Université Paris 7
Laboratoire d'Automatique Documentaire et Linguistique

Composition d'automates linguistiques

Philippe Laval

Vendredi 16 octobre 1992

"Combien de fois ai-je dit que, une fois
éliminées toutes les impossibilités,
l'hypothèse restante, AUSSI
IMPROBABLE QU'ELLE SOIT, doit être
la bonne !"

(Discours de S. Holmes dans Le signe des Quatre.)

Remerciements

Je remercie avant tout le professeur Maurice Gross pour avoir rendu cette thèse possible. Mes remerciements vont également à la société CORA qui m'a permis de travailler dans un cadre agréable et stimulant au cours de ces trois années.

De nombreuses idées de cette thèse sont apparues lors de discussions longues et passionnées avec mes collègues et amis, en particulier Paul Duquennoy et Pascal Pellegrini sur le plan informatique et Alain Guillet, Béatrice Pelletier et Morris Salkoff sur le plan linguistique. Ma gratitude leur est acquise.

Enfin, je n'aurais pas pu accomplir ce travail sans le soutien constant et chaleureux de ma femme Ilona.

Sommaire

Notations

Les notations suivantes sont utilisées dans le reste du texte :

Adj	Adjectif
Adv	Adverbe
Dét	Déterminant
N	Nom. Si besoin est, des chiffres en indice indiquent le placement du substantif dans les constructions :
	N_0 sujet
	N_1 premier complément
	N_2 second complément
	On peut également spécifier des classes de substantif à droite de l'indice, par exemple :
	N_{hum} substantif humain
Poss	Déterminant possessif.
Ppv	Particule préverbale
Pro	Pronom
V	Verbe conjugué
V-inf	Verbe à l'infinitif

Introduction

Aujourd'hui, l'informatique est de plus en plus présente dans la vie de l'entreprise et l'on peut penser que, après la révolution de l'informatique, nous allons assister à la révolution de l'information.

Après les bases de données classiques (hiérarchique, navigationnelle, relationnelle, etc.) qui permettent de manipuler des données structurées (bilan, inventaire, facturation, etc.), il devient nécessaire de gérer le texte.

En effet, les informations textuelles représentent une part non négligeable de la mémoire de l'entreprise, que ce soit en tant que source d'information interne : statut, catalogue, compte rendu de réunion... ou bien en tant que source d'information externe : journaux, contrats, appels d'offre...

Les traitements automatiques à effectuer sur ces textes sont multiples :

— Indexation et archivage intelligent.

— Résumé.

— Traduction complète ou partielle.

— Vérifications variées : orthographique, syntaxique, stylistique, voire sémantique ou pragmatique (vérification de cohérence par exemple).

— Routage de messages — il s'agit de systèmes qui décident du destinataire d'un texte en fonction de son contenu.

— Commande en langage naturel de systèmes automatisés.

— Etc.

Ces applications ne sont pas toutes réalisables à l'heure actuelle, mais il ne fait pas de doute qu'elles le seront un jour, même si elles ne s'appliquent qu'à des domaines restreints (comme la traduction par exemple). Elles ne peuvent toutefois être réalisées que s'il est possible d'effectuer une analyse syntaxique des textes à traiter. Une telle analyse dépend évidemment des objectifs : un programme d'indexation automatique nécessite une analyse moins fine qu'un programme de traduction automatique.

L'analyse automatique est donc un point de passage obligatoire pour réaliser des programmes de traitement du langage. Cependant, les analyseurs automatiques existants sont difficiles à exploiter de manière industrielle pour trois raisons :

— Manque de possibilités d'évolution : les analyseurs sont des programmes très complexes et il est généralement difficile de les faire évoluer dans le temps.

— Manque de fiabilité : en général, un analyseur ne peut fournir d'analyses partielles pour une phrase qu'il ne peut traiter entièrement, que ce soit parce qu'elle est agrammaticale ou simplement non prévue par la grammaire de l'analyseur.

— Lenteur d'exécution : cette lenteur est due d'une part à toutes les ambiguïtés présentes dans les textes et d'autre part à des choix technologiques discutables pour des produits industriels.

Ces problèmes ne nous ont pas amené à concevoir un nouvel analyseur car nous n'aurions sans doute pu faire mieux, mais plutôt à imaginer un système qui facilite le travail d'analyse en utilisant des règles locales — donc faciles à manipuler — pour diminuer le nombre d'ambiguïtés lexicales des textes.

Cette thèse décrit ce système nommé Tomas.

Les deux premiers chapitres présentent quelques problèmes spécifiques à l'analyse automatique dont Tomas doit faciliter la résolution. Le chapitre 1 ("L'analyse automatique") introduit la notion de "règles négatives" tandis que le chapitre 2 ("Expressions figées et analyse automatique") traite de l'analyse des expressions figées.

Le troisième chapitre ("Présentation de Tomas") introduit le système Tomas. Ce système est détaillé dans les cinq chapitres suivants. Enfin, le dernier chapitre ("Applications de Tomas") décrit d'une part les résultats obtenus et d'autre part les applications de Tomas.

I. L'analyse automatique

Le problème de l'analyse automatique est très complexe, suffisamment pour nécessiter une démarche progressive qui facilite au maximum le travail d'analyse en résolvant préalablement les problèmes les plus simples.

Parmi ceux-ci, la levée des ambiguïtés lexicales (homographies) peut être résolue à peu de frais en exploitant les régularités locales des phrases. Nous verrons toutefois qu'un tel travail est plus facile à réaliser si l'on décrit les constructions impossibles au lieu de décrire celles qui sont possibles.

Cette stratégie, dite des «règles négatives», sera celle utilisée pour ce projet. Nous examinerons dans la fin de ce chapitre comment situer ce formalisme par rapport aux autres formalismes existants.

1. Les difficultés de l'analyse automatique.

L'objectif d'un analyseur est d'identifier les structures syntaxiques (par exemple : GN, GV, Qu-P ...) d'une phrase. Ceci se fait en général en examinant les catégories syntaxiques des mots qui constituent la phrase et en les combinant à partir d'une «grammaire» pour former des structures plus générales qui peuvent elle-mêmes se combiner pour former d'autres structures, jusqu'à obtenir une description complète de la phrase.

De tels analyseurs sont très complexes à réaliser car ils doivent décrire totalement tous les phénomènes qui peuvent advenir dans une phrase. A titre indicatif, Morris Salkoff a montré qu'il faudrait environ un milliard de règles pour écrire une grammaire context-free du français [Salkoff 79][1]. Il n'est pas forcément nécessaire de faire une description extensive de ces règles : elles peuvent tout à fait être générées à la volée au cours de l'analyse, par exemple à partir d'un mécanisme d'unification (comme en Prolog) portant sur certains traits syntaxiques (comme le genre, le nombre ou la personne).

Réelles ou virtuelles, un tel nombre de règles amène l'analyseur à générer un grand nombre d'hypothèses en cours d'analyse, en partie à cause du nombre élevé d'ambiguïtés lexicales qui peuvent survenir dans une phrase.

Il est par exemple symptomatique de constater qu'un analyseur peut mettre moins de temps pour produire la première analyse d'une phrase que pour déterminer si cette analyse est la seule, ce qui montre bien qu'il doit parcourir de nombreuses hypothèses pour chaque phrase.

Enfin, aucun analyseur — industriel tout au moins — ne peut traiter de séquences agrammaticales, même dans le cas où des analyses partielles existent.

Face à ces phénomènes, il semble possible d'améliorer nettement les performances de tels analyseurs en levant préalablement à leur utilisation les homographies présentes dans le texte.

---

1. Ceci nous fournit une borne supérieure dans la mesure où tout autre formalisme plus concis que les grammaires CF générera moins de règles.

Afin d'éviter de reproduire la complexité des analyseurs, nous allons tenter de nous limiter à des règles locales, c'est-à-dire des règles qui n'utilisent que le contexte immédiat[2] du mot en cours de traitement.

a. Règles locales.

L'utilisation de règles locales permet de gérer simplement les régularités locales de la langue. Par exemple, après une particule préverbale, on trouve forcément un verbe ou une autre particule préverbale; de même, après un déterminant, on trouve un quantifieur, un adverbe, un adjectif ou un nom. Ce type de règle est tout à fait adapté au traitement de séquences agrammaticales : les phrases pourront ainsi être analysées localement même s'il n'est pas possible de construire une représentation globale.

Toutefois, l'utilisation de telles règles présente quelques difficultés, en particulier :

— La gestion des conflits entre règles.

— L'application des règles.

Gestions des conflits entre règles.

En théorie, il ne devrait pas y avoir de conflits entre les règles. Toutefois, dans le cas où, à la suite d'une erreur, deux règles contradictoires sont applicables, le résultat risque de dépendre de l'ordre d'application de ces règles, ce qui est particulièrement dangereux. Ceci n'est cependant pas le cas le plus général. Par exemple, on peut considérer les deux règles suivantes :

1 Un homographe pronom/déterminant suivi d'un verbe est un pronom

2 Un homographe nom/verbe précédé d'un déterminant est un nom.

Ces règles semblent correctes mais en fait elles ne prennent pas compte des homographies pouvant porter sur le verbe (première règle) et sur le déterminant (deuxième règle). Dans le cas de séquences comportant une double homographie, l'ordre d'application des règles modifiera le résultat obtenu. En effet, pour les séquences suivantes :

Dét/Pro	*Nom/Verbe*
le	*juge*
la	*classe*
les	*tables*

l'application de la règle 1 puis de la règle 2 analysera les séquences comme un pronom suivi d'un verbe, alors que les règles appliquées dans l'ordre contraire analyseront ces séquences comme un déterminant et nom.

2. Par immédiat, nous entendons un nombre fini, connu à l'avance et petit de mots à gauche ou à droite.

Application des règles.

La deuxième difficulté est beaucoup plus sérieuse. En effet, l'utilisation de règles de description des séquences possibles implique une prise d'hypothèse lors de leur application.

Considérons la règle qui indique qu'un déterminant est forcément suivi par un quantifieur, un adverbe, un adjectif ou un nom. Cette règle possède trois interprétations, qui ne sont pas toutes exactes linguistiquement, à savoir :

(1) Le mot qui suit un déterminant est forcément un quantifieur, un adverbe, un adjectif ou un nom.

(2) Le mot qui précède un nom, un adjectif un adverbe ou un quantifieur est forcément un déterminant.

(3) Les deux interprétations (1) et (2) à la fois.

Le plus simple est de considérer que seule l'interprétation (1) doit être retenue. Même dans ce cas, l'application d'une telle règle reste difficile. Considérons par exemple la séquence

... la table ...

et essayons de lui appliquer cette règle. Nous sommes obligé de générer trois analyses :

— Soit *la* est un déterminant et *table* est un nom.

— Soit *la* est un pronom et , en l'absence d'autres règles, *table* est soit un nom, soit un verbe conjugué.

L'application de cette règle a réduit le nombre d'analyses potentielles de la séquence, puisqu'il est passé de quatre analyses :

Dét V

Dét N

Pro V

Pro N

à trois analyses :

Dét N

Pro V

Pro N

Cependant, la gestion des multiples analyses dues à l'application de toutes ces règles est difficile à réaliser et risque de pénaliser la levée des homographies du point de vue de la complexité de l'algorithme et donc du temps d'analyse.

Pour éviter cet écueil, on se rend compte qu'il faut écrire des règles de plus en plus longues avec le risque de perdre l'aspect local de la description.

Une règle est une suite de conditions terminée par un point. Chaque condition est entourée de crochets droits ([]) et est constituée d'une suite de contraintes séparées par des virgules. Une contrainte peut avoir deux formes :

—Positive : Nom de catégorie = (Listes de valeurs)

CAT=(NOM,ADJ)

Si la contrainte est de la forme Nom de catégorie = (Une seule valeur), elle peut être abrégée sous la forme Valeur :

Les contraintes CAT=(ADV), CAT=ADV *et* ADV *sont équivalentes.*

—Négatives : Nom de catégorie /= (Listes de valeurs)

TPS /=(IN,PPR)

Une condition peut être facultativement suivie par une deuxième condition qui porte sur les nœuds homographes. Cette deuxième condition est comprise entre des parenthèses et peut être précédée d'un signe plus (+) ou d'un signe moins (-). Si elle est précédée d'un plus, cela signifie qu'il y a au moins un nœud qui doit vérifier cette deuxième contrainte. S'il elle est précédée d'un moins, cela signifie qu'aucun nœud ne doit vérifier cette deuxième contrainte.

Par exemple, la contrainte :

[NOM](+[ADJ]) désigne un nom homographe avec un adjectif;

et la contrainte :

[NOM](-[ADJ]) désigne un nom non homographe avec un adjectif.

Voici quelques exemples de règles :

[NOM,S] désigne un nom singulier :

chat, ordinateur

[VER,TPS/=IN] désigne un verbe qui n'est pas à l'infinitif :

joue, acceptaient

[NOM] (+[ADJ]) désigne un nom qui est aussi un adjectif :

vieux, gros

[DET][VER] désigne un déterminant et un verbe en séquence

le juge, la frappe

Les abréviations utilisées pour les noms de catégories et les valeurs sont fournies dans les tableaux des deux pages suivantes.

FIGURE 1 - FORMALISME DES REGLES DE TOMAS.

Catégorie générale CAT	
NOM	Nom commun
ADJ	Adjectif
VER	Verbe sauf les participes passés
ADV	Adverbe
DET	Déterminant
INT	Interjection
PRE	Préposition
PRO	Pronom
SMC	Segment de mot composé
PPS	Participe passé
QUA	Quantifieur
CSU	Conjonction de subordination
CCO	Conjonction de coordination
NPR	Nom Propre

Fonction CAS	
SUJ	Sujet
COD	Objet direct
COI	Objet indirect

Nombre NBR	
S	Singulier
P	Pluriel
SP	Singulier ou pluriel

Personne PER	
JE	Première personne du singulier
TU	Deuxième personne du singulier
IL	Troisième personne du singulier
NOUS	Première personne du pluriel
VOUS	Deuxième personne du pluriel
ILS	Troisième personne du pluriel

Genre GEN	
M	Masculin
F	Féminin
MF	Masculin ou féminin

FIGURE 1 (SUITE) CATEGORIES ET ABREVIATIONS

Temps TPS	
PI	Indicatif présent
II	Indicatif imparfait
FI	Indicatif futur
PS	Passé simple
PC	Conditionnel présent
SUP	Subjonctif présent
SUI	Subjonctif imparfait
IM	Impératif
IN	Infinitif
PPR	Participe présent

Sous-catégorie SCA	
PER	Personnel
POS	Possessif
REF	Réfléchi
NUM	Numérique
INTER	Interrogatif
REL	Relatif
DEM	Démonstratif
ARD	Article défini
ARI	Article indéfini
IND	Indéfini

FIGURE 1 (SUITE ET FIN) CATEGORIES ET ABREVIATIONS

Les deux difficultés précédentes nous ont amené à utiliser une autre stratégie : celle des règles négatives. Le principe est simple : il s'agit de décrire les séquences impossibles, c'est-à-dire les séquences agrammaticales, puis de les détruire automatiquement. De cette manière, il n'y a plus de problème d'interprétation ou d'application des règles. En effet, si l'on utilise une structure en graphe[3] de la phrase, nous verrons que l'application des règles négatives se résume à la simple destruction de chemins à l'intérieur de ce graphe.

b Règles négatives et contexte local.

Examinons maintenant comment des règles négatives permettent de prendre facilement en compte le contexte local. En fait, on s'aperçoit que dans ces règles négatives, le contexte fait partie de la règle de manière intrinsèque. Nous allons présenter deux exemples qui permettent de mieux comprendre ce point.

L'accord dans le groupe nominal.

Il s'agit d'interdire les séquences composées de noms, d'adjectifs, de déterminants et de quantifieurs qui ne respectent pas les règles d'accord en genre et en nombre.

Pour indiquer qu'un déterminant, un quantifieur ou un adjectif s'accorde avec le mot qui le suit, on peut utiliser la règle suivante :

[CAT=(DET,QUA,ADJ),NBR=S] {ADV}
[CAT=(NOM,ADJ,QUA),NBR=P].[4]

Cette règle indique seulement qu'une séquence constituée d'un déterminant, d'un quantifieur ou d'un adjectif (CAT=(DET,QUA,ADJ)) au singulier (NBR=S) suivi d'un adverbe facultatif ({ADV}) et d'un nom, d'un adjectif ou d'un quantifieur (CAT=(QUA,NOM,ADJ)) au pluriel (NBR=P) est agrammaticale[5]. Remarquons que cette règle permet de gérer l'accord jusqu'au nom, c'est-à-dire dans des séquences du type[6] :

Dét Qua Adj* Nom* (avec éventuellement des insertions d'adverbes[7])

---

3. Cette structure en graphe de la phrase sera présentée en détail dans le chapitre "Structure de la phrase".
4. Le formalisme d'écriture des règles est présentée succinctement dans la figure 1 et détaillé dans le chapitre "Configuration de Tomas".
5. Les autres cas de non-accord (Pluriel-Singulier, Masculin-Féminin et Féminin-Masculin) sont traités par des règles identiques.
6. Le signe * utilisé ici correspond au signe de répétition des grammaires formelles. Il signifie que le symbole auquel il est appliqué peut être répété un nombre quelconque (éventuellement nul) de fois.
7. La règle supporte des insertions d'adverbes dans toute la séquence, ce qui est incorrect puisqu'ils ne peuvent être insérés que devant l'adjectif. Ceci n'est toutefois pas gênant puisque ces règles sont négatives et ne sont pas utilisés en production.

De manière à propager l'accord à l'intérieur du groupe nominal, il reste à accorder le nom avec l'adjectif ou le nom qui le suit. Ceci pourrait être fait avec une règle comme :

[CAT=NOM,NBR=S] [CAT=(NOM,ADJ),NBR=P].

Ces deux règles vérifient alors l'accord pour des séquences du type :

Dét Qua (Adj* Nom*)** (avec éventuellement des insertions d'adverbes)

Comme par exemple :

Les trois petits hommes verts

La très jolie jeune femme

Le nouvel ordinateur Apple

On observe ainsi que l'on peut gérer ce problème d'accord — à l'exclusion de la conjonction — avec uniquement huit règles[8].

Remarquons toutefois que le problème se complique dans le cas de mots composés. Par exemple, pour des mots composés de type N de N [Gaston Gross 86,90], [Pelletier 91] comme :

un chemin de ronde

une mine de sel

des bas de soie

les règles précédentes sont erronées. En effet, les séquences suivantes :

... un chemin de ronde tortueux ...

... une mine de sel abandonnée ...

... des bas de soie transparents ...

sont syntaxiquement correctes bien que l'adjectif final ne soit pas accordé en genre et/ou en nombre avec le mot qui le précède. Il s'agit cependant de mots composés et nous verrons dans le chapitre suivant que la meilleure solution est de ne pas appliquer de règles à l'intérieur de ces expressions.

Cet exemple montre donc que les conditions d'application d'une règle se confondent avec la règle elle-même, c'est-à-dire qu'une règle spécifie à la fois une impossibilité syntaxique et la séquence à éliminer. Nous allons examiner maintenant comment prendre en compte les homographies.

Les homographies Nom/Adjectif.

Le problème est de déterminer dans une séquence d'homographes nom ou adjectif lesquels sont réellement des noms et lesquels sont réellement des adjectifs. Prenons par exemple les séquences suivantes :

(1a) ... *une vieille femme* ...

(2a) ... *un homme vieux* ...

8. En effet, les deux règles précédentes sont multipliées par quatre pour traiter les quatre combinaisons en genre et en nombre.

ces deux séquences possèdent respectivement les structures :

(1b) ... *Dét Adj N* ...

(2b) ... *Dét N Adj* ...

Or les séquences (1) et (2) ne diffèrent que par les homographies qu'elles contiennent :

vieille = N + Adj et homme = N

femme = N et vieux = N + Adj

Il est donc nécessaire de pouvoir préciser les homographies dans les règles négatives afin de pouvoir traiter ce type de problème. C'est l'autre manière de prendre le contexte en compte.

Les règles négatives pour l'homographie nom/adjectif ont alors la forme suivante :

Règle 1 : [NOM] (+[ADJ]) [NOM] (-[ADJ]).

Règle 2: [[NOM] (-[ADJ]) [NOM] (+[ADJ]).

Ces deux règles interdisent d'analyser une succession de deux mots comme deux noms dans le cas où au moins l'un des deux est également un adjectif. En d'autres termes, ces deux règles permettent de choisir l'analyse où apparait au moins un adjectif.

Ces deux règles ne traitent bien évidemment qu'une petite partie du problème des homographies N/Adj. Grâce à ce formalisme, une dizaine de règles ont pus être écrites[9], qui, associées à une partition des adjectifs en «adjectifs pouvant se placer avant le nom» et en «adjectifs ne pouvant pas se placer avant le nom», permettent dans la majorité des cas de lever ce type d'homographie.

Ces deux exemples nous ont permis de montrer les deux manières d'utiliser le contexte local avec des règles négatives. Nous allons maintenant tenter de situer les règles négatives par rapport aux autres formalismes d'analyses.

2. Les règles négatives par rapport aux autres formalismes

Nous allons d'abord examiner les règles négatives par rapport aux langages formels définis par Chomsky avant de voir si d'autres analyseurs ont déjà utilisé ce type de règles.

a. Règles négatives et langages formels.

Rappelons que pour nous une règle négative est simplement la description d'une séquence impossible.

Si l'on considère comme Gross et Lentin ([Gross et Lentin 67]) que :

9. Ces règles ont été écrites par Alain Guillet dans le cadre d'un projet de la société CORA.

> "Le but d'une grammaire est donc de donner des règles qui permettent de construire un mot bien formé (une phrase correcte) ou de reconnaître si une séquence de lettres (de mots) est ou non un mot bien formé (une phrase correcte)."

On voit que les règles négatives constituent bien une grammaire. En effet, pour vérifier qu'une séquence de mots est bien formée, il suffit de vérifier qu'aucune règle négative ne s'applique, c'est-à-dire que la séquence ne contient pas de suite de mots interdite. Par contre, les règles négatives ne peuvent pas être utilisées de manière pratique en génération car le seul moyen de générer des phrases bien formées serait de produire toutes les séquences de mots — correctes ou non — et d'éliminer ensuite celles qui ne seraient pas bien formées. Tout ceci est bien évidement théorique dans la mesure où, pour décrire toutes les phrases du français et uniquement les phrases du français avec des règles négatives, il faudrait énumérer toutes les séquences impossibles du français...

Le pouvoir d'expression des règles négatives est faible puisqu'elles ne sont même pas capables de reconnaître un langage de type $a^n b^n$. Par contre, ce formalisme facilite la description d'un grand nombre de régularités locales qui, combinées, permettent de lever beaucoup d'homographies. Remarquons également que grâce à l'indépendance des règles négatives, l'extension de la grammaire par ajout d'un nouveau phénomène (comme par exemple la prise en compte du caractère transitif des verbes) influe peu sur les règles existantes, contrairement aux formalismes basés sur des règles de réécriture. Enfin, comme nous l'avons déjà dit, ces règles peuvent fonctionner localement même si la phrase est agrammaticale.

b. Utilisation antérieure des règles négatives

Il est toujours délicat de dire quand un concept est utilisé pour la première fois. Toutefois, si l'on se réfère à [Sabah 88 et 89] et à [Miller et Torris 90], il semble qu'un tel formalisme n'ait jamais été utilisé tel quel dans le passé. Les restrictions de la grammaire en chaîne sont sans doute ce qui s'en approche le plus, comme nous le montre cet extrait de [Salkoff 77] :

> "Une restriction a généralement la forme suivante : si la classe X a la valeur X_i, alors une autre classe Y, qui lui est liée grammaticalement doit avoir (ou ne pas avoir) la valeur Y_j"

Toutefois, les restrictions dans la grammaire en chaîne servent à valider ou à invalider le choix d'une chaîne, alors qu'ici les règles négatives servent uniquement à invalider des séquences qui donneraient lieu à des analyses incorrectes.

Il n'est pas très étonnant de constater que les règles négatives n'ont jamais été utilisées précédemment. En effet, ce type de règles ne permettrait pas de réaliser un analyseur complet dans la mesure où elles n'identifient pas les structures syntaxiques de la phrase : elles ne servent qu'à diminuer le nombre d'analyses potentielles.

Conclusion

Il semble donc que le formalisme des règles négatives, qui est utilisé ici pour la première fois, soit tout à fait adapté à la réalisation d'un programme de levée d'homographie. Un tel programme simplifierait considérablement la tâche d'analyseurs plus complexes en éliminant les analyses incorrectes et en fournissant une analyse partielle des séquences agrammaticales.

Au cours de ce chapitre, nous avons vu que les règles régissant l'accord en genre et en nombre posaient des problèmes en présence de mots composés. Nous allons examiner ce phénomène d'une manière plus systématique dans le chapitre suivant.

II. Expressions figées et analyse automatique

Les "expressions figées" sont des expressions qui ne sont pas — ou pas totalement — régies par les principes de compositionalité[10]. Voici quelques exemples d'expressions figées :

un cercle vicieux

prendre le taureau par les cornes

un poisson volant

un va-et-vient

Dans ce chapitre, nous nous intéresserons particulièrement aux expressions figées dans le domaine de l'analyse automatique et nous montrerons que l'identification de certaines expressions figées est une condition préalable à une analyse automatique correcte du langage.

1. L'analyse des expressions figées.

Les expressions figées sont présentes dans tous les types de textes de manière non négligeable, il suffit d'examiner n'importe quel article de journal pour s'en convaincre.

En ce qui concerne ces expressions, deux types d'analyses s'opposent : ne pas en tenir compte, c'est-à-dire les traiter comme des expressions syntaxiques normales, ou au contraire les reconnaître et les traiter à part.

Suivant le type d'expression, l'une ou l'autre de ces analyses est pertinente. Pour en juger, nous allons dresser une typologie des expressions figées en fonction des besoins de l'analyse automatique.

---

10. Rappelons que les principes de compositionalité sont au nombre de quatre :
    - Un mot est une séquence de lettres entre deux séparateurs.
    - Un mot possède une catégorie syntaxique (Nom, Verbe, Adverbe, etc.)
    - Ces catégories permettent de définir des constituants syntaxiques à partir de séquences :

    GN = Dét Adj N
    P = GN GV

    - Le sens d'un constituant est fonction du sens de ses composants.

2. Une typologie des expressions figées.

Notre objet n'est pas de classifier les expressions figées selon leurs propriétés syntaxiques, ce qui a déjà été fait dans [Danlos 80, 81, 88], [Gaston Gross 86, 88] et [Maurice Gross 82a, 88, 90] entre autres, mais plutôt de fournir une typologie qui permette de savoir à quel moment une expression figée doit être traitée lors de l'analyse (cette question a déjà été abordée dans [Silberztein 89] et dans [Laporte 88]).

a. Syntaxe normale *vs* syntaxe «exotique».

Le premier critère de sélection sera syntaxique : il s'agit de distinguer entre les expressions qui ont des constructions syntaxiques normales de celles qui ont des constructions syntaxiques «exotiques». Nous appelerons «exotique» une construction syntaxique qui ne se retrouve jamais en dehors d'une expression figée. Ceci peut être dû à la présence d'un "mot" qui n'existe pas dans des expressions libres, comme :

*au **fur** et à mesure*

*un **prud'**homme*

aujourd'hui

*être à **jeun***

a priori[11]

ou bien parce qu'insérées dans des phrases, elles produisent toujours des séquences de catégories syntaxiques agrammaticales[12] si elles ne sont pas analysées comme des expressions figées.

Par exemples, les séquences suivantes sont agrammaticales si on les considère mot par mot :

— * Dét V ...

un va-et-vient

un pousse rapière (apéritif)

un boit sans soif

un étouffe chrétien

— * Dét à N

un à-coup

un à coté

— * Dét sur N

le sur-moi

11. Ce *a* n'est pas une forme conjuguée du verbe *avoir* mais vient du latin *ab.*
12. Rappelons que le critère du trait d'union n'est pas valable : sur les 30.000 mots composés N ADJ recensés à ce jour, une centaine seulement prend obligatoirement un trait d'union [Mathieu-Colas 88].

— * Dét contre N

un contre amiral

la contre-réforme

h Ambiguë *vs* certaine

Parmi les expressions syntaxiquement normales, nous pouvons encore distinguer celles qui sont ambiguës de celles qui sont certaines[13]. En effet, certaines séquences de mots peuvent être figées ou non suivant le contexte. Par exemple, les séquences :

entrée en fonction

au moins

sont figées dans les phrases :

Max a fait une entrée en fonction remarquée

Au moins, Max n'a pas souffert

et libres dans :

Cette information est entrée en fonction de sa validité

Le contrat est attribué au moins cher des soumissionnaires

De la même manière, les séquences :

casser Poss^0 [14] *pipe*

une pomme de terre

peuvent être prises ou non dans leur sens libre (c'est-à-dire dans leur sens littéral) :

Max a cassé sa pipe = Max est mort (sens figé)

Max a cassé sa pipe = Max a brisé sa pipe (sens libre)

Max a vu une pomme de terre = Max a vu une patate (sens figé)

Max a vu une pomme de terre = Max a vu une pomme en terre (sens libre)

Remarquons que ces ambiguïtés sont sémantiques alors que les ambiguïtés précédentes étaient d'ordre syntaxique, nous reviendrons plus loin sur cette distinction.

13. Par "certaine", nous entendons non ambiguë...
14. La notation *Poss^0* est utilisée pour indiquer un déterminant possessif dont la personne est la même que celle du verbe. Par exemple : *Il casse sa pipe*, mais pas *Il casse notre pipe.*

A l'inverse des expressions comme :

à Poss^0 propre jeu :
Max s'est fait prendre à son propre jeu
annuaire électronique :
L'annuaire électronique est disponible sur le 11
plus ou moins :
Il est plus ou moins question de partir.

sont certaines dans la mesure où seule l'interprétation figée est acceptable, quelle que soit la phrase que nous avons été capable d'imaginer. De telles expressions peuvent donc être identifiées à coup sûr.

c. Conclusion sur la typologie.

Nous obtenons ainsi une partition en trois classes des expressions figées :

— Expressions syntaxiquement «exotiques».
— Expressions syntaxiquement correctes ambiguës.
— Expressions syntaxiquement correctes certaines (*ie* syntaxiquement non ambiguës).

Ces trois classes ne sont pas homogènes et ne présentent sans doute que peu d'intérêt du point de vue purement linguistique. Par contre, nous allons voir dans la suite que cette classification est pertinente pour décider quand et comment doivent être analysées les expressions figées.

3. Analyse des expressions.

Nous allons examiner pour chaque classe d'expression la manière dont elle doit être analysée.

a. Expressions syntaxiquement «exotiques».

Du fait de leur syntaxe, ces expressions ne peuvent être analysées directement. En effet, si la structure d'une expression figée pouvait être reconnue par l'analyseur, celui-ci serait alors susceptible d'analyser incorrectement d'autres séquences, qui ne sont pas elles-mêmes figées, mais qui sont calquées sur la structure de ces expressions figées. Evidemment, l'analyseur peut n'accepter une séquence agrammaticale que si elle fait partie d'une expression figée, mais ce cas est équivalent à une reconnaissance préalable de l'expression.

Par exemple, supposons qu'un analyseur traite la séquence *V N* comme un *N* à cause de l'expression *pousse rapière*, dans ce cas, une phrase comme :

Le juge donne la mesure à ses élèves

risque de fournir l'analyse incorrecte :

Dét V N Pro V ...

au lieu de la bonne analyse :

Dét N V Dét N ...

A l'opposé, si l'analyseur considère qu'une telle séquence est impossible, il ne pourra pas générer une analyse complète de phrases contenant de telles expressions.

Il semble donc que ce type d'expression doive être identifié avant l'analyse, de manière à isoler l'analyseur de la structure syntaxique de ces expressions.

b Expressions syntaxiquement correctes certaines.

Ce type d'expression peut être traité normalement lors de l'analyse dans la mesure où elles respectent la syntaxe normale. Toutefois, elles peuvent être reconnues avant l'analyse car elles ne sont pas ambiguës et apportent donc une information supplémentaire à peu de frais.

En effet, la reconnaissance préalable d'une expression telle que :

plus ou moins

évite d'avoir à considérer les ambiguïtés lexicales des mots *plus*, *ou* et *moins* de l'expression.

Pour ces expressions, une reconnaissance préalable n'est donc pas obligatoire mais s'avère profitable — si les expressions ont été préalablement répertoriées — pour la suite de l'analyse.

c. Expressions syntaxiquement correctes ambiguës.

Pour ce type d'expression, il faut encore affiner notre classification de manière à distinguer entre les expressions ambiguës syntaxiquement et les expressions ambiguës sémantiquement.

En effet, une séquence comme :

casser Poss^0 pipe

ne peut être désambiguïsée sans faire intervenir des critères sémantiques puisqu'une phrase comme :

Max a cassé sa pipe

possède deux analyses, comme nous l'avons vu précédemment. A l'opposé, une séquence comme :

au moins

peut être désambiguïsée par des critères purement syntaxiques :

Au moins = Adv dans *Au moins, Max survivra*

Au moins = Prep Dét Adv dans *Le contrat est attribué au moins cher des soumissionnaires*

Expressions ambiguës sémantiquement

La reconnaissance de ce type d'expression est inutile pour une analyse purement syntaxique et ne devrait intervenir que lors de l'analyse sémantique du texte (si une telle analyse existe). En effet, si l'on considère une séquence ambiguë comme *pomme de terre*, il est clair qu'une identification au niveau syntaxique ne lève aucun ambiguïté dans la mesure où l'on ne peut pas savoir s'il s'agit d'une expression figée ou d'une expression littérale sans faire intervenir des critères sémantiques ou pragmatiques.

Expressions ambiguës syntaxiquement

Ces séquences doivent être identifiées comme des expressions potentielles tout en laissant à l'analyseur le rôle de choisir entre l'interprétation figée ou non de la séquence.

Remarquons que la décision de classer une expression comme ambiguë syntaxiquement ou sémantiquement dépend du type d'analyse effectuée.

En effet, il peut sembler que l'ambiguïté d'une séquence comme *col blanc* est d'ordre sémantique alors qu'elle en fait syntaxique. Si l'on place cette séquence dans une phrase, la sélection du verbe peut permettre de décider s'il s'agit d'une expression figée ou d'une séquence libre. Par exemple, dans la phrase

Les cols blancs n'en appeleront pas au droit de grève.

le verbe *en appeler à* n'acceptant guère que des sujets humains[15], seule l'interprétation figée est valide syntaxiquement (ce critère est toutefois à la limite de la syntaxe).

Conclusion

Nous avons donc fourni une typologie des expressions figées en vue de leur traitement lors d'une analyse automatique. Il est remarquable que dans trois cas sur quatre la reconnaissance des expressions figées facilite — voire conditionne — une bonne analyse.

Ceci implique qu'un analyseur efficace doit être capable d'une part d'identifier les expressions figées et d'autre part de gérer des ambiguïtés structurelles provenant de l'analyse d'une séquence comme une expression figée ou comme une suite de mots simples.

15. Ce verbe se trouve dans la table 33 de [Boons, Guillet, Leclère 76] et est indiqué comme prenant uniquement un sujet humain : il possède un plus dans la colonne "N_0=:N_{hum}" et un moins dans la colonne "N_0=:N_{-hum}".

III. Présentation de Tomas

L'objectif de ce chapitre est d'une part de présenter le projet Tomas et d'autre de part de montrer le lien entre le sujet de cette thèse, "composition d'automates linguistiques" et ce projet. Ce chapitre est divisé en trois parties, la première est une présentation rapide de Tomas, la seconde en montre les aspects novateurs. Enfin, la troisième donne l'état d'avancement du projet.

1. Tomas : Qu'est ce ?

Tomas est un système de levée d'homographie dont le but est de supprimer ou tout au moins de diminuer le nombre des ambiguïtés lexicales présentes dans une phrase et de repérer les expressions figées qui pourraient s'y trouver. De plus, Tomas effectue une partition de la phrase en propositions de manière à pouvoir appliquer simplement des règles négatives.

Tomas est fondé sur quatre principes :

① Se limiter au niveau syntaxique de l'analyse.
② Utiliser des règles locales sans chercher à identifier les structures globales de la phrase.
③ Isoler chaque phase du traitement de manière à obtenir des composants logiciels modulables à volonté.
④ Séparer les connaissances linguistiques du traitement informatique.

Revenons plus en détail sur chacun de ces quatre principes afin de les justifier et de mieux appréhender leurs conséquences.

a. Rester au niveau syntaxique.

A l'heure actuelle, il n'existe pas de description sémantique opérationnelle de la langue qui serait applicable à n'importe quel domaine. A l'inverse, la syntaxe d'une langue est largement indépendante du domaine (au problème près du vocabulaire spécialisé).

Le but de Tomas est d'arriver à une couverture des textes aussi large que possible, sans toutefois prendre en compte des textes littéraires ou poétiques. Un objectif raisonnable serait de pouvoir traiter les textes d'un quotidien généraliste.

Ceci implique que les lexiques utilisés doivent être les plus complets possibles afin de pouvoir étiqueter efficacement tous les mots rencontrés : c'est pourquoi le lexique des mots simples de Tomas est le DELAF dans son intégralité, soit un dictionnaire électronique de 800.000 formes. Pour les expressions figées, Tomas utilise déjà des listes développées par la société CORA (listes d'expressions du logiciel DARWIN) et intégrera à terme une partie des listes de mots composés (DELAC) du LADL.

b. Utiliser des règles locales

L'utilisation de règles locales permet de bien cerner les phénomènes linguistiques : l'idéal serait d'avoir une règle pour chaque phénomène. Ceci offre quatre avantages :

- — Identification claire des phénomènes réguliers et des exceptions.
- — Lisibilité accrue de la grammaire constituée par l'ensemble des règles.
- — Modularité et réemploi aisés.
- — Mise au point et maintenance facilitées : chaque règle, de par son caractère local, a un domaine d'application précis et n'interfère pas avec les autres règles.

Le problème qui surgit immédiatement si l'on veut employer de nombreuses règles locales est celui de l'application de ces règles. Pour être réellement locales, ces règles doivent être indépendantes les unes des autres. C'est là que Tomas rejoint la composition des automates linguistiques : si l'on assimile chaque règle à un automate, le problème est bien de les composer. Enfin, afin d'obtenir un pouvoir expressif suffisant, ces règles utilisent le contexte dans lequel elles s'appliquent. Ce contexte est d'ailleurs fourni par la règle elle-même.

c. Isoler chaque phase du traitement.

Chaque phase du traitement doit être isolée de manière à obtenir non seulement un système de levée d'homographie en état de marche, mais aussi des outils (ou des composants logiciels) qui pourront être réutilisés dans d'autres projets[16]. Ceci est d'autant plus aisé que Tomas est développé en ADA. Tirant ses racines de Pascal, ADA est le premier langage industriel incorporant des caractéristiques importantes telles que l'abstraction des données, le multi-tâche, le traitement des exceptions, les paquetages et les génériques. Ce langage a ceci d'original qu'il a été défini dans les moindres détails avant d'être réalisé, ce qui a permis d'en faire un langage réellement utilisable dans le domaine du génie logiciel. Ceci nous a conduit à d'implanter chaque module de Tomas sous forme d'une tâche qui communique avec les autres modules par des canaux bien définis.

16. C'est d'ailleurs déjà le cas à la société CORA avec le projet "TRANSPOSITION" de Béatrice Pelletier, Morris Salkoff et Pascal Pellegrini.

Actuellement, Tomas est divisé en six modules :

① Analyse lexicale du texte.
② Etiquetage des mots.
③ Première reconnaissance des expressions figées et construction de la structure en graphe[17] de la phrase.
④ Découpage en propositions.
⑤ Expansion des homographies et des mots contractés
⑥ Levée des homographies (en appliquant les règles locales) et reconnaissance des expressions semi-figées.

d. Séparation des données linguistiques.

Le but est évident : il faut éviter de mélanger les problèmes linguistiques et les problèmes informatiques. En particulier, il ne faut pas que les linguistes qui écrivent les règles soient pénalisés par des problèmes de recompilation des programmes, de gestion du retour-arrière ("backtrack" en anglais) ou de temps d'accès aux lexiques.

Pour cela, toutes les données linguistiques ont été séparées des programmes. Ces données sont spécifiées par le linguiste dans des formalismes simples et clairs, que ce soit pour les lexiques (formalisme proche de celui du DELAF) ou pour les règles syntaxiques dont le formalisme est cohérent avec les lexiques.

2. Tomas : Quoi de nouveau ?

Il existe déjà de nombreux systèmes d'analyse syntaxique et il peut sembler inutile d'en développer d'autres. Toutefois, Tomas se distingue par quatre points :

— Une couverture très large de la langue (DELAF : 800.000 formes).
— Une reconnaissance efficace des expressions figées.
— Une structure originale pour représenter la phrase et ses ambiguïtés : la structure en graphe.
— Des règles locales qui sont uniquement des règles négatives.

Ce sont ces trois derniers points que nous allons aborder maintenant.

a. La reconnaissance des expressions figées.

Comme nous l'avons vu précédemment, une reconnaissance efficace des expressions figées est indispensable si l'on veut obtenir une analyse correcte de la langue.

17. La structure en graphe de la phrase, qui est explicitée dans la page suivante, permet de représenter simplement les homographies de la phrase.

Toutefois, compte tenu du nombre d'expressions figées qui est vraisemblablement de l'ordre de plusieurs centaines de milliers, il faut absolument disposer d'une méthode efficace d'identification de ces expressions. En effet, la méthode immédiate qui consiste à construire toutes les expressions possibles de deux mots contigus ou plus et à vérifier leur existence dans un lexique de formes figées s'avère en pratique extrêmement pénalisante, comme l'exemple suivant le montre.

Prenons la phrase suivante :

Le cordon bleu cuisine.

Avec cette méthode, on voit qu'il faut effectuer 6 recherches dans le lexique pour être sûr que seule la séquence *cordon bleu* est potentiellement une expression figée (d'une manière plus générale, pour une séquence de n mots de long, une telle méthode implique $n(n-1)/2$ recherches pour identifier toutes les expressions figées potentielles).

Tomas utilise une méthode plus efficace[18] qui permet de diminuer le nombre d'accès au lexique. De plus, cette méthode supporte les insertions de mots — comme par exemple des adverbes — à l'intérieur des expressions.

Il reste toutefois deux autres types d'expressions qui ne sont pas encore traitées :

— Les expressions du type *casser Poss^0 pipe* où la personne du pronom *Poss^0* doit être accordée avec le sujet. Pour ce type d'expression, il est envisageable de les reconnaître comme des expressions figées sans contraintes en plaçant dans le lexique des règles de bonne formation.

— Les expressions discontinues : *pousser Nhum à bout*. Pour ces expressions, la difficulté est de définir les limites gauche et droite du groupe nominal *Nhum* qui est de longueur indéfinie. A l'heure actuelle, Tomas ne reconnaît pas ces expressions.

b La structure en graphe de la phrase.

Dans une phrase, il existe deux sortes d'homographies : les homographies qui portent sur des mots isolés : *joue* (Nom) ou *joue* (Verbe) et les homographies qui portent sur des expressions :

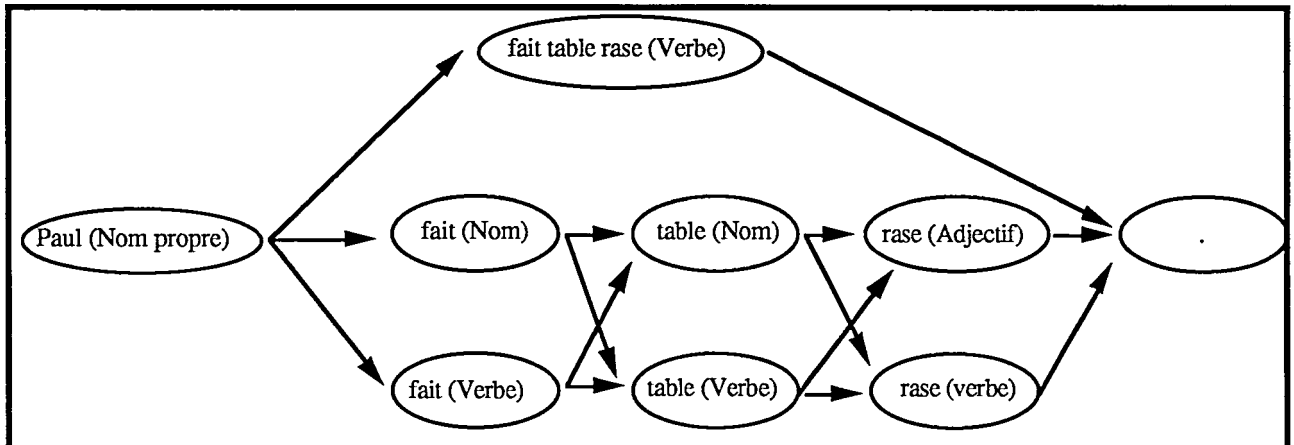
faire table rase

cette séquence peut être un verbe complexe ou une suite de trois mots :

(Verbe ou Nom[19]) (Verbe ou Nom) (Verbe ou Adjectif).

18. Cette méthode est présentée en détail dans le chapitre VI.
19. Selon le dictionnaire Lexis, *faire* peut être nom qui désigne "Le genre la manière de faire d'un écrivain, d'un artiste ; son mode d'exécution : *Des faires tout différents.*

Pour représenter ces ambiguïtés, Tomas construit un graphe orienté de tous les chemins possibles parmi les ambiguïtés de la phrase. L'idée de construire un graphe pour représenter une phrase a déjà été utilisée dans [Silberztein 89], toutefois le principe était alors de construire une expression rationnelle représentant la phrase alors que Tomas utilise un graphe complexe dans le même but. Ces deux approches sont théoriquement équivalentes, il faut tout de même remarquer que dans notre formalisme, les propositions et les expressions figées sont représentées comme des nœuds du graphe.



Paul fait table rase.

FIGURE 1 -UN GRAPHE DE PHRASE

Dans ce graphe, les nœuds peuvent être des mots simples, des expressions figées ou des sous-graphes représentant des propositions.

Une fois ce graphe construit, les règles locales permettent de supprimer les chemins impossibles, Tomas se chargeant du parcours du graphe et de l'application des règles.

Grâce à cette méthode, les règles s'appliquent sur des données sans ambiguïté. En effet, si l'on a pris le chemin :

Paul (Npropre) fait (V) table (N)

Il est inutile de savoir que *fait* peut être aussi un nom ou une partie d'expression.

La structure en graphe permet d'éviter de parcourir tous les chemins: si on interdit la séquence :

(V) (V)

La suppression du sous-chemin :

fait (V) table (V)

permet de détruire deux sous-chemins qui n'auront donc pas à être explorés par la suite, à savoir les chemins :

fait (V) table (V) rase (V)
fait (V) table (V) rase (Adj)

Une fois la deuxième séquence de type *(V)(V)*, *table* (Verbe) *rase* (Verbe) supprimée, deux chemins disparaissent (les chemins supprimés apparaissent en gras) :

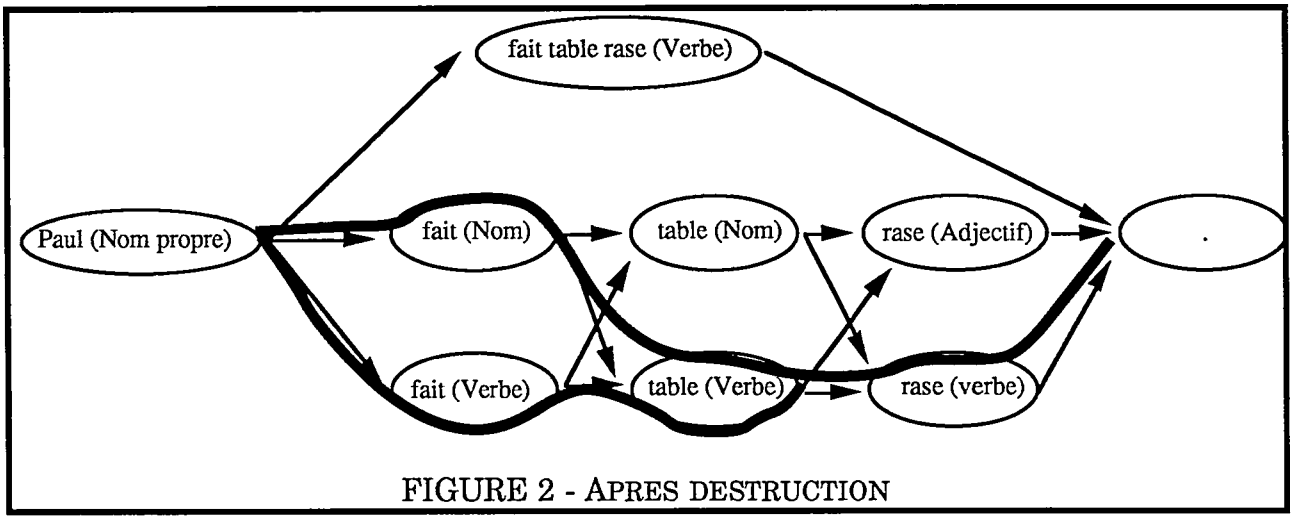


FIGURE 2 - APRES DESTRUCTION

Ceci nous amène tout naturellement au point suivant :

c. Des règles exclusivement négatives

Dans une structure en graphe de la phrase, il est très difficile de définir des critères de validité d'un chemin. De plus, ceci impliquerait de parcourir tous les chemins pour décider de leur validité. Enfin, l'aspect local des règles disparaîtrait puisque la validité doit être établie pour un chemin dans sa globalité.

A l'inverse, des règles identifiant des parties de chemin impossibles sont d'une part parfaitement locales, c'est-à-dire contextuelles, et permettent d'autre part d'éliminer — comme nous l'avons vu précédemment — plusieurs chemins d'un coup.

Enfin, des règles de ce type sont indépendantes les unes des autres.

3. Tomas : Où en est-on ?

Les modules ① (analyse lexicale), ② (étiquetage),③ (expressions figées et constructions de la phrase), ⑤ (expansion) et ⑥ (application des règles locales et expressions semi-figées) sont déjà opérationnels.

Le lexique utilisé est le DELAF pour les mots simples et des listes propres à CORA pour les expressions figées.

Le module ⑥ peut encore être amélioré en lui ajoutant :

— Des règles pour les expressions du type *casser Poss^0 pipe*. Ces règles seront associées au lexique des expressions figées.

— Des mécanismes de récupération des séquences agrammaticales. En effet, dans le cas où le système est confronté à une phrase incorrecte syntaxiquement, une application stricte des règles supprime tous les chemins du graphe. Dans ce cas, il serait intéressant de disposer d'un mécanisme qui permette de repérer la ou les erreurs de syntaxe afin d'analyser les parties correctes de la phrase..

Le module ④ (découpage en propositions) est encore en cours de réalisation, la principale difficulté étant de définir un formalisme — s'il existe — qui permette d'exprimer simplement les règles utilisées pour identifier les différentes propositions. A l'heure actuelle, ce découpage est effectué directement par une série de procédures en Ada.

Ce module est indispensable pour l'analyse des phrases complexes. En effet, les règles locales ne peuvent s'appliquer efficacement que sur des propositions simples (voir par exemple la règle interdisant deux verbes conjugués successifs).

Conclusion

Il peut sembler surprenant d'aboutir à un tel système en partant simplement d'un problème de composition d'automates linguistiques. C'est toutefois logique dans la mesure où il était difficile de concevoir un tel mécanisme "dans le vide" : chaque problème en amenant un autre, chaque solution imposant ses contraintes, c'est ainsi que Tomas est né.

IV. Structure de la Phrase

1. Introduction

L'unité de traitement à l'intérieur de Tomas est la phrase, c'est-à-dire que Tomas considère un texte comme une succession de phrases et que chaque module (analyse lexicale, étiquetage, etc.) sera appliqué à une phrase entière.

Il apparaît donc que la manière dont Tomas représente une phrase joue un rôle très important pour ses capacités d'analyse.

La représentation d'une phrase doit satisfaire aux quatre critères suivant :

Premier critère : La représentation doit contenir toutes les ambiguïtés de la phrase, en particulier :
- — les homographies
- — les expressions ambiguës
- — les différents découpages en propositions de la phrase.

Deuxième critère : La représentation doit permettre d'éviter l'explosion combinatoire liée à la succession d'ambiguïtés. Rappelons qu'un exemple d'école comme *Le juge juge le juge* génère potentiellement 864 (ou 2 x 6 x 6 x 2 x 6) analyses.

Troisième critère : La représentation doit permettre de développer progressivement les ambiguïtés. En effet, il est parfois inutile de considérer toutes les ambiguïtés d'une entrée lexicale. Par exemple, il est inutile de distinguer entre les noms et les adjectifs lors de la phase de découpage de la phrase en propositions.

Quatrième critère : La représentation doit permettre de visiter les différentes analyses potentielles de la phrase de trois manières différentes :
- — En profondeur : il s'agit de visiter chaque analyse linéairement.
- — En largeur : il s'agit de visiter toutes les analyses simultanément en considérant toutes les homographies de chaque graphie.

— De manière aléatoire : il s'agit de visiter toutes les homographies de la phrase sans tenir compte de l'ordre des graphies dans la phrase (ce type de parcours est inutile pour une analyse syntaxique de la phrase. Toutefois, il peut être utiliser dans des applications extra-linguistiques, par exemple pour recenser tous les verbes ou tous les noms dans une application documentaire).

Ces trois modes de parcours sont présentés dans la figure 1 (ci-dessous).

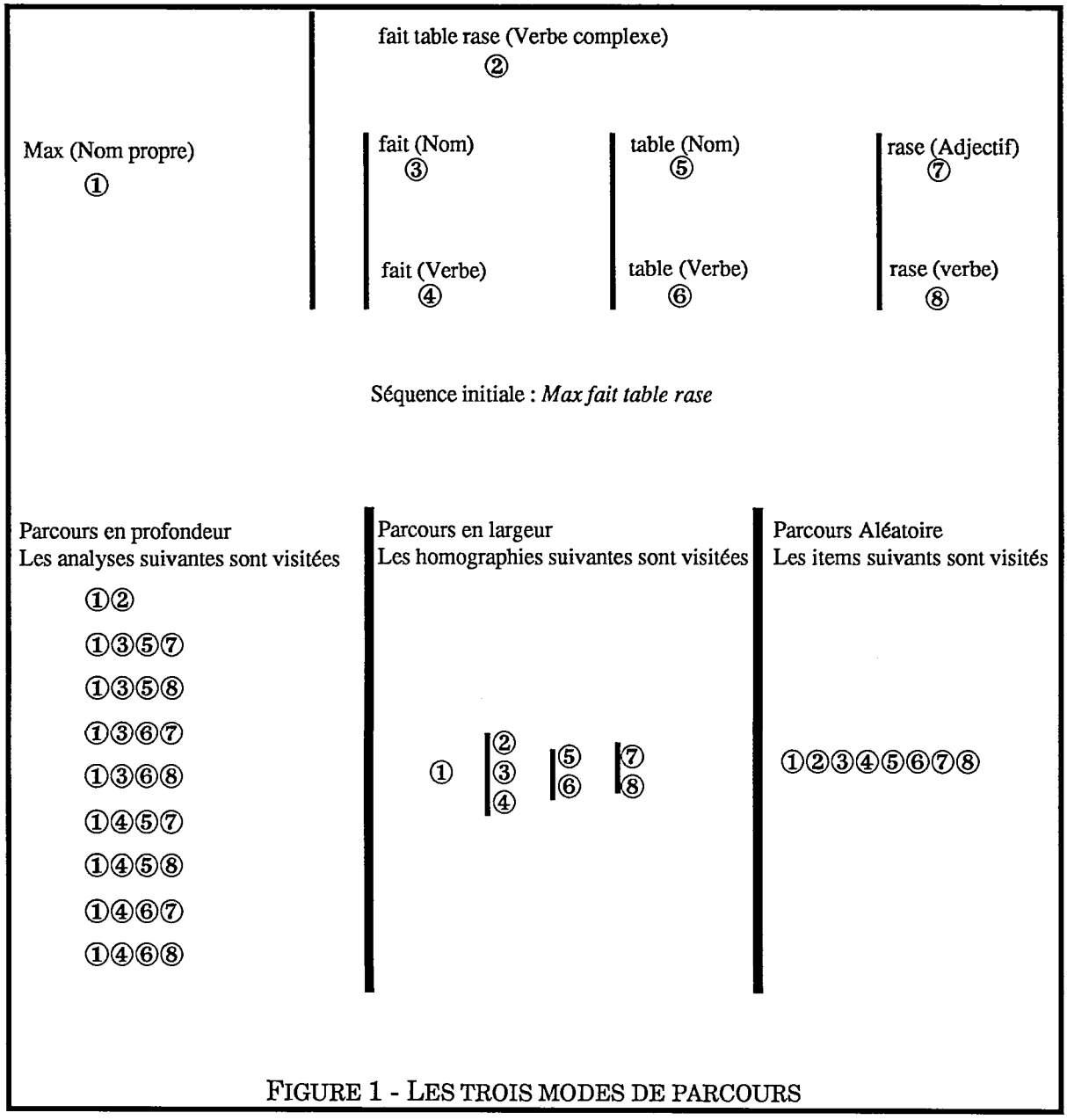


FIGURE 1 - LES TROIS MODES DE PARCOURS

2. Le graphe de la phrase

a. Le graphe simple

Pour satisfaire ces quatre critères, nous avons choisi de représenter une phrase par un graphe orienté acyclique.

Rappelons qu'un graphe est constitué d'un ensemble de nœuds et d'un ensemble d'arcs qui relient certains de ces nœuds entre eux. Un graphe orienté est un graphe dans lequel chaque arc possède une source et une destination, de sorte que l'arc reliant le nœud A au nœud B n'est pas le même que celui reliant le nœud B au nœud A (cet arc peut ne pas exister). Un graphe acyclique est un graphe qui ne contient pas de cycle, un cycle étant un ensemble d'arcs successifs tel que la destination du dernier arc soit la source du premier. Il existe un cas particulier de graphe orienté acyclique, le treillis, qui nous intéressera dans la suite. Un treillis est un graphe orienté acyclique tel que pour tout couple de nœuds du graphe, il existe toujours un nœud qui les précède l'un et l'autre et un nœud qui les suit l'un et l'autre.

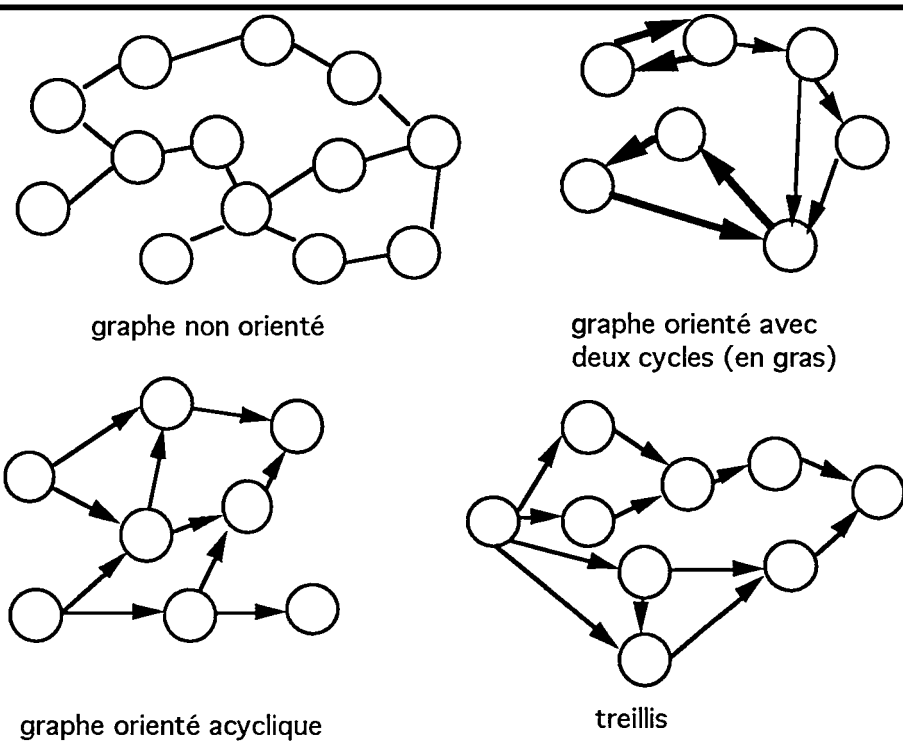


FIGURE 2 - EXEMPLES DE GRAPHES

La figure 2 présente quatre types de graphe. Tous ces graphes ne sont pas utiles pour la linguistique; par exemple, les graphes non orientés peuvent servir à modéliser des réseaux de télécommunication ; chaque nœud correspond alors à un central téléphonique. Les graphes orientés constituent une analogie du système nerveux : les nœuds représentent les neurones et les arcs les connections synaptique. On utilise les graphes orientés acycliques

pour les réseaux sémantiques ; chaque nœud correspond alors à un concept et les arcs symbolisent des relations entre ces concepts. Enfin, nous utiliserons les treillis pour notre graphe de phrase ; dans ce cas, les nœuds représentent les entrées lexicales et les arcs illustrent simplement la relation de succession dans la phrase.

Par exemple, la phrase *Max fait table rase* peut être représentée par le graphe de la figure 3. Remarquons que notre graphe contient deux nœuds (les nœuds 1 et 2) qui ne sont présents qu'implicitement dans la phrase, à savoir un nœud indiquant le début de la phrase (nœud 1 sur la figure 3) et un nœud indiquant la fin de la phrase (nœud 2 sur la figure 3).

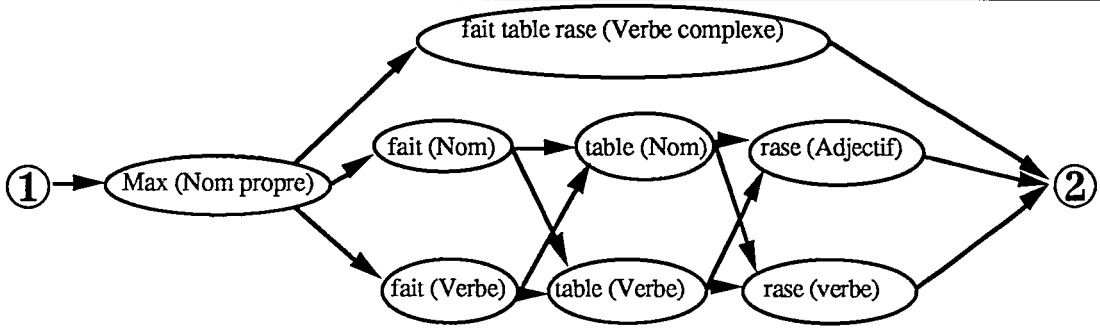


FIGURE 3 - EXEMPLE DE GRAPHE DE PHRASE

Ceci nous permet de préciser la définition de notre graphe de phrase : il s'agit d'un graphe orienté acyclique possédant deux nœuds particuliers, à savoir :

— Un noeud où aucun arc n'arrive et qui est connexe à tous les autres nœuds du graphe (c'est-à-dire qui est relié par un chemin orienté à tous les autres nœuds du graphe). Ce nœud est le "nœud initial" du graphe et correspond au début de phrase.

— Un nœud d'où aucun arc ne part et à qui tous les autres nœuds du graphe sont connexes (c'est-à-dire que tous les autres nœuds du graphe lui sont reliés par un chemin orienté). Ce nœud est le "nœud final" du graphe et correspond à la fin de la phrase.

Remarquons immédiatement que, muni de ces deux nœuds, notre graphe de phrase est en fait un treillis.

Les deux définitions nous permettent de préciser la notion d'analyse potentielle d'une phrase : une analyse potentielle est un chemin orienté continu partant du nœud initial du graphe et aboutissant au nœud final du graphe.

En résumé, la représentation d'une phrase est un graphe orienté acyclique possédant un nœud initial et un nœud final, tel qu'à chaque chemin orienté continu reliant le nœud initial au nœud final corresponde une unique analyse potentielle de la phrase, et qu'à chaque analyse potentielle de la phrase corresponde un unique chemin orienté continu reliant le nœud initial au nœud final[20].

b Le graphe complexe

Jusqu'à présent, nous avons supposé qu'à un nœud du graphe correspondait une entrée lexicale simple. Toutefois, ceci ne répond ni au premier critère (représentation des propositions), ni au troisième (développement plus ou moins complet des homographies). Ceci nous amène donc à étendre la définition d'un nœud du graphe :

Un nœud peut être constitué soit par plusieurs entrées lexicales homographes, soit par un graphe représentant lui-même une proposition — remarquons d'ailleurs que certains nœuds de ce graphe peuvent également contenir des propositions, ce qui illustre les possibilités d'enchâssement des propositions dans la langue. Enfin, les expressions figées, une fois identifiées, sont également représentées comme un nœud unique.

Dans la suite, nous distinguerons deux types de graphes :

— Les graphes simples, dont aucun nœud n'est lui-même un graphe.

— Les graphes complexes, dont certains nœuds peuvent être des graphes.

c. Définition de quelques termes

A ce stade, il semble nécessaire de définir certains termes qui seront utilisés dans la suite :

• Chemin	Il s'agit d'une succession d'arcs, telle que la destination de chaque arc soit la source de l'arc suivant.
• Chemin complet	Il s'agit d'un chemin qui relie le nœud initial du graphe au nœud final.
• Sous-chemin	Un sous-chemin C' d'un chemin C est un chemin C' tel que tous les arcs de C' fassent partie de C.
• Source d'un chemin	Il s'agit du nœud source du premier arc du chemin.
• But d'un chemin	Il s'agit du nœud destination du dernier arc du chemin.

20. En d'autres termes, il existe une bijection entre l'ensemble des analyses potentielles de la phrase et l'ensemble des chemins orientés continus du graphe reliant le nœud origine au nœud final.

• Antécédent d'un nœud	L'antécédent d'un nœud A est un nœud B tel qu'il existe un arc reliant le nœud B au nœud A.
• Successeur d'un nœud	Le successeur d'un nœud A est un nœud B tel qu'il existe un arc reliant le nœud A au nœud B.

d. Construction et transformation du graphe

A l'intérieur de Tomas, le traitement d'un graphe peut être divisé en deux phases :

— La première phase, dite "phase de construction" .

— La deuxième phase, dite "phase de transformation".

La phase de construction

Lors de cette phase, Tomas construit le graphe à partir du flot de texte qui lui est fourni en entrée et des différents lexiques pertinents. Ce processus fournit un graphe simple fortement connecté, c'est-à-dire que tous les nœuds de rang n sont connectés à tous les nœuds de rang $n+1$, sauf pour le nœud final du graphe C'est également pendant cette phase que sont identifiées les expressions figées continues.

La figure 3 donne un exemple d'étiquetage pour la phrase *Max fait table rase.*

La phase de transformation

Cette phase, beaucoup plus variée que la précédente, regroupe toutes les opérations qui, à partir du graphe simple fortement connecté issu de la phase précédente, permettent d'obtenir un graphe complexe avec un petit nombre d'analyses potentielles.

Cette phase se décompose en trois modules :

— Découpage de la phrase en propositions, ce qui fournit une phrase complexe.

— Expansion des mots contractés (comme *aux* ou *des*).

— Levée des homographies par identification des chemins impossibles.

Ces deux phases s'appuient sur un certain nombre d'opérations de base applicables à un graphe ou à une partie de graphe. Nous allons expliciter ces opérations dans la suite.

3. Les opérations sur le graphe

a. L'insertion

En phase de construction

Durant la phase de lecture du texte, des nœuds sont ajoutés au graphe au fur et à mesure de l'analyse de la phrase.

Notre mode de lecture favorisant l'analyse de la phrase de la gauche vers la droite, c'est cette procédure qui est retenue dans Tomas. Lors de la construction du graphe de phrase, il faut distinguer entre l'insertion d'un mot simple et l'ajout d'une expression figée.

L'insertion d'un mot simple s'appuie sur son rang dans la phrase, en attribuant arbitrairement le rang 2 au premier mot de la phrase (le rang 1 correspond au nœud initial du graphe). Chaque nœud possède donc un rang qui sera utilisé lors de la construction pour relier les nœuds entre eux. Comme nous l'avons déjà dit, l'analyse va dans le sens des rangs croissants.

Dans cet esprit, pour ajouter un mot simple de rang n comportant h homographies, il faut :

- Créer un nœud pour chaque homographie, donc h nœuds au total.
- Relier tous les nœuds de rang n-1 aux h nœuds créés précédemment.

Un exemple d'insertion simple est fourni dans la figure 4.

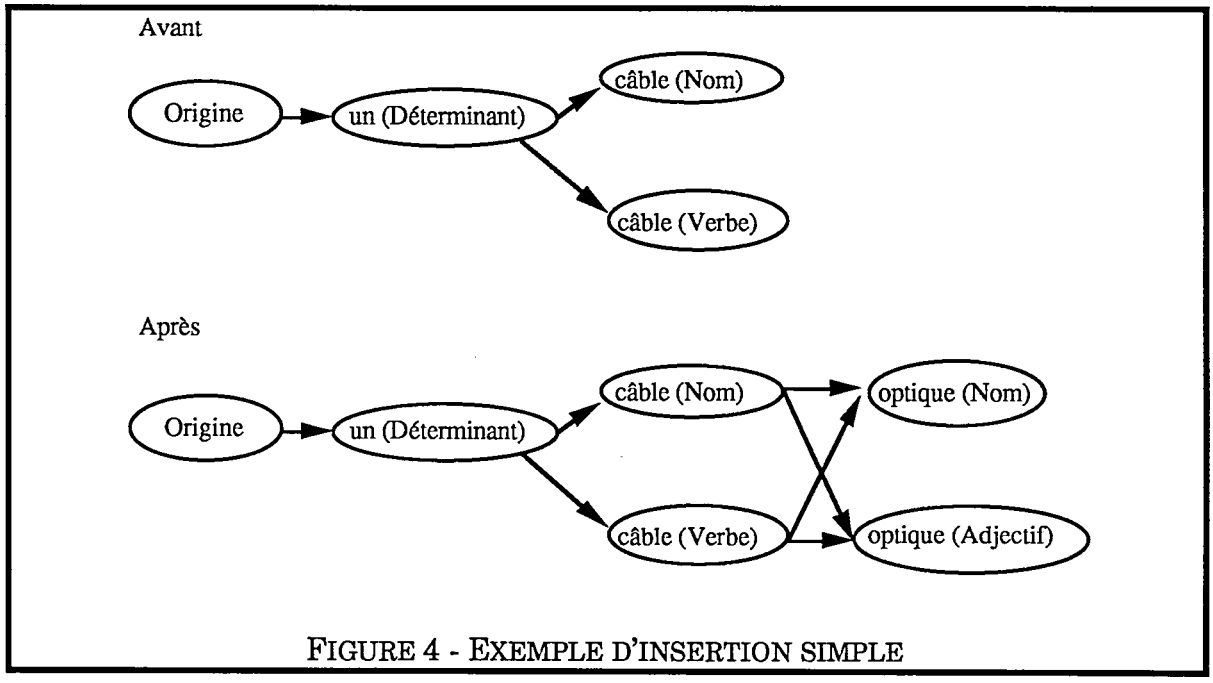


FIGURE 4 - EXEMPLE D'INSERTION SIMPLE

A l'opposé, une expression figée ne pouvant être reconnue qu'a posteriori, une fois tous ses lexèmes étiquetés, le mécanisme d'ajout est légèrement différent. Pour ajouter une expression figée continue allant du mot n_1 au mot n_2 (inclus) et comportant h homographies, il faut :

- Créer h nœuds représentant toutes les homographies de l'expression.
- Relier tous les nœuds de rang n_1-1 au h nœuds créés précédemment.
- Considérer pour la suite les h nœuds créés comme des nœuds de rang n_2 (c'est-à-dire que ces nœuds seront reliés à tous les nœuds de rang n_2+1).

De plus, dans le cas d'une expression figée certaine, il faut détruire les chemins parallèles provenant des étiquetages des lexèmes simples de rang n_1 n_1+1 .. n_2.

Prenons par exemple le cas de l'ajout d'une expression figée ambiguë. Soit l'expression :

câble optique

dans la séquence :

un câble optique rapide

L'ajout de cette expression se fait après que les mots *câble* et *optique* aient été insérés. Le graphe obtenu est celui de la figure 4 (en bas). L'expression *câble optique* recouvre les mots *câble* et *optique* de rang trois et quatre. Le nœud correspondant à cette expression doit donc être relié aux nœuds de rang deux, c'est-à-dire aux nœuds représentant le mot *un*. Le graphe se présente alors ainsi :

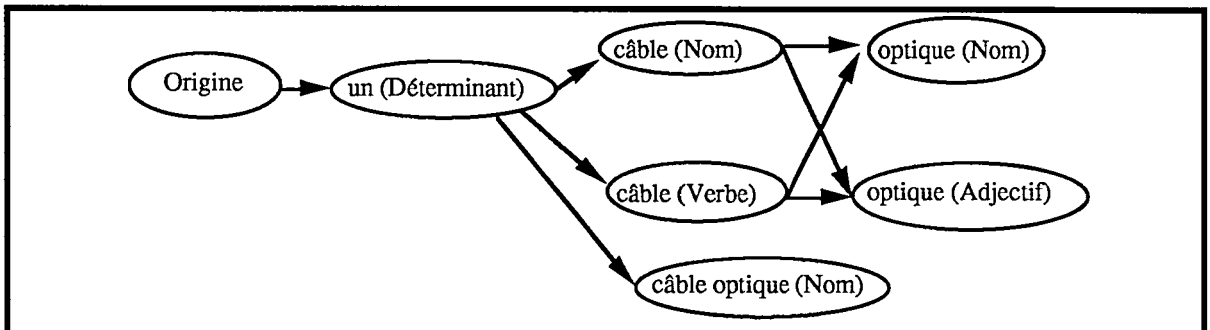


FIGURE 5a — AJOUT D'UNE EXPRESSION FIGEE AMBIGUE

Enfin, le nœud représentant l'expression *câble optique* est maintenant considérée comme un nœud de rang quatre. C'est-à-dire que les nœuds de rang cinq seront reliés à la fois aux nœuds du mot *optique* et aux nœuds de l'expression *câble optique*. Par exemple, en insérant le mot *rapide*, on obtient le graphe de la figure 5b.

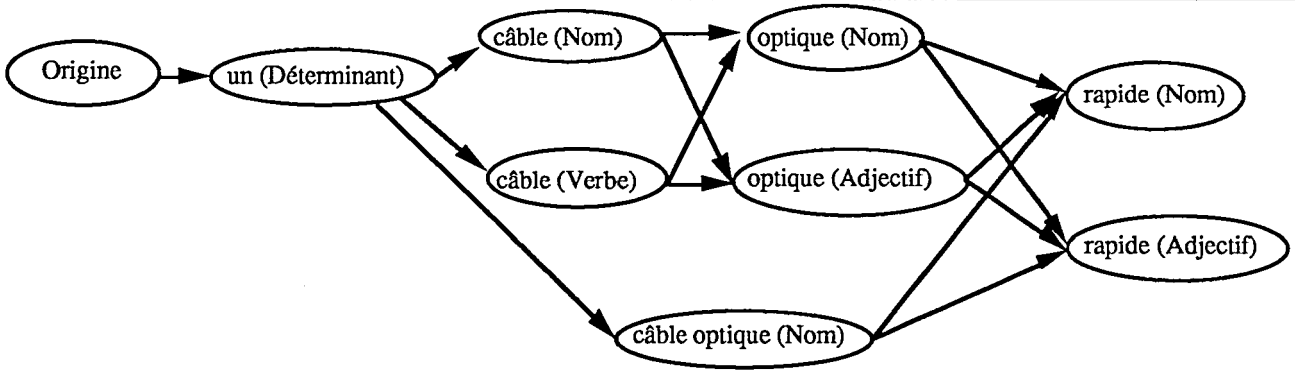


FIGURE 5b — AJOUT D'UN MOT APRES UNE EXPRESSION FIGEE AMBIGUE

En phase de transformation

L'ajout d'un nœud en phase de transformation est nécessaire lors de la reconnaissance d'expressions semi-figées. En effet, ces expressions ne peuvent être reconnues qu'une fois le graphe constitué car elles dépendent des homographies (par exemple, l'expression *faire table rase* ne sera reconnue dans la séquence *fait table rase* que si *fait* est un verbe). Dans le cas où l'expression n'est pas ambiguë, il faut remplacer le chemin identifié — c'est à dire les mots constituant l'expression — par des noeuds représentant les différentes fonctions syntaxiques de l'expression. Ce cas sera traité dans la partie "substitution". Par contre, dans le cas où l'expression est ambiguë, ceci nous amène à devoir insérer autant de nœuds que d'homographes de l'expression avec les mêmes connexions que le chemin identifié.

Le processus est le suivant :

- Créer les h nœuds représentant les h homographies de l'expression.
- Connecter tous les antécédents de l'origine du chemin identifié à ces nœuds.
- Connecter ces nœuds à tous les successeurs du but du chemin identifié.

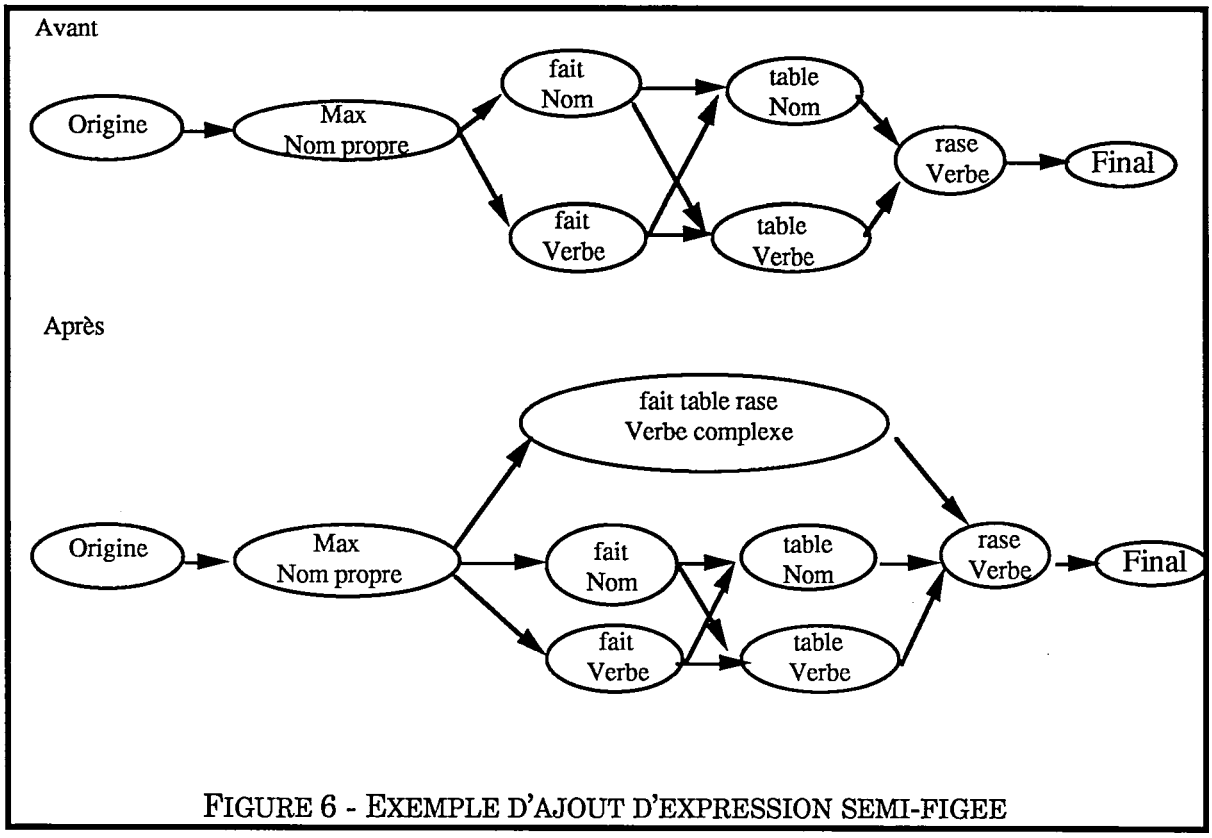


FIGURE 6 - EXEMPLE D'AJOUT D'EXPRESSION SEMI-FIGEE

b La suppression

La suppression n'intervient que pendant la phase de transformation du graphe (sauf dans le cas de la reconnaissance d'une expression figée non ambiguë, mais ce cas a déjà été traité au paragraphe précédent).

Une suppression intervient lors de la détection d'un chemin qui conduirait à une analyse fausse de la phrase. Ceci peut intervenir lors du découpage en propositions, par exemple dans le cas d'un chemin complet ne comportant pas de verbe, ou lors de l'application des règles de levée des homographies, par exemple dans le cas d'un chemin contenant un déterminant pluriel immédiatement suivi d'un nom singulier.

Deux cas sont à distinguer :

— La suppression d'un noeud

— La suppression d'un chemin

Suppression d'un nœud

Ce cas peut par exemple se présenter avec un lexème simple qui n'existe pas en dehors d'une expression figée (par exemple : *fur*). La suppression d'un nœud est une opération simple à effectuer : il suffit de détruire tous les arcs ayant ce nœud comme origine ou destination, puis de supprimer ce nœud (cf l'exemple de la figure 7).

Remarquons que la suppression d'un seul nœud permet d'éliminer un grand nombre d'analyses potentielles, au minimum $A_+ \times A_-$, où A_+ est le nombre d'arc ayant le nœud détruit comme destination et A_- le nombre d'arcs ayant le nœud détruit comme origine. Malheureusement, ce cas de figure se présente assez rarement.

Suppression d'un chemin

La suppression d'un chemin est nettement plus compliquée dans la mesure où un sous-chemin du chemin à détruire peut appartenir à un autre chemin parfaitement valide. La difficulté réside précisément dans le maintien des sous-chemins du chemin à détruire. On ne peut donc pas se contenter de supprimer les arcs et les nœuds du chemin concerné.

Considérons le cas suivant (Figure 8) : la destruction du chemin constitué par les arcs **A** et **B** ne doit détruire ni le chemin **CB**, ni le chemin **AD**, or si l'on se contente de détruire les arcs **A** et **B**, on supprime également tous les chemins contenant un de ces deux arcs (par exemple, les chemins **CB** et **AD**), ce qui est incorrect..

L'algorithme utilisé nécessite de dupliquer les nœuds intermédiaires de manière à maintenir les sous-chemins potentiels. Dans le cas de notre exemple, le nœud ① est dupliqué, ce qui donne le résultat de la figure 9.

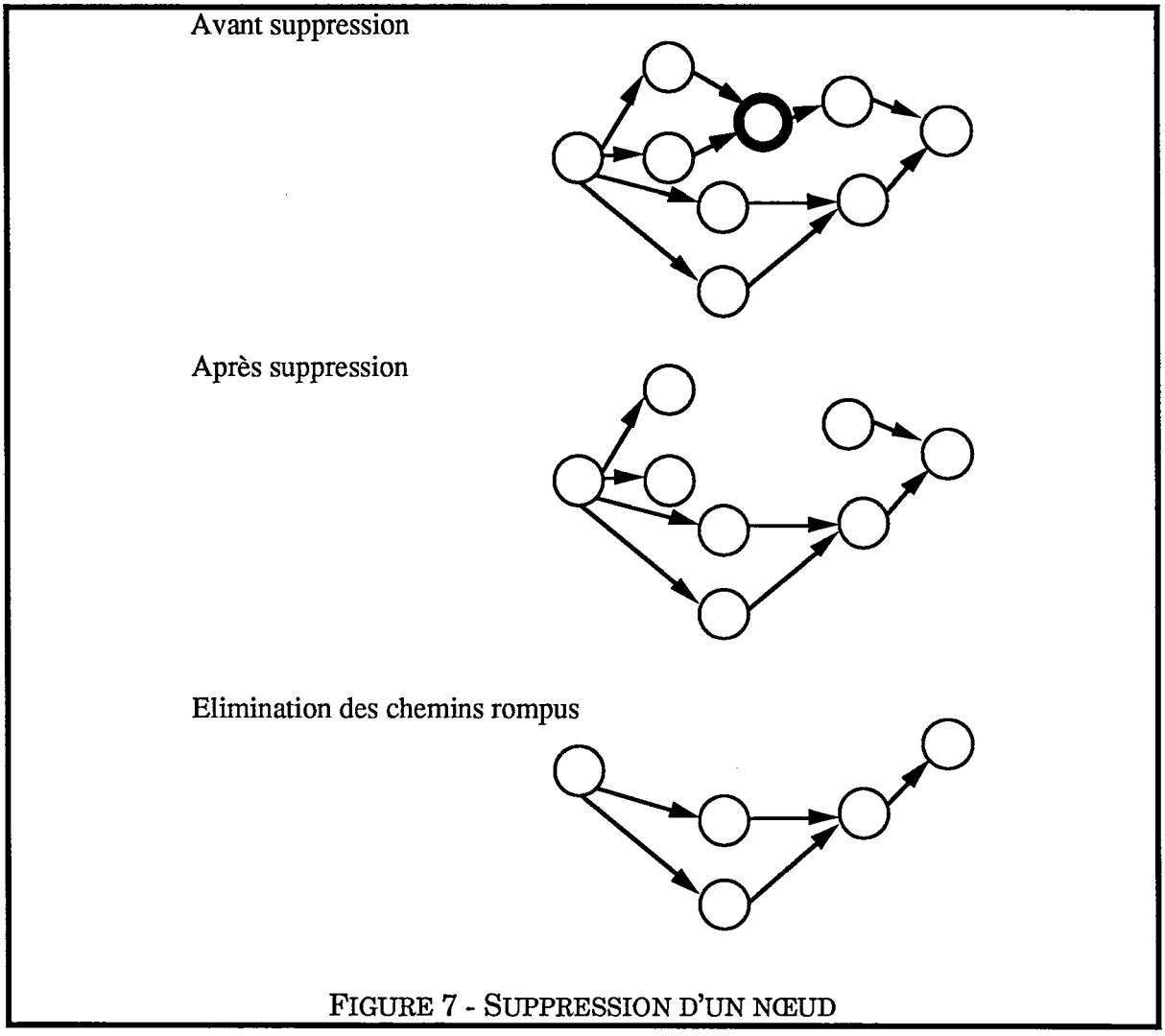


FIGURE 7 - SUPPRESSION D'UN NŒUD

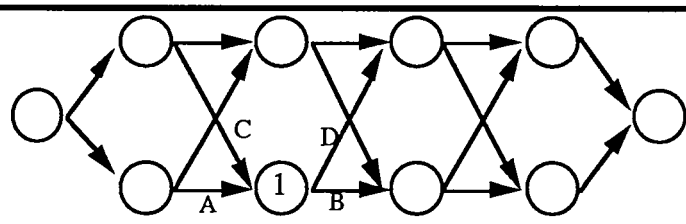


FIGURE 8- EXEMPLE DE CHEMIN A DETRUIRE

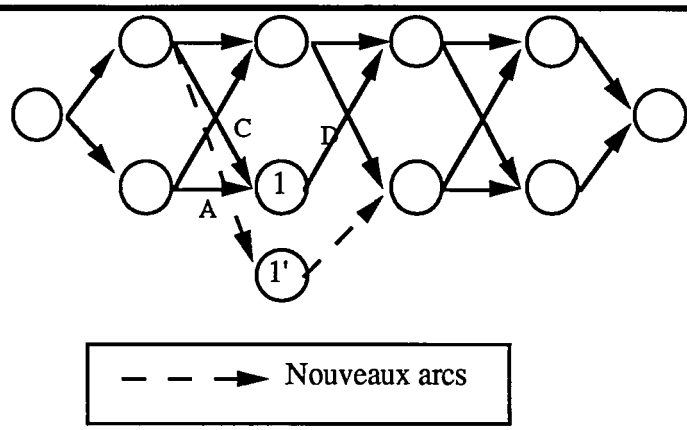


FIGURE 9 - APRES LA DESTRUCTION

D'une manière plus générale, l'algorithme est le suivant :

Pour invalider un chemin constitué de n nœuds nommés N_1, N_2, .. N_n, il faut :

- Dupliquer les nœuds N_2 à N_{n-1}, ce qui fournit n-2 nœuds nommés N'_2 .. N'_{n-1}.
- Relier les nœuds dupliqués entre eux : pour tout i compris entre 3 et n-1, le nœud N'_{i-1} est relié au nœud N'_i.
- Relier chaque nœud dupliqué N'_i à tous les antécédents du nœud N_i, **sauf au nœud N_{i-1}**.
- Relier le nœud N'_{n-1} au nœud N_n.
- Détruire l'arc reliant les nœuds N_{n-1} et N_n.

Pour simplifier, le principe est de créer deux sous-chemins parallèles à intersection vide qui permettent de conserver les chemins complets ne contenant pas le chemin à détruire, ceci est illustré dans la figure 10.

Il est également intéressant de remarquer que plus le chemin détruit est court, plus le nombre d'analyses potentielles éliminées est grand. Ceci nous amènera dans la suite à identifier les séquences impossibles les plus courtes possibles.

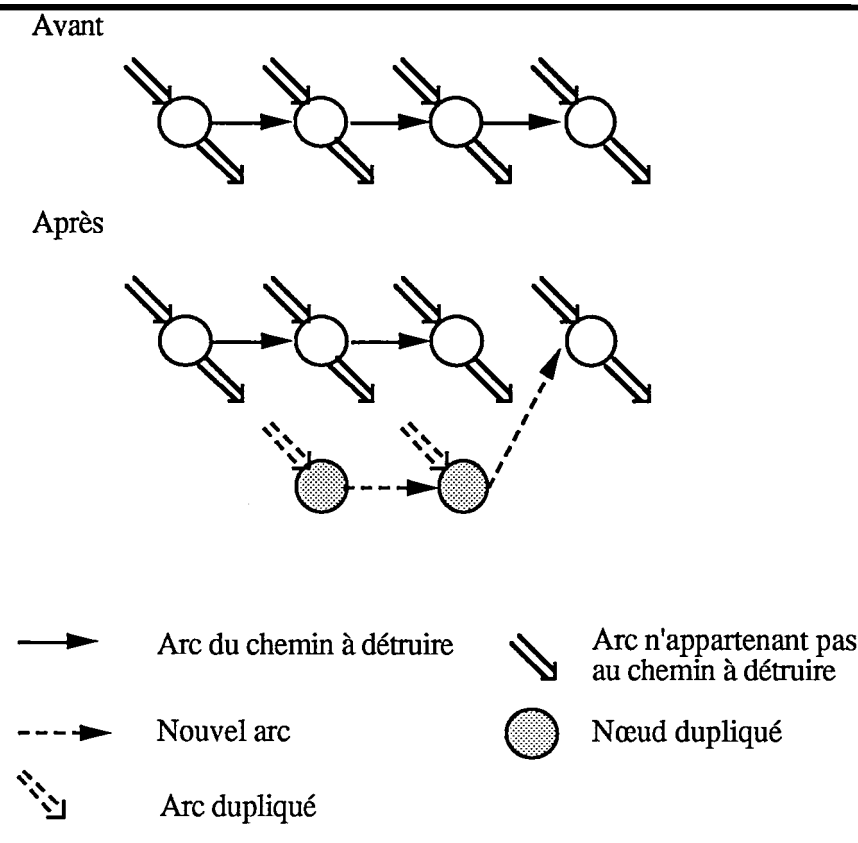


FIGURE 10 - PRINCIPE DE SUPPRESSION

e. La substitution

L'opération de substitution est effectuée lorsque l'on désire remplacer un nœud par un chemin ou vice-versa. Cette opération n'intervient que durant la phase de transformation de la phrase.

Substitution d'un nœud par un chemin

La substitution d'un nœud par un chemin intervient lors de l'expansion de certains mots comme par exemple la préposition *des* en *de les*.

Le processus d'expansion d'un nœud N en un chemin C comprend quatre étapes :

- Créer le nouveau chemin C.
- Raccorder tous les antécédents du nœud N à l'origine du nouveau chemin C.
- Raccorder le but du nouveau chemin C à tous les successeurs du nœud N.
- Supprimer le nœud N et les arcs ayant ce nœud comme origine ou destination (cf. paragraphe précédent : "La suppression")

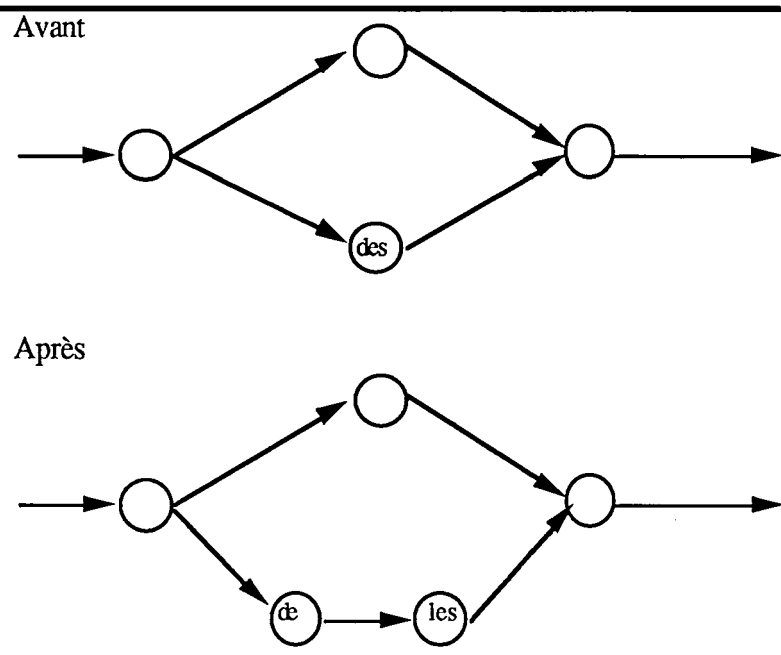


FIGURE 11 - SUBSTITUTION D'UN NŒUD

Substitution d'un chemin

Remplacer un chemin par un nœud est nécessaire lors de l'identification d'une expression semi-figée. Cette opération fait appel à deux opérations précédemment définies : l'ajout d'un nœud durant la phase de transformation, puis la suppression d'un chemin.

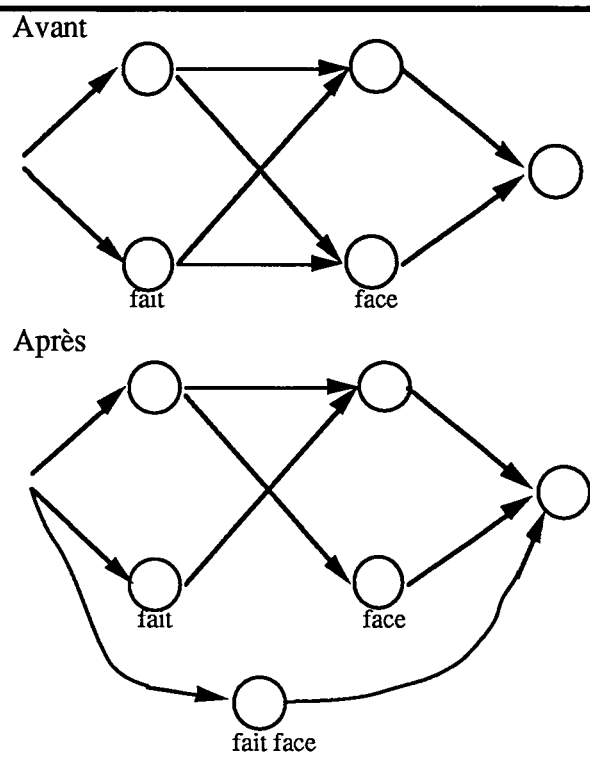


FIGURE 12- SUBSTITUTION D'UN CHEMIN

Conclusion

Le graphe de phrase et les opérations possibles sur ce graphe sont au cœur de Tomas. Nous verrons par la suite comment ces opérations permettent de limiter fortement le nombre d'analyses potentielles de la phrase pour un coût raisonnable.

V. Le lexique de Tomas

Le lexique de Tomas contient toutes les informations propres au vocabulaire. Nous allons d'abord présenter ce lexique d'un point de vue fonctionnel (que fait-il ?) puis d'un point de vue technique (comment est-il fait?).

1. Description fonctionnelle

Le lexique de Tomas est dérivé du DELAF (Dictionnaire Electronique du LADL pour les mots Fléchis) de B. Courtois avec quelques ajouts, ainsi que de lexiques internes à la société CORA pour les expressions figées. Le formalisme du lexique de Tomas est proche de celui du DELAF et un programme de traduction entre ces formats a été développé.

Dans Tomas, le lexique est un ensemble de graphies décorées. Ces graphies peuvent être des mots simples ou des expressions figées. Chaque graphie est décrite par l'ensemble des rôles syntaxiques qu'elle peut avoir. Par exemple, la graphie *joue* est définie comme suit :

joue :	(joue)NOM.F.S	(nom féminin singulier) ou
	(jouer)VER.PI.JE	(verbe présent de l'indicatif, 1ère personne) ou
	(jouer)VER.PI.IL	(verbe présent de l'indicatif, 3ème personne) ou
	(jouer)VER.SUP.JE	(verbe présent du subjonctif, 1ère personne) ou
	(jouer)VER.SUP.IL	(verbe présent du subjonctif, 3ème personne) ou
	(jouer)VER.IM.TU	(verbe impératif, 2ème personne)

Nous appelerons entrée syntaxique simple l'association entre une graphie et un rôle syntaxique. Par exemple, la définition de la graphie *joue* contient six entrées syntaxiques simples.

a. Contenu du lexique

Pour chaque entrée syntaxique simple, les informations suivantes sont présentes dans le lexique :

- Graphie.
- Lemme : le lemme est le représentant canonique d'une classe flexionnelle. Pour les verbes, il s'agit de l'infinitif, pour les adjectifs, de la forme masculin singulier, etc.
- Matrice des expressions figées (pour les mots simples uniquement) : il s'agit d'une matrice utilisée lors de la reconnaissance des expressions figées.
- Information de classement transversal : il s'agit de d'informations qui ne découlent pas directement des catégories et sous-catégories syntaxiques.

— Expression ambiguë (pour les expressions figées seulement) : indique si l'expression présente une ambiguïté. Par exemple, l'expression *au fur et à mesure* n'est pas ambiguë, alors que le mot composé *pomme de terre* l'est.
— Catégorie syntaxique.
— Sous-catégories propres à chaque catégorie syntaxique.

Catégories syntaxiques

La catégorie syntaxique définit la fonction du mot, à choisir parmi les catégories suivantes : (les abréviations sont fournies entre parenthèses) :

— Nom commun (NOM)
— Adjectif (ADJ)
— Verbe (VER), sauf les participes passés qui ont droit à une catégorie à part.
— Adverbe (ADV)
— Déterminant (DET)
— Interjection (INT)
— Préposition (PRE)
— Pronom (PRO)
— Segment de mot composé (SMC), cette catégorie est utilisée pour les mots qui n'apparaissent pas en dehors d'expressions figées, comme par exemple le mot *fur*.
— Participe passé (PPS), cette catégorie n'est pas une sous-catégorie du verbe car un participe passé, à l'inverse d'un verbe, possède un genre et un nombre mais pas de personne.
— Quantifieur (QUA), par exemple : *tout, chaque, deux...*
— Conjonction de subordination (CSU)
— Conjonction de coordination (CCO)
— Nom propre (NPR)

Chacune de ces catégories syntaxiques possède des sous-catégories telles que le genre ou le nombre qui permettent de définir plus précisément les entrées syntaxiques simples.

Sous-catégories syntaxiques

Nom, nom propre, adjectif, participe passé et quantifieur

- Le genre (GEN) :
  - Masculin (M)
  - Féminin (F)
  - Masculin ou féminin (MF)
- Le nombre (NBR) :
  - Singulier (S)
  - Pluriel (P)
  - Singulier ou pluriel (SP)

Déterminant

- Le genre et le nombre
- La sous-catégorie (SCA)
  - Personnel (PER)
  - Possessif (POS)
  - Réfléchi (REF), cette sous-catégorie, qui n'est pas utilisée pour les déterminants, sert uniquement pour les pronoms.
  - Numérique (NUM)
  - Interrogatif (INTER)
  - Relatif (REL)
  - Démonstratif (DEM)
  - Article défini (ARD)
  - Article indéfini (ARI)
  - Indéfini (IND)
- La personne (PER)
  - Première personne du singulier (JE)
  - Deuxième personne du singulier (TU)
  - Troisième personne du singulier (IL)
  - Première personne du pluriel (NOUS)
  - Deuxième personne du pluriel (VOUS)
  - Troisième personne du pluriel (ILS)
  - Sans

Pronom

- Le genre, le nombre, la personne et la sous-catégorie
- La fonction (CAS)
  - Sujet (SUJ)
  - Objet direct (COD)
  - Objet indirect (COI)

Verbe

- La personne
- Le temps
  - Indicatif présent (PI)
  - Indicatif imparfait (II)
  - Indicatif futur (FI)
  - Passé simple(PS)
  - Conditionnel présent (PC)
  - Subjonctif présent (SUP)
  - Subjonctif imparfait (SUI)
  - Impératif (IM)
  - Infinitif (IN)
  - Participe présent (PPR)

Les autres catégories syntaxiques n'ont pas de sous-catégories.

h Classement (ordre lexicographique)

Le lexique étant accessible par un index, il est nécessaire de définir un tri sur ses entrées.

Le lexique est classé de telle manière que les mots ne différant que par des signes diacritiques soient regroupés ensemble. Ce tri s'effectue en deux temps :

- Un premier tri en ignorant les signes diacritiques.
- Un deuxième tri des classes obtenues, en tenant uniquement compte de ces signes.

Ceci permet de rechercher des graphies en partie capitalisées (le cas le plus fréquent est celui des graphies dont la première lettre est en majuscule parce qu'elles sont en début de phrase).

2. Description technique

Le lexique se compose de cinq fichiers distincts :

- Le fichier des index
- Le fichier des résidus
- Le fichier des informations syntaxiques
- Le fichier des lemmes
- Le fichier des matrices d'expressions figées

Les relations entre ces fichiers sont schématisées dans la figure 1.

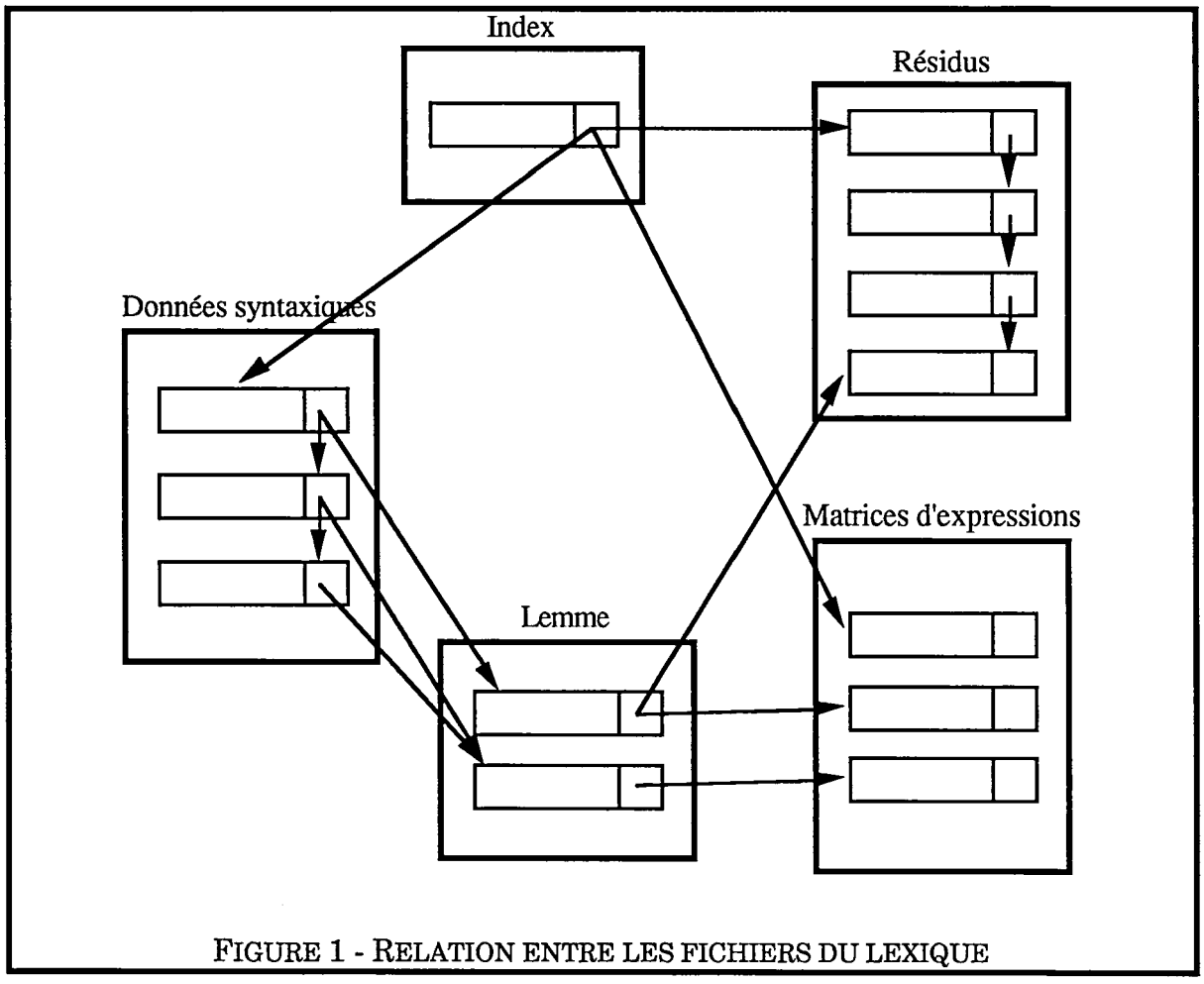


FIGURE 1 - RELATION ENTRE LES FICHIERS DU LEXIQUE

Pour des raisons d'efficacité, tous ces fichiers sont composés d'enregistrements de longueur fixe. De plus, ces fichiers sont paginés grâce à un algorithme qui permet de conserver en mémoire les pages les plus utilisées.

Le fichier des index

Ce fichier contient le début des graphies et sert à identifier les mots. La structure d'index utilisée est celle des arbres de Bayer [Knuth 73]. Dans cette représentation, chaque page contient n couples (clé, pointeur). Quand on cherche une clé K, on cherche dans la page racine du fichier la plus grande clé K' inférieure ou égale à la clé recherchée K. Si la clé trouvée K' est égale à la clé K, on a fini, sinon on recommence dans la page spécifiée par le pointeur de la clé K'. Cette méthode est particulièrement rapide car à chaque étape, elle divise l'espace de recherche par n. De plus, elle permet de se déplacer séquentiellement dans les index et enfin assure un remplissage optimal des pages (chaque page, sauf la page racine, contient entre n/2 et n clés).

Comme nous venons de le dire, seul le début du mot est placé dans les index, pour des raisons de place consommée sur le disque. Le reste de la graphie est placé dans le fichier des résidus qui sera détaillé dans le paragraphe suivant.

Chaque clé du fichier d'index contient trois liens :

— Un lien (éventuellement nul) vers le fichier des résidus où se trouve la fin de la graphie.

— Un lien vers le fichier des données syntaxiques.

— Un lien (éventuellement nul) vers le fichier des matrices d'expressions figées.

Le fichier des résidus

Ce fichier contient la fin des graphies, du moins la fin de celles qui sont trop longues pour figurer en entier dans le fichier d'index. Cette méthode permet de diminuer la place perdue dans le fichier des clés.

Dans ce fichier, la donnée de base est une liste, chaque élément de la liste contenant une partie de la fin de la graphie.

La figure 2 fournit un exemple pour le mot *anticonstitutionnellement*.

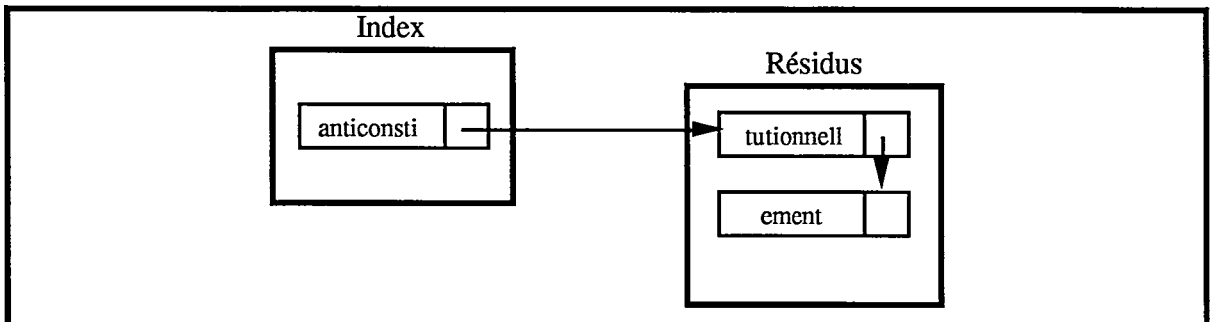


FIGURE 2 - RESIDUS DE *ANTICONSTITUTIONNELLEMENT*

Le fichier des données syntaxiques

Ce fichier contient toutes les données syntaxiques. La donnée de base est également une liste dont chaque élément correspond à une entrée syntaxique simple. Chaque entrée syntaxique simple contient un lien vers le fichier des lemmes.

Par exemple, pour le mot *joue*, on obtient le résultat de la figure 3.

Le fichier des lemmes

Ce fichier contient les informations liées aux lemmes, à savoir :

— La graphie du lemme, avec un lien (éventuellement nul) vers le fichier des résidus (la méthode est la même que pour le fichier d'index).

— Un lien vers les matrices d'expressions figées propres aux lemmes.

En reprenant l'exemple précédent, on obtient la figure 4.

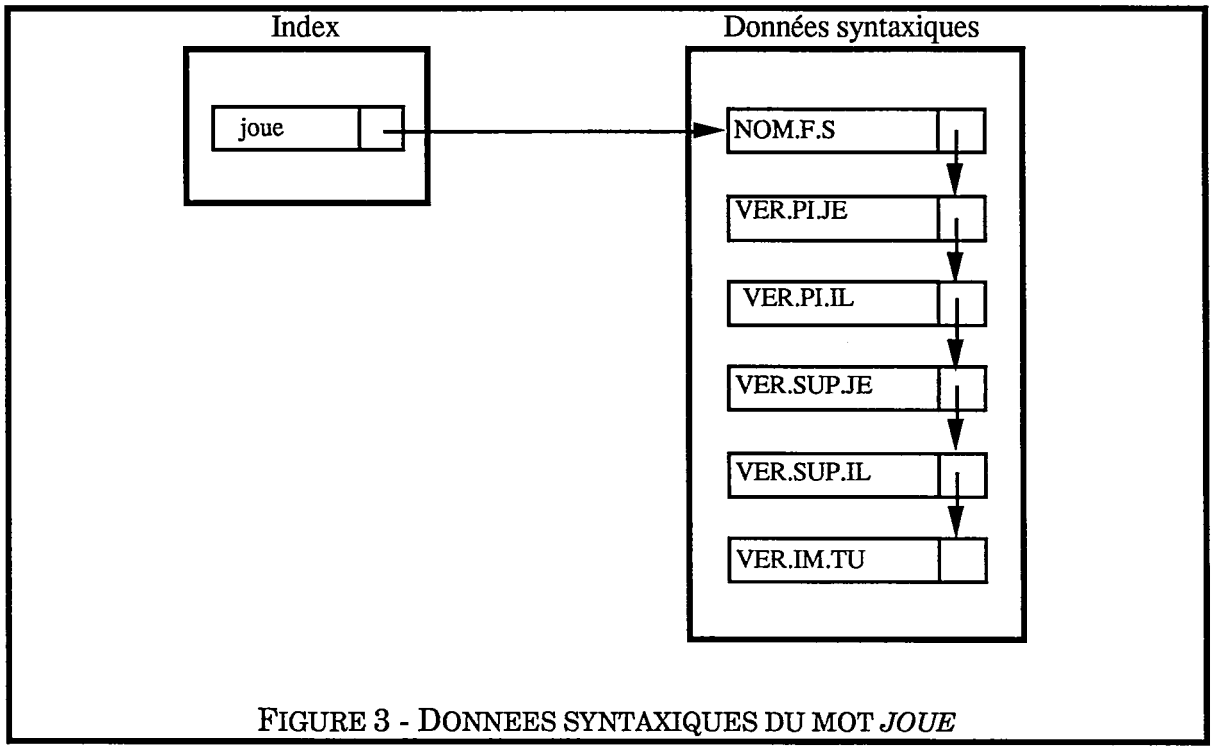


FIGURE 3 - DONNEES SYNTAXIQUES DU MOT *JOUE*

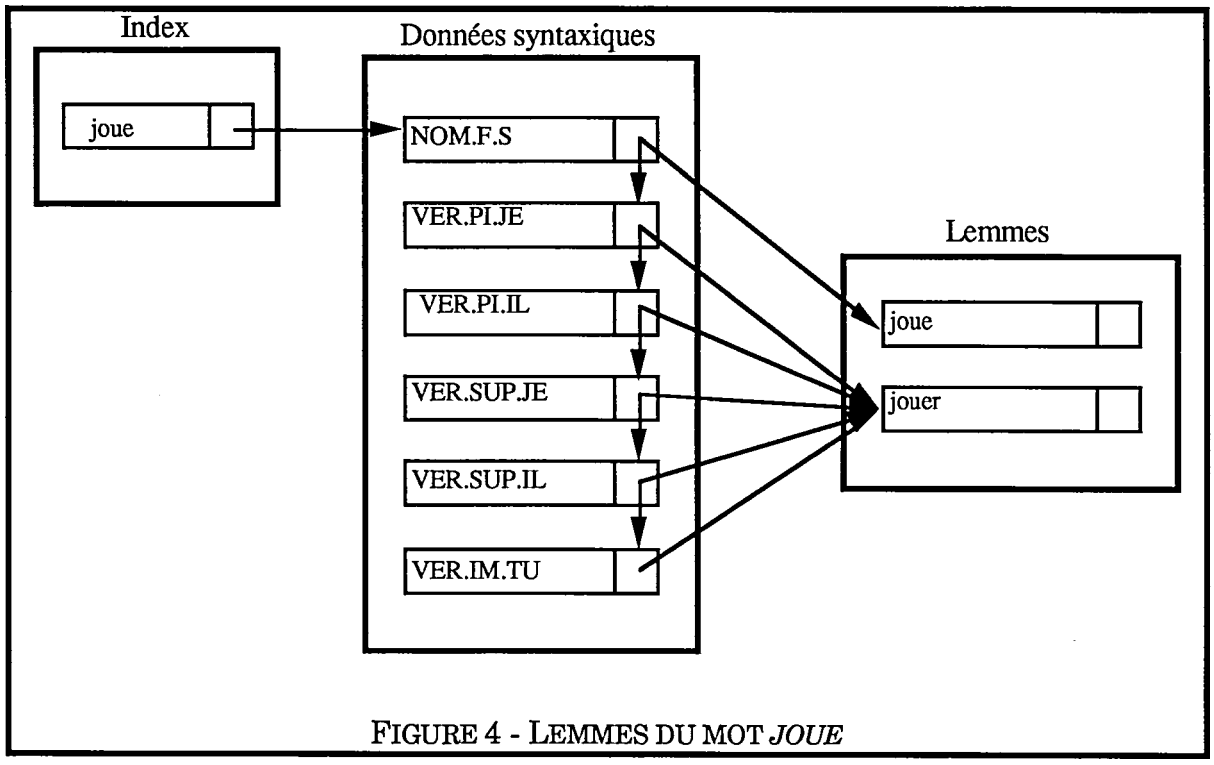


FIGURE 4 - LEMMES DU MOT *JOUE*

Le fichier des matrices d'expressions figées

Les matrices d'expressions figées sont définies dans le chapitre suivant. Pour l'instant, il suffit de savoir que l'on peut les modéliser comme des fonctions à deux arguments telles que pour tout mot M,

$f_M(i,j) = 1$ si et seulement s'il y a au moins une expression figée de longueur j dont M soit le $i^{ème}$ mot,

$f_M(i,j) = 0$ sinon.

Ces matrices sont utilisées lors de la reconnaissance des expressions figées.

Ce fichier contient tout simplement les matrices d'expressions figées, chaque élément de la matrice étant un bit.

Implantation

Tous ces fichiers sont gérés à l'aide de trois composants logiciels :

— Gestion de fichiers indexés pour le fichier d'index.

— Gestion de fichiers de données simples pour les fichiers des lemmes et des matrices d'expressions figées.

— Gestion de fichiers de listes pour les fichiers de résidus et de données syntaxiques. Ce dernier composant est une extension du composant précédent.

Conclusion

Le lexique de Tomas contient environ 800.000 formes fléchies (les mêmes que le DELAF) et quelques 2.000 expressions figées dans le domaine de l'informatique (extrait d'un corpus d'une trentaine de pages issues des journaux Le Monde Informatique et 01 Informatique). L'accès à ce lexique, bien que relativement rapide grâce aux index et à la pagination, reste le goulot d'étranglement de Tomas.

VI. Les Machines de Tomas

Après avoir décrit les structures de phrase et de lexique utilisées par Tomas, il nous reste à examiner les traitements effectués sur ces structures. Dans ce chapitre, nous allons nous intéresser à deux "machines" qui seront réutilisées par Tomas. La première partie de ce chapitre est consacrée au "génie logiciel" de manière à situer Tomas à l'intérieur de cette discipline.

1. Le génie logiciel

Pour la suite de notre exposé, il nous semble nécessaire d'exposer quelques notions de génie logiciel. En effet, Tomas a été réalisé en ADA et utilise au maximum les techniques du génie logiciel.

a. Les buts et principes du génie logiciel

Le génie logiciel est un ensemble de techniques qui permettent de mieux contrôler les phases de développement et de maintenance des logiciels. Les cinq buts fondamentaux du génie logiciel sont :

— Modifiabilité : Les logiciels doivent pouvoir être adaptés et modifiés facilement que ce soit parce que les données du problème ont changé ou parce qu'il faut corriger des erreurs.

— Efficacité : Les logiciels doivent utiliser au mieux les ressources disponibles.

— Fiabilité : Les logiciels doivent — bien sûr — comporter le moins d'erreurs possibles, mais en plus ils doivent, quand c'est inévitable à cause des conditions externes, se dégrader harmonieusement, c'est à dire que les conséquences des erreurs doivent rester les plus limitées possible en sorte que les fonctions qui peuvent encore fonctionner le fassent.

— Clarté : Les logiciels doivent être écrits de la manière la plus claire possible, tant pour faciliter la maintenance que pour permettre de vérifier leur validité.

— Réutilisation : Pour diminuer les développements (et donc les risques d'erreurs), il faut pouvoir réutiliser au maximum les développements déjà effectués.

Pour atteindre ces quatre buts, le génie logiciel applique les sept principes suivants :

— Abstraction
— Dissimulation de l'information
— Modularité
— Localisation
— Uniformité
— Intégralité
— Validité

Les deux premiers principes (abstraction et dissimulation) permettent de considérer un problème sans avoir à prendre en compte les questions d'implantation de bas niveau.

Les deux principes suivants (modularité et localisation) permettent de diviser un problème en plusieurs sous-problèmes afin de les traiter plus facilement, et de ne traiter qu'un seul problème à la fois.

Enfin, les trois derniers principes (uniformité, intégralité et validité) permettent d'avoir des logiciels clairs, fiables et corrects.

b Les concepts du génie logiciel

Le génie logiciel utilise un certain nombre de concepts que nous allons définir dans la suite. Ces concepts permettent de mieux comprendre la manière dont Tomas est structuré du point de vue logiciel.

Composant logiciel

Un composant logiciel est à l'informatique ce qu'un composant classique est à l'électronique. Il s'agit d'un ensemble de structures, de données et d'algorithmes qui traite d'un problème particulier. Voici quelques exemples de composants utilisés dans Tomas:

— Définition et gestion de graphes.
— Gestionnaires de listes, de files d'attentes, de piles, d'ensembles, de messages ...

Un composant possède une "spécification" qui définit la manière dont le monde extérieur peut interagir avec ce composant et une "implantation" qui réalise les traitements nécessaires au composant. Seule la spécification est connue du monde extérieur, ce qui assure une totale indépendance vis-à-vis de l'implantation.

Généricité et instantiation

La généricité est une notion très puissante qui permet de définir des composants sans avoir à décider, au moment de la définition, de la nature exacte des objets qu'ils manipulent. Prenons par exemple le cas d'un composant qui réalise le tri d'un tableau d'objets. Il est tout à fait possible d'écrire ce composant sans savoir si les objets à trier seront des nombres, des chaînes de caractères ou d'autres objets. Tout ce qui est nécessaire, c'est de posséder une fonction qui permette de comparer deux objets. En ADA (francisé pour plus de clarté), l'interface d'un tel composant aurait cette forme :

```
générique
        type OBJET est privé;
        type INDICE est discret;
        type TABLE est tableau (INDICE) de OBJET;
        avec fonction INFERIEUR (A,B : OBJET) retourne booléen;


fonction TRI (MA_TABLE : TABLE) retourne TABLE;
```

L'instantiation est l'opération qui consiste à obtenir un composant utilisable à partir d'un composant générique en fournissant les objets nécessaires. Par exemple, pour obtenir un composant qui trie des tableaux de réels indexés par des naturels, on écrirait :

```
type TABLE_DE_REEL est tableau (NATUREL) de REEL;
fonction TRI_DE_REEL est nouveau
            TRI (       OBJET           = REEL,
                        INDICE          = NATUREL,
                        TABLE           = TABLE_DE_REEL
                        INFERIEUR       = "<");
```

Remarquons qu'en fournissant la fonction ">" au lieu de la fonction "<", on renverserait l'ordre du tri.

La notion de généricité est très importante pour la réutilisation, puisqu'elle permet de définir des composants généraux sans préjuger de leur usage final.

Machine virtuelle

Une machine virtuelle est un composant particulier : ce type de composant possède un état interne qui n'est pas directement accessible de l'extérieur. Par exemple, un moteur de voiture peut être considéré comme une machine virtuelle. en effet, un moteur possède un état interne, caractérisé par sa vitesse de rotation. Cet état n'est pas directement visible de l'extérieur du moteur. Par contre, il existe des moyens définis pour connaître cet état (comme le compte-tours) ou pour modifier cet état (comme l'accélérateur ou la clé de contact).

Ces machines virtuelles peuvent être utilisées pour traiter tout phénomène qui nécessite de conserver une trace de ce qui est arrivé dans le passé. Tomas utilise deux machines : une machine de reconnaissance de structures, qui est employée pour l'application des règles de levée d'homographies et une machine de reconnaissance d'expressions qui sert pour l'identification des expressions figées et semi-figées.

Conclusion

Cette présentation rapide du génie logiciel devrait permettre de mieux appréhender les principes de Tomas. Le lecteur intéressé pourra se référer à [Booch 88]

2. Machines de Tomas

Nous allons maintenant présenter deux machines (au sens du génie logiciel) qui jouent un rôle important à l'intérieur de Tomas. La première de ces deux machines est nommée "machine générique de reconnaissance d'objets" et permet d'identifier une séquence d'objets à partir d'une série de modèles. La deuxième machine, nommée "Machine générique de reconnaissance d'expressions figées", est utilisée, comme son nom l'indique, pour reconnaître des expressions figées.

a. Machine générique de reconnaissance d'objets

Objectifs

L'objectif principal de cette machine est de reconnaître dans un flot d'objets une séquence d'après un modèle. Présenté ainsi, le problème est simple; toutefois, cette machine doit être efficace, performante, réutilisable et surtout s'adapter à la structure en graphe de Tomas.

Avant tout, définissons ce que nous entendons par modèle : un modèle est une suite de contraintes dont certaines peuvent être facultatives. Une contrainte peut être vue comme une fonction de l'ensemble des objets vers l'ensemble des valeurs de vérité {vrai, faux}. Une contrainte C est compatible avec un objet O si et seulement si C (O) = vrai. Une séquence $O_1,O_2 \ldots O_n$ est identifiée par un modèle $C_1, C_2 \ldots C_n$ si chaque objet O_i de la séquence est compatible avec la contrainte C_i de même rang.

Dans Tomas, les objets sont les nœuds du graphe de phrase, c'est-à-dire des mots, des expressions figées ou des propositions, et les contraintes sont des conditions sur ces nœuds. Par exemple, il peut être nécessaire de détecter les séquences constituées de deux verbes conjugués. Dans ce cas, le modèle se présentera ainsi :

[VER,TPS/=(IN,PPR)] [VER,TPS/=(IN,PPR)]

Si l'on désire reconnaître ce type de séquence même dans le cas où il y a une insertion d'adverbe, deux solutions sont possibles. Soit l'on place une contrainte facultative dans le modèle :

[VER,TPS/=(IN,PPR)] {ADV} [VER,TPS/=(IN,PPR)]

Soit l'on décide que les adverbes sont transparents, c'est-à-dire qu'ils n'influent pas sur la reconnaissance. Ces deux méthodes ne sont pas tout à fait équivalentes ; en effet, la première n'autorise que l'insertion d'un seul adverbe alors que la seconde permet d'insérer un nombre d'adverbes quelconque.

Pour être efficace, une telle machine doit permettre :

— de gérer plusieurs modèles simultanément.

— de supporter la présence d'objets transparents (un objet transparent est un objet qui ne doit pas influer dans la comparaison avec les modèles).

Pour être performant, il est important de limiter au maximum le nombre de comparaisons entre les objets et les modèles.

Pour être réutilisable, il est important de ne pas limiter les objets et les modèles et surtout de laisser l'utilisateur fournir les moyens de comparaison entre un objet et une contrainte.

Enfin, cette machine doit prendre avantage de la structure en graphe de Tomas. En effet, la structure en graphe permet de factoriser deux séquences commençant de la même manière : sur la figure 1, on voit que dans les séquences 1234 et 1235 la sous-séquence 123 est représentée de manière unique. De la même manière, il serait dommage, après avoir cherché à identifier les modèles de la séquence 1234, de recommencer pour la séquence 1235 sans prendre avantage de ce qui a déjà été fait pour la sous-séquence 123.

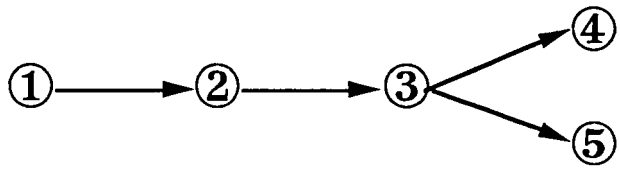


FIGURE 1 - UN GRAPHE AVEC DEUX SEQUENCES

Moyens

Examinons maintenant les moyens qui nous permettront d'atteindre ces objectifs. Nous allons d'abord spécifier les paramètres à fournir à cette machine, puis la façon dont cette machine interagit avec l'extérieur avant de nous intéresser, dans la dernière partie, à son fonctionnement.

Paramètres génériques

Pour utiliser cette machine, il faut fournir les éléments suivants :

- La définition des objets.
- La définition des contraintes.
- Une fonction qui permette de tester la transparence d'un objet.
- Une fonction qui effectue la comparaison entre un objet et une contrainte.
- Une procédure dite "procédure action" qui sera appelée si un modèle a été identifié. Dans ce cas, cette procédure sera appelée avec les paramètres suivants :
    - La sous-séquence reconnue.
    - Le modèle identifié.

Interaction avec le monde extérieur

Cette machine agit sur le monde extérieur grâce à la fonction de comparaison et à la procédure action. La fonction de comparaison est appelée chaque fois qu'il est nécessaire de comparer un objet et une contrainte ; la procédure action est appelée après reconnaissance d'un sous-modèle, c'est cette procédure qui permet au monde extérieur de réagir à l'identification d'un modèle.

Le monde extérieur agit sur la machine de deux manières différentes :

— Lors de la configuration de la machine, en lui fournissant les modèles à identifier.

— Au cours du fonctionnement : il est possible de remettre la machine à zéro, puis de fournir un par un les objets des séquences. Il est également possible de revenir en arrière sur une séquence : pour l'exemple précédent, on fournira d'abord les objets 1, 2, 3 et 4, puis on reviendra en arrière d'un objet avant de fournir l'objet 5. De cette manière, la séquence 1235 est traitée sans avoir à retraiter la sous-séquence 123. De cette manière, les deux séquences 1234 et 1235 sont traitées en six opérations au lieu de dix.

Fonctionnement interne de la machine

Les trois principales difficultés sont liées à la possibilité de revenir en arrière sur les séquences, à la possibilité de gérer plusieurs modèles simultanément et au fait qu'une sous-séquence à identifier peut se retrouver n'importe où dans la séquence totale.

Pour régler le problème du retour sur les séquences, il est nécessaire d'utiliser une pile qui stocke au fur et à mesure l'état interne de la machine de manière à pouvoir revenir en arrière en rétablissant simplement l'état antérieur de la machine.

Pour pouvoir gérer de manière efficace plusieurs modèles simultanément, la méthode la plus économique consiste à rassembler ces modèles dans un arbre syntaxique.

Un arbre syntaxique est un arbre dans lequel chaque branche représente une contrainte et dont chaque feuille identifie un modèle. Le principe de construction d'un tel arbre à partir d'une série de modèles est de rassembler sur une même branche les contraintes identiques, et ce jusqu'à la première discordance entre les modèles. Par exemple, prenons les quatre modèles suivants (chaque lettre correspond à une contrainte) :

— Modèle 1 : ABCDE

— Modèle 2 : ABCG

— Modèle 3 : AEF

— Modèle 4 : BGH

Ces quatres modèles fournissent l'arbre syntaxique suivant :

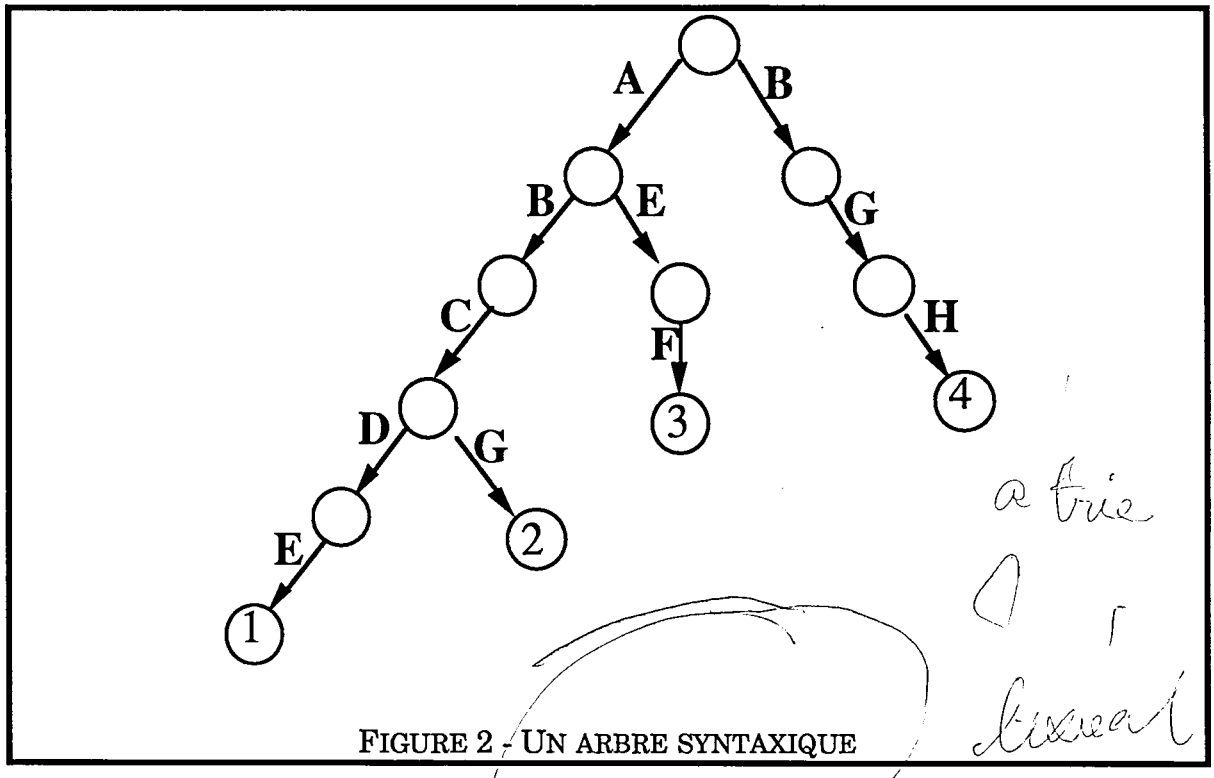


FIGURE 2 - UN ARBRE SYNTAXIQUE

Le troisième problème est réglé en examinant à chaque étape (donc à chaque fois qu'un objet est fourni)

— s'il est possible d'appliquer l'arbre syntaxique général, ce qui créera pour la suite autant de sous-arbres que de branches compatibles avec le nouvel objet.

— pour chaque sous-arbre créé lors des étapes précédentes, s'il s'applique, ce qui créera, pour la suite, autant de sous-arbres que de branches compatibles avec le nouvel objet.

Lors de cet examen, on aura soin de remarquer que lorsqu'un sous-arbre est réduit à une feuille, c'est que le modèle correspondant à cette feuille a été identifié.

Exemple

Considérons par exemple la séquence *aabcg,* que l'on compare avec l'arbre présenté dans la figure 2. Les contraintes sont représentées par des lettres majuscules et les objets par des lettres minuscules. Nous allons fournir à chaque étape l'état interne de la machine, c'est-à-dire la liste des sous-arbres créés lors des étapes précédentes.

Au démarrage, il n'y a pas de sous-arbres en mémoire.

Première étape

L'objet *a* est fourni à la machine. L'application de l'arbre syntaxique général fournit le sous-arbre suivant de la figure 3.

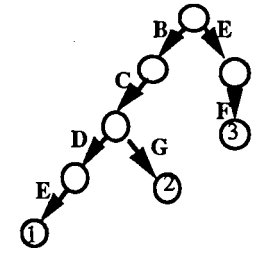


FIGURE 3 - PREMIERE ETAPE (*A*)

Deuxième étape

L'objet *a* est fourni à la machine. L'application de l'arbre syntaxique général fournit un sous-arbre identique à celui de la figure 3. Par contre le sous-arbre créé lors de la première étape ne peut plus s'appliquer et disparaît. L'état interne de la machine est donc identique à celui de l'étape précédente, à un décalage près.

Troisième étape

L'objet *b* est fourni à la machine. L'application de l'arbre syntaxique général fournit un sous-arbre (sous-arbre 2 de la figure 4) et l'application de l'arbre créé lors de l'étape précédente fournit un autre sous-arbre (sous-arbre 1 de la figure 4).

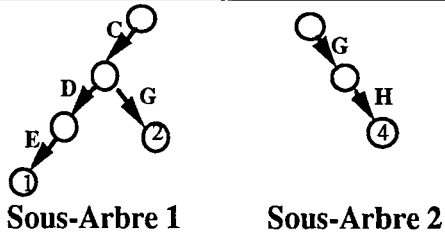


FIGURE 4 - TROISIEME ETAPE (*AAB*)

Quatrième étape

L'objet *c* est fourni à la machine. L'application de l'arbre syntaxique général ne fournit pas de sous-arbre. L'application du sous-arbre 1 créé lors de l'étape précédente fournit un autre sous-arbre, alors que l'application du sous-arbre 2 de l'étape précédente ne fournit pas de sous-arbre.

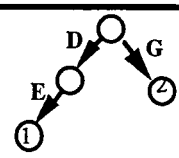


FIGURE 5 - QUATRIEME ETAPE (*AABC*)

Cinquième étape

L'objet *g* est fourni à la machine. L'application de l'arbre syntaxique général ne fournit pas de sous-arbre. L'application du sous-arbre créé lors de l'étape précédente fournit une feuille qui identifie le modèle 2 à partir de la sous-séquence *abcg*.

Retour sur la séquence

Si l'on désire revenir sur le dernier objet fourni — c'est-à-dire l'objet *g* — de manière à analyser une autre séquence proche *aabcd*, l'état interne de la machine redevient celui de la quatrième étape et l'objet *d* lui est alors fourni, ce qui provoquera de nouveau la création d'un sous-arbre.

Conclusion

Cette machine générique de reconnaissance d'objets est elle-même basée sur un composant de gestion d'arbres syntaxiques. Elle est utilisée non seulement dans Tomas lors de l'application des règles de levée d'homographies, mais aussi dans d'autres projets à l'intérieur de la société CORA (en particulier, elle sert pour l'identification des concepts dans une révision future du logiciel de recherche documentaire Darwin).

b. Machine générique de reconnaissance d'expressions figées

A première vue, le problème de la reconnaissance d'expressions figées pourrait être traité grâce à la machine décrite précédemment. Toutefois, la taille du problème est considérablement plus importante en ce qui concerne les expressions figées. En effet, la machine précédente est efficace pour un nombre de modèles ne dépassant pas le millier alors que le nombre d'expressions figées à reconnaître se chiffre plutôt en centaines de milliers.

Objectifs

De même que pour la machine décrite précédemment, l'objectif principal de cette machine est de reconnaître une expression figée parmi une suite d'objets. Cette machine doit également être performante, réutilisable et supporter la structure en graphe de Tomas.

La performance de cette machine est directement liée au nombre de recherches qui devront être effectuées dans le lexique des expressions figées. La performance optimale serait obtenue en ne recherchant dans le lexique que les expressions figées qui s'y trouvent, c'est-à-dire en utilisant le lexique pour rechercher des informations (étiquetage des expressions) et non pour tester la validité d'expressions figées potentielles.

Pour être réutilisable, il ne faut pas décider *a priori* du nombre maximum de mots d'une expression figée et surtout, il faut séparer le mécanisme de détection des expressions figées de l'accès aux lexiques.

Enfin, et comme pour la première machine, la structure en graphe de Tomas doit permettre de réutiliser les résultats obtenus sur le début d'une séquence.

Moyens

Pour examiner les moyens qui vont nous permettre d'atteindre ces objectifs, nous allons d'abord spécifier les paramètres à fournir à cette machine, puis la façon dont elle interagit avec le monde extérieur avant d'exposer la méthode utilisée pour identifier les expressions.

Paramètres génériques

Pour utiliser cette machine, il faut préciser :

- La taille maximum d'une expression — par défaut, seules les expressions comportant moins de neuf mots sont reconnues.
- Une fonction qui permette de tester la transparence d'un objet.
- Une procédure dite "procédure action" qui sera appelée si une expression figée potentielle a été identifiée, à la charge de cette procédure de tester l'existence de l'expression dans les lexiques et d'agir ensuite en conséquence. Dans ce cas, cette procédure sera appelée avec les paramètres suivants :
  - La suite de mots identifiée comme une expression figée potentielle.
  - La liste des mots transparents — c'est-à-dire des mots qui peuvent être insérés librement dans les expressions, comme par exemple des adverbes — de cette suite de mots.

Interaction avec le monde extérieur

Cette machine agit sur le monde extérieur grâce à la procédure action. Cette procédure est appelée après identification d'une expression figée potentielle. C'est cette procédure qui devra gérer l'accès au lexique, ce qui nous assure de l'indépendance de notre machine vis-à-vis des lexiques.

Le monde extérieur agit sur la machine en lui fournissant au fur et à mesure les matrices d'expressions[21] des objets rencontrés. Comme pour la machine précédente, il est également possible de revenir en arrière sur un objet sans perdre les résultats obtenus sur les objets précédents.

Méthode d'identification des expressions figées

Méthode classique

Pour reconnaître une expression constituée de plusieurs mots, la méthode la plus immédiate consiste à construire toutes les expressions possibles de deux mots contigus ou plus et à vérifier leur existence dans un lexique de formes figées.

Une telle méthode est bien évidemment très pénalisante à utiliser; prenons par exemple la phrase suivante :

Le cordon bleu cuisine.

21. Ce terme est défini dans la suite.

On voit qu'il faut effectuer 6 recherches pour être sûr que seule la séquence *cordon bleu* est potentiellement une expression figée (d'une manière plus générale, pour une séquence de n mots de long, il faut effectuer $n(n-1)/2$ recherches pour identifier toutes les expressions figées potentielles).

De plus, si l'on veut pouvoir reconnaître des expressions figées en autorisant des discontinuités — par exemple des adverbes comme dans *faire* ***vraiment*** *table rase* — cette méthode devient encore plus difficile à appliquer.

Méthode SDL

Notre machine utilise une méthode un peu plus sophistiquée : pour chaque mot, le lexique contient une matrice qui indique si ce mot peut être (ou ne pas être) le $n^{ième}$ mot d'une expression de l mots de long. C'est cette matrice qui est fournie à la machine. Une expression figée potentielle sera identifiée à chaque fois que la machine rencontrera une séquence de l mots qui sont respectivement 1^{er}, $2^{ème}$ ... $l^{ième}$ mots d'une expression de l mots de long.[22]

Cette méthode permet de reconnaître des expressions totalement figées comme *cordon bleu*, des expressions semi-figées comme *faire table rase* où le verbe *faire* peut être fléchi, et des expressions avec insertions comme *faire* ***vraiment*** *table rase*. Une fois que l'expression a été reconnue comme une expression figée potentielle, on peut soit décider de la considérer comme certaine (typiquement la séquence de mots *au fur et à mesure* est toujours une expression figé), soit la considérer comme ambiguë (comme par exemple pour la séquence *cordon bleu* qui peut être ou ne pas être figée). En pratique, cette distinction est faite préalablement lors de la constitution du lexique.

Pour les expressions avec insertion, la machine gère un décalage dans la matrice à chaque fois qu'il rencontre un mot qui peut être inséré de manière transparente dans une expression figée.

Il existe deux autres types d'expressions non encore traitées :

— Les expressions semi-figées du type *casser* $Poss^0$ *pipe* où la personne du pronom $Poss^0$ doit être accordée avec le sujet. Pour ce type d'expression, il est envisageable de les reconnaître comme des expressions figées sans contraintes mais d'avoir dans le lexique des règles de bonne formation.

— Les expressions discontinues : *pousser Nhum à bout*. Pour ces expressions, la difficulté est de définir ce qu'il peut y avoir dans le trou. A l'heure actuelle, la machine ne reconnaît pas ces expressions.

Exemple

Considérons par exemple les deux expressions suivantes :

petit à petit

à pas de loup

22. D'après une idée originale de Morris Salkoff.

On voit que *petit* peut être 1er et 3ème mot d'une expression de 3 mots de long, *à* peut être 2ème mot d'une expression de 3 mots de long et 1er mot d'une expression de 4 mots de long et enfin que *pas*, *de* et *loup* peuvent être respectivement 2ème, 3ème et 4ème mot d'une expression de 4 mots de long, ce qui nous donne les matrices suivantes[23] :

petit :

	Rang			
Longueur	1	2	3	4
1				
2				
3	X		X	
4				

à :

	Rang			
Longueur	1	2	3	4
1				
2				
3		X		
4	X			

pas :

	Rang			
Longueur	1	2	3	4
1				
2				
3				
4		X		

de

	Rang			
Longueur	1	2	3	4
1				
2				
3				
4			X	

23. Les matrices sont seulement de dimension 4 pour améliorer la lisibilité.

loup :

	Rang			
Longueur	1	2	3	4
1				
2				
3				
4				X

La séquence *petit à petit à pas de loup* sera traitée comme suit :

petit	*à*	*petit*	*à*	*pas*	*de*	*loup*
X(3)	X(3)	X(3)				
	X(4)					
		X(3)	X(3)			
			X(4)	X(4)	X(4)	X(4)

Un X (n) dans la i^{ème} case à partir de la gauche signifie que le mot peut être le i^{ème} mot d'une expression de n mots de long.

On ne teste l'existence d'une expression dans le lexique que si l'on trouve n X(n) successifs. Dans notre exemple, seules les cases en gras ont provoqué une recherche dans le lexique. Pour gérer des mots transparents, il suffit d'autoriser des décalages dans les séquences de X(n). Cette méthode limite énormément le nombre de recherches dans le lexique et permet ainsi de gérer un grand nombre d'expressions figées.

Enfin, dans le cas d'expressions certaines se chevauchant en partie, notre stratégie consiste à choisir l'expression certaine la plus longue. Cette méthode est arbitraire et peut conduire à des erreurs, mais de toute façon, il s'agit d'un cas douteux dans la mesure où, par définition, deux expressions certaines ne devraient jamais se chevaucher. En fait, cette méthode permet simplement de remédier à certaines erreurs du lexique.

Conclusion

Nous avons choisi de décrire ces deux machines dans un chapitre à part car elles sont relativement indépendantes du reste de Tomas, qui les utilise comme des "boîtes noires". De plus, du point de vue du génie logiciel, il est tout à fait bénéfique d'isoler ainsi, quand c'est possible, certains modules du reste du projet de manière à pouvoir les réutiliser ultérieurement.

VII. Les différents modules de Tomas

Ce chapitre décrit en détail les six modules de Tomas :

① Analyse lexicale du texte.
② Etiquetage des mots.
③ Première reconnaissance des expressions figées et construction de la structure en graphe de la phrase.
④ Découpage en propositions.
⑤ Expansion des homographies et des mots contractés
⑥ Levée des homographies (en appliquant les règles locales) et reconnaissance des expressions semi-figées.

Précisons comment l'information transite d'un module vers le suivant. Le premier module transforme le texte en une suite d'éléments lexicaux. Le deuxième module étiquette ces lexèmes en consultant le lexique de Tomas. Il produit donc des mots décorés (c'est-à-dire munis de leurs informations syntaxiques). C'est également ce module qui détermine le découpage en phrases des textes.

A partir de ces mots décorés, le troisième module construit le graphe simple de la phrase. A ce stade, les nœuds correspondent à des mots ou à des expressions figées. Ce graphe est fourni en entrée au quatrième module qui se charge du découpage en propositions et fabrique donc le graphe complexe de la phrase.

Enfin, les deux derniers modules raffinent ce graphe complexe. Le cinquième module effectue l'expansion des homographes et des mots contractés tandis que le sixième supprime les chemins interdits et réduit ainsi les homographies. On obtient donc un graphe complexe de phrase en sortie de Tomas.

Nous nous attacherons à décrire en détail pour chaque module les objectifs et le mode de fonctionnement, en joignant des exemples d'applications.

1. Analyse lexicographique

L'analyse lexicographique est la première étape de toute analyse du langage écrit. Cette étape permet d'obtenir une suite de lexèmes à partir du flot de texte en entrée.

Dans Tomas, ce travail est effectué grâce à l'instantiation d'une "machine" CCCP. CCCP est un ensemble de programmes écrits en Ada qui permettent de faciliter la réalisation d'un compilateur en simplifiant l'écriture des analyseurs lexicaux et syntaxiques[24]. Pour l'étape lexicographique, nous n'utiliserons que la première partie de CCCP, à savoir l'analyseur lexical.

Pour réaliser un analyseur lexical, il faut d'une part spécifier les lexèmes que l'analyseur doit identifier, et d'autre part les règles de formations de ces mêmes lexèmes à partir des éléments de bases que sont les caractères du français.

a. Les lexèmes

Tomas reconnaît les lexèmes suivants : Un_Retour_A_La_Ligne, Une_Fin_De_Paragraphe, Une_Ponctuation , Un_Nombre, Un_Mot, Un_Mot_Capitalise, Un_Mot_En_Majuscules, Un_Sigle et Un_Inconnu. Ces lexèmes sont détaillés dans la figure 1.

b Règles de formations

Les règles de formations des lexèmes sont décrites dans un langage externe et sont donc modifiables à volonté. D'une manière générale, nous considérons qu'un mot est un ensemble de lettres délimité, de part et d'autre, par des séparateurs (par exemple l'espace ou le saut de ligne) ou par des signes de ponctuation (par exemple la virgule ou le point). Il faut toutefois considérer deux cas particuliers le tiret (-) et l'apostrophe (').

Le tiret peut être utilisé de quatre manières différentes dans la langue française, à savoir :

— En début de ligne, pour signaler une partie d'énumération (comme sur cette page). Il s'agit dans ce cas d'une ponctuation.

— Pour mettre une partie de la phrase en apposition, comme dans :

Les experts - à notre connaissance - sont unanimes [25]

Dans ce cas, le tiret est encore une ponctuation.

— Pour lier deux mots afin de former un mot composé, comme :

garde-manger

---

24. Le terme syntaxique est utilisé ici dans son sens informatique, c'est-à-dire que CCCP facilite l'écriture de d'analyseurs lexicaux et syntaxiques de langages formels.

25. Le bon usage voudrait que l'on utilise un tiret long (—) au lieu d'un tiret court (-), mais ceci n'est pas toujours possible, ne serait-ce que parce que peu de traitement de textes font la distinction.

— En fin de ligne, pour indiquer une césure. Dans ce cas, l'analyse lexicographique doit reconstituer le mot de manière à rendre ce phénomène invisible pour la suite du traitement.

Classe lexicale	Description
La_Fin_Du_Fichier	Il s'agit d'un lexème de contrôle qui signale, par définition, la fin du flot de texte en entrée.
Un_Retour_A_La_Ligne	Ce lexème de contrôle signale une fin de ligne.
Une_Fin_De_Paragraphe	Ce lexème de contrôle signale une fin de paragraphe. Dans la lexicographie française, toute succession de deux sauts de ligne ou plus est considérée comme une fin de paragraphe.
Une_Ponctuation	Ce lexème signale une ponctuation. dans la lexicographie française, les caractères suivants sont toujours des ponctuations : ! " ( ) , : ; et /. Les caractères . et - peuvent être des ponctuations ou des caractères normaux.
Un_Nombre	Ce lexème signale un nombre.
Un_Mot	Ce lexème signale un mot (par exemple : *ordinateur*, *l'*, *très*).
Un_Mot_Capitalise	Ce lexème annonce un mot qui commence par une majuscule (par exemple : *Le*, *Apple*).
Un_Mot_En_Majuscules	Ce lexème annonce un mot tout en majuscules (par exemple : *APPLE*).
Un_Sigle	Ce lexème annonce un sigle, c'est-à-dire une succession de lettres et de . (par exemple : *I.B.M*).
Un_Inconnu	Ce lexème correspond à une unité lexicale non identifiée.

FIGURE 1 : LES LEXEMES

Pour identifier et traiter ces quatre cas, nous utilisons l'heuristique suivante :

① Un tiret en fin de ligne, précédé d'une partie de mot et suivi, sur la prochaine ligne, d'une autre partie de mot (éventuellement lui-même précédé de délimiteurs) est un tiret de césure.

② Un tiret immédiatement encadré par des mots indique un mot composé.

③ Dans tous les autres cas, il s'agit d'un tiret de ponctuation.

Dans le cas de l'apostrophe, le problème principal est de décider si l'apostrophe doit être rattachée au mot qui la précède ou bien traitée comme un mot (ou même un signe de ponctuation) à part. Nous avons choisi la première méthode, c'est-à-dire que les séquences *l'enfant* et *aujourd'hui* seront chacune considérées comme deux mots :

l'enfant = *l'+enfant*

aujourd'hui =*aujourd'+hui*

Cette solution augmente les lexiques — de manière négligeable toutefois, puisqu'il s'agit au plus d'une vingtaine d'entrées — mais permet de résoudre l'élision sans calcul. En effet, s'il est très simple de voir que *j'* est dérivé de *je*, le cas de *l'*, qui dérive de *le* ou *la,* est moins simple. Il nous a donc semblé plus facile d'enrichir légèrement le lexique que de résoudre algorithmiquement le problème de l'élision.

Une fois les mots convenablement séparés, il reste à les identifier du point de vue grammatical. C'est le rôle de l'étape suivante.

2. Etiquetage et expressions figées

Cette étape est basée sur le lexique central de Tomas qui a été décrit dans le chapitre "Le lexique de Tomas". Elle peut être divisée en deux parties :

— Etiquetage des mots simples

— Identification et étiquetage des expressions figées

a. L'étiquetage

L'étiquetage consiste à rechercher pour chaque mot du texte les informations qui lui sont associées dans le lexique. Cette recherche s'effectue uniquement sur la graphie du mot et est donc très sensible à la manière dont le mot est écrit. Ceci nous amène à considérer les trois cas particuliers suivants :

— Les mots inconnus : les mots inconnus sont les mots absents du lexique. Cette absence peut être due soit à une mauvaise orthographe du mot, soit à une insuffisance du lexique. Dans les deux cas, le mot sera marqué comme un terme inconnu pour la suite du traitement.

— Les mots capitalisés en début de phrase : ces mots peuvent être des noms propres ou des mots communs. Dans l'ignorance, ils sont étiquetés comme des mots communs avec toutefois une prise en compte de la majuscule. En effet, le mot *A* ne peut pas être étiqueté comme le mot *a* :

A = verbe *avoir* ou préposition *à* (ou même la lettre *a*)

a = verbe *avoir* (ou même la lettre *a*)

Enfin, dans l'hypothèse où le mot serait absent du lexique, il sera étiqueté comme un nom propre.

— Les noms propres : ces noms sont repérés par leur majuscule (en dehors du début de phrase, cf. supra). Une succession de noms propres sera considérée comme un nom propre, par exemple *Le Mans* ou *San Francisco*. Le lexique est consulté afin de tester la présence d'éventuelles expressions figées (cf. infra) comme dans *calissons d'Aix*, *bêtises de Cambrai* ou *Notre Dame de Paris*.

Un dictionnaire du texte est constitué au fur et à mesure de l'étiquetage grâce aux mots rencontrés. Ce dictionnaire est généralement suffisamment petit pour être placé en mémoire et présente l'avantage d'accélérer substantiellement l'étiquetage dans la mesure où il sera consulté avant le lexique central de Tomas. Il sert en quelque sorte d'antémémoire et est donc conceptuellement transparent.

Une fois cet étiquetage effectué, il reste à identifier les expressions figées présentes dans la phrase en cours d'analyse.

b La reconnaissance des expressions figées

Pour la reconnaissance des expressions figées, Tomas utilise la machine du même nom décrite dans le chapitre VI, "Les machines de Tomas". Les matrices d'expressions sont liées à la graphie et la procédure action effectue la recherche des expressions potentielles dans le lexique. Si l'expression potentielle ne s'y trouve pas, le traitement s'arrête là. Si l'expression est présente dans le lexique, elle est étiquetée puis insérée dans le graphe de la phrase. Dans le cas où l'expression n'est pas ambiguë, les chemins parallèles sont détruits.

c. Résultats

La phrase suivante *L'écran que l'ordinateur affiche intéresse les ingénieurs support* donnera lieu à l'étiquetage suivant :

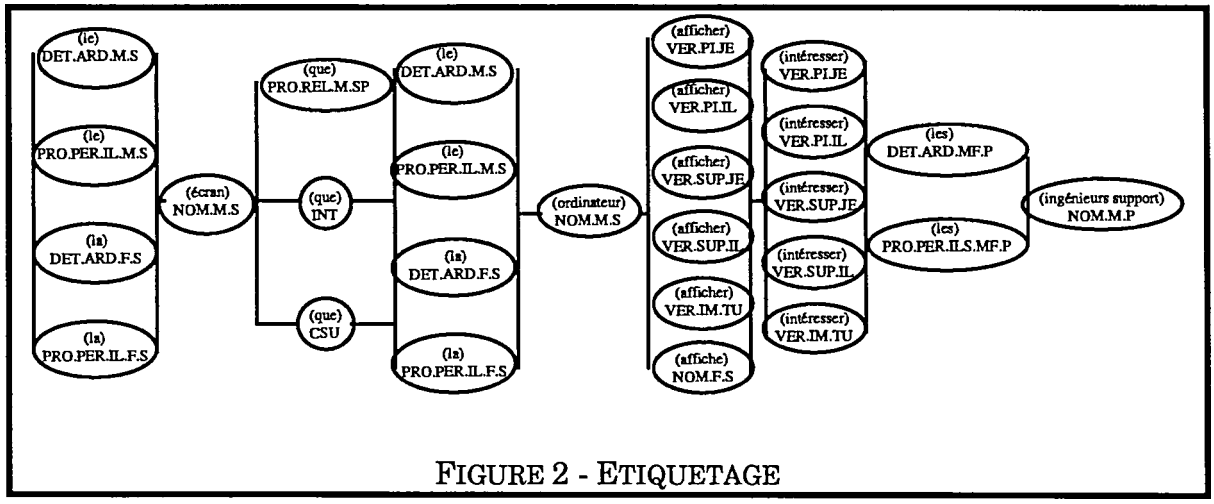


FIGURE 2 - ETIQUETAGE

3. Découpage en proposition

Cette étape consiste à identifier les propositions de la phrase de manière à obtenir un graphe complexe où chaque proposition sera représentée comme un nœud à l'intérieur du graphe de la proposition principale.

Il est indispensable d'effectuer cette opération avant d'appliquer les règles de levées d'homographies dans la mesure où la juxtaposition des propositions peut entraîner l'apparition de séquences impossibles à l'intérieur d'une proposition. Par exemple, dans la phrase suivante on remarque la séquence verbe conjugué-verbe conjugué :

*La pomme que je **mange est** bonne.*

Or c'est justement sur l'identification de ce type de séquence impossible[26] que s'appuie la levée des homographies. Remarquons également que si l'on n'avait pas déjà isolé les expressions figées comme *coupe-coupe* ou *pousse-pousse*, on serait confronté à des séquences de deux verbes conjugués dans une même proposition.

La méthode d'identification des propositions ne sera pas explicitée ici dans la mesure où elle a été développée indépendamment par Béatrice Pelletier et Alain Guillet. Il nous a cependant paru nécessaire de la présenter rapidement ici car elle est indispensable pour l'analyse de phrases complexes. Deux points méritent particulièrement d'être développés : la limitation des ambiguïtés et les résultats obtenus.

26. C'est-à-dire impossible à l'intérieur d'une proposition ; dans le cas d'une énumération verbale, les verbes sont séparés par des virgules ou des conjonctions de coordination : *Paul mange, boit et rit.*

a. Limitation des ambiguïtés

Afin de limiter la taille du problème, il est possible de ne considérer que six classes syntaxiques :

— Les verbes conjugués
— Les subordonnants
— Les prépositions
— Les adverbes
— Les coordonnants *et*, *ou*, et virgule (,)
— Les autres mots

Par exemple, en regroupant les homographies suivant ces six catégories, on obtient à partir du graphe de la figure 2 le graphe suivant :

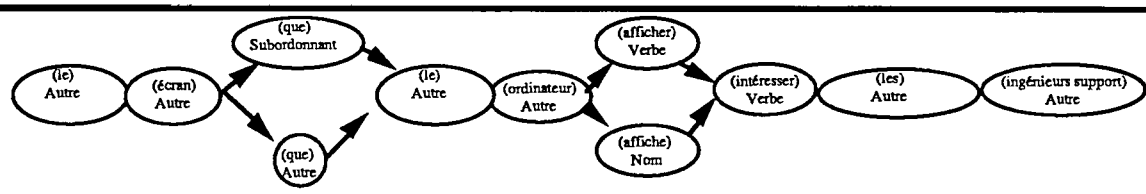


FIGURE 3 - GRAPHE FACTORISE

b Résultats

Le résultat de ce découpage est un graphe de phrase complexe où, comme nous l'avons déjà dit, les propositions sont représentées par des nœuds.

De plus, certaines homographies sont déjà réduites, en particulier les homographies de type Verbe conjugué/Autre et Subordonnant/Autre.

Par exemple, notre phrase *L'écran que l'ordinateur affiche intéresse les ingénieurs support* donnera lieu au graphe complexe de la figure 4.

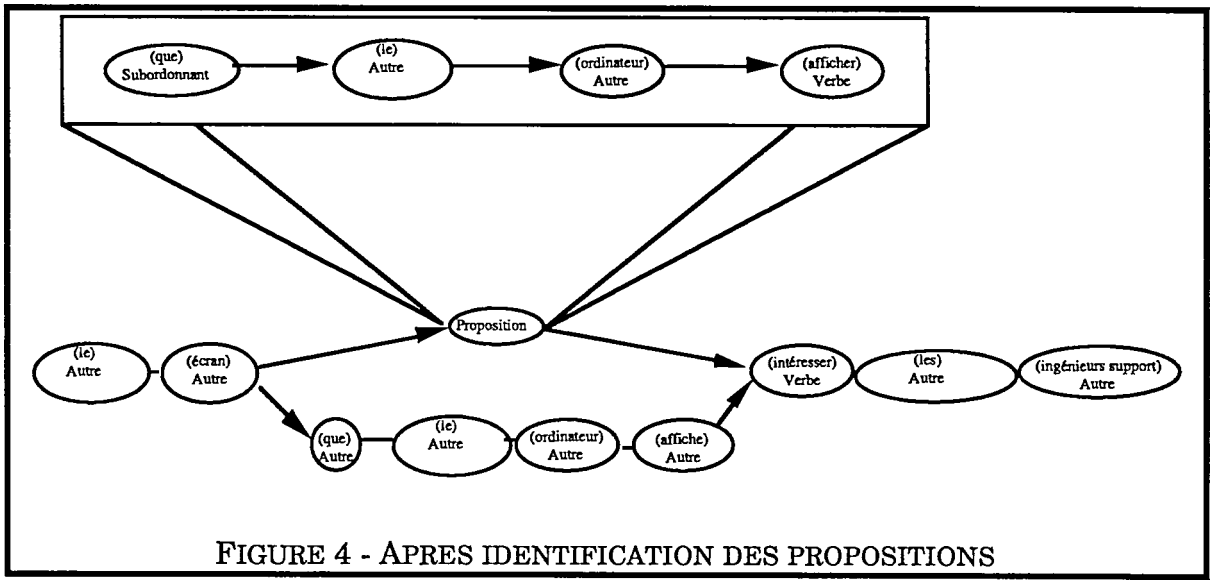


FIGURE 4 - APRES IDENTIFICATION DES PROPOSITIONS

4. Expansion de la phrase

A ce stade, la phrase est un graphe complexe dans lequel les homographies sont factorisées en six classes (cf. supra). Pour pouvoir continuer notre analyse et en particulier appliquer les règles de levée des homographies, il faut d'une part développer ces homographies et d'autre part dilater certains mots contractés.

a. Développement des homographies

Pour la levée des homographies, il faut développer chacune de ces six classes de manière à obtenir un nouveau graphe avec toutes les homographies de la phrase. Il y a cependant une exception : les homographies limitées aux verbes conjugués ne sont pas développées. En effet, pour pouvoir résoudre les problèmes d'accord et de temps des verbes, il est nécessaire d'identifier le sujet, le mode de la phrase et le temps du discours, ce qui est en dehors des objectifs (et des capacités) de Tomas.

Ce développement des homographies s'effectue simplement en combinant un parcours aléatoire de la phrase et une opération d'insertion dans la phrase (cf. chapitre IV, "Structure de la phrase").

b Traitement des mots contractés

Ce traitement à pour but de rétablir certains mots contractés dans leur forme d'origine. En effet, il existe en français des mots qui sont la contraction d'autres mots. Par exemple :

— *Au* et *aux* sont respectivement les contractions de *à le* et *à les*, *le* et *les* étant dans ce cas des déterminants.

— *Des* peut être soit un article indéfini (ou partitif) comme dans *des hommes sont venus* qui est la forme plurielle de *un homme est venu*, ou la contraction de *de les* (*de* préposition, *les* déterminant) comme dans *je viens des montagnes* qui est la forme contractée de *je viens de les montagnes.*

Ces mots sont :

au	⇒ à (Préposition) le (Déterminant)
aux	⇒ à (Préposition) les (Déterminant)
du	⇒ de(Préposition) le (Déterminant)
des (Préposition)[27]	⇒ de(Préposition) les (Déterminant)
auquel	⇒ à (Préposition) lequel (Pronom)
auxquels	⇒ à (Préposition) lesquels (Pronom)
auxquelles	⇒ à (Préposition) lesquelles (Pronom)
duquel	⇒ de (Préposition) lequel (Pronom)
desquelles	⇒ de (Préposition) lesquelles (Pronom)
desquels	⇒ de (Préposition) lesquels (Pronom)

27. Nous considérons que l'article partitif *des* n'est pas une contraction.

Cette opération s'effectue simplement en combinant un parcours aléatoire de la phrase et une opération de substitution dans la phrase (cf. le chapitre IV "Structure de la phrase"). Le traitement des mots contractés permet d'uniformiser les règles de levée des homographies appliquées dans la suite. De plus, cette étape d'expansion est indispensable pour l'analyse d'autres langues comme l'italien, où des pronoms peuvent être accolés aux verbes infinitifs et impératifs, ou l'allemand qui fabrique des mots composés par agglutination.

c. Résultats

Après expansion, notre phrase *L'écran que l'ordinateur affiche intéresse les ingénieurs supports* sera représentée par le graphe complexe de la figure 5.

5. Levée des homographies

La levée des homographies est l'étape centrale de Tomas. Toutes les étapes précédentes ont transformé le flot de textes en entrée en un graphe de phrase de manière à pouvoir appliquer le plus efficacement possible les règles de levée des homographies.

a. Type de règle

Ces règles sont des règles d'identification des séquences impossibles. Le principe est qu'une fois toutes les impossibilités éliminées, ce qui reste dans le graphe correspond aux analyses valides de la phrase. Ces règles sont donc des règles négatives : elles ne décrivent pas les analyses valides d'une phrase mais bien les séquences impossibles. De plus, ces règles sont des règles certaines, qui ne nécessitent pas de considérer les homographies : en effet sur un chemin du graphe, la partie déjà visitée est connue.

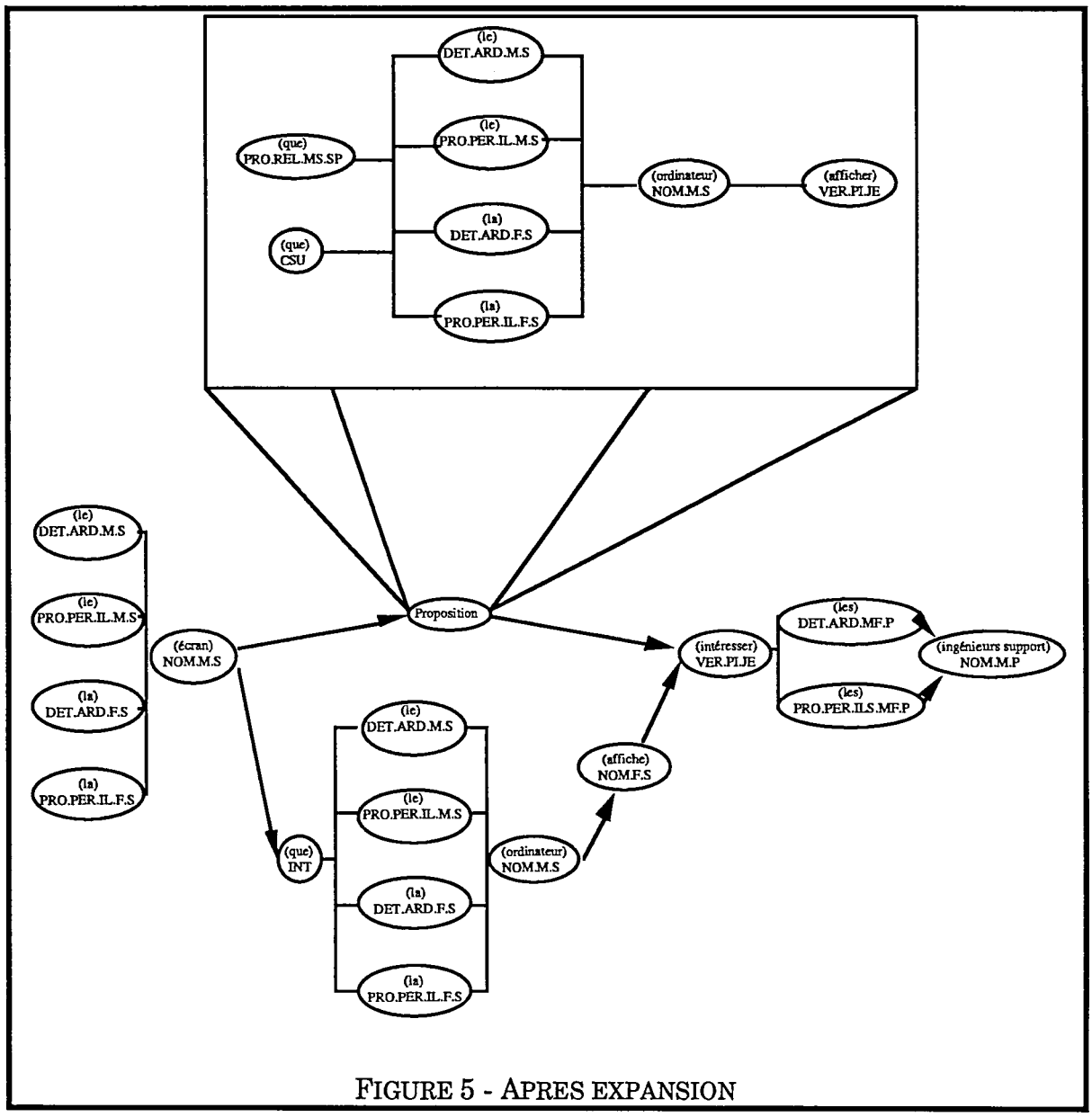


FIGURE 5 - APRES EXPANSION

b Application de ces règles

Pour appliquer ces règles, il faut résoudre trois difficultés :

— Parcours de la phrase
— Identification des structures impossibles
— Destruction de ces structures

Remarquons immédiatement que les deux derniers points ont déjà été traités. L'identification des structures est effectuée grâce à la machine générique de reconnaissance d'objets décrite dans le chapitre "Les machines de Tomas" et le mécanisme de destruction d'une structure est explicité dans le chapitre "Structure de la phrase". Reste le premier point que nous allons examiner dans la suite.

Parcours de la phrase

Le parcours dans la phrase doit répondre à deux impératifs :

— Visiter toutes les séquences susceptibles d'être impossibles.

— Limiter l'explosion combinatoire.

Pour visiter toutes les séquences susceptibles d'être impossibles dans le phrase, seul le parcours en profondeur est possible car il respecte l'ordre des mots dans la phrase. Ce mode de parcours présente toutefois un inconvénient : un même nœud peut être visité un grand nombre de fois. En effet, considérons l'exemple suivant :

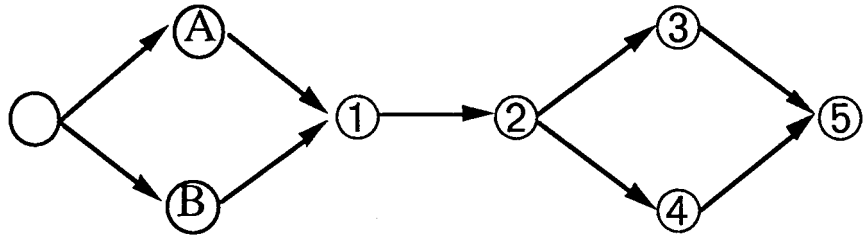


FIGURE 6 - UN GRAPHE

Les nœuds 1, 2, 3 et 4 seront explorés deux fois, une fois en passant par le chemin contenant le nœud A et une deuxième fois en passant par le chemin contenant le nœud B. Quant au nœud 5, il sera visité quatre fois. Cette exploration multiple est parfois nécessaire : en effet, si l'on suppose qu'il existe une règle applicable à la séquence B1, alors la deuxième visite du nœud 1 était indispensable.

Ceci nous a amené à imaginer un type de parcours original basé sur l'exploration en profondeur qui évite au maximum de visiter plusieurs fois les nœuds.

Le principe est le suivant :

- Chaque nœud du graphe est marqué comme étant non visité.
- L'exploration en profondeur commence au nœud initial de la phrase.
- Pour chaque nœud atteint, deux cas se présentent :
  - — Le nœud est déjà marqué comme ayant été visité. Dans ce cas, on inhibe l'application de nouvelles règles et on continue l'exploration tant qu'il y a encore de règles issues des nœuds précédents succeptibles de s'appliquer.
  - — Le nœud n'est pas marqué comme ayant été visité. Dans ce cas, on autorise l'application de nouvelles règles et on marque le nœud comme ayant été visité.

Pour simplifier, ce type de parcours est basé sur le fait que si un nœud a déjà été visité, tous les nœuds qui le suivent ont également été visités. Ceci implique que toute séquence qui commence après ce nœud a déjà été testée précédemment et donc que lorsque les séquences débutant strictement avant ce nœud auront été testées il sera inutile d'aller plus loin.

c. Résultats

Après levée des homographies, notre phrase se présente comme suit :

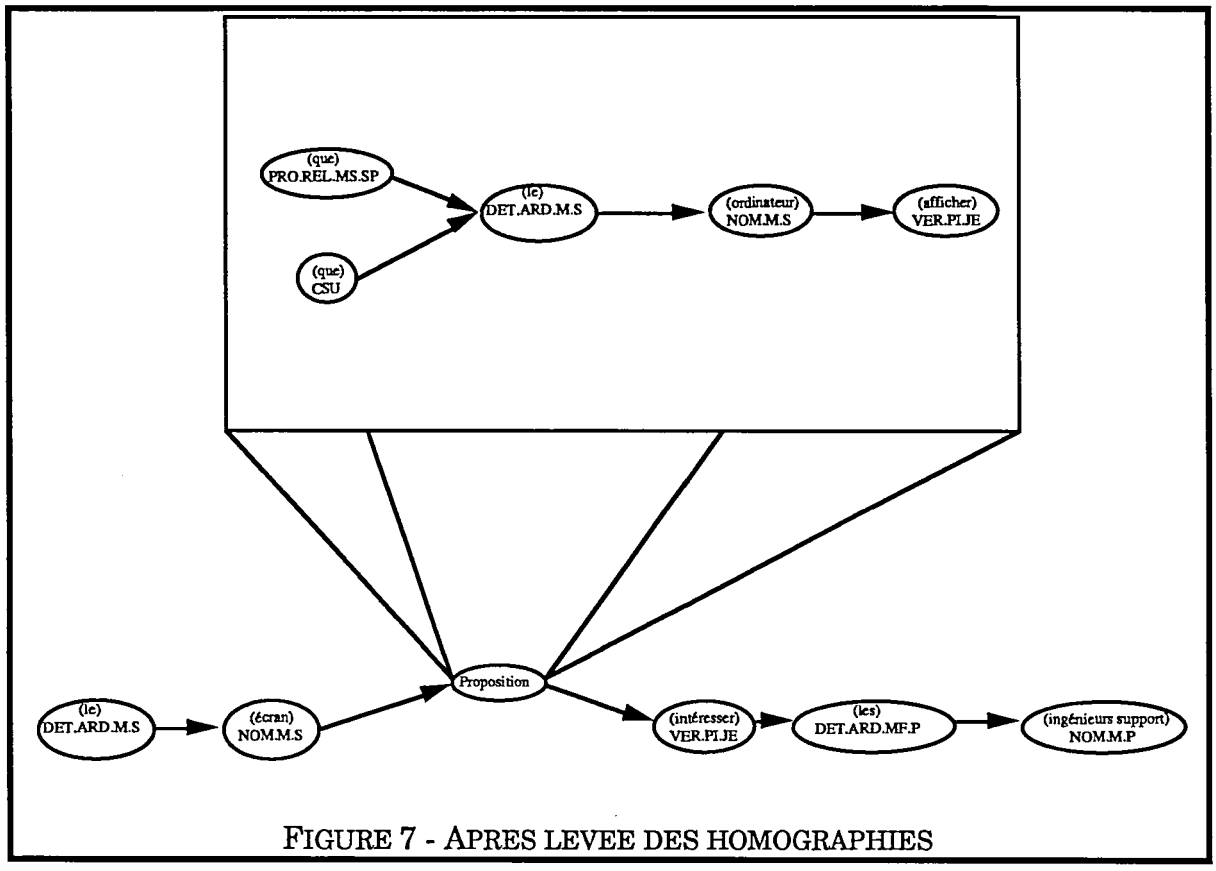


FIGURE 7 - APRES LEVEE DES HOMOGRAPHIES

Les règles suivantes ont été appliquées :

— Un pronom n'est jamais suivi d'un nom, ce qui élimine les séquences suivantes :

(le)PRO.PER.M.S.COD + (ordinateur)NOM.M.S

(la)PRO.PER.F.S.COD + (ordinateur)NOM.M.S

(le)PRO.PPV2.PER.ILS.MF.P.COD + (ingénieurs support)NOM.M.P

— Les accords en genre et en nombres non respectés éliminent les séquences suivantes :

(la)DET.ARD.F.S + (ordinateur)NOM.M.S

(la)DET.ARD.F.S + (écran)NOM.M.S

La première règle (un pronom n'est jamais suivi d'un nom) peut sembler fausse, comme le montre l'exemple suivant :

regarde-le, crétin !

Toutefois, le tiret entre le verbe et le pronom et la virgule entre le pronom et le nom empêcheront la règle de s'appliquer dans ce cas.

6. Reconnaissance des expressions semi-figées

Rappelons que par "expression semi-figée", nous entendons une expression figée dont un ou plusieurs mots peuvent être fléchis, comme *prendre le taureau par les cornes* ou *faire table rase*.

La reconnaissance des expressions semi-figées ne peut se faire que durant cette phase dans la mesure où il est nécessaire de connaître les lemmes des mots pour valider ces expressions. En effet, pour savoir si la séquence *fait table rase* est bien une forme du verbe complexe *faire table rase*, il est nécessaire de s'assurer que la graphie *fait* a bien pour lemme le verbe *faire* et non le nom *fait*. Ce problème ne se posait pas dans le cas des expressions totalement figées, ce qui explique que ces deux types d'expression ne soit pas traités dans le même module.

Comme dans le cas des expressions totalement figées, le traitement est effectué par la machine générique de reconnaissance d'expressions figées décrite dans le chapitre "Les machines de Tomas". La seule différence porte sur les matrices d'expressions figées : au lieu d'utiliser des matrices liées à la graphie, nous utiliserons cette fois des matrices liées aux lemmes. Une fois l'expressions identifiée, l'insertion dans le graphe de la phrase se fait soit par substitution d'un chemin si l'expression est non ambiguë, soit par ajout d'un chemin dans le cas d'une expression ambiguë. Ces deux opérations ont déjà été décrites dans le chapitre "Structure de la phrase".

VIII. Configuration de Tomas

Comme nous l'avons vu précédemment, les données linguistiques sont clairement séparées du traitement algorithmique. Ces données peuvent être fournies par l'utilisateur sous forme de fichiers textes, ces fichiers étant ensuite compilés avant d'être utilisés par Tomas[28].

Tomas utilise des données externes dans les modules suivants :

- — Analyse lexicographique : Description des règles de formation des unités lexicales.
- — Lexiques : Définitions des lexiques de mots simples et composés utilisés lors de l'étiquetage et de la reconnaissance des expressions figées.
- — Mots contractés : Description des règles d'expansion des mots contractés (par exemple : *au* ou *du*).
- — Levée des homographies : Description des règles de levée d'homographie.

Nous allons détailler dans la suite les caractéristiques de ces quatre fichiers de données.

1. Analyse lexicographique

L'analyse lexicographique permet de découper le texte en unités lexicales (mots, ponctuations, lexèmes spéciaux, etc.). Le problème de l'analyse lexicographique est loin d'être simple, comme nous l'avons déjà montré dans le chapitre précédent. Il suffit, pour s'en assurer, de considérer les exemples suivants :

- — Traitement des caractères spéciaux :
  - — Le tiret (-), qui peut être soit un signe typographique, soit une partie de mot composé, comme dans *grand-mère*.
  - — L'apostrophe comme dans *aujourd'hui* ou *l'*.
- — Les sigles et abréviations, qui mélangent des lettres et des points, comme dans : *I.B.M.*, *R.A.T.P.*, *Adm.*
- — Les mots mélangeant majuscules et minuscules comme dans : *TenNet*, *MsDos*.

De plus, les règles lexicographiques peuvent changer suivant le domaine — un télex n'est pas formaté comme un article de journal — et bien sûr suivant la langue, il suffit de considérer le β en allemand ou l'adjonction de 's pour le possessif en anglais.

Pour une discussion plus complète du problème de l'analyse lexicographique, on pourra utilement se reporter à [Silberztein 89].

Toutes ces raisons nous ont donc amené à externaliser la description lexicographique.

28. La compilation de ces fichiers se fait grâce à CCCP.

2. Lexiques

L'étiquetage et les reconnaissances d'expressions figées s'effectuent à partir d'un lexique intitulé Lexique Central de Tomas (LCT). Ce lexique est constitué à partir de fichiers textes qui sont compilés et placés dans des fichiers indexés pour des raisons de performances (cf. le chapitre "Le lexique de Tomas").

Le format de ces fichiers textes, que nous allons exposer dans la suite, est relativement proche du DELAF (cf. [Courtois 84,89]).

Ces fichiers textes peuvent contenir des mots ou des expressions figées. Chaque entrée commence soit par un caractère '%', soit par un caractère '+'. Le caractère '+' signale qu'une expression figée est non ambiguë. Les catégories syntaxiques sont séparées par des points (.) et les différentes fonctions d'une même graphie sont séparées par le caractère deux points (:) si elles ont des catégories en commun et par le caractère slash (/) sinon. Enfin, le lemme est placé entre parenthèses.

Les paragraphes suivants détaillent quelques exemples.

a. Définition d'une graphie

Cet extrait de lexique

```
%active : (actif)NOM.F.S:ADJ.F.S/(active)NOM.F.S/(activer)VER.PI.JE:IL
                                                    :SUP.JE:IL:IM.TU
```

définit les huit entrées suivantes :

active :	Nom Féminin Singulier ayant le mot *actif* pour lemme.
active :	Adjectif Féminin Singulier ayant le mot *actif* pour lemme.
active :	Nom Féminin Singulier ayant le mot *active* pour lemme.
active :	Verbe *activer* à la première personne du présent de l'indicatif.
active :	Verbe *activer* à la troisième personne du présent de l'indicatif.
active :	Verbe *activer* à la première personne du présent du subjonctif.
active :	Verbe *activer* à la troisième personne du présent du subjonctif
active:	Verbe *activer* à la seconde personne de l'impératif.

Le mot *active* sera donc étiqueté comme sur la figure 1. De la même manière, le mot *joue* qui avait été donné en exemple dans le chapitre V serait décrit ainsi :

```
%joue : (joue)NOM.F.S/(jouer)VER.PI.JE:IL:SUP.JE:IL:IM.TU
```

b Définition d'une expression figée ambiguë

La ligne suivante :

```
% cordon bleu : NOM.M.S
```

définit l'entrée lexicale d'un nom composé masculin singulier *cordon bleu* qui est ambigu, c'est-à-dire que la séquence *cordon bleu* conduit à un double étiquetage, comme on peut le voir sur la figure 2.

c. Définition d'une expression figée certaine

La ligne suivante :

```
|+ au fur et à mesure : ADV
```

définit l'entrée lexicale d'un adverbe composé *au fur et à mesure* qui n'est pas ambigu, c'est-à-dire que la séquence *au fur et à mesure* conduit à un étiquetage simple, quelles que soient les entrées définissant les mots de l'expression (cf. figure 3).

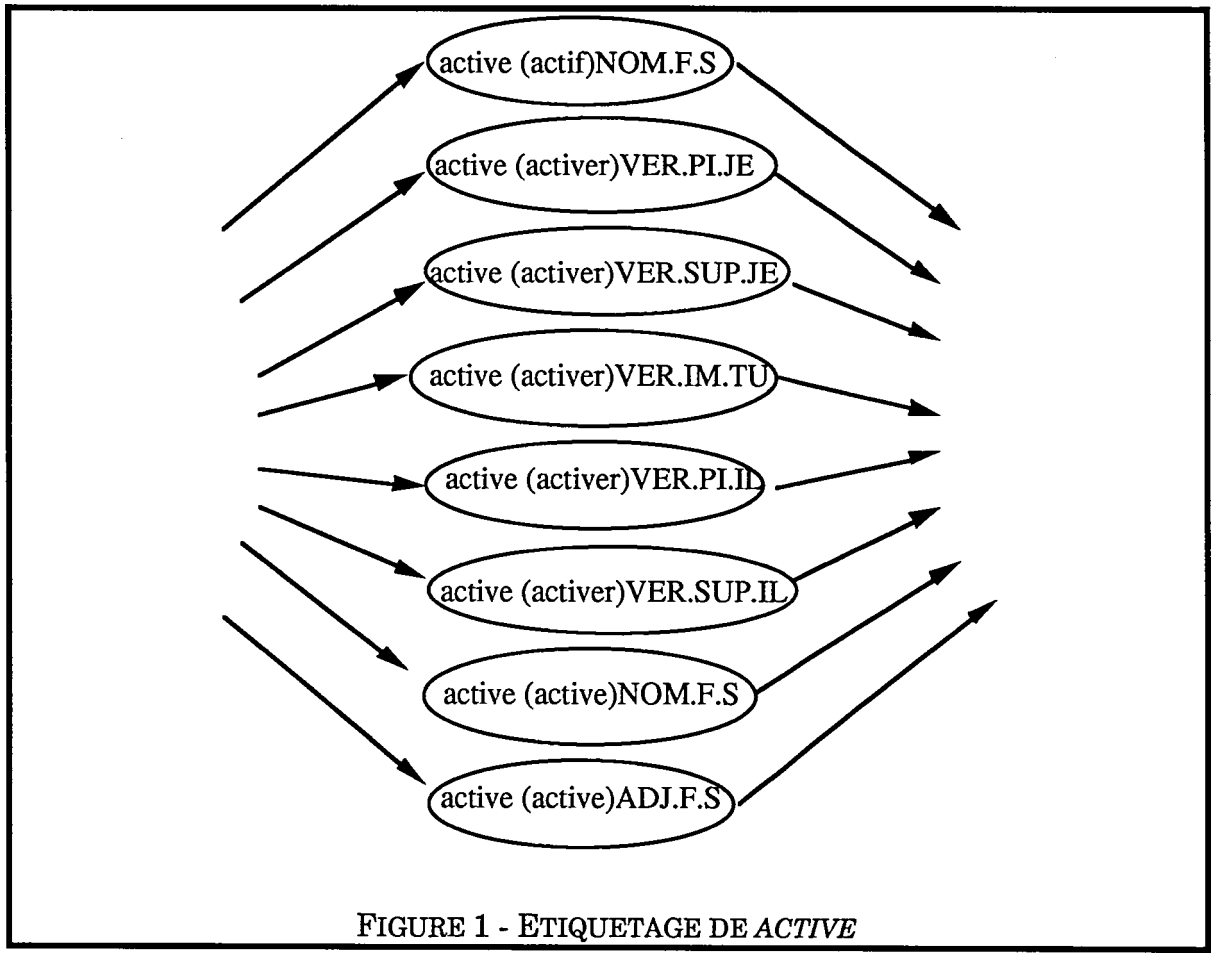


FIGURE 1 - ETIQUETAGE DE *ACTIVE*

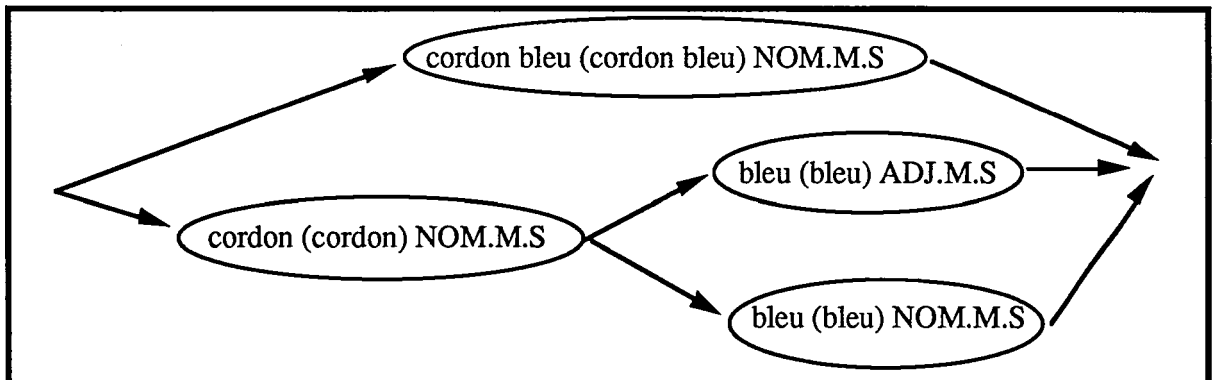


FIGURE 2 - ETIQUETAGE DE *CORDON BLEU*

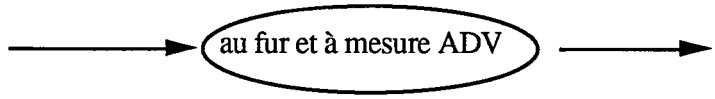


FIGURE 3 - ETIQUETAGE DE *AU FUR ET A MESURE*

d. Définition d'une expression semi-figée certaine

La ligne suivante :

| + (faire) table rase : VER

définit l'entrée lexicale d'un verbe composé non ambigu. Les parenthèses entourant le mot *faire* signalent que toutes les formes fléchies du lemme *faire* font partie de l'expression. Par exemple, la séquence *faisons table rase* sera étiquetée comme un verbe composé :

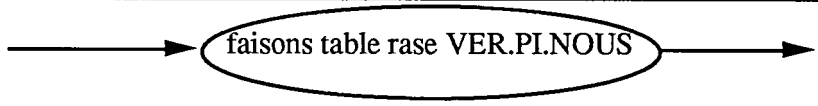


FIGURE 4 - ETIQUETAGE DE *FAISONS TABLE RASE*

On trouvera en annexe 2 des extraits de ce lexique.

3. Mots contractés

Nous avons déjà vu dans le chapitre précédent qu'il fallait parfois rétablir la forme d'origine de certains mots contractés.

Pour prendre en compte ce phénomène, il est possible de créer un fichier d'équivalence que Tomas utilisera après l'étiquetage simple.

Voici un extrait qui traite les cas de *au*, *aux* et *des* :

```
%[au:(au)DET.ARD.M.S] : [à(à)PRE] [le:(le)DET.ARD.M.S]
%[aux:(aux)DET.ARD.M.P] : [à(à)PRE] [les:(les)DET.ARD.MF.P]
%[des:(du)PRE] : [de:(de)PRE] [les:(les)DET.ARD.MF.P]
```

Cette méthode est plus pertinente qu'une simple expansion lexicale (de type des = de les) dans la mesure où elle permet de bien préciser les catégories syntaxiques des mots contractés et expansés.

4. Règles de levée des homographies

Rappelons que les règles de levée d'homographies permettent d'identifier les séquences interdites : ces règles sont fournies sous forme de fichiers textes qui sont ensuite compilés. La syntaxe de ces règles est la suivante :

- La première ligne du fichier doit indiquer le nom complet du fichier compilé contenant les règles, précédé d'une étoile ('*') et encadré de guillemets droits ('"').
- Deux tirets accolés l'un à l'autre définissent un commentaire qui se prolonge jusqu'à la fin de la ligne.
- Une règle doit se terminer par un point ('.').
- Les séparateurs (espaces, tabulations, saut de lignes...) ne sont pas pris en compte.
- Voici un exemple de fichier valide:

```
*"REGLES.HOM"

Det-Det      :    [DET] [DET] .

Accord-Nbr-1 :    [CAT = (ADJ,PPS,QUA,DET), NBR = S]
                  [CAT = (NOM,ADJ,PPS,QUA), NBR = P] .

Une-Regle :       [DET,PRE] [VER, TPS /= (IN,PPR)] .
```

1. La première ligne du fichier indique le nom du fichier compilé, ainsi que son chemin d'accès.
2. La première règle a pour nom *Det-Det* et elle interdit tout chemin formé de deux déterminants consécutifs[29].

29. Pour pouvoir interdire ainsi deux déterminants qui se suivent, il faut utiliser les catégories quantifieur et pronom pour les séquences suivantes :
les deux hommes: Det Qua Nom
l'un Det Pro

3. La deuxième règle s'interprète de la manière suivante: la première *contrainte [CAT=(ADJ, PPS, QUA, DET), NBR=S]* représente tout mot ayant l'une des catégories *ADJ*, *PPS*, *DET* ou *QUA* de nombre *S*. La deuxième *contrainte [CAT=(NOM, ADJ, PPS, QUA), NBR=P]* représente donc tout mot ayant l'une des catégories *NOM*, *ADJ*, *PPS* ou *QUA* de nombre *P*. Cette règle interdit tout chemin compatible avec ces deux contraintes. Il faut noter que les noms de catégorie ainsi que les parenthèses sont facultatifs s'ils sont suivis du signe '='. Ainsi cette règle peut se réécrire comme suit:

```
Accord-Nbr-1 :  [ADJ,PPS,QUA,DET, S] [NOM,ADJ,PPS,QUA, P] .
```

Toutefois la première écriture a l'avantage d'être plus lisible, même si elle est moins compacte.

4. La troisième régle interdit tout chemin formé d'une part d'un mot ayant l'une des catégories *DET* ou *PRE*, d'autre part d'un mot ayant la catégorie *VER* et n'ayant pas l'un des temps *IN* ou *PPR*. La notation /= a été créée uniquement comme commodité d'écriture. Dans ce cas, le nom de la catégorie est obligatoire. Afin de clarifier les idées, cette règle peut aussi s'écrire de la façon suivante:

```
Une-Regle :     [DET,PRE]
                [VER, TPS = (PI,II,IF,PS,PC,SUP,SUI,IM)] .
```

Cette écriture, si elle est plus simple à comprendre, a l'inconvénient d'être plus lourde et impose de mentionner tous les temps qui n'intéressent pas la règle.

IX. Applications de Tomas

Après avoir présenté Tomas sur le plan technique, nous allons maintenant examiner de quelle manière il peut être utilisé. La première partie de ce chapitre présente les résultats obtenus en analysant quelques phrase issues de journaux informatiques. Nous verrons ensuite comment Tomas est employé pour la segmentation et l'analyse de phrase en français dans le cadre d'un projet de recherche interne de la société Cora. Enfin, nous discuterons des possibilités d'utilisation de TOMAS en conjonction avec un analyseur syntaxique.

1. Résultats

L'annexe 1 fournit les analyses d'une vingtaine de phrases extraites des journaux Le Monde Informatique et 01 Informatique. Ces phrases ont été saisies strictement comme elles se présentaient dans les journaux et constituent un échantillon représentatif de ce genre de littérature. On remarquera en particulier leur longueur variée : la phrase la plus courte ne comporte que trois mots alors que la plus longue en compte trente-huit. Pour la commodité du lecteur, ces phrases sont présentées sur la page suivante.

a. Imperfections des textes... et des lexiques.

La première constatation concerne l'étiquetage des mots. En effet, deux phénomènes apparaissent : les fautes d'orthographe et les erreurs de lexiques.

Les phrases analysées ne contiennent qu'une seule faute d'orthographe : dans les phrases n°2 et 3, la forme *taiwanais* est utilisée au lieu de la forme *taïwanais*.. De tels barbarismes sont traités comme des mots inconnus. La deuxième source d'étiquetage erroné se trouve dans le lexique. En effet, ces tests ont été réalisés avec le lexique du projet "transposition" (cf infra), lexique encore en cours de modification lors de ces tests. Deux types d'erreurs sont imputables aux lexiques :

— mots existants absents du lexique : ce cas ne se présente que dans la phrase n°4 où l'on voit que le verbe conjugué *présentera* est étiqueté comme un mot inconnu.

1. *Acer rachète Altos.*
2. *En acquérant Altos pour 94 millions de dollars, le constructeur taiwanais de micro-ordinateurs se dote d'une compétence en systèmes multipostes sous Unix.*
3. *Les firmes taiwanaises n'en finissent pas de grossir.*
4. *Lors d'infopro, ABC Informatique présentera le dernier né de ses logiciels, une gestion commerciale en 5 modules (facturation, stock).*
5. *Les filiales françaises de 3Com et Rank Xerox ont conclu un accord de commercialisation par lequel Rank Xerox intègre à son offre la famille des logiciels de réseaux d'entreprises 3Com, 3+ et 3Open.*
6. *Rank Xerox SA enrichit ainsi son offre en matière de réseaux locaux d'entreprises et s'ouvre au monde DLC/LLC, TCP/IP, ISO, NetBIOS et MSNet.*
7. *Cette offre permet ainsi à ses clients, qui n'ont pas les moyens d'accéder à des solutions XNS sous Ethernet, d'évoluer vers des solutions réseaux performantes.*
8. *Les produits 3Com seront diffusés par la force de vente grands comptes de Xerox et par son réseau de distribution indirecte, les Xerox Store notamment.*
9. *Ce sont donc 80 ingénieurs vente-système spécialisés dans les réseaux, 60 ingénieurs technicaux-commerciaux, 24 boutiques Xerox Store, 30 concessionnaires et 10 VAR que Rank Xerox investit sur le marché des réseaux locaux.*
10. *L'objectif est de vendre au moins 700 réseaux en 89 avec 4 ou 5 points de connexion par réseau et de prendre ainsi 5 % de l'offre réseau.*
12. *Depuis plus d'un an déjà, elles travaillent sur des réalisations communes de systèmes experts dans ce créneau du secteur tertiaire.*
13. *L'accord se traduit par une offre unique, des outils et des moyens communs.*
14. *Les deux sociétés comptent ainsi apporter aux compagnies d'assurances des solutions bien situées dans le contexte de leur informatique.*
15. *CMM Diffusion, c'est le bouquet.*
16. *Le distributeur - et éditeur - lyonnais décerne avec Bull les trophés d'or du meilleur logiciel à CTI, Scalinfor et CIEE.*
17. *Pour la quatrième fois consécutive CMM Diffusion, le premier distributeur Bull en région Rhône-Alpes, a décerné en décembre dernier les Fleurs d'Or du meilleur logiciel en présence d'Etienne Grand Jacques, directeur général de Bull Grande Diffusion.*

FIGURE 1 - LES PHRASES A ANALYSER

— entrées incomplètes dans le lexique ; ce cas est plus fréquent, en particulier dans les phrases suivantes :
  — phrase n°2
    • *dote* est étiqueté seulement comme un verbe alors qu'il peut également s'agir d'un nom.
    • *sous* est étiqueté seulement comme une préposition alors qu'il peut aussi s'agir d'un nom.
  — phrase n°4
    • *dernier né de* est considéré comme une expression figée. Il s'agit bien évidemment d'une erreur et l'expression est *dernier né*.
  — phrases n°5 et 8
    • *par* est étiqueté uniquement comme une préposition alors qu'il s'agit aussi d'un nom (dans le langage des golfeurs).
  — phrase n°7
    • *vers* est étiqueté seulement comme une préposition alors qu'il peut aussi s'agir d'un nom au pluriel.
  — phrases n°9 et 12
    • *sur* est étiqueté uniquement comme une préposition alors qu'il peut également s'agir d'un adjectif.
  — phrase n°12
    • *depuis* est étiqueté comme une préposition alors qu'il peut aussi s'agir d'un adverbe.

Remarquons enfin que le lexique est parfois presque trop complet et ne facilite pas toujours l'analyse : dans la phrase n°6, le mot *monde* est étiqueté non seulement comme un nom mais également comme une forme du verbe *monder* ; ce verbe est suffisamment éloigné du domaine de l'informatique pour ne pas apparaître dans des textes de ce domaine et introduit donc une homographie parasite

b Les mots composés dans les articles.

Dans ces articles, on constate un nombre élevé d'expressions figées : trente-quatre occurrences, soit une moyenne de deux expressions figées par phrase. La relative technicité des textes nous assure que les expressions figées ne sont pas ambiguës. Enfin, ces expressions figées sont majoritairement des noms composés. Ces deux derniers points sont particulièrement favorables pour l'analyse dans la mesure où de telles expressions figées fournissent des "points fixes" qui ne sont pas ambigus lexicalement.

c. Les homographies.

Les homographies sont relativement nombreuses, puisque l'on compte à peu près un mot homographe tous les trois mots. Par contre, le degré de ces homographies est faible : sur les 204 mots constituant le vocabulaire des phrases analysées, 156 ne sont pas homographes, 36 sont homographes deux fois, six sont homographes trois fois (*en, fois, informatique, pas, que* et *s'*) et six sont homographes quatre fois (*deux, intègre, la, l', lyonnais* et *quatrième*).

d. Les analyses.

L'objectif visé est atteint. En effet, il ne reste quasiment plus d'homographie une fois les phrases analysées. Plus particulièrement :

— phrase n°3 et 7
- l'homographie sur le mot *pas* n'est pas levée.

— phrase n°9
- les homographies sur les mots *que* et *boutique* ne sont pas levées, ce qui conduit à deux analyses :

que comme une conjonction de subordination et *boutique* comme un verbe

que comme un pronom relatif et *boutique* comme un nom (cette dernière analyse est la bonne).

— phrase n°13
- les homographies sur les mots *du* et *des* ne sont pas levées.

— phrase n°14
- les homographies sur les mots *bien* et *situées* ne sont pas levées.

— phrase n°16
- l'homographie portant sur le genre,le nombre et la fonction du mot *lyonnais* n'est pas levée (en partie à cause du tiret qui sépare les mots *éditeur* et *lyonnais...*)

— phrase n°17
- l'homographie portant sur le mot *de* n'est pas levée.

Remarquons enfin que dans tous les cas — sauf un — la bonne analyse se trouve parmi les analyses proposées, ce qui est particulièrement satisfaisant. Le seul cas douteux concerne la phrase n°4 dans laquelle le mot *module* est analysée uniquement comme un verbe alors qu'il s'agit ici d'un nom. Ceci s'explique par l'absence du verbe *présenter* dans le lexique au moment de l'expérience. En effet, *module* devient alors le seul mot susceptible de jouer le rôle du verbe dans cette phrase et se retrouve ainsi mal analysé.

2. Applications

a. Le projet transposition (M. Salkoff/ P. Pellegrini/ B. Pelletier).

L'objectif du projet "transposition" ([Pellegrini 91]) est de traduire partiellement des textes du français vers l'anglais : seuls les "concepts" des phrases sont traduits. La notion de "concept" recouvre grosso modo la notion de syntagme nominal avec quelques extensions pour les infinitives. Voici quelques exemples de concepts issus d'un corpus de textes du Monde Informatique :

une forte croissance (50 % par an)

Pierre Taussac, fondateur d'ABC Informatique et applemaniaque émérite

le premier stand de vente de micro-ordinateurs et de logiciels au Printemps

créer une société d'édition de logiciels

Principe

Le système de transposition est articulé en trois modules :

— Levée des homographies.
— Extraction des concepts.
— Traduction des concepts.

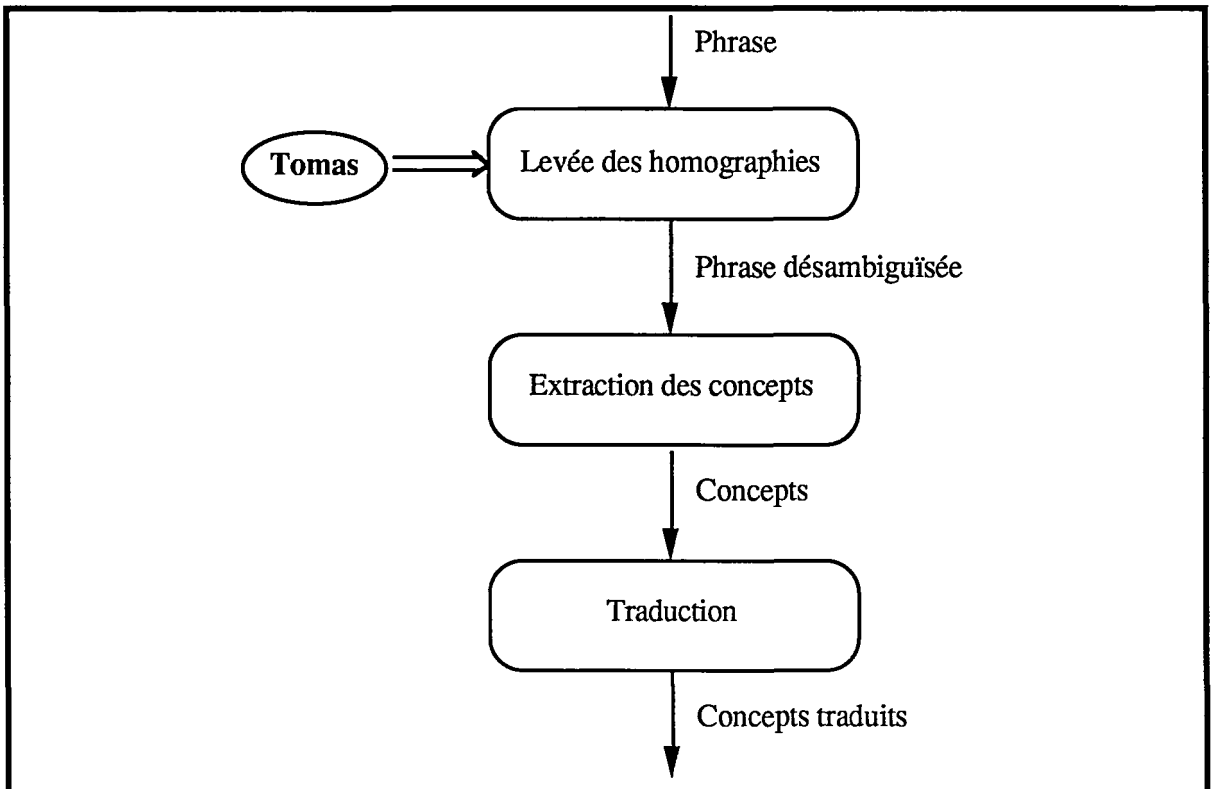


FIGURE 2 - LA TRANSPOSITION

Utilisation de Tomas

Tomas est utilisé pour la levée des homographies. Cette phase de désambiguïsation lexicale est nécessaire pour deux raisons :

— Pour l'extraction des concepts, il est nécessaire d'identifier les catégories syntaxiques des mots de la phrase, et en particulier les verbes et les noms. Une fois ce travail effectué, le repérage des concepts est effectué par une instantiation de la machine générique de reconnaissance d'objets présentée dans le chapitre "Les machines de Tomas".

— Lors de la traduction, il est fondamental de ne plus avoir d'ambiguïtés lexicales : en effet, le mot *parler* en tant que nom se traduit par *dialect* et en tant que verbe à l'infinitif par *to speak*, de même, le mot *personnel* en tant que nom se traduit par *staff* et par *personal* en tant qu'adjectif.

Résultats

Les résultats sont très satisfaisants, en effet sur un corpus de 2000 concepts issus d'une cinquantaine d'articles du Monde Informatique, on obtient :

- 80% de traductions correctes.
- 10% de traductions multiples (dont 95% contiennent la bonne traduction)
- 5 % de phrases mal traduites.
- 5 % de phrases non analysées, c'est-à-dire de phrases pour lequel Tomas ne fournit aucune analyse. Il est symptomatique de constater que 75 % de ces phrases sont agrammaticales (par exemple : *"au par pays"* ou *"la vente indirect."*).

Par exemple, les quatre concepts fournis plus haut sont traduits de la manière suivante :

a strong development (50 % per year)

Pierre Taussac, founder of highly skilled ABC Informatique and Applemaniac

the first sales stand of personal computers and of software to the Printemps

to create a publishing company of software

La seconde traduction est incorrecte mais dépasse les capacités du projet car il est extrêmement difficile de décider quelles sont les portées respectives de l'adjectif *émérite* et du mot *fondateur*. En effet, *émérite* peut qualifier le nom *applemaniaque,* ou les noms *ABC Informatique* et *applemaniaque*. De la même manière, *fondateur* peut s'appliquer à *ABC Informatique* uniquement ou bien à *ABC Informatique* et à *applemaniaque* à la fois. Tomas ne peut pas résoudre ce type de problème dans la mesure où il ne construit pas de représentations syntaxiques complexes de la phrase.

Les performances sont également satisfaisantes : le programme utilisé sur une station de travail de 0,9 Mips[30] traduit en moyenne une phrase à la minute.

b Frontal d'un analyseur automatique

Comme nous l'avons dit précédemment, l'objectif de Tomas n'est pas de parvenir à une analyse complète de la phrase. Toutefois, un tel objectif peut être atteint en couplant Tomas avec un analyseur syntaxique plus ambitieux. De cette manière, le travail de l'analyseur est grandement facilité dans la mesure où la plupart des ambiguïtés auront déjà été supprimées, ce qui permet à l'analyseur de se concentrer sur la construction d'une représentation cohérente de la phrase.

Deux autres arguments militent en faveur de l'utilisation de Tomas en tant que frontal d'un analyseur syntaxique : d'une part les phrases sont déjà découpées en propositions, et d'autre part les expressions figées présentes dans la phrase sont identifiées, ce qu'à l'heure actuelle peu d'analyseurs font.

30. C'est-à-dire moins puissant qu'un simple PC-386...

Il serait par exemple très intéressant de combiner Tomas avec l'analyseur en chaînes du français de Morris Salkoff [Salkoff 73, 79], c'est d'ailleurs une partie du travail que Pascal Pellegrini a effectué pour sa thèse.

Conclusion

Le projet de transposition a permis de tester Tomas en taille réelle et de valider ainsi la démarche utilisée. D'autres projets de la société Cora utiliseront Tomas, de manière à obtenir un ensemble de programmes permettant de traiter au mieux l'information textuelle.

3. Liens avec d'autres travaux

D'autres projets menés au LADL pourraient permettre d'améliorer le système Tomas, en particulier les automates de traitement de dictionnaire de Dominique Revuz [Revuz 91] et les automates de reconnaissance de dates de Denis Maurel [Maurel 89].

a. Automates de traitement de dictionnaire

Ces automates permettent de comprimer de manière extrêmement efficace des dictionnaires informatiques. Une fois généré l'automate représentant le dictionnaire, il est possible de le charger entièrement en mémoire ce qui accélère énormément les accès au dictionnaire. L'intégration de ce type d'automate dans Tomas améliorerait nettement les performances dans la mesure où l'étiquetage des mots est la phase la plus lente du programme.

La conception modulaire de Tomas permet d'intégrer ces automates sans avoir à modifier la structure du système. Grâce au langage Ada, les modifications resteront localisées dans le composant d'accès au dictionnaire.

b Automates de reconnaissance de dates

Denis Maurel a étudié le problème de la reconnaissance des adverbes de dates comme *le trois mai*, *hier matin* ou *lundi 3* et a réalisé un automate qui permet d'identifier ce type d'adverbes dans une phrase.

Cet automate pourrait être utilisé dans Tomas dans le module "étiquetage et reconnaissance des expressions figées". De cette manière, il serait plus facile de traiter les expressions adverbiales de date complexes. De plus, la coopération de ces deux programmes donnerait de meilleurs résultats que ceux obtenus avec chaque programme séparément.

En effet, considérons la phrase suivante[31]:

Lundi 3 hommes sont venus[32]

31. Cet exemple est issu d'une discussion avec Alain Guillet.
32. Le bon usage imposerait d'écrire : *Lundi, trois hommes sont venus*, mais ceci n'est pas toujours respecté.

L'automate de reconnaissance des adverbes de dates identifiera deux possibilités :

Lundi 3

Lundi

et ensuite seule une étude plus fine permettra de choisir entre ces deux analyses :

<Debut de Phrase> <Adverbe de temps> <Numéral> <Nom pluriel> <Verbe>

et

<Adverbe de temps> <Nom pluriel> <Verbe>

Tomas est tout à fait capable de lever cette ambiguïté, par exemple à partir d'une règle interdisant une séquence *Adv N*[33] en début de phrase.

D'une manière plus générale, tout mécanisme qui permette d'identifier des parties spécifiques du discours présente un intérêt certain pour Tomas.

33. Comme me l'a fait remarqué M. Salkoff, cette règle est incorrecte ou tout au moins incomplète. Elle n'est fournie qu'à titre indicatif.

Conclusion

L'association de la structure en graphe de la phrase et des règles négatives permet de traiter simplement et efficacement les problèmes d'ambiguïté lexicale, en particulier en prenant en compte les expressions figées.

Ces deux techniques ont été appliquées dans le projet Tomas. Ce projet, développé dans le cadre d'une convention CIFRE entre le LADL (CNRS) et la société CORA, a déjà été utilisé comme désambiguïseur pour un système de traduction de groupes nominaux. La société CORA compte également réemployer Tomas en tant que frontal d'un analyseur en chaîne du français.

Ceci illustre bien l'intérêt de ce projet : en pratique, Tomas est surtout intéressant dans ses possibilités d'association avec d'autres programmes d'analyse. Tomas a en effet été développé dans une optique de réutilisabilité.

Il est donc possible de le réemployer tel quel ou bien par module. Cette possibilité de réemploi par module (par exemple, la gestion du graphe de phrase ou bien l'application des règles négatives) est rendue possible par le choix du langage ADA pour l'écriture de Tomas. Ce langage associé à une optique génie logiciel a permis de rendre Tomas modulaire.

Le projet Tomas continuera à évoluer à la société CORA dans deux directions :

— extension des possibilités de traitement, en particulier pour les séquences agrammaticales, dans le cadre d'un correcteur grammatical à l'usage de la presse écrite;

— prise en compte d'autres langues pour suivre l'évolution du marché européen (allemand, italien).

Ce projet montre, s'il en est encore besoin, la puissance des règles locales pour la levée des ambiguïtés. Il démontre également que les techniques de génie logiciel permettent de développer des outils logiciels réutilisables facilement.

Bibliographie

- BOOCH, Grady. 1988. *Ingénierie du logiciel avec ADA,* InterEditions, Paris.
- BOONS, Jean-Paul ; GUILLET, Alain ; LECLERE Christian. 1976. *La structure des phrases simples en français*, Droz, Genève.
- COURTOIS, Blandine. 1984 et 1989. *DELAS : Dictionnaire Electronique du LADL pour les mots Simples du français,* Rapport technique du LADL, Paris.
- COURTOIS, Blandine. 1990. *Un système de dictionnaires électroniques pour les mots simples du français*, Langue française n° 87, Larousse, Paris.
- DANLOS, Laurence. 1980. *Représentation d'informations linguistiques : constructions «N être Prép X»* , Thèse de troisième cycle, Université Paris 7.
- DANLOS, Laurence. 1981. *La morphosyntaxe des expressions figées*, Langages 63, Larousse, Paris.
- DANLOS, Laurence. 1988. *Les expressions figées*, Langages 90, Larousse, Paris.
- GROSS, Gaston. 1986. *Typologie des noms composés*, Rapport final de l'ATP «Recherches nouvelles sur le langage», Université Paris 13.
- GROSS, Gaston. 1988. *Degré de figement des noms composés*, Langages 90, Larousse, Paris.
- GROSS, Gaston. 1990. *Définition des noms composés dans un lexique-grammaire,* Langue française n° 87, Larousse, Paris.
- GROSS, Maurice. 1968. *Grammaire transformationnelle du français. 1. Syntaxe du verbe,* Cantilène, Paris.
- GROSS, Maurice ; LENTIN, André. 1967. *Notions sur les grammaires formelles,* Gauthier-Villars, Paris.
- GROSS, Maurice. 1975. *Méthodes en syntaxes,* Hermann, Paris.
- GROSS, Maurice. 1982a. *Une classification des phrases figées du français*, Revue québécoise de linguistique, volume 11, n°2, Presses de l'université du Québec.
- GROSS, Maurice. 1982b.*Grammaire transformationnelle du français. 2. Syntaxe du nom,* Cantilène, Paris.
- GROSS, Maurice. 1988. *Sur les phrases figées complexes du Français,* Langue française n°77, Larousse, Paris.
- GROSS, Maurice. 1990. *Grammaire transformationnelle du français. 3. Syntaxe de l'adverbe,* Asstril, Paris.
- KNUTH, Donald. 1973. *The art of computer programming,* Addison Wesley,

- LAPORTE, Eric. 1988. *La reconnaissance des expressions figées lors de l'analyse automatique,* Langages 90, Larousse, Paris.
- MATHIEU-COLAS, Michel. 1988. *Variations graphiques des mots composés dans Le petit Larousse et Le Petit Robert,* Linguisticae Investigationes, XII : 2, John Benjamins B.V., Amsterdam.
- MAUREL, Daniel. 1989. *Reconnaissance de séquences de mots par automate,* Thèse de doctorat, Université Paris 7.
- MILLER, Philip ; TORRIS, Thérèse. 1990. *Formalismes syntaxiques pour le traitement automatique du langage naturel,* Hermès, Paris.
- PELLEGRINI, Pascal ; PELLETIER, Béatrice ; SALKOFF, Morris. 1991. *Transposition, rapport final,* Rapport du MRT pour le contrat n° 903 40 41 00.
- PELLETIER, Béatrice. 1991. *Classification des N de N du français,* Rapport interne de CORA.
- PERRIN, Daniel. 1989. *Automates et algorithmes sur les mots*, Annales des télécommunications, tome 44, CNET.
- REVUZ, Dominique. 1991. *Dictionnaires et lexiques - Méthodes et algorithmes,* Thèse de doctorat, Université Paris 7.
- SABAH, Gérard. 1988. *L'intelligence artificielle et le langage, 1. Représentation des connaissances,* Hermes, Paris.
- SABAH, Gérard. 1989. *L'intelligence artificielle et le langage, 2. Processus de compréhension,* Hermes, Paris.
- SALKOFF, Morris. 1973. *Une grammaire en chaîne du français,* Dunod, Paris.
- SALKOFF, Morris. 1977. *Les principes d'un analyseur syntaxique du Français,* Linguisticæ Investigationes, John Benjamins B.V., Amsterdam.
- SALKOFF, Morris. 1979. *Analyse syntaxique du Français : Grammaire en chaîne*, John Benjamins B.V., Amsterdam.
- SILBERZTEIN, Max Dan. 1989. *Dictionnaires électroniques et reconnaissance lexicale automatique,* Thèse de doctorat, Université Paris 7.
- SILBERZTEIN, Max Dan. 1990. *Le dictionnaire électronique des mots composés,* Langue française n°87, Larousse, Paris.
- VIVES, Robert. 1990. *Les composés nominaux par juxtaposition*, Langue française n° 97, Larousse, Paris.

Annexe 1
Résultats

Dans les pages suivantes, nous présentons les analyses fournies par Tomas pour une vingtaine de phrases extraites des journaux Le Monde Informatique et 01 Informatique. Pour chaque phrase, nous fournissons d'abord l'étiquetage de la phrase avant analyse puis l'étiquetage après analyse.

1. Acer rachète Altos.

```
Acer                |(Acer)NPR
rachète             |(racheter)VER.TR.PI
Altos               |(Altos)NPR
============================
Acer                |(Acer)NPR
rachète             |(racheter)VER.TR.PI
Altos               |(Altos)NPR
```

2. En acquérant Altos pour 94 millions de dollars, le constructeur taiwanais de micro-ordinateurs se dote d'une compétence en systèmes multipostes sous Unix.

```
en                      |(en)CSU            |(en)PRE            |(en)PRO.PPV3.PER.MF
acquérant               |(acquérir)VER.TR.PP
Altos                   |(Altos)NPR
pour                    |(pour)PRE
94                      |(94)NUMERAL
millions                |(million)NOM.M.P
de                      |(de)PRE            |(de)DET.ARI.MF.SP
dollars                 |(dollar)NOM.M.P
,                       |(,)PNC
le                      |(le)DET.ARD.IL.M.S |(le)PRO.PPV2.PER.IL
constructeur            |(constructeur)NOM.M
taiwanais               |(taiwanais)INC
de                      |(de)PRE            |(de)DET.ARI.MF.SP
micro - ordinateurs     |NOM.NC.M.P
se                      |(se)PRO.PPV3.PER.IL
dote                    |(doter)VER.TR.PI.JE
d'                      |(de)PRE            |(de)DET.ARI.MF.SP
une                     |(un)QUA.F.S        |(un)DET.ARI.F.S
compétence              |(compétence)NOM.F.S
en                      |(en)CSU            |(en)PRE            |(en)PRO.PPV3.PER.MF
systèmes multipostes    |NOM.NC.M.P
sous                    |(sous)PRE
Unix                    |(Unix)NPR
=========================
en                      |(en)PRE
acquérant               |(acquérir)VER.TR.PP
Altos                   |(Altos)NPR
pour                    |(pour)PRE
94                      |(94)NUMERAL
millions                |(million)NOM.M.P
de                      |(de)PRE
dollars                 |(dollar)NOM.M.P
,                       |(,)PNC
le                      |(le)DET.ARD.IL.M.S
constructeur            |(constructeur)NOM.M
taiwanais               |(taiwanais)INC
de                      |(de)PRE
```

```
micro - ordinateurs |NOM.NC.M.P
se                  |(se)PRO.PPV3.PER.IL
dote                |(doter)VER.TR.PI.JE
d'                  |(de)PRE
une                 |(un)DET.ARI.F.S
compétence          |(compétence)NOM.F.S
en                  |(en)PRE
systèmes multipostes|NOM.NC.M.P
sous                |(sous)PRE
Unix                |(Unix)NPR
```

3. Les firmes taiwanaises n'en finissent pas de grossir.

```
les                 |(les)DET.ARD.MF.P   |(le)PRO.PPV2.PER.IL
firmes              |(firme)NOM.F.P
taiwanaises         |(taiwanaises)INC
n'                  |(ne)ADV
en                  |(en)CSU             |(en)PRE             |(en)PRO.PPV3.PER.MF
finissent           |(finir)VER.PI.ILS
pas                 |(pas)NOM.M.P        |(pas)ADV            |(pas)NOM.M.S
de                  |(de)PRE             |(de)DET.ARI.MF.SP
grossir             |(grossir)VER.TR.IN
============================
les                 |(les)DET.ARD.MF.P
firmes              |(firme)NOM.F.P
taiwanaises         |(taiwanaises)INC
n'                  |(ne)ADV
en                  |(en)PRO.PPV3.PER.MF
finissent           |(finir)VER.PI.ILS
pas                 |(pas)NOM.M.P        |(pas)ADV            |(pas)NOM.M.S
de                  |(de)PRE
grossir             |(grossir)VER.TR.IN
```

4. Lors d'infopro, ABC Informatique présentera le dernier né de ses logiciels, une gestion commerciale en 5 modules (facturation, stock).

```
lors d'              |PRE.PRC
Infopro              |(Infopro)NPR
,                    |(,)PNC
ABC Informatique     |(ABC Informatique)N
présentera           |(présentera)INC
le                   |(le)DET.ARD.IL.M.S |(le)PRO.PPV2.PER.IL
dernier né de        |NOM.NC.M.S
ses                  |(ses)DET.POS.IL.MF.
logiciels            |(logiciel)ADJ.M.P  |(logiciel)NOM.M.P
,                    |(,)PNC
une                  |(un)QUA.F.S        |(un)DET.ARI.F.S
gestion              |(gestion)NOM.F.S
commerciale          |(commercial)NOM.F.S|(commercial)ADJ.F.S
en                   |(en)CSU            |(en)PRE            |(en)PRO.PPV3.PER.MF
5                    |(5)NUMERAL
modules              |(moduler)VER.PI.TU |(module)NOM.M.P
(                    |(()PNC
facturation          |(facturation)NOM.F.
,                    |(,)PNC
stock                |(stock)NOM.M.S
)                    |())PNC
=============================
lors d'              |PRE.PRC
Infopro              |(Infopro)NPR
,                    |(,)PNC
ABC Informatique     |(ABC Informatique)N
présentera           |(présentera)INC
le                   |(le)DET.ARD.IL.M.S
dernier né de        |NOM.NC.M.S
ses                  |(ses)DET.POS.IL.MF.
logiciels            |(logiciel)NOM.M.P
,                    |(,)PNC
une                  |(un)DET.ARI.F.S
gestion              |(gestion)NOM.F.S
commerciale          |(commercial)ADJ.F.S
en                   |(en)PRE
5                    |(5)NUMERAL
```

```
modules              |(moduler)VER.PI.TU
(                    |(()PNC
facturation          |(facturation)NOM.F.
,                    |(,)PNC
stock                |(stock)NOM.M.S
)                    |())PNC
```

5. Les filiales françaises de 3Com et Rank Xerox ont conclu un accord de commercialisation par lequel Rank Xerox intègre à son offre la famille des logiciels de réseaux d'entreprises 3Com, 3+ et 3Open.

```
les                  |(les)DET.ARD.MF.P   |(le)PRO.PPV2.PER.IL
filiales             |(filiale)NOM.F.P
françaises           |(français)ADJ.F.P   |(français)NOM.F.P
de                   |(de)PRE             |(de)DET.ARI.MF.SP
3                    |(3)NUMERAL
Com                  |(Com)NPR
et                   |(et)CCO
Rank Xerox           |(Rank Xerox)NPR
ont                  |(avoir)VER.TR.PI.IL
conclu               |(conclure)PPS.TR.M.
un                   |(un)QUA.M.S         |(un)DET.ARI.M.S
accord de commercial |NOM.NC.M.S
par                  |(par)PRE
lequel               |(lequel)PRO.REL.M.S
Rank Xerox           |(Rank Xerox)NPR
intègre              |(intègre)ADJ.MF.S   |(intégrer)VER.TR.SU|(intègre)ADJ.M.S    |(intègre)ADJ.F.S
à                    |(à)PRE
son                  |(son)DET.POS.IL.MF. |(son)NOM.M.S
offre                |(offrir)VER.TR.SUP. |(offre)NOM.F.S
la                   |(le)PRO.PPV2.PER.F. |(le)DET.ARD.F.S     |(la)NOM.NRARE.M.S  |(la)NOM.NRARE.M.P
famille              |(famille)NOM.F.S
des                  |(du)PRE             |(du)DET.ARI.MF.P
logiciels            |(logiciel)ADJ.M.P   |(logiciel)NOM.M.P
de                   |(de)PRE             |(de)DET.ARI.MF.SP
réseaux locaux       |NOM.M.P
d'                   |(de)PRE             |(de)DET.ARI.MF.SP
entreprises          |(entreprise)NOM.F.P |(entreprendre)PPS.T
3                    |(3)NUMERAL
Com                  |(Com)NPR
,                    |(,)PNC
3                    |(3)NUMERAL
et                   |(et)CCO
3                    |(3)NUMERAL
Open                 |(Open)NPR
=========================
les                  |(les)DET.ARD.MF.P
```

```
filiales                |(filiale)NOM.F.P
françaises              |(français)ADJ.F.P
de                      |(de)PRE
3                       |(3)NUMERAL
Com                     |(Com)NPR
et                      |(et)CCO
Rank Xerox              |(Rank Xerox)NPR
ont                     |(avoir)VER.TR.PI.IL
conclu                  |(conclure)PPS.TR.M.
un                      |(un)DET.ARI.M.S
accord de commercial    |NOM.NC.M.S
par                     |(par)PRE
lequel                  |(lequel)PRO.REL.M.S
Rank Xerox              |(Rank Xerox)NPR
intègre                 |(intégrer)VER.TR.SU
à                       |(à)PRE
son                     |(son)DET.POS.IL.MF.
offre                   |(offre)NOM.F.S
la                      |(le)DET.ARD.F.S
famille                 |(famille)NOM.F.S
de                      |(de)PRE
                        |(le)DET.ARD.MF.P
logiciels               |(logiciel)NOM.M.P
de                      |(de)PRE
réseaux locaux          |NOM.M.P
d'                      |(de)PRE
entreprises             |(entreprise)NOM.F.P
3                       |(3)NUMERAL
Com                     |(Com)NPR
,                       |(,)PNC
3                       |(3)NUMERAL
et                      |(et)CCO
3                       |(3)NUMERAL
Open                    |(Open)NPR
```

6. Rank Xerox SA enrichit ainsi son offre en matière de réseaux locaux d'entreprises et s'ouvre au monde DLC/LLC, TCP/IP, ISO, NetBIOS et MSNet.

```
Rank Xerox SA       |(Rank Xerox SA)NPR
enrichit            |(enrichir)VER.PI.IL
ainsi               |(ainsi)ADV
son                 |(son)DET.POS.IL.MF.|(son)NOM.M.S
offre               |(offrir)VER.TR.SUP.|(offre)NOM.F.S
en matière de       |PRE.PRC
réseaux locaux      |NOM.M.P
d'                  |(de)PRE            |(de)DET.ARI.MF.SP
entreprises         |(entreprise)NOM.F.P|(entreprendre)PPS.T
et                  |(et)CCO
s'                  |(se)PRO.PPV3.REF.IL|(si)CSU          |(se)PRO.PPV3.REF.IL
ouvre               |(ouvrir)VER.TR.PI.J
au                  |(au)DET.ARD.M.S
monde               |(monder)VER.IM.TU  |(monde)NOM.M.S
DLC                 |(DLC)NPR
/                   |(/)PNC
LLC                 |(LLC)NPR
,                   |(,)PNC
TCP                 |(TCP)NPR
/                   |(/)PNC
IP                  |(IP)NPR
,                   |(,)PNC
ISO                 |(ISO)NPR
,                   |(,)PNC
NetBIOS             |(NetBIOS)NPR
et                  |(et)CCO
MSNet               |(MSNet)NPR
============================
Rank Xerox SA       |(Rank Xerox SA)NPR
enrichit            |(enrichir)VER.PI.IL
ainsi               |(ainsi)ADV
son                 |(son)DET.POS.IL.MF.
offre               |(offre)NOM.F.S
en matière de       |PRE.PRC
réseaux locaux      |NOM.M.P
d'                  |(de)PRE
entreprises         |(entreprise)NOM.F.P
```

```
et                  |(et)CCO
s'                  |(se)PRO.PPV3.REF.IL|(se)PRO.PPV3.REF.IL
ouvre               |(ouvrir)VER.TR.PI.J
au                  |(au)DET.ARD.M.S
monde               |(monde)NOM.M.S
DLC                 |(DLC)NPR
/                   |(/)PNC
LLC                 |(LLC)NPR
,                   |(,)PNC
TCP                 |(TCP)NPR
/                   |(/)PNC
IP                  |(IP)NPR
,                   |(,)PNC
ISO                 |(ISO)NPR
,                   |(,)PNC
NetBIOS             |(NetBIOS)NPR
et                  |(et)CCO
MSNet               |(MSNet)NPR
```

7. Cette offre permet ainsi à ses clients, qui n'ont pas les moyens d'accéder à des solutions XNS sous Ethernet, d'évoluer vers des solutions réseaux performantes.

```
cette                |(ce)DET.DEM.F.S
offre                |(offrir)VER.TR.SUP.|(offre)NOM.F.S
permet               |(permettre)VER.IT.P
ainsi                |(ainsi)ADV
à                    |(à)PRE
ses                  |(ses)DET.POS.IL.MF.
clients              |(client)NOM.M.P
,                    |(,)PNC
qui                  |(qui)PRO.REL.MF.SP.
n'                   |(ne)ADV
ont                  |(avoir)VER.TR.PI.IL
pas                  |(pas)ADV            |(pas)NOM.M.P        |(pas)NOM.M.S
les                  |(les)DET.ARD.MF.P    |(le)PRO.PPV2.PER.IL
moyens               |(moyen)ADJ.M.P       |(moyen)NOM.M.P
d'                   |(de)PRE              |(de)DET.ARI.MF.SP
accéder              |(accéder)VER.IT.IN
à                    |(à)PRE
des                  |(du)PRE              |(du)DET.ARI.MF.P
solutions            |(solution)NOM.F.P
XNS                  |(XNS)NPR
sous                 |(sous)PRE
Ethernet             |(Ethernet)NPR
,                    |(,)PNC
d'                   |(de)PRE              |(de)DET.ARI.MF.SP
évoluer              |(évoluer)VER.IT.IN
vers                 |(ver)PRE
des                  |(du)PRE              |(du)DET.ARI.MF.P
solutions            |(solution)NOM.F.P
réseaux              |(réseau)NOM.M.P
performantes         |(performant)ADJ.F.P
==========================
cette                |(ce)DET.DEM.F.S
offre                |(offre)NOM.F.S
permet               |(permettre)VER.IT.P
ainsi                |(ainsi)ADV
à                    |(à)PRE
ses                  |(ses)DET.POS.IL.MF.
```

```
clients          |(client)NOM.M.P
,                |(,)PNC
qui              |(qui)PRO.REL.MF.SP.
n'               |(ne)ADV
ont              |(avoir)VER.TR.PI.IL
pas              |(pas)ADV          |(pas)NOM.M.P      |(pas)NOM.M.S
les              |(les)DET.ARD.MF.P
moyens           |(moyen)NOM.M.P
d'               |(de)PRE
accéder          |(accéder)VER.IT.IN
à                |(à)PRE
des              |(du)DET.ARI.MF.P
solutions        |(solution)NOM.F.P
XNS              |(XNS)NPR
sous             |(sous)PRE
Ethernet         |(Ethernet)NPR
,                |(,)PNC
d'               |(de)PRE
évoluer          |(évoluer)VER.IT.IN
vers             |(ver)PRE
des              |(du)DET.ARI.MF.P
solutions        |(solution)NOM.F.P
réseaux          |(réseau)NOM.M.P
performantes     |(performant)ADJ.F.P
```

8. Les produits 3Com seront diffusés par la force de vente grands comptes de Xerox et par son réseau de distribution indirecte, les Xerox Store notamment.

```
les                    |(les)DET.ARD.MF.P   |(le)PRO.PPV2.PER.IL
produits               |(produit)NOM.M.P    |(produire)PPS.TR.M.
3                      |(3)NUMERAL
Com                    |(Com)NPR
seront                 |(être)VER.IT.FI.ILS
diffusés               |(diffuser)PPS.TR.M.
par                    |(par)PRE
la                     |(le)PRO.PPV2.PER.F.|(le)DET.ARD.F.S     |(la)NOM.NRARE.M.S   |(la)NOM.NRARE.M.P
force de vente         |NOM.NC.F.S
grands comptes         |NOM.NC.M.P
de                     |(de)PRE             |(de)DET.ARI.MF.SP
Xerox                  |(Xerox)NPR
et                     |(et)CCO
par                    |(par)PRE
son                    |(son)DET.POS.IL.MF.|(son)NOM.M.S
réseau de distributi   |NOM.NC.M.S
indirecte              |(indirect)ADJ.F.S
,                      |(,)PNC
les                    |(les)DET.ARD.MF.P   |(le)PRO.PPV2.PER.IL
Xerox Store            |(Xerox Store)NPR
notamment              |(notamment)ADV
============================
les                    |(les)DET.ARD.MF.P
produits               |(produit)NOM.M.P
3                      |(3)NUMERAL
Com                    |(Com)NPR
seront                 |(être)VER.IT.FI.ILS
diffusés               |(diffuser)PPS.TR.M.
par                    |(par)PRE
la                     |(le)DET.ARD.F.S
force de vente         |NOM.NC.F.S
grands comptes         |NOM.NC.M.P
de                     |(de)PRE
Xerox                  |(Xerox)NPR
et                     |(et)CCO
```

```
par                    |(par)PRE
son                    |(son)DET.POS.IL.MF.
réseau de distributi|NOM.NC.M.S
indirecte              |(indirect)ADJ.F.S
,                      |(,)PNC
les                    |(les)DET.ARD.MF.P
Xerox Store            |(Xerox Store)NPR
notamment              |(notamment)ADV
```

9. Ce sont donc 80 ingénieurs vente-système spécialisés dans les réseaux, 60 ingénieurs technicaux-commerciaux, 24 boutiques Xerox Store, 30 concessionnaires et 10 VAR que Rank Xerox investit sur le marché des réseaux locaux.

```
ce                    |(ce)DET.DEM.M.S     |(ce)PRO.PPV1.DEM.MF
sont                  |(être)VER.IT.PI.ILS
donc                  |(donc)CCO
80                    |(80)NUMERAL
ingénieurs            |(ingénieur)NOM.M.P
vente - système       |NOM.NC.F.S
spécialisés           |(spécialiser)PPS.TR
dans                  |(dans)PRE
les                   |(les)DET.ARD.MF.P   |(le)PRO.PPV2.PER.IL
réseaux               |(réseau)NOM.M.P
,                     |(,)PNC
60                    |(60)NUMERAL
ingénieurs technico   |NOM.NC.M.P
,                     |(,)PNC
24                    |(24)NUMERAL
boutiques             |(boutiquer)VER.II.T|(boutique)NOM.F.P
Xerox Store           |(Xerox Store)NPR
,                     |(,)PNC
30                    |(30)NUMERAL
concessionnaires      |(concessionnaire)NO|(concessionnaire)NO
et                    |(et)CCO
10                    |(10)NUMERAL
VAR                   |(VAR)NPR
que                   |(que)PRO.REL.MF.SP.|(que)INT            |(que)CSU
Rank Xerox            |(Rank Xerox)NPR
investit              |(investir)VER.TR.PI
sur                   |(sur)PRE
le                    |(le)DET.ARD.IL.M.S  |(le)PRO.PPV2.PER.IL
marché                |(marché)NOM.M.S
des                   |(du)PRE             |(du)DET.ARI.MF.P
réseaux locaux        |NOM.M.P
============================
ce                    |(ce)PRO.PPV1.DEM.MF
sont                  |(être)VER.IT.PI.ILS
donc                  |(donc)CCO
80                    |(80)NUMERAL
```

ingénieurs	|(ingénieur)NOM.M.P			
vente - système	|NOM.NC.F.S			
spécialisés	|(spécialiser)PPS.TR			
dans	|(dans)PRE			
les	|(les)DET.ARD.MF.P			
réseaux	|(réseau)NOM.M.P			
,	|(,)PNC			
60	|(60)NUMERAL			
ingénieurs technico	|NOM.NC.M.P			
,	|(,)PNC			
24	|(24)NUMERAL	|(24)NUMERAL		
boutiques	|(boutiquer)VER.II.T	|(boutique)NOM.F.P		
Xerox Store	|(Xerox Store)NPR	|(Xerox Store)NPR		
,	|(,)PNC	|(,)PNC		
30	|(30)NUMERAL	|(30)NUMERAL		
concessionnaires	|(concessionnaire)NO	|(concessionnaire)NO	|(concessionnaire)NO	|(concessionnaire)NO
et	|(et)CCO	|(et)CCO		
10	|(10)NUMERAL	|(10)NUMERAL		
VAR	|(VAR)NPR	|(VAR)NPR		
que	|(que)PRO.REL.MF.SP.	|(que)CSU		
Rank Xerox	|(Rank Xerox)NPR			
investit	|(investir)VER.TR.PI			
sur	|(sur)PRE			
le	|(le)DET.ARD.IL.M.S			
marché	|(marché)NOM.M.S			
de	|(de)PRE			
	|(le)DET.ARD.MF.P			
réseaux locaux	|NOM.M.P			

10. L'objectif est de vendre au moins 700 réseaux en 89 avec 4 ou 5 points de connexion par réseau et de prendre ainsi 5 % de l'offre réseau.

```
l'                    |(le)DET.ARD.M.S    |(le)PRO.PER.M.S.COD|(la)DET.ARD.F.S    |(la)PRO.PER.F.S.COD
objectif              |(objectif)ADJ.M.S  |(objectif)NOM.M.S
est                   |(est)NOM.MF.S      |(être)VER.IT.PI.IL |(est)ADJ.M.S       |(est)NOM.M.S       |(est)ADJ.F.S
de                    |(de)PRE            |(de)DET.ARI.MF.SP
vendre                |(vendre)VER.TR.IN
au moins              |ADV
700                   |(700)NUMERAL
réseaux               |(réseau)NOM.M.P
en                    |(en)CSU            |(en)PRE            |(en)PRO.PPV3.PER.MF
89                    |(89)NUMERAL
avec                  |(avec)PRE
4                     |(4)NUMERAL
ou                    |(ou)CCO
5                     |(5)NUMERAL
points de connexion   |NOM.NC.M.P
par                   |(par)PRE
réseau                |(réseau)NOM.M.S
et                    |(et)CCO
de                    |(de)PRE            |(de)DET.ARI.MF.SP
prendre               |(prendre)VER.TR.IN
ainsi                 |(ainsi)ADV
5                     |(5)NUMERAL
de                    |(de)PRE            |(de)DET.ARI.MF.SP
l'                    |(le)DET.ARD.M.S    |(le)PRO.PER.M.S.COD|(la)DET.ARD.F.S    |(la)PRO.PER.F.S.COD
offre réseau          |NOM.NC.F.S
==========================
l'                    |(le)DET.ARD.M.S
objectif              |(objectif)NOM.M.S
est                   |(être)VER.IT.PI.IL
de                    |(de)PRE
vendre                |(vendre)VER.TR.IN
au moins              |ADV
700                   |(700)NUMERAL
réseaux               |(réseau)NOM.M.P
en                    |(en)PRE
89                    |(89)NUMERAL
avec                  |(avec)PRE
```

```
4                    |(4)NUMERAL
ou                   |(ou)CCO
5                    |(5)NUMERAL
points de connexion  |NOM.NC.M.P
par                  |(par)PRE
réseau               |(réseau)NOM.M.S
et                   |(et)CCO
de                   |(de)PRE
prendre              |(prendre)VER.TR.IN
ainsi                |(ainsi)ADV
5                    |(5)NUMERAL
de                   |(de)PRE
l'                   |(la)DET.ARD.F.S
offre réseau         |NOM.NC.F.S
```

11. Act et Unilog préparent leur retraite.

```
Act                  |(Act)NPR
et                   |(et)CCO
Unilog               |(Unilog)NPR
préparent            |(préparer)VER.TR.PI
leur                 |(leur)DET.POS.IL.MF|(leur)PRO.PPV2.PER.
retraite             |(retraite)VER.II.J|(retraite)NOM.F.S
==========================
Act                  |(Act)NPR
et                   |(et)CCO
Unilog               |(Unilog)NPR
préparent            |(préparer)VER.TR.PI
leur                 |(leur)DET.POS.IL.MF
retraite             |(retraite)NOM.F.S
```

12. Depuis plus d'un an déjà, elles travaillent sur des réalisations communes de systèmes experts dans ce créneau du secteur tertiaire.

```
depuis              |(depuis)PRE
plus d'             |PRE.PRC
un                  |(un)QUA.M.S          |(un)DET.ARI.M.S
an                  |(an)NOM.M.S
déjà                |(déjà)ADV
,                   |(,)PNC
elles               |(il)PRO.PPV1.PER.IL
travaillent         |(travailler)VER.IT.
sur                 |(sur)PRE
des                 |(du)PRE              |(du)DET.ARI.MF.P
réalisations        |(réalisation)NOM.F.
communes            |(commune)NOM.F.P     |(commun)ADJ.F.P
de                  |(de)PRE              |(de)DET.ARI.MF.SP
systèmes experts    |NOM.NC.M.P
dans                |(dans)PRE
ce                  |(ce)DET.DEM.M.S      |(ce)PRO.PPV1.DEM.MF
créneau             |(créneau)NOM.M.S
du                  |(du)PRE              |(du)DET.ARI.M.S
secteur tertiaire   |NOM.NC.M.S
============================
depuis              |(depuis)PRE
plus d'             |PRE.PRC
un                  |(un)DET.ARI.M.S
an                  |(an)NOM.M.S
déjà                |(déjà)ADV
,                   |(,)PNC
elles               |(il)PRO.PPV1.PER.IL
travaillent         |(travailler)VER.IT.
sur                 |(sur)PRE
des                 |(du)DET.ARI.MF.P
réalisations        |(réalisation)NOM.F.P
communes            |(commun)ADJ.F.P
```

```
de                  |(de)PRE
systèmes experts    |NOM.NC.M.P
dans                |(dans)PRE
ce                  |(ce)DET.DEM.M.S
créneau             |(créneau)NOM.M.S
de                  |(de)PRE            |(le)DET.ARD.M.S
secteur tertiaire   |NOM.NC.M.S
```

13. L'accord se traduit par une offre unique, des outils et des moyens communs.

```
l'                  |(le)DET.ARD.M.S     |(le)PRO.PER.M.S.COD|(la)DET.ARD.F.S     |(la)PRO.PER.F.S.COD
accord              |(accord)NOM.M.S
se                  |(se)PRO.PPV3.PER.IL
traduit             |(traduire)VER.TR.PI|(traduire)PPS.TR.M.
par                 |(par)PRE
une                 |(un)QUA.F.S         |(un)DET.ARI.F.S
offre               |(offrir)VER.TR.SUP.|(offre)NOM.F.S
unique              |(unique)ADJ.F.S     |(unique)ADJ.M.S
,                   |(,)PNC
des                 |(du)PRE             |(du)DET.ARI.MF.P
outils              |(outil)NOM.M.P
et                  |(et)CCO
des                 |(du)PRE             |(du)DET.ARI.MF.P
moyens              |(moyen)ADJ.M.P      |(moyen)NOM.M.P
communs             |(commun)ADJ.M.P
============================
l'                  |(le)DET.ARD.M.S
accord              |(accord)NOM.M.S
se                  |(se)PRO.PPV3.PER.IL
traduit             |(traduire)VER.TR.PI
par                 |(par)PRE
une                 |(un)DET.ARI.F.S
offre               |(offre)NOM.F.S
unique              |(unique)ADJ.F.S
,                   |(,)PNC
des                 |(du)DET.ARI.MF.P    |(de)PRE
                                         |(le)DET.ARD.MF.P
outils              |(outil)NOM.M.P
et                  |(et)CCO
de                  |(de)PRE
                    |(le)DET.ARD.MF.P    |(du)DET.ARI.MF.P
moyens              |(moyen)NOM.M.P
communs             |(commun)ADJ.M.P
```

14. Les deux sociétés comptent ainsi apporter aux compagnies d'assurances des solutions bien situées dans le contexte de leur informatique.

```
les                    |(les)DET.ARD.MF.P   |(le)PRO.PPV2.PER.IL
deux                   |(deux)PRO.NUM.MF.P.|(deux)NOM.NRARE.M.P|(deux)NOM.NRARE.M.S|(deux)QUA.MF.P
sociétés               |(société)NOM.F.P
comptent               |(compter)VER.IT.PI.
ainsi                  |(ainsi)ADV
apporter               |(apporter)VER.TR.IN
aux                    |(au)DET.ARD.MF.P
compagnies d' assura|NOM.NC.F.P
des                    |(du)PRE             |(du)DET.ARI.MF.P
solutions              |(solution)NOM.F.P
bien                   |(bien)ADV           |(bien)NOM.M.S
situées                |(situé)ADJ.F.P      |(situer)PPS.TR.F.P
dans                   |(dans)PRE
le                     |(le)DET.ARD.IL.M.S  |(le)PRO.PPV2.PER.IL
contexte               |(contexte)NOM.M.S
de                     |(de)PRE             |(de)DET.ARI.MF.SP
leur                   |(leur)DET.POS.IL.MF|(leur)PRO.PPV2.PER.
informatique           |(informatique)NOM.F|(informatique)ADJ.F|(informatique)ADJ.M
============================
les                    |(les)DET.ARD.MF.P
deux                   |(deux)QUA.MF.P
sociétés               |(société)NOM.F.P
comptent               |(compter)VER.IT.PI.
ainsi                  |(ainsi)ADV
apporter               |(apporter)VER.TR.IN
aux                    |(au)DET.ARD.MF.P
compagnies d' assura|NOM.NC.F.P
de                     |(de)PRE
                       |(le)DET.ARD.MF.P
solutions              |(solution)NOM.F.P
bien                   |(bien)ADV           |(bien)NOM.M.S
situées                |(situé)ADJ.F.P      |(situer)PPS.TR.F.P
```

```
dans                |(dans)PRE
le                  |(le)DET.ARD.IL.M.S
contexte            |(contexte)NOM.M.S
de                  |(de)PRE
leur                |(leur)DET.POS.IL.MF
informatique        |(informatique)NOM.F
```

15. CMM Diffusion, c'est le bouquet.

```
CMM Diffusion       |(CMM Diffusion)NPR
,                   |(,)PNC
c'                  |(ce)PRO.DEM.M.S.SUJ
est                 |(être)VER.IT.PI.IL |(est)NOM.MF.S      |(est)ADJ.M.S       |(est)NOM.M.S       |(est)ADJ.F.S
le                  |(le)DET.ARD.IL.M.S |(le)PRO.PPV2.PER.IL
bouquet             |(bouquet)NOM.M.S
============================
CMM Diffusion       |(CMM Diffusion)NPR
,                   |(,)PNC
c'                  |(ce)PRO.DEM.M.S.SUJ
est                 |(être)VER.IT.PI.IL
le                  |(le)DET.ARD.IL.M.S
bouquet             |(bouquet)NOM.M.S
```

16. Le distributeur - et éditeur - lyonnais décerne avec Bull les trophés d'or du meilleur logiciel à CTI, Scalinfor et CIEE.

```
le                  |(le)DET.ARD.IL.M.S |(le)PRO.PPV2.PER.IL
distributeur        |(distributeur)NOM.M
-                   |(-)PNC
et                  |(et)CCO
éditeur             |(éditeur)NOM.M.S
-                   |(-)PNC
lyonnais            |(lyonnais)NOM.M.S  |(lyonnais)ADJ.M.S  |(lyonnais)NOM.M.P  |(lyonnais)ADJ.M.P
décerne             |(décerner)VER.TR.PI
avec                |(avec)PRE
Bull                |(Bull)NPR
les                 |(les)DET.ARD.MF.P  |(le)PRO.PPV2.PER.IL
trophés             |(trophés)INC
d'                  |(de)PRE            |(de)DET.ARI.MF.SP
or                  |(or)CCO            |(or)NOM.M.S
du                  |(du)PRE            |(du)DET.ARI.M.S
meilleur            |(meilleur)ADV      |(meilleur)ADJ.M.S
logiciel            |(logiciel)ADJ.M.S  |(logiciel)NOM.M.S
à                   |(à)PRE
CTI                 |(CTI)NPR
,                   |(,)PNC
Scalinfor           |(Scalinfor)NPR
et                  |(et)CCO
CIEE                |(CIEE)NPR
============================
le                  |(le)DET.ARD.IL.M.S
distributeur        |(distributeur)NOM.M
-                   |(-)PNC
et                  |(et)CCO
éditeur             |(éditeur)NOM.M.S
-                   |(-)PNC
lyonnais            |(lyonnais)NOM.M.S  |(lyonnais)ADJ.M.S  |(lyonnais)NOM.M.P  |(lyonnais)ADJ.M.P
décerne             |(décerner)VER.TR.PI
avec                |(avec)PRE
Bull                |(Bull)NPR
les                 |(les)DET.ARD.MF.P
trophés             |(trophés)INC
d'                  |(de)PRE
```

```
or                  |(or)NOM.M.S
de                  |(de)PRE
                    |(le)DET.ARD.M.S
meilleur            |(meilleur)ADJ.M.S
logiciel            |(logiciel)NOM.M.S
à                   |(à)PRE
CTI                 |(CTI)NPR
,                   |(,)PNC
Scalinfor           |(Scalinfor)NPR
et                  |(et)CCO
CIEE                |(CIEE)NPR
```

17. Pour la quatrième fois consécutive CMM Diffusion, le premier distributeur Bull en région Rhône-Alpes, a décerné en décembre dernier les Fleurs d'Or du meilleur logiciel en présence d'Etienne Grand Jacques, directeur général de Bull Grande Diffusion.

```
pour                |(pour)PRE
la                  |(le)PRO.PPV2.PER.F.|(le)DET.ARD.F.S     |(la)NOM.NRARE.M.S  |(la)NOM.NRARE.M.P
quatrième           |(quatrième)NOM.F.S |(quatrième)NOM.M.S |(quatrième)ADJ.F.S |(quatrième)ADJ.M.S
fois                |(fois)NOM.F.P      |(fois)NOM.F.S      |(foi)NOM.F.P
consécutive         |(consécutif)ADJ.F.S
CMM Diffusion       |(CMM Diffusion)NPR
,                   |(,)PNC
le                  |(le)DET.ARD.IL.M.S |(le)PRO.PPV2.PER.IL
premier             |(premier)NOM.M.S   |(premier)ADJ.M.S
distributeur        |(distributeur)NOM.M
Bull                |(Bull)NPR
en                  |(en)CSU            |(en)PRE            |(en)PRO.PPV3.PER.MF
région              |(région)NOM.F.S
Rhône-Alpes         |(Rhône-Alpes)NPR
,                   |(,)PNC
a                   |(avoir)VER.TR.PI.IL|(a)NOM.M.SP
décerné             |(décerner)PPS.TR.M.
en                  |(en)CSU            |(en)PRE            |(en)PRO.PPV3.PER.MF
décembre            |(décembre)NOM.MOIS.
dernier             |(dernier)NOM.M.S   |(dernier)ADJ.M.S
les                 |(les)DET.ARD.MF.P  |(le)PRO.PPV2.PER.IL
Fleurs              |(Fleurs)NPR
d'                  |(de)PRE            |(de)DET.ARI.MF.SP
Or                  |(Or)NPR
du                  |(du)PRE            |(du)DET.ARI.M.S
meilleur            |(meilleur)ADV      |(meilleur)ADJ.M.S
logiciel            |(logiciel)ADJ.M.S  |(logiciel)NOM.M.S
en présence d'      |PRE.PRC
Etienne             |(Etienne)NPR.PRENOM|(Etienne)NPR.PRENOM
Grand Jacques       |(Grand Jacques)NPR
,                   |(,)PNC
directeur général   |NOM.NC.M.S
de                  |(de)PRE            |(de)DET.ARI.MF.SP
Bull Grande Diffusio|(Bull Grande Diffusion
============================
pour                |(pour)PRE
```

```
la                    |(le)DET.ARD.F.S
quatrième             |(quatrième)ADJ.F.S
fois                  |(fois)NOM.F.S
consécutive           |(consécutif)ADJ.F.S
CMM Diffusion         |(CMM Diffusion)NPR
,                     |(,)PNC
le                    |(le)DET.ARD.IL.M.S
premier               |(premier)ADJ.M.S
distributeur          |(distributeur)NOM.M
Bull                  |(Bull)NPR
en                    |(en)PRE
région                |(région)NOM.F.S
Rhône-Alpes           |(Rhône-Alpes)NPR
,                     |(,)PNC
a                     |(avoir)VER.TR.PI.IL
décerné               |(décerner)PPS.TR.M.
en                    |(en)PRE
décembre              |(décembre)NOM.MOIS.
dernier               |(dernier)ADJ.M.S
les                   |(les)DET.ARD.MF.P
Fleurs                |(Fleurs)NPR
d'                    |(de)PRE
Or                    |(Or)NPR
de                    |(de)PRE              |(du)DET.ARI.M.S
les                   |(le)DET.ARD.M.S
meilleur              |(meilleur)ADJ.M.S
logiciel              |(logiciel)NOM.M.S
en présence d'        |PRE.PRC
Etienne               |(Etienne)NPR.PRENOM|(Etienne)NPR.PRENOM
Grand Jacques         |(Grand Jacques)NPR
,                     |(,)PNC
directeur général     |NOM.NC.M.S
de                    |(de)PRE
Bull Grande Diffusio|(Bull Grande Diffusion)NPR
```

Annexe 2
Extrait du lexique

Voici un extrait du lexique de Tomas, autour de la lettre "Q". On constate au premier coup d'œuil la filiation avec le DELAF...

%pussions :(pouvoir)VER.IT.SUI.NOUS
%pût :(pouvoir)VER.IT.SUI.IL
%pûtes :(pouvoir)VER.IT.PS.VOUS
%qu' :(que)CSU:INT:PRO.REL.MF.SP.COD
%qualifia :(qualifier)VER.TR.PS.IL
%qualifiai :(qualifier)VER.TR.PS.JE
%qualifiaient :(qualifier)VER.TR.II.ILS
%qualifiais :(qualifier)VER.TR.II.TU:VER.TR.II.JE
%qualifiait :(qualifier)VER.TR.II.IL
%qualifiâmes :(qualifier)VER.TR.PS.NOUS
%qualifiant :(qualifier)VER.TR.PPR --(qualifiant)ADJ.M.S/
%qualifias :(qualifier)VER.TR.PS.TU
%qualifiasse :(qualifier)VER.TR.SUI.JE
%qualifiassent :(qualifier)VER.TR.SUI.ILS
%qualifiasses :(qualifier)VER.TR.SUI.TU
%qualifiassiez :(qualifier)VER.TR.SUI.VOUS
%qualifiassions :(qualifier)VER.TR.SUI.NOUS
%qualifiât :(qualifier)VER.TR.SUI.IL
%qualifiâtes :(qualifier)VER.TR.PS.VOUS
%qualifié :(qualifier)PPS.TR.M.S --/(qualifié)ADJ.M.S
%qualifie :(qualifier)VER.TR.IM.TU:VER.TR.SUP.JE:VER.TR.SUP.IL:VE
R.TR.PI.IL:VER.TR.PI.JE
%qualifiée :(qualifier)PPS.TR.F.S --/(qualifié)ADJ.F.S
%qualifiées :(qualifier)PPS.TR.F.P--/(qualifié)ADJ.F.P
%qualifient :(qualifier)VER.TR.SUP.ILS:VER.TR.PI.ILS
%qualifier :(qualifier)VER.TR.IN
%qualifiera :(qualifier)VER.TR.FI.IL
%qualifierai :(qualifier)VER.TR.FI.JE
%qualifieraient :(qualifier)VER.TR.PC.ILS
%qualifierais :(qualifier)VER.TR.PC.TU:VER.TR.PC.JE
%qualifierait :(qualifier)VER.TR.PC.IL
%qualifieras :(qualifier)VER.TR.FI.TU
%qualifièrent :(qualifier)VER.TR.PS.ILS
%qualifierez :(qualifier)VER.TR.FI.VOUS
%qualifieriez :(qualifier)VER.TR.PC.VOUS
%qualifierions :(qualifier)VER.TR.PC.NOUS
%qualifierons :(qualifier)VER.TR.FI.NOUS
%qualifieront :(qualifier)VER.TR.FI.ILS
%qualifiés :(qualifier)PPS.TR.M.P --/(qualifié)ADJ.M.P
%qualifies :(qualifier)VER.TR.SUP.TU:VER.TR.PI.TU
%qualifiez :(qualifier)VER.TR.IM.VOUS:VER.TR.PI.VOUS
%qualifiiez :(qualifier)VER.TR.SUP.VOUS:VER.TR.II.VOUS
%qualifiions :(qualifier)VER.TR.SUP.NOUS:VER.TR.II.NOUS
%qualifions :(qualifier)VER.TR.IM.NOUS:VER.TR.PI.NOUS

%quand :(quand)ADV
%quantitativement :(quantitativement)ADV
%quantité :(quantité)NOM.F.S
%quantités :(quantité)NOM.F.P
%quarante :(quarante)QUA.MF.P:NOM.NRARE.M.S:NOM.NRARE.M.P:PRO.NUM.MF.P.SUJ--plusieurs NOM enlevés
%quasiment :(quasiment)ADV
%quatorze :(quatorze)QUA.MF.P:NOM.NRARE.M.S:NOM.NRARE.M.P:PRO.NUM.MF.P.SUJ--plusieurs NOM enlevés
%quatre :(quatre)QUA.MF.P:NOM.NRARE.M.S:NOM.NRARE.M.P:PRO.NUM.MF.P.SUJ--plusieurs NOM enlevés
%quatre vingt :QUA.MF.P:NOM.NRARE.M.S:NOM.NRARE.M.P:PRO.NUM.MF.P.SUJ--plusieurs NOM enlevés
%quatre vingts :QUA.MF.P:NOM.NRARE.M.P:PRO.NUM.MF.P.SUJ--plusieurs NOM enlevés
%quatre vingt
dix :QUA.MF.P:NOM.NRARE.M.S:NOM.NRARE.M.P:PRO.NUM.MF.P.SUJ--plusieurs NOM enlevés
%quatrième :(quatrième)ADJ.ADJG.M.S:ADJ.ADJG.F.S:NOM.M.S:NOM.F.S-- rétréci
%quatrièmes :(quatrième)ADJ.ADJG.M.P:ADJ.ADJG.F.P:NOM.M.P:NOM.F.P-- retiré doublon
%que :(que)CSU:INT:PRO.REL.MF.SP.COD
%Quentin :(Quentin)NPR.PRENOM.M.S:P.PRENOM
%question :(question)NOM.F.S
%questions :(question)NOM.F.P
%qui :(qui)PRO.REL.MF.SP.COD
%quinze :(quinze)NOM.NRARE.M.S:NOM.NRARE.M.P:QUA.MF.P:PRO.NUM.MF.P.SUJ-- retiré ADJ et PPS
%quitta :(quitter)VER.TR.PS.IL
%quittai :(quitter)VER.TR.PS.JE
%quittaient :(quitter)VER.TR.II.ILS
%quittais :(quitter)VER.TR.II.TU:VER.TR.II.JE
%quittait :(quitter)VER.TR.II.IL
%quittâmes :(quitter)VER.TR.PS.NOUS
%quittant :(quitter)VER.TR.PPR
%quittas :(quitter)VER.TR.PS.TU
%quittasse :(quitter)VER.TR.SUI.JE
%quittassent :(quitter)VER.TR.SUI.ILS
%quittasses :(quitter)VER.TR.SUI.TU
%quittassiez :(quitter)VER.TR.SUI.VOUS
%quittassions :(quitter)VER.TR.SUI.NOUS
%quittât :(quitter)VER.TR.SUI.IL
%quittâtes :(quitter)VER.TR.PS.VOUS
%quitte:(quitte)ADJ.MF.S/(quitter)VER.TR.PI.JE:VER.TR.SUP.JE:VER.TR.SUP.IL:VER.TR.PI.IL:VER.TR.IM.TU
%quitté :(quitter)PPS.TR.M.S
%quittée :(quitter)PPS.TR.F.S
%quittées :(quitter)PPS.TR.F.P
%quittent :(quitter)VER.TR.SUP.ILS:VER.TR.PI.ILS
%quitter :(quitter)VER.TR.IN
%quittera :(quitter)VER.TR.FI.IL
%quitterai :(quitter)VER.TR.FI.JE
%quitteraient :(quitter)VER.TR.PC.ILS

%quitterais :(quitter)VER.TR.PC.TU:VER.TR.PC.JE
%quitterait :(quitter)VER.TR.PC.IL
%quitteras :(quitter)VER.TR.FI.TU
%quittèrent :(quitter)VER.TR.PS.ILS
%quitterez :(quitter)VER.TR.FI.VOUS
%quitteriez :(quitter)VER.TR.PC.VOUS
%quitterions :(quitter)VER.TR.PC.NOUS
%quitterons :(quitter)VER.TR.FI.NOUS
%quitteront :(quitter)VER.TR.FI.ILS
%quittes :(quitte)ADJ.MF.P/(quitter)VER.TR.SUP.TU:VER.TR.PI.TU
%quittés :(quitter)PPS.TR.M.P
%quittez :(quitter)VER.TR.IM.VOUS:VER.TR.PI.VOUS
%quittiez :(quitter)VER.TR.SUP.VOUS:VER.TR.II.VOUS
%quittions :(quitter)VER.TR.SUP.NOUS:VER.TR.II.NOUS
%quittons :(quitter)VER.TR.IM.NOUS:VER.TR.PI.NOUS
%quotidien :(quotidien)ADJ.M.S:NOM.M.S-- ajouté l'entrée
%quotidienne :(quotidien)ADJ.F.S-- ajouté l'entrée
%quotidiennes :(quotidien)ADJ.F.P-- ajouté l'entrée
%quotidiens :(quotidien)ADJ.M.P:NOM.M.P
%raccorda :(raccorder)VER.PS.IL
%raccordai :(raccorder)VER.PS.JE
%raccordaient :(raccorder)VER.II.ILS
%raccordais :(raccorder)VER.II.TU:VER.II.JE
%raccordait :(raccorder)VER.II.IL
%raccordâmes :(raccorder)VER.PS.NOUS

Table des illustrations