DOCTORAT DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

spécialité : Informatique

Thèse

PRÉSENTÉE PAR
Serge ROSMORDUC

Pour obtenir le grade de
Docteur de l'École Normale Supérieure de Cachan

Analyse morpho-syntaxique de textes non ponctués

Application aux textes hiéroglyphiques

Soutenue le 18 décembre 1996 devant le jury composé de :

M. Michel EYTAN, Professeur, Université de Strasbourg	Rapporteur
M. Alain FINKEL, Professeur, ENS de Cachan	Président
M. Benoît HABERT, Maître de conférences, ENS Fontenay	Examinateur
M. Éric LAPORTE, Professeur, Université de Marne la Vallée	Rapporteur
M. Yvan LAVALLÉE, Professeur, Université Paris VIII	Examinateur
Mme. Violaine PRINCE, Professeur, Université Paris VIII	Directrice de Recherche
M. Pascal VERNUS, Directeur de Recherche, ÉPHÉ	Rapporteur

61, av. du Président Wilson 94235 Cachan cedex (FRANCE)

À mes grands parents, affectueusement.

J'ai appris par correspondance... une très bonne école... qui proclame avec raison, que si vous savez dessiner, vous savez écrire !

— René Goscinny et Albert Uderzo, *Astérix et Cléopâtre*, p. 12

Remerciements

Je tiens à remercier ici ceux sans qui cette thèse n'aurais pas vu le jour : Violaine Prince, ma directrice de thèse, et mes parents, dont le soutien constant m'a permis d'arriver au bout de ce travail.

Je remercie Alain Finkel d'avoir bien voulu m'accueillir au LIFAC.

Le séminaire de Pascal Vernus à l'École Pratique a été une source constante d'inspiration et d'ouverture à la linguistique.

Enfin, que soient remerciés ici mes condisciples du LIFAC ; l'ambiance studieuse qui y règne m'a été d'un grand secours.

Et enfin, ma sœur, qui m'a nourri de gâteaux aux chocolats astrophysiques.

Table des matières

Introduction

Le développement considérable des capacités de stockage et d'échange des ordinateurs durant les dernières années, l'amélioration générale de la convivialité des systèmes d'exploitation, l'accroissement de la vitesse des processeurs, ont permis à l'informatique de s'installer sans conteste comme outil dans les sciences humaines. Le grand problème, ici comme ailleurs, est logiciel ; la disponibilité informatique des textes, à elle seule, s'est déjà révélée fructueuse ; elle permet de retrouver facilement les occurrences d'un mot donné dans un texte, et se révèle un auxiliaire précieux pour le lexicographe. En revanche, les possibilités de ce type de méthode sont en deçà de ce dont on aimerait disposer pour l'étude de la syntaxe ; l'informatique linguistique a donc un rôle à jouer dans ce domaine, afin de permettre de manipuler, non plus des mots, mais des constructions complexes.

Il se trouve que ces mêmes évolutions techniques ont remis au goût du jour l'usage de larges corpus en informatique linguistique. C'est dans cette double perspective que nous avons inscrit le présent travail.

0.1 L'informatique dans les sciences humaines

0.1.1 Traitements à partir de corpus

L'informatique sur corpus[1] s'est beaucoup développée ces dernières années, pour des raisons matérielles (la disponibilité accrue de corpus) et théoriques. À côté d'une linguistique statistique bien établie, sont apparues de nombreux modèles, numériques et symboliques, qui ont peu à peu touché tous les domaines du traitement automatique de la langue (TAL).

Deux grands types d'applications, très différentes, ont émergé : l'indexation et la lemmatisation automatique. Cependant, les programmes d'analyse

1. Corpus : ensemble de données linguistiques recueillies, par opposition aux méthodes de l'intelligence artificielle classique, qui s'intéresse aux productions linguistiques « possibles » et utilise des exemples inventés pour les besoins de la cause.

syntaxique qui prétendent à une grande couverture se sont eux aussi multipliés, répondant à des paradigmes très divers.

L'informatique linguistique d'aujourd'hui est sensible à de nouveaux problèmes, autrefois dédaignés, tels que la segmentation des textes ou l'identification des symboles, problèmes qui étaient absents des jeux d'essais produits spécialement pour le TAL, mais que l'on rencontre abondamment dans les textes réels. Un texte explicitement saisi en vue d'une exploitation pour l'informatique linguistique sera sans doute mieux relu, et en particulier mieux ponctué, que ne le sera par exemple l'archive d'une liste de diffusion électronique.

Le problème de la corruption du texte fait depuis longtemps partie de ceux que le philologue doit résoudre ; nous constatons ici une convergence entre deux points de vues d'origines diverses.

Le développement des corpus a d'autre part eu pour conséquence la multiplication des travaux informatiques visant à en assurer l'usage. Le présent travail ne s'occupant pas de sémantique, nous ne mentionnerons que pour mémoire les nombreuses études portant sur l'indexation. L'une des principales méthodes de mise en forme des corpus est l'enrichissement du texte par des annotations. L'un des projets les plus importants dans ce domaine est celui de la Text Encoding Initiative (collectif 1994b), sur lequel nous reviendrons. Il s'agit d'un standard permettant d'annoter un texte, afin d'en faciliter la représentation et l'utilisation. L'un des intérêts de ce codage est qu'il essaie d'être *ouvert* tout en proposant des représentations *partageables* des textes. Il est d'une part possible de l'étendre pour l'adapter aux besoins spécifiques de types très différents d'utilisateurs de textes ; mais dans le même temps, la grande variété des situations prévues, ainsi que l'existence de formalismes destinés à servir de briques de base pour les extensions, permettent une certaine portabilité des textes.

Ce genre de système ne peut se concevoir dans l'absolu ; la TEI est de fait un travail collégial, dans lequel les utilisateurs putatifs du formalisme sont actifs.

Ce genre de codage est souvent assez lourd ; des logiciels spécialisés ont donc été développés. Leur domaine d'usage est principalement la création de documents techniques, mais ils peuvent s'adapter à d'autres types d'usage, et en particulier à l'édition scientifique de textes.

0.1.2 Choix du matériau : une langue morte, une écriture utilisant des idéogrammes

Nous avons décidé d'utiliser comme matériau pour notre étude la langue égyptienne. En grande partie par goût personnel, mais aussi parce que certaines de ses caractéristiques nous semblaient à même de mettre en lumière des phénomènes généraux de traitement des langues.

Langue morte, l'égyptien ancien ne peut s'étudier qu'en considérant des corpus ; langue reconstituée, sa grammaire se justifie principalement par eux. De par sa longue durée de vie, elle pose le problème de la diachronie, problème largement ignoré par le TAL jusqu'à récemment, mais que l'apparition de grandes bases de textes posera forcément.

Le système d'écriture, enfin, possède un double intérêt : d'une part, il n'utilise pas (ou peu) de ponctuation et, d'autre part, ses qualités iconiques et sémantiques peuvent indiquer des pistes profitables, d'une part pour le traitement de la langue, et, d'autre part, comme système de représentation des connaissances.

0.1.3 Choix de la démarche : l'analyse morphosyntaxique du matériau

La syntaxe est l'un des domaines classiques du traitement du langage naturel ; l'apparition, dans les années 50, de modèles mathématisés, souleva l'espoir d'un traitement élégant et définitif du problème. Quarante ans plus tard, on constate que l'on dispose d'une pléthore de formalismes et d'algorithmes, mais qu'aucun d'entre eux ne résout tous les problèmes.

De fait, c'est une situation normale étant donné la complexité du domaine. Il est donc nécessaire, avant de commencer un travail dans notre domaine, de bien choisir le formalisme, et d'en discerner le statut. Doit-il modéliser finement quelques phénomènes de la langue, permettre d'extraire des informations sémantiques, traiter avec robustesse tous les textes ?

Nous tenterons de mettre en évidence les différences entre les diverses approches que l'on peut rencontrer dans le domaine, et de situer le présent travail dans ce cadre.

Le but que nous nous fixons est la création d'outils pour mettre en forme un corpus afin de permettre des recherches non plus seulement lexicales mais aussi syntaxiques. L'ensemble de ces aspects est détaillé dans le chapitre 1, consacré à la problématique.

0.2 Plan de l'étude

0.2.1 Analyse de corpus

Avant de vraiment choisir le formalisme, nous commencerons par considérer un corpus, et par en établir la grammaire d'une manière un peu formelle. De cette manière, nous pourrons par la suite mieux apprécier l'adéquation des formalismes inforatiques que nous utiliserons.

Le corpus choisi est le *papyrus Westcar*. Il s'agit d'un texte littéraire, assez long pour un texte égyptien non religieux. Il a pour nous un l'avantage d'être grammaticalement assez simple. En effet, les textes plus complexes comportent souvent des passages sur lesquels les spécialistes sont en désacord, voire ne se prononcent pas. De plus, les constructions de Westcar sont relativement bien marquées.

Une analyse formalisée de ce corpus est détaillée dans le chapitre 2 de ce document.

0.2.2 Problèmes de segmentation

Le chapitre 3 est consacré à la décomposition du texte en ses constituants, ce qui signifie une procédure de segmentation, et la mise en place d'un lexique. Le problème de la segmentation du texte, et plus généralement, du passage d'un codage du texte hiéroglyphique vers une forme utilisable pour l'analyse syntaxique n'est pas simple à résoudre dans le cas général ; il n'y a pas unicité de la solution, et la complexité est potentiellement exponentielle. Nous montrons quelles solutions nous proposons et décrivons la mise en œuvre de notre lexique hypertexte, muni d'un translittérateur. Cet outil est important pour l'utilisateur égyptologue, dans la mesure où nous mettons en place une double structure, dictionnaire et base de donnée textuelle, avec des liens de navigation adaptés aux besoins de la recherche et de la translittération des hiéroglyphes.

0.2.3 Application d'un paradigme classique d'analyse syntaxique

Le chapitre 4 est consacré à l'apport théorique et pratique de ce travail en matière de conception d'un analyseur. Nous avons choisi de traiter le problème de l'analyse syntaxique de notre texte, en partant d'un formalisme classique pour étudier ses limitations. Nous détaillerons les algorithmes à utiliser, puis les résultats obtenus. L'accent sera principalement mis sur la difficulté d'obtenir une analyse à la fois couvrante et non ambiguë. Lorsqu'un système tel qu'un analyseur syntaxique ne donne pas entière satisfaction, deux solutions sont envisageables : adapter la grammaire, d'une part, et modifier le système lui même, d'autre part. Nous considèrerons successivement ces deux approches, en remarquant pour terminer que des méthodes très simples, telles que les automates finis, peuvent donner des résultats intéressants. Cette remarque nous conduira à proposer une remise en perspective de notre système d'analyse syntaxique, et introduira les extensions et la structure couplée automate-grammaire qui nous a semblé la plus adaptée pour le TAL. Le discours en langage naturel, dans quelque langue que ce soit, est rarement « grammatical ». Or, c'est ce discours que le système automatique doit être en mesure d'analyser robustement ; le but n'est pas pas d'en évaluer la grammaticalité. Nous pensons que la robustesse de l'analyse d'un texte est très dépendante d'un couplage flexible de modèles différents ; le couplage que nous proposons apparaît comme une solution théorique possible, dont nous tenterons d'évaluer la faisabilité. Selon la formule de P. Vernus (Vernus 1990, p. V), « Investigations should rely on the available data, not on the available grammars. »

0.2.4 Analyse guidée par l'usage

Le chapitre suivant (5) est justement consacré à l'évaluation de cette faisabilité par la mise en œuvre du modèle bi-partite automate-grammaire. Nous proposons une architecture où un système simple représente les constructions fréquemment usitées, afin de pallier le très grand nombre d'analyses proposées par le système évoqué dans le chapitre 4. Nous détaillerons l'architecture du nouveau système, et ses composants.

0.2.5 Évaluation et conclusion

L'architecture et les programmes précédemment décrits sont évalués selon plusieurs critères :

- la couverture et l'efficacité pour le corpus présent, *versus* l'extensibilité pour d'autres corpus ;
- la comparaison avec d'autres systèmes ayant les mêmes prétentions ;
- la généralisation de notre modèles en termes méthodologiques.

Cette évaluation est décrite dans le dernier chapitre de notre document. Nous terminons ce même chapitre (6) en suggérant des perspectives de « portage » de notre démarche vers des problèmes non forcément linguistiques qui peuvent se poser en informatique : la modélisation à partir de données rares ou corrompues, ou la structuration et la segmentation de flots d'entrée font partie des thèmes susceptibles de bénéficier de notre travail.

Chapitre 1

Problématique

Sommaire

Nous allons exposer ici l'articulation de notre recherche, en précisant d'abord les objectifs que nous nous sommes fixés, puis en examinant les données dont nous disposons. Nous conclurons en exposant les contraintes qui en résultent.

1.1 Buts

Comme mentionné en introduction, l'objet du présent travail est la réalisation d'un système de marquage grammatical en vue de la constitution de bases de données linguistiquement riches.

Nous traitons donc des problèmes de l'analyse syntaxique appliquée ; nous étudions plus spécifiquement les questions qui surgissent lorsque l'on aborde des corpus réels, en souhaitant disposer d'un système robuste.

Le but final est en fait plus ambitieux, mais dépasse le cadre de ce manuscrit. Nous voulons proposer un environnement complet et intelligent qui permette d'informatiser la quasi-totalité des fichiers réalisés par le philologue, et en particulier ceux qui ne sont sont pas *a priori* structurés. On peut déjà observer que les traitements de texte sont plus ou moins utilisés par les chercheurs pour écrire des notes. Si ce pas a déjà été franchi, dans le but probable, dans un premier temps, d'avoir une plus grande qualité d'impression, de gagner de l'espace physique, et enfin de pouvoir utiliser directement les fiches lors de la réalisation de documents destinés à être échangés, il est certain que la disponibilité même des documents sur des supports informatiques va faire naître le besoin de les organiser. C'est souvent réalisé lorsque les données sont structurées ou facilement dénombrables. Ainsi, les bases de données archéologiques sont monnaie courante.

Nous voudrions cependant aller plus loin et proposer à terme un système qui

- travaille essentiellement sur des données peu structurées (typiquement du texte intégral) ;
- permette l'enrichissement du texte par un gloses[1] ;
- permette de faire simplement des liens entre plusieurs textes, comme par exemple entre plusieurs variantes d'une même œuvre ;
- permette à l'utilisateur de définir lui-même les structures qu'il utilise, si le besoin s'en fait sentir ; et cela sans exiger de lui une expertise informatique ;

1. Rajout d'informations sur un texte, que ce soit un commentaire ou une traduction

– permette la réutilisation et l'échange entre différents travaux ; par exemple, l'incorporation dans un dictionnaire personnel de fiches lexicographiques de provenance diverse, ou la citation d'un travail extérieur.

Les travaux de la TEI *(Text Encoding Initiative)* (Ide 1995), brièvement mentionnés en introduction, apportent des éléments de réponse quant aux modes de représentations des données, mais il y a un important travail d'interfaçage et d'automatisation partielle du glosage.

Le présent travail vise simplement à construire semi-automatiquement une représentation syntaxique sur laquelle pourront s'appuyer des recherches de mots en contexte. Une telle structuration est surtout utile dans le cas où l'on souhaite traiter statistiquement un corpus. L'étude lexicologique habituelle peut probablement se satisfaire de la recherche brute des termes.

Nous préciserons donc (en 1.5) sous quel angle nous considérons l'analyse syntaxique.

Dans le prochain paragraphe, nous resituerons le cadre général de notre travail par rapport à la problématique engendrée par les travaux du type de ceux de la TEI, travaux que nous présenterons.

1.2 Construction de bases de données

Le document « intelligent » fait de plus en plus partie des priorités du TAL. Le concept existe depuis déjà quelques années, mais il semble que l'accroissement spectaculaire du volume des données disponible doive consacrer le phénomène. Les travaux sur la représentation de textes enrichis que mène la TEI (collectif 1994b) ainsi que les bases de textes qui existent pour les langues classiques (projet Perseus), montrent que le philologue a réellement l'usage de l'ordinateur.

Il s'agit donc pour nous de considérer notre travail comme un outil pouvant, à terme, permettre la réalisation de bases de données textuelles « intelligentes » pour le moyen égyptien. Nous entendons par là une base qui permettrait d'effectuer des recherches sur des liens syntaxiques, non plus sur la simple occurrence de mots. Au delà des préoccupations directement liées à l'avancement théorique de l'analyse syntaxique en TAL, notre domaine d'application a des retombées possibles non négligeables sur le plan des fond documentaires archéologiques informatisés.

1.2.1 La TEI

La TEI est probablement le projet le plus important dans la domaine de la saisie électronique de documents déjà existants. Il s'agit de proposer un standard de représentation relativement souple, à partir du langage SGML, *Structured General Markup Language,* afin de faciliter *l'échange* des textes. Le document (collectif 1994b) décrivant ce standard de présentation, est disponible sur papier et par le réseau ; il explique la syntaxe utilisable pour annoter les textes.

SGML est un langage qui permet de décrire formellement le marquage d'un texte. Il est possible de définir des symboles de base, afin, par exemple, de résoudre les problèmes liés aux standards de codage des caractères. Il est aussi possible de décrire la *grammaire* (au sens informatique du terme) des documents à saisir. Ces grammaires sont appelées *document type definition.* L'exemple de la figure 1.1 donne un format sommaire décrivant un article scientifique ; celui-ci a un titre, et est composé de plusieurs *sections,* ayant chacune leur titre. La DTD est utilisée pour valider et structurer des documents comme celui de la figure 1.2. Il est bien entendu possible de décrire des structures biens plus complexes, et en particulier d'utiliser des pointeurs entre différentes parties. On constatera que le principe central de SGML est le parenthésage : chaque partie est entourée de marques `<MARQUE>` (début) et `</MARQUE>` (fin).

Un document rédigé en SGML a tendance à être très structuré ; de fait, ce formalisme a été conçu pour la rédaction de documentations techniques. Les textes dont s'occupe la TEI sont extrêmement variés. En général, ils n'ont pas été conçus pour s'intégrer dans une structure informatique. Le principal problème vient du caractère enchevêtré des structures rencontrées. Un document est généralement organisé selon plusieurs axes indépendants. Par exemple, la structure logique du texte – mettons, sa séparation en paragraphes – n'a pas de lien avec certains éléments de sa structure physique, comme la pagination. Dans l'absolu, on rencontre des documents où ces deux structures se croisent. Si `<page>` et `<p>` sont respectivement les marqueurs de page et de paragraphe, on peut rencontrer la configuration suivante :

```
<page>
 <P>texte .....</P>
 <P>texte .....</P>
 <P>texte .....
</page>
<page>
   suite du paragraphe précédent...</P>
 <P>texte .....</P>
</page>
```

Or, le chevauchement

```
<page><P></page><page></P></page>
```

est illégal.

SGML propose un certain nombre de méthodes pour résoudre le problème ; il est possible d'annoter un document à l'aide de plusieurs marquages concurrents, ou de recourir à des pointeurs pour réunir des éléments séparés par le formalisme, mais logiquement liés. La TEI propose donc diverses méthodes pour noter les informations qui rentrent difficilement dans le cadre d'une structure parenthésée, telles que les variantes des textes ou des commentaires philologiques.

Le standard fournit aussi une notation à l'usage des linguistes ; celle-ci est extrêmement générale, de manière à pourvoir être utilisée quelque soit le cadre théorique dans lequel s'inscrit l'annnotation.

Ces mécanismes entraînent cependant des notations assez lourdes, et rendent souhaitable l'usage d'outils de composition automatique.

FIG. **1.1** – DTD pour un article

```
<!element article - - (TITRE,SECTION+)>
<!element TITRE - - (#PCDATA)>
<!element SECTION - - (TITRE,CORPS)>
<!element CORPS - - (#PCDATA)>
```

FIG. **1.2** – Exemple de texte en SGML

```
<!doctype article system "article.dtd"
[
]>
<article>
  <titre>Analyse Syntaxique </titre>
  <section>
    <titre>Introduction</titre>
    <corps>Du texte ...</corps>
  </section>
  <section>
    <titre>Conclusion</titre>
    <corps>Encore du texte ...</corps>
  </section>
</article>
```

1.2.2 Quelques exemples de travaux

Le chevalier à la charette,

de Chrestien de Troyes, est l'objet de deux travaux d'édition électronique, l'un réalisé à Faculté des Lettres et Sciences Humaines de Nantes, et l'autre à l'université de Princeton. Il s'agit dans les deux cas de l'édition d'un certain nombre de manuscrits, le codage en SGML permettant de représenter en texte un certain nombre de caractéristiques de ceux-ci. La représentation du texte (figure 1.3) rend surtout compte de ses caractéristiques graphiques ainsi que des liens entre les différents manuscrits. Il s'agit surtout de proposer au lecteur une navigation aisée entre les différents textes.

FIG. **1.3** – Exemple de l'édition de Lancelot à Princeton `http://www.princeton.edu/~ lancelot/`

```
A C E T 35 S iriche com au ior e&s;tut
        36 A &p-hbar;&s; mangier ne &s;e remut
        37 L iroi&s; dentre le&s; compagn&o-hbar;&s;
        38 M l&apost;t ot &e-hbar; la &s;ale baron&s;
FOLIO 34v IMAGE
COLUMN a
        39 &et1; &s;i fu la roine en&s;amble
A C E T 40 S iot auec li come &s;amble
        41 M ainte bele dame cortoi&s;e
        42 B &n-hbar; &par;lant en langue fra&n-hbar;coi&s;e
        43 &et1; ke&s; ki ot &s;erui a&s; table&s;
        44 M angoit auec le&s; cone&s;table&s;
```

L'index des Textes des sarcophages

L'index des Textes de Sarcophages[2] est un index informatique des textes des sarcophages tels qu'édités par A. De Buck. Ceux-ci constituent l'un des corpus de textes les plus importants pour le moyen Égyptien. Ils ont été fort utiles aux grammairiens, en particulier grâce à leurs nombreuses variantes, qui ont permis d'observer de légères variations paradigmatiques et de mieux établir la valeur d'un grand nombre de constructions.

La recherche dans l'index se fait sur la translittération[3] des mots, éven-

2. disponible au `http://www.ccer.ggl.ruu.nl/ct/pirei6.html`
3. la *translittération* d'un texte en hiéroglyphes est la transcription des mots de ce texte en caractères latins.

tuellement accompagnée de leur catégorie syntaxique. Il est possible de faire la recherche sur un couple de mots. Le système permet la visualisation des textes trouvés (et non pas simplement de leurs références).

C'est dans l'état actuel des choses un excellent outil lexicographique, qui permet d'éviter la manipulation de l'édition papier des textes. Quoique cela semble être peu de choses, la rapidité de la recherche et de la consultation incite beaucoup plus à l'exploration et à l'exhaustivité que ne le ferait un index sur papier (lequel au reste n'est pas édité).

Il n'y a pas encore eu, en revanche, de travail effectué sur cet index, dans des domaines plus automatisés de la recherche sur les textes, comme, en particulier, les analyses statistiques. Cependant, les données existantes permettraient de procéder à de tels travaux.

Le Thesaurus Linguæ Græcæ

Le TLG (Donald 1990) est une base de données dont la création remonte déjà à 1972 ; il regroupe tous les textes connus de la Grèce Antique, entre -800 et 600. Pour ce faire, les auteurs du TLG ont mis au point un codage du grec ancien en caractères ASCII[4]. Tel qu'il existe, le TLG est principalement une base de donnée en texte intégral ; il contient peu d'informations *sur* le texte. Néanmoins, rien n'empêcherait d'en rajouter, si ce n'est le problème des droits commerciaux, qui pourraient éventuellement empêcher la distribution d'un texte annoté.

Ce point est sans doute sans portée du point de vue théorique du traitement automatique des langues, mais il est en pratique extrêmement important et courant. Il est idéal de posséder pleinement les corpus sur lesquels on travaille, puisque dans ce cas, l'on a la pleine liberté de décider comment ils pourront être échangés. Travailler sur des corpus commerciaux est souvent nécessaire, ne serait-ce que parce que les moyens peuvent manquer pour créer son propre corpus. Il serait alors bon de disposer d'un standard de notation permettant de lier de façon univoque un commentaire à un corpus.

Utiliser les coordonnées des caractères dans un fichier n'est pas une bonne méthode ; en cas de modifications du corpus lors d'éditions postérieures, les coordonnées peuvent devenir obsolètes. L'idéal est que le corpus lui-même fournisse des points de repères dont la fiabilité soit garantie.

Les textes égyptiens étant généralement établis[5] à partir d'un petit nombre de sources, la meilleure méthode est probablement d'utiliser les manuscrits eux-mêmes comme source du système de coordonnées, et non, comme le font les latinistes ou les hellénistes, d'utiliser une édition particulière.

4. on trouve ce codage à l'adresse `http://www.tlg.uci.edu/~tlg/BetaCode.html`

5. En philologie, établir un texte signifie tenter de lui donner sa forme première ;

Le projet Perseus :

Ce projet[6], similaire au TLG, mais plus récent, apporte une dimension supplémentaire au corpus utilisé : chaque texte est *lié* à sa traduction ; cette simple mise en regard augmente considérablement les possibilités de la base de données. Du simple point de vue de l'interrogation, elle multiplie les entrées vers un passage ou vers un concept.

Énumérant les possibilités de PERSEUS, F. Charpin (Charpin 1994, p. 13–20) montre en particulier qu'en cherchant un terme dans la traduction, le système permet de retrouver plusieurs mots — synonymes ou non — dans le texte de départ, ce qui dépasse les possibilités d'un simple index.

Cela ne va pas sans risque. En faisant état de cette fonctionnalité, F. Charpin écrit : « un tel traitement permet d'espérer la reconnaissance exhaustive de tous les concepts présents dans une œuvre. » Cet optimisme n'est pas forcément infondé mais mérite discussion du point de vue du TAL.

L'informatique peut effectivement rendre le philologue plus à même d'être exhaustif. C'est bien évidemment le cas lorsqu'il s'agit de trouver un terme précis, d'autant plus que l'on dispose depuis quelques années de systèmes de recherche tolérant de légères variations de graphie, par exemple `egrap`, qui est utilisé dans Glimpse (Manber et Wu 1993), un système de recherche en texte intégral.

Cependant, l'informatique, dès que l'on quitte des problèmes bien définissables, se révèle plus un instrument d'*exploration* que de *validation.*

Quelles sont donc les méthodes exploratoires possibles ? D'abord, l'alignement de corpus multilingues (Blank 1995), qui fonctionne sur une idée analogue à celle évoquée plus haut, mais en la systématisant. L'un des grands intérêts des méthodes d'alignement est que les concepts qui s'expriment au moyen d'un seul mot dans le texte d'origine et dans sa traduction pourront être déterminés automatiquement, fournissant ainsi un premier tri. Leur limitation principale sera qu'il ne s'agit que d'un premier tri. L'usage d'un vocabulaire varié, l'absence de correspondance exactes, la difficulté d'aligner des phrases qui ne sont pas forcément, dans une bonne traduction, exactement en regard, rendent l'alignement parfois hasardeux. En tout état de cause (et c'est vrai de la plupart des techniques d'explorations automatiques de textes) il faut considérer qu'il fournit une certaine vision du texte, ni exaustive, ni sûre, et qu'il demande donc une validation, et un complément, ce dernier pouvant être assuré par l'usage d'autres méthodes de recherche.

L'autre grande famille de méthodes d'exploration de textes est bien entendu l'étude statistique. Quelques unes ont été réalisées sur des textes égyptiens (Prévot 1992), mais tout reste à faire dans ce domaine en égyptologie.

6. `http://www.perseus.tufts.edu/`

L'usage d'éditeurs intelligents pour les humanités

Une équipe de l'IRISA (projet Opéra) développe un éditeur structuré, GRIF, dont la fonction principale est l'édition de documents scientifiques et techniques. Une expérience a cependant été menée, pour la création d'un index « actif », sur des actes médiévaux (André et Richy 1994). Cette expérience a montré l'intérêt de tels éditeurs, qui s'adaptent fort bien aux besoins des philologues ; et en particulier aux références croisées, qui sont l'un des principaux outils de l'édition.

Par contre, il est peu probable qu'un outil prévu pour l'édition de documents structurés soit adapté au glosage d'un texte proprement dit. En effet, les liens et la structure sont alors relativement lâches. Il ne s'agit pas de faire rentrer le texte à toute force dans un cadre rigide, mais bien plutôt de disposer d'une structure suffisamment souple pour s'adapter à lui. Dans le cas qui nous occupe, de plus, le texte est disponible *avant* structuration. Seuls, donc, des éditeurs permettant le marquage *a posteriori* du texte conviendront.

1.2.3 Quelques problèmes induits par la structuration des textes

L'humaniste n'est pas forcément à la recherche d'un texte comportant une interprétation. Il est en effet fréquent que celles-ci ne fassent pas l'unanimité ; cette opinion est exprimée avec force dans la critique du guide d'encodage de la TEI réalisée par le groupe travaillant sur la littérature (collectif 1994a) :

> Literature scholars are not interested in, in fact many object vehemently to, the perspective of obtaining texts which already contain — explicitly or implicitly — literary interpretations. The responses and comments elicited by the Survey bear eloquent witness to this.

C'est probablement d'autant plus vrai dans le cas où l'interprétation n'est que le support d'une recherche automatique, ou d'un travail statistique : les erreurs peuvent alors être masquées par le processus de traitement. Ceci dit, il peut être souhaitable de coder *des* interprétations, à condition de fournir leur origine, citer leur auteur, et de permettre la gestion d'une multiplicité de commentaires.

On constate dans le document qu'il existe un clivage entre la communauté informaticienne qui travaille sur la production et l'utilisation de textes, et la communauté des philologues. En effet, les premiers militent pour une représentation sémantiquement riche du texte ; celui-ci vaut principalement par l'information qu'il fournit et la facilité qu'il y a à la récupérer. La représentation physique du texte est alors un *produit* de cette représentation sémantique. Au contraire, pour le philologue, le manuscrit est premier ; l'interprétation qui est faite d'une particularité graphique est le résultat d'un travail, et, comme tel, il doit être possible de le critiquer au vu du document.

Il faut que nous tenions compte de cette attitude lors de l'élaboration du codage grammatical que nous utiliserons. Le système de représentation linguistique de la TEI est d'ailleurs très général, et la volonté explicite de ses auteurs est d'être indépendants d'une théorie donnée. En l'absence de consensus, il est préférable de rester neutre, c'est vrai en particulier pour un format d'échange comme celui proposé par la TEI, mais c'est aussi un objectif désirable dans une recherche comme la nôtre.

Le problème d'un système qui propose une interprétation est de savoir *jusqu'où* aller pour être utilisable. Ce problème est très général ; il est indépendant de la manière dont l'interprétation est produite, en ce sens qu'il se pose aussi lorsque l'interprétation est créée manuellement.

Une interprétation est une prise de position, et ce sur deux axes. Elle peut, d'une part, être plus ou moins juste : si un lemmatiseur analyse *souris* comme un verbe dans « le chat mange la souris », il se trompe. Dans le même registre, un terme peut être ambigu. Mais surtout, une interprétation se fait dans un certain cadre théorique, et ce cadre n'est pas neutre.

Pour rester dans le monde de la lemmatisation, on peut rappeler que le choix des catégories syntaxiques, qui implique une certaine vision de la grammaire, et qui, d'autre part, précise la forme du résultat souhaité, influe beaucoup sur la réussite quantitative du processus, ce qui ne laisse pas de poser des problèmes pour l'évaluation de tels systèmes (Véronis et Khouri 1995).

Dans le domaine des bases de données égyptologiques, Jean Winand, souligne dans son article *les bases de données de textes en néo-égyptien* (1990) combien il est important de ne pas perdre de vue les données initiales, car elles seules, vierges d'interprétation[7] peuvent convenir en cas de problème ou d'ambiguïté.

Il est évident qu'un système qui veut proposer une information aussi détaillée qu'une amorce d'analyse grammaticale ne peut éluder la question en se rabattant sur les sources. De plus, une analyse grammaticale suppose une théorie grammaticale, sujet sensible s'il en est. Le choix de celle-ci ne se fera pas uniquement sur des critères linguistiques, les critères informatiques entrant en ligne de compte.

En conséquence, il faut que l'utilisateur potentiel soit averti de ces problèmes, et en tienne compte. La base de donnée grammmaticale est selon nous plus un instrument d'exploration que de validation. Elle peut aider le

7. ou presque : même un *facsimile* est une interprétation

linguiste à formuler des hypothèses, mais il sera sain de s'en méfier lorsqu'il s'agira de vérifier les dites hypothèses. En particulier, les analyses de la base de données ne sauraient prétendre à l'exhaustivité.

L'autre volet est que la représentation doit correspondre aux recherches probables. Il faut pouvoir extraire un schéma actantiel[8] correct, et pouvoir chercher les circonstants possibles suivant un verbe.

1.3 Informatique et Égyptologie

L'égyptologie a commencé à faire usage d'informatique dès la fin des années 60, mais assez sporadiquement. L'une des applications les plus ancienne est un index de citations (Crozier-Brelot 1972). À la même époque, W. Schenkel proposait d'appliquer à l'égyptien les méthodes de la linguistique formelle (Schenkel 1972) et réalisait concurremment un index lemmatisé et informatisé d'un texte des sarcophages (Gundlach et Schenkel 1970) (différent de celui cité en 1.2.2).

En 1984 se tient la première réunion du groupe Informatique et Égyptologie (Plas 1994). Les disciplines relevant de l'archéologie utilisaient déjà l'informatique depuis plusieurs années, en particulier pour automatiser les inventaires réalisés au cours des fouilles.

Parmi les applications les plus marquantes de l'informatique en égyptologie, on peut citer la reconstitution du temple de Karnak en CAO, qui a, en retour, permis d'améliorer les procédures de conception utilisées par EDF, mécène du projet ; et, toujours à la DER, l'utilisation d'un système expert pour reconstituer les reliefs des temples d'Akhenaton à Karnak à partir de descriptions standardisées de l'iconographie des pierres éparses qui les constituaient.

Cela montre incidemment qu'encore une fois, les applications ont un effet positif sur l'amélioration des méthodes et procédures de l'informatique et qu'elle sont un moteur de l'évolution de notre discipline.

1.4 Informatique et langue Égyptienne

Certains philologues ont utilisé, à partir des années 70, l'informatique comme outil. Ils se sont heurtés au problème de la représentation des textes et l'ont généralement résolu en ayant recours à la translittération, solution qui entraîne malheureusement une perte d'information.

8. Le terme *actants* désigne les unités syntaxiques les plus directement liées à un verbe : son sujet, son objet direct, et son objet indirect.

L'ensemble des hiéroglyphes n'étant pas clos, le codage des textes est délicat ; de plus, la puissance du matériel était encore récemment tout juste suffisante pour que la saisie soit un peu conviviale. Après quelques essais d'édition électronique d'hiéroglyphes à la fin des année 70 (Hainsworth 1979) , sur les textes des pyramides, il apparut comme souhaitable de disposer d'un format standardisé pour la saisie ; un tel format est adopté en 1984 lors des premières réunions du groupe *Informatique et Égyptologie ;* ce format porte le nom du texte qui le décrit : le « manuel de codage » (Buurman, Grimal, Hainsworth, Hallof, et Plas 1988). Cette standardisation était nécessaire si l'on voulait disposer un jour de bases de données. Le *manuel* est toujours utilisé, avec des modifications qui permettent une représentation iconographique plus fidèle ; on peut cependant lui reprocher d'être essentiellement tourné vers l'impression de textes et peu extensible.

Jean Winand a utilisé des bases de données informatiques pour réaliser tant ses travaux sur *les mésaventures de Wenamun* que ceux sur la morphologie du néo-égyptien (Winand 1992) ; il a en particulier tenté une ébauche de lemmatisation automatique (Winand 1986; Winand 1990). Ses recherches sur la morphologie du néo-égyptien traitent en particulier de la variation de l'écriture des formes grammaticales sur quelques centaines d'années. Le décompte statistique qui en résulte est bien évidemment aidé par l'informatique, encore qu'une partie seulement du corpus soit informatisée, pour des raisons de ressources humaines. La saisie informatique est longue, coûteuse si on la fait réaliser par des vacataires, et peu rentable en termes académiques.

La translittération automatique a fait l'objet de quelques essais effectués par Michael Hainsworth (Hainsworth 1982), en associant à chaque signe sa valeur la plus courante. Tout récemment, Sophie Billet s'est attaquée à ce problème, en le considérant sous l'angle du transfert d'expertise (Billet 1995). La translittération est importante parce qu'elle est le moyen le plus simple pour communiquer des mots égyptiens, le recours aux hiéroglyphiques ne s'imposant qu'en cas d'ambiguïté.

Nous avons, quant à nous, abordé le problème de deux côtés : d'une part, la translittération d'un mot donné, et d'autre part, la *reconnaissance* d'un mot. Cependant, le thème étant périphérique par rapport à l'analyse syntaxique, nous avons reporté son étude approfondie.

Nous sommes attachés, dans un premier temps, à poser les bases de l'analyse syntaxique automatique du moyen égyptien, en vue de réaliser un système qui permettrait de simplifier la recherche, et, en particulier, de fournir *rapidement* une amorce de réponse à des questions telles que *quels compléments d'objet admet tel verbe ?* ou *a-t-on des occurrences assurées de l'infinitif dans telle construction ?* Le chercheur étudiant en détail tel ou tel phénomène de la langue, y trouverait un corpus de départ ; mais surtout, notre travail permettrait d'effectuer rapidement un certain nombre de ces recherches ponctuelles que celui qui étudie un texte est amené à conduire lorsqu'il veut éclaircir tel ou tel point grammatical ou lexical ; la question étant alors généralement de savoir comment les constructions syntaxiques ou lexicales rencontrées dans le texte étudié se comportent dans les autres textes. C'est là une des finalités de notre système, pour l'utilisateur égyptologue (et plus généralement linguiste). Sa finalité informatique sera examinée tout au long des prochains paragraphes.

1.5 Analyse syntaxique, couverture et profondeur

Il y a plusieurs manières de concevoir l'analyse syntaxique automatique. On peut vouloir mettre en œuvre une théorie grammaticale ; on peut aussi chercher à obtenir une analyse aussi fine que celle que produirait un humain. On peut enfin essayer d'analyser le plus de textes possibles.

Le premier point de vue est plutôt le fait de linguistes théoriques ; il s'agit pour eux de proposer un modèle dont la puissance explicative est l'une des caractéristiques importantes, (voir par exemple Chomsky, *Aspects de la théorie syntaxique* (Chomsky 1971)) et de tester son fonctionnement.

Les deuxième et troisième points de vue sont les plus courants. En général, les analyseurs se situent entre ces deux extrémités, et opèrent *de facto* un compromis entre les deux (Constant 1991). Notre but étant utilitaire, nous choisissons naturellement d'adopter le dernier point de vue, c'est à dire de viser à traiter le plus de textes possible, et non à traiter le mieux possible peu de textes.

En effet, une analyse « couvrante » du texte est nécessaire dans une application qui a vocation à traiter du texte fourni sans contrainte. L'état des recherches et nos moyens propres font que cette analyse sera forcément peu précise. Nous montrerons d'ailleurs pourquoi tout au long de ce document. La précision requiert une foule d'informations qui n'appartiennent pas toutes à la syntaxe ; il n'existe pas encore de méthode fiable *et générale* pour résoudre les multiples problèmes qui ressortent à la sémantique, de la pragmatique, qui ne manquent pas de se poser lorsque l'on cherche un traitement fin.

Notre système propose à l'utilisateur un petit nombre d'analyses et laisse le soin au spécialiste, d'en choisir une. C'est un type de fonctionnement raisonnable pour le Traitement du Langage Naturel ; sauf cas d'espèce, il est douteux que ce contrôle par un lecteur humain soit évitable. Notre système propose des analyses ; il a pour but, en réalisant une partie importante du travail, de permettre à un expert de se concentrer sur les réelles difficultés. Son but premier n'est pas de conseiller un utilisateur non expert, ce qui demanderait, outre une représentation de la langue en tant que telle, une méta-représentation des connaissances de l'interlocuteur humain. Le programme peut bien entendu proposer systématiquement les analyses possibles (voir page 115 pour un exemple) ; mais en règle générale, celles-ci seront trop nombreuses pour que cela constitue une aide intéressante pour le traducteur. De plus, il est peu fréquent de se retrouver dans un cas où une telle fonctionnalité serait utile.

Notre système se doit d'être utilisable. Il faut donc que les analyses qu'il propose soient, malgré tout, en nombre restreint et qu'elles soient de bonne qualité. II faut d'autre part que le système fonctionne en un temps raisonnable en utilisant un espace mémoire raisonnable. Il faut donc disposer d'une évaluation par le système des analyses qu'il fournira, et d'une méthode de limitation du temps de calcul.

1.6 Données

Nous allons ici présenter les contraintes que nous imposent les données disponibles.

L'absence de segmentation explicite dans un texte peut se rencontrer dans un nombre non négligeable de textes. Elle pose un certain nombre de problèmes d'ambiguïté, en particulier lorsque l'on rencontre des mots inconnus. Or l'orthographe égyptienne est relativement flottante, ce qui rend difficile la distinction entre un mot inconnu et une orthographe nouvelle. De plus, la grammaire du moyen égyptien est plutôt synthétique[9], mais les variations des mots apparaissent peu, car elles étaient souvent portées par les voyelles, qui ne sont pas écrites. En conséquence, le nombre d'ambiguïtés grammaticales est élevé.

Cet aspect est intéressant parce qu'il est partagé par plusieurs langues modernes, dont l'arabe et l'hébreu. Le type d'ambiguité de ces langues peut être rapproché du moyen égyptien. Il ne serait pas déraisonnable d'envisager, réciproquement, report de nos méthodes de désambiguisation de l'égyptien vers le traitement syntaxique automatique de ces langues.

9. c'est-à-dire que la langue utilise un grand nombre de formes pour un même mot, par opposition à une grammaire analytique, dans laquelle les tournures sont rendues par des périphrases. Le même phénomène se produit en français, le passé simple, synthétique, est remplacé par le passé composé, analytique.

1.6.1 Corpus utilisé

Il est peu de textes égyptiens dont une partie ne résiste aux efforts des traducteurs. Nous avons donc jugé préférable de choisir un texte simple comme base de notre étude ; il n'était pas raisonnable d'ajouter aux difficultés informatiques des difficultés philologiques.

Par ailleurs, il fallait un texte un peu étendu. Les plus grands corpus sont fournis par les textes religieux, mais il posent fréquemment des problèmes grammaticaux et lexicaux. Il était donc préférable de se tourner vers des textes littéraires. Parmi ceux-ci, nous avons retenu le papyrus Westcar, car il est raisonnablement simple, et contient peu de passages qui pourraient embarrasser un grammairien.

Le texte comporte un peu plus de 2500 mots, ce qui est assez long pour un texte narratif égyptien.

Le papyrus Westcar est cependant très particulier, en ce sens que la langue qu'il contient est très évoluée. C'est de l'égyptien classique, mais il annonce par bien des points l'état de langue qui suivra.

1.6.2 Problèmes de segmentation

Notre système a enfin pour vocation d'être un cas d'école pour l'analyse des textes sans segmentation. Cela recouvre les textes sans ponctuation, c'est à dire bon nombre de textes anciens, car la ponctuation systématique des textes est un phénomène relativement récent, mais cela peut aussi concerner des textes modernes, dont la méthode de saisie n'aurait pas pris en compte la ponctuation. C'est partiellement le cas du langage parlé lorsqu'il est retranscrit ; c'est souvent le cas du japonais, du chinois, ou de l'arabe.

Le but poursuivi est ici de mettre en place quelques techniques d'analyse adaptées à ces textes.

La question de la segmentation se pose pour les *mots*, pour les *syntagmes*, et pour les *phrases*. En règle générale, les systèmes d'analyse syntaxique s'occupent du second cas, mais les deux autres furent longtemps laissés au second plan. La disponibilité actuelle de très grands corpus, provenant souvent du monde de l'édition électronique, rend la résolution de ces problèmes nécessaire.

La reconnaissance de mots, et en particulier de mots composés, est une activité fort importante, en particulier dans le cadre de l'indexation automatique. Certaines catégories de mots, notamment les noms propres, sont par nature ouvertes ; d'autre part, les corpus scientifiques ou journalistiques sont grands consommateurs de néologismes ; dans les deux cas, il est fréquent que le terme soit composé.

Dans le cas des corpus provenant de systèmes de saisie automatique, soit par reconnaissance optique de caractères, soit éventuellement par reconnaissance vocale, il est possible que les espaces ne soient pas détectés. Il est donc souvent nécessaire, à des degré divers, de pouvoir procéder à une segmentation du texte en mots. Dans le cas d'écriture ne comportant pas d'espacement entre les mots, ce problème est évidemment bien plus important.

Le découpage du texte en syntagmes, quant à lui, est consubstantiel à l'analyse syntaxique.

Enfin, le découpage du texte en phrases est nécessaire pour permettre de borner l'analyse à des parties raisonnablement courtes de l'entrée ; elle ne va cependant pas d'elle même.

Dans un texte provenant, par exemple, d'une bande typographique, la ponctuation n'est pas toujours présente à tous les endroits où elle serait nécessaire ; il arrive en effet que la disposition initiale ait permis de la sous entendre. Il nous est souvent arrivé de rencontrer des articles de journaux où des titres secondaires, en gras, étaient directement suivis par le texte qu'ils introduisaient. Le procédé ne posait aucune difficulté au lecteur ; il en allait bien entendu différemment pour l'ordinateur. Le codage typographique, en outre, n'est pas forcément univoque — et nous avons pu constater, cette fois-ci dans le texte d'un dictionnaire, que les bornes fixées par le changement de code typographique ont tendance à être moins cohérentes que ne le seraient des séparations formelles. Il faut souligner que le travail de segmentation est multiple : dans le cas de textes d'origine diverse (journal, livres, dictionnaires, etc.), les règles de segmentations peuvent varier considérablement (Rajman 1995a).

Une première tâche est donc la séparation et l'identification des mots. Nous l'avons abordée dans un premier temps, mais il apparaît que le sujet mérite un travail à lui seul, travail qui a d'ailleurs été réalisé depuis dans la thèse de Sophie Billet (Billet 1995). Nous suggérerons cependant, plus loin, quelques méthodes utilisables.

Pour ce qui est de la ponctuation, la conséquence de son absence presque totale[10] en égyptien est la suivante : les algorithmes analysant des textes à partir de grammaires hors contexte étant typiquement en $\mathcal{O}(n^3)$, il n'est pas envisageable de structurer un texte entier par la simple application d'un analyseur syntaxique. Nous envisagerons donc par la suite une méthode de découpage.

Nous pourrons en effet mettre à profit l'existence d'un certain nombre

10. Dans certains textes, probablement versifiés, les propositions sont séparées par des points rouges. Il s'agit cependant d'une exception. On rencontre aussi, à un niveau supérieur, des marques de début de chapitre.

de formes grammaticales qui impliquent une segmentation du texte. Il s'agit d'une série de constructions qui introduisent de nouvelles sections, et sur lesquelles s'appuient les propositions qui les suivent. Nous en parlons plus avant en 2.4.3 ; pour l'instant, contentons-nous de signaler qu'il est possible grâce à ces formes d'obtenir un découpage grossier du texte. Cependant, la longueur des segments qu'elles délimitent varie selon le style de l'auteur. Les grandes œuvres classiques, en particulier *Sinouhé,* déroulent de très longues phrases, dont la structure grammaticale est extrêmement complexe.

1.6.3 Les recherches grammaticales en moyen égyptien

Le déchiffrement des hiéroglyphes par Champollion en 1822 ressuscitait 3000 ans d'histoire d'une langue. On conçoit que durant ce laps de temps, celle-ci avait évolué. Cependant, cette idée, qui fait partie de la vulgate linguistique au XX$^{\text{e}}$ siècle, n'était pas complètement acquise au début du XIX$^{\text{e}}$. Vers la fin du siècle, l'école Berlinoise, en particulier A. Ermann, mettent en évidence les états successifs de la langue.

Au XX$^{\text{e}}$ siècle, les contributions de Sir Alan H. Gardiner, avec son *Egyptian Grammar* (Gardiner 1963), puis celles de Hans Jacob Polotsky (Polotsky 1976), qui introduit les méthodes de la linguistique structurale en égyptologie, ont façonné ce qu'est aujourd'hui l'étude du moyen égyptien, la langue classique de l'Égypte antique (classique au sens où elle fut ressentie comme telle par les Égyptiens eux-mêmes).

L'Egyptian Grammar est un ouvrage double, à la fois grammaire d'initiation et de référence ; elle repose sur une linguistique essentiellement morphologique et sémantique.

L'apport de H. J. Polotsky (Polotsky 1976) fut de distinguer, sur la base de comparaisons en contexte, un certain nombre de formes verbales, et de mettre en lumière les valeurs nominales ou adverbiales que celles-ci pouvaient avoir (voir en particulier 2.4.11). Les derniers avatars de sa théorie, liés à une application quelque peu rigide du Générativisme, sont maintenant remis en question ; la grammaire actuelle semble accorder plus d'importance à la diachronie, et se défier d'une vision par trop mathématisée de la langue.

1.7 Contraintes

Le cadre de la réalisation de notre étude, et les idées que nous voulons développer, nous imposent un certain nombre de contraintes, que nous allons détailler ci-dessous. Ces contraintes sont :

- l'absence d'un gros corpus informatique préexistant ;

– la multiplicité des problèmes ;

– la volonté d'aboutir à un analyseur de large couverture ;

– la limitation de nos ressources humaines.

L'absence d'un gros corpus informatique préexistant : Il n'existe pas encore de gros corpus *informatisé* de textes égyptiens ; cette situation est probablement transitoire, dans la mesure où certains des obstacles à la saisie d'un corpus ont disparu. Il existe maintenant un format de codage qui peut prétendre être un standard, et des outils de saisie raisonnablement performants. Plusieurs problèmes subsistent cependant. En premier lieu, le nombre des personnes disponibles pour faire ce travail est réduit. D'autre part, des problèmes, intrinsèques à la langue et à son écriture, subsistent : la liste des signes n'est pas close, et la procédure d'intégration d'un nouveau code, qui consiste à passer par des codes temporaires, rend lente la publication des nouveaux signes.

On peut espérer que ce système fonctionnera plus vite grâce au réseau ; le petit nombre d'utilisateurs du codage est à ce stade un avantage puisqu'il facilite la communication ; il serait alors souhaitable de proposer un protocole de mise à jour permettant d'utiliser le nouveau signe de manière robuste.

La multiplicité des problèmes : Le moyen égyptien a été peu abordé jusqu'à présent par l'informatique linguistique. Son traitement suppose plusieurs étapes, et nous avons dû faire des choix. Ces étapes sont :

— **L'analyse des signes :** Les problèmes posés par la reconnaissance optique de caractères ont été plus ou moins résolu en ce qui concerne les caractères latins imprimés. L'intérêt d'une semblable procédure pour les hiéroglyphes est limité. D'une part, on se trouve en face de plus de 800 signes ; d'autre part, il est rare de disposer d'une qualité de gravure suffisante sur les monuments. Quand aux textes saisis par les égyptologues, la majeure partie d'entre eux ont été édités sous forme manuscrite, ce qui rendrait bien plus aléatoire la récupération automatique des corpus. La seule méthode raisonnable est donc la saisie manuelle des textes.

— **La segmentation et l'analyse lexicale :** Dans le cas de l'égyptien ancien, la segmentation est *a priori* complexe ; en effet, il n'y a pas de marque explicite de fin de mot. Néanmoins, la situation est plus favorable que dans le cas d'une écriture alphabétique sans ponctuation, dans la mesure où certains signes se trouvent préférentiellement en fin de mot ; ce sont les déterminatifs

(2.1.2). Ils permettent un découpage grossier du texte, limitant ainsi l'espace de recherche.

— **L'analyse syntaxique elle-même :** Les problèmes évoqués précédemment ont de fortes répercussions sur l'analyse syntaxique ; comme leur résolution ne peut être que partielle, ils augmentent la combinatoire de l'analyse syntaxique.

la volonté d'aboutir à un analyseur de large couverture Il nous a semblé que notre projet impliquait, de part sa nature même, de proposer une analyse couvrante du texte. En effet, un programme qui se veut une aide au spécialiste a peu à gagner à proposer ponctuellement, pour un nombre réduit de phrases, une analyse fine. Cela, l'utilisateur en est capable, et bien mieux que le programme ne pourrait l'être. Par contre, si le programme peu prendre en charge le tout-venant des phrases, il sera alors d'une utilité certaine.

Les implications de ce choix sur la conception de notre analyseur sont multiples, un facteur central se dégage : le but poursuivi implique une attitude méthodologique souple. En particulier, le formalisme employé est validé ou invalidé selon son adéquation au projet. Ainsi, le cadre théorique formel varie, dans une certaine mesure, tout au long de la recherche, en fonction des obstacles rencontrés. On n'hésitera pas, par exemple, à prendre un formalisme peu puissant dans l'absolu mathématique, si ce formalisme s'avère localement efficace.

Inversement, on n'accordera pas de valeur ontologique au formalisme utilisé (ce qui ne signifie pas qu'il n'en puisse avoir, mais bien plutôt que ce problème précis dépasse le cadre de cette étude puisqu'il relève de l'épistémologie des modèles).

la limitation de nos ressources humaines Ce problème est étroitement lié aux précédents. Il nous interdit d'essayer les méthodes d'apprentissage automatique (en particulier pour la lemmatisation, qui réclame souvent des corpus de l'ordre du million de mot), puisque nous ne pouvons pas les créer ; il nous a aussi conduit à nous consacrer à l'analyse syntaxique, au détriment de la translittération automatique.

1.7.1 Extensibilité

Un analyseur à grande couverture doit bien se comporter dans un certain nombre de cas ; s'il traite des textes réels, il doit être tolérant aux erreurs ; il serait regrettable de perdre une information simplement parce que l'auteur du texte a omis une préposition grammaticalement nécessaire, et plus encore parce que l'auteur n'a pas noté quelque terminaison qui allait de soit.

Extension synchronique

Pour être robuste, un analyseur doit pouvoir gérer au mieux les irrégularités qui se présentent lors du traitement d'un texte. Ces irrégularités peuvent correspondre à des erreurs du scribe ; elles peuvent aussi correspondre à des lacunes de la grammaire, qu'il s'agisse de la grammaire élaborée par les linguistes ou de la grammaire formelle utilisée par la machine.

Dans le cas de l'égyptien, une « erreur » du texte, c'est-à-dire un passage erroné selon la grammaire égyptologique en vigueur, est parfois un terme rare, mais légitime. De même, d'ailleurs, telle tournure française est regardée comme impropre par les puristes, mais est en réalité passée comme règle *de fait* dans la langue telle qu'on la pratique. Comme ce type de tournure est souvent annonciateur de modifications à venir dans la langue, cela nous amène à considérer le problème de la diachronie.

extension diachronique

Un système d'aide à la création de bases de données peut, nous l'avons dit, structurer « faiblement » le corpus qu'il traite. En un sens, les limitations techniques recoupent heureusement les exigences probables de l'utilisateur (voir 1.2.3), puisque dans les faits, celui-ci se méfie d'un système qui lui en dit « trop ».

Comme l'analyse est superficielle, il n'est pas concevable qu'elle soit limitée en couverture. L'analyseur ne doit pas être restreint à un seul texte : son existence n'aurait alors plus aucun sens.

Dans le cas de l'égyptien, les corpus homogènes sont rares. De plus, le but de notre programme est d'être une aide à la construction de bases de données. Il faut donc pouvoir traiter le plus grand nombre possible de textes. Mais ceux-ci sont espacés dans le temps ; il faut donc, tout en nous cantonnant à un seul état de langue, le moyen égyptien, essayer de le couvrir le plus possible, depuis les textes de la fin de la sixième dynastie jusqu'à la date la plus tardive possible.

Pour cela, il est souhaitable de distinguer quelques grandes structures qui resteront invariantes des tendances qui marqueront les divers états.

1.8 Problèmes de formalismes

La plupart des formalismes destinés à l'analyse syntaxique se rapportent, de près ou de loin, à la hiérarchie des grammaires telle qu'énoncée par N. Chomsky (1957); les grammaires sont vues comme des systèmes de réécriture permettant de générer des phrases bien formées.

1.8.1 Paradigmes d'analyse

Historiquement, le traitement automatique des langues et la linguistique se sont beaucoup côtoyées. Il semble cependant que leurs problématiques respectives doivent être bien séparées, en particulier lorsque l'on évoque le Générativisme.

Générativisme et grammaires formelles

Dans le paradigme typique du Générativisme, une grammaire est :

> *Les règles qui caractérisent les séquences bien formées d'unités syntaxiques minimales et qui assignent une information structurale de nature diverse à ces séquences aussi bien qu'aux séquences qui s'écartent en quelque façon de la bonne formation.*

d'après Chomsky (Chomsky 1971).

Dans les faits, la dernière partie de la proposition a souvent été négligée, du fait même de la recherche d'un bon formalisme pour décrire la *norme*.

Ce paradigme tire sa force de plusieurs racines ; la tradition de la grammaire prescriptive, notamment, qui se donne pour but de décrire une forme « légale » de la langue. La richesse théorique de la théorie mathématique des langages vient encore renforcer cette manière d'aborder l'analyse syntaxique.

Il est effectivement impressionnant que ces systèmes, somme toute assez naturels, s'accommodent de traitements mathématiques féconds, et que la vérification de la grammaticalité d'une phrase en fournisse une analyse.

Cependant, dans le cas de l'analyse automatique d'une langue morte, on peut se demander si ce point de vue est pertinent. En cas de conflit entre un texte et une grammaire déjà établie, le texte peut avoir raison. Mais surtout, un texte, même erroné, est porteur d'information. Le fait de savoir s'il est grammaticalement correct ou pas est secondaire : d'un texte donné, nous voulons extraire le plus de matière possible. Il s'agit donc ici de considérer la grammaire du point de vue de la *structuration,* et non pas de la *génération.*

1.8.2 Les grammaires hors contexte

Le formalisme hors contexte occupe une place centrale dans le traitement des langues, et ce pour plusieurs raisons.

En premier lieu, il reflète l'expression usuelle des règles de grammaire. Considérons la citation suivante, extraite de la grammaire de G. Lefèbvre (1955, p. 280) :

> **L'ordre des mots** dans une phrase verbale est, en principe, le suivant : 1 verbe 2 sujet 3 complément d'objet 4 datif 5 complément circonstanciel, représenté soit par un adverbe, soit par un substantif ou un pronom suffixe précédé d'une préposition

Ce passage se traduit immédiatement :

```
phrase verbale ==> verbe sujet cod datif cc.

cc ==> adverbe.
cc ==> preposition suffixe.
cc ==> preposition substantif.
```

la grammaire traditionnelle ayant ensuite pour usage d'énoncer les exceptions et cas particuliers contrevenant à ce type de règles.

Du reste, le formalisme lui-même a été utilisé par le grammairien indien Panini pour la grammaire du Sanscrit entre 200 et 400 av. J.-C. (Ingerman 1967; Auroux 1989). Il a donc une longue histoire derrière lui.

Qu'un certain nombre de linguistes se soit donné la peine de mettre en évidence des constructions grammaticales irréductibles à ce formalisme est, d'une certaine façon, une preuve indirecte de sa qualité ((Gazdar et Mellish 1989, p. 133-134) pour un exemple de la forme $a^m b^n c^m d^n$).

La popularité des grammaires *hors-contexte* en informatique linguistique est aussi liée à l'existence de bons algorithmes d'analyse, qui proviennent essentiellement du domaine de la compilation (Aho, Sethi, et Ullman 1989).

Les algorithmes développés pour analyser les langages informatiques ont été extrêmement optimisés, ce qui les rend intéressants. Par contre, ils imposent généralement des limitations sur le langage et sur la forme de la grammaire ; c'est le cas par exemple des analyseurs LALR. Les outils informatiques de construction de compilateurs, tels que *lex* et *yacc*, ne sont pas adaptés à l'analyse du langage naturel. De manière plus générale, dans le domaine de la compilation, c'est le langage qui doit avoir été défini pour le générateur d'analyseur, et non le contraire. Nous avons utilisé ces outils pour segmenter des dictionnaires, et nous basant essentiellement sur des marques typographiques ; notre conclusion est que le logiciel résultant est impossible à maintenir ou à réutiliser. Le but même de *yacc* est en fait incompatible avec l'analyse d'un texte dont le formalisme serait imprécis.

Un certain nombre de *variantes* de ces algorithmes sont cependant utilisables. M. Tomita (Tomita 1985) a proposé un algorithme efficace en temps et en espace, représentant de façon compacte l'ensemble des analyses possibles d'une phrase. Briscoe (1993) par exemple, met à profit le mécanisme d'analyse des automates à piles, en dotant les actions *shift* et *reduce* de l'automate de probabilités, pour obtenir un analyseur probabiliste capable d'apprentissage. Il en résulte que le système obtenu est difficile à interpréter, car une partie de la connaissance est codée sous une forme qui ne s'interprète pas immédiatement en terme de langue. Cela dit, ce problème se retrouve à des niveaux divers dans tous les systèmes qui recourrent à des données numériques.

Les grammaires hors-contexte permettent, en toute généralité, d'analyser un texte en un temps proportionnel au cube[11] de la longueur de celui-ci. Cela les rend utilisables, mais peu pratiques pour des textes longs.

Pour en revenir à l'expressivité des grammaires *hors-contexte*, les critiques sont de deux ordres.

Une première critique est d'ordre théorique, au sens plein du terme : il existerait des constructions de la langues impossibles à retranscrire dans ce formalisme. Cela revient par exemple à montrer que certaines constructions exigent de reconnaître $a^n b^n c^n$, où a, b et c sont des symboles de la grammaire, et où l'exposant la ré-duplication n fois du symbole. De telles constructions sont très rares ; il en existe un exemple en suisse allémanique (Gazdar et Mellish 1989; Shiebert 1985; Huybregts 1985) : on rencontre des constructions du type $GN^m Verbe^n$, où les GN sont les objets des verbes. La langue étant dotée de déclinaisons, certains verbes construisent leur objet à l'accusatif, certains autres au datif. On peut donc avoir des phrases du type

$$\mathrm{GN}_a^m \mathrm{GN}_d^n \mathrm{Verbe}_a^m \mathrm{Verbe}_d^m$$

les Verbes_a prennent leur objets à l'accusatif, les GN_a étant lesdits objets ; il en va de même pour les Verbe_d. La construction a la forme $a^n b^m c^n d^m$, qui n'est pas représentable par une grammaire hors contexte. On remarquera cependant que ce type de construction est extrêmement complexe, et qu'il y a peu de chances pour qu'elles soient très longues. On remarquera de plus que la représentation sommaire de cette règle par $\mathrm{GN}^n\mathrm{Verbe}^m$ rentre, elle, tout à fait dans le cadre de ce formalisme.

Ce type d'argument théorique met en fait en évidence un paradoxe : il n'est valable que si n peut être arbitrairement grand. Or, la pratique de la

11. La limite théorique est en fait un peu meilleure

langue limite la complexité des constructions imbriquées. Il n'est donc pas gravissime de ne pas pouvoir toutes les représenter dans le formalisme *hors-contexte*. D'un autre côté, cette réponse elle-même met en évidence le caractère peut-être trop généraliste des grammaires formelles générativistes. En fin de compte, elles manquent de restrictions plutôt que d'être trop pauvres en constructions. Elles proposent une certaine vision de la langue, comme un ensemble de possibles ; la distinction Saussurienne entre Langue et Parole n'est pas vraiment utilisable en analyse automatique, puisqu'aussi bien nous sommes toujours confrontés à la parole (Marandin 1993). Il faut donc adapter cet ensemble de possibles à la forme du texte effectif ; c'est-à-dire qu'il faut guider les analyses.

Il faut de plus se garder d'accorder à une grammmaire, quelle qu'elle soit, un statut auquel elle ne peut prétendre. La langue n'est pas représentable « dans l'absolu ». Si elle l'était, ce serait bien le seul objet naturel qui se laisserait *complètement* décrire par un formalisme.

Un autre reproche qui peut être fait aux grammaires *hors contexte* est que, somme toute, elles ne fournissent pas toujours les meilleurs outils de représentation.

Par exemple, le formalisme simple ne permet pas de représenter aisément le mécanisme des accords. Il faut pour ce faire multiplier les entités : Des

Code 1.1 Gestion de l'accord par une grammaire Hors Contexte

```
groupe_nominal_singulier -->
          nom_singulier, adjectifs_singulier.
groupe_nominal_pluriel -->
          nom_pluriel, adjectifs_pluriel.
```

extensions plus ou moins simples du formalisme ont donc été proposées. Du point de vue du traitement automatique des langues, ces extensions ne sont utilisables que si elles débouchent sur des procédures automatisables ; aussi les formalismes évolués que N. Chomsky a proposés ont été peu employé en traitement automatique.

L'extension la plus courante est de rajouter au système hors contexte des structures de traits, qui peuvent représenter aussi bien les catégories lexicale d'un mot que l'analyse complète d'une phrase. Dotées de telles structures, les grammaires hors contexte constituent un formalisme grammatical raisonnablement puissant, en ce sens qu'elles ont une bonne expressivité, que les constructions qui leur échappent sont peu nombreuses, et qu'il existe de bons algorithmes pour ce type de formalisme.

1.8.3 Systèmes d'analyse

Un certain nombre d'études ont déjà choisi, explicitement ou non, d'essayer de structurer le texte au lieu de vérifier son adéquation à la grammaire.

L'analyseur de P. Constant (1991)

Cet analyseur syntaxique repose sur un modèle syntaxique simple, composé de groupes et de liens entre groupes. La grammaire en tant que telle n'est pas représentée ; en effet, le système, partant d'un découpage sommaire et minimaliste, procède, de manière impérative (grâce à du code C), à une ré-organisation incrémentale des données. Cette ré-organisation est réalisée par une série de sous-programmes, qui prennent en charge des problèmes spécifiques (par exemple, le groupe des pronoms autour du verbe ou la recherche du sujet).

Le système fonctionne de la façon suivante : le rédacteur de la grammaire écrit des fonctions qui sont appliquées à chaque mot du texte, dans chaque interprétation possible. Il est possible, dans une fonction, de se déplacer de manière transparente dans le texte, comme si celui-ci ne comportait pas d'ambiguïtés. Le texte est en fait représenté sous forme de graphe, et des points de synchronisation permettent de localiser l'ambiguïté. Si un déplacement passe par un point de synchronisation, les diverses branches correspondant aux interprétations sont générées, et l'analyse est reprise depuis le début sur la nouvelle structure. La réduction de l'ambiguïté se fait en rajoutant des interprétations et en supprimant des arcs. Tout ceci est uniquement procédural.

Ce système est très bien adapté pour la résolution de liens locaux ; il est par contre plus difficile à manier pour les liens de grande portée. le système envisage rarement des constructions très ambigües, et est de ce fait rapide et robuste.

Il s'inspire des théories syntaxiques de Lucien Tesnière (1959), et met explicitement en œuvre la *translation,* que Tesnière tient pour l'un des trois ressorts essentiels de la syntaxe, avec la *coordination* et la *jonction.* La *translation* est le phénomène qui permet à un groupe syntaxique donné d'acquérir une valeur syntaxique différente de celle qu'il a par défaut. Ainsi, « ma voisine » est un groupe nominal ; il peut prendre place dans toute construction syntaxique qui demande un nom ; par exemple, il peut être sujet d'un verbe. En rajoutant « de » à ce groupe, il peut prendre la place d'un adjectif : « le chat *de ma voisine* » fonctionne comme « le chat *tigré* ». De même, une proposition comme « il mange » devient adjectivale grâce à un pronom relatif : « qui mange ». La représentation explicite des translations peut permettre d'effectuer une analyse syntaxique locale, puis de prendre en compte les marques de translations, et ainsi de mieux localiser l'ambiguïté. La translation a été utilisée, pour l'égyptien, par François Daumas (1962), égyptologue et éditeur posthume de l'œuvre de Tesnière, pour étudier les formes relatives, et est, sous le nom de transposition, au centre de la théorie de H. J. Polotsky sur le verbe.

Les unités syntaxiques utilisées par l'analyseur sont très particulières, surtout dans les premières phases de l'analyse. En effet, le premier découpage effectué délimite des « groupes syntaxiques » qui sont, en gros, des amorces possibles de groupes nominaux, verbaux, ou prépositionnels. Ce découpage s'effectue grâce à des mots tels que les articles, les prépositions, et aux ponctuations. Ce serait évidemment problématique pour une langue comme l'égyptien, qui ne comporte pas de ponctuations, et qui, de plus, fait beaucoup moins usage de mots grammaticaux que le français ou l'anglais.

L'analyseur d'E. Brill (1992)

Eric BRILL a récemment proposé un paradigme général d'apprentissage et d'analyse pour résoudre plusieurs problèmes en langage naturel (de Loupy 1995; Brill 1993b).

Le système apprend à modifier un texte annoté, l'usage des annotations variant de la lemmatisation à l'analyse syntaxique.

On part d'un texte dont on connaît la forme annotée correcte. On se donne un canevas permettant d'énoncer des règles de transformation ; un procédé arbitraire permettant d'obtenir une annotation initiale du texte, et une procédure d'évaluation de l'annotation courante. Le système apprend une séquence de règles à appliquer pour passer de l'annotation initiale à l'annotation finale. La méthode d'apprentissage est simple : on dote l'espace des annotations d'une métrique, et on choisit à chaque étape la règle qui réduit le plus cette distance. Le processus s'arrête lorsqu'aucune règle ne permet d'obtenir une amélioration.

Une fois que cela est réalisé, le système s'utilise en appliquant systématiquement la séquence de règles trouvée sur les textes à analyser, après les avoir annoté arbitrairement.

L'usage le plus poussé de ce paradigme concerne la lemmatisation automatique ; le système donne des résultats comparables à ceux des systèmes statistiques, mais en utilisant des bases de données beaucoup plus réduites (Brill 1992; Brill 1994).

En analyse syntaxique, BRILL procède en trois phases :

lemmatisation le texte est d'abord lemmatisé, en utilisant un analyseur bâti sur le modèle que nous venons de décrire.

parenthésage le processus de parenthésage, que nous allons décrire *infra,* groupe les termes en syntagmes sans étiquettes ; le groupement est systématiquement binaire ;

marquage syntaxique enfin, une fois les syntagmes découverts, un dernier système les étiquette.

Le processus de parenthésage

initialisation au départ, le texte lemmatisé est parenthésé en groupant systématiquement à droite ; ainsi, la proposition qui précède sera initialement analysée comme :

```
(le/ART
  (texte/NC
    (est/AUX
      (parenthésé/PP
        (en/PREP
          (groupant/PPRES
            (systématiquement/ADV
              (à/PREP
                (droite/NC)))))))))
```

Le but étant d'apprendre à transformer le texte en :

```
((le texte)
 ((est parenthésé)
  (en ((groupant systématiquement)
       (à droite)))))
```

format des règles Il y a deux types de formats de règles possibles :

```
(ajouter|retirer) une parenthèse
    à (gauche|droite) des lemmes X
```

(où X est le nom d'une étiquette) et

```
(ajouter|retirer)
  une parenthèse (gauche|droite) entre des mots
étiquetés X et Y
```

évaluation La mesure employée est « le pourcentage de syntagmes au sens du système qui ne sont pas à cheval sur des syntagmes réels »

Les règles de transformation sont à première vue un peu complexes ; elles doivent en effet garantir que la structure du texte analysé restera celle d'un arbre binaire. Cependant, elles restent compréhensibles : la meilleure règle choisie par le système, par exemple, est « détruire les parenthèses gauche à gauche des noms communs », ce qui revient à dire qu'un nom commun n'est pas un début de syntagme.

L'étiquetage des syntagmes Les syntagmes définis par le système sont les groupes parenthésés. Ils sont étiquetés par des règles dont la forme générale est :

1. transforme l'étiquette en X si Y est une étiquette fille ;

2. transforme le l'étiquette en X si Y et Z sont des étiquettes filles contigues.

L'initialisation se fait en étiquetant tous les nœuds intérieurs comme des groupes nominaux.

Commentaire Outre le fait qu'il s'agisse d'un apprentissage automatique, ce qui, en soit, est déjà séduisant, cette méthode présente un intérêt dans sa conception même. Au lieu de vérifier l'adéquation du texte à un modèle, elle se contente de structurer son entrée. Tout texte sera ainsi analysable. La qualité de l'analyse obtenue est une autre affaire.

Il est clair que l'on peut approcher les mêmes résultats en modifiant l'analyse de grammaires Hors Contexte, en proposant par exemple les fragments de texte analysé les plus longs possibles ; cependant, la méthode d'E. BRILL a deux avantages. Le premier est que la robustesse est inhérente à la méthode. Le second est que la méthode est en $\mathcal{O}(n)$.

Ce dernier avantage suggère que vraisemblablement, le formalisme sous-jacent est relativement peu puissant (au sens des structures qu'il peut décrire), mais il est difficile savoir dans quelle mesure cela se répercute sur les possibilités de l'analyseur.

Un point nous interdit temporairement d'utiliser la méthode ici décrite. C'est un formalisme essentiellement adapté à l'apprentissage automatique. Disposant de données peu nombreuses, nous ne pouvons effectuer cet apprentissage, même s'il réclame des corpus moins fournis que ceux dont ont besoin les systèmes stochastiques.

On pourrait envisager de créer manuellement les séquences de règles utilisées par l'analyseur ; ce serait cependant irréaliste. Les règles en question sont peu signifiantes par elles-même, ce qui les rend difficile à écrire. D'autre part, leur ordre d'application est un facteur critique dans le déroulement du programme ; mais il est déjà difficile d'ordonner des règles dont le sens est clair ; des règles par trop locales seront ingérables sans automatisation.

L'analyseur d'Helsinki (1994)

L'analyseur syntaxique en question a été développé concurremment à l'Université d'Helsinki et au centre de recherches Xerox de Grenobles. l'analyseur en question se veut structurant et robuste. Dans la présentation qui en fut faite à COLING (Tapanainen et Järvinen 1994), le texte était analysé en deux étapes, l'une globale, et la seconde, locale. Le système fonctionne de la manière suivante :

Chaque mot du texte est suivi d'une description de ses fonctions possibles ; le système choisit une interprétation.

L'analyse globale utilise une description des positions relatives des mots ayant certaines fonctions dans la phrase, cette description étant un sous ensemble des automates finis. Par exemple, une règle peut être :

```
SUJET + ... + VAUXILIAIRE + ...
```

précisant que le sujet précède (pas forcément directement directement) un verbe auxiliaire. Chaque règle concerne un *sous ensemble* des fonctions possibles, et est construite à partir d'un corpus. Le système permet des généralisations de deux types : on peut grouper plusieurs fonctions sous une même appellation, et toute séquence qui se répète est susceptible de se répéter un nombre quelconque de fois. Le système conserve les analyses qui satisfont aux plus grand nombre possible de règles.

On remarque que les mots concernés peuvent être loin les uns des autres : cette première analyse tente donc de dégager les grandes lignes de la phrase. De plus, les liens syntaxiques sont sous-entendus : le sujet dans l'exemple de règle est implicitement le sujet de l'auxiliaire.

L'analyse locale permet de désambiguïser les mots dans un contexte réduit. Elle se fait grâce à une liste de contextes fréquents attestés. Entre deux contextes, elle choisit le plus long possible.

Le résultat de l'analyse est qu'à chaque mot sont associés une série de traits, dont la fonction du mot. L'indication de fonction est partielle, en ce sens qu'elle ne relie pas le mot au reste de la phrase. Ainsi, « they » dans l'exemple de la figure 1.4, page 37 est identifié comme sujet (`@SUBJ`), mais sans qu'il soit précisé de quel verbe il est sujet.

Une version plus récente de l'analyseur (Chanod et Tapanainen 1996) donne des résultat de même forme, mais augmentés d'un parenthésage qui permet entre autres de détecter les incises. Elle utilise un formalisme différent, celui des expressions régulières. Il semble que cette dernière version de l'analyseur ait été élaborée afin de donner aux créateurs de la grammaire plus de facilités et de contrôle. La première mouture avait par certain côté des affinités avec des systèmes d'apprentissage, en ce sens que le processus d'écriture de la grammaire faisait beaucoup intervenir le corpus, qui produisait lui même des règles. Le second s'apparente plus aux systèmes contrôlés de l'IA.

1.8.4 Conclusion

Le système d'E. Brill est un peu à part, étant essentiellement organisé autour de l'apprentissage automatique. Les deux autres systèmes partagent un certain nombre d'idées. En particulier, ils séparent les phénomènes locaux (par exemple l'organisation des pronoms autour du verbe) des phénomène globaux (les attachements prépositionnels, par exemple).

Nous sommes, quant à nous, partis des grammaires hors contexte, parce qu'elles se prêtaient au codage direct de la grammaire «classique». On verra par la suite que le système auquel nous avons abouti a plusieurs points communs avec celui d'Helsinki. En particulier, il fait usage de contextes récupérés dans un corpus. Cela semble montrer que les contraintes du type d'analyse que nous voulons réaliser appellent naturellement un mode de représentation privilégiant le corpus comme source de structure. Les différences essentielles entre notre système et le leur sont, du point de vue fonctionnel, que nous voulons pouvoir analyser un texte non ponctué, et que nous préférons pouvoir utiliser la puissance des grammaires hors contexte. La conséquence de cela est que nous disposons de deux formalismes au lieu d'un seul, le premier (à base d'expressions régulières) étant plus simple que celui utilisé à Grenoble, et le second, à base de grammaires hors contexte, étant plus puissant.

FIG. **1.4** – Exemple de sortie de l'Analyseur d'Helsinki
résultat de l'analyse de la phrase

> *they have an idea, but of course it doesn't mean that they would be able to talk*

```
"<*they>"
        "they" <*> <NonMod> PRON PERS NOM PL3 SUBJ @SUBJ
"<have>"
        "have" <SVO> <SVOC/A> V PRES -SG3 VFIN @+FMAINV
"<an>"
        "an" <Indef> DET CENTRAL ART SG @DN>
"<idea>"
        "idea" N NOM SG @OBJ @ADVL
"<$,>"
"<but>"
        "but" CC @CC
"<of=course>"
        "of=course" ADV @ADVL
"<it>"
        "it" <NonMod> PRON NOM SG3 SUBJ @SUBJ
"<does_>"
        "do" <SVO> <SVOO> <SV> V PRES SG3 VFIN @+FAUXV
"<_n't>"
        "not" NEG-PART @NEG
"<mean>"
        "mean" <Vcog> <SVO> <SV> V INF @-FMAINV
"<that>"
        "that" <**CLB> CS @CS
"<they>"
        "they" <NonMod> PRON PERS NOM PL3 SUBJ @SUBJ
"<would>"
        "would" V AUXMOD VFIN @+FAUXV
"<be>"
        "be" <SV> <SVC/N> <SVC/A> V INF @-FAUXV @-FMAINV
"<able>"
        "able" A ABS @PCOMPL-S
"<to>"
        "to" INFMARK> @INFMARK>
"<talk>"
        "talk" <SVO> <SV> <P/with> V INF @<NOM-FMAINV
```

Chapitre 2

Analyse du Corpus

Sommaire

Nous allons ici décrire la manière dont nous avons abordé notre corpus. Nous commencerons par un bref aperçu de son contenu, pour ensuite situer le texte diachroniquement.

Le texte utilisé, le Papyrus Westcar, est un recueil de contes. Il est organisé à la manière des *mille et une nuits,* en ce sens que le cadre global est lui-même un conte. Le lecteur intéressé trouvera une traduction du texte dans *romans et contes égyptiens*, de Gustave Lefèbvre (1949). Résumons ici le contenu du papyrus : le roi Khéops se fait raconter des prodiges par ses fils. La partie qui est conservée contient les contes suivants :

– un magicien, Ouba-iner, est trompé par sa femme. Il s'en aperçoit, et se venge au moyen d'un crocodile magique en cire ;

– le roi Snéfrou s'ennuie. Le très sage magicien Djadjaemankh trouve une solution : que le roi organise donc une petite fête nautique, où les beautés du palais, vêtues de résilles de perles, rameront sur l'étang, afin de « rafraîchir le cœur de sa Majesté ». Las ! un bijou appartenant à une rameuse tombe à l'eau, et avec lui le ballet nautique. Heureusement, Djadjaemankh veille, et, repliant l'eau comme s'il s'agissait d'un vulgaire tapis, il récupère le bijou à pied sec.

– Enfin, le prince Hardedef propose à Khéops de lui montrer un prodige contemporain : un vieux sage du nom de Djédi, qui sait recoller les têtes. Le magicien, mandé, s'exécute sur du bétail ; mais il va un peu plus loin et annonce la fin de la IVième dynastie (celle de Khéops) et l'avènement de la cinquième. Suit l'histoire de la naissance miraculeuse des rois de la cinquième dynastie, sous la protection de sage-femmes qui sont en fait des déesses envoyées par Rê, le dieu soleil.

Il est probable, au vu du contenu du dernier conte, que celui-ci date de la cinquième dynastie. Mais linguistiquement le texte est écrit en moyen égyptien plutôt évolué. En particulier, notre corpus fait un grand usage des adjectifs démonstratifs *pꜣ, tꜣ* et *nꜣ,* qui sont déjà presque utilisés comme des articles ; or cet emploi se généralise dans l'état de langue suivant c'est-à-dire le néo-égyptien. De plus, le papyrus Westcar, seule source connue pour ces contes, est datable paléographiquement [1] de la seconde période intermédiaire.

1. i.e. d'après la forme des signes.

La question de la datation du texte n'est pas anodine. Elle soulève en effet le problème de la gestion des états de langue. Notre texte, à certains égards, n'est pas écrit dans la langue classique — celle des grands textes littéraires de la XII^{e} dynastie. Il se peut (c'est en tout cas une hypothèse qui n'est pas écartée par plusieurs traducteurs, dont G. Lefèbvre (o.c.) et M. Lichteim) (1973), que la date du conte soit quand même la XII^{e} dynastie. Mais en ce cas, la différence entre le niveau de langue de Westcar et celui des textes classiques est telle que l'on peut considérer que nous avons ici affaire à deux systèmes différents. Lefèbvre parle, pour Westcar, d'une « langue extrêmement relâchée». Une stèle funéraire [2] du début de la XIIième dynastie nous livre d'ailleurs ce commentaire :

> Je parlais le langage des officiels, je ne disais pas *pꜣ*

pꜣ, démonstratif, puis article défini, est caractéristique du moyen égyptien tardif, puis du néo égyptien. Il est donc vu par l'auteur de la stèle comme la marque d'un langage récent et populaire ; or, cet article se trouve en abondance dans Westcar. Cela nous amène à poser qu'il est désirable d'agencer notre système afin de lui garantir une certaine souplesse face à ce genre de variations, faute de quoi sa validité serait étroitement limitée.

2.1 Présentation de l'écriture et de la langue

Nous présenterons d'abord le système d'écriture et la langue, pour ensuite détailler les points qui sont pertinents pour le Traitement Automatique des Langues.

2.1.1 Histoire de la langue

On trouvera un historique complet de la langue, ainsi qu'une description de celle-ci d'un point de vue linguistique dans l'ouvrage récent de Loprieno, *Ancient Egyptian, a linguistic introduction* (Loprieno 1995).

Les premières traces connues de la langue égyptienne datent de -3000, et la langue a survécu, comme langue liturgique des chrétiens d'Égypte, sous la forme du Copte, jusqu'au quatorzième siècle. Les différents états de la langue habituellement distingués sont :

l'ancien égyptien (-3000 à -2000) langue de l'Ancien Empire, qui nous est connue principalement par les *textes des pyramides* ;

2. citée par Parkinson (Parkinson 1991)

le moyen égyptien ou égyptien classique, est la langue littéraire du moyen empire. Elle a survécu ensuite comme langue sacrée jusqu'au 24 août 394, date de la dernière inscription connue ;

le néo égyptien (-1300 à -700) est la langue de l'époque des Ramsès ;

le démotique à la fois nom d'une langue et d'une écriture ;

le copte est le dernier état de la langue égyptienne ; il est fortement teinté de grec quand à son vocabulaire, ainsi d'ailleurs que dans son écriture, qui utilise l'alphabet grec augmenté de quelques signes hérités du démotique.

Ces états se séparent en deux grands groupes : l'ancien égyptien et le moyen égyptien d'une part, les autres d'autre part. Le premier groupe est caractérisé par des constructions synthétiques, et un riche éventail de formes verbales ; le second groupe correspond à un état essentiellement analytique, où la langue fait grand usage d'auxiliaires.

Nous nous occupons ici de la langue classique, qui est celle de la plupart des inscriptions hiéroglyphiques.

2.1.2 Le système d'écriture

L'écriture hiéroglyphique a pour principe fondamental le rébus ; mais il s'agit d'un système d'écriture, et le rébus est codifié. Il n'en reste pas moins que la nature iconique du signe demeure, et peut toujours être utilisée ; le système hiéroglyphique a cette particularité que, tout en comportant principalement un nombre réduit de signes (à peu près 800), il n'est clos, ni dans le nombre de ses signes, ni dans la gamme des valeurs de ceux-ci. Les nécessités de la communication demandant une certaine régularité, ces capacités productives se manifestent principalement lorsque le contexte est assez contraignant pour lever d'éventuelles ambiguïtés.

Les hiéroglyphes se classent en trois catégories : les signes phonétiques, les idéogrammes, et les déterminatifs.

Les signes phonétiques représentent 1, 2 ou 3 *consonnes*, l'égyptien, comme la plupart des langues sémitiques, ne notant pas les voyelles. Ces signes sont choisis selon le principe du rébus ; ainsi le scarabée, 𓆣, « khéper », sert à noter *le son* « képer ». Pour certains signes, notamment les unilitaires, qui notent une consonne unique, le système utilisé est plutôt l'acrophonie[3] : la valeur phonétique du signe est la première consonne du nom de l'objet représenté. Ainsi, si le signe 𓉐, qui représente un plan de maison, est usuellement utilisé pour représenter la suite de consonnes « pr », il peut cependant être utilisé, par acrophonie, pour représenter la consonne « p ».

les idéogrammes sont des signes-mots, qui notent l'objet représenté ; par exemple 𓃒 permet d'écrire le mot « taureau » ; en règle générale, un idéogramme utilisé comme tel est suivi d'un trait, pour bien spécifier cet emploi : 𓃒𓏤 ;

les déterminatifs sont des classifieurs sémantiques ; ils sont écrits en fin de mots et permettent d'en spécifier le sens ; par exemple, un mot abstrait sera terminé par le signe du rouleau de papyrus : 𓏛.

Ce classement des signes est un peu schématique, les diverses classes ayant tendance à se recouper ; il est assez vain de se demander si le signe 𓁹 est un idéogramme ou un phonogramme lorsqu'il sert à écrire le mot « œil », alors qu'il est évidemment phonétique lorsqu'il sert à écrire le verbe «faire», les deux se prononçant « iret ». De même, le signe du canard dans le mot 𓄿𓊪𓂧𓅭, « oiseau », est-il un idéogramme précédé de sa lecture phonétique ou un déterminatif ? L'on peut classer sans ambage certains signes dans l'une des trois catégories, mais pour d'autres, l'on passe de manière continue de l'une à l'autre valeur.

Néanmoins, les catégories en question sont utilisables, du moins par l'être humain. Informatiquement parlant, on peut les reprendre telles quelles, un signe ayant des valeurs possibles dans trois domaines : idéogramme, phonétique, ou déterminatif. Il est peut-être plus séduisant de coder les différentes transformations qui permettent de passer de l'un à l'autre de ces domaines. Un tel système aura le mérite de la souplesse ; il faudra cependant représenter explicitement les signes fréquents, certains jeux d'écriture trop virtuoses pour être prédictibles, et se doter d'un mécanisme de résolution de conflits.

Le système hiéroglyphique est présenté dans de nombreux ouvrages. Signalons notamment le catalogue de l'exposition *Naissance de l'écriture* (collectif 1982), et, plus récemment, un chapitre dans *Reading the Past, Ancient Writing from Cuneiform to the Alphabet* (Bonfante, Chadwick, Cook, Davies, Healey, Hooker, et Walker 1994).

3. Certains auteurs privilégient plutôt l'hypothèse selon laquelle on ne retiendrait, dans certains signes comportant des consonnes faibles, que la consonne forte pour obtenir un unilitaire. Serge Sauneron, dans *L'écriture figurative dans les textes d'Esna* (1982, p. 105), penche pour un système mixte.

2.1.3 Problèmes de morphologie

La morphologie égyptienne est simple, du moins en ce qui concerne la morphologie écrite. L'orthographe, par contre, est flottante ; elles varie selon les époques et les supports. Il n'est donc pas toujours simple de chercher un mot dans un lexique à partir de sa forme hiéroglyphique. En contrepartie, les véritables homographes sont rares ; les graphies vont jusqu'à distinguer explicitement différents sens d'un même mot : Ainsi, selon que l'on écrira ou l'on signifiera « travail » ou « serviteur ». De même, le verbe , « tirer », et le verbe , « transpercer du regard », de même racine, sont distingués par leurs déterminatifs.

Il y a donc peu de vrais problèmes d'homographie, sinon, mais c'est là le plus gênant, en ce qui concerne la forme grammaticale d'un mot.

2.2 Méthode d'analyse du corpus

Nous avons d'abord effectué une translittération complète du texte, afin de l'avoir sous une forme aisément manipulable ; puis nous avons dressé la liste de toutes les propositions (au sens grammatical du terme). À partir de cette liste, nous avons groupé les différentes tournures utilisés. Le travail d'écriture des grammaires a essentiellement consisté à représenter ces constructions grammaticales, en les généralisant de manière « raisonnable ».

Nous avons distingué discours et récit dans notre analyse. Comme le français (Émile Benvéniste 1974, pp. 237-250), l'égyptien possède des tournures spécialisées dans le récit. Il est d'autant plus naturel de les utiliser pour structurer l'analyse qu'elles sont bien marquées, et particulièrement fréquentes dans notre texte.

2.3 Notations utilisées

Dans ce qui suit, lorsque nous énoncerons une règle syntaxique, nous utiliseront les signes, symboles et conventions suivants :

- les signes `[]` parenthèseront des éléments optionnels ;
- les catégories terminales seront signalées en ***gras***.
- Les éléments constants, les mots-outils, seront notés en translittération suivis, entre parenthèse, de leur écriture en hiéroglyphes, et d'une éventuelle traduction entre guillemets ; par exemple, nous écririons : *pw* (, « ça »).

Alternativement, nous utiliserons des diagrammes syntaxiques.

Les exemples seront donnés de la façon suivante : sur une ligne, le texte hiéroglyphique ; sous chaque mot, sa *translittération*. Celle-ci correspond au *squelette consonantique* des mots. Chaque signe correspond donc à une consonne, et une valeur phonétique, en partie arbitraire, lui est assignée. Pour lire les translittérations, on prononce des « é » ou des « è » entre les consonnes. Certaines consonnes sont faibles et sont généralement prononcées comme des voyelles. La liste des symboles et de leur valeur est donnée dans la table 2.1.

Signe	prononciation	remarque
ꜣ	a	Prononciation très conventionnelle. Le signe a pu correspondre à une pause, à un « n » ou un « l »
ỉ	i	
ꜥ	â	consonne analogue au ayin de l'arabe
w	ou	
b	b	
p	p	
f	f	
m	m	
n	n	
r	n	
h	h	
ḥ	h très aspiré	
ḫ	jota espagnole	
ẖ	sh allemand	
s	s	
š	ch	
ḳ	q liquide	
k	k	
g	gu	
t	t	
ṯ	tj	
d	d	
ḏ	dj	

TAB. 2.1 – *Signes conventionnels de translittération*

2.4 Notions sur la grammaire égyptienne

2.4.1 Introduction

Nous allons d'abord détailler le vocabulaire utilisé. Une notion de base en syntaxe est le *syntagme*. Un syntagme est un groupe de mots entretenant des relations avec d'autres groupes de mots.

Un syntagme est dit *nominal*, *adjectival*, *adverbial*, *verbal*, selon qu'il peut assumer la fonction d'un nom, d'un adjectif, d'un adverbe ou d'un verbe.

exemple

le petit chat est un syntagme nominal

bleu est un syntagme adjectival

qui mange est un aussi syntagme adjectival

ici est un syntagme adverbial

dans la cuisine est un syntagme adverbial

Un énoncé est constitué par un lien spécial entre deux syntagmes, appelés *sujet* et *prédicat*. Claude Hagège(Hagège 1982, pp. 32 sp.) souligne le caractère syntaxique de la notion ; en effet, il est important de la distinguer d'un certain nombre de notions voisines en apparence, mais de caractère sémantique ou pragmatique. Il s'agit donc d'une notion parente de celle employée en logique informatique, mais différente sur le fond.

Le sujet est optionnel ; il peut en effet être sous entendu. Généralement, en français, comme les phrases contiennent presque toutes un verbe, le sujet est le sujet du verbe principal de la phrase, et le prédicat est composé par le verbe et ses compléments. En Égyptien, le prédicat peut être un syntagme quelconque ; ainsi, au lieu de recourir au verbe *être* comme le fait le français, l'égyptien peut dire

$$\underbrace{\textit{le soleil}}_{\text{Sujet}}\ \underbrace{\textit{est dans le ciel}}_{\text{prédicat}}$$

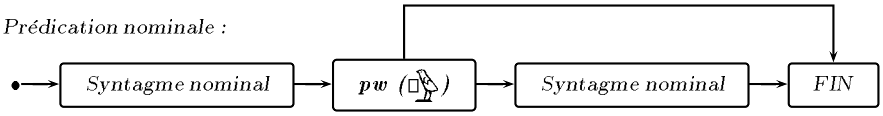
en écrivant simplement :

𓂋𓂝𓇳	𓅓	𓊪𓏏𓇯
rʿ	*m*	*pt*

$$\underbrace{\textit{le soleil}}_{\text{sujet}}\underbrace{\textit{dans le ciel}}_{\text{prédicat}}$$

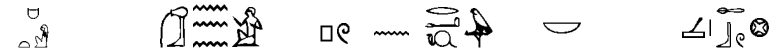
2.4.2 Prédication nominale, prédication adverbiale, prédication adjectivale

Le prédicat peut en égyptien être verbal, adjectival, nominal, ou adverbial. La prédication nominale établit l'appartenance du sujet à un groupe. Elle peut se faire par simple juxtaposition de deux groupes nominaux, mais la construction la plus fréquente emploie le pronom personnel *pw,* (« ça »), et est de la forme



Désignant par A le premier syntagme nominal et par B l'éventuel second, nous traduirions cette forme par : « *C'est un A, à savoir B* », soit « *B est un A* ». « *B* » est optionnel, et, sans lui. la phrase signifie « *c'est A* ». Par exemple :

Exemple 1 *WESTCAR* 0

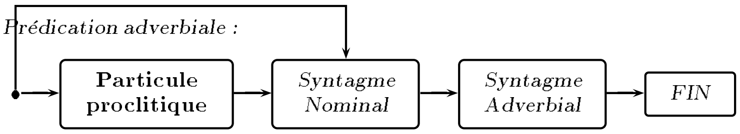


ḥmt wꜥb pw n rꜥ nb sꜣẖbw

(La) femme (d'un) prêtre cela de Rê seigneur (de) Sakhebou
C'est la femme d'un prêtre de Rê seigneur de Sakhebou.

Noter que « *pw* » peut couper, comme dans notre exemple, le groupe A si celui-ci est trop long. Ce qui fait que notre grammaire ne convient pas tout à fait ; il faut lui rajouter cette possibilité. Cependant, la modification ne correspond à aucun choix. En fait, on a une règle de réécriture que l'on pourrait exprimer comme « groupe nominal suivi de *pw* → groupe nominal où *pw* a pris la place normale de l'adjectif démonstratif. »

La *prédication adverbiale* précise l'état de son sujet ; elle a la forme



le syntagme adverbial pouvant être un adverbe, une préposition suivie d'un groupe nominal, ou une forme adverbiale du verbe.

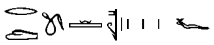
Exemple 2 *WESTCAR* 0

à propos de l'eau contenue dans un bassin

ỉw=f m mḥ 12 ḥr ỉ3t=f

elle en tant que coudées 12 sur profondeur-sienne
elle faisait 12 coudée[4] *de profondeur.*

La *prédication adjectivale* sert à affirmer que le sujet a intrinsèquement telle ou telle qualité ; les états passagers sont rendus par des constructions adverbiales. On conçoit qu'une telle forme est peu fréquente dans une narration, et de fait, nous n'en avons très peu d'exemples dans le papyrus Westcar :
Exemple 3 (p. 0, l. 0.)



rwḏ ḳs.w =f

solide os à lui
ses os étaient solides

La prédication verbale est un phénomène complexe en égyptien. En effet, la langue classique construit les verbes à l'aide d'un certain nombre de tournures, dont l'origine est essentiellement adverbiale. Ces tournures sont figées, et donc peuvent être considérées comme verbales ; elles sont cependant, d'un point de vue syntaxique, toujours interprétables comme des tournures adverbiales. De même, les tours modaux, apparentés à notre subjonctif, ont une affinité avec les syntagmes nominaux. De plus, les formes verbales en question peuvent apparaître dans des endroits où l'on attend un adverbe (respectivement un nom).

Nous avons donc gardé pour la commodité de l'analyse automatique ces formes dans les catégories nominales, adverbiales... en considérant qu'une analyse *a posteriori* permettrait de leur rendre leur statut de prédicats verbaux.

2.4.3 Formes initiales et séquentielles

La langue égyptienne classique comporte peu de structures subordonnées marquées ; elle a plutôt tendance à marquer le début de grandes unités textuelles. C'est-à-dire que les marques syntaxiques explicites désignent plutôt des formes *non subordonnées*, mais surtout des coupures dans la trame du discours ou du récit. Cela rend parfois le texte ambigu, même pour un traducteur humain. Il arrive en effet qu'il soit impossible de trancher entre une

4. En fait, le nombre 12 se prononçait avant l'unité

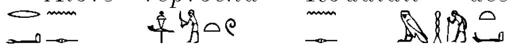
forme subordonnée, qui marque un circonstant de la forme principale qui régit la phrase, et une forme séquentielle, qui est placée sur le même plan que la forme principale, mais ne se distingue pas toujours morphologiquement de la forme subordonnée.

Ainsi, la phrase

Exemple 4 (p. 12, l. 9–10.)

ꜥḥꜥ.n *šnt.n* *rdddt* *ḫt* *n tꜣ wbꜣyt*

Alors *reprocha* *Roudidit* *des choses* *à la servante*



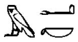
rd.n=s *ḫsf=tw* *n=s* *m ḥwt*

elle fit *qu'on punisse* *elle* *par des coups*

peut grammaticalement se comprendre de deux manières : soit *rd.n=s* est un verbe subordonné, auquel cas la traduction devrait être « Alors Roudidit se disputa avec la servante, *après* l'avoir fait battre », ou *rd.n=s* a une valeur séquentielle, auquel cas la traduction est : « Alors Roudidit se disputa avec la servante, *et* la fit battre ». Le sens permet de choisir le second terme de l'alternative.

Les marques diffèrent entre le récit et le reste du texte. En moyen égyptien, ce sont surtout les formes autonomes qui sont marquées. La marque est généralement le premier mot d'une proposition ; elle dépend du contexte, car elle porte aussi d'autres indications :

***iw*** marque du discours, a une nuance d'objectivité ;

***mk*** marque du discours, annonce la proposition comme un fait constatable ;

Les auxiliaires narratifs fonctionnent de la même manière et seront décrits plus avant en 2.4.11.

Les structures subordonnées et séquentielles se trouvent après la forme initiale ; on peut distinguer les formes verbales et les autres ; S'il n'y a pas de marque formelle, elles ont toutes une valeur qui peut être comprise, soit comme une valeur de subordination, soit comme une mise en séquence.

2.4.4 Les pronoms

Il y a trois types de pronoms en égyptien :

les pronoms suffixes

Ils s'accolent au mot ; ils servent à former des désinences verbales, à marquer la possession et sont aussi utilisés après les prépositions. On les note en translittération en les faisant précéder d'un signe « = ».

Exemple 5 (p. 0, l. 0.)

di͗=i͗
que-donne je
puissé-je donner

Exemple 6 (p. 0, l. 0.)

tp=f
tête sienne
sa tête

Exemple 7 (p. 0, l. 0.)

Hr=f
sur lui

Les pronoms dépendants

Ils sont utilisés comme compléments d'objet direct, et comme sujet dans certaines constructions. Leur nom provient de ce qu'ils sont toujours subordonnés à la présence d'un autre mot, particule, ou verbe, sur lequel ils s'appuient sans lui être forcément directement accolé.

Les pronoms indépendants

Ils sont principalement utilisés dans la précication nominales, comme sujet.

2.4.5 Système verbal

Le système verbal du moyen égyptien semble avoir essentiellement marqué l'aspect des verbes (duratif, accompli, non accompli, aoriste[5]), et non le temps. L'état de langue suivant, le néo-égyptien étant, lui, muni d'un système verbal doté de temps, on trouve, assez naturellement, des formes intermédiaires.

2.4.6 Syntaxe des syntagmes

Les syntagmes formés autour du verbe

Le verbe égyptien prend très facilement des valeurs adverbiales, adjectivales, ou nominales, en fonction de la forme verbale choisie. Le syntagme constitué par le verbe, son sujet, et ses divers compléments prend alors la valeur syntaxique correspondante.

Les syntagmes nominaux formés autour du verbe : Les formes du verbe pouvant avoir des valeurs nominales sont nombreuses. Plusieurs d'entre elles désignent l'action elle-même, et sont en particulier utilisables dans des propositions complétives, ou après des prépositions ; ce sont :

L'infinitif forme nominale du verbe sans autre marque, souvent utilisée quand on veut faire l'économie de l'expression du sujet ;

Le prospectif forme nominale à valeur modale ; peut aussi être employée avec une valeur adverbiale pour indiquer le but ; on peut le rapprocher du subjonctif français qui a des utilisations voisines : « Puisse-t-il manger ! » ou « je lui explique *pour qu'il comprenne* ».

La forme nominale accomplie *stp.n=f* forme nominale dont l'aspect accompli est marqué, mais dont le sujet est normalement exprimé ; les temps composés du français *classique* se rapprochent de ces valeurs : « quand *j'ai mangé,* je n'ai plus faim ».

La forme nominale *mrr=f* forme nominale non marquée modalement, mais dont le sujet est normalement exprimé. On la traduirait littéralement *mrr=f* par « *le fait qu*'il aime ».

5. Du grec a-oristos, non délimité. En égyptien, c'est une forme verbale qui ne marque pas la temporalité de l'action, et sert généralement à indiquer les actions quotidiennement répétées.

De plus, l'égyptien dispose de deux formes adjectivales, qui peuvent elles-même être marquées en fonction de leur aspect ; ces formes sont :

Le participe le participe du verbe X est une forme signifiant «celui qui X» ; par exemple *stp* peut signifier *celui qui a choisi.*

La forme verbale relative c'est une forme où le sujet du verbe, différent de l'antécédant éventuel de la forme, est exprimé : *stp=i͗, celui que je choisis.*

Syntagmes adverbiaux

Il existe trois formes à valeur adverbiale :

le *stp=f* circonstanciel est une forme adverbiale non accomplie : *quand je choisis.*

le *stp.n=f* circonstanciel forme adverbiale active accomplie : *après que j'ai choisi / quand j'aurai choisi*, etc.

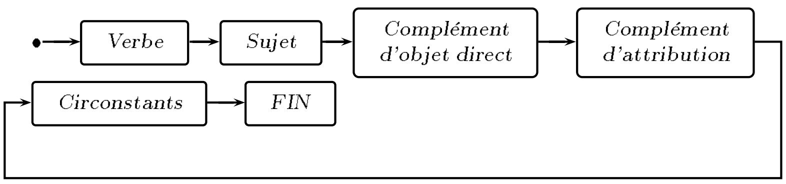
le pseudo-participe rapproché par certains du parfait sémitique, cette forme est à valeur passive et résultative : *stp=kw ayant été choisi.*

Formes diverses du verbe

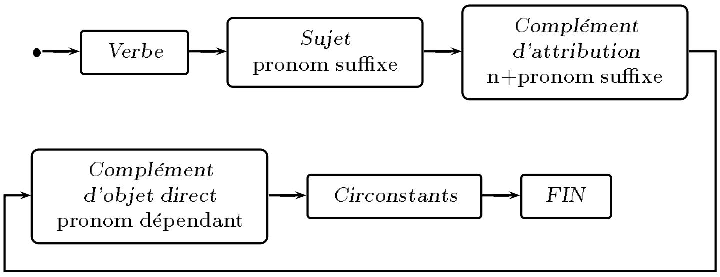
La classification que nous venons d'énumérer permet de couvrir bon nombre d'emplois du verbe, à tel point que celui-ci peut à la limite être considéré comme n'ayant aucune forme propre. Cependant, l'impératif ne se laisse pas facilement insérer dans l'une des catégories précédentes.

2.4.7 Construction des syntagmes utilisant le verbe

La syntaxe égyptienne est généralement centrifuge. L'ordre des mots autour du verbe conjugué est le suivant :



Il est perturbé par les pronoms ; si l'un des trois actants (au sens où l'entendait Lucien Tesnière (Tesnière 1966) : le sujet, le complément d'objet direct, et le complément d'objet indirect, appelés respectivement premier, second, et troisième actants) est pronominal, il passe *avant* les noms ; l'ordre dans lequel les actans pronominaux se trouvent est le suivant :



Exemple 8 (p. 0, l. 0.)

ʿḥʿ.n ʿw.n n=f sꜣ-nswt ḥr-dd=f ʿwy=fy

Alors ouvrit à-lui (le) prince Hardedef ses deux bras
Alors le Prince Hardedef lui ouvrit les bras.

Exemple 9 (p. 0, l. 0.)

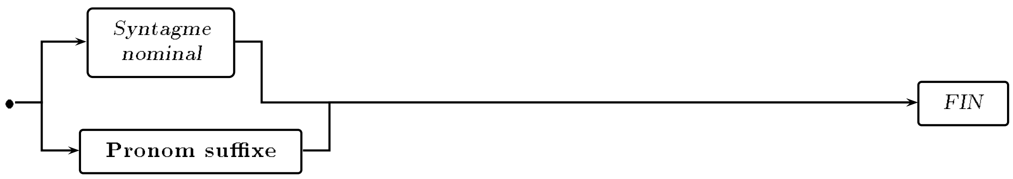
i͗nt=k n=i͗ sw

tu apporteras à-moi lui
tu me l'apporteras.

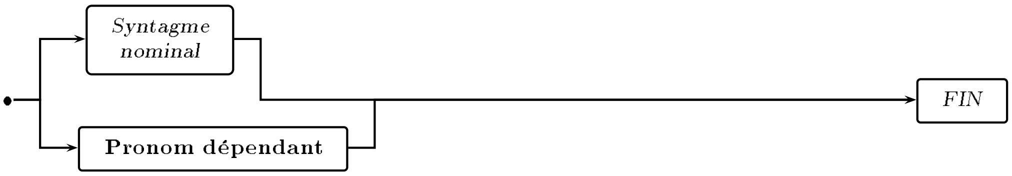
Quand le verbe est à l'infinitif, le complément d'objet direct, s'il est pronominal, est indiqué par un pronom suffixe, tandis que le sujet est introduit comme un complément d'agent par la particule , *i͗n ,par.*

On introduit, pour simplifier les notations par la suite, les catégories suivantes :

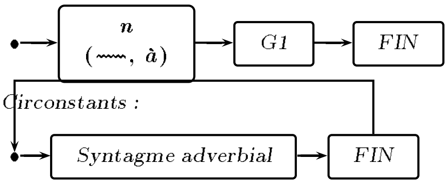
G1 :



G2 :



G3 :



Sachant donc que ces groupes se déplacent *mécaniquement* lorsque tout ou partie d'entre eux sont des suffixes, $G_aG_b \ldots G_n$ dénotera une suite de tels groupes, sans préciser explicitement leur ordre, un groupe optionnel étant noté entre crochets :

$\boxed{G1G2[G3]}$

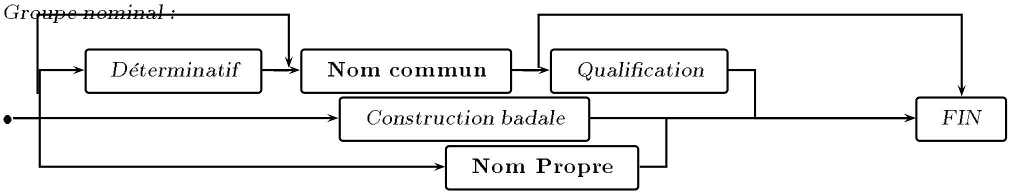
désigne ainsi l'un des groupes suivants :

- *Syntagme Nominal + Syntagme Nominal + n + Syntagme Nominal*
- *n + pronom Suffixe + Syntagme Nominal + Syntagme Nominal*
- *n + pronom Suffixe + pronom dépendant + Syntagme Nominal*
- *pronom dépendant + Syntagme Nominal + n + Syntagme Nominal*
- *pronom Suffixe + Syntagme Nominal + n + Syntagme Nominal*
- *pronom Suffixe + n + pronom Suffixe + Syntagme Nominal*
- *pronom Suffixe + n + pronom Suffixe + pronom dépendant*
- *pronom Suffixe + pronom dépendant + n + Syntagme Nominal*
- *Syntagme Nominal + Syntagme Nominal*
- *pronom dépendant + Syntagme Nominal*
- *pronom Suffixe + Syntagme Nominal*
- *pronom Suffixe + pronom dépendant*

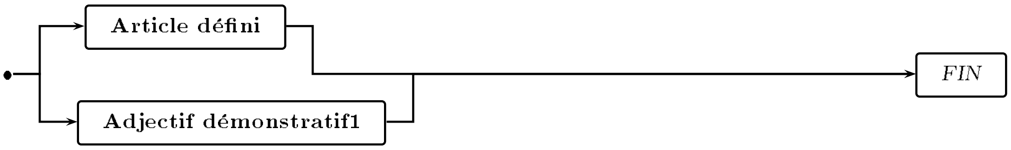
2.4.8 Le groupe nominal

La syntaxe des groupes nominaux est :

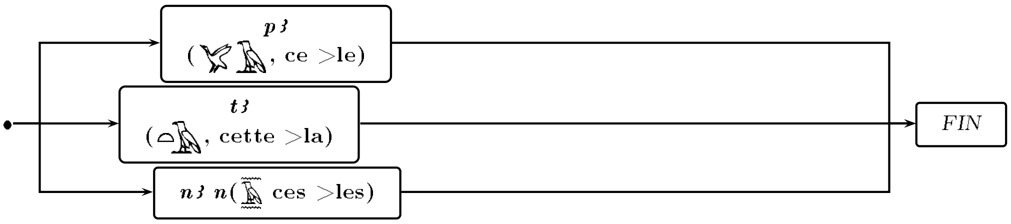
Groupe nominal :



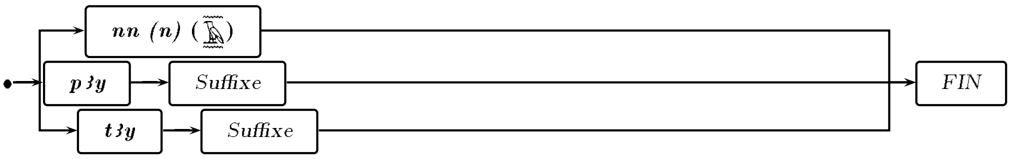
Déterminatif :



Article défini[6] *:*

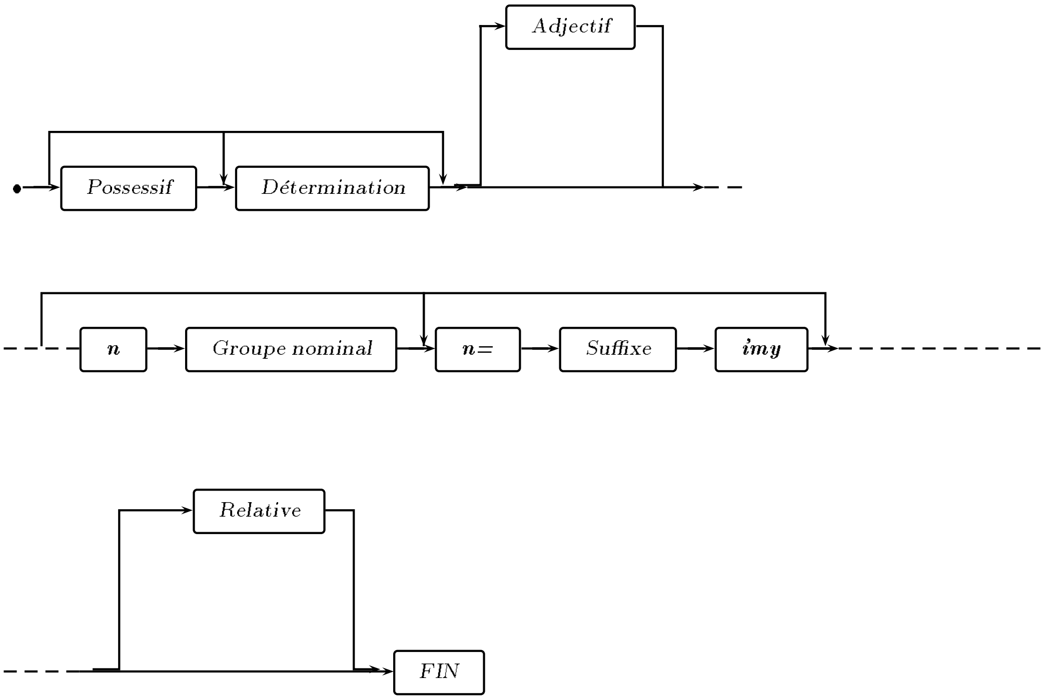


Adjectif démonstratif1 :

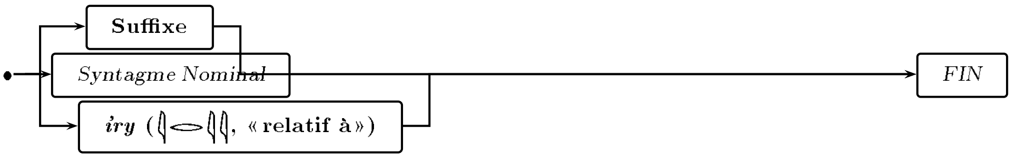


6. Nous employons ici le mot article par commodité. Dans notre texte, les mots *pꜣ*, *tꜣ* et *nꜣ n* sont à mi-chemin entre l'adjectif démonstratif et l'article

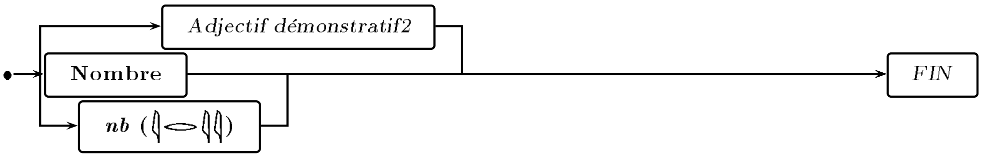
Qualification :



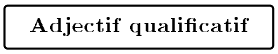
Possessif :



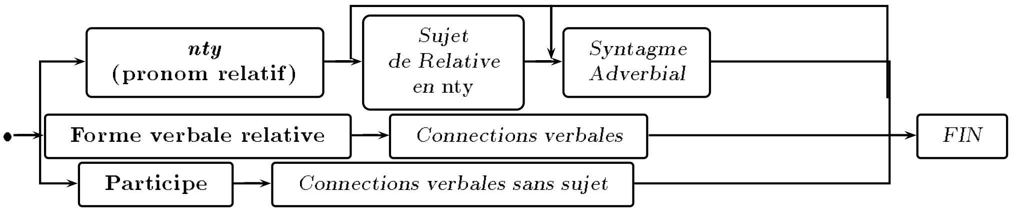
Détermination :



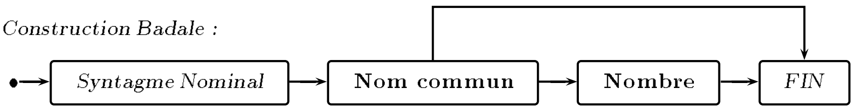
Adjectif :



Relative :



Construction Badale :



Quelques remarques :

la construction badale consiste à accoler deux noms, le premier désignant un tout, le second la partie dont ce tout est extrait ; ainsi, on dira *bière, une cruche* et non *une cruche de bière.* C'est aussi vrai pour des termes géographiques : *Nome Thinite, Abydos,* et non *Abydos, dans le nome Thinite.*

Cette grammaire est trop lâche et ne tient pas compte de toutes les contraintes. Par exemple, il est rare (mais pas forcément impossible) d'adjoindre un adjectif qualificatif à un génitif direct.

Les constructions commençant par un article défini et celles contenant le suffixe sont mutuellement exclusives. Au lieu de cela, on utilise l'adjectif démonstratif suivi du suffixe. De même, *nb,* « chaque, tout », ne peut se trouver utilisé en même temps que l'article.

2.4.9 Séquence et subordination de groupes nominaux

Deux groupes nominaux juxtaposés peuvent être en apposition, désignant la même entité ; coordonnés, désignant plusieurs entités mises sur un même plan ; ou enfin former un génitif direct, le second étant le complément de nom du premier.

Apposition Deux substantifs accolés, désignant la même réalité ; ainsi, *st-ḥmt,* (mot composé) littéralement «*une femme-épouse* », en fait utilisé alors pour «*femme* », ou *sꜣ-nswt ḥrddf,* le fils-royal Hardedef ; on constate que le groupe de ces constructions peut être cerné de près ; il s'agit principalement de véritables mots composés ou de l'énumération de titres. On peut envisager, même en dépassant le cadre du corpus, une description pratiquement exhaustive de cette dernière catégorie.

Coordination La coordination est un problème syntaxique majeur ; on pourrait presque considérer qu'il est d'un autre ordre que les problèmes d'actants ou d'apposition. Soit :

Exemple 10 (p. 12, l. 1-2.)

ḥsy šmʿ ẖbt wꜣg ỉrr.wt nb.t n

louanges chants danses cris (?) ce qui est fait tout pour

nswt

un roi

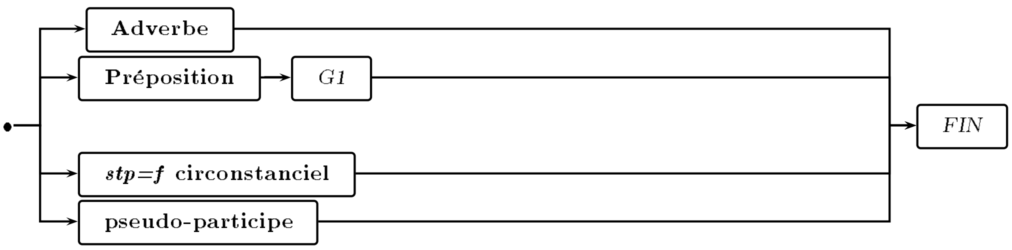
louanges, chants, danses, cris, tout ce qui est fait pour un roi

on voit ici que la coordination est complexe ; en particulier, notre corpus recèle plusieurs énumérations comme celle-ci, assez longues, formées d'une série de groupes nominaux de même structure (ici, un nom), et terminées par un groupe plus étendu, qui résume généralement l'ensemble. On peut se demander si la description que nous venons de donner de ces groupes ressort à la syntaxe ; il est parfaitement possible d'envisager de décrire peu ou prou un certain nombre de groupes de cette forme dans la grammaire. Cependant, il est théoriquement impossible de représenter par des grammaires *hors contexte* une construction comme *étant une suite de syntagmes composés de manière similaire.* On constate d'ailleurs que cette description est d'un degré d'abstraction supérieur.

Génitif direct le génitif direct est déjà traité par la grammaire de 2.4.8 ;

2.4.10 Les syntagmes adverbiaux

Syntagme adverbial :



2.4.11 Syntaxe du récit

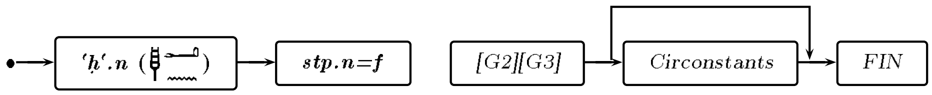
Le récit comporte un certain nombre de constructions spécifiques, qui marquent le caractère narratif du texte. Leurs valeurs respectives ne sont pas encore totalement définies. Selon l'usage, les constructions sont nommées selon leur forme.

La construction *ʿḥʿ.n* marque les grandes séparations du récit ; la construction *stp.ỉn* marque des épisodes moins tranchés. Cela dit, ce sont plutôt des tendances que des règles absolues. La construction *stp pw ỉr.n=f* est, quand à elle, utilisée en lieu et place de la construction *ʿḥʿ.n=f ḥr stp*, pour les verbes intransitifs de mouvements.

La construction *ʿḥʿ.n*

ʿḥʿ.n est une forme autonome du récit ; elle sert à marquer une nouvelle circonstance et peut être suivie par une proposition à valeur adverbiale. Dans Westcar, toutes les formes existantes sont suivies de formes à valeur accomplie.

Voici les formes présentes dans Westcar :

- 

  Où *ʿḥʿ.n* est l'auxiliaire, *stp.n=f* est le verbe "choisir" à l'accompli (*.n*), troisième personne masculin singulier (*=f*). D'où la traduction : « *Alors il choisit ...* » On trouve toutes les variations avec des pronoms énoncées en 2.4.7.

  Exemple 11 (p. 0, l. 0.)

  ʿḥʿ.n ṯꜣ.n=f sw
  Alors il-prit lui
  Alors il le prit

  Exemple 12 (p. 0, l. 0.)

  ʿḥʿ.n ms.n sy msḫnt r=f
  Alors présenta elle Meskhenet vers-lui
  Alors Meskhenet la lui présenta

- *ʿḥʿ.n + Sujet Nominal + stp.n=f + Circonstants* Dans quelques cas, le sujet nominal peut se trouver avant le verbe ; il est alors rappelé par un pronom :

Exemple 13 *WESTCAR* 0

ꜥḥꜥ.n rwḏḏdt wꜥb.n=s m wꜥb n hrw 14

Alors Roudidit elle-fut-pure de (la) purification de jour 14

Alors Roudidit fut pure de la purification des quatorze jours[7]

- *ꜥḥꜥ.n + sḏmw N*
- *ꜥḥꜥ.n + X + pseudo-participe + circonstants*

Ces deux dernières constructions ont à peu près le même emploi et la même valeur. Les formes verbales ont, pour les verbes *transitifs,* une valeur passive accomplie, et, pour les verbes *de mouvement intransitifs, ou d'état,* une valeur accomplie.

Exemple 14 (p. 0, l. 0.)

ꜥḥꜥ.n wꜣḥ pꜣ ḳniw
Alors fut-posé le palanquin
Alors le palanquin fut posé

Exemple 15 (p. 0, l. 0.)

ꜥḥꜥ.n pꜣ smn ꜥḥꜥ ḥr gꜣgꜣ
Alors l'oie étant-debout à caqueter
Alors l'oie se trouva sur pieds à caqueter[a].

[a] La malheureuse avait eu la tête coupée. Elle vient d'être recollée.

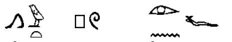
On trouve un unique cas de construction où le verbe suivant *ꜥḥꜥ.n* n'est pas à l'accompli ; c'est probablement une erreur.

La construction *stp pw ir.n=f*

Cette construction pallie l'absence totale pour les verbes de mouvement de forme *ꜥḥꜥ.n* non accomplie. Elle se construit en mettant l'infinitif du verbe comme attribut du sujet, qui est la forme verbale relative du verbe faire, *ir.n=f,* et signifie « qu'il fit ».

7. après l'accouchement

Exemple 16 (p. 0, l. 0.)



ỉwt pw ỉr.n=f

venir ça ce qu'il fit
« *C'est un venir qu'il fit* » ou, en français correct *Alors, il vint.*

La construction *wn.ỉn*

La construction est utilisée pour marquer que la phrase suit le cours du reste du discours. Sa forme est simple :

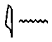
wn.ỉn + Sujet + Syntagme adverbial

où le *Sujet* peut être un syntagme nominal, ou un pronom suffixe.

Le syntagme adverbial ne peut être n'importe quoi : c'est, soit un pseudo-participe, soit une préposition suivie d'un nom (cela ne se produit qu'une fois dans notre corpus), soit la préposition *ḥr* suivie de l'infinitif, avec un sens inchoatif[8].

On traduis conventionellement « *wn.ỉn + X + ḥr + stp* » par « *Puis X commença à choisir* », « *wn.ỉn X spt(=w)* » par « *Puis X fut choisi* », et, dans le cas des verbes intransitifs, « *wn.ỉn X aHa* » par «*Puis X se trouva levé* ».

La construction *stp.ỉn=f*

C'est une forme verbale narrative, composée d'un verbe suivi de  La valeur de cette construction est discutée. Certains pensent qu'elle se distingue essentiellement par des critères sociaux (elle aurait pour sujet uniquement des êtres « respectables »), d'autres lui donnent simplement une valeur de séparateur moins forte que celle de *ʿḥʿ.n*.

Autres constructions

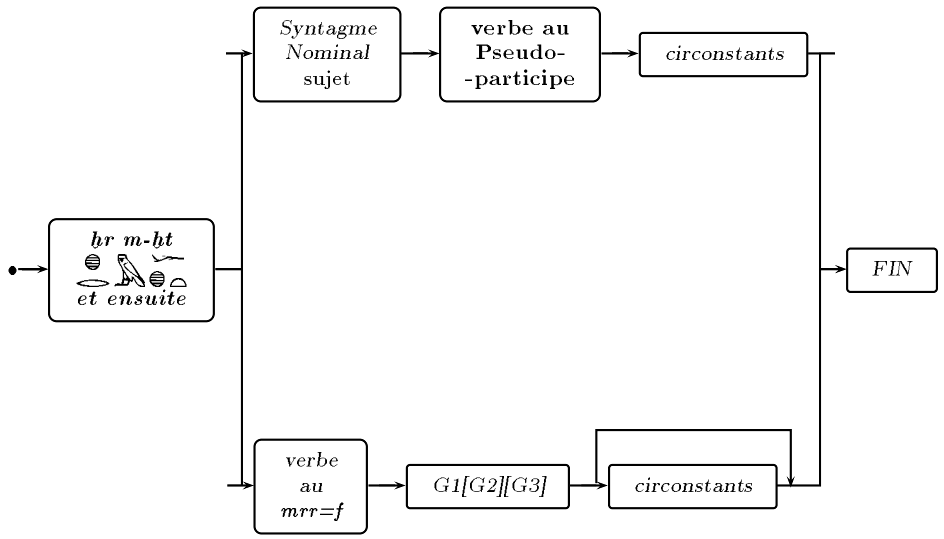
Un certain nombre de constructions sont aussi observables :

8. inchoatif signifie « qui marque le début, le commencement ». On traduit donc par *se mettre à* ou *commencer à*

Protases

L'égyptien met plutôt les circonstants en fin de phrase. Mais, s'il les marque, il peut, en particulier lorsque ceux-ci sont des complément de temps, les mettre en début de phrase *(en protase)*. Dans Westcar, de tels circonstants sont introduit par *ḫr m-ḫt,* et après que... La construction est :

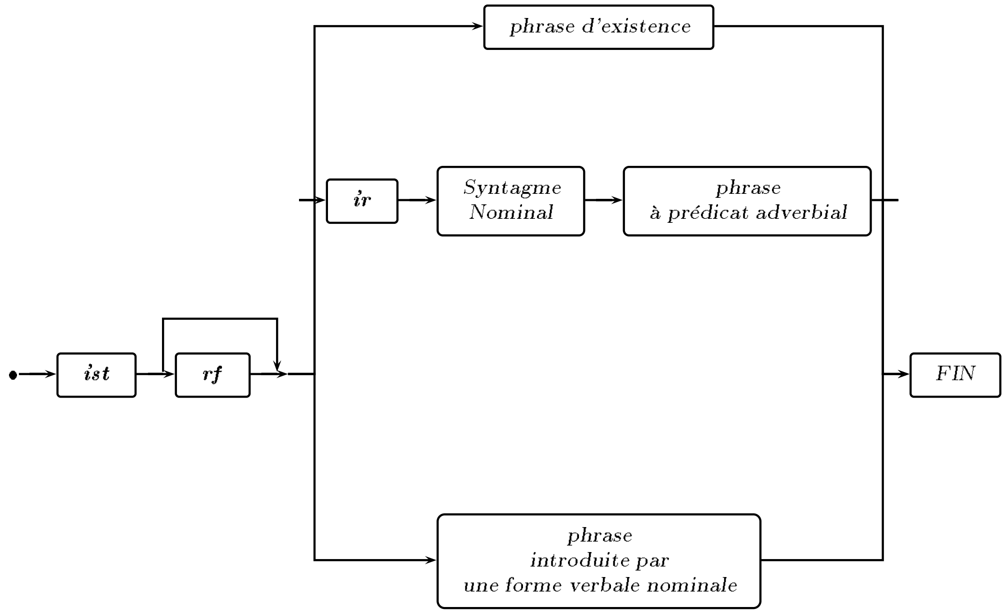
Protase :



Incises

La copule *ỉsṯ* sert, dans Westcar, à introduire des incises dans le cours du récit.

incise :



Forme emphatique

La seule construction autonome que l'on trouve dans le récit de Westcar qui ne soit pas une forme du recit est la phrase dite « *emphatique* », où le verbe est nominalisé, si bien que le syntagme dépendant du verbe devient le sujet d'une phrase à prédicat adverbial, dont le prédicat est ainsi mis en valeur. Comparer :

$$\underbrace{\text{moi}}_{\text{Sujet}}\ \underbrace{\text{dans l'état de manger dans la cuisine}}_{\text{Prédicat}}$$

et

$$\underbrace{\text{Le fait que je mange}}_{\text{Sujet}}\ \underbrace{\text{dans la cuisine}}_{\text{Prédicat}}$$

Soit : *je mange dans la cuisine,* et *c'est dans la cuisine que je mange.*

stp.n nominal + Sujet + [COD] + [CA] + Circonstants ou
stp.n nominal + Sujet + [COD] + CA + [Circonstants]
Notez bien que la présence d'un élément adverbial est *nécessaire.*

Exemple 17 (p. 0, l. 0.)

wrš.n=s ỉm ḥr swỉr ḥnʿ pꜣ nḏs
C'est à boire avec le jeune homme, qu'elle passa la journée là.

Dans Westcar, certains verbes s'utilisent uniquement avec cette forme : les verbes *gm, trr,* et *wrš, passer le jour ,* parce qu'ils ont essentiellement pour fonction d'introduire, respectivement, « l'état dans lequel on trouve l'objet », et à quoi « l'on passe le jour ». L'on trouve cependant deux occurrences d'autres verbes, mais elles sont suspectes.

Tournures négatives

Les tournures du récit se nient comme les tournures du discours, en utilisant des constructions simples :

𓂜 suivi d'un verbe au *stp.n=f* sert à nier la possibilité ou l'habitude :

Exemple 18 (p. 0, l. 0.)

n	*gm.n=f*	*sy*
pas	*il pouvais trouver*	*cela?*

Il ne pouvais pas en trouver

Exemple 19 (p. 0, l. 0.)

n	*rḫ.n=tw*	*mꜢꜥt*	*r*	*grg*
pas	*on peut distinguer*	*la vérité*	*contre*	*le mensonge*

On ne peut distinguer le vrai du faux.

𓂜 suivi d'un verbe au *stp=f* sert à nier l'accomplissement ;

Exemple 20 (p. 0, l. 0.)

n	*ỉn=tw*
pas	*on a apporté*

On n('en) a pas apporté

On remarquera que la majeure partie des phrases d'exemples ne sont pas narratives.

l'Auxiliaire *pr.n*

C'est un auxiliaire rare qui fonctionne comme *ꜥḥꜥ.n ;* il sert à indiquer les conséquences d'un récit, et il se pourrait traduire par « *en fin de compte.* »

2.4.12 Syntaxe en dialogue

La grammaire des dialogues est bien plus complexe que celle du récit ; les structures utilisables comme énoncés sont en effet plus nombreuses. On peut, en particulier rencontrer des syntagmes nominaux ou adverbiaux seuls, avec alors une valeur exclamative. Cependant ces irrégularités se rencontrent principalement en début de réplique, soit qu'elles aient pour but d'attirer l'attention d'un interlocuteur, soit qu'elles prennent la suite de ce que celui-ci dit. De plus, l'urgence et l'émotion conduisent à des énoncés plus courts ; ainsi, la phrase suivante se résume-t-elle à un syntagme adverbial :

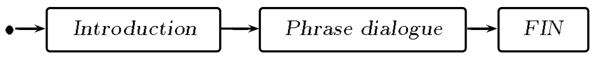
Exemple 21 (p. 0, l. 0.)

n ỉs	*n*	*rmṯ*	*ỉty*	*ʿ.w.s.*	*nb=ỉ*
pas	*à*	*(un) homme*	*souverain*	*vie santé force*	*seigneur mien*

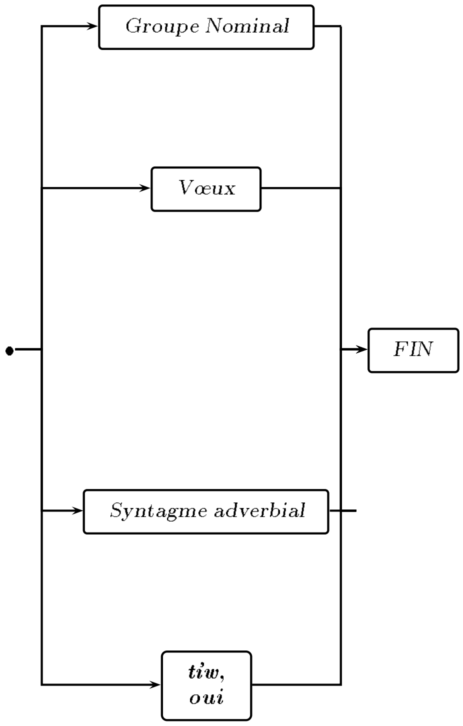
Pas à un homme, ô souverain, vie, santé, force, mon maître !

L'analyse des dialogues nous donne donc la grammaire suivante :

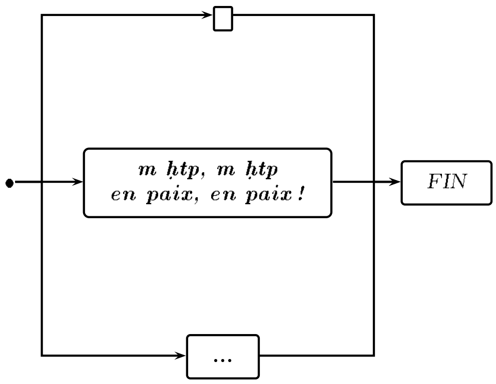
Réplique :



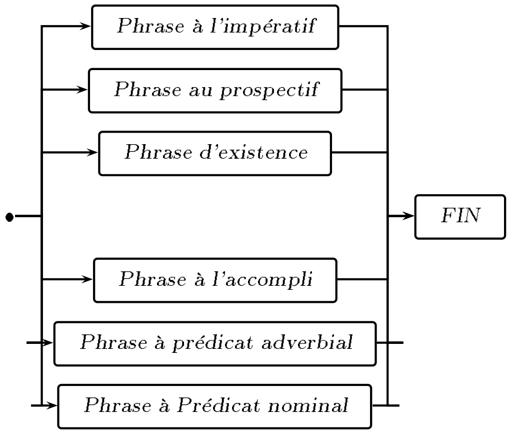
Introduction :



Vœux :

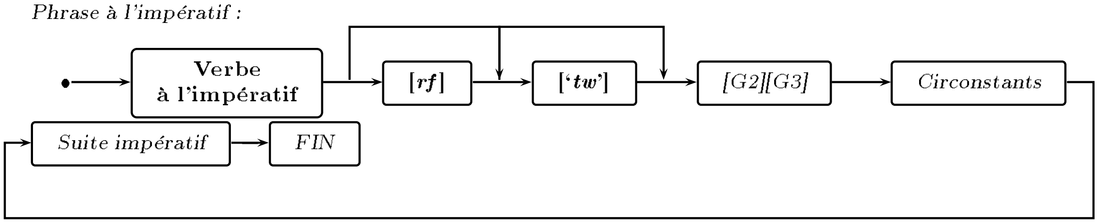


Phrase de dialogue :

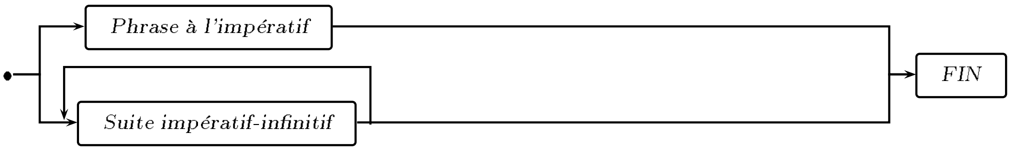


phrases possibles :

Phrase à l'impératif :



Suite impératif :



2.4.13 Structure des dialogues

Les dialogues dans *Westcar* sont courts, souvent composés d'une seule réplique. On peut cependant leur trouver une structure globale. Celle-ci est fortement marquée par des considérations sociales. Un dialogue où l'un des interlocuteurs est un roi laisse peu de place à l'echange. Un autre facteur important dans la structure des dialogues est le caractère même du texte. C'est un conte, d'un style presque oral ; il ne rechigne donc pas à la répétition. Cependant, les dialogues sont trop complexes et trop divers pour qu'apparaissent de grandes tendances. Les seules conclusions que nous avons pu tirer, c'est l'emploi de formes polies pour s'adresser au roi, et l'emploi fréquent, par celui-ci, de l'impératif.

2.4.14 Macro-syntaxe

La structure globale du texte est assez complexe. Si l'on dépasse le niveau de la proposition, ou de la phrase, on trouve des constructions grammaticales qui ne peuvent s'analyser qu'en considérant un volume de texte non négligeable.

On pense ici, en premier lieu, aux dialogues imbriqués. À vrai dire, ceux-ci participent de la syntaxe de la phrase, puisqu'un dialogue est généralement le complément d'objet du verbe *dire*. De plus, les implications grammaticales de cette structure sont d'assez haut niveau dans la chaîne d'analyse ; elles concernent essentiellement les référents des pronoms.

D'autres constructions portent sur de longs passages du texte. En particulier, le choix des verbes narratifs est un phénomène assez nettement grammatical dans le texte. Un passage le montre bien : *la naissance des trois fils de Rê.* Chaque naissance est décrite par un récit, le même pour chaque naissance, avec d'infimes variantes. Parmi celles-ci, la forme des verbes. Les verbes qui, dans la première naissance, utilisent la forme *ʿḥʿ.n stp.n=f,* utilisent dans les récits suivants la forme *stp.ỉn=f.* Chaque épisode fait 113 mots. Néanmoins, ce phénomène peut être réglé dans un second temps, car il ne perturbe pas l'analyse locale du texte.

De fait, ces problèmes, à mi-chemin entre l'analyse du récit et la syntaxe, peuvent être laissés de côté sans grand dommage.

2.4.15 Critique de cette grammaire

Nous avons déjà insisté sur les insuffisances de la grammaire formelle que nous énoncions. Elle a d'abord le défaut d'être trop large. Prenons quelques exemples. Nous avons permis, en précisant qu'une forme verbale nominale nominalisait toute l'expression qui dépend d'elle, à tout verbe d'avoir comme complément d'objet direct une phrase au prospectif. C'est assez absurde, comme par exemple dans le cas de **ỉw=ỉ ḥr wnm mȝȝ=f, je mange qu'il voie.* Non seulement la phrase n'a pas de sens, mais encore notre verbe *wnm,manger,* n'aura jamais de tel complément d'objet.

Néanmoins, l'on peut se demander si ce type de phénomène ne doit pas faire l'objet d'un autre niveau de description.

2.4.16 Conclusion

Nous allons préciser dans les chapitres suivants (en particulier les chapitres 4 et 5) la manière dont nous formalisons cette analyse du texte. Cette informatisation montrera qu'il existe un hiatus entre la manière dont nous venons de présenter les données, et les problèmes qui se posent effectivement pour l'automatisation, ce qui entraînera une remise en perspective de notre approche.

Chapitre 3

Segmentation et lexique

Sommaire

Les problèmes de segmentation lexicale sont beaucoup plus généraux que ne pourrait laisser penser un examen superficiel. Il est clair qu'il se posent dans les langues qui ne marquent pas typographiquement les fins de mots, par exemple le chinois, pour lequel existent un certain nombre d'études (Sproat, Shih, Gale, et Chang 1994). Le problème se pose aussi, quoi qu'en termes légèrement différents, lors de l'analyse du langage parlé.

Mais d'une certaine manière, le problème est beaucoup plus large. En fait, la segmentation en mots est une donnée relativement arbitraire, dont linguistes discutent la pertinence ; A. Martinet (Martinet 1985, p. 70–85) préfère parler de *monèmes* et de *synthèmes* plutôt que de mots, ce dernier terme recouvrant selon lui plusieurs réalités distinctes[1].

1. Sur le problème des unités linguistiquement pertinentes, voir aussi Benvéniste (Émile Benvéniste 1974, 1, pp. 119–131)

Du point de vue du traitement automatique de la langue, la question est différente. Il s'agit moins, dans un premier temps, de déterminer les *unités significatives de la langue,* problème éminemment linguistique, mais bien plutôt de découper le texte en unités traitables, c'est-à-dire auquelles on puisse associer :

- une extension ;
- une série de propriétés utiles pour le traitement.

Ce dernier point est sans doute proche de la problématique linguistique, mais les propriétés à extraire ne sont pas forcément motivées par une théorie digne de ce nom. Les systèmes à base de mots-clefs, parmi lesquels on peut distinguer Elisa (Weizenbaum 1966), ont une vision assez brutale de la segmentation.

Les problèmes à résoudre sont : le découpage du texte en mots, et la reconnaissance de ceux-ci. Ces deux fonctionnalités ne sont pas simplement utiles en vue de l'analyse syntaxique ; elles sont utiles par elles-mêmes, car elles permettent d'utiliser des textes saisis en hiérolyphes comme base de données. En effet, le mode le plus naturel pour chercher un mot égyptien est de passer par la translittération. Elle ne permet pas à elle seule une recherche non ambiguë, mais elle réduit considérablement l'espace de recherche. Nous avons donc conçu notre logiciel de gestion de bases de données hiéroglyphiques en conséquence.

Nous allons maintenant décrire plus avant la segmentation, la reconnaissance des mots, puis le système de structuration et de gestions de la base de données lexicale que nous avons programmée.

3.1 Découpage en mots

Le découpage en mot est nécessaire pour la plupart des opérations automatiques menées ultérieurement, y compris la simple indexation des termes.

Il est à première vue envisageable de découper le texte en utilisant directement un lexique. Cependant, le découpage obtenu sera potentiellement ambigu, certaines ambiguïtés étant levées par la suite, et certaines subsistant. Considérons à titre d'exemple un texte en français, sans espace :

ledécoupageenmotsestnécessaire

On s'aperçoit malheureusement que le découpage suivant est lexicalement valide :

le dé cou page en mots est né cessa ire

Encore n'a-t-on pas ici considéré que toute lettre est potentiellement un mot ! On remarque en passant que dans le cas présent, si l'on introduit une relation d'ordre sur les segmentations, telles qu'une segmentation s_1 est supérieure à une segmentation s_2 ssi toutes les coupures de s_1 sont dans s_2, la bonne segmentation de cette phrase est maximale pour ce critère.

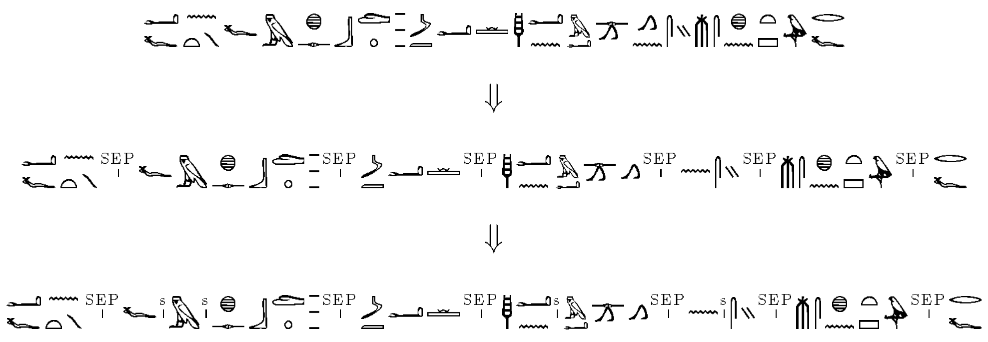
En fait, la situation est un peu meilleure en égyptien qu'elle ne le serait en français, parce que le système d'écriture n'utilise certains signes qu'en fin de mot.

De toutes manières, le système évoqué suppose un lexique contenant les mots du texte ; si le but est partiellement de construire ce lexique, le système échoue. De plus, l'orthographe de l'égyptien a varié avec le temps et parfois avec les scribes ; un système utilisant un lexique serait dans ce cas très limité.

On peut, en disposant d'un corpus de bonne taille, dresser la liste des séquences de 2 signes observables dans un mot. Une séquence absente de la liste est probablement une fin de mot ; ainsi dans la figure 3.1. Pour ne pas introduire de fins de mots abusives, il faudra travailler sur une liste comportant des formes fléchies, en particulier les pluriels.

FIG. **3.1** – Découpage en mots

Westcar, 10,26



Ce système fournit des frontières de mots à peu près sûres (si le corpus de départ est assez étendu). Il est éventuellement possible d'utiliser des transducteurs[2] pour décrire des fins de mots. Cette dernière méthode est probablement plus raisonnable dans un premier temps ; les deux méthodes sont de toutes façons complémentaires. On doit obtenir un texte suffisament découpé pour ne plus risquer d'explosion combinatoire.

Il faut ensuite traiter le texte ainsi coupé, en prêtant une attention particulière à la recherche des mots fréquents, qui ne se terminent pas forcément

2. Un transducteur fini est un automate fini, à chaque arc duquel on a associé des productions.

par un déterminatif ; par exemple, la seconde séquence de la figure 3.1, , commence par deux mots-outils, le pronom et la préposition . Comme les mots de ce type sont en nombre limité, il est concevable d'en dresser une liste quasi exhaustive.

Les méthodes statistiques, en particulier celles qui ont été utilisées, avec succès, en lemmatisation automatique, constituent une autre méthode d'approche. Typiquement, dans ce type de méthode (Merialdo 1995), on cherche à associer une étiquette à chaque élément d'une liste. On calcule, sur des fenêtres réduites (classiquement trois éléments), les probabilités qu'ont les étiquettes de se succéder. Il s'agit ensuite de trouver l'étiquetage global de probabilité maximale.

Le principe que nous utiliserons est le suivant : la transition entre deux signes successifs est dotée de trois valeurs possibles. Elle peut être interne à un mot, séparer deux mots, ou séparer un mot d'un suffixe (au sens large du terme).

Ensuite, sur un corpus déjà renseigné, l'on calcule les probabilités qu'un signe soit placé dans un environnement donné, c'est-à-dire : $P(-S-)$, $P(=S-)$, $P(/S-)$, $P(-S=)$, $P(=S=)$, $P(/S=)$, $P(-S/)$, $P(=S/)$, $P(/S/)$ où S représente le signe, $-$ représente la liaison entre deux signes dans un même mot, / la liaison entre deux mots, et = la séparation entre un mot et une terminaison grammaticale.

Ainsi, notre texte de la figure 3.1, complètement traité, donne-t-il :

il serait souhaitable de calculer la probabilité pour qu'une liaison donnée soit entre deux signes, pour tout couple de signes, c'est à dire les probabilités $P(S_1 - S_2)$, $P(S_1{=}S_2)$ et $P(S_1/S_2)$. En effet, la succession des signes est une information loin d'être négligeable pour le calcul des frontières de mots. Cependant, ces probabilités sont difficiles à estimer. La valeur cardinal$(S_1 - S_2)$ peut être obtenue dans un lexique, puisque $S_1 - S_2$ est interne aux mots. La valeur cardinal$(S_1{=}S_2)$ dépend de la composition des mots et de celle des suffixes possibles, qui sont en nombre réduits ; En revanche, cardinal(S_1/S_2) ne peut être évalué de manière correcte que grâce à un très grand corpus.

On peut en obtenir artificiellement un tel corpus, pas forcément très représentatif, à partir d'un lexique. Il suffit de considérer comme corpus un ensemble de « phrases » constituées des couples de mots du lexique. On peut si besoin est pondérer chaque mot par sa fréquence d'apparition dans un véritable corpus. On obtient alors une représentation passable des successions possibles de signes.

Un corpus ainsi traité nous permet d'estimer les probabilités liées à un signe donné. Le court exemple que nous donnons fournit ainsi des renseignements sur le comportement du signe : on trouve deux fois = / et une fois --.

Disposant, pour chaque signe, des probabilités citées plus haut, on pourra, face à un texte inconnu, calculer le découpage le plus probable. Pour limiter la taille des calculs, on peut utiliser le pré-traitement évoqué au début.

3.2 Translittération

La méthode employée pour translittérer les mots est la suivante : nous définissons des règles de simplification, qui décrivent la façon dont les signes se combinent : ainsi, la règle :

```
20 regle     % 20 : priorité
   L1 *** A/[X] *** C/[X,Y] *** B/[Y] *** L2
==>
   L1 *** A/[(X,1)] *** C/[(X,1),(Y,2)] *** B/[(Y,2)] *** L2.
```

signifie qu'un bilitère (signe écrivant 2 consonnes, ici représentées par X et Y) entouré de deux unilitaires (signes écrivant une consonne) correspondant à sa lecture doit se lire seul (par exemple :). La règle a une priorité, qui permet d'ordonner l'application des simplifications. Ainsi, les simplifications les plus complexes et les plus contraintes seront-elles effectuées avant les simplifications plus simples. En revenant sur les simplifications, l'on peut énumérer tout ou partie des lectures possibles d'un mot. Notons que lorsque celui-ci présente des anomalies, par exemple une métathèse[3] graphique, nous avons choisi pour l'instant de traiter le problème en doublant l'entrée lexicale.

3.3 Recherche dans le lexique

Considérons un mot, par exemple . Une fois les algorithmes de translittération appliqués, nous aurons plusieurs valeurs possibles : *wʿrt, wʿrtỉnr...* fournies dans cet ordre par notre programme. Celui-ci nous laissera aussi les déterminatifs, qui seront distingués par des traits : ainsi, possèdera-t-il le trait « terrain ». Les traits peuvent être plus ou moins précis,

3. Une métathèse est l'inversion de deux consonnes. Elle peut être le fruit d'une erreur, d'une évolution phonétique, ou, dans le cas de la *métathèse graphique*, être simplement dûe au fait que les signes peuvent se disposer d'une manière plus harmonieuse que celle qu'exigerait la simple succession phonétique, un peut comme si nous écrivions *ATALANTIQUE* pour *ATLANTIQUE,* afin d'éviter la configuration *TL,* jugée disgracieuse.

et certains en impliquent d'autres : par exemple, les idéogrammes de dieux particuliers impliquent le trait «dieu». Le trait générique«oiseau» est partagé par les signes ...

Ces traits nous permettront donc de choisir entre les mots , «plateau désertique», et , «jambe». Un tel système oblige d'ailleurs à structurer plus fortement son lexique qu'il n'est d'usage. Le mot , «district», se place ainsi naturellement sous la même entrée que «plateau désertique».

Quand les déterminatifs ne suffisent pas à distinguer les homonymes, il est possible de recourir à la comparaison des trilitères puis des bilitères qui forment le mot.

Enfin, les abréviations et autres formes irrégulières, en particulier celles que Gardiner (o.c.) qualifiait de «sportives», se trouvent à la place où l'algorithme de translittération les range, ce qui permet de renvoyer commodément à leur vraie valeur. Ainsi, un rébus véritable comme , constitué du signe de l'eau *mw* sous (*ẖr*) le signe du vase *nw*, qui se lit donc littéralement *m(w) ẖ(r) nw* et sert parfois à écrire *m-Xnw*, à l'intérieur, sera enregistré à *nw,* mais renverra à *m-ẖnw.* C'est d'ailleurs la méthode employée par les dictionnaires lorsqu'ils veulent permettre à un débutant de trouver la prononciation d'un tel mot dont il ignorerait la lecture.

3.4 Le problème de la reconnaissance des mots

Dans certaines applications, il s'agira, non de trouver la translittération d'un mot, mais, étant donné une translittération, de trouver des mots auquel elle correspond. Ce problème est beaucoup plus simple à résoudre que son dual. On peut obtenir des résultats passables en utilisant un algorithme tel que le suivant, qui fournit une «distance» entre un mot et une translittération.

- choisir une valeur pour chaque signe du mot, telle que le nombre de consonnes dans la translittération soit le plus petit possible, sans être nul. En particulier, le premier signe ne peut être un déterminatif.

- Construire un vecteur ; chaque coordonnée correspond à une lettre, la valeur de la coordonnée étant binaire selon qu'elle apparaît ou non;

- comparer ce vecteur et celui obtenu à partir de la translittération.

On peut pondérer certaines des coordonnées, voire en réunir quelques unes, pour représenter le fait que certains signes tendent à se confondre.

3.5 Un système hypertexte

Nous avons développé un système destiné à servir de support aux techniques décrites dans le présent document. Il s'agit à la fois d'un éditeur de textes hiéroglyphiques, d'un gestionnaire de bases de données en texte intégral, et d'un lexique. L'idée est de permettre de stocker l'intégralité du travail effectué sur un texte, afin de pouvoir facilement retrouver et réutiliser des parties de ce travail. Cette réutilisabilité inclut *a priori* la possibilité de partager les données avec d'autres chercheurs.

Nous allons évoquer brièvement les possibilités d'édition de texte, de représentation de celui-ci, puis décrire les problèmes liés au lexique.

3.5.1 Fonctions d'édition

L'état actuel de l'interface d'édition est décrit en annexe ; nous nous contentons ici d'en dégager les détails intéressants.

L'interface que nous avons réalisé s'inspire de celles qui existent, par exemple *MacScribe ;* elle a cependant quelques particularités. Les autres systèmes existants sont fortement influencés par leur usage premier, qui est l'impression de textes ; ils ont donc surtout développé des fonctionnalités permettant la reproduction la plus fidèle possible des textes originaux. Nous avons choisi, de notre côté, de considérer la réalisation de documents informatiques comme notre finalité première, et donc donné la priorité au confort des fonctions de saisie. Les textes que nous saisissons proviennent en majorité d'originaux hiératique[4], qui comportent peu de particularités graphiques intéressantes, et sont de toutes façons modifiés par leur transcription en hiéroglyphe.

La question de savoir si une base de données en texte intégral doit viser au facsimile informatique reste à trancher. La question de savoir si le texte *imprimé* doit lui-même constituer un facsimile est débattue, mais l'opinion commune est que seul le dessin permet un rendu satisfaisant. Certains auteurs[5], quand ils en ont les moyens, proposent simultanément une photographie, un facsimile au trait, et une édition en caractères facilement lisibles, qui rend le texte accessible aux non-épigraphistes[6]. L'édition informatique complète d'un texte devrait dans l'idéal comporter tout cela, et lier les diverses

4. l'hiératique était l'écriture utilisée dans la vie courante. Le principe était le même que pour les hiéroglyphes, mais les signes étaient simplifiés et schématisés

5. par exemple, Manuelian, dans *living in the past* (1994)

6. L'épigraphie est la science qui étudie les inscriptions ; par exemple la forme des caractères de l'alphabet latin a changé au fil du temps, et la lecture d'une inscription ancienne, quand bien même elle est dans une langue connue, demande un certain entraînement.

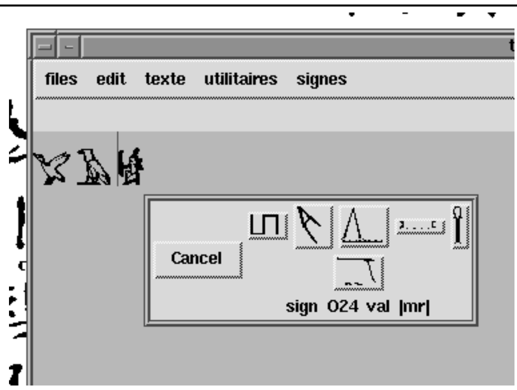
représentations. On imagine facilement de lier les signes du texte original à leur interprétation. Dans le cas d'un texte hiératique, cela constituerait une aide pour les débutants ; de plus, l'existence d'un objet informatique reliant le document d'origine à son interprétation est un support logique pour un *commentaire* sur l'interprétation.

Pour revenir à l'interface, les signes sont entrés, comme dans les programmes analogues, soit selon leur code typographique, soit selon leur translittération, soit enfin à l'aide de menus. La translittération est pour la plupart des signes la méthode la plus naturelle.

Nous avons surtout essayé de permettre la saisie la plus rapide possible. En particulier, les hiéroglyphes sont généralement groupés afin de remplir l'espace. L'unité de groupement, appelé quadra, est un carré. Les signes 𓊪, 𓏏 et 𓇯, qui écrivent le mot *pt, ciel,* sont disposés ainsi dans les textes : 𓊪𓏏𓇯. Le groupement peut être plus complexe ; mais nous avons permis, dans les cas simples, qui sont de loin les plus fréquents, que la saisie de la totalité du quadra soit rapide : une pression sur la touche « : » rajoute le signe courant sous le quadra courant ; une pression sur la touche « * » le rajoute à la fin de la dernière ligne de signes du quadra courant.

Nous avons aussi permis la création de menus de signes « à la volée » : étant donné une translittération, tapée par l'utilisateur, on lui propose un menu listant les signes qui peuvent y correspondre(figure 3.2).

Fig. **3.2** – Menus créés à la demande



3.5.2 Représentation interne

Le système est écrit en TCL/TK et en C. Le gros du travail est un *widget* TK capable d'afficher des hiéroglyphes. Il comprend des commandes telles que « rajouter tel texte à tel endroit », supprimer du texte à tel endroit », etc.

Le texte rajouté est actuellement représenté selon les standards du *manuel de codage* (Buurman, Grimal, Hainsworth, Hallof, et Plas 1988). La raison à cela est essentiellement qu'il s'agit d'un standard, et permet donc l'échange de textes. Néanmoins, il apparaît que le standard en question date d'il y a plus de dix ans, et a été développé essentiellement en vue de l'impression de textes. De plus, il s'agissait au départ d'une représentation qui était directement saisie par un opérateur humain, et qui se devait donc d'être concise. Il en résulte un certain nombre de problèmes ; en particulier, il est difficile d'étendre proprement le formalisme, qui utilise beaucoup de caractères. Il est, de plus, difficilement manipulable. Une méthode d'extension possible est celle utilisée par ADOBE pour POSTSCRIPT : détourner la possibilité d'inclure des commentaires dans le texte pour rajouter des commandes. L'intérêt est que le texte reste lisible par un interpréteur qui ne comprend pas les nouvelles extensions.

Nous allons cependant développer pour nos besoins propres un formalisme qui sera basé sur des structures de listes, faciles à traiter en TCL, et séparera la représentation de la disposition des signes de la liste des signes eux-mêmes, afin de simplifier les opérations de traitement sur le texte. Le problème principal est que la disposition des signes et le découpage en mots sont indépendants.

3.5.3 Structure du dictionnaire

La base de textes est associée à un certains nombre d'autres éléments, dont un dictionnaire. Ce dernier est réalisé à partir de la base ; c'est un produit de la création de celle-ci, qui va contenir les notes lexicographiques prises lors du travail sur les textes. Ce dictionnaire aura de plus la fonction de fournir des données pour l'analyse automatique.

Un lexique fourni constitue en général une base de donnée extrêmement riche ; on peut en extraire de multiples informations, qui dépassent largement la simple consultation du texte associé à une entrée.

L'utilisateur potentiel, qui peut très bien être un programme (qu'il calcule des statistiques ou qu'il tente d'extraire des structures), est susceptible de chercher des donnée à partir de critères extrêmement variés : catégories grammaticales, constructions syntaxiques admises, traductions, hiéroglyphes contenus dans les graphies, sources où le mots est attesté, etc.

La structuration des entrées du lexique par la syntaxe, la ponctuation, le choix des polices de caractères, et tout critère typographique en général, est à la fois insuffisante et peu cohérente.

Elle est insuffisante, parce que ces codes, élaborés pour des dictionnaires papier, tendent à être polysémiques ; d'autre part, les opérateurs de saisie ne font pas toujours preuve d'une constance absolue dans leurs pratiques. Ce qui est un inconvénient mineur pour un lecteur humain devient un problème difficilement surmontable lorsque l'on essaie d'automatiser des recherches.

Notre expérience dans la structuration *a posteriori* des sources dictionnairiques nous incite à utiliser une structure explicite, rendue possible par l'usage d'un outil de saisie adéquat.

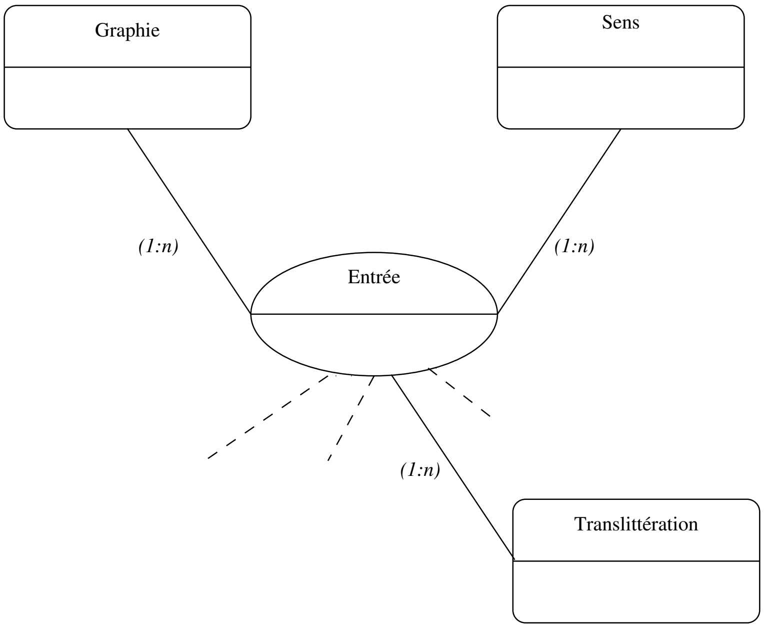
Cependant, un lexique comme celui-ci, construit petit à petit, par des utilisateurs éventuellement divers, aux besoins différents, se devra d'être souple. Une structure trop pauvre ou trop complexe pour les entrées serait gênante. Nous souhaitons disposer d'une structure qui permette d'écrire naturellement les entrées, sans demander trop de décisions au rédacteur.

Les principaux champs que l'on peut isoler sont :

- les graphies : ;
- les translittérations : *wꜣḥ*;
- les catégories grammaticales : *verbe transitif* ;
- les liens avec d'autres entrées : racine de *wꜣḥwt*;
- les indications de domaines, diachroniques ou synchroniques : *mathématique, médecine* ;
- les traductions : *poser* ;
- les exemples ;
- les constructions grammaticales : *wꜣḥ tp m* x *r sp* y ;
- des commentaires ;
- l'identification de l'auteur de la notice ;
- des références de textes : *Papyrus Rhind* ;
- des références d'études sur le mot.

Les constructions grammaticales seront plus utiles si elles sont suffisament formelles pour faire l'objet de traitements automatiques ; il faut donc définir une syntaxe pour les représenter. D'autre part, les traductions doivent permettre des recherches automatiques. Il est donc nécessaire de bien délimiter les termes qui forment chaque traduction, qu'il s'agisse d'expressions ou de mots seuls. La représentation du sens peut se faire comme dans la base de données WORDNET (Miller, Beckwith, Fellbaum, Gross, et Miller 1993), par des ensembles d'expressions presque synonymes.

Tous ces champs entretiennent entre eux des rapports complexes. Un mot ne peut être représenté ainsi :



Par exemple, une graphie donnée peut impliquer une traduction, mais la traduction en question peut être possible pour d'autres graphies du même mot. Inversement, un sens donné peut n'être attesté que pour certaines graphies, sans que celles-ci soient spécialisées dans la représentation de ce sens.

Nous avons donc décidé d'utiliser une représentation imbriquée du lexique ; le contexte détermine alors quelles sont les relations entre les différents champs. Ainsi,

```
<traduction>
        <graphie>
        </graphie>
</traduction>
```

représente la première possibilité (la graphie n'existe que pour la traduction donnée), et

```
<graphie>
        <traduction>
        <traduction>
</graphie>
```

représente la seconde.

Nous avons donc développé un modèle de dictionnaire (pour l'instant une description de document SGML) que l'on trouvera en annexe. La TEI propose un modèle pour la représentation de dictionnaires, mais nous ne l'avons pas utilisé ; en effet, il vise à représenter des dictionnaires *existant*, alors que notre but est de proposer un format pur leur *création*. Nous avons besoin d'un format moins général et plus précis que celui proposé par la TEI ; nous voulons en particulier permettre le plus grand nombre possible de manipulations automatiques. Rien ne nous interdira d'ailleurs de transcrire le dictionnaire au format de la TEI, dont le but explicite est l'échange de textes plus que leur manipulation (collectif 1994a).

La partie actuellement fonctionnelle du lexique est plus fruste que le modèle en annexe, mais a achevé de nous convaincre de l'utilité d'une hiérarchisation plus fine (d'autant qu'il est coûteux en temps de modifier la structure d'un dictionnaire).

3.5.4 Liens entre le dictionnaire et les textes

Le dictionnaire est lié à la base de textes, par le biais des exemples. Un usage donné d'un mot peut ainsi être référencé, de préférence lors de l'étude du texte. Inversement, lors de l'usage du dictionnaire, il est nécessaire de pouvoir consulter les exemples en contexte. En effet, il ne faut pas perdre de vue que dans nombre de cas, le sens d'un mot est déduit de son observation dans un nombre plus ou moins grand de textes. Toute traduction d'un mot rare est susceptible de remise en cause. L'intérêt de disposer d'un ensemble de textes informatisés liés au lexique est que cette mise en cause devient matériellement aisée, et que le philologue y recourra plus volontiers, améliorant ainsi son travail.

Le système permet donc de créer des liens entre le lexique et le texte étudié. Ces liens ont la même forme que les références trouvée dans les travaux sur papier, ce qui facilite la communication ; ils sont en effet signifiants par eux-mêmes. Cela impose de pouvoir doter le texte d'un système de référence paramétré. Dans l'exemple de la figure 3.3, le texte commence « page 3, ligne 6 ». Le système de repérage dans le texte peut être modifié, pour inclure des repères différents : colonnes, murs, par exemples, dans le cas de textes gravés. De plus, la base contient un fichier associant les noms symboliques des textes (ici, Prisse) aux entités informatiques concernées.

Lors de l'édition d'un terme dans le lexique, il est possible d'insérer automatiquement une référence hypertexte à la ligne courante dans le texte (figure 3.4). Le système a actuellement quelques défauts : il serait souhaitable de rendre les références multilingues, d'une part, et d'autre part de permettre

FIG. **3.3** – Un texte édité avec notre système : le papyrus Prisse

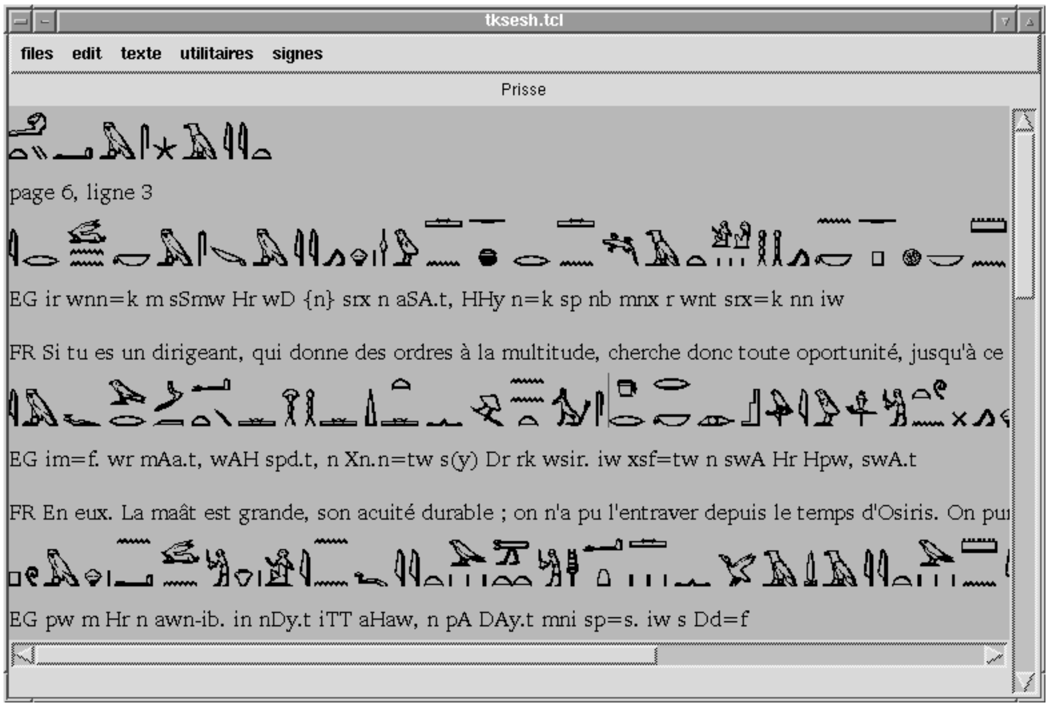
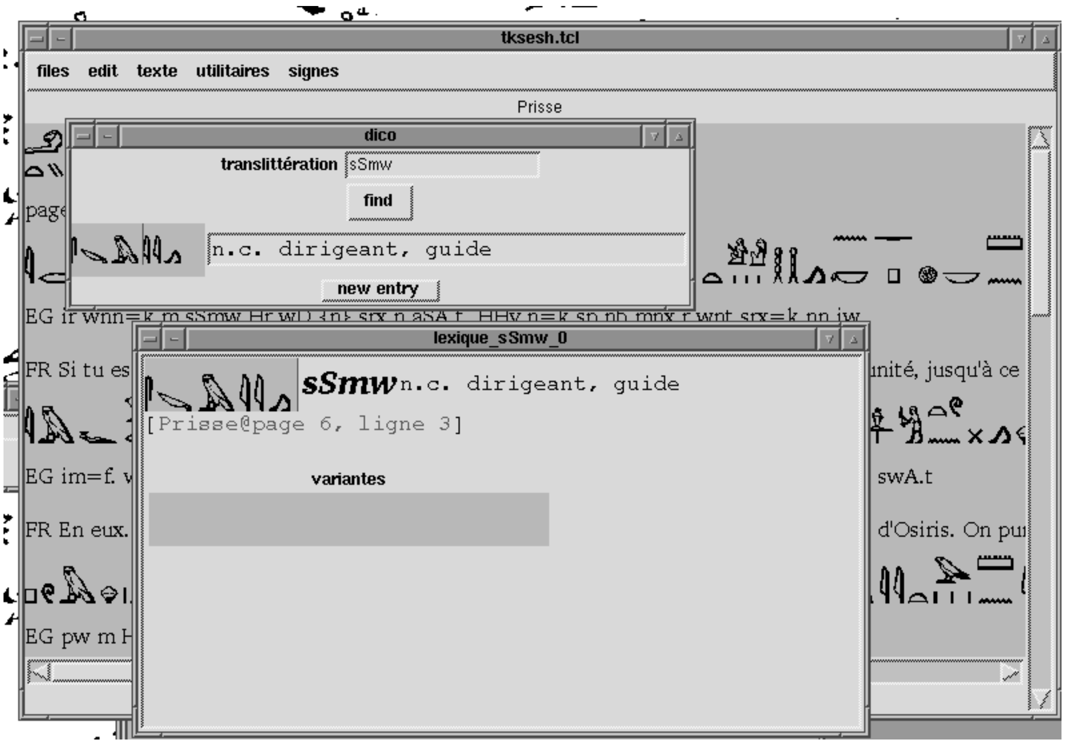


FIG. **3.4** – Dictionnaire et liens hypertextes



l'usage de plusieurs systèmes de références (par exemple, par rapport à un document original et à une édition « papier » canonique).

Il manque encore à ce système une gestion efficace des liens entre le texte et ses interprétations (translittération, traduction, commentaires, ...), entre le texte et ses parallèles (autres copies de la même œuvre).

Nous voulons pouvoir représenter le contenu de l'étude d'un texte un peu complexe, de manière à ce que toutes les parties qui sont logiquement reliées le soient physiquement. Les textes que nous utiliserons pour ce faire seront, d'une part, *l'enseignement d'Amenemopé,* étudié au cours du Professeur P. Vernus à l'ÉPHÉ, et l'*enseignement loyaliste,* parce que l'étude qu'en a fait Georges Posener nous semble constituer un document assez complexe pour que la possibilité de le représenter soit une validation de notre démarche dans ce domaine.

Chapitre 4

Conception de l'Analyseur

Sommaire

Introduction

Notre but étant simplement de proposer l'analyse la plus utile possible, et non de démontrer la valeur d'un type donné de grammaire formelle, nous avons choisi une approche expérimentale. Pour ce faire, nous avons commencé en utilisant un modèle classique, celui des grammaires hors contexte, munies de l'unification. Ce modèle est proche des représentations usuellement utilisées en grammaire, ce qui nous permet d'utiliser directement les connaissances que nous avons sur la langue. Il est aussi central dans le traitement du langage naturel, en ce sens que la plupart des autres formalismes s'y ramènent. Choisir de partir d'un modèle plus complexe nous eût engagés à adhérer à ce dernier, ou tout au moins, à centrer notre travail sur sa validation, ce qui ne correspondait pas à notre intention.

Le formalisme utilisé, tant pour le lexique que pour la grammaire, et le fonctionnement de notre analyseur sont décrits dans les sections qui suivent.

Nous avons donc commencé par coder la grammaire telle qu'elle se présente dans les précis. En outre, la structure même de notre corpus, extrêmement régulier à bien des égards, nous a poussé à dépasser le cadre de la proposition pour essayer de modéliser un tant soit peu le texte. Ce premier codage, ses choix et sa justification, font l'objet de la section 4.4.2 de ce travail.

Nous avons assez rapidement discerné les limites de cette approche. Elle conduit en effet à une explosion combinatoire du nombre des solutions proposées à l'analyse. L'analyse automatique ayant révélé des défauts d'adéquation de ce paradigme au problème posé, nous avons tenté, plutôt que d'abandonner totalement les représentations *hors contexte,* de les adapter au mieux à notre problème. En examinant les analyses proposées, nous avons discerné un certain nombre de constructions grammaticales génératrices d'ambiguïtés. Il fut naturel de tenter de les rendre moins ambiguës, quitte pour cela à perdre en précision.

Nous avons donc écrit une seconde grammaire, fruit d'une simplification de la première, qui fait l'objet de la section 4.4.3, suivie d'un commentaire et d'une comparaison des processus d'écriture des deux de nos deux grammaires.

Au vu des résultat, il est apparu que les formalismes proposés ne permettaient pas, à eux seuls, de résoudre le problème ; c'est pourquoi, en conclusion de ce chapitre, nous proposons d'étendre notre système à l'aide d'automates finis.

4.1 Grammaires

Nous allons présenter ici le formalisme que nous avons utilisé pour décrire nos grammaires. Il s'agit, nous l'avons dit, d'un formalisme classique, fondé sur des règles hors contexte. Le formalisme utilisé pour le lexique est exposé en 4.1.1, le formalisme grammatical en 4.1.2.

Le problème de la disponibilité de grands corpus se pose de manière aigüe pour le moyen égyptien, surtout s'il s'agit de corpus informatiques, et nous avons pour cette raison évité toute méthode demandant un ensemble important de données.

Nous abordons donc le problème avec un formalisme choisi, non pas trop pour ses qualités propres, mais justement pour sa neutralité, parce que nous nous intéressons plus ici aux problèmes pratiques qui vont se poser qu'aux qualités représentatives de nos grammaires.

La taille des grammaires est de 151 règles pour la première grammaire, avec 3598 formes fléchies dans le lexique, pour environ 800 entrées lexicales au départ.

4.1.1 Formalisme lexical

Un mot est représenté par un ensemble de traits grammaticaux, un trait pouvant être composé. Ainsi, le mot , *nḏs,* « roturier, » est-il décrit *in fine* par les traits :

```
(1) [
     graphie :: nDs,
     accord :: [genre :: masculin, nombre :: singulier],
     cat :: nom
    ]
```

où l'accord est un trait composé. De même que le formalisme grammatical, le formalisme lexical a été choisi pour sa simplicité. Il s'agit du formalisme habituel pour les grammaires d'unifications, décrit par G. Sabah (Sabah 1989). Nous discuterons plus loin le choix du système de traits lui-même, cette section ne s'occupant que du formalisme.

La description du lexique se fait à partir d'un certain nombre d'opérateurs, que nous allons détailler.

Les mots sont groupés en *familles,* une famille étant déterminée par, d'une part, la propriété d'avoir les mêmes traits, et d'autre part, de connaître les mêmes variations morphologiques. Ainsi les deux règles :

```
(2) declare_famille(ncm, [cat::nom],[ncms,ncmp]).
```

```
declare_famille(collectifs, [cat::nom],[identite]).
```

définissent deux familles, l'une, celle des *noms communs masculins,* l'autre celle des noms collectifs. Le troisième argument de chacune des règles est une liste de transformations, qui, dans le cas du nom commun masculin, génèrent ses formes singulier (`ncms`) et pluriel (`ncmp`), et dans le cas du nom collectif, le laissent identique à lui-même, puisqu'il est invariable.

Les déclarations :

(3)
```
declare_famille(ncm, [cat::nom],[ncms,ncmp]).

  ncms(CHAINE,DEP,
        [NOM/[graphie::NOM,
              accord::[genre::masculin,
              nombre::singulier]|DEP]]):-
        name(NOM,CHAINE).

  ncmp(CHAINE,DEP,
        [NOM/[graphie::NOM,
              accord::[genre::masculin,
                       nombre::pluriel]|DEP]]):-
        append(CHAINE,"w",PLUR),
        name(NOM,PLUR).
```

Permettent de déclarer les noms communs masculins, en précisant qu'ils prennent un *w* au pluriel. Une fois cette déclaration faite, une ligne comme :

(4)
```
ncm famille sn, msH.
```

déclare *sn* et *msḥ* comme des noms communs masculins. On génère alors les entrées :

(5)
```
lexword(nom([graphie :: sn,
             accord :: [genre :: masculin,nombre :: singulier],
             cat :: nom]), sn).
lexword(nom([graphie :: snw,
             accord :: [genre :: masculin,nombre :: pluriel],
             cat :: nom]), snw).
lexword(nom([graphie :: msH,
             accord :: [genre :: masculin,nombre :: singulier],
             cat :: nom]), msH).
lexword(nom([graphie :: msHw,
```

```
             accord :: [genre :: masculin,nombre :: pluriel],
             cat :: nom]), msHw).
```

Ce système fonctionne correctement pour les mots que l'on ne veut pas définir plus avant, mais pose un problème pour les verbes, car il ne permet pas au premier abord de décrire de façon orthogonale la morphologie et la syntaxe. Les verbes égyptiens connaissent en effet des variations morphologiques qui dépendent de leur structure consonantique. Cette structure est indépendante[1] des caractéristiques syntaxiques des verbes (transitivité...).

Pour cela, les verbes sont aussi groupés en *groupes,* qui permettent de préciser leur comportement morphologique, leurs caractéristiques syntaxiques étant décrites par leur famille.

Enfin il est possible de décrire directement un mot fléchi.

Le système de traits que nous utilisons est relativement classique, mais utilise des disjonctions explicites pour réduire certaines ambiguïtés. Le verbe, en particulier, est extrêmement ambigu, car certaines classes de verbes ne présentent pratiquement pas de variations morphologiques. Le verbe 𓄔𓅓, «*sḏm*», «*écouter*», a la même orthographe pour plus de dix formes différentes. Il serait gênant de gérer autant d'analyses.

La liste de traits fournies par le lexique a donc actuellement la forme :

```
(6)   [graphie :: sDm,
       forme :: accompli or circonstanciel or imperatif
                or indicatif or infinitif or nominal
                or prospectif or prospectif_ancien,
       racine :: sDm,
       cat :: verbe,
       transitif :: plus,
       mouvement :: moins,
       qualite :: moins,
       causatif :: moins]
```

Cette liste est générée automatiquement.

Il est tentant de généraliser l'idée de regrouper ainsi les valeurs possibles des traits. Mais en fait, ce système n'a d'intérêt que lorsque les différentes interprétations ont beaucoup en commun. Les groupements sont effectués à partir d'une expertise humaine, et non d'un calcul. Aussi est-il préférable de décider quels traits seront concernés. Une approche alternative a cependant été proposée (Briscoe, de Paiva, et Copestake 1993; der Luiden 1992), qui

1. À une exception près, puisqu'il existe des familles de verbes *causatifs*, marqués par le préfixe *s* : *mn, durer* >*smn, établir, rendre durable*‘

permet d'optimiser le lexique en hiérarchisant les structures de traits, afin de cerner l'ensemble des possibles.

4.1.2 Formalisme grammatical

Nous décrivons la grammaire par un formalisme dont les règles sont, à des détails cosmétiques près, proches de celles de PATR. PATR est un langage de représentation, qualifié par Gazdar et Mellish de *lingua franca* du traitement du langage naturel (Gazdar et Melish, o. c. p. 103).

PATR est un formalisme à base de grammaire *hors contexte*, augmentées par un système d'unification de traits. Chaque règle est suivie d'une série d'unifications. Comme nous le repréciserons plus tard, cette particularité augmente la puissance expressive du formalisme, et doit être utilisée avec précaution, puisqu'alors le nombre d'analyses possibles n'est plus borné. À chaque symbole en partie droite peut être associée une série de règles re-

Code 4.1 Une règle PATR

```
S -> NP VP :
        S = VP
        <S subj> = NP.
```

liant la valeur des traits associés aux productions filles à celle produites par l'en-tête.

Comme le montre la figure 4.2, les actions comprises dans les règles sont essentiellement des unifications entre certains des traits des structures mères, désignées par la lettre `u`, et la structure fille, designée par `d`, ainsi qu'avec des valeurs de traits. Comme il est habituel avec les grammaires d'unification, les structures de traits construites représentent la totalité de l'analyse. Elles sont donc complexes, car imbriquées.

Les actions sont pour l'instant de deux types (mais il est simple d'en rajouter) : `X = Y` unifie X à Y, les deux valeurs restant liées pour la suite de l'analyse ; `X \= Y`, où Y est un ensemble de valeurs, est une action qui réussit si une partie des valeurs possibles de `X` n'est pas dans `Y`. `X` doit être instancié. La règle

(7)
```
principale
==>
        verbeC([!d::forme=sdm_in, !d::racine \= wn, u=d]).
```

reconnaît tout syntagme `verbeC` (verbe accompagné de ses actants) une proposition principale, pourvu qu'il contienne un verbe à la forme *sḏm.ỉn=f*

Code 4.2 Exemple de règles

```
propositionl
==>
        proposition([u = d]),
        & propositionl([u::suite = d]).
```

une `propositionl` est une `proposition`, optionnellement suivie d'une `propositionl`; le `&` qui précède le second `propositionl` indique son caractère optionnel.

`[u = d]` recopie dans la liste de traits de `propositionl` celle de `proposition`;

`[u::suite = d]` rajoute éventuellement à la liste de traits associée au propositionl en partie gauche un trait `suite`, dont la valeur est la liste de traits associée au `propositionl` en partie droite.

```
proposition
==>
        debut_proposition([u::actants::actant1=d]),
        syntagme_adverbial([u=d]).
```

Cette seconde règle illustre la structuration des phrases. Elle analyse une proposition dont le prédicat est un adverbe, et dont le sujet (ou premier actant, pour reprendre la terminologie de L. Tesnière (Tesnière 1966)), est analysé dans `debut_proposition`.

(une forme narrative), et que ce verbe *ne soit pas wn*, le verbe *être* (celui-ci étant forcément un auxiliaire dans de telles formations).

De même que les traits pour leur valeur, les règles admettent aussi les disjonctions dans leurs contraintes. Ainsi, soit une règle du type :

```
(8) groupe_verbal
    ==>
       verbe([d:forme = circonstanciel or prospectif or sdm_n]),
       suffixe([u::actant1 = d])
```

elle signifie qu'une forme verbale des types considérés suivie d'un pronom suffixe forme un groupe verbal valide (cette règle est quelque peu simplifiée). Elle nous permettra d'obtenir, si nous analysons l'énoncé *sḏm=k,* « *entendre = il* », composé d'un verbe (pouvant avoir un très grand nombre de valeurs syntaxiques), suivi d'un pronom suffixe, l'analyse :

```
(9) [graphie :: sDm, forme :: circonstanciel or prospectif,
     racine :: sDm,cat :: verbe,transitif :: plus,
     mouvement :: moins,qualite :: moins, causatif :: moins,
     actants :: [
                  actant1 :: [graphie :: '=k',
                              cat :: suffixe,
                              nature :: pronom,
                              accord ::
                                [genre :: masculin,
                                 nombre :: singulier],
                              personne :: 2]],
     type :: verbal]
```

Cette analyse nous indique que le verbe, dont le sujet (premier actant) est le pronom *=k,* est soit à la forme circonstancielle, soit à la forme prospective. On constate qu'un certain nombre de formes, n'admettant pas l'expression du sujet par « *=k* », ont été supprimées. On a, par exemple, éliminé l'impératif. Seules sont donc conservées, après application de la règle, les deux formes, *circonstanciel* et *prospectif*, que celle-ci admet. Il ne s'agit pas là des seules analyses possibles pour ce bout de texte ; une autre règle pourrait reconnaître là un éventuel infinitif, auquel cas le pronom *=k* serait un complément d'objet. L'intérêt de la représentation, c'est que plusieurs formes au comportement syntaxique similaire peuvent être *localement* regroupées, nous évitant ainsi de gérer séparément l'hypothèse `forme :: circonstanciel` et l'hypothèse `forme :: prospectif`.

Nous localisons donc ici l'ambiguïté, sans même être forcé de prendre une décision, *à condition toutefois la grammaire s'y prête.* En effet, si nous écrivons les règles :

(10)
```
r1 ==> verbe([d::forme = circonstanciel]), suffixe.
r2 ==> verbe([d::forme = prospectif]), suffixe.
```

nous obtiendrons bien évidemment deux interprétations séparées. Le mécanisme de groupage des valeurs des traits doit donc être géré par le rédacteur de la grammaire.

Ce traitement permet d'avoir une grammaire plus compacte ; mais surtout il génère moins d'hypothèses, celles-ci étant regroupées, et donc il allège d'autant la tâche de l'analyseur. Nous insistons sur le fait que les regroupements opérés ne le sont que par décision de l'auteur de la grammaire. Il serait sans doute intéressant de chercher si le choix des traits concernés par ce type de phénomène est automatisable. Il ne peut l'être *a priori,* car on ne peut le faire qu'au vu de l'interaction entre la grammaire et le lexique. Une automatisation demanderait une analyse profonde de la structure de la grammaire, et en particulier une mesure de la similarité des règles.

4.2 Pré-traitement

La base de règles que nous avons décrite n'est pas facilement manipulable par un système automatique.

Les règles sont donc traitées ; chaque extension possible d'une partie droite d'une règle (en tenant compte des « & », qui, rappelons-le, marquent les éléments optionnels) est étendue séparément. Pour éviter tout problème, les règles permettant de dériver la chaîne vide ne sont pas enregistrées dans le système final, ce qui n'est pas grave puisqu'elles sont *directement* prises en compte dans les règles qui les utilisent.

L'algorithme mis au point est le suivant :

POUR toute règle $H \rightarrow T$ **FAIRE**
  calculer toutes les formes possibles de T, selon que les éléments optionnels sont utilisés ou non
FIN
generent_vide := $\{\}$
POUR toute règle de la forme $H \rightarrow []$ **FAIRE**
  supprimer $H \rightarrow []$ de la base
  SI ($H \notin generent\_vide$) **ALORS**
    generent_vide := generent_vide$\bigcup\{H\}$
    pour toute règle $H' \rightarrow T$ **FAIRE**

SI $H \in T$ **ALORS**
générer toutes les règles possibles où la(les) occurrences de H sont supprimées.
FIN
FIN
FIN

Comme une tête donnée n'est supprimée qu'une fois, l'algorithme s'arrête. En fin de compte, on obtient des règles sans production vide, mais équivalentes aux règles de départ, à ceci près que si le langage (au sens mathématique) d'origine pouvait engendrer la production vide, il ne le peut plus.

4.3 Algorithmes

Nous utilisons un *chart parser*, ou *analyseur par graphe,* pour l'analyse syntaxique, en nous inspirant de Gazdar et Melish (Gazdar et Mellish 1989). Un *chart parser* est un système efficace pour analyser des grammaires hors-contexte. Les ancêtres des algorithmes de ce type remontent aux années 70 (Kay 1973; Kaplan 1973). C'est un système en $\mathcal{O}(n^3)$, qui de plus, permet de garder une trace des analyses partielles. Ce dernier point est évidemment intéressant, car il permet de traiter, au moins partiellement, les phrases non grammaticales.

L'idée du *chart parser* est la suivante : on crée un graphe dont les nœuds sont les frontières des mots et dont les arcs sont labellés par les règles partiellement dérivées.

Un arc entre a_i et a_j, qui a pour label $(R, (A_1, \ldots \bullet A_k, \ldots, A_n))$ signifie qu'il existe une règle $R \rightarrow A1, \ldots, A_n$, et une dérivation $A_1, \ldots A_{k-1} \rightarrow a_i, \ldots, a_j$; c'est-à-dire que l'on a reconnu le début d'une séquence que la règle R peut générer. Dans la pratique, les labels peuvent être de la forme $(R, (\bullet A_k, \ldots, A_n))$, car les $A_{i,i<k}$ ne sont d'aucune utilité.

La méthode générale d'utilisation de la charte est la suivante : l'on construit un *agenda*, qui est une liste d'arcs à traiter. À chaque fois que l'on retire un arc de l'agenda pour le rajouter dans la charte, on déduit du rajout de cet arc l'existence d'un certain nombre d'autres arcs, qui sont rajoutés dans l'agenda.

Lorsque l'algorithme s'arrête, l'on dispose d'un ensemble d'arcs, qui correspondent aux dérivations possibles.

Soit par exemple le lexique :

le ART

la ART

chat NomC

souris NomC

souris Verbe

mange Verbe

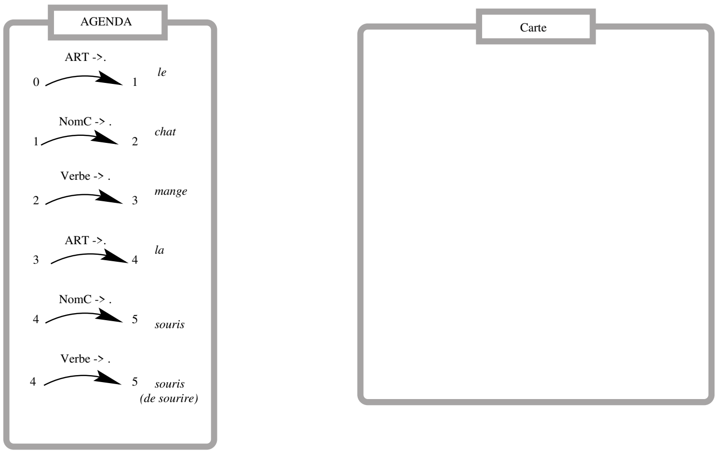
et la grammaire :

```
Phrase ==> GN GV.
GN ==> ART NomC.
GV ==> Verbe GN.
```

et soit à analyser la phrase « le chat mange la souris ».

On met dans l'agenda les arcs correspondant aux mots de la phrase (figure 4.1 ;

FIG. **4.1** – Initialisation de la charte



puis un arc de l'agenda est retiré et placé dans la charte, qui est un graphe initialement vide ; ici, il s'agit de l'arc correspondant à « le ». Le rajout de cet arc dans la charte entraîne la création d'un nouvel arc,

$$(0, 0, \mathrm{GN} \rightarrow \bullet \mathrm{ART\ NomC.})$$

FIG. **4.2** – Le début du traitement

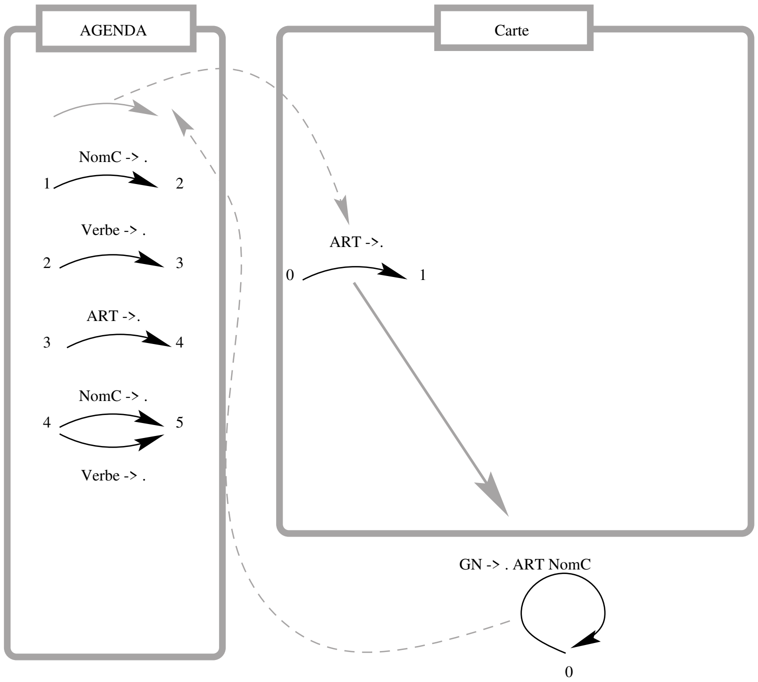


FIG. **4.3** – reconnaissance du groupe nominal

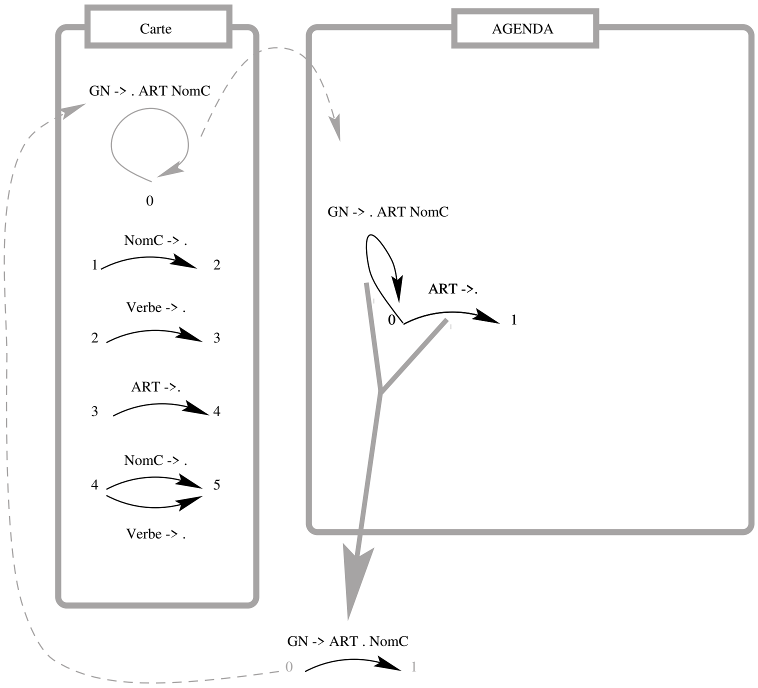
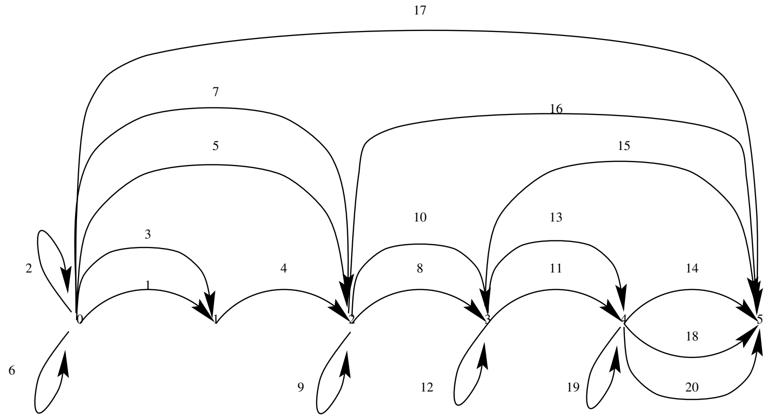


FIG. **4.4** – État final de l'analyse



arc	: label	arcs pères
1	: ART ->•	
2	: GN ->• ART NomC	1
3	: GN ->ART • NomC	1 2
4	: NomC ->•	
5	: GN ->ART NomC•	3 4
6	: P ->• GN GV	5
7	: P ->GN • GV	5 6
8	: Verb ->•	
9	: GV ->• Verb GN	8
10	: GV ->Verb • GN	8 9
11	: ART ->•	
12	: GN ->• ART NomC	11
13	: GN ->ART • NomC	11 12
14	: NomC ->•	
15	: GN ->ART NomC•	13 14
16	: GV ->Verb GN•	10 15
17	: P ->GN GV•	7 16
18	: Verb ->•	
19	: GV ->• Verb GN	18
20	: GV ->Verb • GN	18 19

qui indique que si le système reconnaît ART puis NomC après le nœud 0, on reconnaîtra le groupe GN. Ce nouvel arc est ajouté dans l'agenda. Plusieurs stratégies sont possibles ; dans le cas présent, l'arc est ajouté au sommet de l'agenda ; il en sera retiré en premier. Le symbole « • », selon l'usage courant, sépare la partie du texte déjà analysée (à sa gauche) de la partie du texte à analyser (figure 4.2).

Lors de l'étape suivante, l'arc est rajouté à la charte. Son ajout provoque création et l'insertion dans l'agenda d'un arc : $(0, 1, \mathrm{GN} \rightarrow \mathrm{ART} \bullet \mathrm{NomC.})$, qui indique que si par la suite on reconnaît NomC à partir du nœud 1, on reconnaîtra GN à partir du nœud 0 (figure 4.3).

La figure 4.4 montre l'analyse terminée ; les arcs sont numérotés suivant l'ordre dans lequel ils ont été insérés dans la charte. En légende figure, pour chaque arc, la liste des arcs dont il a causé la création.

4.3.1 Gestion de la charte

Les arcs de la charte portent les informations suivantes : leur départ et leur arrivée, et l'état d'analyse auquel ils correspondent. par exemple,

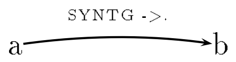
```
(11) arc(0, 1,
         verbe_d([graphie :: 'Hmst',
                  forme :: participe,
                  aspect :: perfectif,actif :: passif,
                  accord :: [genre :: feminin],
                  racine :: 'Hmsi',cat :: verbe,
                  transitif :: moins,mouvement :: plus,
                  qualite :: moins,causatif :: moins]),
         [], regleN660, 1)
```

précise qu'il y a un arc entre les nœuds 0 et 1, correspondant au symbole non terminal `verbe_d`, que celui-ci a alors comme ensemble de traits associé l'ensemble listé entre crochets (`[graphie :: 'Hmst', forme :: participe... causatif::moins]`), et qu'il correspond à la fin de l'application de la règle `regleN660`, puisque l'ensemble de symboles à dériver listé comme quatrième argument est vide.

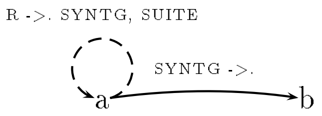
L'analyse est ascendante et en profondeur d'abord. On ajoute d'abord dans l'agenda, pour tous les symboles terminaux, les arcs correspondants, puis on applique les règles suivantes :

TANTQUE l'agenda n'est pas vide **FAIRE**
prendre le premier arc de l'agenda.

SI l'arc a la forme :

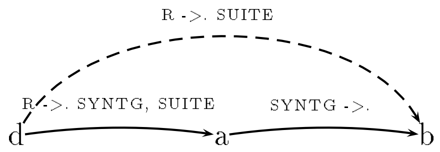


entre a et b.
ALORS pour toutes règle `R -> . SYNTG, SUITE`,
rajouter *dans l'agenda* l'arc (en pointillé):



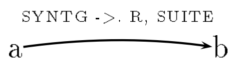
(production 1)

pour nœud d, pour tout arc entre d et a, étiqueté par
`R -> . SYNTG SUITE`, *rajouter dans l'agenda*
l'arc entre d et b étiqueté par
`R -> . SUITE:`



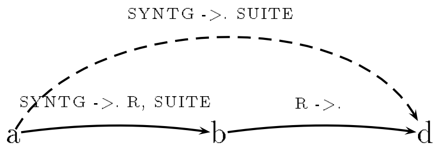
(production 2)

SINON , l'arc a la forme :



(production 3)

entre a et b. Pour tout arc `R-> .` entre b et d on rajoute *dans l'agenda* un arc `SYNTG-> . SUITE` entre a et d.



FIN

4.3.2 Fonctionnement de l'analyseur

Nous allons prouver que l'analyseur fonctionne.

Supposons, dans un premier temps, qu'un arc n'est pas annoté avec les traits correspondants. Alors, le nombre des arcs possibles est majoré par rn^2 où r dépend uniquement des règles et n est le nombre de mots dans la phrase.

Ce nombre d'arcs est, entre autres, fini. Cela garantit la terminaison de l'algorithme. Reste à prouver sa correction et sa complétude.

Les arcs initiaux sont créés à partir des catégories syntaxiques associées aux mots du texte. Ces arcs ont donc une justification. Toutes les opérations créatrices d'arcs sont elles aussi justifiées et correspondent à des étapes de l'analyse grammaticale du texte. En conséquence de quoi, le résultat de l'analyse par *chart parser* correspond à une analyse grammaticale correcte.

Reste à prouver que toute analyse grammaticale sera trouvée par l'analyseur. Pour cela, considérons une analyse. Nous pouvons représenter tous les éléments qui la composent par des arcs. Il nous suffit donc de prouver que tous ces arcs sont construits par notre algorithme.

Pour cela, nous allons associer à chaque arc α un nombre, $\mathrm{h}(\alpha)$, défini comme suit :

- tout arc qui peut être engendré par le lexique est de hauteur 0 ;
- un arc α engendrable par l'application (éventuellement partielle) d'une règle à partir de deux arcs α_1 et α_2 sera de hauteur

$$\mathrm{h}(\alpha) = \max(\mathrm{h}(\alpha_1), \mathrm{h}(\alpha_2)) + 1$$

– un arc bouclant sur un nœud, labellé par $S \to A_1, \ldots, A_n$ signifie « on soupçonne qu'il est possible grâce à cette règle de reconnaître S à partir de ce nœud ». On pourrait, sans faute logique, supposer que toutes les règles possibles doivent être envisagées à partir de tout nœud ; néanmoins, ce serait inefficace. Aussi a-t-on décidé, dans l'algorithme, de ne rajouter de tels arcs que lorsqu'il existe un arc partant du nœud et labelé par $A_1 \to \ldots \bullet$. Dès lors, nous décidons que pour un arc a bouclant sur un nœud et de label $S \to \bullet A1 \ldots$,

$$\mathrm{h}(()a) = 1 + \min \{\mathrm{h}(b) | b \mathrm{commencepar} A_1\}$$

Cette définition n'est pas circulaire, grâce à l'opérateur *min*, car notre arc a ne saurait être présent si aucun arc b n'a été construit.

Soit P(n) la proposition

tous les arcs de hauteur inférieure ou égale à n sont construits par le chart parser.

Par construction, P(0) est vraie, car tous les arcs « lexicaux » sont dans l'agenda.

Prouvons que P(n) $\Rightarrow$ P($n+1$).

Supposons P(n) ; soit a un arc de hauteur $n+1$. a correspond à l'application (éventuellement partielle) d'une règle entre deux frontières de mots w_1 et w_k. Si le label provient de l'application partielle d'une règle de la forme :

$$S \to A_1, \ldots, A_i, \bullet, A_{i+1}, \ldots, A_n. \tag{4.1}$$

alors, par hypothèse, un arc labellé

$$S \to \bullet, A_i, \ldots, A_n. \tag{4.2}$$

a été généré entre w_1 et $w_{k'}$, et un arc labellé

$$A_i \to \bullet. \tag{4.3}$$

a été lui aussi généré, entre w'_k et w_k. Si l'arc 4.3 est inséré dans la charte après l'arc 4.2, la règle de production 2) de l'algorithme engendrera un arc entre w_1 et w_k, de label

$$S \to A_1, \ldots, A_i, \bullet, A_{i+1}, \ldots, A_n \tag{4.4}$$

Si au contraire, l'arc 4.2 est inséré dans la charte après l'arc 4.3, alors, c'est la règle de production 3) qui intervient.

Un arc dont le label correspond à une application partielle de la forme

$$S \rightarrow \bullet, A_1, \ldots, A_n. \tag{4.5}$$

ne conduit à une reconnaissance que si le nœud sur lequel il s'appuie est la frontière gauche d'un syntagme A_1.

Dans ce cas, on a bien génération, par la règle de production 1), de l'arc labellé correspondant.

4.3.3 Complexité

Remarquons qu'en construisant un tableau de taille n^2, on peut obtenir l'accès aux arcs en un temps indépendant de n (après construction du tableau).

Le temps d'ajout d'un arc est proportionnel, au pire, à n, car le nombre d'arcs dotés d'un label donné pouvant partir ou aboutir à un nœud est au plus n.

Il y a (au plus) n^2 arcs labellés par un label donné dans la charte, et le nombre de labels possibles dépend seulement de la base de règles.

La complexité dans le pire cas est donc $\mathcal{O}(n^3)$.

4.3.4 Gestion des traits

Pour gérer l'unification, la méthode la plus simple consiste à appliquer les règles au fur et à mesure de la construction des arcs, en décorant ceux-ci de la valeur des traits correspondants. Mais si un *chart parser* simple est *a priori* en $\mathcal{O}(n^3)$, c'est parce que le nombre maximal d'arcs est en $\mathcal{O}(n^2)$ (n étant la longueur du texte analysé). Si les arcs sont accompagnés des valeurs des traits qui leur correspondent, la complexité, tant spatiale que temporelle, ne peut plus être correctement définie car le nombre d'analyses possibles peut être infini. Avec des règles comme :

```
(12) p ==> p([u::suite = d]).
     p ==> s([u::debut=1]).
```

on peut obtenir une infinité d'analyses :

```
(13) [debut :: 1]
     [suite::[debut :: 1]]
     [suite::[suite::[debut :: 1]]]
     [suite::[suite::[suite::[debut :: 1]]]]
```

La complexité, tant spatiale que temporelle, ne peut plus être correctement définie.

Par bonheur, ce comportement peut être évité. Il suffit d'écrire des règles «raisonnables». On peut par exemple décider d'écrire des règles telles que les dérivations (au sens des grammaires) de la forme $r \xrightarrow{*} r$ soient exclues. Le nombre de règles applicables entre deux nœuds sera alors borné. Une méthode simple pour obtenir ce résultat est de hiérarchiser les règles selon leur niveau d'abstraction. Une règle non-récursive dans un niveau hiérarchique donné ne peut faire appel qu'à des règles de niveau hiérarchique strictement inférieur. On limite de plus les règles récursives aux traitements des listes.

Dans ce cas, les structures de traits se contentent de refléter les analyses syntaxiques. Le nombre de celles-ci peut être exponentiel : pour la grammaire

```
(14) Phrase ==> Syntagme &Phrase.
       Syntagme ==> mot &Syntagme.
```

une interprétation est une suite de `Syntagme`, chaque `Syntagme` étant une suite arbitraire de mots. Une interprétation correspond alors à un découpage arbitraire de la phrase en groupes connexes de mots.

Soit $N(n)$ le nombre de tels découpages possibles, n étant la longueur de la phrase. Le premier groupe de mots peut faire de 1 à n mots. S'il fait 1 mot, il reste $n-1$ mots à grouper, et il y a donc $N(n-1)$ groupements possibles si le premier groupe compte 1 mot. En général, il y a $N(n-i)$ groupements possibles si le premier groupe comporte i mots. D'où :

$$\begin{aligned}
N(1) &= 1 \\
N(n) &= N(n-1) + \underbrace{N(n-2) + \cdots + N(1) + 1}_{\text{on reconnaît } N(n-1)} \\
&= N(n-1) + N(n-1) \\
&= 2N(n-1) \\
&= 2^{n-1}
\end{aligned}$$

La complexité spatiale et temporelle du système est beaucoup trop importante. Il est possible de la ramener à $\mathcal{O}(n^3)$, pour la construction des arcs, en n'appliquant pas les règles de traitement des traits. Une fois les arcs construits, on n'aura plus qu'à re-synthétiser les analyses en appliquant cette fois les règles d'unification.

Certaines des règles limitent le nombre d'analyses ; il est donc malencontreux de ne pas les utiliser dès l'abord. Ce sont principalement des règles destinées à vérifier précisément la nature d'un mot. Nous adoptons donc

Code 4.3 Actions immédiates et retardées

```
verbeC1 ==>
    verbe_c([! d::transitif = plus, ! u=d]),
    a1a2oa3o([u::actants=d]).
```

`verbeC1` *correspond à des groupes verbaux où le sujet est exprimé.* `a1a2oa3o` *correspond à l'expression du sujet (premier actant), optionnellement de l'objet direct (second actant), et de l'objet indirect (troisième actant)*

la convention suivante : le «!» sera utilisé pour spécifier qu'une règle sera immédiatement exécutable. Dans la figure 4.3, il importe de savoir immédiatement si le verbe est transitif, afin d'éviter, dans le cas contraire, de chercher un éventuel COD (complément d'objet direct); en revanche, recopier la valeur transmise par `a1a2oa3o` à la construction `verbeC1`, par l'entremise de « `[u::actants=d]` » risquerait de poser plus de problème que cela n'en résoudrait. En effet, `a1a2oa3o` représente les actants [2] de notre verbe ; on constate à l'expérience que ceux-ci génèrent des ambiguïtés. La forme verbale, par contre, est circonscrite (c'est **un** mot), et sa connaissance est utile dans les règles de plus haut niveau, qui utilisent `verbeC1` ; il faut donc transmettre sa valeur.

La méthode utilisée est donc la suivante : on construit de manière ascendante, selon l'algorithme donné page 100, une charte correspondant aux analyses possibles ; la procédure d'ajout d'un arc dans l'agenda est la suivante :

SI les unifications précédées de «!» s'appliquent
  ALORS
  rajouter l'arc dans l'agenda
FIN

Il reste, lorsque l'algorithme se termine, à construire les interprétations.

On applique pour ce faire les prédicats que nous allons décrire :

Le prédicat `find_analyse_aux` effectue la synthèse des analyses. Son travail est de reconstituer, en analyse ascendante, les calculs effectués lors de la création de la charte.

2. sujet, complément d'objet, complément d'objet second

`find_analyse_aux` prend cinq arguments :

- `DEBUT` est le symbole à reconnaître;
- `NUM` et `NUM1` sont les bornes entre lesquelles il faut reconnaître ce symbole ;
- `VI` est la valeur initiale de l'ensemble des traits ;
- `VF` est la valeur finale de cet ensemble.

La sémantique du prédicat est que les mots entre `NUM` et `NUM1` permettent de dériver `DEBUT`, et fournissent les traits `VF` à partir de `VI`.

```
(15) find_analyse_aux(DEBUT,NUM,NUM1,VI,VF):-
             rule(DEBUT, R),
             existe_chaine(R,NUM,NUM1,ARCS),
             se_trouve_aux(R,NUM,NUM1,ARCS,VI,VF).
```

S'il y a une règle

```
(16) DEBUT ==> R
```

et que l'analyse a appliqué cette règle pour analyser les mots entre `NUM` et `NUM1`, alors chercher les valeurs des traits correspondant avec le prédicat `se_trouve_aux`.

```
(17) find_analyse_aux(DEBUT,NUM,NUM1,[],VF):-
             S=.. [DEBUT, VF],
             (\+ rule(DEBUT, _)),
             NUM1 is NUM + 1,
             arc(NUM,NUM1, S, []).
```

Si `DEBUT` est terminal, alors la valeur des traits est donnée par l'arc.

```
(18) se_trouve_aux([],NUM,NUM,_,VF,VF).

     se_trouve_aux([R0|Reste],NUM,NUMF,[arc(NUM,NUM1,[])|ARCS], VI,VF):-
             R0=..[Rd,Regles],
             find_analyse_aux(Rd, NUM,NUM1,[],DOWN),
             applique_toutes_regles(V2,VI,DOWN,Regles),
             se_trouve_aux(Reste, NUM1, NUMF,ARCS, V2, VF).
```

La théorie, nous l'avons vu, ne garantit rien. Il est possible d'écrire des structures de traits qui rejettent toute les interprétations que fournirait la grammaire seule. En pratique, le système permet de construire la première interprétation en un temps raisonnable.

Typologiquement, les règles d'unification sont de deux types : un premier type correspond à la construction d'un résultat, il engendre des structures complexes, mais les unifications auquelles il procède fonctionnent toujours, parce qu'elles sont un reflet de structure grammaticale.

D'autres règles, celles pour lesquelles nous avons introduit le «! », effectuent des vérifications, sur des structures de taille limitée, à des fins essentiellement de classification. Ces règles, qui correspondent à une réécriture partielle de certains éléments, peuvent être appliquées sans dommage lors de la construction de la charte.

Si cette contrainte est respectée, la création de la charte sera effectuée en temps $\mathcal{O}(n^3)$, et la reconstruction de l'analyse se fera à coup sûr.

Dans le cas contraire, l'exploration de l'espace des solutions se ferait dans le pire cas en temps proportionnel au nombre de solutions, puisqu'il serait possible d'invalider une analyse lors de la reconstruction, et donc éventuellement de les invalider toutes.

En particulier, dans le cas où une phrase n'a aucune analyse correcte, un usage correct de «! » permettra de détecter l'absence de solutions lors de la construction des arbres, et non par échec de l'exploration systématique de l'ensemble des solutions.

4.4 Écriture d'une grammaire

4.4.1 Commentaires préliminaires

L'analyseur que nous venons de décrire a été utilisé sur notre corpus.

Nous avons écrit pour ce faire une première grammaire, relativement ambitieuse quant à la précision de l'analyse ; l'expérimentation a démontré qu'elle comportait un certain nombre de défauts, que nous présenterons ci-après. Nous avons donc écrit une seconde grammaire en vue de les corriger.

Nous avons commencé en essayant de représenter d'assez près la grammaire telle que la décrivent les recherches. Il nous semblait en effet intéressant de voir dans quelle mesure elle se prêtait à l'automatisation.

Comme le système d'écriture ne comporte pas de ponctuations, nous avons essayé de représenter (au moins partiellement) la syntaxe du texte (par opposition à la syntaxe de la phrase). Il nous a en effet semblé nécessaire que notre système fût capable de proposer une segmentation des propositions.

Nous étions alors naturellement amenés à écrire, non une grammaire de la phrase, mais une grammaire du texte. Cette orientation a plusieurs conséquences. En particulier, elle nous a conduit, lors de la rédaction de la première grammaire, dans un but de structuration et de précision, à tenter de représenter les différents registres du texte. La distinction entre formes du récit et formes du discours nous semblait pertinente et efficiente dans le cadre de notre étude, car les formes sont extrêmement marquées dans notre corpus. Allant plus loin, nous avons tenté de représenter les différentes parties des répliques.

Un problème de méthode se pose à nous dès l'abord. Nous disposons d'un corpus diachroniquement marqué ; un certain nombre de tournures n'y apparaissent pas. Quelle généralité faut-il donc donner à notre grammaire ? Si celle-ci ne devait rendre compte que du corpus, alors, à l'extrême, elle pourrait être une liste des phrases du texte. Notre but est de viser une plus grande généralité, afin que le système soit utilisable sur d'autres textes. Dans l'état actuel des choses, nous avons omis les constructions non attestées dans notre corpus. Il serait néanmoins bon de pouvoir les intégrer, en particulier si nous considérons que certains s'inscrivent dans une dialectique entre la règle et la compréhensibilité. Prenons un exemple. Les compléments adverbiaux se situent normalement en fin de phrase. Mais, dans un énoncé

Verbe Sujet-groupe-nominal Compléments-circonstantiels

un sujet trop long empêcherait le lecteur de réaliser le rattachement prépositionnel entre le verbe et ses circonstants. Aussi trouve-t-on dans quelques cas la forme

Verbe Compléments-circonstantiels Sujet-groupe-nominal

Ce problème ne se pose pas dans notre corpus, mais il est nécessaire que la grammaire puisse évoluer pour en tenir compte.

Toutes les constructions récursives, les compléments de noms et les énumérations, posent un problème. Il est hasardeux d'en limiter la profondeur, quoique sur un corpus fini cela soit trivial. Il n'en reste pas moins que plus une structure est imbriquée, plus sa syntaxe est contrainte. Une longue énumération, par exemple, tendra à exhiber un certain parallélisme entre ses termes, sauf éventuellement les derniers qui pourront fonctionner comme une récapitulation.

4.4.2 Commentaires sur la première version de notre grammaire

La première grammaire que nous avons réalisée pourrait être qualifiée, selon les goûts, de « *savante* » ou de « *naïve* », en ce sens qu'elle correspond à peu près à celle qui se trouverait dans un livre, et qu'utiliserait un humain.

Cette grammaire distingue la narration du discours, distinction marquante dans un conte. Cela a des effets importants, car la grammaire des formes narratives est extrêmement simple dans notre texte.

Les formes employées répondent en effet à un nombre très réduit de schémas, introduits par des marqueurs bien déterminés. La liste qui peut en être dressée est courte. Donnons-en quelques exemples :

(19)
```
principale
==>
        ['aHa.n'],
        a1([u::actants= d]),
        verbe([! d::transitif=moins,! d::forme=accompli, u=d]),
        & psp,
        & a3([u::actants = d]).
```

Cette règle reconnaît les phrases introduites par l'auxiliaire narratif *ꜥḥꜥ.n*. `a1` et `a3` reconnaissent respectivement le premier et le troisième actants ; `psp` quant à lui reconnaît un pronom de rappel du premier actant. Le verbe est ici une forme accomplie, le tout a une valeur de passé narratif accompli. C'est une forme rare (trois occurrences dans notre texte).

Exemple 22 (p. 0, l. 0.)

translittération :	*ꜥḥꜥ.n*	*nḫꜣw*	*n*	*mfkꜣt*	*mꜣt*	*ḫr*	*ḫr*	*mw*
littéralement :	*(aux.)*	*broche*	*de*	*turquoise*	*neuve*	*fut tombée*	*sur*	*l'eau*

Alors une broche de turquoise neuve se trouva tombée a l'eau.

(20)
```
principale
==>
        verbe([! d::forme= infinitif, ! d::transitif= moins,u=d]),
        ['pw'],

        verbe([! d::racine = iri,
        ! d::forme = forme_relative,
```

```
        ! d::aspect=perfectif]),

        a1a3o([u::actants= d]).
```

reconnaît la phrase

Exemple 23 (p. 0, l. 0.)

translittération :	*ỉwt*	*pw*	*ỉr.n=f*
littéralement :	*venir*	*cela*	*ce que il fit*

Alors, il vint

```
(21) principale
     ==>
          ['aHa.n'],
          verbe([! d::forme=sdm_n,u=d]),
          a1a2oa3o([u::actants= d]).
```

reconnaît la phrase

Exemple 24 (p. 0, l. 0.)

translittération :	*ˁḥˁ.n*	*ḏd.n*	*ḥm=f*	*n*	*pꜣ*	*msḥ*
littéralement :	*(aux)*	*dit*	*sa Majesté*	*à*	*le*	*crocodile*

Alors, sa Majesté dit au crocodile...

```
(22) principale
     ==>
          verbe([!d::forme=sdm_in, !d::racine = wni]),
          a1([u::actants::sujet=d]),              % PB avec les actants des. infinitives
          preposition([u::prep=d]),
          gn([d::cat \= verbe]).
```

reconnaît la phrase

Exemple 25 (p. 0, l. 0.)

translittération :	*wn.ỉn=f*	*m*	*ḏrt=f*	*m*	*msḥ*	*n*	*mnḥ*
littéralement :	*il fut*	*dans*	*sa main*	*en tant que*	*crocodile*	*de*	*cire*

il se fut dans sa main sous forme de crocodile en cire

```
(23) principale
```

```
==>
        verbe([!d::forme=sdm_in, !d::racine = wni]),
        a1([u::actants=d]),
        verbe([!d::forme=accompli,u=d]), &psp.
```

reconnaît la phrase

Exemple 26 (p. 0, l. 0.)

translittération :	*wn.ỉn*	*ỉb=f*	*nfr*
littéralement :		*son coeur*	*étant bien*

Son cœur s'en trouva bien

On constate cependant que dans nombre de cas, ces schémas restent des constructions analysables, même si leur analyse a peu de sens. Ainsi, la construction

translittération :	*wḏꜣ*	*pw*	*ỉr.n=f*
littéralement :	*aller*	*ça*	*ce qu'il fit*

Alors il alla

s'analyse comme une phrase nominale : « C'est un aller, ce qu'il fit » et doit se comprendre « alors il alla ». Si nous ne disposons pas de l'indication que nous sommes en contexte narratif, l'interprétation : « c'est un qui va, ce qu'il fit » devient grammaticalement possible. Et il est même possible de trouver un sens à cette traduction, puisque le verbe « faire » est utilisable, un peu comme en français familier, pour désigner l'action d'embrasser une carrière. On exhorte le jeune homme : « *ỉr sš* », « fait scribe ! ». On pourrait donc comprendre, avec beaucoup de bonne volonté, notre passage comme « il a embrassé la carrière de voyageur ». Seuls des critères contextuels ou pragmatiques nous permettent d'écarter tout à fait cette interprétation.

Les dialogues, en revanche, donnent lieu à des difficultés beaucoup plus grandes. Tout syntagme peut y être employé de façon autonome :

« *n ỉs n rmṯ !* » « *pas à un homme !* »

Cette structure courante dans des situations de dialogues réels nous montre un énoncé réduit à un syntagme adverbial. Une grammaire complexe devient alors un handicap, car les constructions décrites de manière fine sont souvent des cas particuliers de constructions plus générales, et on obtient par conséquent une ambiguïté entre les deux séries de constructions.

Nous avons tenté, dans un premier temps, d'éviter ce problème en ayant une grammaire précise et globale du discours. Nous pensions pouvoir délimiter différentes parties dans les répliques : une « introduction » et un « corps ».

Dans cette optique, l'introduction aurait concentré les formes les plus irrégulières, c'est à dire les syntagmes, adverbiaux ou nominaux, ayant à eux seuls une valeur propositionelle. En effet, les constructions irrégulières correspondent souvent au début des répliques, soit que l'implication du locuteur[3] y soit plus forte, soit, plus simplement, que la construction de l'énoncé se base sur les répliques précédentes, voire sur le contexte.

Les syntagmes concernés sont essentiellement les vocatifs, comme dans cet exemple typique :

Exemple 27 (p. 0, l. 0.)

translittération :	*ḥnwt*	*ỉrr=t*	*pꜣ*	*ỉb*	*r*	*m*
littéralement :	*Maîtresse*	*le fait que tu fasses*	*ce*	*pensée*	*pour*	*quoi?*

: Maîtresse, pourquoi te mets-tu dans cet état ?

Il s'agit ici du cas régulier, qui ne pose pas de problème. Mais le placement du vocatif en début de phrase n'est qu'une tendance, et des interjections peuvent se trouver à peu près n'importe où dans une réplique.

Les rattachements : décrire « finement » les constructions oblige à résoudre, au moins partiellement, le problème des rattachements prépositionnels. En effet, comme une phrase entière peut être nominalisée, un complément adverbial peut faire partie d'un groupe nominal, analyser complètement les groupes nominaux requiert donc une analyse des syntagmes prépositionnels.

Malheureusement, la langue égyptienne utilise, dans le discours, des formes difficilement distinguables (sinon identiques) pour indiquer les circonstances ou la simple séquentialité. C'est-à-dire qu'elle rajoute, après une construction initiale, une série de circonstants que nous traduirions par des propositions indépendantes. Une phrase comme

	translittération :	*ỉw*	*swỉ.n=f*	*wnm.n=f*
*	*littéralement :*	*(morphème)*	*il a bu*	*il a mangé*

peut se comprendre comme *il a bu, puis/et il a mangé*, en analysant

ỉw (morphème de l'indicatif, annonçant une proposition indépendante) + swỉ.n=f + wnm.n=f (deux verbes coordonnés)

3. Plus le locuteur s'implique dans la locution, plus la phrase tend à être elliptique ; c'est le cas des phrases contenant un impératif, et plus encore des interjections, qui sont, selon TESNIÈRE, des « mots phrases » (pp. 94 à 99 d'Élements de syntaxe structurale (Tesnière 1966))

mais elle peut aussi se comprendre *il a bu, après avoir mangé*, le second verbe étant compris comme un circonstant du premier[4]. De telles phrases peuvent être extrêmement longues (ce n'est pas le cas dans notre corpus). Toute ambiguïté de rattachement peut donc se répercuter de manière « catastrophique », au sens de la théorie du chaos, sur une partie du texte à analyser, c'est-à-dire que la longueur des constructions, conjuguée avec l'incertitude entre constructions circonstancielles et constructions continuatives, peut engendrer un nombre exponentiel d'analyses.

Il n'est pas forcément facile de distinguer récit et discours de façon interne, la séparation des deux pose des problèmes, car les marques formelles de début et surtout de fin de discours sont assez lâches en égyptien. Le papyrus Westcar utilise de manière assez constante (à une exception près) le verbe « *ḏd* », « dire », pour commencer une réplique, mais le seul indice qu'une réplique est terminée est l'introduction de la réplique suivante. Or, et c'est là qu'intervient la structure de notre conte, un discours peut très bien contenir un récit, qui lui-même peut comporter un dialogue...

Les connaissances nécessaires pour résoudre ce genre de problèmes dépassent largement le cadre purement syntaxique, et, croyons-nous, les possibilités actuelles d'un analyseur automatique humainement réalisable.

4.4.3 Seconde grammaire

La seconde grammaire est l'aboutissement d'une évolution de la première. Nous l'avons réalisée, d'une part, en généralisant les constructions, et, d'autre part, en essayant de limiter les ambiguïtés. Nous avons regroupé et supprimé les analyses génératrices d'ambiguïtés, l'idée étant de faire une grammaire de ce qui est observé, quitte à perdre des informations syntaxiques. Si nous voulons aller plus loin, il faudra structurer le résultat de cette analyse-là. On se rapproche un peu, en ce qui concerne les formes du verbe, de la description de la langue que Gardiner a fournie, à l'usage des débutants, dans les premiers chapitres de sa grammaire (Gardiner 1963) (voir 1.6.3).

Par exemple, dans le groupe nominal, une séquence de substantifs peut être un génitif direct ou une coordination. La grammaire de départ exhibait les deux solutions, la seconde grammaire se contentera de noter que nous avons là une séquence de substantifs. Au lieu d'avoir une règle du type :

```
(24) gns
     ==>
             gn, &gns([u::gn_suivant = d]).
```

4. Ce type de construction, où un circonstant *logique* est indiqué par simple juxtaposition, existe aussi en français ; on appelle ce phénomène *parataxe*.

pour marquer la coordination de groupes nominaux (équivalent au français : GN1 et GN2), et une autre du type :

(25)
```
gn
==>
        nom([u=d]), &gn([u::complement_de_nom=d]).
```

pour marquer le génitif (équivalent au français : GN1 de GN2), nous utiliserons une seule et unique construction qui n'aura d'autre fonction que de préciser que deux noms sont en séquence, puisqu'en Égyptien, les deux peuvent être notés de la même façon (c'est aussi vrai pour de nombreuses langues, dont celles du groupe sémitiques).

À un niveau syntaxique plus élevé, nous avons considéré tout groupe à valeur nominale, y compris les groupes désignant une action, en particulier les propositions bâties sur un infinitif, comme des noms.

La généralité ainsi acquise donne parfois naissance à des ambiguïtés artificielles, mais nous réserve aussi quelques agréables surprises. En analysant avec notre système la phrase

Exemple 28 (p. 12, l. 12.)

translittération :	*ỉw=ỉ*	*r*	*šmt*	*ḏd*	*st*	*n*	*ḥm*	*n*
littéralement :	*je*	*vers*	*aller*	*dire*	*cela*	*à*	*la majesté*	*du*

nswt-bỉty	*ḫwfw*	*mꜣꜥ-ḫrw*
roi de Haute et Basse Égypte	*Khéops*	*juste de voix*

: Je vais aller le dire à la Majesté du Roi de Haute et Basse Égypte Khéops, justifié

dans laquelle la préposition *r* est, d'une façon assez inhabituelle, mise en facteur devant deux verbes[5], nous n'avions absolument pas prévu ce cas dans notre grammaire, mais celle-ci nous fournit comme analyse possible la coordination des deux verbes.

Cette méthode a cependant des limites : la suppression de la structuration là où elle est source de conflit n'est pas toujours possible.

Ainsi, les propositions relatives interviennent dans les groupes nominaux. Or, elles peuvent comporter des constructions arbitrairement complexes. Si, par exemple, nous avons décidé de ne pas résoudre le problème des attachements prépositionnels, parce qu'il n'y a pas de bonne méthode globale pour ce faire, ceux-ci ne cessent pas pour autant d'être générateurs d'ambiguïtés. En

5. Il existe une autre façon d'interpréter grammaticalement la phrase, au prix d'une correction mineure du texte, mais la complexité de la tâche dans le cas de l'égyptien est telle qu'il nous semble prématuré d'envisager son automatisation.

effet, un groupe prépositionnel suivant une proposition relative pourra être rattaché, soit à la relative, soit aux constructions syntaxiques qui englobent la relative.

Il serait alors tentant d'utiliser le fait que les compléments adverbiaux aient tendance à se trouver en fin de phrase pour éviter totalement de devoir résoudre le problème des attachements. Nous poserions par exemple que tous les groupes prépositionnels sont rattachés à la construction la plus haute, quitte à reconsidérer ce postulat *a posteriori*.

Mais dans une phrase phrase comme :

Exemple 29 (p. 4, l. 3.)

translittération :	*wn.ỉn*	*ḫry-ḥbd*	*ḥry-tp*	*wbꜣ-ỉnr*
littéralement :	*Et*	*le prètre lecteur*	*en chef*	*Oubainer*

ḥr	*wḥm*	*mdt*	*tn*	*ỉr.n*	*pꜣ*	*nḏs*
sur	*répéter*	*affaire*	*cette*	*qu'avait commise*	*ce*	*roturier*

m	*pr=f*	*ḥnꜥ*	*tꜣy=f*	*ḥmt*	*n*	*ḥm*	*n*
dans	*maison sienne*	*avec*	*sa*	*femme*	*à*	*la majesté*	*du*

nswt-bỉty	*nb-kꜣ*	*mꜣꜥ-ḫrw*
roi de Haute et Basse Égypte	*Nebka*	*justifié*

Et le prêtre lecteur en chef Ouba-iner relata à la majesté du Roi de Haute et Basse Égypte Nebka ce délit qu'avait commis le roturier dans sa maison avec sa femme

la proposition relative « *ỉr.n pꜣ nḏs m pr=f ḥnꜥ tꜣy=f ḥmt* » qualifie « *mdt tn* », et est insérée *avant* le complément d'objet indirect du verbe de la phrase. Or elle contient des compléments adverbiaux.

La simplification envisagée devient alors problématique.

De même, les séquences de noms peuvent correspondre à un unique syntagme nominal par coordination, mais aussi, dans de nombreux cas, au sujet d'une phrase verbale suivi du complément d'objet direct de ladite phrase. Le conflit n'est pas alors localisé dans le syntagme nominal, il influe sur le schéma actantiel du verbe, et il est difficile d'admettre qu'une structure aussi fondamentale que celui-ci reste dans le flou. Nous sommes donc obligés de conserver une distinction génératrice d'ambiguïtés.

4.4.4 Commentaires sur les limitations de ces grammaires

Nous avons conçu notre seconde grammaire en considérant que mieux valait disposer de peu d'informations, faiblement ambiguës que de beaucoup d'information très ambiguë. Grâce à cela, la seconde grammaire est supérieure à la première du point de vue de l'extensibilité et de la généralité.

En effet, la précision *a priori* de la première grammaire la rend difficile à étendre. En insérant une nouvelle règle, on peut créer des ambiguïtés non désirées dans certains cas. Essayer de restreindre la portée des règles pour pallier ce problème peut inversement rendre trop faible la couverture de la grammaire, qui aura alors tendance à rester trop près du texte.

Il est difficile de chiffrer, en terme de succès, sur un corpus réduit, les différences entre les deux grammaires ; les deux couvrent en effet notre corpus. Il est par contre possible de les comparer d'un point de vue « génie logiciel ».

certains problèmes subsistent, dûs en partie au paradigme d'analyse lui-même. le formalisme que nous développerons pour résoudre ces difficultés sera probablement meilleur que notre formalisme original. Nous allons donc nous intéresser aux défauts communs aux deux grammaires.

Les constructions semi-figées

L'un des grands problèmes que le TAL a rencontré ces dernières années concerne les constructions figées ou semi-figées. D'apparence anodine, et jadis négligé, ce problème a des répercussions pratiques importantes, et des racines théoriques profondes (Laporte 1988).

D'un point de vue utilitaire, la reconnaissance des groupes nominaux figés (Émile Benvéniste 1974, 2, pp. 103–112) est en particulier un problème important en lexicographie appliquée, par exemple pour l'indexation automatique. Se pose alors le problème de l'établissement d'une nomenclature, à partir de termes qui, parce qu'ils sont des termes de spécialité, ne sont pas répertoriés dans les ouvrages généraux, mais qui, parce que les techniques évoluent rapidement, ne sont pas non plus toujours répertoriés dans des ouvrages spécialisés.

En pareil cas, il est parfois possible de distinguer des groupes figés grâce à leur construction grammaticale. En général, ce type d'approche ne fournit que des groupes *susceptibles* d'être des groupes figés ; une vérification s'impose, soit sur des critères statistiques, soit par décision d'un intervenant humain. Il importe de préciser, et le nombre important de définitions existantes du phénomène le prouve, qu'une part d'arbitraire entre dans le jugement qui donne pour figée telle ou telle expression.

Le figement se conçoit mieux en diachronie, et c'est là son sens profond : c'est par là en effet qu'il est fécond, car porteur de nouveautés *grammaticales*. Du quantitatif (une forme est plus ou moins figée), on passe au qualitatif (une forme grammaticale nouvelle apparaît). On passe ainsi du latin *ego amo* (*moi, j'aime*) où *ego* sert à mettre en relief le sujet, au français *j'aime* où *j* n'est que la marque de la personne.

Certaines constructions figées correspondent à des états de langue antérieurs ; elles ne sont plus décomposables en fonction de la grammaire synchronique. Un certain nombre de constructions posent cependant un problème, car elles restent analysables. Leur caractère figé devrait nous indiquer quelle est la bonne analyse, mais en toute généralité, d'un point de vue structurel, d'autres analyses sont possibles. Ce problème est relativement facile à régler pour le système : sa vocation première est de proposer des analyses, non d'être un modèle de la langue. En conséquence, il est raisonnable de trancher systématiquement en faveur de l'expression figée. Un système comme le nôtre, dont le but est de permettre à un opérateur humain de gérer plus facilement une base de données, peut se permettre un tel comportement. Les cas d'erreur seront rares, et il sera loisible de les corriger manuellement.

La nominalisation

Le traitement de la nominalisation est particulièrement délicat : en égyptien, à peu près tout syntagme peut être nominalisé, souvent sans marque apparente. Les constructions ayant vocation à être des groupes nominaux sont donc fréquentes. Faut-il, pour chaque groupe, gérer à part l'hypothèse de sa nominalisation, ou bien mettre ce phénomène au rang des constructions de haut niveau ?

Un syntagme verbal nominalisé, par exemple autour d'un infinitif, n'a pas exactement la même distribution qu'un groupe nominal. La question est de savoir si, d'une façon ou d'une autre, on doit intégrer la règle

```
(26) gn ==> groupe_infinitif.
```

dans la grammaire, ou détailler, cas par cas, les différents emplois possibles de `groupe_infinitif`. Par exemple, l'emploi de l'infinitif comme COD est limité à certains verbes, de même que seules certaines prépositions peuvent l'admettre comme régime[6]. Décider ou non de traiter explicitement ces cas a une grande influence sur la forme de la grammaire.

Le traitement explicite a été effectué dans la première grammaire. Il a des avantages et des inconvénients. Ce traitement interdit des hypothèses qui se révèlent absurdes à l'examen.

Ainsi, la coordination entre un groupe nominal et un infinitif n'est pas envisagée, ce qui est souhaitable en première analyse. Dans le cas contraire, on introduit une ambiguïté dans l'analyse des groupes *infinitif + GN* . Un

6. *terme régi par une construction, spécialement par un verbe ou une préposition* (Marouzeau 1951).

tel groupe doit *a priori* être interprété comme :

infinitif

second actant

groupe nominal

Donner aux groupes constitués autour de l'infinitif un statut de *GN* à part entière crée une interprétation concurrente, qui est :

séquence

infinitif *groupe nominal*

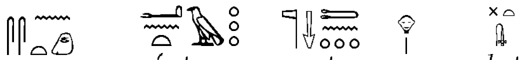
La question se pose cependant de savoir jusqu'à quel point la dernière interprétation est irrecevable. De fait, on peut la trouver réalisée dans certains textes :

Exemple 30 *STATUE DE HAT-RA* 1–4

di͗=s | *prr.t nb.t ḥr wḏḥw=s m ẖr.t hrw n.t rʿ nb*

puisse-t-elle donner | *ce qui sort tout sur ses autels chaque jour*



ssn.t ʿntyw snṯr ḥr ḫ.t

de respirer la myrrhe l'encens sur le brasier

ḫft ḥtp ḥm.t=s m-ḫt=s

selon les offrandes de sa Majesté après elle

n k3 n *i͗t-nṯr n tm*

pour le kA du père divin d'Atoum (Hat-ka)

traduction : puisse-t-elle donner tout ce qui est offert journellement sur son autel, (permettre de) sentir la myrrhe et l'encens sur son brasier lorsque sa Représentation se repaît, après Elle, pour le ka du père divin d'Atoum ... (Hat-ra)

Le verbe *di͗,* « donner, placer, causer, permettre » a ici pour complément aussi bien *prr.t*, forme substantive du verbe qui signifie « ce qui sort », que l'infinitif *ssn.t*, « respirer ». Bien entendu, ce genre de phénomène est ici favorisé par la longueur de la phrase, qui distend les liens syntaxiques. Ce cas montre néanmoins que coder des restrictions aussi fortes dans la base de règles est s'exposer à laisser de côté des cas qui sont attestés et réguliers, quoique rares.

Multiplier les règles pour tenir compte de multiples cas n'est pas non plus une bonne chose en soi. Cela nuit à la structure de la base de règles, et donc à sa généralité, ce que montre d'ailleurs l'exemple précédent ; c'est aussi un générateur potentiel d'ambiguïté.

Les séquences

La coordination est un générateur puissant d'ambiguïté, en égyptien[7] peut-être plus encore que dans d'autres langues. En effet, la conjonction de syntagmes peut se faire en égyptien par simple juxtaposition. L'ambiguïté est alors multiple : dans une langue qui fait systématiquement usage de conjonction de coordination, il est déjà difficile de déterminer la portée de ces joncteurs ; en égyptien, l'analyse est compliquée par le fait que le génitif peut s'exprimer lui aussi par juxtaposition.

Donnons un exemple en français pour fixer les idées ; une phrase comportant une ellipse du verbe comme :

> * Marie a mangé la charlotte, Pierre, la tarte.

peut s'interpréter syntaxiquement de plusieurs manières :

> [[S Marie] [GV a mangé [COD la charlotte]]] [[S Pierre,] [GV ∅ [COD la tarte]]]

mais aussi

> [S Marie] [GV a mangé [COD la charlotte, Pierre, la tarte]]

Une meilleure ponctuation eût levé l'ambiguïté, mais justement, dans le cas qui nous occupe, nous ne disposons pas de ponctuation. Dans nos textes, un groupe de la forme *Verbe transitif + Nom1 + Nom2* peut s'interpréter :

1. [GV Verbe [Sujet Nom1] [COD Nom2]]
2. [GV Verbe [Sujet [Coordonné Nom1 Nom2]]]
3. [GV Verbe [Sujet Nom1 [génitif Nom2]]]

Soit trois possibilités pour un cas relativement simple. On trouvera dans la *syntaxe générale* d'A. MARTINET (Martinet 1985), 7.2.2, quelques exemples intéressants sur les problèmes de la coordination.

7. ... et dans nombre de langues sémitiques, telles l'hébreu et l'arabe.

L'importance du contexte

Les différents registres existants influent fortement sur l'analyse. La trame narrative est, dans notre texte, extrêmement simple ; sa macro-syntaxe est assez bien définie. En comparaison, les dialogues sont beaucoup plus complexes quand à leur variabilité syntaxique. De fait, dans un dialogue, tout syntagme peut former un énoncé.

On peut lever certaines ambiguïtés à condition de disposer du contexte de la phrase. Le problème est que le contexte lui-même doit être déterminé, ce qui est difficile à réaliser dans le cadre que nous avons fixé. Nous suggérerons plus loin quelques pistes pour réaliser cette tâche.

Généralité et précision

La première grammaire est difficile à étendre, et ne couvre vraiment que la partie du langage sur laquelle on la teste. Dès que le texte concerné prend des libertés avec une syntaxe rigide, cela nous pose des problèmes. En particulier, pour en revenir aux groupes nominaux, les vocatifs forment un énoncé en eux-mêmes. Nous avons espéré, dans un premier temps, que ceux-ci se situeraient toujours en début de réplique. Malheureusement, ce n'est pas le cas. Ces constructions peuvent se trouver intercalées entre deux propositions d'une réplique. Les détecter de manière absolue demanderait de pouvoir déterminer que l'on a affaire à un individu, ce qui n'est pas forcément facile, et, d'autre part que l'on s'adresse à cet individu, bref de disposer, d'une part, d'importantes informations syntaxiques, sémantiques, et pragmatiques, et d'autre part, d'une formalisation du déroulement des dialogues. Si l'étude du dialogue est un des grands thèmes du TAL (Sabah 1989, pp. 283–316), elle est majoritairement menée en utilisant des contextes précis, par exemples dans des systèmes pour la réservation de billets d'avion, ou d'aide à l'apprentissage de notions de théorie du signal (Dessalles 1993), où il est possible d'extraire, soit d'une étude de corpus, soit d'un modèle du domaine, des régularités qui permettent au système de fonctionner essentiellement en reconnaissant une situation attendue. C'est sans aucun doute plus difficile dans un conte, où les dialogues ne sont, au reste, pas « vrais », en ce sens qu'ils participent de la narration.

Certaines formes ont des emplois très marqués selon les registres : dans une narration, un infinitif, forme susceptible d'occuper une place nominale dans un texte, peut être utilisé comme un tour narratif autonome.

Enfin, les informations dont on dispose sur le texte jouent un très grand rôle :

Exemple 31 (p. 2, l. 4.)

ꜥḥꜥ.n	*ḏd.n*	*pꜣ*	*nḏs*	*n*	*tꜣ*	*ḥmt*	*wbꜣ-ỉnr*
il arriva alors que	*dit*	*ce*	*roturier*	*à*	*la*	*femme*	*d'Ouba-Iner*

Alors ce roturier dit à la femme d'Ouba-iner

Si ce texte est analysé par notre première grammaire en précisant qu'il s'agit d'une proposition, on obtient 61 analyses, qui correspondent à des découpages différents pour les groupes nominaux. Si nous supprimons cette information, nous obtenons 84 analyses, dont certaine coupent mal notre construction. La seconde grammaire fournit respectivement 32 et 45 analyses.

Sur le grand nombre d'analyses fournies par le système pour la première grammaire, le fait de ne considérer que les analyses spécifiques au récit réduit leur nombre à 4 et 6 occurrences. L'information contextuelle permet donc de réduire considérablement l'ambiguïté.

Plusieurs analyses erronées, dans ce cas précis, proviennent de ce que *ḏd.n* peut être considéré comme la tête d'une proposition relative, qui signifierait alors quelque chose comme «*le que a dit le roturier*». Le problème, ou plutôt, la solution, est ici qu'en bonne grammaire, l'expression ne peut signifier «*ce qu'a dit le roturier*», puisqu'en égyptien une telle forme, de sens neutre, serait exprimée par un féminin, et que toute autre interprétation demanderait la présence, dans la relative, d'un pronom rappelant l'antécédent.

Malheureusement, ce critère ne nous semble, ni très efficace dans le cas général, ni surtout très facile à implémenter. Il est certes tout à fait faisable de chercher un pronom, mais le texte même nous met en garde contre nos critères, puisque dans l'exemple 29, l'accord entre l'antécédent d'une forme (*ỉr.n*) et la forme n'est pas fait, alors qu'il est grammaticalement nécessaire.

Ce qui signifie qu'il est extrêmement difficile d'être sûr de tels critères.

Dépendances à distance

Le mode de représentation choisi pour la grammaire est déficient lorsqu'il s'agit de représenter des constructions coupées. Par exemple, dans les phrases nominales, les groupes nominaux peuvent être coupé par le pronom *pw,* qui sert à marquer la prédication nominale.

Exemple 32 (p. 9, l. 9.)

translittération :	$\underset{.}{h}mt$	$w^{c}b$	pw	n	r^{c}	nb	$s\mathit{3}\underline{h}bw$
littéralement :	*l'épouse*	*(d'un) prêtre*	*C'est*	*de*	*Rê*	*Seigneur*	*de Sakhebou*

C'est l'épouse d'un prêtre de Rê, seigneur de Sakhebou.

Faut-il tenir compte de ce phénomène dans la description des groupes nominaux eux-mêmes, ou définir une classe à part de «groupe nominal prédicat» ? Dans un cas, on enchevêtre des descriptions de phénomènes différents, ce qui n'est guère satisfaisant du point de vue de la modularité. Dans le second cas, l'on dédouble nombre d'informations.

Conclusion

Nous constatons que notre système génère énormément d'hypothèses pour les groupes nominaux, et que dans les constructions qu'il propose, beaucoup comprennent des groupes nominaux complexes. Or, une analyse de notre corpus révèle que la grande majorité des groupes nominaux *réellement observés* sont simples.

De plus, l'on rencontre nombre d'expressions figées, ou semi-figées, en particulier lorsqu'il s'agit de nom propres accompagnés de titres.

4.5 Extensions

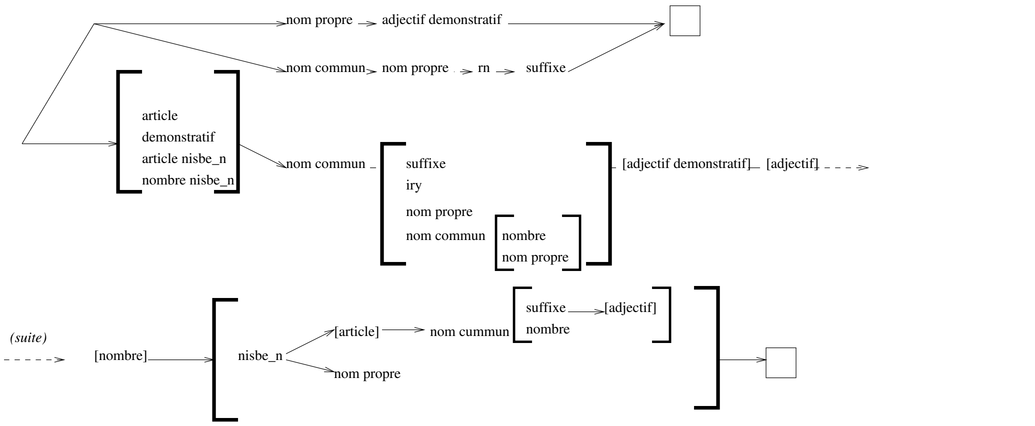
Il s'agit donc maintenant, à partir des limitations constatées dans notre analyseur, de proposer un modèle qui permettra de les surmonter. Nous commencerons par décrire quelques approches pertinentes pour notre travail ; nous proposerons ensuite une méthode de représentation des constructions fréquentes par des automates finis, pour terminer en envisageant une intégration de cette méthode au corps de notre analyse.

4.5.1 Modèle proposé

Extension par des expressions régulières

Les problèmes rencontrés concernant tout particulièrement le statut des groupes nominaux, nous avons eu l'idée de commencer une analyse quantitative des groupes nominaux observés dans le texte. Une certaine régularité s'en dégageant, nous avons écrit un automate fini reconnaissant les plus fréquents (figure 4.5)

FIG. **4.5** – Expression régulière reconnaissant les groupes nominaux les plus fréquents



Extraction de groupes nominaux

À titre d'expérience, utilisons cette expression pour extraire des groupes nominaux. Nous utiliserons l'algorithme : *si une expression régulière est reconnue, le signaler, et avancer à la fin du texte reconnu, sinon, avancer d'un mot.* Nous obtenons, à partir du texte :

Exemple 33 (p. 6, l. 18.)

ỉmỉ	*ḏ*	*=tw*	*mꜣˁt*	*1000*	*ḥnḳt*	*ds*	*100*	*ỉwꜣ*
faites	*que donne*	*on*	*pain*	*1000*	*bière*	*cruche*	*100*	*bœuf*

1	*snṯr*	*pꜣd*	*2*	*n*	*nswt-bỉty*	*snfrw*
1	*encens*	*mesure*	*2*	*à*	*Roi de Haute et Basse Égypte*	*Snefrou*

mꜣˁ-ẖrw	*ḥnˁ*	*rdt*	*ḏ*	*=tw*	*šns*	*1*	*ḥnḳt*	*ḏwỉw*	*1*
juste de voix	*et*	*faire*	*que donne*	*on*	*pain*	*1*	*bière*	*mesure*	*1*

snṯr	*pꜣd*	*1*	*n*	*ẖry-ḥbd*	*ḥry-tp*	*sš*	*mḏꜣt*
encens	*mesure*	*1*	*à*	*prêtre-lecteur*	*en chef*	*scribe*	*des archives*

ḏꜣḏꜣ-m-ˁnẖ
Djadjaemankh

l'analyse :

> *ỉmỉ ḏ =tw [mꜣˁt,1000] [ḥnḳt,ds,100] [ỉwꜣ,1] [snṯr,pꜣd,2] n [nswt-bỉty,snfrw,mꜣˁ-ẖrw] ḥnˁ rdt ḏ =tw [šns,1][ḥnḳt,ḏwỉw,1] [snṯr,pꜣd,1] n [ẖry-ḥbd,ḥry-tp] [sš,mḏꜣt,ḏꜣḏꜣ-m-ˁnẖ]*

ce qui est un excellent découpage. Le point important est que dans ce passage, une ambiguïté est normalement créée du fait que les nombres peuvent se construire de plusieurs façons, et que notre méthode résout correctement ladite ambiguïté.

Malheureusement, notre expérience de découpage à partir d'automates finis nous montre que cette méthode n'est pas une panacée : si la construction « A *n* B », signifiant « A de B », est reconnue, alors on a un bon découpage pour

ỉst rf wn [šspt] m [pꜣ,š,n,wbꜣ-ỉnr]

mais il devient mauvais pour

ꜥḥꜥ.n ḏd.n [pꜣ,nḏs,n,tꜣ,ḥmt][wbꜣ-ỉnr]

En fait, dès que notre expression reconnaît des constructions peu fréquentes dans notre corpus, les ambiguïtés se multiplient. « Peu fréquent » ne signifie pas « très rare » ; on constate simplement sur le texte que la majorité des compléments de noms sont simples.

La méthode de découpage proposée induit parfois de mauvaises coupures de phrase, et elle est sensible à l'ambiguïté lexicale. Par exemple, *nb* est souvent gênant. Il peut être l'adjectif «*tous*», mais aussi le substantif «*seigneur*». Cela pose parfois des problèmes aux traducteurs humains eux-mêmes. Soit, par exemple l'expression:

nṯrw	*nbw*	*n*	*wꜣst*
Les dieux	*tous? seigneurs?*	*de*	*Thèbes*

elle se traduira par :

Tous les dieux de Thèbes ou *Les dieux seigneurs de Thèbes*

Dans ce dernier cas, la structure syntaxique locale est la seule à changer. Dans le fragment de phrase :

nfrwt	*nbt*	*nt*	*ẖnw*	*ꜥḥ*	*=k*	*ỉb*	*n*
les beautés	*toutes*	*de*	*l'intérieur*	*(de) palais*	*tien*	*le cœur*	*de*

ḥm	*=k*	*r*	*ḳbb*
majesté	*tienne*	*vers*	*être rafraîchi*

(Équipe-toi un bateau chargé de) toutes les beautés de ton palais, et le cœur de ta majesté sera rafraîchi

Le découpage devrait être, dans l'absolu (les groupes nominaux étant indiqués entre crochets) :

[nfrwt,nbt,nt,ẖnw,ꜥḥ,=k][ỉb,n,ḥm,=k] r ḳbb

Si notre système se trompe et prend *nbt* pour un nom, il proposera le découpage suivant :

[nfrwt,nbt] nt [ẖnw,ʿḥ,=k][i͗b,n,ḥm,=k] r ḳbb

Au reste, comme ce groupe nominal n'est de toute façon pas analysable par notre automate, il conduit à une mauvaise segmentation quand il est correctement lemmatisé :

*[nfrwt,nbt/*adjectif*,nt,ẖnw][ʿḥ,=k,i͗b] n [ḥm,=k] r ḳbb*

Un certain nombre de problèmes surgissent donc, et nous interdisent d'utiliser nos expressions régulières telles quelles pour une analyse préliminaire du corpus. Le premier problème est que, si l'intérêt des expressions régulières est qu'elles permettent une analyse en temps linéaire d'un texte, ce n'est vrai que d'un texte sans ambiguïté lexicale. Dans le cas présent, on est obligé de se ménager la possibilité de revenir en arrière, et la complexité dans le pire cas redevient exponentielle. Cette dernière difficulté n'est pas rédhibitoire, nous le verrons, car l'exponentielle a pour facteur la longueur du texte *reconnu* par l'expression.

4.5.2 Couplage des automates et des grammaires

En revanche, ce système peut être utilisé pour choisir une analyse parmi plusieurs, ou pour guider l'analyse grammaticale. On peut envisager de l'insérer dans le cours de l'analyse, à plusieurs endroits :

- soit lors de la construction de la charte, auquel cas elle proposerait des liens considérés comme plus sûrs que ceux provenant de l'analyse par grammaire hors-contexte ;
- soit, *a posteriori,* lors de la construction des analyses à partir de la charte.

Dans le premier cas, on peut envisager la méthode suivante : on sélectionne un certain nombre de groupes validés par les expressions régulières, et on les utilise comme s'il s'agissait d'entrées lexicales dans l'algorithme d'analyse par charte. Bien évidemment, il faudra au pire lancer l'analyse complète par charte $\mathcal{O}(\mathcal{P}(E))$ fois, où E est l'ensemble des analyses proposées par les expressions régulières. En effet, rien ne garantit, en théorie, que l'analyse ainsi trouvée satisfera les contraintes associées aux règles, une partie de ces contraintes n'étant d'ailleurs évaluée que lors de la seconde phase de l'analyse.

Mais l'expérience montre que le nombre de groupes identifiés par les automates finis est relativement restreint, ce qui rend la méthode viable.

Dans cas d'un usage des expressions régulières *a posteriori*, nous les utiliserons pour choisir les analyses les plus conformes possibles à la description de l'attendu. Dans le cadre de l'exemple 33, la bonne construction est celle qui comporte la plus grande proportion de groupes nominaux correspondant à nos motifs. Néanmoins, si cette approche est tout à fait intéressante du point de vue des résultats qu'elle fournit, elle pose un grand problème : il n'est possible de faire un choix qu'en considérant l'ensemble des analyses possibles. Or le nombre d'analyses est exponentiel. Il faut donc songer à le réduire. Une méthode pour ce faire serait de trouver directement, lors de la phase de reconstruction de l'analyse, les interprétations qui utilisent le plus nos automates. En fait, cette méthode différerait peu de la précédente quand à son fonctionnement. Nous nous en tiendrons donc à une analyse *a priori ;* cela ne nous empêchera d'ailleurs pas de tenter certaines optimisations.

Analyse guidée par l'usage

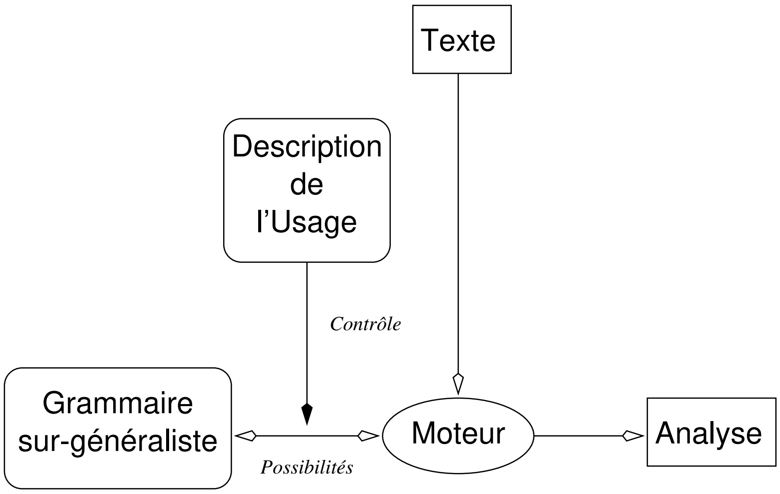
L'analyse telle qu'elle a été présentée ici utilise une grammaire et modélise la langue « possible. » Cette attitude n'est tenable que jusqu'à un certain degré ; en l'occurence, elle est fort loin des us et coutumes des philologues. Ceux-ci ont de bonnes raisons de s'en méfier. Confronté à une langue dont nous ne pouvons connaître que les performances, au sens chomskien du terme, la recherche d'une compréhension de la langue impose une certaine humilité devant le texte.

Bien entendu, ce postulat doit être nuancé. L'égyptien n'est pas un langage formel, mais il n'a pas non plus le privilège d'avoir été écrit uniquement par des scribes d'élite. Sir Alan GARDINER ne disait-il pas du scribe KENHERKHOPESHEF, à l'écriture désastreuse, qu'il avait au moins l'excuse de l'incompétence ? C'est une chose de constater que les textes peuvent contenir des erreurs ; c'en est une autre de décider qu'à un endroit précis le texte est erroné. Il n'en demeure pas moins que c'est ici le document qui règne. Il est donc souhaitable que l'analyseur automatique se plie dans la mesure du possible aux particularités et irrégularités constatées.

Or, le système d'expressions régulières que nous avons évoqué va dans ce sens. Foncièrement, nous l'avons utilisé pour représenter de manière compacte un certain nombre de fragments de texte. On peut espérer obtenir ainsi une couverture raisonnable des constructions fréquentes. En accordant une certaine confiance aux constructions détectées, nous pourrons, en amont, guider l'analyse, et, en aval, choisir une solution plausible, en ceci qu'elle suivra la loi du moindre effort.

Le système, à ce stade, a l'avantage « esthétique » de ne pas introduire de représentations numériques sans justification théorique solide. Nous re-

FIG. **4.6** – Modèle de l'Analyse
h



présentons son principe de fonctionnement dans la figure 4.6.

Un certain nombre de restrictions s'imposent néanmoins. L'opérateur de répétition * des expressions régulières est bien trop puissant pour être intéressant ici. Nous l'éliminons donc, mais cela met en relief le problème posé par un certain nombre de constructions. En particulier, les énumérations ne sont pas rares dans les textes. Comme il est impossible de les borner *a priori*, nous avons donc choisi de les analyser *a posteriori*.

Dans cette optique, nous pouvons par exemple, lorsqu'une suite de groupes nominaux a été détectée, utiliser des critères de parallélisme des formes, critères coûteux en temps normal, mais raisonnables si leur usage est limité aux cas favorables.

Le système doit en outre être capable d'évaluer peu ou prou la qualité des résultats produits par les expressions régulières pour déterminer si celles-ci sont ou non utilisables.

Les expressions régulières sont d'un usage limité. Elles ne nous fourniront que des informations parcellaires. Tout au plus, grâce à la simplicité de ce formalisme, pourrons-nous mieux l'analyser, ce qui est plus difficile avec des systèmes formels plus expressifs. Il faudra cependant, dans un souci de couverture et de structuration, disposer de représentations plus approfondies de la langue. Nous utiliserons notre formalisme de départ, en partant des arcs proposés par les expressions régulières.

Chapitre 5

Description de l'analyseur

Sommaire

Nous avons donc développé, en partant de l'idée de représenter l'usage, une architecture simple, qui vise en premier lieu à réduire les ambiguïtés. Nous commencerons par la décrire, puis nous détaillerons le formalisme sous-jacent, pour finalement donner les algorithmes utilisés.

5.1 Architecture

L'architecture globale du système est décrite dans la figure 5.1. Nous en détaillons ci-dessous les composantes.

5.1.1 Segmentation en propositions

Nous essayons dans un premier temps de donner une segmentation raisonnable du texte en propositions, afin d'avoir des bases saines pour les traitements ultérieurs. Un découpage par frontières sûres, s'appuyant sur les marqueurs d'initialité qui existent dans la langue égyptienne fournit un premier découpage.

Ensuite, des schémas réguliers sont utilisé pour proposer des arcs correspondant à d'éventuelles propositions. Ces arcs sont validés par des heuristiques.

5.1.2 Construction des arcs préférés

Pour chacune des propositions (ou éventuellement suite de propositions) ainsi créées, le même système de schémas réguliers est appliqué afin de trouver des syntagmes nominaux, avec bien entendu des règles différentes. On dispose alors d'un certain nombre groupes, qui peuvent être incompatibles entre eux (i.e. se chevaucher). Nous les utiliserons pour bâtir les fondations de nos analyses subséquentes.

5.1.3 Détection de séquences

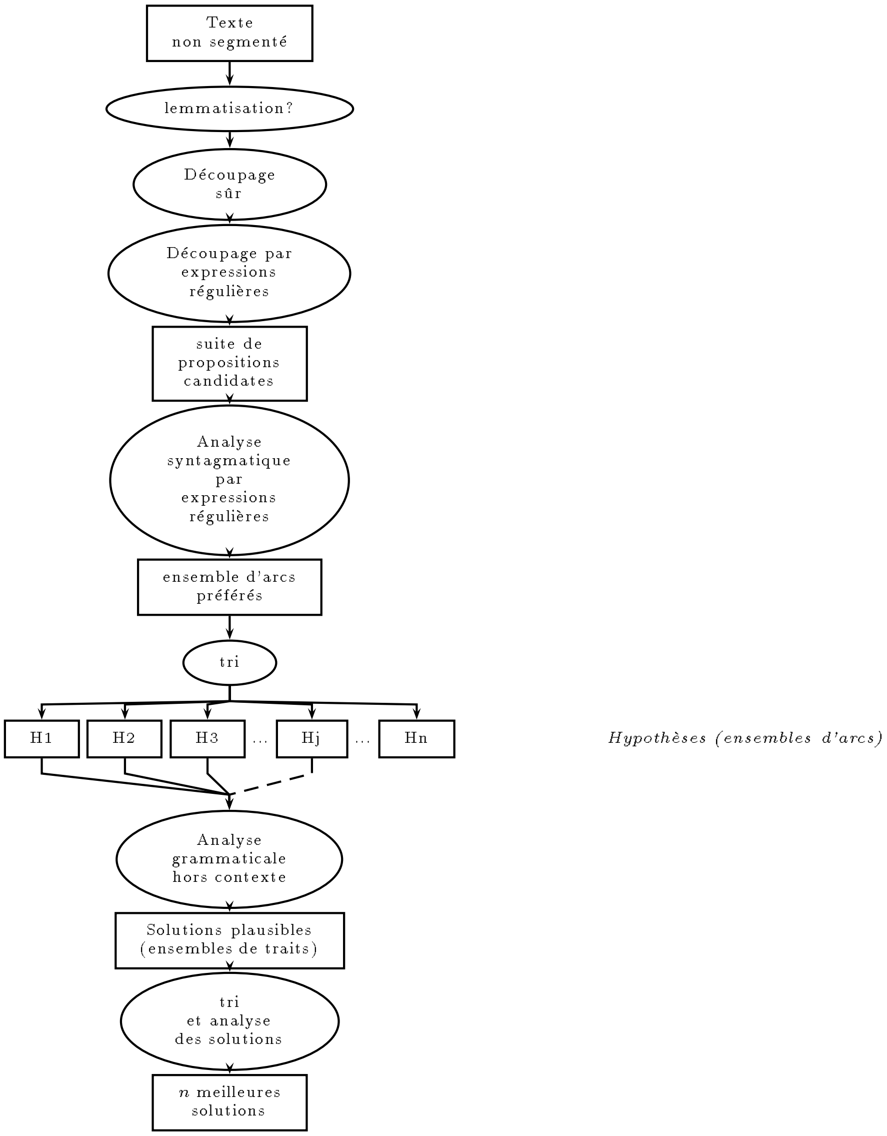
Une fois que nous avons déterminé un certain nombre de groupes syntagmatiques, nous pouvons essayer de détecter des groupes coordonnés.

5.1.4 Génération d'hypothèses

Les arcs générés par le système précédent ne sont pas forcément compatibles entre eux ; des hypothèses concurrentes peuvent apparaître. Il faut donc choisir, pour constituer l'amorce d'une analyse, un sous-ensemble de ce choix initial d'arcs. Ces sous-ensembles sont relativement peu nombreux, puisque le nombre d'arcs sélectionné est lui-même faible.

Chaque sous-ensemble correspond à un choix d'hypothèses sur le texte. Ils seront proposés en ordre décroissant de préférences aux étapes suivantes. Il importe donc de disposer d'un ordre sur les hypothèses qui choisisse les

FIG. **5.1** – Architecture générale du système



« meilleures » hypothèses, mais en assurant que des variations significatives seront prises en compte.

5.1.5 Analyse hors contexte

Les hypothèses générées sont ensuite fournies à un analyseur hors-contexte, qui construit des analyses complètes compatibles avec les arcs qui constituent les hypothèses. Quand le nombre d'analyses créées par le système atteint une limite arbitrairement fixée, le processus s'arrête sans considérer les hypothèses non encore analysées.

5.1.6 Post-structuration

Les phases précédentes proposent des analyses peu structurées, mais supposées correctes. Il reste à les parfaire, en travaillant sur l'ensemble de traits résultant de chaque analyse. C'est lors de cette phase que le système va utiliser le plus les informations lexicales fines dont il disposera.

5.2 Formalismes

5.2.1 Formalisme des motifs

Le formalisme employé pour décrire les motifs est simple. Un motif simple est, soit une catégorie syntaxique, un ensemble de traits, ou un nom de motif, les emplois récursifs de cette dernière possibilité étant bien entendu interdit.

Les opérateurs proposés sont « , » pour la mise en séquence, « ou » pour l'alternative, et « & » pour désigner un élément optionnel, et enfin les parenthèses. La répétition, usuellement notée « * », a été exclue du formalisme. Elle est en effet antinomique avec l'esprit dans lequel il est conçu, qui est d'enregistrer l'usage effectif des formes, en évitant autant que possible les généralisations dangereuses.

Un exemple partiel de la description des propositions explicite l'emploi des motifs :

```
proposition -->
                ( verbe, suffixe, gn)
        or      ( verbe, [cat::demonstratif, graphie::'pw'],
                  [graphie::'ir.n'], (gn or suffixe), &(preposition, gn))
        or      ( [cat::verbe, forme::sdm_n,graphie::'aHa.n'],
                  [cat::verbe, forme::sdm_n],
                  (suffixe or gn), &([cat::preposition, graphie::n],
```

```
                gn),
                &([cat::preposition], gn))
        or    ( [cat::preposition, graphie::xr],
                [cat::preposition, graphie::'m-xt'], gn, verbe)
        or    ( [cat::verbe, forme::sdm_in], gn).
```

5.2.2 Formalisme grammatical

Les formalismes lexicaux et grammaticaux utilisés ensuite sont ceux qui ont été définis précédemment : règles de production hors contexte augmentées de traits, pour la grammaire, et description lexicale à base de traits pour le lexique.

La description lexicale a l'avantage d'être utilisable, tant avec les motifs qu'avec les règles de production. Elle permet une certaine souplesse avec les premiers, et est pratiquement nécessaire aux secondes.

Les règles de productions hors contextes sont conservées parce qu'elles permettent une description structurée et large de la langue ; nous reprenons les méthodes utilisées pour écrire la seconde grammaire.

5.2.3 Structuration

Le formalisme utilisé pour la post-structuration doit permettre de manipuler des structures de traits.

5.3 Algorithmes

5.3.1 Traitement des motifs

Construction d'un automate fini déterministe

L'algorithme utilisé est une variante des algorithmes classiques (Aho, Sethi, et Ullman 1989). Le seul problème sérieux qui se pose est qu'un même mot peut convenir à plusieurs motifs simples. Par exemple,

```
[cat::verbe, forme::sdm_n,graphie::'aHa.n']
```

et

```
[cat::verbe, forme::sdm_n]
```

Lors de la création d'un automate déterministe, il importe d'en tenir compte. Nous modifions donc l'algorithme classique de construction d'automates finis déterministes pour l'adapter à notre cas. La méthode pour ce faire est la suivante :

Nous disposons d'un premier automate, non déterministe, A_0. Nous construisons un second automate A_1, qui est en fait un arbre du langage reconnu par A_0. Tout nœud de A_1 est un ensemble de nœuds de A_0. Soit un nœud $N \in A_1$ déjà construit dans notre automate déterministe. Soit $L = \{(a_i, b_i, l_i) \in A_0 | a_i \in N\}$; nous construisons les arcs sortants de N de la façon suivante : Soit L' l'ensemble des labels de L ; pour obtenir un label dans A_1, nous partitionnons L' en deux classes : l'arc ainsi labelé sera passable si et seulement si l'entrée est compatible avec tous les labels de la première classe, et avec aucun des labels de la seconde classe. Dans le pire cas, nous créons donc $2^{\mathrm{cardinal}(L')}$ arcs ; mais en fait, un arc ne peut être créé que si tous les labels de la première classe sont compatibles entre eux. L'observation des données nous apprend que ce cas est exceptionnel, ce qui nous permet d'avoir une taille de données et un temps de calcul acceptables. Un label dans A_1 sera composé de deux éléments : un ensemble d'attributs, correspondant à la valeur la plus restrictive des labels positifs, d'une part, et l'ensemble d'interprétations interdites, d'autre part. Nous noterons les labels de A_1 ainsi : (l, e) où l est un label au sens précédent du terme, et e est un ensemble de labels.

Le premier ensemble est obtenu simplement en faisant l'unification de tous les éléments positifs. Si l'unification échoue, alors le label ne peut exister. Le second ensemble peut être restreint. En effet, les labels négatifs qui ne peuvent s'unifier avec les éléments du premier ensemble apportent une information redondante.

D'autre part, il est possible que certains labels en subsument d'autres. $l_1 \Rightarrow l_2$, dans A_0, et si (l, e) est un label de A_1, alors :

1. on ne peut avoir $l_2 \in e$ et $l \Rightarrow l_1$; tout label ainsi construit peut donc être supprimé
2. si $(l_1, l_2) \in e^2$, on peut supprimer l_1

Soient par exemple les labels :

```
[cat::verbe, forme::sdm_n,graphie::'aHa.n']    (a)
[cat::verbe]                                    (b)
[cat::verbe, forme::sdm_in]                     (c)
```

ils donnent *a priori* naissance à 8 labels possibles ;

$a \wedge b \wedge c$ est toujours faux, à cause des valeurs de `forme` ;

$a \wedge \neg b \wedge c$ de même ;

$\neg a \wedge \neg b \wedge c$ comme $c \Rightarrow b$, le label peut être supprimé ;

$\neg a \wedge b \wedge c$ équivaut à $(c, \{a\})$;

$a \wedge b \wedge \neg c$ équivaut à $(a, \emptyset)$;

$a \wedge \neg b \wedge \neg c$ comme $a \Rightarrow b$, le label peut être supprimé ;

$\neg a \wedge \neg b \wedge \neg c$ $([], \{a, b, c\})$; ce label ne menant nulle part, il peut être supprimé ;

$\neg a \wedge b \wedge \neg c$ $(b, \{a, c\})$;

soit en fin de compte, trois labels. Si n_a, n_b et n_c désignent les nœuds de A_0 auquels conduisent les arcs étiquetés respectivement par a, b et c, nous avons donc dans A_1 trois nœuds :

le label $(c, \{a\})$	nous mène au nœud $\{n_b, n_c\}$
le label $(a, \emptyset)$	nous mène au nœud $\{n_a, n_b\}$
le label $(b, \{a, c\})$	nous mène au nœud $\{n_b\}$

L'automate fini ainsi obtenu est déterministe pour une séquence données d'interprétations lexicales. Il n'est pas déterministe vis-à-vis de l'ambiguïté trouvée dans le texte même. Il faut alors représenter ce texte sous forme d'automate finis, et construire l'intersection des deux langages.

Utilisation des arcs

Chaque arc proposé par le système d'expressions régulières doit être validé par une analyse hors contexte locale. Celle-ci ne pose que peu de problèmes, puisqu'elle porte sur une portion de texte limitée, et cherche à reconnaître une structure syntaxique simple. Cette analyse permet de disposer d'une structure de traits associée à l'arc.

Les arcs proposés par notre système sont, pour une interprétation donnée, les plus longs possibles.

D'autre part, une fois les arcs créés, le processus de détection des séquences de groupes syntaxiques est lancé. Il s'agit de détecter des structures telles que les titélatures[1] de personnages, et, en règle générale, les énumérations.

Une liste est typiquement constituée d'une suite de groupes nominaux *de même structure syntaxique ;* elle est souvent close par un groupe de structure différente, qui résume ou étend l'énumération. On se reportera aux exemples 10 p. 58, 30 p. 119 et 33 p. 124.

1. liste des titres et des noms d'une personne

La détection d'une énumération permet de lever beaucoup d'ambiguïtés ; en particulier, dans le cas d'une construction *Verbe + Sujet + Objet* elle permet de délimiter les frontières respectives du sujet et de l'objet ; d'autre part, elle permet de lever l'ambiguïté entre coordination et génitif (cf. page 4.4.3 et 4.4.4)

Enfin, dans le cas d'une ambiguïté entre nom et verbe, la détection de séquence permet de trancher en faveur du nom.

Les régularités n'étant pas absolues, il n'est pas question de détecter toutes les énumérations, mais seulement le plus grand nombre possible d'entre elles.

Lorsqu'une suite d'arcs représentant des groupes nominaux a été créée par la détection de motifs, une énumération doit être trouvée si les groupes en question ont même structure et/ou sont sémantiquement proches.

Ce dernier critère est très gênant, mais est difficile à contourner. L'exemple 10 : *louanges, chants, danses, cris, tout ce qui est fait pour un roi*, nous montre une succession de groupes réduits à des substantifs. Il est donc hors de question dans ce cas précis de se reposer sur la structure de ces groupes. Il y a en fait un marqueur explicite d'énumération dans cet exemple : c'est le « tout » de *tout ce qui est fait pour un roi*. Mais il permet seulement de soupçonner fortement la fin d'une liste ; le début est difficile à marquer.

Malgré notre réticence à faire rentrer la sémantique « par la fenêtre », l'observation du texte hiéroglyphique nous porte quelque peu (dans ce cas précis) à nous demander si un ensemble réduit de traits n'est pas envisageable ; considérons le texte :

ḥsy šmʿ ẖbt w3g ỉrr.wt nb.t n nswt

louanges chants danses cris (?) ce qui est fait tout pour un roi

On constate que, à l'exception du mot « danser », tous les mots de l'énumération se terminent par le signe 𓀁, qui désigne les actions effectuées par la bouche en général.

Ainsi, on pourrait utiliser comme système de traits le système même des déterminatifs, qui constitue un formalisme de représentation éprouvé (3000 ans d'usage!). Cependant, quelques tentatives opérées par des égyptologues dans le domaine de la classification des verbes en fonction de leurs déterminatifs se sont révélées peu fructueuses. De plus, cet usage n'est sans doute pas extensible aux textes plus tardifs, pour lesquels la recherche de la précision phonétique s'est faite au détriment des qualités iconiques du système (Winand 1990).

Il existe deux types de traits donnant un bon rendement pour l'analyse des énumérations : le trait « humain » et le trait « produit ». Les listes les plus fréquentes sont en effet soit des listes de personnel, soit des listes d'offrandes ou de rétribution.

Dans des cas structurellement plus favorables, on peut choisir comme énumération toute suite de groupes nominaux dont la plupart contiennent des nombres (en particulier ceux des extrémités).

5.3.2 Choix des hypothèses, génération des analyses

Quand les arcs provenant du système de motifs ont été construits, il s'agit d'en sélectionner un certain nombre afin de générer des hypothèses. Une hypothèse est constituée d'un ensemble d'arcs proposés par le système de motifs, qui ne se chevauchent pas.

Pour toute hypothèse envisagée, on initialise un agenda qui comprend, d'une part, les arcs constituant l'hypothèse, et d'autre part, les arcs lexicaux correspondant aux mots qui ne font pas partie d'un groupe couverts par une expression régulière. À partir d'une hypothèse donnée, le système peut alors procéder à une analyse hors contexte, et fournir une interprétation complète. Les questions qui se posent sont donc : d'une part, comment constituer les hypothèses, et d'autre part, comment éviter qu'une erreur dans le choix d'une hypothèse ne compromette le succès du système.

Constitution automatique des hypothèses

Chaque méthode de sélection des arcs implique une idée, exprimée ou non, de ce qu'est une bonne hypothèse. Cependant, une méthode qui semble *a priori* raisonnable peut avoir des effets pervers, en favorisant de manière pathologique tel ou tel type de choix.

La constitution des hypothèses n'est en outre pas indépendante des motifs utilisés. Si, par exemple, un mot isolé constitue un groupe nominal selon les motifs, et que le système ordonne les hypothèses selon le nombre de groupes reconnus, ou selon la proportions du texte expliquée par les motifs, on favorisera systématiquement la résolution d'ambiguïtés nom/verbe en faveur du nom, ce qui n'est évidemment pas souhaitable.

Le degré de couverture des phrases constitue néanmoins un critère intéressant. Si des motifs correspondant à des groupes verbaux sont prévus, le second système est viable.

Il est cependant souhaitable de tenir compte de la « qualité » des groupes identifiés. Celle-ci s'exprime par la complexité des groupes identifiés. Dans notre, cas, pour les groupes provenant d'un analyseur hors-contexte, plus ils sont est compliqués, moins ils sont bons. Par contre, dans le cas des motifs, la situation est inverse. La raison en est simple : quand un système de règles de production puissantes engendre une structure très complexe, et qu'il en existe une plus simple, c'est la seconde qui est préférable, parce que les deux interprétations ont une valeur globale : on a donc dû faire plus d'hypothèses dans le cas d'une analyse complexe. Dans le cas des motifs, le système représente à peu de choses près des groupes attestés et relativement fréquents. Choisir un groupe long plutôt qu'un qui soit court correspond alors simplement à une meilleure reconnaissance du texte, par rapport à ce qui était attendu.

En conséquence, le système doit tenir compte à la fois du pourcentage du texte expliqué par les motifs, et de la longueur des syntagmes reconnus par ceux-ci. Nous disposons d'un nombre pour mesurer le premier critère : $a = \frac{\text{longueur du texte reconnu}}{\text{longueur du texte}}$; étant donné une mesure adaptée du second, nous disposerons d'un nombre b. Plus grands seront a et b, meilleure sera l'hypothèse.

Il y a plusieurs manières de combiner a et b : l'ordre lexicographique d'une part (a est d'abord considéré seul ; à a égaux, on compare les b) ; diverses « normes » d'autre part : $a + b$, $\sqrt{a^2 + b^2}$, etc...

Les formules à bases de normes comparent des éléments incomparables *a priori*. Cela ne signifie pas qu'elles soient forcément inefficaces, moyennant des renormalisations empiriques. L'ordre lexicographique a l'avantage d'avoir un sens clair, même s'il est un peu trop brutal.

Un dernier type de combinaison est le produit ab (ou la moyenne géométrique, qui revient au même) ; il a l'avantage de ne pas demander de renormalisation.

Le calcul de b peut s'effectuer, soit en considérant la longueur moyenne des groupes, l'inverse du nombre de groupes, ou la taille du groupe le plus long.

La dernière méthode rend bien compte de ce que les groupes courts sont majoritaires, mais que la présence d'un groupe long est un bon signal de qualité pour l'analyse. Elle a par contre le défaut d'être incapable de distinguer deux hypothèses sur la longueur des groupes restants. Une formule du type de $\sum_{i=1}^{n} \frac{l_i}{2^i}$ où les l_i sont les longueurs des groupes, ordonnées par ordre décroissant, prend en compte ce phénomène, en assurant toujours la prééminence de la chaîne la plus longue.

Nous essaierons donc, pour les valeurs suivantes de b :

$$\begin{aligned} b &= \sum_{i=1}^{n} \frac{l_i}{2^i} \\ b &= \sup l_i \end{aligned}$$

les combinaisons suivantes : valeurs croissantes de ab et ordre lexicographique.

Génération des analyses

Pour chaque hypothèse, on considère l'ensemble des solutions engendrées par l'analyse hors contexte. Selon le but de l'analyse, plusieurs tactiques sont possibles. On peut s'arrêter à la première solution trouvée. Grâce au système des hypothèses, les solutions proposées sont construites selon un ordre décroissant de plausibilité.

Néanmoins, dans le cas d'une analyse supervisée par un être humain, il est intéressant de proposer quelques solutions, ordonnées, afin que le choix définitif soit effectué par un spécialiste. Un tel système interactif se doit d'être clair quant aux choix qu'il propose. Il est donc naturel de poser que le système de gestion des hypothèses devra lui aussi être supervisé par l'utilisateur ; on lui proposera différents choix (déjà ordonnés), et il validera la bonne hypothèse.

5.4 Conclusion

Nous avons proposé un système composé de deux parties ; l'une représente le texte en le généralisant au minimum ; l'autre utilise toute les possibilités des grammaires hors contexte. Les deux formalismes se complètent.

La détection de séquences coordonnées, l'une des tâches les plus ardues de l'analyse de textes réels, trouve une place naturelle dans ce processus.

Le problème réside dans le côté très heuristique des modes de résolution envisagés, que ce soit pour la gestion des coordinations ou pour le choix des hypothèses. Une plus grande précision des analyses pourrait permettre une amélioration, à condition que ces analyses soient très ponctuelles, c'est-à-dire se cantonnent à évaluer l'environnement immédiat d'un mot donné du lexique.

Le problème est en fait qu'une analyse syntaxique qui privilégie la couverture ne peut se fonder que sur deux types d'arguments : des arguments micro-syntaxiques, car il est souvent possible d'étudier en détail les environnements habituels d'un mot, et des arguments de nature plus ou moins statistique (Briscoe et Caroll 1993). Il est malheureusement rare de disposer d'une bonne métrique dès que le problème devient un peu complexe, et plus encore d'une métrique linguistiquement justifiée.

Une solution partielle est de faire superviser le système par son utilisateur, ce qui est souvent possible (l'autre terme de l'alternative étant généralement la correction *a posteriori* du résultat proposé par la machine). Dans ce type d'approche, lorsque le problème est exponentiellement complexe, l'ordinateur se doit de présenter à son utilisateur des choix qui aient un sens, c'est-à-dire qu'il faut identifier les vrais problèmes et essayer de résoudre automatiquement les autres.

Chapitre 6

Évaluation

Sommaire

6.1 Couverture et efficacité

L'efficacité temporelle du système final est étroitement liée à l'efficacité des motifs. Si ceux-ci guident effectivement vers la bonne solution, alors, le temps de calcul sera réduit, puisqu'on n'aura pas de retours en arrière.

Exemple 34 (p. 0, l. 0.)

rd.ỉn	*sy*	*ꜣst*	*ḫft*	*ḥr=s*	*nbt-ḥwt*	*ḥr*	*sꜣ=s*	*ḥḳt*
plaça	*soi*	*Isis*	*devant*	*son visage*	*Nephtis*	*sur*	*son arrière*	*Heqet*

ḥr	*sḫꜣḫ*	*mswt*
sur	*accélérer*	*l'accouchement*

Puis Isis se plaça devant elle, Nephtis derrière elle, alors que Heqet accélérait l'accouchement.

Dans le cas de cet exemple, les motifs détectant les syntagmes prépositionnels permettent de choisir directement la bonne solution, alors que les séquences de groupes nominaux (par exemple *ḥr=s nbt-ḥwt*) étaient génératrices d'ambiguïtés.

6.2 Extensibilité

Le système est relativement extensible. En effet, la grammaire hors contexte est sommaire, et convient à presque tous les types de textes de la période. Elle manque bien entendu de précision et génère un nombre d'analyses beaucoup trop important. Mais le système de génération d'hypothèses permet de choisir des hypothèses de qualité raisonnable.

Il faut cependant essayer le système sur des textes complexes. Une des raison de l'efficacité des motifs est leur simplicité. Finalement, entre plusieurs hypothèses, on choisit la plus simple. Si le texte est complexe, cette stratégie devient moins pertinente. Cela dit, la complexité d'un texte comme Sinouhé, grand classique de la littérature de la XII^{e} dynastie, se situe beaucoup plus au niveau de la macro syntaxe que de la complexité des syntagmes. Aussi notre système a-t-il de bonnes chances même face à ce type de textes.

6.3 Comparaison avec d'autres systèmes

La comparaison avec d'autres systèmes d'analyse syntaxique est quelque peu spécieuse, puisque tant les langues que les formalismes de description finaux sont différents. Dans un domaine à première vue plus facile à définir, comme la lemmatisation, les chercheurs sont tout à fait conscients de la vanité des comparaisons purement numériques (Véronis et Khouri 1995) ; nous comparerons donc plus des approches que des résultats.

Une première remarque est la convergence de tous les systèmes que nous avons évoqués en 1.8.3. Un certain nombre d'idées similaires se dégagent. Un trait remarquable est l'unanimité avec laquelle tous les formalismes, d'une manière ou d'une autre, séparent des traitements locaux, plutôt précis, des traitements globaux.

Les formalismes dans lesquels les résultats sont édités diffèrent beaucoup. Nous utilisons pour l'instant des traits, l'analyseur de P. Constant utilise des graphes, et l'analyseur d'Helsinki associe des marques syntaxiques aux mots.

Le texte marqué est sans doute le mode de sortie le plus pratique ; remarquons cependant que nous pouvons le générer.

Une différence plus intéressante est le côté «plat» de l'analyse d'Helsinki. Elle ne comporte pas de groupes syntaxiques (au contraire de son successeur, qui lui, utilise des structures parenthésées).

Étant donné la difficulté (voire l'impossibilité) qu'il y a à résoudre correctement les problèmes de rattachement, cette approche n'est pas sans attraits. De fait, seule l'utilisation effective du résultat de l'analyse peut apporter ici une réponse.

Aucun de ces formalismes n'est dans l'absolu prévu pour des textes non segmentés ; ils sont capables de gérer de longues phrases, mais sans découpage préalable du texte, aucun de ces systèmes ne peut espérer le traiter.

La spécificité de notre système est que la phase de pré-traitement est robuste de ce point de vue. Nous limitons drastiquement les possibilité génératives de notre formalisme. L'usage de motifs, et non de véritables expressions régulières, peut paraître limitatif ; dans les faits, la répétition habituellement codée par l'opérateur « * » est une construction beaucoup trop puissante. Nous avons montré que les séquences effectivement rencontrées entraient en fait dans des cadre relativement bien définis et *marqués*. Il est donc préférable de les traiter à part. L'usage de motifs de longueur bornée nous permet par contre de représenter les configurations usuellement rencontrées ; il peut être intéressant de valider cette approche « minimaliste » sur d'autres langues.

Chapitre 7

Conclusion

Nous pensons avoir ici montré que deux points de vues, provenant d'univers forts différents, celui de l'informaticien spécialiste de langage naturel, et celui du philologue, peuvent interragir dans le cadre des paradigmes qui se développent actuellement en traitement automatique. Dans le numéro de TAL consacré à l'analyse syntaxique, J. M. Marandin écrivait (Marandin 1993, p. 5) :

> [...] l'analyseur occupe dans le TAL la place de l'instance de performance telle que la définit le modèle standard de la grammaire générative et transformationelle ; il en hérite les difficultés [...]

Nous avons effectivement constaté qu'un analyseur fondé sur une grammaire hors contexte, tel que celui du chapitre 4, n'était guère efficace. Il faut un modèle de la performance.

Quel est alors le statut de notre système ? En particulier, étant donné que nous ne sommes jamais confrontés qu'à des instances de performances linguistique, la coexistence de deux représentations est-elle fortuite ? Qu'est notre grammaire hors contexte ?

Nous pensons qu'elle n'est pas un modèle de la compétence, en ce qu'elle ne vise nullement la précision, et qu'elle n'est pas conçue pour rejeter des textes non grammaticaux. Le formalisme hors contexte se trouve simplement être pratique ; de notre point de vue, la propriété qu'il a de décrire un langage au sens mathématique du terme, c'est-à-dire un ensemble de productions grammaticales, est plutôt gênante.

7.1 Quelle méthodologie pour l'analyse syntaxique ?

En pratique, l'analyse syntaxique sert généralement à structurer un texte ; les seules applications où la grammaticalité importe sont les correcteurs grammaticaux ; ce sont justement des applications qui ne peuvent s'autoriser à rejeter un texte gratuitement ; elles doivent essayer de disposer d'une analyse minimale des phrases erronées, afin de pouvoir proposer des corrections. Dans les autres cas, l'intérêt porté au texte beaucoup plus tourné vers l'extraction de fonctions ou de syntagmes que vers la valuation binaire est/n'est pas dans le langage.

La méthode que nous proposons est bipartite : un moteur tente d'un côté de prendre en compte les grands phénomènes grammaticaux ; de l'autre, un système fondé essentiellement sur la recension de groupes courants procède à un choix. Ce second système doit être peu ambitieux. Selon le formalisme employé, il est plus ou moins général ; le degré exact de généralité qu'il convient de lui donner doit être choisi avec soin. Il nous a ainsi paru souhaitable de distinguer les constructions bornées de celles qui ne l'étaient pas.

On remarquera que le modèle proposé fournit en quelque sorte des exemples sur lesquels l'analyseur s'appuie pour choisir ses hypothèses. Techniquement, les choses sont un peu différentes, mais l'idée est présente. Or, qu'utilise la grammaire classique pour aider à la mémorisation des règles ? Des exemples, justement.

On remarquera enfin que les deux mécanismes peuvent être séparés. À partir d'un schéma d'analyse non déterministe quelconque, il est souvent possible d'évaluer *a priori* les groupes fréquents, pour constituer des amorces d'analyses, débarassées d'ambiguïtés gênantes. Le tout est que ce guidage ne restreigne pas l'espace des possibles ; en effet, le choix des groupes qui est fait n'a pas de valeur prescriptive, en ce sens qu'il décrit ce qui est commun, non ce qui est juste.

7.2 Analyse en données rares

Un cas dans lequel notre approche se justifie pleinement est celui où les données sur le type de texte analysé sont rares. Cela concerne un certain nombre de langues mortes, mais peut aussi convenir à des langues vivantes peu étudiées. De plus, la rareté des données peut provenir tout simplement du manque de ressources humaines. Si un nombre très restreint de personnes travaillent sur un analyseur, une approche similaire à la nôtre est probablement plus raisonnable que l'imitation de gros analyseurs syntaxico-sémantiques, souvent fruit du travail d'une équipe entière.

Lorsque les données sont rares, de nouveau éléments peuvent conduire à modifier la grammaire. Si celle-ci est suffisamment robuste, elle acceptera un grand nombre de tournures non prévues, et pourra ainsi traiter des cas non attendus. C'est ce que nous avons voulu gérer lors de l'écriture de notre seconde grammaire. Cependant, cette robustesse est le fruit d'une grande généralité ; une telle généralité crée immanquablement des ambiguïtés. C'est là que le guidage par l'usage intervient. On voit donc qu'il permet de reporter dans un formalisme descriptif des règles qui auraient autrement participé d'un formalisme prescriptif.

La forme effective prise par notre système n'est sans doute pas la seule possible ; on peut développer et adapter l'idée générale à d'autres systèmes d'analyse.

7.3 Étude de la segmentation

Le problème de la segmentation est très général ; c'est foncièrement de la reconnaissance de formes. Il peut se poser, nous l'avons dit, dans la quasi totalité des langues, car il dépend du système d'écriture ou de la forme de la source. Pour parler topologiquement, reconnaître une forme implique de reconnaître son complémentaire, sa frontière, ou son intérieur (car nos formes sont convexes et fermées).

Nous avions donc en fait le choix entre reconnaître les frontières de nos unités, qu'il s'agisse des mots ou des propositions, et reconnaître nos unités sur des critères d'organisation interne.

Nous avons choisi d'utiliser les deux méthodes, la première ayant priorité sur la seconde. En effet, elle est plus simple à mettre en œuvre et permet d'éviter une explosion combinatoire.

Dans le cas des propositions, la reconnaissance sur critères internes est effectuée, soit par les règles de grammaire hors contexte, dans le cas le plus général, soit par des motifs ; il est cependant douteux que ceux-ci soient facilement généralisables.

Il serait souhaitable de comparer les résultats de ces méthodes avec ceux que donneraient des systèmes statistiques, et, surtout en ce qui concerne la reconnaissance des mots, avec ce que donnent des méthodes de plus haut niveau (de type système experts), comme celles employées par S. Billet.

7.4 Diachronie et autres corpus

Notre système utilise une grammaire très vague ; il peut donc travailler sur des textes très variés ; et cette variation peut être temporelle. Nous avons à plusieurs reprises évoqué cette possibilité ; il faut la remettre en perspective : de fait, pour un analyseur syntaxique, une différence de registres ou une différence d'époque sont traitées de la même façon. Du moment que la coupure diachronique n'est pas trop importante, la grammaire peut accepter le texte. C'est même une nécessité pour un analyseur syntaxique, dès qu'il veut pouvoir travailler sur des textes un peu libres ; un locuteur est toujours susceptible de recourir à des tournures marquées comme peu usuelles, soit par archaïsme, soit au contraire par néologisme.

Si la grammaire accepte une grande variété de textes, l'ensemble de motifs ne sera pas également adapté. Les textes sont de complexité fort diverse. En fait, il serait intéressant de structurer les motifs selon leur application possible. On peut même envisager le chargement dynamique d'un ensemble de règles, sur des critères statistiques (la répartition des signes dans un texte est par exemple un assez bon critère de datation). On peut, tout simplement, envisager que l'utilisateur choisisse le jeu de motifs à charger.

Un domaine dans lequel le système devrait être performant est celui des textes « poétiques ». Certains textes, en particulier les hymnes, ainsi que les sagesses, sont très probablement métrés. La métrique égyptienne s'appuyant sur les unités syntaxiques, ces textes sont formés de courtes propositions, souvent fort ardues à comprendre, mais présentant généralement une structure grammaticale simple.

7.5 Langues et icônes

L'écriture hiéroglyphique est riche de possibilités. Nous avons montré qu'elle codait explicitement (mais non systématiquement) un certain nombre de traits sémantiques. C'est sans doute un domaine qui mériterait plus d'attention. On pourrait envisager de construire un système d'indexation de textes égyptiens, utilisant, entre autres, les déterminatifs comme base de recherche. L'intégration de ceux-ci dans une hiérarchie d'objet devrait permettre ce traitement.

Le lien entre image et écriture est aussi assez séduisant pour l'informatique, qui consomme de plus en plus d'images, en particulier dans tout ce qui concerne l'interfaçage. L'une des possibilités les plus fascinantes des hiéroglyphes est d'être à la fois textes et images. Les scribes ont abondamment joué de cette richesse. Ainsi,  écrit *phonétiquement* le nom du dieu Ptah. Dans le même temps, il compose un tableau qui évoque un dieu séparant le ciel de la terre, ce qui est parfois une caractéristique de Ptah (Vernus 1995).

On constate cependant que cette idée est un jeu savant, pas vraiment adaptée au développement d'interfaces ergonomiques. La qualité icônique pure de l'hiéroglyphe n'est pourtant plus à démontrer. Les icônes utilisées sur les sites Web spécialisés dans l'Égypte ancienne, et en particulier le CCER, *(Centre for Computer-aided Egyptological Research)* en sont un témoignage.

Plus profondément, on a récemment fait remarquer (Plas et Vergnieux 1993) qu'il était possible dans une large mesure d'encoder les bas reliefs égyptiens en utilisant le formalisme du *manuel de codage,* c'est à dire celui que nous utilisons pour les hiéroglyphes. Voilà donc un système de représentation, sans aucun doute lié à une iconographie largement codifiée, mais permettant néanmoins de représenter cette dernière de façon satisfaisante.

Perspectives pour un système avancé d'aide à la philologie égyptienne

Les bases du système sont posées ; il nous reste à intégrer les diverses couches logicielles. Rappelons qu'elles sont complexes et relèvent de paradigmes différents. Depuis un dictionnaire hypertexte jusqu'à l'analyse syntaxique robuste en passant par la segmentation en unités signifiantes, nous avons mis au point des modules appartenant à des domaines du TAL très distincts du point de vue des méthodes et de la démarche. Nous envisageons maintenant de nous concentrer plus sur les problèmes d'interfaçage, et, en particulier, de création de bases de données faiblement structurées par plusieurs personnes.

Ces bases de données pourront être généralisées dans leur principe à tous les fonds documentaires qui sont des outils de recherche pour l'étude scientifique de données rares, difficiles d'accès ou de formalisation, voire corrompues (dégradées et partielles).

Références

Aho, A., R. Sethi, et J. Ullman (1989). *Compilateurs, principes, techniques et outils*. Interéditions.

André, J. et H. Richy (1994, juin). Utilisation des index d'un éditeur structuré dans le cadre d'actes médiévaux. Publication interne 841.

Auroux, S. (Ed.) (1989). *Histoire des idées linguistiques*. Pierre Mardaga.

Barkema, H. (1994). Determining the lexical flexibility of idioms. In U. Fries, G. Tottie, et P. Schneider (Eds.), *Creating and Using English Text Corpora*. Amsterdam: Rodops.

Billet, S. (1995). *Apports à l'acquisition interactive de connaissances contextuelles*. Thèse de doctorat, Université Montpellier II.

Billet, S., M. Gondran, et R. Vergnieux (1989a). Intelligence Artificielle et Bases de Données pour l'assemblage des Talatat. In *Actes des IXemes journées Les Systèmes Experts et leurs applications, Avignon89*.

Billet, S., M. Gondran, et R. Vergnieux (1989b). Intelligence Artificielle et Bases de Données pour l'assemblage des Talatat. In *Actes des IXemes journées Les Systèmes Experts et leurs applications, Avignon89*.

Black, E., J. Lafferty, et S. Roukos (1992). Development and evaluation of a Broad-Coverage Probabilistic Grammar of English-Language Computer Manuals. In *30th Annual Meeting of theAssociation for Computational Linguistics*.

Blackman, A. M. (1988). *The Story of King Kheops and the Magicians*. J. V. Books.

Blank, I. (1995). Sentence alignment : methods and implementations. In B. Habert (Ed.), *Traitements probabilistes et corpus*, Volume 36, pp. 81–100. Association pour le traitement automatique des langues.

Boccon-Gibod, H. et J.-C. Golvin (1990). Le grand temple d'Amon-Rê à Karnak reconstruit par l'ordinateur. *Dossiers d'archéologie 153*.

Bonfante, L., J. Chadwick, B. F. Cook, W. V. Davies, J. F. Healey, J. T. Hooker, et C. B. F. Walker (1994). *Reading the Past, Ancient Writing from Cuneiform to the Alphabet.* British Museum Publication.

Brill, E. (1992). A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing, ACL*, Trento, Italy.

Brill, E. (1993a). Automatic Grammar Induction and Parsing Free Text: A Transformation-Based Approach. In *Proceedings of the 31st Meeting of the Association of Computational Linguistics*, Columbus, Oh.

Brill, E. (1993b). *A Corpus-Based Approach to Language Learning.* Ph. D. thesis, Department of Computer and Information Science, University of Pennsylvania.

Brill, E. (1993c). Transformation-Based Error-Driven Parsing. In *Proceedings of the Third International Workshop on Parsing Technologies*, Tilburg, The Netherlands.

Brill, E. (1994). Some advances in rule-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Seattle, Wa.

Brill, E. et S. Kapur (1993). An information-theoretic solution to parameter setting. In *Proceedings of the Georgetown University Round Table on Languages and Linguistics: Session on Corpus Linguistics.*

Brill, E. et M. Marcus (1992a). Automatically acquiring phrase structure using distributional analysis. In *Darpa Workshop on Speech and Natural Language*, Harriman, N.Y.

Brill, E. et M. Marcus (1992b). Tagging an Unfamiliar Text With Minimal Human Supervision. In *Proceedings of the Fall Symposium on Probabilistic Approaches to Natural Language.* American Association for Artificial Intelligence (AAAI).

Brill, E. et P. Resnik (1994). A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of COLING-94*, Kyoto, Japan.

Briscoe, T. et J. Caroll (1993). Generalised Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars. *Computational Linguistics 19*(1), 25–59.

Briscoe, T., V. de Paiva, et A. Copestake (Eds.) (1993). *Inheritance, defaults and the lexicon.* Cambridge University Press.

Buurman, J., N. Grimal, M. Hainsworth, J. Hallof, et D. V. D. Plas (1988). *Inventaire des signes hieroglyphiques en vue de leur saisie informatique.* Mémoires de l'Académie des Inscriptions et Belles Lettres. Paris: Institut de France.

Callender, J. B. (1975). *Middle Egyptian*, Volume 2 de *Afroasiatic Dialects.* Malibu: Uneda Publications.

Champollion, J.-F. (1984). *Principes généraux de l'écriture sacrée égyptienne.* Institut d'Orient. ré-édition de l'original de 1836.

Chang, J.-S., Y.-F. Luo, et K.-Y. Su (1992). GPSM: a Generalized Probabilistic Semantic Model for Ambiguity Resolution. In *30th Annual Meeting of theAssociation for Computational Linguistics.*

Chanod, J.-P. et P. Tapanainen (1996). A Robust Finite-State Grammar for French. In *ESSLLI'96 Workshop on Robust Parsing*, Prague.

Charpin, F. (1994). La pratique de l'informatique dans l'enseignement des langues anciennes. Publication du LITALA, Paris 7.

Chesnutt, D. R. (1996). The Model Editions Partnership: Creating Editions of Historical Documents for the Internet. A proposal for the TEI workshop at the ACM Digital Library '96. on the web.

Chomsky, N. (1971). *Aspects de la théorie syntaxique.* Éditions du Seuil.

Church, K. W. (1988). A Stocastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *Second Conference on Applied Natural Language Processing.* Association for Computational Linguistics.

collectif (1982). *Naissance des écritures : cunéiformes et hiéroglyphes.* Réunion des Musées Nationaux.

collectif (1994a). The TEI Guidelines, a Critique. archives ftp de UICVM.UIC.EDU.

collectif (1994b). TEI P3: Guidelines for Electronic Text Encoding and Interchange. archives ftp de UICVM.UIC.EDU.

Constant, P. (1991). *Analyse syntaxique par couches.* Thèse de doctorat, École nationale supérieure des télécommunications.

Crozier-Brelot, C. (1972). L'ordinateur remplacera-t-il le scribe? In *Textes et langages de l'Égypte Pharaonique*, Volume LXIV/3 de *Bibliothèque d'étude.* Institut Français d'Archéologie Orientale.

Daumas, F. (1962). Application de la syntaxe structurale : La proposition relative égyptienne étudiée à la lumière de la syntaxe structurale. *Orbis II*, 21–32.

de Loupy, C. (1995). La méthode d'étiquetage d'Éric BRILL. In B. Habert (Ed.), *Traitements probabilistes et corpus*, Volume 36, pp. 37–46. Association pour le traitement automatique des langues.

de Saussure, F. (1916). *Cours de linguistique générale*. Payot.

der Luiden, E. J. V. (1992). Incremental Processing and the hierarchical Lexicon. *Computational Linguistics 18*(2), 218–237.

Dessalles, J.-L. (1993). *Modèle cognitif de la communication spontanée, appliqué à l'apprentissage des concepts*. Thèse de doctorat, École Nationale Supérieure des Télécommunications.

DiMarco, C. et G. Hirst (1993). A computational Theory of Goal-Directed Style in Syntax. *Computational Linguistics 19*(3), 451–501.

Donald, M. M. (Ed.) (1990). *Thesaurus Linguae Graecae Canon of Greek Authors and Works* (troisième ed.). Oxford University Press. sur le WWW : http://www.tlg.uci.edu/~tlg.

Ejerhed, E. (1993). Nouveaux courants en analyse syntaxique. *Traitement automatique des langues 34*(1).

Erbach, G. (1993). Using Preference Values in Typed Feature Structures to Exploit Non-Absolute Constraints for Disambiguation. In H. Trost (Ed.), *Feature Formalism and Linguistic Ambiguity*. Horwood.

Gardiner, A. H. (1963). *Egyptian Grammar, Being an Introduction to the Study of Hieroglyphs* (3 ed.). Griffith Institute.

Gazdar, G. et C. Mellish (1989). *Natural Language Processing in Prolog*. Addison-Wesley.

Gundlach, R. et W. Schenkel (1970). *Lexicalisch-grammatische Liste zu Spruch 335a der altägyptischen Sargtexte LL/CT. 335A*. Deutsches Rechenzentrum.

Habert, B. (1991). OLMES *: un système d'exploration et de structuration de textes*. Thèse de doctorat, Université Paris 7.

Habert, B. et C. Jacquemin (1993). Noms composés, termes dénominations complexes : problématiques linguistiques et traitements automatiques. In *Traitements automatiques de la composition nominale*, Volume 34, pp. 5–42. Association pour le traitement automatique des langues.

Habert, B. et A. Salem (1995). L'utilisation des catégorisations multiples pour l'analyse quantitative de données textuelles. In B. Habert (Ed.), *Traitements probabilistes et corpus*, Volume 36, pp. 249–276. Association pour le traitement automatique des langues.

Hagège, C. (1982). *La structure des langues*. Que sais-je? Presses Universitaires de France.

Hainsworth, M. (1979). Traitement automatique des textes en hiéroglyphes égyptiens. In *L'égyptologie en 1979, axes prioritaires de recherche*, Volume 2, pp. 19–23.

Hainsworth, M. (1982). Les derniers scribes de Pharaon : Apple II et IBM. *Programmation et sciences de l'homme*.

Haton, J. P., J. M. Pierre, G. Perennou, J. Caelen, et J. L. Gauvain (1991). *Reconnaissance automatique de la Parole*. Dunod.

Huybregts, R. (1985). The weak inadequacy of context-free phrase structure grammar. In G. de Haan, M. Trommelen, et W. Zonneveld (Eds.), *Van periferie naar Kern*, pp. 81–99. Foris: Dordrecht.

Ide, N. (1995). Encoding Standards for Large Text Ressources : the TEI. In B. Habert (Ed.), *Traitements probabilistes et corpus*, Volume 36, pp. 201–213. Association pour le traitement automatique des langues.

Ide, N. et J. Veronis (Eds.) (1995). *Text Encoding Initiative: Background and Context*, Volume 29. Computer and the Humanities.

Ingerman, P. (1967). Panini-Backup form suggested. *Communications of the ACM 10*(3), 137.

Kaplan, R. M. (1973). A general syntactic processor. In R. Rustin (Ed.), *Natural Language Processing*, pp. 193–241. New York: Algorithmics Press.

Kay, M. (1973). The MIND system. In R. Rustin (Ed.), *Natural Language Processing*, pp. 155–188. New York: Algorithmics Press.

Laporte, E. (1988). La reconnaissance des expressions figées lors de l'analyse automatique. *Langages 90*.

Lefèbvre, G. (1949). *Romans et contes égyptiens de l'époque pharaonique*. Paris: Librairie d'Amérique et d'Orient.

Lefèbvre, G. (1955). *Grammaire de l'égyptien classique* (II ed.). Institut Français d'Archéologie Orientale.

Lichteim, M. (1973). *Ancient Egyptian Literature*, Volume 1. University of California Press.

Loprieno, A. (1995). *Ancient Egyptian: a linguistic introduction*. Cambridge University Press.

Magerman, D. M. et C. Weir (1992). Efficiency, Robustness and Accuracy in Picky Chart Parsing. In *30th Annual Meeting of theAssociation for Computational Linguistics*.

Manber, U. et S. Wu (1993). GLIMPSE : A Tool to Search Through Entire File Systems. Technical Report 93-94, The University of Arizona. disponible à `http://glimpse.cs.arizona.edu`.

Marandin, J.-M. (1993). Analyseurs syntaxiques. Équivoques et problèmes. In *Traitement automatique des languesAnalyse syntaxique*, Volume 34, pp. 5–34. ATALA.

Marouzeau, J. (1951). *Lexique de la terminologie linguistique*. Paul Guethner — Paris.

Martinet, A. (1985). *Syntaxe générale*. Armand Collin.

Maurel, D. (1994). Le traitement automatique de la dérivation des noms de ville. *Traitement automatique des langues 35*(2), 111–129.

Merialdo, B. (1995). Modèles probabilistes et étiquetage automatique. In B. Habert (Ed.), *Traitements probabilistes et corpus*, Volume 36, pp. 7–22. Association pour le traitement automatique des langues.

Émile Benvéniste (1974). *Problèmes de linguistique générale*. Gallimard.

Miller, G. A., R. Beckwith, C. Fellbaum, D. Gross, et K. Miller (1993). Introduction to Wordnet : An On-Line Lexical Database. Technical report, Cognitive Science Laboratory, Princeton University, Princeton. disponible à `ftp://clarity.princeton.edu/pub/wordnet/5papers.ps`.

Parkinson, R. B. (1991). *Voices from Ancient Egypt*. British Museum Press.

Plas, D. V. D. (1994). Dix ans de Table ronde « Informatique & Égyptologie » : Rétrospective et perspective. In R. Vergieux (Ed.), *X^{e} conférence Informatique et Égyptologie, Bordeau*, Volume 10, Paris IV-Sorbonne/CCER.

Plas, D. V. D. et R. Vergnieux (1993). Computer-aided Research on Egyptian Stelae from the Middle Kingdom. In J.-L. Chappaz et S. Poggia (Eds.), *IX^{e} conférence Informatique et Égyptologie, Genève*, Volume 9, Paris IV-Sorbonne/CCER.

Polotsky, H. J. (1976). Les transpositions du verbe en égyptien classique. *Israel Oriental Studies*.

Prévot, P. (1992). Observations sur des stèles du Sérapéum de Memphis. *Revue d'égyptologie 43*, 215–221.

Rajman, M. (1995a). *Apports d'une approche à base de corpus aux techniques de traitement automatique du langage naturel.* Thèse de doctorat, École Nationale Supérieure des Télécommunications.

Rajman, M. (1995b). Approche probabiliste de l'analyse syntaxique. In B. Habert (Ed.), *Traitements probabilistes et corpus*, Volume 36, pp. 157–201. Association pour le traitement automatique des langues.

Role, F. (1996). Le codage informatique des apparats critiques : évaluation des recommandations de la Text Encoding Initiative. *Cahiers Gutemberg 24*.

Rosmorduc, S. (1994). Analyse syntaxique automatique du moyen égyptien. In R. Vergieux (Ed.), *X[e] conférence Informatique et Égyptologie*, Volume 10, Bordeaux.

Sabah, G. (1989). *L'IA et le langage.* Hermès.

Sauneron, S. (1982). *L'écriture figurative dans les textes d'Esna*, Volume VIII de *Esna*. Institut Français d'Archéologie Orientale.

Sblebor, S. M. (????). *An Introduction to Unification Based Approaches to Grammar.* Center for the Study of Language and Information.

Schenkel, W. (1972). Neuel linguistische Methoden und Arbeitstechische Verfahren in der Erschliessung der ägyptischen Grammatik. In *Textes et langages de l'Égypte Pharaonique*, Volume LXIV/1 de *Bibliothèque d'étude*, pp. 167–176. Institut Français d'Archéologie Orientale.

Shiebert, S. M. (1985). Evidence against the non-context-freeness of natural language. *Linguistic and Philosophy* (8), 333–343.

Sleator, D. et D. Temperley (1993). Parsing English with a Link Grammar. In *Third International Workshop on Parsing Technologies, August 1993.*

Sproat, R., C. Shih, W. Gale, et N. Chang (1994). A stochastic finite-state word segmentation algorithm for Chinese. *Proceedings of ACL.*

Stéphane Harié, Élisabeth Murisasco, J. L. M. et J. Véronis (1996). SgmlQL : un langage de requêtes pour la manipulation de documents SGML. *Cahiers Gutemberg 24.*

Tapanainen, P. et T. Järvinen (1994). Syntactic Analysis of Natural Language Using Linguistic Rules and Corpus-Based Patterns. In *proceedings of the Fifteenth International Conference on Computational Linguistics (CoLing 94)*, Volume I, pp. 629–634.

Tesnière, L. (1966). *Élements de syntaxe structurale.* Éditions Klincksieck, Paris.

Tomita, M. (1985). An efficient Context-free Parsing Algorithm for Natural Language. In *9th International Joint Conference on Artificial Intelligence*.

Vernus, P. (1982). Les jeux d'écriture. In *Naissance des écritures : cunéiformes et hiéroglyphes*. Réunion des Musées Nationaux.

Vernus, P. (1990). *Future at Issue. Tense, Mood and Aspect in Middle Egyptian*, Volume 4 de *Yale Egyptological studies*. Yale University.

Vernus, P. (1995). Du bon usage des hiéroglyphes. *l'Histoire 190*, 42–46.

Véronis, J. et L. Khouri (1995). Étiquetage grammatical multilingue : le projet multext. In B. Habert (Ed.), *Traitements probabilistes et corpus*, Volume 36, pp. 233–248. Association pour le traitement automatique des langues.

Weischedel, R., M. Meteer, R. Schwartz, R. Ramshaw, et J. Palmucci (1993). Coping with Ambiguity and Unknown Words through Probabilistic Models. *Computational Linguistics 19*(2), 359–382.

Weizenbaum, J. (1966). Eliza, a computer program for the study of natural language communication between man and machine. *CACM 9*, 36–45.

Winand, J. (1986). Constitution des fichiers-textes en néo-égyptien : lemmatisation et analyse automatique. *Informatique et Statistique dans les sciences humaines XXII*(1-4), 179–190.

Winand, J. (1990). Les bases de données de textes en égyptien. In N. Grimal (Ed.), *Informatique et Égyptologie*, Volume 7.

Winand, J. (1992). *Études de néo-égyptien, 1, la morphologie verbale*. Université de Liège: Centre Informatique de philosophie et lettres.

L'analyseur par charte

On trouvera ci-dessous le code prolog de l'analyseur syntaxique, écrit en SWI prolog.

```
/***
 *** Un moteur de Chart Parser
 ***/
:- module(chartparser,[essai/2, find_analyse/4, arc/6,
          vide_charte/1,extraire_regexpsA/3, lave/0,
          essai_cut/5]).
:- use_module(traits).
:- use_module(union).
:- use_module(graphes).
%%% 1) les chartes
%%% Une charte est une collection contenant :
%%% * des arcs entre les mots (de 0, premiere position,
    a N, nombre de mot
%%% * un etat de la regle qui nous a mene la.
%%% *   i.e. arc(DEBUT,FIN,BUT,PARTIE_DROITE,REGLE,
    POS_DANS_REGLE)
%%% L'etat de la regle, lui, contient des regles
    parenthesees
%%% avec des actions et des valeurs
%%% 2) l'agenda :
%%%     agenda_vide(AGENDA) -> unifie AGENDA a un
    agenda vide.
%%%     push_agenda(arc(....), AGENDA, AGENDA_RESULTAT)
%%%     pusha_agenda(arc(....), AGENDA,
    AGENDA_RESULTAT) mets arc au debut de
%%%
    l'agenda
%%%     pop_agenda(ARC, AGENDA,  AGENDA_RESULTAT)
%%% En fait, charte et agenda sont le meme type.
%%% Quand une partie de l'agenda peut ge'ne'rer une
    expression reguliere,
%%% Elle le fait.
```

```
%%% arc : 1/ 2/ 3/ 4/ 5/coordonnees de la regle
      generatrice 6/position
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%%                  Utilitaires pour les chartes
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% POUR l'instant, sous forme de listes.
:- dynamic arc/6,agenda/6, arc_ut/4.
:- index(arc(1,1,0,0,1,1)).
:- index(arc_ut(1,1,0,0)).
:- index(agenda(1,1,0,0,1,1)).
agenda_vide([]).
agenda_videp([]).
arc2agenda(ARC,AGENDA):-
        ARC =.. [arc,A,B,C,D,E,F],
        AGENDA =.. [agenda,A,B,C,D,E,F].
push_agenda(ARC,A,A):-
        arc2agenda(ARC,AGENDA),
        call(AGENDA), !.
push_agenda(ARC,AG1,[ARC|AG1]):-
        arc2agenda(ARC,AGENDA),
        assert(AGENDA).
pusha_agenda(ARC,A,A):-
        arc2agenda(ARC,AGENDA),
        call(AGENDA), !.
pusha_agenda(ARC,AG1,[ARC|AG1]):-
        arc2agenda(ARC,AGENDA),
        asserta(AGENDA).
pop_agenda(ARC,[ARC|AG1],AG1):-
        arc2agenda(ARC,AGENDA),
        retract(AGENDA).
detruit_agenda([]):-
        retractall(agenda(_,_,_,_)).
% chartes avec la BD du systeme.
% la charte elle meme est la liste des clef a supprimer
      a la fin.
charte_vide([]).
arc_from_charte(ARC,_):-
        ARC= arc(_,_,_,_,_,_),
        call(ARC).
vide_charte(_):-
        retractall(arc(_,_,_,_,_,_)).
```

```
push_charte(ARC,C,C):-
        arc_from_charte(ARC,C), !.
push_charte(ARC,C,C):-
        assert(ARC).
ajouter_arcs([],Ag,Ag).
ajouter_arcs([H|T], Ag, Ag1):-
        push_agenda(H, Ag, Ag0),
        ajouter_arcs(T,Ag0,Ag1).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%      La gestion des traits
%%
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
initialise_val_cat(Cat,Catres):-
        Cat =.. [A,_], Catres =.. [A,[]].
candidat_categorie(Catex, Catregle):-
        Catex =.. [V,_],
        Catregle =.. [V,_].
% calcule_categorie(CUP,CUP,CDOWN,CREGLES):- !.
calcule_categorie(CRES,CUP,CDOWN,CREGLES):-
        CUP =.. [CATEG, UP],
        CRES =.. [CATEG, RES],
        CDOWN =.. [_, DOWN],
        CREGLES =.. [_,REGLES],
        applique_regles(RES,UP,DOWN,REGLES).
applique_regles(UP,UP,_,[]).
applique_regles(RES,UP,DOWN,[REGLE|REGLES]):-
        (
        REGLE = (!(TRUE)) ->

      applique_regle(UP1,DOWNR1,UP,DOWN,TRUE);
        UP1 = UP,
        DOWNR1 = DOWN
        ),
        applique_regles(RES,UP1,DOWNR1,REGLES).
applique_toutes_regles(UP,UP,_,[]):- !.
applique_toutes_regles(RES,UP,DOWN,[REGLE|REGLES]):-
        !,
        (
        REGLE = (!(TRUE)) ->

      applique_regle(UP1,DOWNR1,UP,DOWN,TRUE);
```

```
        applique_regle(UP1,DOWNR1,UP,DOWN,REGLE)
        ),
        applique_toutes_regles(RES,UP1,DOWNR1,REGLES).
%%%     MODIFIER POUR QUE CA MARCHE AVEC a or b or c =
      d or b
applique_regle(RES,DOWNR,UP,DOWN, (A = B)):-
      % REGLE NON SYMETRIQUE :
                                               % EN
      CAS D'affectation C tjrs A
        !,

      prendre_valeur(B,UP,DOWN,NEWUP1,NEWDOWN1,VB,VF),

      prendre_valeur(A,NEWUP1,NEWDOWN1,RES,DOWNR,VA,VF),
        unif(VA,VB,VF).
%%%
%%% \= a une semantique particuliere :
%%% le second terme est instancie
%%% X \= y or z => X ne peut etre Y or Z
%%%
applique_regle(RES,DOWNR,UP,DOWN, (A \= B)):-
        !,

      prendre_valeur(B,UP,DOWN,NEWUP1,NEWDOWN1,VB,VB),

      prendre_valeur(A,NEWUP1,NEWDOWN1,RES,DOWNR,VA,VA),
        nonvar(VB),
        nonvar(VA),
        or2liste(VB,VBB), or2liste(VA,VAA),
        intersection(VAA,VBB,[]).
applique_regle(_,_,_,_, A :: B):-
        nl,
        write('regle fautive : '),
        write((A::B)),
        nl,nl.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
      %%%
%%%
      %%%
%%%                           Le systeme :
      %%%
```

```
%%%
      %%%
%%%
      %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
start_agenda([],_,AG):-
        agenda_vide(AG).
start_agenda([Word|Words], V0, Agenda):-
        V1 is V0 + 1,
        findall(arc(V0,V1,Cat,[],0,0), word(Cat,Word),
      Liste1),
        start_agenda(Words,V1,Agenda2),
        ajouter_arcs(Liste1,Agenda2,Agenda).
extend_arcs(Agenda, Chart, ChartR):-
        pop_agenda(ARC, Agenda, Agenda1) ->
        (
        arc_from_charte(ARC, Chart) ->
                            extend_arcs(Agenda1, Chart,
      ChartR);
                                                   %
      write(('traitement de l arc ',(ARC))), nl,
                            push_charte(ARC, Chart,
      Chart2),
                            new_arcs(ARC, Chart2,
      Edges),
                            add_edges(Edges, Agenda1,
      Agenda2),
                            extend_arcs(Agenda2,
      Chart2, ChartR)
        );
        ChartR= Chart .
/*
***
***
***      Dans cette version, on s'arrete a la premiere
      solution.
***
***
*/
extend_arcs_cut(_, ChartR, ChartR,Trouver,N1,NF):-
        XX=.. [Trouver,_],
        arc_from_charte(arc(N1,NF, XX, _, _,_),ChartR),
```

```
        !.
extend_arcs_cut(Agenda, Chart, ChartR,Trouver,N1,NF):-
        pop_agenda(ARC, Agenda, Agenda1) ->
        (
        arc_from_charte(ARC, Chart) ->
                            extend_arcs_cut(Agenda1,
      Chart, ChartR,Trouver,N1,NF);
                            (
                                expand_in_regexp(ARC,
      ARC1,Chart),
                                pusha_agenda(ARC1,
      Agenda1, Agenda11) ;
                                Agenda11 = Agenda1
                            ),
                            push_charte(ARC, Chart,
      Chart2),
                            new_arcs(ARC, Chart2,
      Edges),
                            add_edges(Edges, Agenda11,
      Agenda2),
                            extend_arcs_cut(Agenda2,
      Chart2, ChartR, Trouver, N1, NF));
                            ChartR= Chart .
%%%% expand_in_regexp() => voir a la fin
add_edges([], Edges, Edges).
add_edges([Edge| Edges], Edges1, Edges2):-
        push_agenda(Edge,Edges1,Edges3),
        add_edges(Edges, Edges3, Edges2).
new_arcs(arc(V1,V2,Cat1,[],_,_),Chart,Edges):-
        findall(arc(V1,V1,Cat2res,[Cat4|Cats],Regle1,
      0),
                (
                candidat_categorie(Cat1,Cat4),
                rule(Cat2, [Cat4|Cats], Regle1),
                initialise_val_cat(Cat2,Cat2res)
                                                    %/*
      calcule_categorie(Cat2res, Cat2, Cat1, Cat4)*/
                ), Edges1),
        findall(arc(V0,V2,Cat3res,Cats2,Regle2, Pos2),
                (
                candidat_categorie(Cat1,Cat5),

      arc_from_charte(arc(V0,V1,Cat3,[Cat5|Cats2],Regle2,
```

```
      Pos), Chart),
                Pos2 is Pos + 1,

      calcule_categorie(Cat3res,Cat3,Cat1,Cat5)
                ),
                Edges2),
        append(Edges1,Edges2,Edges).
new_arcs(arc(V1,V2,Cat1,[Cat2| Cats],Regle,
      Pos),Chart,Edges):-
        breakpoint,
        Pos1 is Pos + 1,
        findall(arc(V1,V3,Cat1res,Cats,Regle, Pos1),
                Regle3^(candidat_categorie(Cat2,Cat3),

      arc_from_charte(arc(V2,V3,Cat3,[],Regle3,_),Chart),

      calcule_categorie(Cat1res,Cat1,Cat3,Cat2)
                    ),
                Edges).
/* Un superviseur pour le chart-parser : */
essai(Chaine,Chart) :-
        start_agenda(Chaine, 0, Agenda),
        extend_arcs(Agenda,[],Chart),
        detruit_agenda([]).
essai_cut(Chaine,Chart,Trouver,N1,NF) :-
        start_agenda(Chaine, 0, Agenda),
        extend_arcs_cut(Agenda,[],Chart,Trouver,N1,NF),
        detruit_agenda([]).
breakpoint.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
      %%%%
%%%
      %%%%
%%%             Interpreteur d'automates réguliers
      %%%%
%%%
      %%%%
%%%
      %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
use_regexp([WORD|TEXT], TEXT1, GRAPH, NOEUD,
      [WORD|LU]):-
        bonne_caracteristique(WORD, LABEL),
        est_arc(GRAPH, NOEUD, NOEUD1, LABEL),
        use_regexp(TEXT, TEXT1, GRAPH, NOEUD1, LU).
use_regexp(TEXT, TEXT, GRAPH, NOEUD, []):-
        est_final(GRAPH, NOEUD).
use_regexp_arcs(V0, V2, GRAPH, NOEUD, [LABEL|SUITE]):-
        est_arc(GRAPH, NOEUD, NOEUD1, LABEL),
        arc_from_charte(arc(V0,V1,Cat,[],_,_),_),
                                                % IL
      FAUDRA FOURNIR LA CHARTE
        bonne_caracteristique(arc(V0,V1,Cat,[],_,_),
      LABEL),
        use_regexp_arcs(V1, V2, GRAPH, NOEUD1,SUITE),
      !.
use_regexp_arcs(V, V, GRAPH, NOEUD,[]):-
        est_final(GRAPH, NOEUD).
bonne_caracteristique(arc(_,_,CARACT,_,_,_), LABEL):-
        CARACT =.. [_,CARACT1],
        member(graphie :: WORD, CARACT1),
        a_la_caracteristique(WORD, LABEL).
bonne_caracteristique(arc(_,_,CARACT,_,_,_), LABEL):-
        CARACT =.. [LABEL,_].
bonne_caracteristique(WORD, LABEL):-
        WORD \= arc(_,_,_,_,_,_),
        word(CARACT, WORD),
        CARACT =.. [LABEL,_].
bonne_caracteristique(WORD, LABEL):-
        a_la_caracteristique(WORD,LABEL).
extraire_regexpsA(TEXTE,N,MAX):-
        N =< MAX,
        extraire_regexpsB(TEXTE,N),
        N1 is N + 1,
        extraire_regexpsA(TEXTE,N1, MAX).
extraire_regexpsB([],_).
extraire_regexpsB(TEXTE, GR):-
        use_regexp(TEXTE, TRESTE, GR, 1, RESULT),
        write(RESULT), !,
        extraire_regexpsB(TRESTE, GR).
extraire_regexpsB([U|TEXTE], GR):-
        write(' '), write(U), write(' '), !,
        extraire_regexpsB(TEXTE, GR).
```

```
expand_in_regexp(arc(V0,_,_,[],_,_),
      arc(V0,V2,gn([lu::LU]),[],0,0), _):-
        GR= 1,                                  % On
      fera mieux plus tard
        use_regexp_arcs(V0,V2,GR,1,LU), write(LU), nl.
/*****************************************************************************/
/***
      ***/
/***
      ***/
/***                La construction la plus haute
      ***/
/***
      ***/
/***
      ***/
/*****************************************************************************/
find_analyse(DEBUT,NUM,NUM1, VF):-
        find_analyse_aux(DEBUT,NUM,NUM1,VF).
/* AVEC LA BASE DE PROLOG */
la_regle_a_ete_utilisee(NUM,NUM1,DEBUT,RuleNUM,POS_FINALE):-

        setof(X, VV^TETE^(TETE =.. [DEBUT,VV],

      arc(NUM,NUM1,TETE,[],RuleNUM,POS_FINALE),
                        RuleNUM \= 0,
                        X = (RuleNUM, POS_FINALE)),
              _).
find_analyse_aux(DEBUT,NUM,NUM1,VF):-
        NUM1 is NUM + 1,
        S=.. [DEBUT, VF],
        arc(NUM,NUM1, S, [],0,0) ;
        montre_trace(('recherche ', DEBUT , NUM, '=->',
      NUM1)),
%%% OLD arc(NUM,NUM1,S,[],RuleNUM,POS_FINALE),

      la_regle_a_ete_utilisee(NUM,NUM1,DEBUT,RuleNUM,POS_FINALE),
%%%
      existe_chaine_back(RuleNUM,NUM,NUM1,ARCS,POS_FINALE),
        rule(NomRegle,R,RuleNUM),
        existe_chaine_forw(R,NUM,NUM1, ARCS),
        se_trouve_aux(R,NUM,NUM1,ARCS,[],VF),
```

```
        montre_trace(('existe ', RuleNUM, NomRegle, '
      ', NUM, '=->', NUM1, ARCS)).
/*
%
%find_analyse_aux(DEBUT,NUM,NUM1,VF):-
%       S=.. [DEBUT, _],
%       arc(NUM,NUM1,S,[],RuleNUM,POS_FINALE),
%       RuleNUM \= 0,
%       montre_trace(('echec ', RuleNUM, NomRegle, ' ',
      NUM, '=->', NUM1, DEBUT)), fail.
*/
/* montre_trace(X):-
        write(X), nl.
*/
montre_trace(_).
%find_analyse_aux(DEBUT,NUM,NUM1,VF):-
%       SR=..[DEBUT, _],
%       S=.. [DEBUT, VF],
%       (\+ rule(SR, _,_)),
%       NUM1 is NUM + 1,
%       arc(NUM,NUM1, S, [],0).
/* ANCIENNE VERSION QUI MARCHE : */
existe_chaine_forw([],NUM,NUM,[]).
existe_chaine_forw([A|RESTE], NUM, NUMF,
      [arc(NUM,NUM1,[])|SUITE]):-
        A=..[Rd,_],
        setof(X, R1^TR^Reg^PF^(R1=..[Rd,TR],
      arc(NUM,NUM1,R1,[],Reg,PF),X=arc(NUM,NUM1,[])),
      _),
        NUM1 =< NUMF,
        existe_chaine_forw(RESTE,NUM1,NUMF,SUITE).
existe_chaine_back(RuleNum, OR,DEST, ARCS,
      POS_FINALE):-
        rule(_,ITEMS,RuleNum),
        reverse(ITEMS,ITEMSR),
        existe_chaine_aux0(RuleNum, ITEMSR, OR,DEST,
      ARCSR, POS_FINALE),
        reverse(ARCSR,ARCS).
existe_chaine_aux0(RuleNum, ITEMSR, OR,DEST, ARCSR,
      POS_FINALE):-
        setof(ARC,
              existe_chaine_aux(RuleNum, ITEMSR,
      OR,DEST, ARC, POS_FINALE),
```

```
               ARCS),
               member(ARCSR, ARCS).
il_y_a_un_arc(NUM,NUM1,Rd):-
        setof(X, VV^TETE^X^REG^PF^(
        TETE=..[Rd,VV],
        arc(NUM,NUM1,TETE,[],REG,PF),
        X= arc(NUM,NUM1,Rd)
    ), _).
il_y_a_un_arc_regle(NUM,NUM1,Regle,POS):-
        setof(X, TETE^CORPS^X^(
        arc(NUM,NUM1,TETE,CORPS,Regle,POS),
        X= arc(NUM,NUM1,Regle)
    ), _).
est_debut([],_).
est_debut([A|As],[A|Bs]):- est_debut(As,Bs).
%%% existe_chaine_aux  REGLE ITEMS DEPART FIN ARCS POS
existe_chaine_aux(_,[],NUM,NUM,[],_).
existe_chaine_aux(RuleNum, [A|RESTE], NUM, NUMF,
      [arc(NUM1,NUMF,[])|SUITE],POS):-
        A=..[Rd,_],
        il_y_a_un_arc(NUM1,NUMF,Rd),
        NUM =< NUM1,
        POS1 is POS - 1,
        il_y_a_un_arc_regle(NUM, NUM1, RuleNum, POS1),
        existe_chaine_aux(RuleNum,RESTE,NUM,NUM1,SUITE,
      POS1).
se_trouve_aux([],NUM,NUM,_,VF,VF).
se_trouve_aux([R0|Reste],NUM,NUMF,[arc(NUM,NUM1,[])|ARCS],
      VI,VF):-
        R0=..[Rd,Regles],
        find_analyse_aux(Rd, NUM,NUM1,DOWN),

      montre_trace(applique_toutes_regles(V2,VI,DOWN,Regles)),
        applique_toutes_regles(V2,VI,DOWN,Regles),
%       ( applique_toutes_regles(V2,VI,DOWN,Regles) ->
%              montre_trace(('SUCCES' ,
      applique_toutes_regles(V2,VI,DOWN,Regles))) ;
%              montre_trace(echec), fail),
        se_trouve_aux(Reste, NUM1, NUMF,ARCS, V2, VF).
cree_arc_utiles(L):-
        detruire_arc_utiles,
        cree_arc_utiles_aux(L), !.
cree_arc_utiles_aux([]).
```

```
cree_arc_utiles_aux([arc(A,B,C,D)|RESTE]):-
        assert(arc_ut(A,B,C,D)),
        cree_arc_utiles(RESTE).
detruire_arc_utiles:-
        retractall(arc_ut(_,_,_,_)).
lave:-
        detruire_arc_utiles,
        vide_charte(_),
        detruit_agenda([]).
```

Fragment de grammaire

```
   voeux
==>
        ['Hwy'], ['A'].
phrase_dialogue
==>
        & particule([d::racine = 'in',
      u::modalite=interrogation]),
        phrase_dialogue0([u=d]).
phrase_dialogue0
==>
        verbeD([! d::forme=imperatif,u=d]),
        &cc([u::circonstants=d]).               % il
      faudrait etudier les
                                                %
      restrictions dans ce cas
phrase_dialogue0
==>
        & particule([!d::clitique=proclitique]),
        phrase([u=d,u::registre=dialogue]).
phrase_dialogue0
==>
        gn([u::vocatif=d]).
phrase_dialogue0
==>
        voeux([u::voeux=d]).
phrase_dialogue0
==>
        particule([!d::clitique=proclitique,
      !d::pronom=dependant]),
        &a3([d::actant3::cat = suffixe,
      u::actants::actant3= d::actant3]),
        a2([u::actants::actant1= d::actant2]),
        &a3([d::actant3::cat \= suffixe,
      u::actants::actant3= d::actant3]),
        cc([u::predicat=d]).
```

```
phrase_dialogue0
==>
        particule([!d::clitique=proclitique,
      !d::pronom=suffixe]),
        a1a3o([u::actants = d]),
        cc([u::predicat=d]).
phrase_dialogue0
==>
        syntagme_adverbial([u=d, u::cat \= verbe]). %
      VERIFIER Qu'IL EST non VERBAL
phrase_dialogue0
==>
        pronom_gn([!d::initial= plus, u::predicat=d]),
        gn([u::sujet=d]).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% formes communes.
%
phrase
==>
        phrase0([u=d]).
%%% predicat adjectival
phrase0
==>
        adjec-
      tif([u::predicat=d,d::accord::genre=masculin]),
        & a2([u::sujet=d::actant2]).
%%% predicat nominal
phrase0
==>
        nom([u::predicat=d]),
        ['pw'],
        & qualification([u::predicat::adjectifs=d]),
        & relative([u::predicat::suite=d]),
        & gn([u::sujet=d]).
phrase0
==>
        nom([u::predicat=d]),
        ['pw'],
        & qualification([u::predicat::adjectifs=d]),
        & relative([u::predicat::suite=d]),
        & gn([u::sujet=d]).
phrase0
```

```
==>
        pronom_gn([u::predicat=d]),
        ['pw'],
        & gn([u::sujet=d]).
phrase0
==>
        proposition_infinitive([u::predicat=d]),
        ['pw'].
phrase0
==>
        verbeD([!u::forme=participe,u::predicat=d]),
        ['pw'],
        & gn([u::sujet=d]).
%%% phrase verbale aoriste ou accompli
phrase0
==>
        particule([! d::graphie= iw]),
        & a1([u::actants=d]),
        verbeC([!d::forme=sdm_n or circonstanciel,
      u=d]).
%%% phrase prospectif autonome
phrase0
==>
        verbeC([!d::forme=prospectif,u=d]),
        &cc([u::circonstants=d]).
%%% phrase au sDmw=f
phrase0
==>
        verbeD([!d::forme=accompli,u=d]),
        &cc([u::circonstants=d]).
%%% temps second
phrase0
==>
        verbeC([!d::forme=sdm_n or nominal,
      u::actants::actant1=d]),
        cc([u::predicat=d]).
%%% negation de l'aoriste
phrase0
==>
        negation([!d::racine=n, u::affirmative=moins]),
        verbeC([!d::forme=sdm_n,u=d]).
%%% negation de l'accompli
phrase0
```

```
==>
        negation([!d::racine=n, u::affirmative=moins]),
        verbeC([!d::forme=indicatif,u=d]).      % A
      verifier !!!!
%%% Cleft-sentence
phrase0
==>
        cleft1([u::theme = d]),
        cleft2([u::rheme = d]).
cleft1
==>
        preposition([!d::racine=in]),
        gn([u=d]).
cleft1
==>
        gn([!d::cat=nom_propre,u=d]).
cleft1
==>
        independant([u=d]).
cleft2
==>
        verbeD([!d::forme=participe, !d::actif =
      actif,u=d]).
cleft2
==>
        verbeC([!d::forme=prospectif_ancien, !d::actif
      = actif,u=d]).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%                        Le groupe nominal
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
gn
==>
        gn0([u=d]),
        & gn([u::suite=d]).
gn
==>
        pronom_gn([u=d]).
gn0
==>
        nom_propre([u=d]),
        & adjectifs([u::adjectifs=d]).
```

```
gn
==>
        verbe_infinitif([u=d]),
        & adjectifs([u::adjectifs=d]).            % rs
      nfr m ....
gn0
==>
        & det1,
        nom([u=d]),
        & qualification([u::adjectifs=d]).
gn0
==>
        & det1,
        relative([u::type=relative,u=d]).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                 Subtilités des goupes nominaux
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
det1
==>
        article.
det1
==>
        article([!d::graphie= wa or wat]),
      preposition([d::racine=m]), article.
det1
==>
        nombre, nisbe_n.
det1
==>
        article([d::graphie='nA']), nisbe_n.
det1
==>
        demonstratif, suffixe.
qualification
==>
        & possessif([u::possessif=d]),
        & determination([u::determination=d]),
        & adjectifs([u::adjectifs=d]),
        & genitifs([u::genitifs=d]),
        & niimy([u::possessif1=d]).
/*
%% grammaticalement juste, mais ge'ne'rateur
      d'ambiguite's.
```

```
possessif([possesseur::N|DEB],DEB)
==>
        a1(N).
*/
%% était impliqué par le précédent
possessif
==>
        suffixe([u=d]).
possessif
==>
        ['iry'].
determination
==>
        demonstratif([u=d]).
determination
==>
        nombre([u=d]).
determination
==>
        ['nb'].
adjectifs
==>
        adjectif([u=d]),
        & adjectifs([u::suivant=d])
.
genitifs
==>
        nisbe_n,
        gn([u=d]),
        & genitifs([u::suivant=d])
.
nisbe_n
==>
        ['n'].
nisbe_n
==>
['nt'].
nisbe_n
==>
['nw'].
niimy
==>
        ['n'], suffixe([u=d]), ['imy'].
```

```
nty
==>
        ['nty'].
nty
==>
 ['ntt'].
nty
==>
 ['iwty'].
relative
==>
        nty,
        &a2([u::actants::actant1=d]),
        cc([u::predicat=d]).
/*
relative(V)
==>
        verbeC(V1),
        {V0=[verbe::V1] union
      [verbe::[forme::forme_relative]]},
        cco(C),
        {append(V0,C,V)}
        .
*/
relative
==>
        verbeC([! d::forme=forme_relative,  u=d]),
        & cc([u::circonstants=d]).
relative
==>
        verbeD([! d::forme=participe, u=d]),
        & cc([u::circonstants=d]).
relative
==>
        verbeC([! d::forme=sdm_ty_fy,
        d::actants::actant1::cat=suffixe,
        d::actants::actant1::personne=3,
        u=d
        ]),
        & cc([u::circonstants=d]).
construction_badale
==>
```

```
        gn([u::matiere=d]), nom([u=d]), &
      nombre([u::nombre=d]).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Les e'lements enclitiques
intrusion
==>
        adjectifs([u::adjectifs=d]).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% LES PRONOMS
% les pronoms !
% groupement i : tout est pre'sent.
gr1
==>
        & intrusion([u::encl=d]),
        gn([u::actant1=d]),
        gn([u::actant2=d]),
        ['n'], gn([u::actant3=d]).
gr1
==>
        suffixe([u::actant1=d]),
        & intrusion([u::encl=d]),
        gn([u::actant2=d]),
        ['n'], gn([u::actant3=d]).
gr1 ==>
        suffixe([u::actant1=d]),
        & intrusion([u::encl=d]),                 %
      verifier
        dependant([u::actant2=d]),
        ['n'], gn([u::actant3=d]).
gr1 ==>
        suffixe([u::actant1=d]),
        ['n'], suffixe([u::actant3=d]),
        & intrusion([u::encl=d]),
        dependant([u::actant2=d]).
gr1 ==>
        suffixe([u::actant1=d]),
        ['n'], suffixe([u::actant3=d]),
        & intrusion([u::encl=d]),
        gn([u::actant2=d]).
gr1 ==>
        & intrusion([u::encl=d]),
        dependant([u::actant2=d]),
        gn([u::actant3=d]), ['n'],
```

```
        gn([u::actant1=d]).
gr1 ==>
        ['n'], suffixe([u::actant3=d]),
        & intrusion([u::encl=d]),
        dependant([u::actant2=d]),
        gn([u::actant1=d]).
% groupement ii : a1 a2
gr2 ==> & intrusion([u::encl=d]), gn([u::actant1=d]),
      gn([u::actant2=d]).
gr2 ==> suffixe([u::actant1=d]), &
      intrusion([u::encl=d]), gn([u::actant2=d]).
gr2 ==> suffixe([u::actant1=d]), &
      intrusion([u::encl=d]),
      dependant([u::actant2=d]).
gr2 ==> & intrusion([u::encl=d]),
      dependant([u::actant2=d]), gn([u::actant1=d]).
% groupement iii : a1 a3
gr3 ==> & intrusion([u::encl=d]), gn([u::actant1=d]),
      ['n'], gn([u::actant3=d]).
gr3 ==> ['n'], suffixe([u::actant3=d]), &
      intrusion([u::encl=d]), gn([u::actant1=d]).
gr3 ==> suffixe([u::actant1=d]), &
      intrusion([u::encl=d]), ['n'],
      gn([u::actant3=d]).
gr3 ==> suffixe([u::actant1=d]), ['n'],
      suffixe([u::actant3=d]), &
      intrusion([u::encl=d]).
% groupement iv : a1
gr4 ==> & intrusion([u::encl=d]), gn([u::actant1 = d]).
gr4 ==> suffixe([u::actant1=d]), &
      intrusion([u::encl=d]).
% groupement v : a2 a3
gr5 ==> & intrusion([u::encl=d]), gn([u::actant2=d]),
      ['n'], gn([u::actant3=d]).
gr5 ==> & intrusion([u::encl=d]),
      dependant([u::actant2=d]), ['n'],
      gn([u::actant3=d]).
gr5 ==> ['n'], suffixe([u::actant3=d]), &
      intrusion([u::encl=d]),
      dependant([u::actant2=d]).
gr5 ==> ['n'], suffixe([u::actant3=d]), &
      intrusion([u::encl=d]), gn([u::actant2=d]).
% groupement vi : a2
```

```
gr6 ==> & intrusion([u::encl=d]), gn([u::actant2=d]).
gr6 ==> & intrusion([u::encl=d]),
      dependant([u::actant2=d]).
% groupement vii : a3
gr7 ==> & intrusion([u::encl=d]), ['n'],
      gn([u::actant3=d]).
gr7 ==> ['n'], suffixe([u::actant3=d]), &
      intrusion([u::encl=d]).
% expression des pronoms :
a1 ==> gr4([u=d]).
a1a2a3 ==> gr1([u=d]).
a1a2a3o ==> gr1([u=d]).
a1a2a3o ==>  gr2([u=d]).
a1a2oa3o ==> gr1([u=d]).
a1a2oa3o ==> gr2([u=d]).
a1a2oa3o ==>  gr3([u=d]).
a1a2oa3o ==>  gr4([u=d]).
a1a2oa3 ==> gr1([u=d]) .
a1a2oa3 ==> gr3([u=d]).
a2a3 ==> gr5([u=d]).
a2a3o ==> gr5([u=d]).
a2a3o ==> gr6([u=d]).
a2oa3o ==> gr5([u=d]) .
a2oa3o ==> gr6([u=d]) .
a2oa3o ==> gr7([u=d]) .
a2oa3 ==> gr5([u=d]) .
a2oa3 ==> gr7([u=d]).
a1a3o ==> gr3([u=d]) .
a1a3o ==> gr4([u=d]).
a1oa3o ==> gr3([u=d]) .
a1oa3o ==> gr4([u=d]) .
a1oa3o ==> gr7([u=d]) .
a2 ==> gr6([u=d]) .
a3 ==> gr7([u=d]) .
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                       Ce qui suit un verbe
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
verbeC1 ==>
        verbe_c([! d::transitif = plus, ! u=d]),
        a1a2oa3o([u::actants=d]).
verbeC2 ==>
        verbe_c([! d::causatif = plus,! u=d]),
      % causatif
```

```
        a1a3o([u::actants=d]),
        completive([u::actants::actant2=d]).
verbeC2 ==>
        verbe_c([! d::negatif = plus,! u=d]),
      % causatif
        a1a3o([u::actants=d]),
        completive_negative([u::actants::actant2=d]).
%% Des verbes intransitifs.
verbeC3 ==>
        verbe_c([! d::transitif = moins,!u=d]),
        a1a3o([u::actants=d]).
%%% Formes a sujet non exprime
verbeD1
==>
        verbe_d([! d::transitif = plus, !u=d]), %
      transitif
        & a2a3o([u::actants=d]).
verbeD2 ==>
        verbe_d([! d::causatif = plus, !u=d]),  %
      causatif
        & a3([u::actants=d]),
        completive([u::actants::actant2=d]).
verbeD2 ==>
        verbe_d([! d::negatif = plus, !u=d]),   %
      causatif
        & a3([u::actants=d]),
        completive_negative([u::actants::actant2=d]).
%% Des verbes intransitifs.
verbeD3 ==>
        verbe_d([! d::transitif = moins,! u=d]),
        & a3([u::actants=d]).
%% verbeC[123] etant utilise pour des verbes avec sujet
      suffixal/exprime
%% il est inutilisable pour les infinitifs, les
      participes, et les imperatifs
%% [forme::racine or indicatif or circonstanciel or
      prospectif
%%    or prospectif_ancien or nominal or forme_relative
      or sdm_n or sdm_in or accompli]
verbe_c
==>
        verbe(
```

```
        [! d::forme = racine or indicatif or
      circonstanciel
        or prospectif or prospectif_ancien or nominal
      or
        sdm_ty_fy or
        forme_relative or sdm_n or sdm_in or accompli,
! u=d]).
%% { X = Y union [forme= participe or imperatif or
      accompli]}
verbe_infinitif                                 %
      transitif
==>
        verbe([! d::forme= infinitif,! d::transitif =
      plus,! u=d]),
        & a1oa3o([u::infinitive::actants = d]).
verbe_infinitif                                 %
      intransitif
==>
        verbe([! d::forme= infinitif,! d::transitif=
      moins,! u=d]),
        & a3([u::infinitive::actants=d]).
verbe_d % verbe sans sujet exprimé
==>
        verbe([! d::forme = imperatif or accompli or
      particípe or forme_verbale_negative,! u=d ])
        .
verbeC    ==>
        verbeC1([! u=d]).
verbeC    ==>
        verbeC2([! u=d]).
verbeC    ==>
        verbeC3([! u=d]).
verbeD ==>
        verbeD1([! u=d]).
verbeD ==>
        verbeD2([! u=d]).
verbeD ==>
        verbeD3([! u=d]).
%% construction avec un infinitif
%% proposition infinitive
completive
==>
        completive_prospectif([u=d]).
```

```
completive
==>
completive_infinitif([u=d]).
completive_prospectif
==>
        verbeC([! d::forme= prospectif,u=d]).
completive_infinitif
==>
        verbe_infinitif([u=d]).
completive_negative
==>
        verbe_infinitif([u=d]).
completive_negative
==>
        verbeD([!d::forme= forme_verbale_negative,
      u=d]).
proposition_infinitive
==>
        verbe_infinitif([u=d]).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%                        éléments adverbiaux de base
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Formes communes à tous les registres
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
un_adjectif
==>
adjectif([u=d]).
un_adjectif
==>
nom([! d::adj_pot= plus,u=d]).
cc
==>
        syntagme_adverbial([u=d]),
        & ccs([u::suite=d]).
ccs
==>
        syntagme_adverbial([u=d]),
        & ccs([u::suite=d]).
ccs
==>
```

```
        suite_adverbiale([u=d]).
/* suite_adverbiale
==>
        particule([!d::racine=iw]),
        suffixe([u::actants::actant1=d]),
        cc([u::predicat=d]).
*/
suite_adverbiale
==>
        gn([u::actants::actant1=d]),
        cc([u::predicat=d]).
syntagme_adverbial
==>
        adverbe([u::type=adverbe,u=d]).
syntagme_adverbial
==>
        preposition([d::graphie='Hr']),
        proposition_infinitive([u::type=extensif,
        u=d
        ]).
syntagme_adverbial
==>
        preposition([d::graphie='r']),
        proposition_infinitive([u::type=futur,
        u=d
        ]).
syntagme_adverbial
==>
        preposition([d::graphie='m']),
        proposition_infinitive([u::type=inchoaactif,
        u=d
        ]).
syntagme_adverbial
==>
        preposition([u::type=prep,u::preposition=d]),
        a1([u::nom=d::actant1]).
syntagme_adverbial
==>
        preposition([u::type=prep,u::preposition=d]),
        completive_prospectif([u::nom=d]).
syntagme_adverbial
==>
```

```
        verbeC([! d::forme= circonstanciel or
      prospectif,
        u=d,
        u::type=verbal]).
                                                  %
      forme adverbiale.
                                                  % ou
      forme finale
syntagme_adverbial
==>
        verbe([! d::forme=accompli,u=d,
      u::type=verbal]),
        & psp.
syntagme_adverbial
==>
        verbeC([! d::forme=sdm_n,u=d, u::type=verbal]).
syntagme_adverbial
==>
        negation([!d::graphie=nn]),
      gn([u::negation=d]).
```

Structure du dictionnaire

Nous donnons ci-dessous la `dtd` du dictionnaire, accompagnée d'un exemple d'entrée.

```
     <!DOCTYPE dico [
   <!ELEMENT DICO - - (ENTREE)+>
   <!element ENTREE O O (GROUPESENS+)>
   <!attlist ENTREE
           REF NUM #REQUIRED
           TRANSLITTERATION PCDATA #REQUIRED
           >
   <!element GROUPESENS O O
         (GRAPHIE|TRADUCTION|USAGE|GROUPESENS)+>
   <!element GRAPHIE - O
         (COMMENTAIRE|GROURESENS|REFERENCE)>
   <!attlist GRAPHIE
           MDCCODE PCDATA #REQUIRED
           >
   <!element TRADUCTION - -
         (#PCDATA|COMMENTAIRE|REFERENCE|GROUPESENS)+>
   <!entity % VARUSAGE "MOT | QQUN | QQCHOSE" >
   <!element MOT - O EMPTY>
   <!element QQCHOSE - O EMPTY>
   <!element QQUN - O EMPTY>
   <!element USAGE - - (#PC-
         DATA|COMMENTAIRE|%VARUSAGE;|REFERENCE|GROUPESENS)+>
   <!element REFERENCE (#PCDATA,COMMENTAIRE?)>
   <!element COMMENTAIRE - - (#PCDATA|GROUPESENS)+>
   <!attlist COMMENTAIRE
           TYPE (grammatical|usage|auteur) #IMPLIED
           >
   ]>
   <dico>
     <entree ref="3" translitteration="TAi">
       <groupesens>
         <!-- one of (usage traduction graphie) -->
```

```
            <graphie mdccode="TA-A-i-i-D51:D40">
              <!-- one of (reference grouresens commentaire)
            -->
              <reference>n</reference>
            </graphie>
            <groupesens>
            <traduction>prendre</traduction>
            <traduction>voler</traduction>
            </groupesens>
            <groupesens>
              <!-- one of (groupesens usage traduction
            graphie) -->
              <usage><mot> n <qqun><reference>P.
            Gurob@2,2</reference></usage>
              <traduction>s'en prendre à qqun</traduction>
              <traduction>faire des reproches à
            qqun</traduction>
            </groupesens>
          </groupesens>
        </entree>
      </dico>
```

L'un des avantages de ce type de description est qu'il est extensible. Il est possible de rajouter des données sans que pour autant des dictionnaires déjà réaliser deviennent inutilisables.

D'autre part, comme nous l'avons dit au chapitre 3, nous avons choisi une structure très lâche pour le dictionnaire. Elle permet de rajouter des entrées dans un ordre quelconque ; par exemple de partir d'un sens pour expliciter un usage, ou le contraire, ce qui nous semble nécessaire pour une base de donnée qui se veut aussi un document de travail. Le système listé n'est pas définitif. Il est en effet nécessaire de mettre au niveau le plus externe d'une entrée le nom de son auteur, ce qui permettra la cohabitation sans conflit, dans un même dictionnaire, d'entrées provenant de plusieurs sources différentes.

Tksesh

Tksesh, notre système de saisie et de gestion de base de données hiéroglyphique, se compose d'un noyau dur écrit en C, et d'une interface réalisée en TCL.

.1 Un peu de technique

La partie C comprend essentiellement la fonction d'affichage du texte hiéroglyphique. Celle-ci ont été réalisées en optimisant le code pour assurer la plus grande rapidité d'affichage possible. Comme le dessin d'un quadra (un groupe de hiéroglyphes arrangés en carré) demande des calculs, nous avons évité au maximum de les redessiner si c'était possible. Ceci est réalisé grâce à un système de buffers multiples.

La partie déjà affichée est sauvegardée dans un buffer, quand la partie à afficher est calculée dans un autre buffer. L'algorithme de dessin est le suivant :

Se positionner au premier quadra de l'écran.
TANTQUE on n'est pas sorti de l'écran **FAIRE**
    SI le quadra courant était visible
        ALORS le recopier à sa nouvelle place
    SINON le dessiner
    FIN

On utilise donc trois buffers : un pour l'ancien écran, un pour construire le nouvel écran, et un troisième, celui qui est affiché, pour améliorer le confort visuel. Pour éviter de devoir recalculer à chaque fois si un quadra était visible, un quadra visible doit être marqué selon la date à laquelle il est visible. Cela garanti que l'information de visibilité est à jour.

Les quadras sont stockés sous formes de liste chaînée, ce qui ne pose pas *a priori* de problème de complexité, sauf lorsque l'on veut gérer des informations globales, comme par exemple, la longueur maximale d'une ligne ou le nombre de lignes. Malheureusement, ce type d'informations est utilisé pour les « ascenceurs » qui permettent un déplacement rapide dans le texte. Le calcul de l'une ou l'autre quantité impose en effet le parcours de la totalité du texte. Un système de stockage arborescent devrait permettre de résoudre ce problème. Cela dit, il n'est pas vraiment sûr qu'il soit nécessaire de le résoudre en priorité, le ralentissement n'étant pas très important en pratique, et la majeure partie des textes étant courts.

.2 Représentation du texte

Le texte est actuellement représenté sous la forme préconisée par le *manuel de codage*. L'objet servant à l'affichage peut accepter de lire ou d'écrire du texte dans ce format. Cependant, s'il est nécessaire de conserver cette fonctionnalité pour des raisons de standardisation, le format du *manuel* est peu maniable. Il est difficile à étendre, et difficile à manipuler automatiquement.

Nous allons donc créer un format de travail, à base de listes. Les entités créees seront donc immédiatement manipulables par TCL. De plus, en nous inspirant de SGML, nous allons pouvoir aller beaucoup plus loin que le système très *ad hoc* utilisé actuellement, et rajouter de façon structurée des informations supplémentaires sur le texte.

Le dictionnaire doit être revu, et véritablement structuré selon les grandes lignes présentées au 3.

Ce texte a été composé à l'aide du programme LaTeX ; les hiéroglyphes ont été insérés grâce au *package* HieroTeX, que j'ai réalisé. Il est disponible actuellement à `http://www.iut.univ-paris8.fr/~rosmord/archives/`