groupe nominal articulé autour d' un nom prédicatif (“*l' enregistrement d' une émission*”). Pour les concepts décrivant une relation binaire (par exemple une succession), les structures de g-dérivation donneront soit un texte (“*premier événement. Puis deuxième événement*”), soit une phrase avec enchâssement d' une subordonnée (“*premier événement, avant que deuxième événement*”), soit une phrase (“*premier événement avant le deuxième événement*”). Chaque structure de g-dérivation est accompagnée d' une batterie de tests sur les attributs du concept auquel elle est associée. Ces tests permettent d' obtenir des textes dans une syntaxe rigoureuse et dans un style soutenu.

Les bases syntaxiques sont construites selon les mêmes principes que les grammaires TAG. Si leur contenu est dépendant de la langue, leur structure ne l' est pas, puisque l' ensemble de la communauté TAG utilise le même formalisme. Flaubert s' inspire largement des grammaires TAG du français développées par (Abeillé 1991), voir (Delaunay 1996a).

Le processus de génération est linéaire, avec des retours arrières en cas d' échec, et se décompose en plusieurs modules :

- choix lexicaux ;
- combinaison des schémas de g-dérivation ;
- détermination/pronominalisation des entités.

Pour chaque concept mis en jeu lors du processus de génération, Flaubert renvoie une liste ordonnée par ordre de préférence des structures de g-dérivation possibles. Cette étape est appelée choix lexical où une structure de g-dérivation est instanciée en un schéma de g-dérivation. Comme G-TAG repose sur une grammaire lexicalisée, les connecteurs entre phrases (signe de ponctuation éventuellement assorti d' un adverbe, conjonctions de subordination ou de coordination) sont considérés comme des items lexicaux au même tire que les verbes ou les noms par exemple. Par conséquent, le calcul de la structure du texte (l' enchaînement des connecteurs entre les phrases) relève de bons choix lexicaux et d' une combinaison des schémas de g-dérivation.

Comme on l' a vu, à chaque relation binaire est associé un ensemble de structures de g-dérivation. Cependant, chacune d' elle peut mettre en relation un événement simple et une relation binaire (et ainsi de suite). Or, il faut éviter les textes tels que :

* *L' enregistrement n' est pas possible parce que l' appareil n' est pas allumé parce qu' il n' est pas sous tension.*

dont le style est lourd. Ainsi lors de la combinaison des différents schémas de g-dérivation le générateur met en œuvre un certain nombre de tests de compatibilité. Certains tests sont faits pendant les choix lexicaux, mais il subsiste des tests globaux effectués à l' issue des choix lexicaux (en cas d' échec on fait un retour arrière), c' est pourquoi le processus n' est pas entièrement linéaire.

Après combinaison des schémas de dérivation, on obtient un schéma de texte où les variables qui restent sont des entités qui doivent être synthétisés en groupe nominal ou pronominal. Un calcul sur l' ensemble de l' arbre (l' ensemble du texte) va permettre de décider le type de déterminant à choisir pour un groupe nominal (défini, démonstratif, etc.) ou s' il faut pronominaliser. Cette étape est rendue possible, car toutes les données nécessaires sont disponibles : les coréférences, la chronologie, et les structures syntaxiques locales.

La Figure 2 présente un exemple de graphe événementiel dans le domaine des instructions pour utiliser un logiciel. Elle est suivie des textes générés en français et en anglais à partir de ce graphe.

```
E1 := Instruction_Gen [
      le_but   => E2
      le_moyen => E3 ]

E2 := Entrer [
      objet => TOK1 ]

E3 := Succession [
      prem_event => E4
      deux_event => {E5, E6} ]

E4 := Ouvrir [
      patient => TOK2 ]

E5 := Obligation [
      objet => E7 ]

E7 := Remplir [
      objet => TOK3
      moyen => TOK4 ]

E6 := Opposition [
      prem_event   => E8
      event_opposé => E9 ]

E8 := Obligation [
      objet => E10 ]

E10 := Remplir [
      objet => TOK5
      moyen => TOK6 ]

E9 := Interdiction [
      objet => E11 ]

E11 := Remplir [
      objet =>
        {TOK7, TOK8, TOK9} ]

TOK1 := Info_Supplèmentaire [ ]

TOK2 := Fenêtre [
      son_nom =>
        "OPERATION LOG DETAIL" ]

TOK3 := Champ [
      légende          =>1
      son_nom =>"PROBLEM ID" ]

TOK4 := Numéro_Identificateur [
      de_quoi => TOK10 ]

TOK5 := Champ [
      légende          =>5
      son_nom =>"SUMMARY" ]

TOK6 := Info_Supplèmentaire [ ]

TOK7 := Champ [
      son_nom =>"ENTRY DATE" ]

TOK8:= Champ [
      son_nom =>"ENTRY TIME" ]

TOK9:= Champ [
      son_nom =>"ENTRY NUMBER" ]

TOK10 := Probleme [ ]
```

Figure 2 : exemple d' un graphe événementiel de Flaubert

*Pour entrer une information supplémentaire, effectuez la procédure suivante : ouvrez la fenêtre OPERATION LOG DETAIL. Puis, vous devez remplir le champ PROBLEM ID (champ 1) avec un numéro identificateur d' un problème. De plus, vous devez remplir le champ SUMMARY (champ 5) avec une information supplémentaire. Cependant, il est interdit de remplir le champ ENTRY DATE, le champ ENTRY TIME et le champ ENTRY NUMBER.*

*In order to enter a piece of additional information, open the OPERATION LOG DETAIL window. Then, fill the PROBLEM ID field (field 1) with an identificator number of a problem. Finally, fill the SUMMARY field (field 5) with a piece of additional information. However, do not fill the ENTRY DATE field, the ENTRY TIME field and the ENTRY NUMBER field.*

Flaubert est implémenté en ADA (Meunier 1997) avec la participation de la société CORA et l' aide financière du MENESR.

## 3. Interface entre SPIN et Flaubert

Pour interfacer SPIN et Flaubert, le point d' arrêt qui s' impose est la structure sémantique. Il s' agit donc de faire correspondre la structure sémantique de SPIN et un graphe événementiel de Flaubert comme dans le tableau suivant.

```
[...]
es structure
------------
element semantique: titre
contenu:
programmer(obj:enregistrement,man:
    par une touche incrémentielle)

element semantique: option
contenu: meilleure qualité d'image
         régler(obj:sélecteur de
vitesse de bande,dest:SP)

element semantique: op_seq
contenu: régler(obj:sélecteur de
vitesse de bande,dest:SP)

element semantique: op_seq
contenu: sélectionner(obj:canal 4)

element semantique: op_seq
contenu: appuyer(dest:touche OTR)
[...]
```

```
E0 := Instruction [
   titre            => E1
   séquence         => E2 ]

E1 := Titre [ action => E11 ]

E11 := Programmer [
   objet   => Enregistrement
   avec    =>
    Par_Une_Touche_Incrémentielle ]

E2 := Séquence [
   premier  => E21
   deuxième => E22 ]

E21 := Option [
   but      =>
    Meilleur_Qualité_D_Image
   manière  => E211 ]

E22 := Séquence [
   premier  => E221
   deuxième => E222 ]

E221 := Sélectionner [
   objet    => Canal_4 ]

       ...
```

Lorsque nous avons voulu automatiser cette correspondance, nous nous sommes heurtés à divers problèmes que l' on peut classifier en "détails techniques" et "difficultés théoriques", les premiers relevant de choix d' implémentation différents mais non insurmontables, les seconds étant plus graves.

### 3.1 Détails techniques

Les principales difficultés ont porté sur les points suivants:

**Modélisation du domaine** (représentation conceptuelle dans SPIN et attributs typés dans Flaubert) : les objets du domaine sont appelés dans G-TAG entités, leur représentation est déduite de la structure de traits typés `Entité`. Dans SPIN, ces objets sont soit des chaînes de caractères (au sens informatique, c' est-à-dire que les objets n' ont pas de représentation formelle), soit des prédicats simples (par exemple on trouve `cassette` et `video` pour l' utilisation d' un magnétoscope). Les opérations du domaine sont appelées dans G-TAG événements simples et leur représentation descend de la structure de traits typés `Event_Simple`. Dans SPIN, les opérations sont toutes représentées par un prédicat argument du prédicat op, dans le fichier des schémas d' opérations.

**Structures linéaires *vs* arborescentes** : la structure sémantique de SPIN se présente comme une liste linéaire bien qu' elle devrait théoriquement être arborescente (Kosseim 1995). Le

graphe événementiel de Flaubert est arborescent ce qui permet de calculer récursivement un schéma global du texte stylistiquement et syntaxiquement bien formé. Le passage automatique d' une liste linéaire à une structure arborescente se passe bien dans la majorité des cas mais il reste quelques cas problématiques.

**Découpage prédicats-arguments-modifieurs** : les efforts de SPIN étant orientés vers le cognitif et non la linguistique, certains éléments de la structure sémantique sont très proches de la réalisation de surface ce qui évite des calculs linguistiques, par exemple *sur-une-touche-incrémentielle*. A l' inverse, Flaubert orienté vers la linguistique repose sur un découpage rigoureux entre prédicats, arguments et modifieurs qui permet de calculer par exemple pour un groupe prépositionnel la préposition, le déterminant, le nom tête et les modifieurs.

### 3.2 Difficultés théoriques

Prendre comme point d' arrêt la structure sémantique de SPIN demande que l' équivalent des opérations "choix d' une structure rhétorique" et "choix d' une forme lexicale" de SPIN soit effectué dans Flaubert lors du choix d' un connecteur. Rappelons par exemple que dans SPIN la relation EFFET se traduit en une relation rhétorique de but, manière ou résultat et qu' à chaque relation rhétorique est associée un ou deux connecteurs, tandis que dans Flaubert la relation EFFET est directement associée à un ensemble de connecteurs. Pour choisir une relation rhétorique traduisant un EFFET, SPIN s' appuie sur des critères comme "effet désirable et voulu" ou non. Considérons les textes suivants :

(1) *Pour éteindre la radio, tourner le bouton VOLUME complètement à gauche.*

(2) *Eteignez le moteur de la voiture, la radio s' éteindra.*

Ils sont tous deux obtenus dans SPIN à partir d' une relation EFFET, mais la relation rhétorique but - associée au connecteur *pour* - est chosie pour (1) car l' effet est désirable et voulu, tandis que la relation résultat - associée à une juxtaposition des deux phrases séparées par une virgule - est choisie pour (2) car éteindre le moteur de sa voiture n' est pas la méthode normale pour éteindre la radio.

Pour obtenir le même résultat dans Flaubert, il faut ajouter des attributs comme "est-désirable" dans la définition de EFFET :

```
EFFET < event-binaire [ action ==> event-simple,
                        effet ==> event-simple,
                        est-désirable ==> oui/non]
```

afin de tester la valeur de cet attribut pour choisir le bon connecteur. Ce processus n' est pas dans la philosophie de Flaubert et il aboutit à une floppée de tests pour le choix d' un connecteur qui alourdit considérablement le processus de génération. La solution dans Flaubert consiste à ne pas considérer que les textes (1) et (2) relèvent tous deux de la relation EFFET mais disons que le premier relève de INSTRUCTION - associée entre autres au connecteur *pour* - et le second de EFFET-DE-BORD - associée entre autres au connecteur virgule.

Cette différence de philosophie entre SPIN et Flaubert est le reflet d' une question qui reste sans réponse aujourd' hui dans la communauté scientifique travaillant tant dans le domaine de la représentation des connaissances que dans celui du traitement automatique du langage, question illustrée de la façon suivante : est-il justifié de considérer que les textes (1) et (2) relèvent de la même relation conceptuelle ? Il est hors de question de trancher sur cette

question ici, mais on espère avoir montré que le caractère totalement arbitraire des représentations sémantico-conceptuelles utilisées à l' heure actuelle pose des problèmes dramatiques lorsqu' on cherche à interfacer deux systèmes de génération de textes.

## 4 Bilan et perspectives

Les échanges franco-québecois ont permis à chaque équipe de compléter leurs connaissances, et ce, d' une manière tout à fait bénéfique puisqu' à travers la confrontation à un problème concret, c.-à-d. l' interface entre SPIN et Flaubert.

Un début d' implémentation de cette interface a été réalisé au cours de la première année du projet. Cette implémentation n' est pas poursuivie dans la seconde année car il nous apparaît que l' on ne pourra arriver à un prototype qu' avec un trop grand nombre d' "astuces" sans fondement théorique. Par contre, les efforts de la seconde année seront orientés vers une question d' évaluation : pour réaliser un produit industriel de génération d' instructions, est-il plus opportun de partir d' un système comme SPIN comprenant un planificateur de tâches, ou d' un système comme Flaubert où la planification des tâches est encapsulée dans l' entrée du générateur, ou encore d' un système hybride à définir ? Cette question met en jeu des aspects comme la taille du système, les connaissances demandées en entrée, sa transportabilité, sa convivialité, et sa capacité à produire des textes en plusieurs langues.

## Bibliographie


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.

Bouayad-Agha, N., 1996, Flaubert o SPIN = GRABIG, Rapport TALANA-SCRIPTUM n°2.

Danlos, L., 1996, "Présentation de G-TAG, un formalisme pour la génération de textes", *Actes de la 3ème conférence sur le Traitement Automatique du Langage Naturel* TALN-96, Marseille.

Danlos, L., Meunier, F., 1996, "G-TAG, un formalisme pour la génération de textes : présentation et applications industrielles", *Actes du colloque Informatique et Langue Naturelle*, Nantes.

Delaunay, M.P., 1996a, "Grammaire d' analyse versus grammaire de génération : comparaison pour les grammaires d' arbres adjoints", Mémoire de DEA, Université Paris 7.

Delaunay, M. P., 1996b, Flaubert o SPIN = GRABIG, Rapport TALANA-SCRIPTUM n° 3.

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.

Kosseim, L., 1995, *Planification de textes d' instructions : sélection du contenu et de la structure rhétorique*, PhD Thesis, DIRO, Université de Montréal.

Kosseim, L., Laplame, G.,1995, "Choosing Rhetorical Relations in Instructionals Texts: the case of Effects and Guidances", in *Proceedings of the 5th European Workshop on Natural Language Generation*, Leiden, Hollande.

Meunier, F., 1996, Flaubert o SPIN = GRABIG, Rapport TALANA-SCRIPTUM n° 1.

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