Une plate-forme informatique de Navigation Textuelle : modélisation, architecture, réalisation et applications de NaviTexte.

Javier Couto

UNIVERSITÉ DE PARIS IV – SORBONNE
ÉCOLE DOCTORALE CONCEPTS ET LANGAGES

THESE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE PARIS IV

Discipline : Informatique

Spécialité : Informatique linguistique

présentée et soutenue publiquement par Javier Couto le 10 juillet 2006

Titre : Une plate-forme informatique de Navigation Textuelle : modélisation, architecture, réalisation et applications de NaviTexte.

Directeur de thèse : Monsieur JEAN-LUC MINEL

Membres du jury

M. Patrice ENJALBERT, Professeur, Université de Caen, Rapporteur

M. Yves JEANNERET, Professeur, Université de Paris-Sorbonne, Examinateur

Mme. Lita LUNDQUIST, Professeur, CBS, Danemark, Examinatrice.

M. Jean-Luc MINEL, Ingénieur de Recherche (HDR), CNRS, Directeur de Thèse

M. Jean-Paul SANSONNET, Directeur de recherche au CNRS, Rapporteur

Mme. Dina WONSEVER, Professeur, Universidad de la República, Uruguay, Examinatrice

RÉSUMÉ

Au long de l'histoire, des instruments de recherche d'information ou d'aide à la lecture, fondés sur la notion de page, tels que la table des matières, les index, les renvoi, etc., ont été introduits. Dès l'arrivée de l'informatique, ces instruments, de nature typiquement statique, se sont multipliés et ils ont vu croître leur puissance. Dans le cadre général de l'histoire du texte numérique, l'*hypertexte* place un jalon du point de vue conceptuel, et l'utilisation massive d'Internet a répandu son utilisation à grande échelle.

Le terme de *navigation textuelle* reçoit de multiples interprétations, la plus commune renvoyant inévitablement au processus mis en œuvre par les outils de navigation utilisés pour circuler dans les documents hypertextes. Néanmoins, notre conception de la *navigation textuelle* se démarque de la navigation hypertextuelle traditionnelle car nous considérons que circuler ou naviguer dans un texte est l'expression d'un processus cognitif qui convoque des connaissances qui sont propres à la finalité de la navigation. Nous formulons l'hypothèse que la démarche du lecteur peut être assistée par l'exploitation de connaissances, présentes dans les textes, qui peuvent être, en partie, modélisées sous une forme déclarative. Autrement dit, il ne suffit pas de créer des liens mais il est nécessaire d'expliciter l'opération de navigation. De plus, ce processus de définition d'opérations de navigation doit être mis en œuvre par un « expert » capable d'encoder ces connaissances.

Ce travail de thèse présente quatre contributions principales. En premier lieu, *une représentation des textes* spécifique à la navigation textuelle est définie. En deuxième lieu, *un langage formel de modélisation des connaissances de visualisation et de navigation*, nommé Sextant est proposé. Les constructions possibles du langage sont données par une syntaxe. Le sens des constructions syntaxiques du langage Sextant est déterminé par une *sémantique opérationnelle*. En troisième lieu, *une plate-forme logicielle dédiée à la navigation textuelle*, nommée *NaviTexte*, a été implémentée. Dans cette implémentation, trois choses ont été développées : une représentation informatique des textes spécifique à la navigation textuelle, fondée sur la proposition d'un encodage XML des textes ; un interpréteur d'une version réduite de Sextant, le langage de modélisation des connaissances, fondé sur la proposition d'un encodage XML de ce langage ; un environnement capable de traiter les textes, d'interpréter le langage Sextant et de gérer l'interaction avec l'utilisateur. En dernier lieu, diverses *applications* de la plate-forme logicielle NaviTexte à des cas réels d'utilisation ont été mises en œuvre.

TITLE

A text navigation system: modeling, architecture, development and applications of NaviTexte.

ABSTRACT

Through history, search information tools based on the notion of page, such as tables of contents, index, references, etc., had been proposed. With the rise of computers, these tools, typically static in nature, had seen their power increased. In the field of digital texts, *hypertext* is a great achievement from a conceptual point of view, and the massive utilization of the Internet has spread his utilization at big scale.

Text navigation term has many interpretations. The most known of them refers to the navigation process carried out by traditional browsers. Nevertheless, our conception of text navigation diverges from the traditional hypertext navigation one because we consider that navigating a text is the expression of a cognitive process that involves navigation specific knowledge. We claim that a reader's activity can be assisted by the exploitation of knowledge that exists in texts, and that this knowledge may be modeled in a declarative way. Creation of links is not enough: it is necessary to make explicit the notion of *navigation operation*, and it has to be done by an "expert" capable of encoding this knowledge.

The present work has four main contributions. First, it defines a computer text representation of texts, specific to our text navigation approach. Second, a formal language of visualization and navigational knowledge modeling, named Sextant, is proposed. While the constructions of the language are defined by a syntax, the meaning of these constructions is determined by an operational semantic. Third, a text navigation system, named NaviTexte, has been implemented. Finally, several applications of NaviTexte in real world situations are presented.

DISCIPLINE – SPÉCIALITÉ DOCTORALE

Informatique – Informatique linguistique

MOTS-CLÉS

Navigation textuelle ; représentation informatique des textes ; modélisation de connaissances ; plate-forme informatique de navigation textuelle.

Laboratoire LaLICC, Langages, Logiques, Informatique, Cognition et Communication, UMR 8139 du CNRS – Université Paris-Sorbonne. Maison de la Recherche, 28 rue Serpente, 75005, Paris, France.

Remerciements

Jamais je n'aurai pu réaliser ce travail de thèse sans le soutien de mon directeur de recherche, Jean-Luc Minel. Je tiens à remercier vivement sa patience, sa sincérité, sa disponibilité pour travailler en équipe, son optimisme, la pertinence de ses conseils et sa profonde générosité. La confiance qu'il m'a accordée dès le début de la thèse m'a permis d'élaborer un plan de travail personnel accorde à mes aspirations. Je tiens ici à lui exprimer ma profonde reconnaissance et ma sincère amitié.

Je remercie Monsieur Jean-Pierre Desclés, directeur du laboratoire LaLICC, qui m'a accueilli dans son laboratoire. J'ai toujours trouvé au sein du laboratoire un environnement de travail propice. Je tiens à remercier également son support : ma participation au projet eVEIL et mes fonctions en tant qu'ATER ont été enrichissants tant du point de vue professionnel que personnel, et ils m'ont permis de me soutenir financièrement.

Je voudrais aussi remercier Gustavo Crispino, qui m'a contacté avec Jean-Luc Minel et qui a crée les conditions nécessaires pour que je puisse faire mes premiers stages au laboratoire LaLICC. Sa confiance, sa générosité et son support ont été indispensables pour que cette thèse puisse se matérialiser.

Je remercie Monsieur Patrice Enjalbert et Monsieur Jean-Paul Sansonnet, qui a eu la amabilité de me faire parvenir une version corrigée de la première version de ce document, d'avoir accepté d'être rapporteurs, et Monsieur Yves Jeanneret, Madame Lita Lundquist et Madame Dina Wonsever de me faire l'honneur de participer au jury.

Je remercie Delphine Battistelli, avec qui j'ai eu le plaisir de travailler dans le cadre du Bi-Deug en Informatique et Linguistique à l'Université de Paris-Sorbonne, pour son travail de relecture.

Je garde de nombreux souvenirs des doctorants avec qui j'ai partagé un repas, un café, une salle ou une fête (chez Jorge, le plus souvent) : Motasem, Jorge, Carine, Marie, Daniela, Jungyeon, Denitsa, Aude, Adélaïde, Nikolay, Elena, Florence, Jérôme…

Je n'oublierai pas les aides permanentes reçues du personnel administratif du laboratoire. Je remercie Madame Anne Armand pour son efficience. Je remercie particulièrement Roselyne Cantarel, pour sa volonté, sa disponibilité et sa gentillesse constantes.

Je remercie l'école doctorale « Concepts et Langages » (ED V), qui a soutenu financièrement ma participation à plusieurs colloques, conférences et écoles d'été.

Je tiens à remercier Lita Lundquist pour son excellente disposition pour le travail en équipe et pour son chaleureux accueil à Copenhague.

Je remercie Alberto Pardo de ses conseils et de son aide concernant les aspects de sémantique formelle du langage développé. Je remercie également mes collègues au « Instituto de Computación de la Facultad de Ingeniería », en Uruguay, pour leur solidarité.

Enfin, pour leur soutien sans faille et permanent, je tiens à remercier de tout cœur Leticia, mes parents et Patricia.

Table de matières

Liste de Tableaux

Table de Figures

Introduction

1. Problématique

Qu'est-ce que la navigation textuelle ?

Au long de l'histoire, des instruments de recherche d'information ou d'aide à la lecture, fondés sur la notion de page, tels que la table des matières, les index, les renvoi, etc., ont été introduits. Dès l'arrivée de l'informatique, ces instruments, de nature typiquement statique, se sont multipliés et ils ont vu croître leur puissance. Actuellement, les logiciels de traitement textuel offrent potentiellement des instruments beaucoup plus puissants, ce qui est en général possible grâce à la possibilité de disposer, en arrière plan, de la représentation structurelle du texte.

Dans le cadre général de l'histoire du texte numérique (*cf.* chapitre 4), l'*hypertexte* place un jalon du point de vue conceptuel, et l'utilisation massive d'Internet a répandu son utilisation à grande échelle. L'implémentation classique de l'hypertexte consiste à offrir à l'utilisateur la possibilité d'activer un lien entre deux nœuds pour déplacer le point de lecture, ce déplacement pouvant être intra ou intertextuel. Plusieurs aspects, que l'on considère désavantageux, sont à souligner dans ce type de navigation hypertextuelle. Tout d'abord, l'activation du lien est « aveugle », dans le sens ou l'utilisateur possède peu d'information sur la cible du lien (un titre ou l'adresse URL qui est en général peu significative), et qu'il doit cliquer afin de tester la véritable pertinence du lien par rapport à ses intérêts d'information. Deuxièmement cette navigation est linéaire, c'est-à-dire qu'une seule voie de navigation est offerte au lecteur quand celui-ci active le lien. Troisièmement, l'orientation de la navigation n'est pas indiquée explicitement : le lecteur ne sait pas si le déplacement se fait vers l'amont ou vers l'aval[1] du texte lu, ce qui entraîne, entre autres, des phénomènes de désorientation cognitive [Elm *et* Woods 1985] [Edwards *et* Hardman 1989] [Cotte 2004$_b$]. Enfin et surtout, les liens sont placés dans le corps même du texte, ce qui implique qu'il n'est pas possible d'adapter les parcours dans ce texte au lecteur. En d'autres termes, aucune information ou connaissances complexes ne peuvent être associées à la navigation.

[1] *L'orientation* n'a de signification que dans le cas d'une navigation intratextuelle.

Le terme de *navigation textuelle* reçoit de multiples interprétations, la plus commune renvoyant inévitablement au processus mis en œuvre par les outils de navigation utilisés pour circuler dans les documents hypertextes. Néanmoins, notre conception de la *navigation textuelle* se démarque de la navigation hypertextuelle traditionnelle[2] car nous considérons que circuler ou naviguer dans un texte est l'expression d'un processus cognitif qui convoque des connaissances qui sont propres à la finalité de la navigation [Minel 2003], [Couto *et* Minel 2004_a, 2004_b]. De ce fait, un documentaliste qui doit écrire un résumé d'un texte [Endres-Niggemeyer *et al.* 1995] ne navigue pas de la même façon qu'un lecteur intéressé par l'évolution des sentiments d'un des personnages d'un roman [Mathieu 2004] ou qu'un linguiste qui explore les annotations placées par un système automatique [Pery-Woodley 2005].

Nous formulons l'hypothèse que la démarche du lecteur peut être assistée par l'exploitation de connaissances, présentes dans les textes, qui peuvent être, en partie, modélisées sous une forme déclarative. Autrement dit, il ne suffit pas de créer des liens mais il est nécessaire d'expliciter l'opération de navigation. De plus, ce processus de définition d'opérations de navigation doit être mis en œuvre par un « expert » capable d'encoder ces connaissances.

Comme on le verra, il existe une différence capitale entre la *navigation textuelle que nous* proposons et la navigation hypertextuelle. Cette différence réside dans le codage et l'interprétation des connaissances navigationnelles. Dans le cas de l'hypertexte, celles-ci sont encodées par l'auteur dans le texte même et en font partie. Notre approche consiste à séparer les connaissances navigationnelles de l'objet texte. En outre, nous considérons qu'un texte peut être visualisé de différentes manières et que chaque visualisation peut déterminer une façon de le naviguer.

Du point de vue du lecteur, la navigation textuelle que nous proposons est alors très différente de la navigation hypertextuelle au sens ou nous considérons que le lecteur, qui active lui aussi des connaissances d'interprétation [Kintsch 2003] [Baccino 2004] doit pouvoir interagir en choisissant la voie de navigation qui lui semble la plus appropriée pour sa tâche de lecture.

Notre proposition de navigation textuelle implique quatre éléments constitutifs :

- disposer d'une représentation du texte pouvant décrire différents phénomènes linguistiques servant, à un utilisateur, de guide à la lecture ;

[2] Il faut souligner que la différence n'est pas due à l'élision du préfixe *hyper*.

- la possibilité de pouvoir isoler les connaissances visuelles et navigationnelles et de représenter celles-ci de manière précise ;

- un agent (une personne, une équipe d'experts, un système, etc.) capable d'encoder ces connaissances en utilisant un langage formel ;

- un système qui interprète les connaissances encodées par les agents, permettant à l'utilisateur de naviguer dans les textes.

La navigation textuelle est-elle utile ?

La navigation textuelle est-elle nécessaire ? Est-elle utile ? L'argument le plus souvent revendiqué est celui référant au nombre de documents publiés sur la toile. Voyons un autre argument. Le 14 décembre 2004, Google a annoncé qu'il aspirait à numériser, en six ans, 15 millions d'ouvrages provenant de prestigieuses bibliothèques américaines et britanniques (New York Public Library ; universités de Michigan, de Harvard, de Stanford, d'Oxford).

Le projet Google Print, fortement contesté de différents points de vue (économiques, culturels, stratégiques…)[3], est une sorte de bibliothèque universelle permettant à tout internaute de consulter entièrement le contenu de ses ouvrages, mais lui interdisant de les télécharger, de les sauvegarder et de les imprimer. La possibilité d'avoir une énorme masse de documents que l'on ne peut manipuler qu'à travers des outils logiciels, et en conséquence par des interfaces homme-machine, semble un champ prometteur quant aux applications de notre approche. Dans cette perspective, notre approche de la navigation textuelle vient enrichir l'*architexte* [Souchier *et* Jeanneret 1999] constitué par et autour de cette bibliothèque universelle.

2. Travaux motivant la réalisation de cette thèse

En 1998, une collaboration entre le laboratoire LaLICC de l'Université Paris–Sorbonne et le groupe de T.A.L. de l'Université de la République d'Uruguay, est mise en place. La plate-forme logicielle *ContextO* [Crispino *et al.* 1999], visant le filtrage sémantique des textes et implémentant la plate-forme conceptuelle *FilText*, est le fruit de la collaboration entre différents

[3] Par exemple, en France, Jean-Noël Jeanneney, président de la Bibliothèque nationale de France (BNF), estime dans *Le Monde* du 24 janvier 2005 que Google défie L'Europe *: « Voici que s'affirme le risque d'une domination écrasante de l'Amérique dans la définition de l'idée que les prochaines générations se feront du monde »*.

chercheurs. Ma participation dans le développement de ContextO [Couto 2002] a eu trait spécifiquement aux aspects de visualisation interactive d'information [Hascoët *et* Beaudouin-Lafon 2001] [Card *et al.* 1999] et à ceux d'interaction avec les utilisateurs [Shneiderman 1998].

Le développement et, notamment, l'utilisation de la plate-forme *ContextO* nous a permis de constater l'importance de l'interactivité dans le traitement de filtrage et, également, le bénéfice dû aux interfaces adaptatives. Les opérations de visualisation, la possibilité d'avoir différents types de vues et divers filtrats pour un même document source, et la coordination automatique de vues, éléments d'une approche dynamique à la construction automatique de résumés [Crispino *et* Couto 2004], représentent pour l'utilisateur des outils lui permettant d'interpréter et de mieux comprendre les filtrats fournis par *ContextO*.

C'est dans le cadre du projet RÉGAL[4] [Ferret *et al.* 2001] [Sabah 2002] [Couto *et al.* 2004] que nous avons pu mettre à l'épreuve nos choix de visualisation et d'interaction. L'objectif principal de ce projet a consisté à fournir à l'utilisateur des outils d'exploration d'un document dont il ne peut pas juger la pertinence par rapport à ses objectifs de recherche, malgré la description présentée par un moteur de recherche ou le résumé indicatif généré par un système de résumé automatique.

Notre participation dans le projet RÉGAL nous a permis, d'une part, d'abstraire la notion de vue par rapport à un document et, d'autre part, d'incorporer la notion *d'opération de navigation*. C'est ainsi que notre conception des outils de visualisation et navigation repose sur un principe général concernant l'objet textuel à produire : à partir de l'objet textuel source T, le résultat de la fouille du texte est un nouvel objet textuel T_f qui intègre d'une part l'objet T et une représentation décorée de T appelée T_d (autrement dit, le résultat de la fouille est une fonction de T et de T_d), et d'autre part un ensemble de vues étant donné que le système permet d'avoir plusieurs vues pour un même texte (une vue étant une forme sémiotique FS du texte T_d, celle-ci peut se concevoir comme une fonction de FS et de T_d). Nous pouvons alors décrire le rôle des vues dans la fouille de textes avec l'expression suivante :

$$T_f\,(T,\, T_d,\, \{V_1(FS_1,\, T_d,\, O_1),\, V_2(FS_2,\, T_d,\, O_2),\, \ldots\, V_n(FS_n,\, T_d,\, O_n)\,\})$$

[4] Le projet RÉGAL (REsumé Guidé par les Attentes du Lecteur), financé par l'Action Concertée Incitative Cognitique 2000 (LAC038), a été mené conjointement par le CEA, le laboratoire LaLICC, le LATTICE et le LIMSI, sous la direction scientifique de G. Sabah.

Les éléments O_1, O_2, …, O_n représentent des ensembles d'opérations de visualisation et sont donc associés à des vues. Cela signifie que chaque vue est fonction d'une forme sémiotique, d'un texte décoré et d'un ensemble, spécifique à celle-ci, d'opérations de visualisation. La fouille du texte T reste fonction de celui-ci, de ses décorations (T_d) et des vues possibles sur lui.

Le travail accompli au sein du projet RÉGAL nous a permis de souligner l'importance des aspects de visualisation et d'interaction pour un système de fouille de textes. Parallèlement, nous avons commencé à formaliser ceux-ci, ce qui nous a amené à constater qu'il s'agissait des connaissances amalgamées à la plate-forme car, de fait, tant les représentations visuelles du texte que les opérations de navigation étaient spécifiques et encodées dans l'applicatif. Afin d'abstraire ces connaissances des systèmes, un langage formel de représentation de ces connaissances devait être proposé.

3. Principales contributions

Je voudrais dégager quatre contributions principales de ce travail de thèse. En premier lieu, *une représentation des textes* spécifique à la navigation textuelle a été définie, comportant les caractéristiques suivantes :

- le type d'unités textuelles qui composent un texte n'est pas restreint à un ensemble prédéterminé ;

- il est possible d'avoir une organisation hiérarchique des unités textuelles et une autre permettant d'exprimer des relations non hiérarchiques ;

- les titres sont considérés comme des unités textuelles ;

- toute unité textuelle, y compris les titres et les relations non hiérarchiques, est susceptible d'avoir un nombre non limité d'annotations de nature quelconque.

En deuxième lieu, *un langage formel de modélisation des connaissances de visualisation et de navigation*, nommé Sextant (par analogie avec les navigateurs du XVIII siècle qui ont parcouru le monde en s'orientant sur les mers avec un sextant), a été proposé. Les constructions possibles du langage ont été données par une syntaxe. Le sens des constructions syntaxiques du langage Sextant est déterminé par une *sémantique opérationnelle*.

En troisième lieu, *une plate-forme logicielle dédiée à la navigation textuelle*, nommée *NaviTexte*, a été implémentée. Dans cette implémentation, trois choses ont été développées :

- une représentation informatique des textes spécifique à la navigation textuelle, fondée sur la proposition d'un encodage XML des textes ;

- un interpréteur d'une version réduite de Sextant, le langage de modélisation des connaissances, fondé sur la proposition d'un encodage XML de ce langage ; par ailleurs, un éditeur spécifique pour ce langage a été développé dans le cadre d'un projet du DESS ILSI de l'Université Paris-Sorbonne ;

- un environnement capable de traiter les textes, d'interpréter le langage Sextant et de gérer l'interaction avec l'utilisateur.

En dernier lieu, diverses *applications* de la plate-forme logicielle NaviTexte à des cas réels d'utilisation ont été mises en œuvre. Dans le cadre d'une collaboration avec le Département de français Institut F.I.R.S.T - Handelshøjskolen i København, un outil d'aide à la compréhension de textes complexes pour des étudiants FLE, nommé NaviLire[5], a été développé en s'appuyant sur la plate-forme NaviTexte. Une expérimentation sur un groupe d'étudiants danois en FLE a été effectuée, obtenant des résultats encourageants. Cette expérimentation va se traduire prochainement par la réalisation d'une série d'exercices proposés dans un CD associé à un ouvrage sur l'apprentissage du français [Lundquist 2006].

4. Plan de la thèse

Ce manuscrit se divise en quatre parties : *préambule*, *représentation des textes*, *modélisation des connaissances* et *une plate-forme de navigation textuelle et ses applications*.

Préambule

Puisque la problématique considérée dans ce travail de thèse est le produit d'une réflexion menée tout au long des différents projets qui l'ont précédé, je présente dans le *chapitre 1* un bref historique évoquant les travaux précédents qui l'ont motivé. Ce chapitre montre comment les différents travaux nous ont permis de souligner l'importance des aspects de visualisation et d'interaction pour un système de fouille de textes.

La nature de ce travail de thèse mérite un chapitre dédié aux interfaces homme-machine (*IHM*), en particulier à la visualisation interactive d'information. Je présente donc au *chapitre 2* une introduction brève aux objectifs du domaine des IHM, les différents styles

[5] Cette application a reçu le soutien financier du département scientifique de l'ambassade de France au Danemark

d'interaction homme-machine, quelques principes pour le développement d'interfaces utilisateur et le rôle de la visualisation dans les systèmes d'information.

Au cours du *chapitre 3*, l'approche traditionnelle de l'hypertexte, et les approches alternatives de l'hypertexte adaptatif et de l'hypertexte dynamique, qui essaient de remonter certains inconvénients existants dans l'approche traditionnelle, sont présentés. Plusieurs aspects de la navigation hypertextuelle, considérés comme désavantageux, sont soulignés. C'est dans ce chapitre que je développe la notion de navigation textuelle, en mettant l'accent sur les différences existantes avec la navigation hypertextuelle.

Dans le *chapitre 4* j'expose certaines questions liées à la conceptualisation d'un texte. Je montre qu'il s'agit d'un objet complexe et que tout essai de le modéliser ou de le représenter, afin de le manipuler informatiquement, présuppose forcement une prise de position et, souvent, une simplification.

Il est clair que le traitement informatique d'un texte exige nécessairement une représentation de celui-ci afin d'être manipulé par un système. Le type de traitement à effectuer déterminera habituellement la représentation la plus adéquate selon des critères à choisir : rapport coût / performance, souplesse, exhaustivité, simplicité, etc. Dans le *chapitre 5* j'analyse différents types de représentations des textes, en identifiant trois types possibles : *synthétisées*, *plates* et *structurées*. Puisque le type de traitement à effectuer détermine la représentation la plus convenable, les représentations présentées s'avèrent insuffisantes pour la navigation textuelle. Les représentations structurées sont plus adéquates dans notre cadre, mais il manque la possibilité de représenter des structures discursives complexes qui ne suivent pas une hiérarchie. Aussi, je m'intéresse spécifiquement à une représentation permettant d'exprimer à la fois la structure syntaxique des textes, les structures discursives, et les valeurs sémantiques des constituants textuels, car toutes ces informations vont pouvoir servir de guide à la lecture dans le cadre de la navigation textuelle. L'intérêt de combiner la structure syntaxique avec des phénomènes discursifs, et le fait de souhaiter manipuler certaines structures complexes en tant qu'entités, motivent la définition d'une représentation spécifique pour la navigation textuelle, développée au *chapitre 6*.

Modélisation des connaissances

Après avoir constaté que, dans les différents systèmes étudiés, les représentations visuelles d'un texte, les opérations de visualisation et les opérations de navigation sont, de fait, des connaissances imbriquées dans ces systèmes, l'idée de formaliser celles-ci est apparue souhaitable. Je présente dans le *chapitre 7* les différentes connaissances à modéliser dans le cadre

de la navigation textuelle : la notion de vue d'un texte et les opérations de navigation, de visualisation et de coordination. De mon point de vue, les connaissances décrites dans le cadre de cette thèse constituent un bon point de départ pour développer la navigation textuelle, mais elles ne doivent pas être considérées comme exhaustives ou définitives. Le *chapitre 8* présente Sextant, un langage déclaratif de formalisation de connaissances de visualisation et de navigation. Dans le chapitre 9, une sémantique opérationnelle est proposée, afin de formaliser le sens des constructions syntaxiques de Sextant. Cette sémantique définit également une partie de la sémantique d'un système interpréteur du langage (*i.e.* une plate-forme de navigation textuelle). Bien que conscient de l'aspect un peu fastidieux de cette description, j'estime que celle-ci offre une explication abstraite du langage et du comportement du système interpréteur, indépendante de toute implémentation informatique, où la notion d'état du système est explicitée, ainsi que la manière dont celle-ci change au fur et à mesure que l'utilisateur interagit avec le système. De plus, la spécification formelle de la sémantique m'a permis de corriger la première implémentation de l'interpréteur du langage développé dans la plate-forme logicielle NaviTexte.

Une plate-forme de navigation textuelle et ses applications

Une plate-forme logicielle, NaviTexte, a été développée afin d'implémenter et de pouvoir tester notre approche. Cette implémentation implique divers composants. Le *chapitre 10* propose un encodage XML permettant d'encoder les textes selon la représentation informatique présentée au chapitre 6, afin d'être manipulés par la plate-forme logicielle NaviTexte. Au *chapitre 11*, une version XML du langage de modélisations de connaissances Sextant est définie. Le point fort de cette version réside dans la facilité, pour un système informatique, de manipuler les différentes instances du langage (des documents XML). Le point faible vient de la complexité des documents XML obtenus. Afin de pallier cette difficulté, un éditeur de modules de connaissances a été développé dans le cadre d'un projet du DESS ILSI de l'Université Paris-Sorbonne. Cet éditeur est actuellement opérationnel et il est utilisé dans le cadre de différentes applications de la plate-forme NaviTexte.

Le *chapitre 12* présente NaviTexte, la première plate-forme logicielle implémentant notre approche de navigation textuelle. Il faut souligner que NaviTexte est une plate-forme logicielle opérationnelle sur laquelle plusieurs applications concrètes ont été développées. De plus, l'aspect modulaire de la plate-forme, dû au très grand soin apporté à la conception générale, a permis d'adapter assez facilement NaviTexte à NaviLire, une application pour enseigner le français à des étudiants danois.

Je présente dans *le chapitre 13* les applications créées en utilisant la plate-forme logicielle NaviTexte. Celles-ci sont assez hétérogènes, ce qui est pour nous une preuve de la souplesse de NaviTexte en tant que plate-forme d'expérimentation. En dehors des applications présentées dans ce chapitre, il y en a d'autres en cours actuellement. Toutes ces applications, et les expérimentations menées avec elles, poussent à la réflexion, favorisant l'enrichissement général de notre approche de navigation textuelle.

Première Partie

Préambule

Sommaire

La problématique considérée dans ce travail de thèse est le produit d'une réflexion menée tout au long des différents projets qui l'ont précédé. Ceux-ci vont des premiers systèmes fondés sur l'exploration contextuelle développés au sein de l'équipe LaLICC, jusqu'au projet RÉGAL, où nous avons esquissé les notions initiales constituant ce que nous avons appelé postérieurement *navigation textuelle*. De plus, ce travail est, d'une part, proche aux travaux portant sur l'hypertexte, notamment à ceux sur l'hypertexte adaptative et sur l'hypertexte dynamique ; d'autre part, les propositions et développements sont fortement liés aux interfaces homme-machine (*IHM*), tout en particulier à la visualisation interactive d'information.

Je présente dans cette partie un bref historique évoquant les travaux précédents qui ont motivé cette thèse (chapitre 1), une introduction brève aux objectifs du domaine des IHM (chapitre 2) et les travaux plus proches à celui développé dans cette thèse (chapitre 3). Ce dernier chapitre présente la navigation textuelle.

Chapitre 1

Motivations

1.1. Introduction

Entre les années 1990 et 1998, différents prototypes informatiques ont été développés au sein de l'équipe LaLICC du CAMS (Centre d'Analyse et de Mathématique Sociales), reprenant des résultats obtenus dans plusieurs thèses de doctorat. Le premier système, baptisé *SECAT* (Système d'Exploration Contextuelle de l'Aspect et du Temps) [Desclés *et al.* 1991], avait pour tâche principale le repérage des valeurs sémantiques des temps de l'indicatif dans un texte. Ce système, réalisé par Christophe Jouis en *SNARK*[6], était semi-automatique, au sens où pour traiter un texte l'utilisateur devait intervenir, en répondant à des questions au fil du traitement. L'objectif du système étant la validation du modèle linguistique conçu, plutôt que la construction d'un produit logiciel, cette contrainte ne représentait pas un problème, quoiqu'une automatisation du processus de traitement restât souhaitable.

Le deuxième prototype développé a été le système *SEEK* (Système Expert d'Exploration (K)Contextuelle) [Jouis 1993], dont l'objectif était l'analyse sémantique de textes en français pour l'aide à l'acquisition et à la modélisation d'un domaine de connaissances. A partir des valeurs sémantiques attribuées par le système à une relation entre des termes identifiés, le système produisait des représentations visuelles des résultats, sous forme graphique. Les relations prises en considération étaient un sous ensemble des relations statiques (*partie-tout*, *d'ingrédience*, *d'inclusion*, etc.). Il faut souligner que SEEK traitait le texte phrase par phrase.

L'un des apports majeurs du système a été l'évidence pragmatique de la faisabilité d'une approche fondée sur l'expression de règles heuristiques pour le repérage de relations sémantiques, sans faire appel préalablement à une analyse « classique » de type morphosyntaxique.

[6] Un langage déclaratif développé par J.L. Laurière [1986], utilisé pour construire des systèmes experts.

La production des résumés automatiques a motivé la conception et le développement d'un nouveau prototype, le système *SERAPHIN* (Système Expert de Repérage Automatique des Phrases Importantes d'un texte et de leur Normalisation) [Berri 1996], issu du projet SERAPHIN[7] [Le Roux *et al.* 1994]. Ce système visait la création des résumés automatiques indicatifs par l'extraction des phrases considérées, d'un point de vue discursif, comme importantes. Les textes à résumer possédaient une structuration très variable et traitaient de domaines très divers ; le système ne dépendait donc pas d'une structuration prédéterminée ni d'une description d'un domaine quelconque, d'un dictionnaire du domaine ou d'une ontologie. La phrase constituant l'unité textuelle minimale d'extraction, l'extrait fourni par le système devait être compréhensible par un lecteur non expert. Les phrases considérées comme importantes étaient des phrases relevant, par exemple, d'une *annonce thématique*, d'une *récapitulation*, d'un *soulignement*, d'une *hypothèse*, d'une *définition*, une *exemplification*, etc. Ce système présentait une division en trois modules logiciels de traitement : *pré-traitement du texte*, *attribution des valeurs sémantiques* et *sélection des phrases importantes. Cette architecture modulaire traduit la volonté des concepteurs* d'élaborer un système logiciel plus complexe que les précédents.

La prise en considération des besoins des utilisateurs a mené vers un quatrième prototype, le système *SAFIR* (Système Automatique de Filtrage d'InfoRmations textuelles) [Berri *et al.* 1996], qui étendait le système *SERAPHIN* en permettant à l'utilisateur d'avoir plusieurs extraits d'un document conformément à ses besoins spécifiques. Ceux-ci sont indiqués par une liste hiérarchisée d'étiquettes sémantiques servant à filtrer ou à ignorer un certain type d'information. Un apport fondamental du système a été l'utilisation d'étiquettes sémantiques, attribuées selon la fonction discursive d'une phrase, au lieu d'utiliser la technique classique de « *scoring* » de phrases.

Enfin, le système *COATIS* (Causalité et Action : Traitement Informatique et Sémiotique des textes) [García 1998] permettait l'analyse des textes d'un domaine technique quelconque afin d'élaborer une structuration des connaissances causales présentes dans le texte. Il identifiait dans les textes l'expression de situations organisées par des rapports de cause à effet, repérant aussi les expressions qui jouent, dans cette relation, le rôle de la cause et celles qui jouent le rôle de l'effet. Vingt-cinq relations causales spécifiques sont analysées, et un index du document traité est crée sous la forme d'un hypertexte représentant un réseau où les

[7] Financé par l'EDF (Électricité De France) dans le cadre d'un contrat de recherche avec le CAMS et le CNRS.

nœuds sont des expressions de situations, et les liens sont les causalités. L'information causale recueillie par *COATIS* est exploitable dans le processus de modélisation d'un domaine, dans celui de la structuration d'une terminologie, dans le filtrage automatique des textes ou encore dans l'indexation sémantique de documents.

Notons que tous ces systèmes avaient comme objectif commun le traitement d'un document afin de créer un document résultat (résumé, index, etc.), tout en utilisant des approches purement linguistiques. Nonobstant cette caractéristique commune, force est de constater que tous ces systèmes avaient été développés indépendamment les uns des autres, ayant chacun ses propres formats d'entrée et de sortie, et sa propre architecture, ce qui ne permettait pas de les intégrer facilement dans une plate-forme plus globale de traitement automatique de textes. Il ne fait aucun doute que, d'une part, cette dispersion des ressources logicielles empêchait l'équipe LaLICC de bénéficier d'un développement incrémental, où les hypothèses de travail seraient testées pour différents types de traitement. D'autre part, les expériences réalisées avec les systèmes mentionnés, en particulier dans le cas du résumé automatique (systèmes *SERAPHIN* et *SAFIR*), avaient révélé les difficultés de réaliser des tels systèmes sans tenir compte des besoins spécifiques de l'utilisateur. En effet, les résumés seront très différents selon les « lecteurs » auxquels ils sont destinés. Ainsi, on ne produira pas le même résumé d'un article scientifique innovant si l'on doit adresser ce résumé à la direction générale, au service des brevets pour consultation juridique, au laboratoire de développement, ou aux services de presse grand public [Minel 2003]. Les résumés dépendent également des types de textes. On ne résume pas de la même manière un texte narratif, un article scientifique portant sur une science expérimentale ou un article juridique. Il n'y a donc pas de résumé idéal qui serait indépendant des demandes des utilisateurs et des types de textes.

Aussi, deux axes de travail se sont montrés prometteurs : celui correspondant à la conception et développement d'une *plate-forme d'ingénierie linguistique* dédiée au traitement textuel ; et celui visant des systèmes automatiques de filtrage d'informations extraites de textes, permettant à *l'utilisateur* d'exprimer ses besoins de recherche d'information.

1.2. La plate-forme conceptuelle *FilText*

Le concept de plate-forme d'ingénierie linguistique dédiée au traitement textuel conçue comme *une boîte à outils* propose de définir un modèle conceptuel général de représentation de certaines connaissances linguistiques, afin de développer ultérieurement des tâches spé-

cialisées coopérant entre elles. La notion de « tâche multiple » est fondamentale car, comme le remarque Jean-Guy Meunier : « *Un logiciel qui, dans l'accès à l'information textuelle, ne réalise qu'un seul type de tâche devient vite insatisfaisant parce qu'il ne correspond pas à la nature cognitive de ce que font les lecteurs et les analystes de textes. Ceux-ci ont des lectures multiples des textes et ils veulent parcourir un texte dans diverses perspectives.* » [Meunier 1998].

La plate-forme conceptuelle *FilText* [Minel *et al.* 2001] reprend partiellement ce paradigme et capitalise l'expérience acquise lors du développement des systèmes précédents. Sa conception ambitionne une plate-forme considérée, à la fois, comme un outil de travail pour des recherches linguistiques, et comme un système capable d'offrir les fonctionnalités propres au processus de fouille de textes. Une telle plate-forme présente plusieurs avantages car elle offre, en même temps, un environnement de test d'hypothèses linguistiques, destinée aux linguistes, mais aussi un système destiné aux utilisateurs finaux. Du point de vue du linguiste, il est impératif de pouvoir confronter ses hypothèses de travail à des textes réels. Une plate-forme que lui permet exprimer ses hypothèses, par exemple sous forme déclarative, et ensuite les tester sur un corpus, représente un gain en terme de temps, et également en méthodologie de travail.

En ce qui concerne la fouille de textes, la plate-forme doit permettre aux utilisateurs d'interpréter les résultats de ce processus, aspect amplement souligné par Jean-Luc Minel : « *L'utilisateur final doit disposer de fonctionnalités qui lui permettent d'interpréter les résultats de la fouille des textes; ce qui signifie que des connaissances spécifiques à la tâche et à la présentation de ces résultats doivent pouvoir venir enrichir la plate-forme FilText. Ce point est très important en TAL, car il n'est pas possible d'envisager de développer une plate-forme qui réponde à tous les besoins. Il est nécessaire de concevoir celle-ci de telle manière qu'elle puisse accueillir des développements spécifiques ou que réciproquement, les résultats produits par la plate-forme puissent être exploités par ceux-ci. C'est donc l'inverse du concept de boîte noire.* » [Minel 2002].

Cette citation souligne le contraste entre la réalité de la plupart des systèmes de résumé automatique existants à l'époque et l'objectif principal de *FilText* : éviter une approche de traitement de type boîte noire. De fait, une approche possible dans les systèmes de construction automatique de résumés par extraction consiste à concentrer les efforts principalement dans le processus de construction du résumé. Les systèmes qui adoptent cette approche s'appuient sur des modèles – linguistiques, statistiques ou hybrides – afin de détecter les parties les plus importantes d'un document pour un utilisateur. Ces systèmes doivent résoudre des problèmes caractéristiques de la construction de résumés, tels que la cohésion et la cohé-

rence de l'extrait. Dans certains cas un profil d'utilisateur est inclus, de manière à ce que le résumé obtenu puisse être adapté aux attentes de l'utilisateur. Néanmoins, en général, la participation de l'utilisateur se limite au choix du taux de réduction du résumé par rapport au texte original. En conséquence, ces systèmes travaillent comme une boîte noire : l'utilisateur leur fournit un document et reçoit un résumé, qui est un autre document, indépendant du premier.

La définition de la plate-forme *FilText* évite explicitement cette approche. Son architecture (*cf.* figure 1.1) a été conçue comme un ensemble de sous-systèmes : un *système de gestion des connaissances linguistiques*, un *moteur d'exploration contextuelle* et un ensemble d'*agents spécialisés* qui permettent de développer des traitements spécifiques à l'utilisateur en exploitant l'information produite par le moteur.

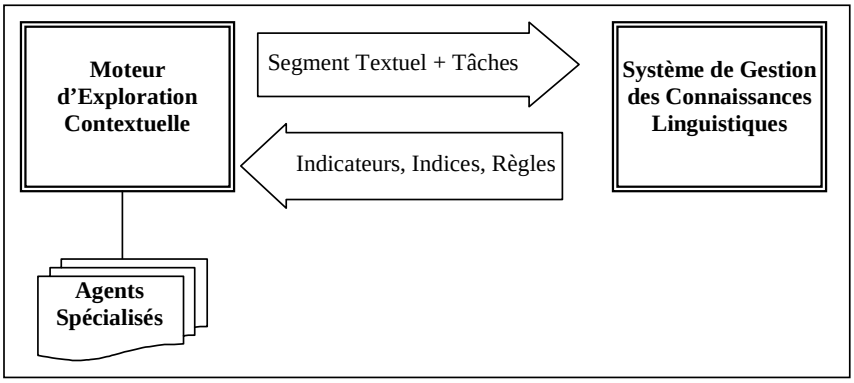

Figure 1.1 – *Architecture de FilText.*

Le *système de gestion des connaissances linguistiques* a pour charge d'accueillir les connaissances linguistiques. Ce système devant être alimenté, le travail préalable du linguiste consiste à étudier systématiquement un corpus de textes pour y rechercher des régularités lexicales et discursives dont l'emploi est représentatif de la catégorie sémantique considérée.

Le *moteur d'exploration contextuelle* exploite les connaissances linguistiques pour une ou plusieurs tâches de traitement choisies par l'utilisateur. Il s'appuie sur un modèle hiérarchisé qui reflète l'organisation structurelle du texte.

Les *agents spécialisés* ont pour tâche d'exploiter les résultats produits par le moteur d'exploration contextuelle en fonction des objectifs définis par l'utilisateur. Ainsi, il existera

un agent *résumeur* chargé de construire un résumé, un agent *filtreur* capable de créer des filtrats des textes, un agent de modélisation de connaissances « Seek-Java », développé par Florence Le Priol [2000], etc.

Dans cette optique, la possibilité d'un développement incrémental de ressources, tant du point de vue linguistique que du type de traitement, est assurée. En effet, l'architecture définie permet d'incorporer facilement *différents types de traitement* car l'indépendance des composants et la séparation entre l'étiquetage sémantique et l'exploitation des étiquettes par les agents spécialisés offre la possibilité d'intercaler de nouveaux traitements et de les combiner. Dans ce cadre de travail il est possible de combiner, par exemple, un traitement de type essentiellement linguistique avec un traitement de type statistique, en créant un agent spécialisé à ces fins.

1.3. La plate-forme logicielle *ContextO*

1.3.1. Un développement collaboratif

En 1998, une collaboration entre le laboratoire LaLICC de l'Université Paris–Sorbonne et le groupe de T.A.L. de l'Université de la République d'Uruguay, est mise en place. La plate-forme logicielle *ContextO* [Crispino *et al.* 1999], qui implémente la plate-forme conceptuelle *FilText*, est le fruit de la collaboration entre différents chercheurs : Jean-Luc Minel, Gustavo Crispino, Slim Ben Hazez et moi-même ; ce travail a été enrichi par les contributions de Florence Le Priol [Le Priol 2000] et de Ghassan Mourad [Mourad 2001].

Nous pouvons identifier deux implémentations informatiques de la plate-forme ContextO. Dans la première, les concepts de base de la plate-forme *FilText* ont été abordés et développés. S'agissant d'un prototype expérimental dont les utilisateurs étaient les développeurs, les aspects d'interface utilisateur étaient plutôt minimalistes. En conséquence, l'objectif stratégique de la plate-forme conceptuelle *FilText* de ne pas travailler comme une boîte noire était partiellement achevé, car l'absence d'un éditeur de profils de filtrage rendait celle-ci peu inter active.

Dans le cadre de mon travail de *Master* [Couto 2002], j'ai collaboré à la conception et au développement de la deuxième version de *ContextO*, spécifiquement sur des aspects de visualisation interactive de l'information, et sur l'interaction avec les utilisateurs. J'ai ainsi par-

ticipé à la matérialisation de la deuxième version de ContextO, une plate-forme informatique opérationnelle réalisant les objectifs de *FilText*.

1.3.2. Première implémentation : pouvoir visualiser le contexte d'extraction de phrases

Les trois sous-systèmes principaux de la plate-forme *FilText* ont été développés prioritairement pour la première version de *ContextO*.

Le *système de gestion des connaissances linguistiques* [Ben Hazez 1999, 2002] utilise une base de données relationnelle pour stocker et exploiter les donnés linguistiques. Il est composé de trois parties : une *base de données linguistiques* permettant de stocker dans des tables relationnelles les donnés linguistiques de l'expertise acquise, une *couche de services* qui permet à plusieurs applications de partager et d'exploiter les données linguistiques et un *ensemble d'outils* d'aide à l'acquisition et à la modélisation de connaissances linguistiques.

Le *moteur d'exploration contextuelle* exploite les connaissances linguistiques pour une ou plusieurs tâches choisies par l'utilisateur. Il est composé de deux systèmes qui coopèrent entre eux : l'analyseur de texte et l'étiqueteur sémantique. L'*analyseur de texte* construit une première représentation qui reflète l'organisation structurelle du texte. La construction de cette structure hiérarchisée peut s'appuyer sur des balises d'un langage du type XML, lorsqu'elles existent. En l'absence d'un tel balisage, l'analyseur fait appel à un *segmenteur* conçu par Ghassan Mourad [2001] et développé par Gustavo Crispino [2003], afin de construire cette structure. L'*étiqueteur sémantique* déclenche, pour toutes les tâches choisies par l'utilisateur, les règles d'étiquetage associées à celles-ci. Toutes les déductions effectuées par les règles sont attribuées aux éléments qui composent la hiérarchie du texte sous forme d'étiquettes sémantiques et produisent ainsi une structure hiérarchisée « décorée » par des informations sémantiques.

Les *agents spécialisés* ont pour tâche d'exploiter les « décorations sémantiques » du texte en fonction des choix de l'utilisateur. Les agents spécialisés, qui possèdent leurs propres interfaces de présentation des résultats obtenus par l'étiqueteur sémantique, permettent de développer des traitements spécifiques pour un utilisateur tout en exploitant le modèle générique de traitement des connaissances linguistiques.

La figure 1.2 présente un exemple de résumé construit par la première version de la plate-forme *ContextO*. Les tâches possibles (filtrage sémantique, relations statiques…) s'affichent dans la partie supérieure gauche de l'écran. Dans l'exemple, l'utilisateur a déclenché la tâche *Filtrage sémantique* et trois fenêtres présentent des résultats complémentaires. La fenêtre prin-

cipale affiche le résumé. Celle placée dans la partie supérieure droite, affiche, à la demande de l'utilisateur, le contexte textuel ; dans l'exemple, le paragraphe qui contient la phrase sélectionnée dans l'extrait, qui apparaît en surlignée dans la fenêtre principale. Une troisième fenêtre, en bas à droite, affiche les différentes propriétés du texte traité : titre, nombre de sections, nombre de paragraphes, etc.

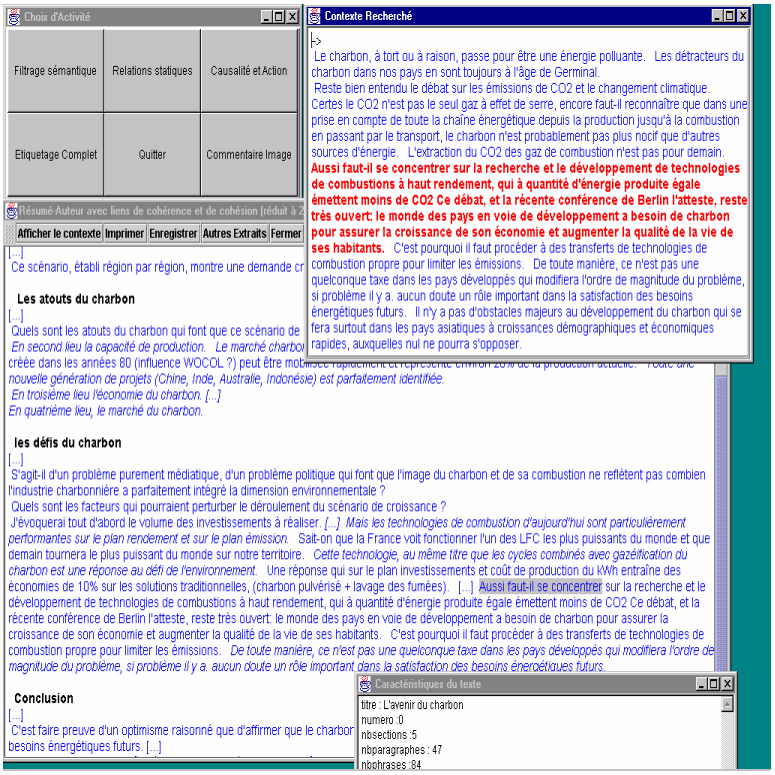

Figure 1.2 – *Première implémentation de la plate-forme ContextO (extrait de [Minel et al. 2001]).*

En matière d'interaction offerte à l'utilisateur, notons que les fenêtres ouvertes ne sont pas des composants intégrés dans un ensemble (dans une plate-forme) mais des unités indépendantes (*cf.* figure 1.3).

Lorsque le système démarre, l'utilisateur est confronté à une fenêtre (« Choix d'Activité ») afin de choisir le traitement à réaliser. Dans l'exemple, l'utilisateur choisit le filtrage sémantique. Comme résultat, la première fenêtre ouvre (c'est la sémantique des flèches dans le diagramme) deux nouvelles fenêtres : une avec les caractéristiques du texte traité ; l'autre affichant le résumé crée. L'utilisateur souhaitant examiner le contexte d'un segment textuel, il

sélectionne le segment, et ensuite il choisit l'option correspondante dans la fenêtre du résumé. Cela entraîne l'ouverture d'une nouvelle fenêtre qui affiche, en rouge, le segment textuel sélectionné, placé dans le texte source.

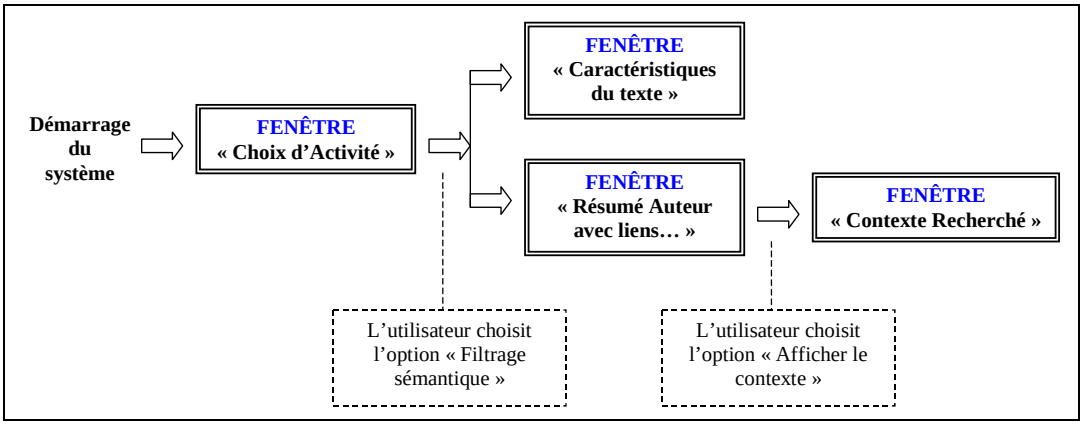

Figure 1.3 – *Diagramme de la création des composants visuels.*

En outre, la contextualisation d'un segment textuel est faite par le biais d'une recherche lexicale séquentielle dans le texte. Cette solution présente différents inconvénients, notamment elle n'assure pas le bon positionnement dans le cas où la chaîne textuelle sélectionnée se répéterait dans le texte.

Enfin, le cycle de traitement se montre figé : l'utilisateur peut choisir un seul texte, un seul profil de filtrage et, une fois les résultats du traitement affichés, il doit recommencer tout le traitement s'il souhaite changer l'un de ses paramètres. Afin de mener jusqu'au bout une approche de traitement ouverte, contraire à celle de type boîte noire, la plate-forme nécessitait une interaction plus souple et riche avec l'utilisateur.

1.3.3. Deuxième implémentation : vers la mise en œuvre d'interfaces utilisateurs

L'objectif principal pour la deuxième implémentation de la plate-forme *ContextO* a été d'avoir une interface utilisateur ergonomique tant pour l'expression de profils de filtrage que pour la présentation des extraits. Cet objectif était lié aux travaux sur la visualisation interactive de l'information [Hascoët *et* Beaudouin-Lafon 2001] [Card *et al.* 1999], et à ceux sur la présentation dynamique du contenu de documents [Boguraev *et al.* 1998]. Nous avons proposé une approche innovante pour la construction de résumés qui, sans négliger les caracté-

ristiques de l'approche traditionnelle, est plus flexible quant aux attentes de l'utilisateur sur le document résultat. Contrairement à l'approche de type boîte noire, un système fondé sur cette approche peut se caractériser comme un outil interactif qui aide l'utilisateur à obtenir le résultat souhaité. Il apportera les instruments nécessaires pour résoudre certains types de problèmes habituels dans la création d'un résumé. Par exemple, au lieu de tenter de résoudre le problème de l'incompréhension lié à l'absence du contexte, il donnera à l'utilisateur la possibilité d'accéder dynamiquement au contexte dans le texte source. Un tel système se focalise sur le développement d'interfaces graphiques.

Il faut souligner que l'une des caractéristiques communes aux systèmes de construction automatique de résumés est le peu d'importance accordé aux interfaces d'utilisateur, et plus particulièrement aux modes de représentation visuelle du résultat : un fragment textuel « résumant » le texte, affiché de manière statique. Ceci n'est pas le cas d'autres domaines, comme celui de la Recherche d'Information, où l'interaction joue un rôle important. Par ailleurs, les travaux en perception visuelle ont montré que l'être humain possède une perception initiale globale d'une scène avant de s'intéresser aux détails, ce que l'on appelle *perception gestaltique* [Mullet *et* Sano 1995]. Les recherches menées dans le domaine de la visualisation d'information, lesquelles se sont multipliés tout au long des quinze dernières années, coïncident pour affirmer que la visualisation doit être interactive pour être efficace car l'être humain est particulièrement habile pour extraire des informations dans un milieu qu'il contrôle directement et activement, par opposition à un milieu dans lequel il ne peut qu'observer passivement.

Plusieurs caractéristiques ont été décrites afin de satisfaire les objectifs fixés dans la conception de ContextO :

- permettre de définir différents profils de filtrage pour un même document ;

- relier d'une façon dynamique un document et se(s) extrait(s) ;

- permettre de modifier d'une façon dynamique un extrait à partir de la modification d'un profil de filtrage ;

- montrer le texte de deux manières : une présentation linéaire et une présentation arborescente révélant la structure du texte ;

- traiter plusieurs documents en simultané ;

- permettre d'alterner rapidement les affichages des différents extraits d'un document ;

- organiser le travail sous forme de projets : possibilité de créer, stocker et récupérer des projets ;

- offrir différents modes d'utilisation selon des niveaux d'utilisateur (*mode assisté* pour l'utilisateur non averti, *mode normal* pour l'utilisateur moyen, et *mode expert* pour l'utilisateur plus expérimenté) ;

- permettre la modification des paramètres de visualisation selon les préférences de l'utilisateur.

Toutes ces caractéristiques étaient fondées sur un ensemble de propriétés, inspirées principalement des travaux de Weinschenk *et al.* [1997] et Fowler [1998] :

- *Cohérence* : les séquences d'actions doivent être cohérentes face à des situations similaires. La cohérence implique aussi que la terminologie employée dans les messages, les menus, les écrans d'aide et l'usage des polices de caractères doivent être uniformes.

- *Minimisation de la charge de mémoire de l'utilisateur* : à titre d'exemple, celui-ci ne doit pas retenir l'information d'un écran pour l'employer dans d'autres écrans.

- *Compatibilité entre les données d'entrée et les données de sortie* : l'utilisateur doit trouver facilement les liens entre le format dans lequel les données sont fournies au système et sa sortie.

- *Flexibilité* : l'utilisateur doit pouvoir accéder à l'information d'affichage de la manière la plus conforme à ses objectifs. Par exemple, dans un affichage tabulaire, l'ordre des colonnes devrait être facilement changeable par les utilisateurs.

A partir d'une étude de cas d'utilisation présentée dans [Couto 2002], un cycle dynamique de traitement a été proposé (*cf.* figure 1.4). Notons que le nouveau cycle élimine la rigidité du cycle précédent, permettant à l'utilisateur de commencer par la sélection de documents ou par la sélection de la tâche ou par la définition des profils de filtrage. Le travail sous forme de projets offre à l'utilisateur un gain de temps considérable, car il peut sauvegarder et récupérer des scénarios de travail. De plus, à tout moment, avant ou après l'exécution de la tâche, l'utilisateur aura la possibilité de modifier un composant quelconque du projet et de relancer l'exécution.

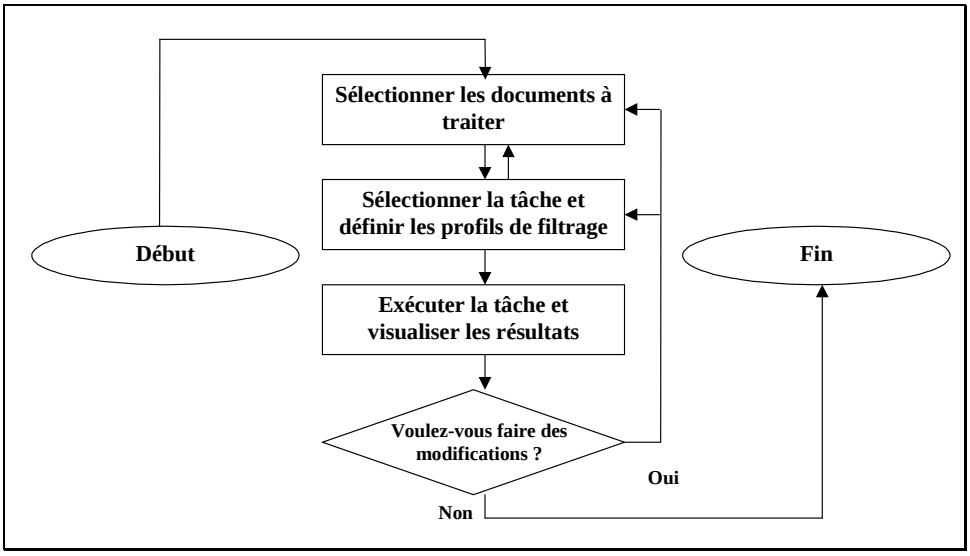

Figure 1.4 – *Cycle dynamique de traitement.*

La figure 1.5 montre un exemple de résumé construit par la deuxième version de la plate-forme *ContextO*. Dans cette interface, nous pouvons distinguer quatre fenêtres principales, signalées par les lettres A, B, C et D. La *fenêtre A* affiche la liste des documents choisis par l'utilisateur pour le projet courant. La *fenêtre B* visualise les propriétés du document sélectionné dans la fenêtre A. Ce sont des propriétés structurelles (nombre de titres, paragraphes, phrases, etc.) ou issues du filtrage de texte (nombre de phrases étiquetées, etc.), si la tâche a déjà été exécutée.

La *fenêtre C* visualise le document source en cours de filtrage et la *fenêtre D* visualise le texte filtré construit à partir de l'application d'une tâche et d'un profil de filtrage choisi par l'utilisateur. Toutes ces fenêtres sont liées de telle manière qu'une modification dans une des fenêtres se propage dans les autres.

En particulier, les fenêtres *C* et *D* présentent une interaction plus riche, qui développe l'un des aspects clé de la plate-forme : la coordination automatique entre différentes vues. Des options de visualisation (les deux menus déroulants intitulées *Texte* et *Profil*) sont offertes pour permettre à l'utilisateur de visualiser plusieurs documents et, pour chaque document, des profils différents. L'utilisateur peut choisir, à l'aide des onglets, deux types de vues différentes pour la présentation des extraits et du document source. Dans l'exemple, la fenêtre correspondant au document source affiche une vue linéaire, tandis que celle de l'extrait mon-

tre une vue arborescente permettant à l'utilisateur de visualiser rapidement dans quelles parties du texte se trouvent les phrases qui constituent l'extrait.

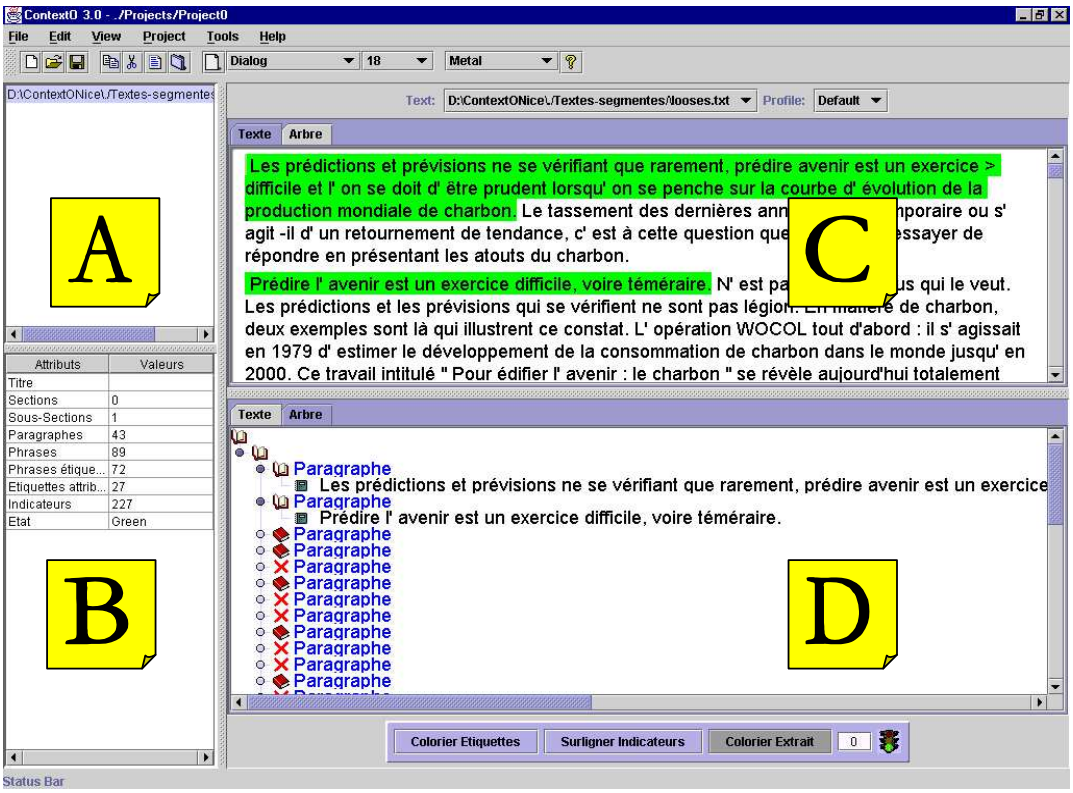

Figure 1.5 – *Deuxième implémentation de la plate-forme ContextO (extrait de [Couto 2002]).*

Le fait de positionner le curseur sur une phrase de l'extrait (fenêtre D) permet de voir dans le texte source, grâce à une coordination automatique, le contexte complet de celle-ci. La coordination est également réalisée si l'utilisateur se positionne sur une phrase du document source, exception faite dans les cas où cette phrase n'a pas été extraite pour le profil sélectionné. La dynamique entre les deux fenêtres permet ainsi à l'utilisateur de visualiser constamment le contexte d'une phrase de l'extrait, indépendamment du type de vue affichée : linéaire ou structurelle.

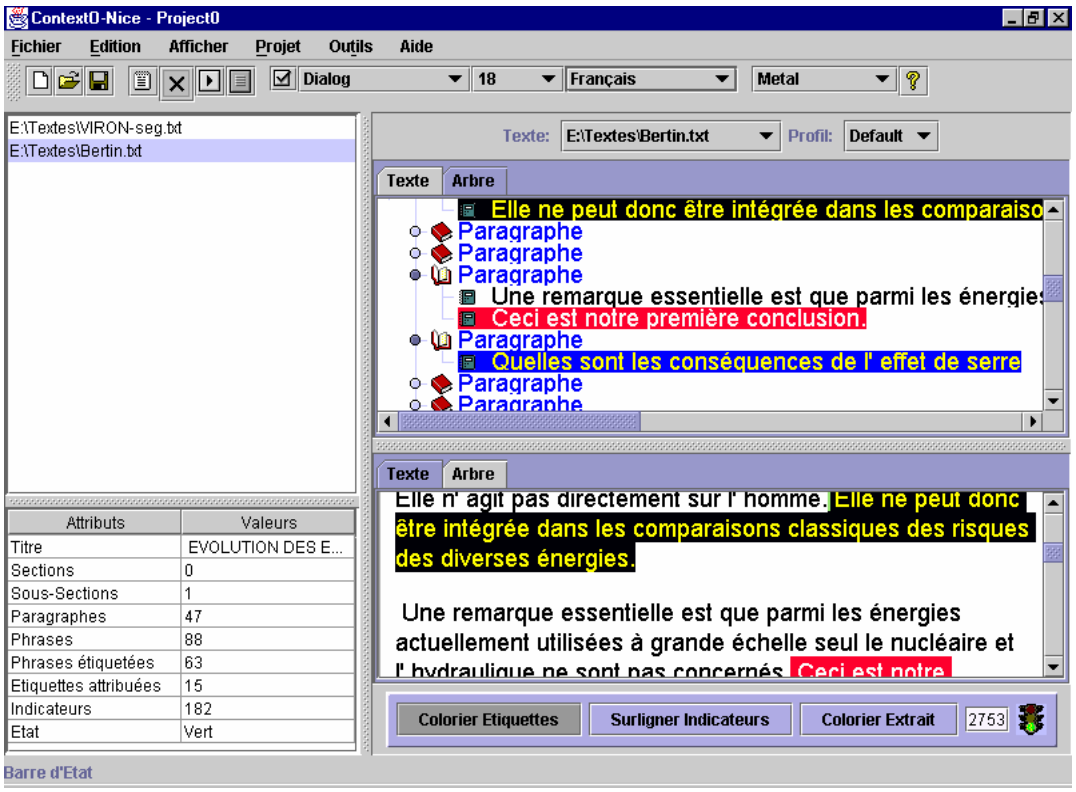

Figure 1.6 – *Visualisation des étiquettes sémantiques (extrait de [Couto 2002]).*

Quant à la visualisation des informations saillantes, elle repose sur des principes d'ergonomie utilisant les possibilités offertes par la colorisation [Murch 1985]. Les segments de texte saillants et les décorations sémantiques sont traduits par des couleurs choisies par l'utilisateur. Dans la figure 1.5, l'utilisateur a choisi de visualiser, dans le texte source, les phrases qui ont été sélectionnées. Il s'agit d'une information booléenne : cette opération indique si une phrase a été extraite ou non. Une option plus fine en ce qui concerne la granularité de l'information affichée est prévue : la plate-forme *ContextO* offre une colorisation des étiquettes sémantiques attribuées (*cf.* figure 1.6), qui permet éventuellement à l'utilisateur d'interpréter le type d'information (sa valeur sémantique) exprimé dans la phrase.

Cette présentation dynamique de l'extrait montre une approche différente de celle habituellement envisagée dans les systèmes de production de résumés automatiques. Plutôt que de mettre l'accent sur des tâches telles que la résolution d'anaphores et le repérage de liens de cohésion et de cohérence, notre objectif a consisté à construire un nouvel objet textuel constitué par les parties les plus importantes du texte source et à fournir à l'utilisateur la pos-

sibilité de se déplacer entre le texte source et l'extrait. Cette navigation permet de placer le contenu de l'extrait dans le contexte du texte source.

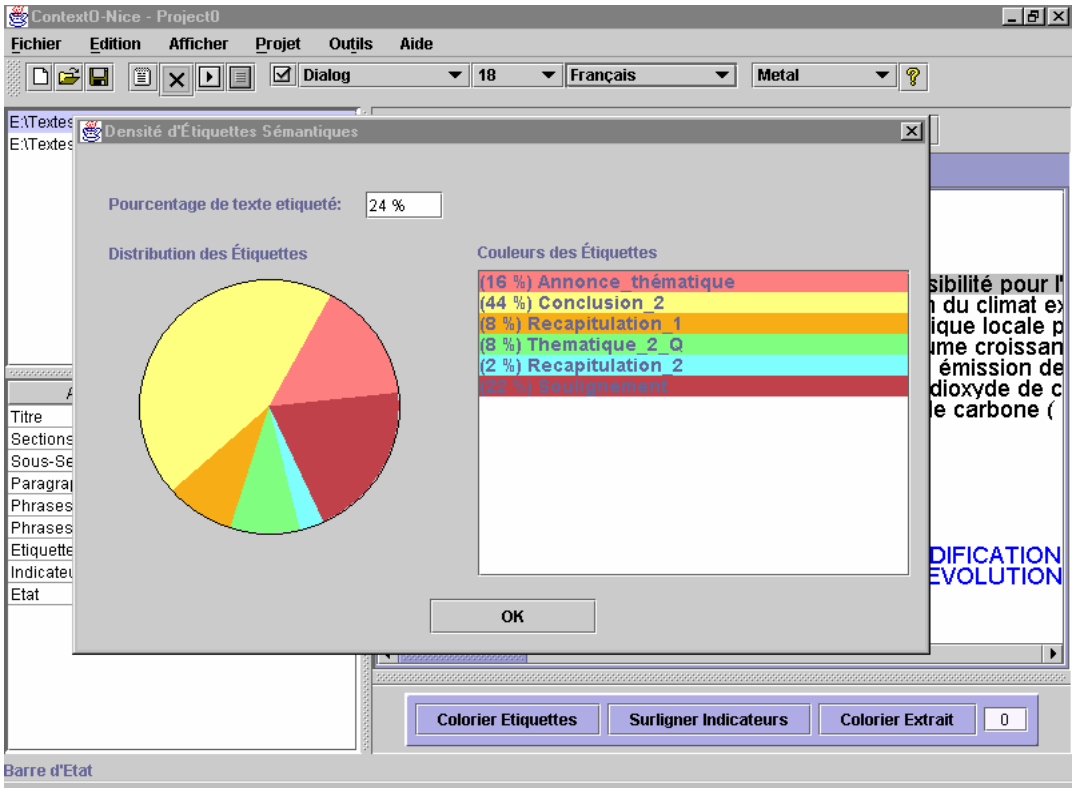

Figure 1.7 – *Distribution d'étiquettes sémantiques (extrait de [Couto 2002]).*

En outre, l'utilisateur pouvant étudier le résultat du filtrage de plusieurs textes, nous avons recherché aussi à fournir des visualisations qui exploitent les résultats quantitatifs calculés à partir des étiquettes sémantiques. Ceci peut permettre de comparer différents genres textuels ou plus spécifiquement la distribution de certaines marques sémantiques comme l'illustre la figure 1.7.

1.3.4. Le résumé automatique guidé par le « point de vue » de l'utilisateur et assisté par des interfaces adaptatives

Certains travaux portant sur l'évaluation de résumés automatiques, comme [Minel *et al.* 1997] ou [Saggion *et* Lapalme 1998], entre autres, ont montré la difficulté de réaliser des résumés standards : il n'existe pas de critères précis pour déterminer ce qui sera un bon résumé

d'un point de vue intrinsèque. En effet, ce qui est jugé pertinent pour un utilisateur ne l'est pas nécessairement pour un autre. Ces difficultés ont amené plusieurs chercheurs à explorer d'autres voies de recherche, dans lesquelles les systèmes s'adaptent mieux aux besoins spécifiques des utilisateurs : « *A l'origine de ce constat vient sans doute en premier lieu le fait qu'il n'existe pas de critères précis pour déterminer ce qu'est un bon résumé. Les résumés sont également très différents selon les utilisateurs auxquels ils sont destinés. Ainsi on ne produira pas le même résumé d'un article scientifique innovant si l'on doit adresser ce résumé à la Direction générale, au service des brevets pour consultation juridique, au laboratoire de développement, aux services de presse grand public … Les résumés dépendent également des types de textes. On ne résume pas de la même façon un texte narratif, un article scientifique relatif à une science expérimentale, un article d'une science théorique ou d'un domaine spéculatif, des articles juridiques, etc. Il n'y a donc pas de résumé idéal qui serait indépendant des demandes des utilisateurs et des types de textes. Ces différents constats ont amené les équipes de recherche à élargir le champ de leurs recherches en visant non plus de simples résumés automatiques non ciblés, mais des systèmes adaptés aux besoins spécifiques d'une tâche d'identification ou de recherche d'information. Nous appelons ces systèmes des systèmes automatiques de filtrage d'information.* » [Minel 2003].

La même problématique a été analysée dans les conférences *Document Understanding Conference* (DUC) : la discussion lors de DUC 2002 [DUC] a tourné autour du difficile problème de discerner ce qui constitue un bon résumé [Harman 2002]. Ainsi, dans DUC 2003 la tâche de résumé multi-document est similaire à celle de DUC 2002, mais les résumeurs humains sont sollicités pour déterminer le point de vue selon lequel ils rédigent le résumé.

Les résumés automatiques doivent donc répondre à des attentes spécifiques : identifier des informations innovantes, mettre en évidence des résultats ou des hypothèses dans un article scientifique ou dans un rapport technique, retrouver les déclarations d'un auteur sur une question controversée, rechercher les causes d'un phénomène, extraire les approches définitoires d'un concept, etc. Il semble donc important d'introduire la notion de « point de vue » de l'utilisateur et d'effectuer le résumé en prenant explicitement en compte ses attentes, ce qui suppose que l'on lui donne, dans un système interactif homme-machine, la possibilité d'exprimer celles-ci.

Les « points de vue » de l'utilisateur, aspect distinctif de *ContextO* par rapport à d'autres systèmes similaires, se concrétisent sous la forme de profils de filtrage. Ainsi, un *profil de filtrage* [Minel *et al.* 2001] exprime les besoins spécifiques d'information d'un utilisateur concernant un texte source. Un profil peut se concevoir comme la définition d'une *stratégie*

d'extraction et de sélection, et sa caractérisation comporte trois paramètres principaux : le *type d'information souhaitée*, la *modalité d'exploration* et le *seuil de filtrage*.

Le *type d'information souhaitée* met en évidence l'importance que l'utilisateur accorde aux différentes valeurs sémantiques susceptibles d'être repérées dans le document source. Ainsi, pour un certain type de traitement, les énoncés *conclusifs* sont considérés comme plus importants que les énoncés d'*annonce thématique* lorsqu'ils se trouvent dans la dernière section du texte. La définition du type d'information souhaitée se présente alors sous la forme d'une liste hiérarchisée de valeurs sémantiques, ce qui permet aussi d'ignorer certains types d'informations.

La *modalité d'exploration* précise l'ordre d'exploration des sections et la profondeur d'exploration (et conséquemment de sélection) du texte. Il est possible de définir diverses modalités d'exploration. En guise d'exemple, nous présentons deux modalités : une *modalité standard* qui explore le texte linéairement en sélectionnant les phrases qui correspondent au profil de filtrage et une *modalité entrelacée* dans laquelle l'introduction puis la conclusion - si elles sont présentes dans le texte - sont d'abord explorées, en tenant compte aussi de la structure en paragraphes de ces deux segments textuels ; ensuite le système poursuit une exploration linéaire du texte. La profondeur d'exploration permet d'ignorer les sections les plus profondes qui correspondent généralement à des explications détaillées de l'auteur sur un point précis. D'autres modalités peuvent être définies au gré des besoins spécifiques d'un utilisateur.

Le *seuil de filtrage* permet de produire un ensemble de phrases qui correspondent à un nombre fixé de phrases, relatif à la taille du texte source. Par exemple, le seuil de sélection peut être de 10% ou 20% du texte source. Si le pourcentage de phrases étiquetées est supérieur au seuil indiqué, la sélection est réalisée selon l'ordre de la liste hiérarchisée de valeurs sémantiques souhaitées. Dans le cas où le pourcentage de phrases étiquetées serait inférieur à celui demandé, l'utilisateur est averti de la réduction automatique du seuil de filtrage.

Dans *ContextO*, les profils de filtrage sont ou bien prédéfinis (profils par défaut), ou bien construits de manière interactive par l'utilisateur. La figure 1.8 montre un exemple de configuration de profils de filtrage. L'écran est divisé en trois sections. La première, dans le cadre supérieur, contient un ensemble de données du profil : le nom du profil, le taux de réduction et le choix des traitements spécifiques. La deuxième, dans le cadre central, contient l'information des étiquettes sémantiques de ce profil. Le cadre à gauche présente la liste de

toutes les étiquettes disponibles. Celui à droite montre la liste des étiquettes qui déterminent le profil. Cette liste est ordonnée de manière descendante selon l'intérêt de l'utilisateur. L'ordre est fixé à l'aide des deux boutons qui contiennent des icônes de flèche supérieure et inférieure. Les boutons qui contiennent les flèches gauche et droite permettent de rajouter ou de retirer des étiquettes du profil. La troisième section de l'écran (boutons *Précédent* et *Suivant*) donne la possibilité de gérer plusieurs profils dans la même fenêtre.

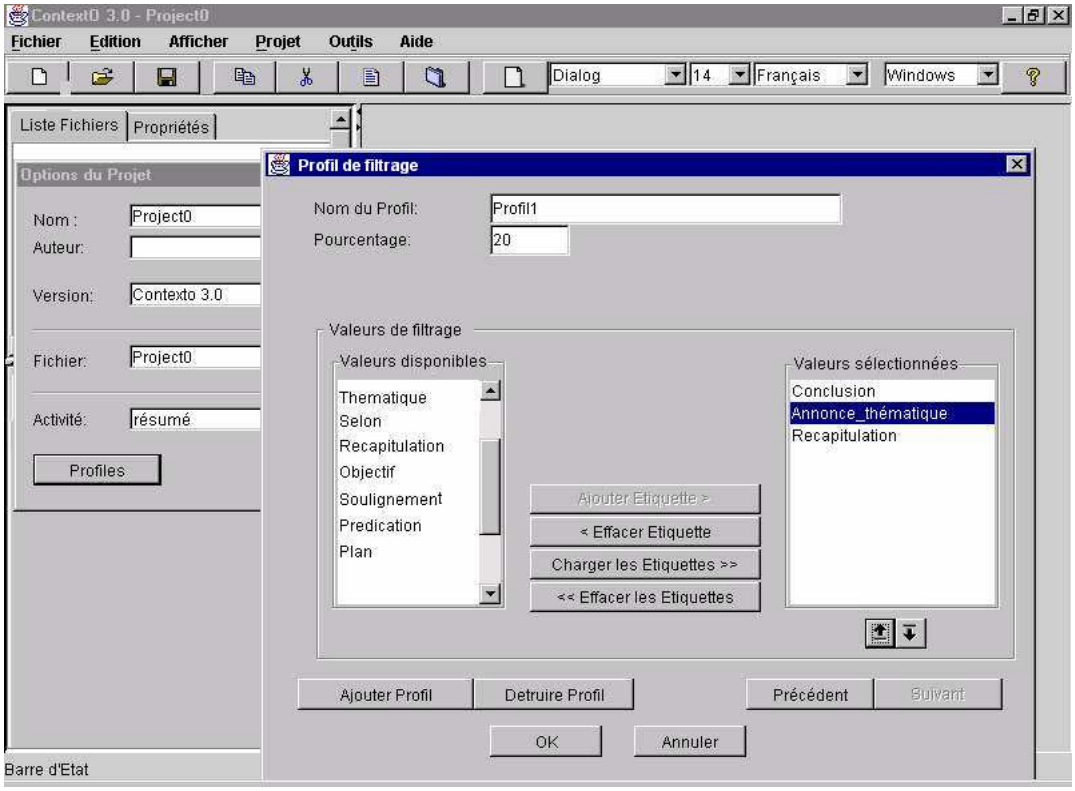

Figure 1.8 – *Construction d'un profil de filtrage (extrait de [Couto 2002]).*

1.3.5. Quelques éléments de réflexion à propos de l'intérêt d'interfaces adaptatives

Le développement et, notamment, l'utilisation de la plate-forme *ContextO* ont nourri la réflexion autour de l'importance de l'interactivité dans le traitement et, également, le bénéfice dû aux interfaces adaptatives. Les expérimentations menées avec cette plate-forme nous ont permis de constater qu'il existe un intérêt dans une approche comme celle proposée, où la notion de profil de filtrage joue un rôle fondamental. Les opérations de visualisation, la pos-

sibilité d'avoir différents types de vues et divers extraits pour un même document source, et la coordination automatique de vues, éléments d'une approche dynamique à la construction automatique de résumés [Crispino *et* Couto 2004], représentent pour l'utilisateur des outils lui permettant d'interpréter et de mieux comprendre les extraits fournis par *ContextO*.

Néanmoins, nous avons aussi pu constater que le bénéfice, pour l'utilisateur, ne se mesure pas nécessairement par un gain en temps de lecture, mais plutôt en termes de qualité de lecture. Les principes décrits mettent en avant le principe de l'interaction plutôt que celui d'une focalisation par granularité de plus en plus fine comme dans [Boguraev *et al.* 1998]. Nous pensons en effet que l'activité consistant à sélectionner des informations à l'intérieur d'un texte donné est une démarche hautement intelligente qui varie en fonction des sujets, de leur expertise du domaine traité ainsi que de leur degré d'attention au moment où ils en prennent connaissance. Actuellement cette activité n'est pas modélisable dans sa globalité. Par contre nous pouvons viser la construction de représentations d'un texte qui fournissent à l'utilisateur des repères sur les organisations discursives qui structurent le texte. Ce repérage permettra à l'utilisateur de déployer ses propres stratégies de recherche d'information. Autrement dit, il faut pallier l'absence de compréhension par l'exploitation des données structurelles et discursives.

1.4. Projet RÉGAL

Dans le cadre du projet RÉGAL[8] [Ferret *et al.* 2001] [Sabah 2002] [Couto *et al.* 2004], nous avons pu mettre à l'épreuve nos choix de visualisation et d'interaction. L'objectif principal de ce projet a consisté à fournir à l'utilisateur des outils d'exploration d'un document dont il ne peut pas juger la pertinence par rapport à ses objectifs de recherche, malgré la description présentée par un moteur de recherche ou le résumé indicatif généré par un système de résumé automatique. La problématique visée est celle des systèmes de recherche d'information, qui renvoient généralement une liste ordonnée de documents, où seulement le titre et éventuellement un extrait comportant les mots de la requête permettent à l'utilisateur d'en évaluer la pertinence par rapport à son besoin initial. Ces types de résultat conduisent souvent

[8] Le projet RÉGAL (REsumé Guidé par les Attentes du Lecteur), financé par l'Action Concertée Incitative Cognitique 2000 (LAC038), a été mené conjointement par le CEA, le laboratoire LaLICC, le LATTICE et le LIMSI.

celui-ci à devoir consulter de nombreux documents pour réellement trouver des réponses pertinentes.

1.4.1. Structure des textes et exploration des documents

L'approche adoptée en RÉGAL se fonde sur l'hypothèse selon laquelle la structure des textes représente un support pertinent pour l'exploration des documents. Plus particulièrement, l'accent est mis sur la structure thématique des documents, sous-entendu que, grâce à sa nature hiérarchique, cette structure offre une vue synthétique du contenu permettant de la parcourir afin d'accéder aux différents niveaux de granularité thématique.

Les informations nécessaires à la visualisation sont extraites automatiquement des textes par l'application d'une analyse thématique, sans présupposer l'existence d'une structure préalable ou d'un formatage du texte, et en combinant des approches fondées sur la cohésion lexicale et le repérage de marques clés [Ferret *et al.* 2001] [Couto *et al.* 2004]. Néanmoins, cette analyse ne s'applique pas à tous les types de documents : le genre des documents traités est limité aux articles scientifiques ou techniques ainsi qu'à certains articles journalistiques, textes dits *expositifs*, par opposition à des textes plus *narratifs*. Par ailleurs, nous pouvons distinguer deux modules d'analyse (*cf.* figure 1.9) : l'un consacrée à la segmentation thématique et l'autre au repérage des cadres thématiques (analyse linguistique).

Ces deux modules analysent le texte en parallèle. Le premier délimite des segments textuels (paragraphes) considérés comme fortement cohérents du point de vue du thème qu'ils développent, en s'appuyant sur la cohésion lexicale des textes. Les informations de segmentation – début et fin de segment – sont décrites dans un format XML [Hernandez *et* Grau 2002] défini par rapport à une *tokenisation* de référence du texte considéré. Les thèmes d'un texte sont calculés par des groupes nominaux extraits du texte. Le deuxième module identifie les débuts de cadre thématique [Porhiel 2003] du texte analysé et des informations considérées comme structurantes du point de vue discursif. La sortie de ce module est un texte également annoté au format XML. Les annotations sont essentiellement des balises de début de segment textuel.

Après ces deux analyses, un module d'intégration ajuste les segments trouvés en fonction des marques linguistiques présentes.

Le module de structuration repose sur le même principe de cohésion lexicale que celui utilisé lors de la segmentation initiale. Chaque segment est décrit par un vecteur de mots et une mesure de similarité permet d'évaluer la proximité entre les segments non contigus. Les

segments les plus similaires sont regroupés et on considère que les segments inclus entre ceux-ci sont à un niveau hiérarchique inférieur, ce qui signifie que des thèmes différents ont été introduits lors du développement d'un thème. Cette inclusion peut correspondre à un changement de thème, une digression ou une spécialisation. Ce mode de structuration respecte la linéarité du texte mais seuls les liens entre segments se rapportant à un même sujet sont marqués ; les liens de dépendance entre sujets différents ne sont pas typés.

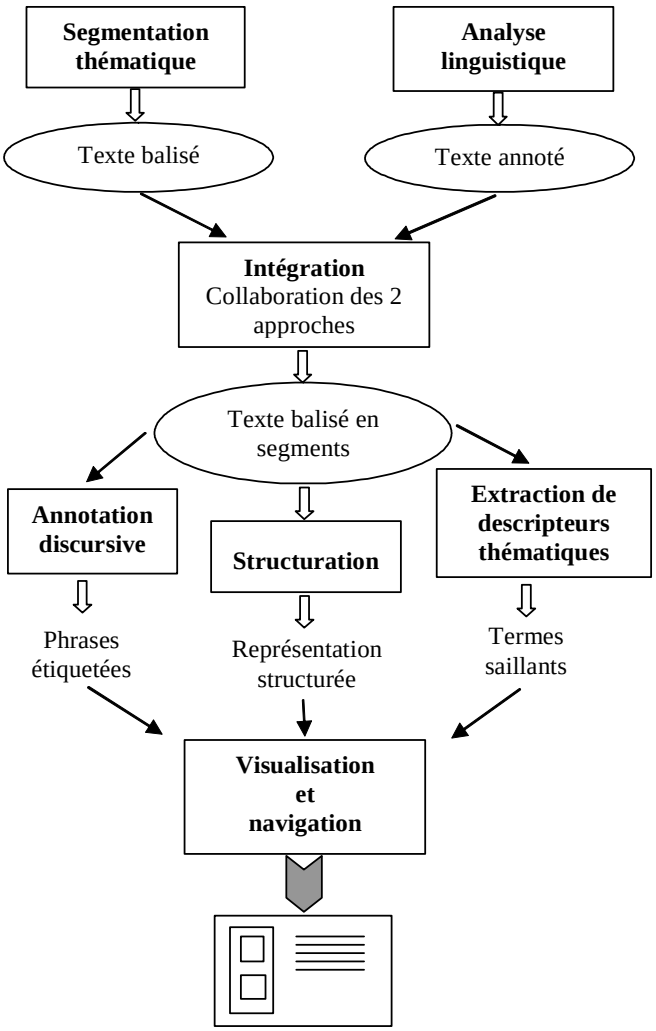

Figure 1.9 – *Architecture du système RÉGAL (extrait de [Couto et al. 2004]).*

Le résultat de cette analyse étant censé se conformer aux standards d'entrée de la plateforme ContextO, il a été nécessaire de réaliser certaines extensions à celle-ci de manière à ce

qu'elle puisse exploiter les nouveaux types d'information. D'une part, il a fallu étendre le modèle de texte proposé par Crispino [2003] afin de lui incorporer les informations issues de l'analyse thématique. D'autre part, nous avons étudié comment mettre visuellement en évidence la structuration thématique d'un document. La solution à cette question n'a pas été évidente car si nous rajoutions celle-ci aux types de vues existantes (*texte plat* et *arborescente*), nous risquions de noyer l'utilisateur avec un excès d'information. En même temps, si nous créions un nouveau type de vue, le risque devenait ergonomique car cela requérait de présenter une nouvelle vue dans la plate-forme et de définir une nouvelle forme de coordination entre les différentes vues d'un même document.

1.4.2. Conception des outils de visualisation et navigation

Notre participation dans le projet RÉGAL nous a permis, d'une part, d'abstraire la notion de vue par rapport à un document et, d'autre part, d'incorporer la notion *d'opération de navigation*. C'est ainsi que notre conception des outils de visualisation et navigation repose sur un principe général concernant l'objet textuel à produire : à partir de l'objet textuel source T, le résultat de la fouille du texte est un nouvel objet textuel T_f qui intègre d'une part l'objet T et une représentation décorée de T appelée T_d (autrement dit, le résultat de la fouille est une fonction de T et de T_d), et d'autre part un ensemble de vues étant donné que le système permet d'avoir plusieurs vues pour un même texte (une vue étant une forme sémiotique FS du texte T_d, celle-ci peut se concevoir comme une fonction de FS et de T_d). Nous pouvons alors décrire le rôle des vues dans la fouille de textes avec l'expression suivante :

$$T_f\,(T,\, T_d,\, \{V_1(FS_1,\, T_d),\, V_2(FS_2,\, T_d),\, \ldots\, V_n(FS_n,\, T_d)\ \})^9$$

La notion de *coordination automatique* des vues consiste alors à répercuter automatiquement un déplacement effectué par l'utilisateur dans V_i sur les autres vues associées au texte décoré T_d.

1.4.3. Utilisation de ContextO dans le cadre du projet RÉGAL

[9] Les accolades dénotent un ensemble d'éléments. Les éléments entre parenthèse, séparés par des virgules, révèlent la dépendance entre les différents éléments.

La figure 1.10 illustre les extensions effectuées à la plate-forme ContextO au sein du projet RÉGAL. Les différentes fenêtres ont été réorganisées :

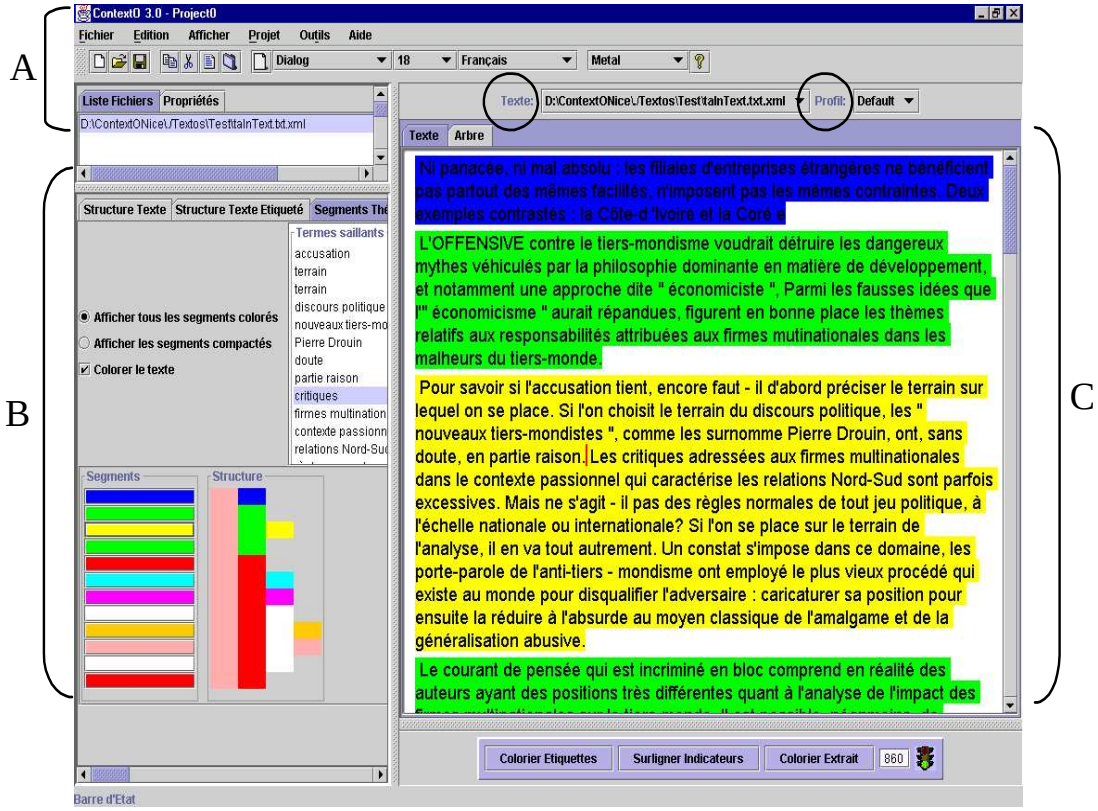

Figure 1.10 – *Utilisation de ContextO dans le projet RÉGAL (extrait de [Couto et al. 2004]).*

La fenêtre de propriétés du document (*cf.* figure 1.5) a été déplacée, rassemblée avec la fenêtre de la liste de documents à traiter. Son ancienne place, signalée par la lettre B, a été utilisée afin d'offrir différentes vues décorées du texte source T, accessibles par des onglets spécifiques. Ces décorations sont le résultat des calculs effectués (repérage des segments thématiques, fréquence des termes saillants, étiquette discursive attribuée à une phrase, etc.) et offrent des points d'accès aux différentes parties du texte. Chaque segment thématique identifié est visualisé sous la forme d'un rectangle coloré, les segments liés à un même thème étant de la même couleur. Sa taille en nombre de phrases est disponible afin d'évaluer ce segment ou ce thème par rapport aux autres thèmes du texte. La visualisation de la structure permet de mieux indiquer l'organisation des segments entre eux. L'inclusion d'un segment dans un

autre signifie qu'il y a un lien thématique entre eux et permet de visualiser ainsi sa place par rapport aux autres thèmes.

Pour chaque segment, il est possible de visualiser la liste des termes saillants avec leur fréquence. D'autres onglets permettent d'obtenir une vue hiérarchique du texte, organisé en arbre, ou une vue globale des annotations attribuées aux phrases. Cette fenêtre offre donc différents résumés du texte : selon les thèmes, selon les marques discursives, ou selon sa structure logique. Il est ainsi possible de prendre connaissance globalement du contenu du texte sans avoir à le lire et d'approfondir seulement certains points si des parties semblent intéressantes.

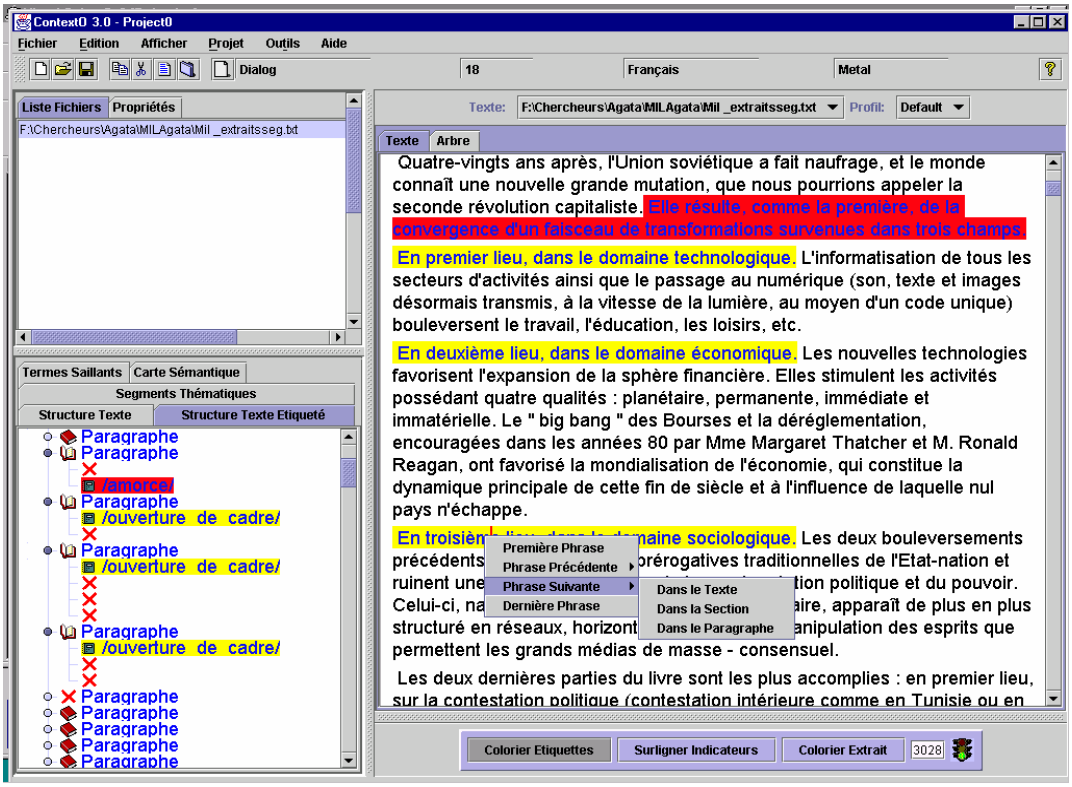

Figure 1.11 – *Opérations de navigation développées dans ContextO pour le projet RÉGAL (extrait de [Couto et al. 2004]).*

La figure 1.11 montre un autre type de vue incorporé dans ContextO[10]. Il s'agit d'une vue de type arborescente où les feuilles ne contiennent pas de chaînes lexicales renvoyant au texte source, mais les annotations attribuées à celui-ci. Cette vue étant coordonnée avec la vue principale du texte source, elle s'est avérée particulièrement utile pour parcourir rapidement les cadres thématiques du texte.

Afin d'essayer une autre approche au parcours du texte, nous avons proposé des opérations de navigation entre les éléments textuels ayant la même annotation. Dans l'exemple, un menu « *popup* » offre à l'utilisateur le choix de se déplacer vers les phrases annotées comme *ouverture de cadre*. Celui-ci peut aller à la première ouverture de cadre thématique, à la précédente, à la suivante ou bien à la dernière. Ce déplacement peut se limiter au texte complet ou à la section ou au paragraphe auxquels la phrase sélectionnée appartient.

L'incorporation des opérations de navigation permet de reformuler l'expression présentée auparavant de la manière suivante :

$$T_f\,(T,\, T_d,\, \{V_1(FS_1,\, T_d,\, O_1),\, V_2(FS_2,\, T_d,\, O_2),\, \ldots\, V_n(FS_n,\, T_d,\, O_n)\,\})$$

Les éléments O_1, O_2, …, O_n représentent des ensembles d'opérations de visualisation et sont donc associés à des vues. Cela signifie que chaque vue est une fonction d'une forme sémiotique, d'un texte décoré et d'un ensemble, spécifique à celle-ci, d'opérations de visualisation. La fouille du texte T reste fonction de celui-ci, de ses décorations (T_d) et des vues possibles sur lui.

1.5. Synthèse

Le travail accompli au sein du projet RÉGAL nous a permis de souligner l'importance des aspects de visualisation et d'interaction pour un système de fouille de textes. Parallèlement, nous avons commencé à formaliser ceux-ci, ce qui nous a amené à conclure qu'il s'agissait des connaissances amalgamées à la plate-forme car, de fait, tant les représentations visuelles

[10] L'implémentation logicielle de cette vue a été prise en charge par F. Prévitali et A. Tessier, qui ont participé aux développements des interfaces dédiées au projet dans le cadre de leur stage respectif de l'IIE et du DESS ILSI de l'Université Paris-Sorbonne.

du texte que les opérations de navigation sont spécifiques et encodées dans l'applicatif. Afin d'abstraire ces connaissances des systèmes, un langage formel de représentation de ces connaissances devait être proposé.

Dans RÉGAL, les opérations de navigation étant une alternative plus riche à la coordination automatique de vues, le système permet à un lecteur de parcourir dynamiquement le texte en passant d'un type d'information à un autre. C'est ce parcours que nous avons dénommé initialement *navigation textuelle* [Minel 2002] [Couto *et* Minel 2004$_a$, 2004$_b$]. Ainsi, il y a potentiellement pour un même texte une multiplicité de parcours de navigation que l'on peut, en partie, comparer au cheminement déambulatoire dans les hyperdocuments proposé par [Géry 2002]. Ces principes de navigation se distinguent néanmoins de la navigation hypertextuelle car les opérations de navigation s'appuient sur les marques sémiotiques et linguistiques présentes dans le texte. La navigation proposée n'est donc pas guidée par l'auteur du texte comme dans le cas de la navigation hypertexte où les hyperliens sont placés par cet auteur, mais elle est le résultat de l'interprétation de certaines informations discursives repérées dans le texte.

Il faut attirer l'attention sur le fait que notre hypothèse, au sein du projet RÉGAL, consiste à considérer la fouille de textes comme un processus de recherche d'information guidé par les besoins spécifiques de l'utilisateur. Dans ce processus jouent également des structures discursives qui, selon son contexte d'utilisation, peuvent guider l'utilisateur vers l'information recherchée.

Chapitre 2

Interfaces Homme-Machine et Visualisation Interactive d'Information

2.1. Introduction

La nature de ce travail de thèse mérite un chapitre dédié aux interfaces homme-machine (*IHM*), en particulier à la visualisation interactive d'information, l'objectif étant de donner une vue globale d'un domaine de recherche assez vaste et complexe. Je présente donc une introduction brève aux objectifs du domaine des IHM, les différents styles d'interaction homme-machine, quelques principes pour le développement d'interfaces utilisateur et le rôle de la visualisation dans les systèmes d'information.[11]

2.2. Définition et objectifs des IHM

Il n'existe pas actuellement un consensus sur ce que recouvre le domaine des IHM. Faute d'une définition formelle du domaine, on trouve dans [SIGCHI] la caractérisation suivante : *« Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them. »*.

D'un point de vue scientifique, l'intérêt principal du domaine est lié spécifiquement à l'interaction entre un ou plusieurs êtres humains et une ou plusieurs machines. Les scénarios d'interaction sont multiples lorsque l'on aborde la notion de machine. Par exemple, au lieu de s'agir d'une interaction typique entre un utilisateur et un ordinateur, on peut concevoir

[11] Ce chapitre est une version condensée, corrigée et actualisée du chapitre 2 de mon mémoire de *Master* [Couto 2002].

l'interaction avec le panneau de contrôle d'un avion ou avec un four à micro-ondes, qui se servent d'un logiciel sous forme de système embarqué.

L'IHM est donc un domaine interdisciplinaire mettant en jeu, entre autres, des disciplines tels les *sciences informatiques* (conception d'applications et ingénierie des interfaces humaines), la *psychologie*[12] (utilisation de théories concernant les processus cognitifs et l'analyse empirique du comportement humain), la *sociologie* et l'*anthropologie* (interaction entre la technologie, le travail et les organisations), et le *dessin industriel* (produits interactifs). Parmi ses intérêts spécifiques, on trouve l'étude de :

- la performance combinée dans des tâches auxquelles participent des êtres humains et des machines ;

- la structure de la communication entre les êtres humains et les machines ;

- les capacités humaines pour utiliser des machines ;

- l'algorithmique inhérente au développement d'interfaces ;

- l'ingénierie appliquée au dessin et construction d'interfaces : le processus de spécification, dessin et implémentation d'interfaces.

Si l'on restreint la notion de *machine* à celle d'*ordinateur*, on obtient le domaine d'interaction homme-ordinateur (IHO), qui est également un domaine extrêmement vaste, dans lequel participent des sujets hétérogènes tels l'interaction avec dispositifs (souris, clavier, etc.), le dessin des menus et formulaires, des théories cognitives et des modèles de performance humaine, le dessin assisté par ordinateur (*DAO*), l'utilisation de graphiques, etc.

Dans le cadre de cette thèse, *Sextant*, un langage de modélisation des connaissances visuelles et navigationnelles (*cf.* chapitre 8) a été proposé, et une plate-forme logicielle de navigation textuelle nommée *NaviTexte* (*cf.* chapitre 12) a été implémentée. Je m'intéresse en conséquence aux aspects généraux des IHO, ayant un intérêt majeur avec ceux liés à la visualisation interactive d'information.

L'un des objectifs étant le dessin d'interfaces utilisateur, la qualité des interfaces est devenue cruciale. L'expert dans le domaine des IHM Ben Shneiderman [1998] indique que : *« Well designed, effective computer systems generate positive feelings of success, competence, mastery, and*

[12] Certains chercheurs reconnus ont cependant mis en question l'applicabilité des théories résultantes de la psychologie cognitive [Landauer 1991].

clarity in the user community. When an interactive system is well-designed, the interface almost disappears, enabling users to concentrate on their work, exploration, or pleasure. »

Les objectifs du dessin des interfaces utilisateurs varieront selon la communauté d'utilisateurs à laquelle l'outil s'adresse. Shneiderman [1998] identifie cinq facteurs humains mesurables afin de réaliser une évaluation d'une interface utilisateur :

- le temps d'appropriation de l'outil ;

- la performance de l'outil ;

- le taux d'erreurs des utilisateurs ;

- la rétention dans le temps de la maîtrise de l'outil ;

- la satisfaction subjective.

Bien qu'un dessinateur d'interfaces puisse souhaiter réussir dans tous les points cités, occasionnellement la tentative de le faire génère des conflits. Dans certaines applications, la satisfaction subjective peut signifier la clé du succès, tandis que dans d'autres, un apprentissage à court terme ou une performance excellente peuvent être indispensables, même si l'utilisateur considère l'outil difficile et cela diminue sa satisfaction subjective. En conséquence, les objectifs du dessin des interfaces utilisateurs sont étroitement liés à la communauté d'utilisateurs.

2.3. Styles d'interaction

Dans [Shneiderman 1992, 1998], cinq styles différents d'interaction primaire homme-machine sont identifiés. Le premier constitue la *manipulation directe*, où l'utilisateur peut manipuler directement une représentation visuelle du monde. Parmi les exemples de ce style d'interaction on peut citer les systèmes de contrôle de trafic aérien, certains systèmes de dessin graphique (du genre AutoCAD), des jeux de vidéo et la technique « *drag and drop* » offerte actuellement par une grande diversité de logiciels. En utilisant des représentations visuelles des objets et des actions (nommées *métaphores*), les utilisateurs accomplissent rapidement leurs tâches et observent les résultats de leurs actions de manière immédiate. Les entrées via un clavier, commandes ou choix dans des menus sont remplacées par l'utilisation de dispositifs auxiliaires (par exemple la souris ou un volant) permettant de sélectionner des éléments d'un ensemble d'objets et d'actions, ou de générer des actions directement.

Un deuxième style d'interaction est la *sélection dans un menu*. Ici, si la terminologie utilisée est adéquate et les options résultantes significatives, les utilisateurs peuvent accomplir leurs

tâches avec peu d'apprentissage ou de mémorisation, et en exécutant des actions. Les points cruciaux sont l'analyse des fonctionnalités offertes par un menu et le choix de la terminologie à utiliser.

Lorsqu'il est nécessaire la saisie de donnés au système, la *saisie de formulaires* devient le style le plus approprié. Il est important que l'utilisateur comprenne parfaitement les libellés associés aux champs à remplir et qu'il connaisse les valeurs acceptées (texte libre, entiers, dates, etc.).

Le *langage de commandes* est le quatrième style d'interaction identifié et il s'avère spécialement utile dans le cas d'utilisateurs experts car ceux-ci expérimentent une forte sensation de maîtrise du système. Les utilisateurs apprennent la syntaxe et la sémantique du langage et ils peuvent généralement exprimer des requêtes complexes rapidement, sans avoir besoin de lire exhaustivement la documentation. Néanmoins, les taux d'erreurs sont habituellement élevés et la rétention dans le temps plutôt pauvre.

Enfin, le *langage naturel* est considéré le dernier style d'interaction, d'après lequel l'utilisateur exprime ses requêtes ou questions en langage naturel, auxquelles le système répond. Bien que tentante, les résultats ont été jusqu'à présent mitigés sauf dans le cas de systèmes où le lexique et la grammaire du langage sont restreints.

2.4. Comment dessiner l'interface utilisateur ?

Au-delà de l'intuition ou du « bon sens », un dessinateur d'interfaces peut se servir d'outils allant des *théories de haut niveau* et *des modèles cognitifs* jusqu'à des *guides pratiques spécifiques*, tout en passant par des *principes de dessin*. Dans le processus de conception des interfaces de la plate-forme NaviTexte (*cf.* chapitre 12) j'ai utilisé certaines guides pratiques [Weinschenk *et al.* 1997] [Fowler 1998], et j'ai suivi notamment un ensemble de principes de dessin, que je présente par la suite.

2.4.1. Principes généraux de dessin

On trouve dans la bibliographie de nombreuses références portant sur des principes de dessin d'interfaces, que l'on peut tracer jusqu'aux travaux de Norman [1988], à ceux de Nielsen [1994] et à ceux de Shneiderman [1998]. Qu'ils soient présentés sous forme d'heuristiques ou sous forme de « règles d'or », les principes sont parfois similaires, parfois complémentaires. Je décris ceux qui ont eu un impact direct dans mon travail :

- *assurer la consistance* : établir des séquences consistantes d'actions en réponse à des situations identiques ; utiliser une terminologie uniforme dans les messages, les menus et les écrans d'aide ; utiliser des couleurs et polices de manière homogène ;

- *permettre l'utilisation de raccourcis* : destinés en particulier aux utilisateurs experts, afin qu'ils puissent minimiser le nombre d'interactions nécessaires pour accomplir une tâche ; les raccourcis, les touches spéciales, les commandes cachées et l'utilisation de macros sont largement appréciés par les utilisateurs fréquents ;

- *réduire la charge de mémoire dans le court terme* : pour les êtres humains, il existe une limitation dans le traitement de l'information dans le court terme ; cela favorise les affichages simples d'information, permettant une pagination multiple où les données d'une page (ou écran) ne devraient pas être nécessaires dans une autre.

Il est clair que ces principes sont généraux et qu'ils doivent s'adapter à chaque environnement de travail.

2.4.2. Affichage de l'information

La manière dont un système affiche l'information a un grand impact sur les utilisateurs. Il existe plusieurs études empiriques mesurant les répercutions dues aux changements dans l'affichage de l'information. En guise d'exemple, dans [Tullis 1981] est présentée une étude des laboratoires Bell sur les conséquences du changement dans la représentation d'un ensemble d'information d'un système téléphonique. L'information, présentée initialement de manière narrative, a commencé à s'afficher de manière structurée. Les évaluations du changement ont montré que les employés, qui utilisaient 8,3 secondes, en moyenne, dans la résolution d'une tâche, sont passés à utiliser 5 secondes avec l'information structurée. Bell a estimé une économie de 79 ans/personne pendant la vie du système.

Un fort degré d'expertise des utilisateurs ne surmonte pas nécessairement les problèmes occasionnés par un affichage non optimisé. Je prends comme exemple l'étude présentée dans [Burns *et al.* 1986], concernant l'information affichée dans les vaisseaux spatiaux à la NASA[13]. Les libellés ont été regroupés selon des critères de « tâche similaire », et ceux-ci ont été redéfinis en utilisant une indentation et des soulignements adéquats, en alignant des valeurs numériques et en supprimant des caractères « rares ». L'étude montre qu'une population de 16

[13] N.A.S.A.: National Aeronautics and Space Administration.

employés techniques a réduit le temps de travail de 31% et le taux d'erreur de 28%, en moyenne.

Le travail [Schmid et Baccino 2002] constitue un dernier exemple visant illustrer l'importance de la manière dont l'information est affichée. L'article présente une expérimentation dont l'objectif est de déterminer le rôle de la forme visuelle d'un texte dans la phase d'intégration involucrée dans la construction d'une représentation mentale cohérente. Dans l'expérimentation, 23 participants doivent lire individuellement des textes sur un écran. En utilisant des techniques d'oculométrie, des données concernant le processus de lecture ont été capturées.

D'une part, d'un total de 24 textes à lire, chacun comportant quatre lignes, la moitié présentait un changement de perspective dans le narrateur à la troisième ligne. D'autre part, la moitié des textes présentait un espace de tabulation à la troisième ligne, de manière à ce qu'il y eut 6 textes dans chacune de quatre possibles conditions. Voici un exemple de texte, version formatée et non formatée :

Présentation formatée

Un bruit sourd parcourt les murs du bâtiment et s'estompe peu à peu.

Un athlète au corps musclé et au torse bronzé soulève des haltères.

Longuement, / il(je) lance / un soupir en s'(m')essuyant le front.

Il fait chaud, le gymnase manque d'aération.

Présentation non formatée

Un bruit sourd parcourt les murs du bâtiment et s'estompe peu à peu.

Un athlète au corps musclé et au torse bronzé soulève des haltères.

Longuement,/ il(je) lance /un soupir en s'(m')essuyant le front.

Il fait chaud, le gymnase manque d'aération.

Tableau 2.1 – *Exemple de texte formaté et non formaté (extrait de [Schmid et Baccino 2002])*

Les résultats de l'expérimentation montrent, premièrement, que les lecteurs regardent plus de temps la troisième ligne lorsqu'elle est indentée. Il en résulte, d'après les auteurs de l'article, que le formatage du texte est pris en considérations par les lecteurs. Deuxièmement, le résultat le plus important est le fait que lorsque la troisième ligne est indentée, le change-

ment de perspective du narrateur a un impact de manière immédiate sur le lecteur. Les auteurs ont conclu que le formatage visuel d'un texte semble aider les lecteurs dans le processus cognitif de détecter des changements dans la cohérence narrative.

2.4.2.1. Principes de dessin dans l'affichage de l'information

Trois principes s'ajoutent aux principes généraux :

- *assimilation efficace de l'information* : le format des données devrait être familier à l'utilisateur, et lié aux taches pour lesquelles ces données sont nécessaires ;

- *compatibilité de l'affichage de données et de la saisie de celles-ci* : le format d'affichage de l'information devrait être clairement lié au format dans lequel elle a été saisie ;

- *flexibilité* : l'utilisateur devrait être capable de configurer l'affichage de l'information. Par exemple, dans un affichage tabulaire, l'ordre des colonnes et le tri de donnés devraient être facilement manipulables par l'utilisateur.

Ces trois principes sont également valides en tout ce qui concerne les processus de saisie de données.

2.4.3. Organisation des menus

Lorsque des stratégies de manipulation directe ne peuvent pas être mises en œuvre, l'utilisation de menus s'avère une alternative attractive. L'effectivité de ceux-ci est liée au fait d'offrir à l'utilisateur des options significatives au lieu de l'obliger à se rappeler d'une syntaxe complexe de commandes. Dans le processus de conception d'un système de menus, au moins trois points sont à considérer. Le premier point consiste à définir *comment regrouper les options*. Dans ce sens, on peut considérer qu'il est nécessaire de :

- *créer des groupes d'éléments logiquement similaires.* Par exemple, un menu affichant les continents au premier niveau, les pays au deuxième, les régions au troisième, etc. ;

- *créer des groupes couvrant toutes les possibilités*. Par exemple, pour des options d'age : « 0 à 17 », « 18 à 29 » y « supérieur à 30 » ;

- *s'assurer que les éléments ne se chevauchent pas*. Par exemple, dans le menu précédent, l'existence des options « 0 à 18 » et « 18 à 29 ».

Le deuxième point a trait à la *séquence d'affichage des éléments*. Si ceux-ci ont un ordre naturel, tels que les jours de la semaine ou les valeurs d'une monnaie, alors la décision est évidente. Souvent, on peut utiliser un ordre chronologique, numérique ou selon certaines propriétés communes aux éléments (taille, are, volume, température, vitesse, etc.). Dans d'autres

cas, le concepteur est obligé de choisir un autre critère. Parmi les critères possibles on trouve l'*ordre alphabétique*, le *regroupement par fonctionnalités*, les *éléments les plus utilisées en premier lieu*, les *éléments les plus importants en premier lieu*, entre autres.

Dans [Card 1982], un groupe d'utilisateurs est confronté à un menu comportant 18 options distribuées linéairement selon trois critères : *ordre alphabétique*, *regroupement fonctionnel* et *ordre aléatoire*. Quant aux options, il s'agit de commandes d'édition de textes, tels que « *insérer* », « *italique* », « *centrer* », etc. L'expérimentation consistait à indiquer une option aux utilisateurs, puis à mesurer le temps de réponse (*i.e.* le temps mis à sélectionner l'option correcte du menu). Les utilisateurs faisant 86 essais pour chaque type de distribution, les résultats ont montré que la moyenne du temps de réponse en secondes a été de 0,81 pour l'ordre alphabétique, de 1,28 pour le regroupement fonctionnel et de 3,23 pour l'ordre aléatoire.

Le troisième point réside dans la nomenclature utilisée afin de libeller les éléments d'un menu. Il est important d'utiliser une terminologie familière, car elle est plus compréhensible, mais cohérente. C'est-à-dire que le concepteur doit s'assurer que les différentes options peuvent être distinguées par l'utilisateur. Pour ce faire, il est utile de décrire chaque option en commençant par un mot qui l'identifiée. Dans ce sens, il est plus facile, par exemple, de discriminer une option libellée « taille de la police » qu'une option libellée « définir la taille de la police » [Shneiderman 1998].

2.5. Visualisation interactive d'information

Le domaine de la *visualisation interactive d'information* est relativement nouveau, motivé principalement par la croissance des données accessibles de manière informatique et par les restrictions de visualisation imposées par les dispositifs de poche (PDA). Ce domaine a pour but de pourvoir aux utilisateurs des descriptions visuelles de vastes espaces d'information, sous l'hypothèse que : «*Humans are highly attuned to images and visual information* » [Baeza et Ribeiro 1998].

Dans [Hascoët et Beaudouin-Lafon 2001], les auteurs présentent la caractérisation suivante : « *Le but de la visualisation d'information est d'exploiter les caractéristiques du système visuel humain pour faciliter la manipulation et l'interprétation de données informatiques variées. Les travaux en perception visuelle ont montré que l'être humain a une perception d'abord globale* (gestalt-perception) *d'une scène, avant de porter son attention aux détails (Myers, 2000).* »

Les travaux dans ce domaine se sont intensifiés pendant les dernières années. En général, ceux-ci coïncident sur le fait que l'efficacité de la visualisation interactive d'information est

fonction de l'interactivité, car « *l'être humain est particulièrement habile à extraire des informations d'un environnement qu'il contrôle directement et activement par rapport à un environnement qu'il ne peut qu'observer de manière passive.* » [Hascoët et Beaudouin-Lafon 2001]

Je présente par la suite quelques techniques communes du domaine.

2.5.1. Brushing and linking

La technique de « *brushing and linking* » est une technique d'interaction qui consiste à lier différentes vues de la même information, de manière à ce qu'un changement dans une vue répercute sur les autres [Ward 1994]. En guise d'exemple d'utilisation de cette technique, dans [McDonald 1990] la sélection d'un item dans une vue entraîne la sélection du même item dans les autres vues.

2.5.2. Panning and zooming

La technique « *panning and zooming* » simule les actions d'une caméra pouvant se mobiliser autour d'une scène (« *panning* ») et pouvant aussi s'en rapprocher et s'en éloigner (« *zooming* »). Un exemple bien connu de systèmes utilisant cette technique le constitue les lecteurs de fichiers PostScript ou PDF, qui permettent amplifier et diminuer la zone à visualiser (« *zooming* ») et déplacer celle-ci en simulant le glisser à l'aide de la souris (« *panning* »).

Les systèmes Pad [Perlin et Fox 1993] et Pad++ [Bederson et Holland 1994] sont un deuxième exemple d'utilisation de cette technique, où elle est utilisée afin d'explorer un espace d'information complexe, les objets de l'espace pouvant être : du texte colorié, des fichiers de texte, de l'hypertexte, des graphiques et des images.

L'avantage de cette technique de visualisation réside dans le fait que l'utilisateur décide quelle information visualiser et le degré de détail. Le point faible vient du fait que l'utilisateur peut se désorienter, c'est-à-dire qu'il peut perdre le contexte, aspect clé dans des espaces vastes d'information, au moment de faire un *zoom*.

2.5.3. Focus-plus-context

Lorsque l'utilisateur se rapproche d'un item, les éléments proches de lui sont généralement perdus de vue. L'idée de cette technique consiste à pallier cet effet, en faisant l'hypothèse que les objets plus proches de la zone de rapprochement sont plus pertinents que ceux qui sont plus éloignés. Il existe alors un centre (« *focus* ») d'attention, puis les autres objets sont redimensionnés de manière à ce que lorsque ceux-ci s'éloignent du « *focus* », leur représentation graphique est plus réduite, ce qui produit un effet visuel de distorsion simi-

laire à celui expérimenté lorsqu'on regarde à travers de certaines caméras. Ce faisant, l'utilisateur peut déplacer son centre d'attention tout en maintenant une vue globale de l'information.

Dans [Lamping et Rao 1996], se présente un *browser* hyperbolique qui utilise exhaustivement cette technique afin de permettre à l'utilisateur de naviguer commodément à travers de sites *Web* de grand taille. Les vues de type *fisheye* [Furnas 1986] constituent un autre exemple d'utilisation de cette technique.

2.5.4. Greeking

Le « *greeking* » [Eick *et al.* 1992] [Pirolli *et* Rao 1996] est l'une des techniques utilisées en visualisation d'informations pour afficher un texte dans un format réduit. Cette technique consiste à représenter chaque ligne du texte par un segment graphique de longueur proportionnelle au nombre de caractères présent dans la ligne.

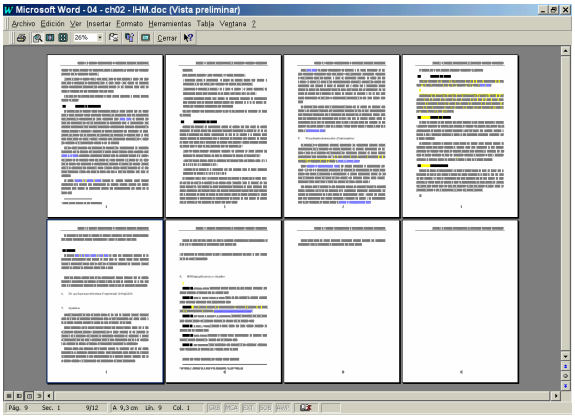

Figure 2.1 – *Exemple d'utilisation de la technique de greeking.*

Je prends comme exemple l'option d'*Aperçu* offerte par Microsoft Word, qui utilise cette technique (*cf.* figure 2.1). Les données ne sont pas visibles, mais la vue offre une vision globale de la distribution du texte. Dans le système SeeSoft [Eick *et al.* 1992] l'utilisateur peut distinguer rapidement les encodeurs auxquels appartiennent les lignes de code d'un programme de grande taille. Le système utilise le « *greeking* » et des techniques de colorisation afin d'attribuer une couleur différente à chaque encodeur.

2.6. Synthèse

Étant donné la nature de ce travail de thèse, une brève introduction au domaine des interfaces homme-machine a été réalisée. Dans les dernières années, ce domaine a été particulièrement pris en considérations par les travaux en linguistique computationnelle, notamment par les travaux de recherche d'information. Une caractérisation du domaine aussi qu'une description des objectifs généraux ont été présentés. Les différents styles d'interaction définis par Shneiderman [1992, 1998], un expert dans le domaine des IHM, ont été développés.

J'ai présenté ensuite quelques principes de dessin qui ont été pris en considération dans le développement de la plate-forme logicielle de navigation textuelle NaviTexte (*cf.* chapitre 12). Enfin, les techniques héritées du domaine de la visualisation interactive d'information qui ont été utilisées dans la conception et développement de cette plate-forme, ont été présentées dans ce chapitre.

Chapitre 3

Navigation Textuelle et Hypertexte

3.1. Introduction

Le terme de *navigation textuelle* reçoit de multiples interprétations. La plus commune renvoie inévitablement au processus mis en œuvre par les outils de navigation utilisés pour circuler dans les documents hypertextes, c'est-à-dire la possibilité d'activer un lien pour déplacer le point de lecture ; ce déplacement pouvant être intra ou intertextuel. Néanmoins, il existe plusieurs différences entre la navigation hypertextuelle et ce que l'on dénomme navigation textuelle. Je présente dans ce chapitre l'approche traditionnelle de l'hypertexte, et les approches alternatives de l'hypertexte adaptatif et de l'hypertexte dynamique, qui essaient de remonter certains inconvénients existants dans l'approche traditionnelle. Ensuite, je développe la notion de navigation textuelle, en mettant l'accent sur les différences existantes avec la navigation hypertextuelle.

3.2. L'hypertexte

3.2.1. L'approche traditionnelle

Au long de l'histoire, des instruments de recherche d'information ou d'aide à la lecture tels que la table des matières, les index, les renvoi, etc., ont été introduits, fondés sur la notion de page. Dès l'arrivée de l'informatique, ces instruments se sont multipliés et ils ont vu croître leur puissance. Actuellement, les logiciels de traitement textuel offrent potentiellement des instruments beaucoup plus puissants, ce qui est en général possible grâce à la possibilité de disposer, en arrière plan, de la représentation structurelle du texte.

Dans le cadre général de l'histoire du texte numérique (*cf.* chapitre 4), l'*hypertexte* place un jalon du point de vue conceptuel, et l'utilisation massive d'Internet a répandu son utilisation à grande échelle. L'implémentation classique de l'hypertexte consiste à offrir à l'utilisateur la possibilité d'activer un lien entre deux nœuds pour déplacer le point de lecture ; ce déplacement pouvant être intra ou intertextuel.

3.2.2. Désavantages de l'approche traditionnelle

Plusieurs aspects, que l'on considère désavantageux, sont à souligner dans ce type de navigation hypertextuelle. Tout d'abord, l'activation du lien est « aveugle », dans le sens où l'utilisateur possède peu d'information sur la cible du lien (un titre ou l'adresse URL qui est en général peu significative), et qu'il doit cliquer afin de tester la pertinence du lien par rapport à ses intérêts d'information.

Deuxièmement cette navigation est linéaire, c'est-à-dire qu'une seule voie de navigation est offerte au lecteur quand celui-ci active le lien. Autrement dit, pour chaque nœud source il existe un seul nœud cible. De notre point de vue, cela constitue une contrainte trop restrictive vis-à-vis des fonctionnalités offertes à l'utilisateur.

Troisièmement, l'orientation de la navigation n'est pas indiquée explicitement : le lecteur ne sait pas si le déplacement se fait vers l'amont ou vers l'aval[14] du texte lu, ce qui entraîne, entre autres, des phénomènes de désorientation cognitive.

Enfin et surtout, les liens sont placés dans le corps même du texte, ce qui implique qu'il n'est pas possible d'adapter les parcours dans ce texte au lecteur. En d'autres termes, aucune information ou connaissances complexes ne peuvent être associées à la navigation.

3.2.3. Le problème de la désorientation cognitive

Malgré le succès incontestable de l'hypertexte, les avis sur l'effectivité de son utilisation comme support à la lecture sont partagés. Par exemple, certaines critiques mentionnent le phénomène de désorientation cognitive [Elm *et* Woods 1985] [Edwards *et* Hardman 1989] [Cotte 2004$_b$] lorsqu'un utilisateur navigue dans un environnement hypertextuel. Lors d'une expérimentation où un ensemble d'utilisateurs doit exécuter certaines tâches de récupération d'information sur un système d'hypertexte, Elm et Woods [1985] identifient trois formes de désorientation :

- ne pas savoir où il faut aller ;

- savoir où il faut aller, mais ne pas savoir comment y aller ;

- ne pas connaître le positionnement dans la structure globale du document.

Les auteurs soulignent que le degré de désorientation des utilisateurs est indépendant de son degré d'expertise dans le domaine d'information. Nous pouvons alors nous demander

[14] *L'orientation* n'a de signification que dans le cas d'une navigation intratextuelle.

s'il s'agit d'un problème inhérent à l'hypertexte ou s'il s'agit plutôt d'un manque de pratique de la part des utilisateurs qui n'ont pas encore assimilé les fonctionnalités de l'hypertexte.

Nombre de travaux portent sur ce problème et certains auteurs ont proposé des métriques mesurant la désorientation [Smith 1996] [Otter *et* Johnson 2000]. D'autres approches de type plutôt qualitatif et non pas quantitatif ont été également tentées. Par exemple, Dominique Cotte [2002] présente des aspects temporels et spatiaux contribuant à la désorientation.

Par ailleurs, les résultats de différentes études sur l'intérêt de l'utilisation de dispositifs s'appuyant sur des techniques hypertextes sont contradictoires. Par exemple, Lee et Tedder [2003] montrent que les performances de rappel sont meilleures pour une lecture d'un texte traditionnel que pour un hypertexte structuré, mais que cette baisse de performance n'est pas constatée pour un hypertexte en réseau. Danielson [2002] montre, contrairement à [Stanton *et al*. 2000] que l'utilisation des cartes peut améliorer la tâche de recherche d'information.

3.2.4. L'hypertexte adaptatif

L'approche de l'hypertexte adaptatif [Mathe *et* Chen 1994] [Brusilovsky 1994, 1996] cherche à pallier le problème de l'adaptabilité à un utilisateur, présente dans l'approche traditionnelle de l'hypertexte. L'objectif principal de cette approche est d'adapter un système hypertexte aux besoins spécifiques d'un utilisateur. Notons que la modélisation de l'utilisateur est un point clé de cette approche.

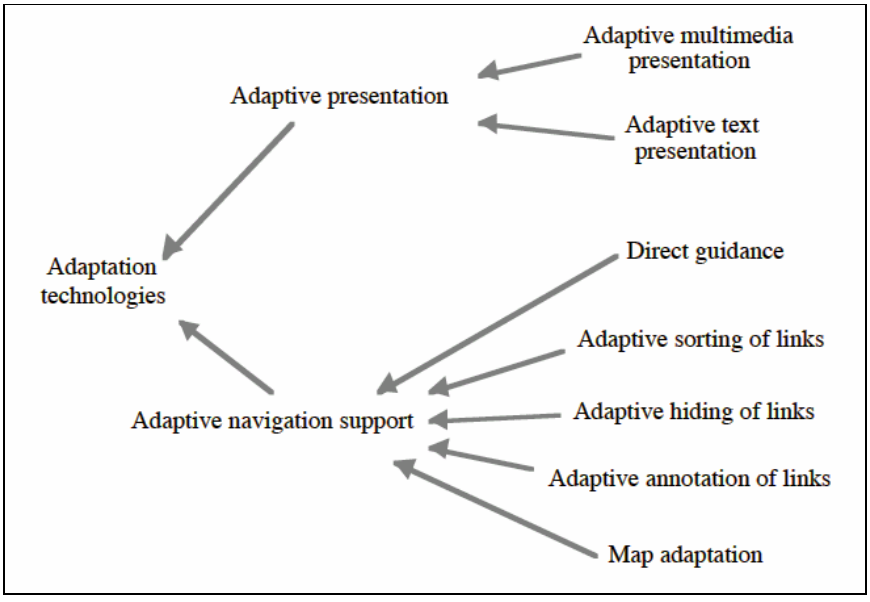

Figure 3.1 – *Deux manières d'adaptation de l'hypertexte (extrait de [Brusilovsky 1996]).*

Il existe deux manières d'adaptation (*cf.* figure 3.1) : *présentationnelle* et *navigationnelle*. La première a trait à l'adaptation des nœuds afin de modifier la manière dont l'information est affichée, tandis que la deuxième concerne les liens qui sont présentés à l'utilisateur. Dans [Kobsa *et al.* 1994], les auteurs suggèrent que les besoins des utilisateurs novices peuvent être satisfaits avec une présentation spécialisée ou simplifiée de l'information, tandis que les besoins des utilisateurs experts peuvent être satisfaits via une spécialisation de la navigation offerte.

3.2.4.1. Adaptation présentationnelle

L'idée des techniques d'adaptation présentationnelle est d'adapter le contenu d'une page accédée par un utilisateur particulier, selon ses connaissances, ses objectifs et autres caractéristiques d'intérêt. Par exemple, la technique nommée « *additional explanations* » consiste à occulter à l'utilisateur une partie de l'information concernant un concept qui n'est pas pertinant pour lui, compte tenu de son niveau de connaissance du concept. La technique « *prerequisite explanations* » consiste à présenter tous les concepts nécessaires pour comprendre un nouveau concept, avant de présenter ce dernier. Enfin, une autre technique que je prends comme exemple est celle de « *comparative explanations* », consistant à montrer, pour un concept, les différences et similarités avec un autre concept connu par l'utilisateur.

3.2.4.2. Adaptation navigationnelle

L'idée des techniques d'adaptation navigationnelle est d'aider l'utilisateur à trouver son chemin dans l'hyperespace, en lui présentant les liens selon ses connaissances, ses objectifs et autres caractéristiques d'intérêt. Dans [Brusilovsky 1996], cinq techniques d'adaptation navigationnelle sont décrites. La technique de « *direct guidance* » consiste à indiquer à l'utilisateur le prochain « meilleur » nœud à parcourir. Cela implique que le système puisse inférer ce meilleur nœud à partir du profil de l'utilisateur. La technique « *adaptive ordering* » consiste à trier les liens d'une page selon le modèle d'utilisateur et un critère de tri (par exemple : le plus important le premier). Une troisième technique, nommée « *hiding* », consiste à cacher les liens vers des pages considérées non relevantes. Une page peut être considérée comme non relevante si, par exemple, celle-ci n'est pas liée aux objectifs de l'utilisateur ; ou si celle-ci contient du matériel que l'utilisateur ne peut pas encore comprendre. L'idée de la technique nommée « *adaptive annotation* » consiste à enrichir les liens avec des commentaires plus significatifs pouvant indiquer à l'utilisateur la nature du contenu des nœuds correspondant aux

liens. Enfin, la technique de « *mapping* » permet à l'utilisateur de comprendre la structure globale de l'hyperespace et de se localiser lui-même dans l'hyperespace.

3.2.5. L'hypertexte dynamique

Tandis que l'hypertexte adaptatif conserve la notion de lien tel qu'il existe dans l'approche traditionnelle de l'hypertexte, l'hypertexte dynamique change la notion même de lien. Dans cette approche, les liens sont créés à l'exécution au lieu de les calculer préalablement (*liens pré-calculés*) ou à travers de modifications ou sélections dans des ensembles existants de liens (*liens adaptatifs*) [Bodner *et* Chignell 1999].

Une possibilité dans cette approche consiste à calculer les liens selon des relations prédéfinies ou selon des critères de similarité dans les textes. Dans ce cas, un lien n'est pas défini comme un pointeur d'un nœud hypertexte vers un autre, mais comme une requête visant un nœud. Ces requêtes peuvent prendre en considération l'historique de navigation, le profil de l'utilisateur, le contenu du document, etc.

Une propriété intéressante de l'hypertexte dynamique est que cette approche peut s'utiliser dans un système classique d'hypertexte aussi bien que dans un système de recherche d'information [Bodner *et* Chignell 1999].

3.3. La navigation textuelle

Notre conception de la navigation textuelle se démarque de la navigation hypertextuelle traditionnelle car nous considérons que circuler ou naviguer dans un texte est l'expression d'un processus cognitif qui convoque des connaissances qui sont propres à la finalité de la navigation [Minel 2003], [Couto *et* Minel 2004$_a$, 2004$_b$]. De ce fait, un documentaliste qui doit écrire un résumé d'un texte [Endres-Niggemeyer *et al.* 1995] ne navigue pas de la même façon qu'un lecteur intéressé par l'évolution des sentiments d'un des personnages d'un roman [Mathieu 2004] ou qu'un linguiste qui explore les annotations placés par un système automatique [Pery-Woodley 2005].

Le fait qu'un texte soit maintenant numérisé et qu'il soit présenté au lecteur sur un écran peut être considéré, de notre point de vue, comme une nouvelle mutation qui place le lecteur devant de nouvelles possibilités qui restent à explorer :

« Le texte [...] offre en effet une richesse sémiotique particulière, qui fournit de multiples objets d'interprétation et de multiples pistes d'actions [...] les lecteurs n'ont pas la même démarche envers l'objet ni la même définition de cet objet, ils ne « voient » pas la même chose ». [Souchier *et al.* 2003].

3.3.1. Exploitation de connaissances

Nous formulons l'hypothèse que la démarche du lecteur peut être assistée par l'exploitation de connaissances, existantes dans les textes, qui peuvent être, en partie, modélisées sous une forme déclarative, ce qui nous a amené à proposer le langage Sextant (*cf.* chapitre 8) pour exprimer ces connaissances. Autrement dit, il ne suffit pas de créer des liens mais il est nécessaire d'expliciter l'opération de navigation. De plus, ce processus de définition d'opérations de navigation doit être mis en œuvre par un « expert » capable d'encoder ces connaissances pour un certain type de textes déterminés. Par type de texte je ne réfère pas à une typologie textuelle mais plutôt aux types qui peuvent se dégager d'après les annotations (les connaissances) qui y sont présentes.

Une différence capitale entre la navigation textuelle proposée par nous et la navigation hypertextuelle réside dans le lieu où les connaissances navigationnelles sont placées. Dans le cas de l'hypertexte, celles-ci sont encodées par l'auteur dans le texte même et en font partie. Il en résulte que si le texte est modifié, les liens doivent éventuellement se modifier aussi. D'autre part, les connaissances navigationnelles sont dispersées au long du texte. Notre approche, peut-être plus traditionnel dans le sens où nous ne proposons pas une métamorphose du texte, consiste à séparer les connaissances navigationnelles de l'objet texte. De plus, nous considérons qu'un texte peut être visualisé de différentes manières et que chaque manière peut déterminer une façon de le naviguer.

Du point de vue du lecteur, la navigation textuelle que nous proposons est alors très différente de la navigation hypertextuelle dans le sens où nous considérons que le lecteur, qui active lui aussi des connaissances d'interprétation [Kintsch 2003] [Baccino 2004] doit pouvoir interagir en choisissant la voie de navigation qui lui semble la plus appropriée pour sa tâche de lecture.

3.3.2. Une approche de la navigation textuelle

Afin de proposer une approche systématisée à la navigation textuelle, quatre éléments sont nécessaires :

- une représentation du texte pouvant décrire différents phénomènes linguistiques ;

- la possibilité de pouvoir isoler les connaissances visuelles et navigationnelles ;

- un agent (une personne, une équipe d'experts, un système, etc.) capable d'encoder ces connaissances ;

- un système qui interprète ces connaissances.

Ces quatre éléments constituent l'approche de la navigation textuelle et ils seront développés au long de ce manuscrit.

Quoique fondé sur des hypothèses différentes, notre approche est assez proche de celle de l'hypertexte dynamique. Néanmoins, nous souhaitons explicitement éviter toute modélisation de l'utilisateur.

3.3.3. Qualité de la navigation textuelle

La qualité de la navigation textuelle est fonction de trois facteurs :

- la richesse de l'information présente dans les textes ;

- la compétence de l'encodeur ;

- la qualité du système logiciel de navigation.

En ce qui concerne le premier point, il existe de plus en plus de documents annotés avec des informations diverses. Or, bien que la compétence de l'encodeur puisse se concevoir initialement comme intuitive (par exemple, une personne qui imagine comment l'utilisateur peut naviguer un texte, ou comment certaines unités textuelles devraient s'afficher), elle peut parfaitement être systématisée au long de différentes expérimentations dans le cadre de la navigation textuelle.

Plusieurs questions se posent : les encodeurs doivent être des experts dans un domaine d'application (par exemple le résumé automatique) ou des experts dans des domaines tels que la linguistique, la sémiotique ou la psychologie cognitive ? Des techniques d'apprentissage automatique peuvent-elles aider à enrichir la qualité des connaissances à partir de grands jeux des données d'utilisation du système de navigation ? Le développement d'une méthodologie d'acquisition de connaissances constitue l'un des objectifs les plus centraux de notre approche.

La qualité du système de navigation dépendra, entre autres, du développement de composants visuels puissant qui utilisent les techniques et qui vérifient les critères de qualité vus au chapitre 2.

3.4. Synthèse

J'ai présenté dans ce chapitre l'approche traditionnelle de l'hypertexte et plusieurs aspects de cette approche considérés désavantageux ont été évoqués. Les approches alternatives de l'hypertexte adaptatif et de l'hypertexte dynamique, qui essaient de remonter certains in-

convénients existants dans l'approche traditionnelle, ont été présentées. La notion de navigation textuelle a été développée, en mettant l'accent sur les différences existantes avec la navigation hypertextuelle.

Deuxième Partie

Représentation des textes

Sommaire

Le traitement informatique d'un texte exige nécessairement une représentation de celui-ci afin d'être manipulé par un système. Le type de traitement à effectuer déterminera habituellement la représentation la plus adéquate selon des critères à choisir : rapport coût / performance, souplesse, exhaustivité, simplicité, etc. En conséquence, il existe différentes représentations d'un texte, de complexité croissante : des plus simples, qui le conceptualisent comme une séquence de caractères, au plus complexes, fortement imprégnées par des approches linguistiques, où la représentation de texte s'appuie sur un modèle du discours.

Dans cette partie de la thèse, je présente tout d'abord, dans le chapitre 4, quelques éléments concernant les enjeux impliqués par l'élucidation de l'objet « texte », afin de montrer qu'il s'agit d'un objet complexe et que tout essai de le modéliser ou de le représenter, présuppose forcement une prise de position et, habituellement, une simplification. Dans le chapitre 5, je présente différentes représentations informatiques de texte possibles, dont certaines sont liées à notre projet de recherche. Je développe ensuite, dans le chapitre 6, la représentation de texte définie dans notre projet, en motivant sa définition et la pertinence de ses différents composants.

Chapitre 4

L'objet « Texte »

4.1. Introduction

Qu'est-ce qu'un texte ?

Cette question a en fait donné lieu à une multiplicité de réponses et a entraîné plusieurs débats tout au long de l'histoire, où la notion de texte varie considérablement selon que l'on se place, par exemple, dans le camp de la linguistique, de la sémiotique, du structuralisme, de la théorie littéraire, de l'informatique ou de la bureautique. D'ailleurs, chaque point de vue ou approche de l'objet « texte » admet plusieurs ramifications, ouvrant un éventail de définitions se focalisant chacune sur des aspects spécifiques du texte. Ainsi, pour la sémiotique peircienne un texte est constitué d'un ensemble de *signes* organisé de manière à ce que ses *interprétants* le constituent en totalité collective [Peirce 1978] [Marty 2005], tandis que dans certaines approches informatiques un texte n'est qu'une suite de caractères, encodé selon un alphabet prédéterminé rendant possible sa décodification.

Bien que le propos de ce chapitre ne soit pas d'essayer de répondre à cette question complexe, ce qui représente un travail dépassant amplement les objectifs de cette thèse, il se révèle pourtant nécessaire, puisque la navigation textuelle se fonde sur l'existence de textes possédant des informations qui servent de guide à la lecture, de représenter l'objet « texte » de manière adéquate.

4.1.1. Nature physique

Afin de mieux cerner notre conception du texte, examinons différentes questions liées à sa conceptualisation. La première concerne sa nature *physique*. Le support matériel est-il indispensable pour l'existence d'un texte ou, par exemple, un ensemble d'énoncés oraux constitue un texte ? Christian Vandendorpe [1999, 2000] rappel le lien très fort existant entre écriture et oralité: *« La lecture à haute voix fut longtemps la forme normale de lecture, et ce travail d'oralisation était souvent confié à des esclaves dans la Grèce et la Rome antiques. Au IVe siècle de notre ère, l'évêque d'Hippone rapporte dans ses* Confessions *combien il avait été surpris, rendant visite au vieil*

Ambroise, de constater que celui-ci lisait sans même remuer les lèvres. ». C'est le passage des siècles et le changement des supports matériels du texte, allant du rouleau de papyrus jusqu'au texte numérique, qui ont dissocié progressivement l'oralité de l'écriture, rapprochant cette dernière du support matériel[15], où certains éléments tels que la page ou la marque de paragraphe ont contribué à spatialiser le texte.

4.1.2. Dimension spatiale et mise en forme

Cette dimension spatiale a été exploitée de diverses manières. D'une part, elle a permis l'introduction d'instruments de recherche d'information ou d'aide à la lecture tels que la table des matières ou l'index, ces instruments reposant sur la notion de page, élément fondamental qui permette, tout à la fois, à un lecteur de se repérer dans le texte et de faire une lecture sélective.

D'autre part, la dimension spatiale du texte a été également exploitée par les écrivains, notamment par les poètes qui, en associant au contenu d'un poème une mise en forme particulière, cherchent à renforcer le thème de celui-ci ou à lui intégrer des significations supplémentaires. Pour un poète comme Mallarmé ou William Blake, les attributs typographiques d'un texte et la disposition des vers sur la page (combinés éventuellement avec des éléments iconographiques) deviennent cruciaux.

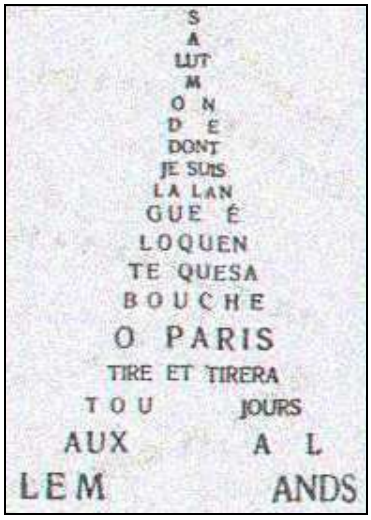

Figure 4.1 – *Calligramme « Tour Eiffel » de Guillaume Apollinaire.*

[15] On examinera dans la section 4.3 le lien particulier entre le texte numérique et le support matériel.

Les calligrammes constituent aussi un excellent exemple du pouvoir expressif d'une mise en forme particulière. La figure 4.1 montre l'un des très célèbres calligrammes publiés par Apollinaire dans son livre *Calligrammes* (1918).

Ici, le contenu du poème, sa mise en forme suggérant la Tour Eiffel et le moment historique où il a été écrit (première Guerre Mondiale) participent au sens du poème.

La mise en forme d'un texte constitue-t-elle toujours le résultat d'un traitement postérieur et plus ou moins indépendant à celui de la production du contenu ? Autrement dit, on écrit et après on fait la mise en forme, on définit d'abord la mise en forme et après on écrit, ou bien c'est un processus hybride ?

Rappelons que dans sa définition de la Mise en Forme Materielle (MFM) [Virbel 1989], Jacques Virbel postule l'existence d'une équivalence fonctionnelle entre formatage visuel et formulation langagière, intégrant les aspects typographiques et dispositionnels du texte. Selon cette approche, les structures visuelles d'un texte ne devraient pas s'entendre comme dissociées de son contenu mais comme fortement indissociables de celui-ci. En conséquence, toute analyse de l'objet « texte » et de ses constituants (théorèmes, définitions, énumérations, titres, etc.) devrait se réaliser en prenant en considération les deux composants (contenu et mise en forme). En guise d'exemple, le constituant textuel « définition » peut s'exprimer en utilisant une expression linguistique particulière (« *Un texte est…* », « *Notre définition de…* », etc.) ou en s'appuyant sur une structure visuelle déterminée, comme celle utilisée par les dictionnaires. Notons que le rapport entre constituant textuel et mise en forme n'est pas nécessairement biunivoque, un même constituant pouvant accepter divers formatages, et une même mise en forme être appliquée à différents constituants.

4.1.3. Définitions du terme « texte »

En ce qui concerne une définition « académique » du terme « texte », nous en trouvons plusieurs dans le Trésor Informatisé de la Langue Française [ATILF]. En voici quelques-unes :

TEXTE, subst. masc. **1.** Suite de signes linguistiques constituant un écrit ou une œuvre. **2.** *En particulier.* Écrit considéré dans ses termes exacts, originaux et authentiques. **3.***TYPOGRAPHIE* **a)** Partie de la page recouverte de caractères composant un écrit.

Tableau 4.1 – *Quelques définitions du terme « texte » [ATILF].*

La première acception est très générale car un signe linguistique, en tant qu'élément minimal, peut être une entité quelconque comportant deux aspects : signifiant et signifié [Saussure 1995] Les signes linguistiques pouvant être visuels, acoustiques, tactiles, etc. ; une suite de phonèmes constitue, d'après cette définition, un texte. Les acceptions subséquentes renvoient cependant le concept de « texte » au concept « écrit », présupposant un support matériel déterminé du texte.

Le même constat est valide dans le cas d'une définition sémiotique, comme celle provisoire (*sic*) formulée par Yves Jeanneret [2004] : « *un texte est une configuration sémiotique empirique attestée, produite dans une pratique sociale déterminée et fixée sur un support* ». Cette définition est une variante de celle donnée par François Rastier [2001], où le terme « configuration sémiotique » remplace celui de « suite linguistique ». Le remplacement est justifié car un objet textuel, en tant qu'objet observable, ne peut pas être « purement linguistique ». Notons que la notion de support fait partie de la définition (*i.e.* elle est indissociable de la notion de texte). Autrement dit, cette définition, comme plusieurs autres, exclut la possibilité d'avoir un texte « sans support ». Comme le souligne Vandendorpe [1999], bien que les origines de l'écriture soient étroitement liées à l'oral, force est de constater que l'on assimile fréquemment texte et texte imprimé. En conséquence, on présuppose un support matériel déterminé, restreignant ainsi la nature physique du texte.

4.2. Texte ou Document ?

Occasionnellement les termes « texte » et « document » sont utilisés indistinctement comme renvoyant au même concept. Néanmoins, il existe des différences notoires. Voici quelques définitions du terme « document » trouvées dans [ATILF] :

DOCUMENT, subst. masc.
A. — *Vx.* Enseignement, oral ou écrit, transmis par une personne
B. — *Usuel*
1. Pièce écrite, servant d'information ou de preuve.
2. *P. ext.* **a)** Ce qui apporte un renseignement, une preuve.

Tableau 4.2 – *Quelques définitions du terme « document » [ATILF].*

Même si intuitivement nous pouvons imaginer qu'un document se réfère à un objet physique tandis qu'un texte pourrait n'être qu'une séquence de phonèmes ou une autre forme intangible, la première acception offre la possibilité de l'oralité. De fait, c'est cette acception qui honore la racine du terme, provenant du vocable latin *documentum* qui veut dire « enseignement ». *Documentum* dérive de *docere*, signifiant « enseigner ». Pourrions-nous en inférer que, par exemple, un exposé oral réalisé dans le cadre d'un cours constitue en même temps un texte et un document ? Sans doute, mais ce que nous pouvons certainement inférer d'après la deuxième acception, c'est qu'un texte écrit constitue un document, sous-entendu que ce qui est écrit porte une information[16].

4.2.1. Notion de document

Le terme « document » n'est cependant pas restreint au papier, rentrant généralement dans la catégorie de documents des objets tels que les photographies, les statues, les drapeaux, etc. C'est Paul Otlet dans son *Traité de documentation* [Otlet 1934] qui étend la définition de document, considérant que les objets eux-mêmes peuvent être, dans certaines conditions, considérés comme documents. Suzanne Briet [1951] définit un document de la manière suivante : « *Un document est une preuve à l'appui d'un fait* », ce fait pouvant être un phénomène

[16] J'utilise ici le terme « information » de manière peu rigoureuse afin de ne pas faire une digression, sans négliger les différences que l'on peut signaler entre « information » et « donnée ».

ou bien physique ou bien intellectuel. Implicitement, Briet offre des critères pour différencier ce qui est un document de ce qui ne l'est pas. En guise d'exemple, elle discute sur la possibilité de considérer une antilope comme un document. D'après elle, une antilope ne devrait pas être jugée comme un document que si l'on la capture pour la confiner, par exemple, dans un zoo en tant qu'objet d'étude et d'exposition. En s'appuyant sur ces règles implicites, Buckland [1997] infère les règles suivantes pour juger ce qui constitue un document :

- Il y a de la matérialité : seulement des objets et des signes physiques sont considérés.

- Il y a de l'intentionnalité : l'objectif est de traiter l'objet comme évidence.

- Les objets doivent êtres traités : ils doivent être constitués en documents.

- Il existe une position phénoménologique : l'objet est perçu comme un document.

Rappelons au passage la notion de *document/monument* exprimée par Michel Foucault dans son livre *L'Archéologie du Savoir* [1969] : *« Disons pour faire bref que l'histoire, dans sa forme traditionnelle, entreprenait de " mémoriser " les monuments du passé, de les transformer en documents et de faire parler ces traces qui, par elles-mêmes, souvent ne sont point verbales, ou disent en silence autre chose que ce qu'elles disent ; de nos jours, l'histoire, c'est ce qui transforme les documents en monuments, et qui, là où on déchiffrait des traces laissées par les hommes, là où on essayait de reconnaître en creux ce qu'ils avaient été, déploie une masse d'éléments qu'il s'agit d'isoler, de grouper, de rendre pertinents, de mettre en relations, de constituer en ensembles. II était un temps où l'archéologie, comme discipline des monuments muets, des traces inertes, des objets sans contexte et des choses laissées par le passé, tendait à l'histoire et ne prenait sens que par la restitution d'un discours historique ; on pourrait dire, en jouant un peu sur les mots, que l'histoire, de nos jours, tend à l'archéologie, — à la description intrinsèque du monument. »*

4.2.2. Un débat actuel

Depuis mai 2002, un débat intense centré sur la notion de document a lieu au sein du Réseau Thématique Pluridisciplinaire numéro 33 du CNRS [RTP-DOC], baptisé « Document et contenus : création, indexation, navigation ». L'accent est mis sur le document dans son passage au numérique (*cf.* section 3) et, en faisant une analogie avec les notions linguistiques correspondant à *syntaxe*, *sémantique* et *pragmatique*, trois notions différentes de document sont proposées dans [Pédauque 2003] : le document comme *forme* (en tant qu'objet, matériel ou immatériel) ; le document comme *signe* (porteur de sens et doté d'une intentionnalité) ; le document comme *médium* (en tant que trace, construite ou retrouvée, de communication). Ces trois notions différentes impliquent trois points de vue, à partir desquels une définition de document numérique a été élaborée pour chacun :

- Document comme forme : « *Un document numérique est un ensemble de données organisées selon une structure stable associée à des règles de mise en forme permettant une lisibilité partagée entre son concepteur et ses lecteurs.* » ;

- document comme signe : « *Un document numérique est un texte dont les éléments sont potentiellement analysables par un système de connaissances en vue de son exploitation par un lecteur compétent.* » ;

- document comme médium : « *Un document numérique est la trace de relations sociales reconstruite par les dispositifs informatiques* ».

Notons que dans les trois définitions le support (numérique dans ce cas) est sous-entendu, et que seulement la deuxième définition utilise le terme « texte » en l'assimilant à la notion de document, tandis que les autres référent au concepts « données » ou « trace ». Alors, on pourrait discuter si un texte est un ensemble de données, ce qui semble à priori « raisonnable », ou si un ensemble de données constitue un texte, ce qui semble plus discutable et qui nécessite, préalablement, d'une définition opérationnelle de la notion de « donnée ». De toute manière, dans les deux cas est discutable, à moins de préciser la notion d'ensemble qui suppose éventuellement que les éléments qui y figurent ont des liens entre eux. La troisième définition est assez générale et elle pose nombreuses questions, que l'on n'examinera pas ici puisqu'elle nous porte hors de notre sujet.

4.2.3. Conclusion

Somme toute, la notion de document supposant un support matériel, nous pourrions peut-être associer celle-ci à la notion de fichier de texte. Pour notre part, dans le cadre de la navigation textuelle, nous privilégierions l'utilisation du terme *texte* au lieu de *document*, et toute utilisation du mot *document* se référera implicitement à un *texte numérique*.

4.3. Le passage au numérique

L'arrivée de l'informatique signifie une rupture dans l'histoire de l'écriture, notamment dans la chaîne de production de l'écriture et de sa restitution lors de la lecture, comme le signalent Emmanuël Souchier et Yves Jeanneret [2002] : « *Dans les dispositifs informatiques, l'espace lu n'est pas celui de la mémoire de l'écrit à proprement parler (le support physique où sont enregistrées les impulsions électroniques), mais celui où l'écriture est actualisée, donnée à voir, c'est-à-dire l'écran. D'où un premier paradoxe : si l'enregistrement de la trace mémorielle de l'écrit est supposé être pérenne dans la machine, l'écrit d'écran, en revanche, est un écrit nomade qui disparaît une fois la représentation terminée.* » Le concept d'*écrit d'écran* s'avère fécond lorsque nous constatons,

comme les auteurs le remarquent, que pour la première fois, l'homme n'est plus capable de lire un texte sans recourir à une machine. Ce concept d'écrit d'écran se présente d'ailleurs lié à celui d'*architexte* [Souchier *et* Jeanneret 1999], qui est, d'après les auteurs, l'ensemble d'outils permettant l'existence de l'écrit d'écran, ces outils ne servant pas seulement à représenter la structure du texte mais aussi à en commander l'exécution et la réalisation. Cela signifie que des outils comme les logiciels auteurs ou les navigateurs Internet, entre autres, sont des architextes. Ainsi, ce regard sémiologique sur l'objet « texte » montre que le phénomène de numérisation des textes n'est pas seulement une question technologique mais que cela implique une métamorphose de l'objet lui-même [Cotte 2004_a].[17]

En outre, le processus de numérisation des textes se révèle, en apparence, irréversible. Les raisons en sont multiples, le rapport entre la dimension du support et la quantité d'information stockée étant parmi les principales. L'apparition d'Internet vient en ajouter une autre : celle de l'ubiquité. L'importance de la numérisation des textes est manifeste, et le projet Google Print, qui ambitionne de numériser 15 millions d'ouvrages, en est la preuve. En même temps, ce phénomène remet en cause des concepts historiquement contestés mais assez reconnus jusqu'à présent comme la propriété intellectuelle ou les droits de reproduction.

4.3.1. L'hypertexte

Attribué habituellement à Vannevar Bush en raison de l'article [Bush 1945], l'*hypertexte* place un jalon dans l'histoire du texte numérique. Dans cet article, Bush imagine le système *Memex*, dispositif destiné à l'utilisation individuelle, sorte de librairie et de système de fichiers mécanisé. Il s'agit d'un dispositif de stockage de données hétérogènes, pouvant relier celles-ci selon des critères spécifiques définis par l'utilisateur. Intuitivement, les notions de lien et de parcours y sont implicites, même si les termes hypertexte ou lien ne sont jamais mentionnés. Néanmoins, l'article, influencé et connoté par la participation des scientifiques à la deuxième guerre mondiale, se présente comme un article général proposant des chemins à suivre par les scientifiques, et formulant quelques réflexions prospectives.

[17] Je propose comme exemple le passage de la musique sous format analogique au format numérique, conséquence également de l'évolution technologique. Il me semble que la métamorphose de l'objet (l'enregistrement musical), si celle-ci existe, est beaucoup plus nuancée que celle du texte. En effet, lorsque l'on compare les deux types d'enregistrements, on parle souvent de qualité sonore ou de souplesse de manipulation (accès, exécution, copie, *mixing*, etc.), mais à ma connaissance on ne mentionne pas de différences notoires quant aux deux types d'objets eux-mêmes.

Le terme hypertexte est, de fait, crée par Theodore Nelson [1965], en qualifiant de cette manière son système *Xanadu*, réseau permettant à tous les utilisateurs d'y accéder afin de lire, écrire et relier des textes. Dans ce système, l'idée de lien est explicite et l'on peut le concevoir comme une base de données virtuelle dans laquelle les utilisateurs peuvent se déplacer, accéder aux informations, les modifier, etc.

L'idée de l'hypertexte n'est cependant pas née de l'informatique mais elle a été proposée par des philosophes et des écrivains avant même d'être répandue et définie telle que nous la connaissons actuellement. Prenons comme exemples le livre *Cent mille milliards de poèmes* (1961) de Queneau, permettant de composer à volonté des poèmes, ou *Rayuela*[18] (1963) de Julio Cortázar, roman avec une table de navigation offrant au lecteur un parcours alternatif du texte, en lisant des chapitres annexes qui enrichissent l'histoire centrale du roman.

Malgré le succès incontestable de l'hypertexte, les avis sur l'effectivité de son utilisation comme support à la lecture sont partagés (*cf.* chapitre 3). D'une part, il existe un phénomène de désorientation cognitive [Elm *et* Woods 1985] [Edwards *et* Hardman 1989] lorsqu'un utilisateur navigue dans un environnement hypertextuel. D'autre part, les résultats de différentes études sur l'intérêt de l'utilisation de dispositifs s'appuyant sur des techniques hypertextes sont contradictoires.

4.3.2. Les livres électroniques

Laissons de côté l'hypertexte, déjà présenté au chapitre 3, et revenons à la question sur la nature physique du texte, en considérant les livres électroniques (*e-books*, en anglais). Ce type de livre est en quelque sorte censé se substituer au livre classique. Au moins, il devrait pouvoir en émuler les caractéristiques principales. les concepteurs des livres électroniques sont alors confrontés à plusieurs défis, car des éléments divers participent à la lecture sur ce type de support, tels que la typographie, la mise en forme ou l'ergonomie du dispositif [Baccino 2004]. Les premiers prototypes (*Cybook* de Cytale, *Dual-Screen* d'Estari, *REB1100* de RCA, *eBookMan* de Franklin, pour n'en mentionner que quelques-uns) n'ont pas recueilli beaucoup de succès puisqu'ils cherchaient à substituer au livre un écran, au lieu d'inventer sa version électronique. Je suis de l'avis du journaliste Michel Alberganti, qui en novembre 2002 exprimait à ce sujet :

> *« Indestructible, le livre ? Alors que tout a changé dans son processus de fabrication, de l'imprimerie aux techniques numériques de composition et d'impression, le "facteur de*

[18] *Marelle*, en français.

forme" du livre n'a que très peu souffert de l'arrivée des écrans électroniques portables. Associé à Internet, qui permet de télécharger à loisir le contenu des ouvrages, certains ont annoncé que ce vénérable support papier allait tomber sous les coups de l'informatique. Quelques entreprises américaines, comme Gemstar eBook, et la française Cytale s'y sont risquées avec des succès très mitigés.

…

Mais les lecteurs tirent profit et plaisir des contraintes imposées par le papier. L'empilement des pages confère au livre un volume, au point même qu'il en a adopté le nom. La mémoire humaine exploite d'ailleurs cette troisième dimension pour retrouver un passage déjà lu. Enfin, l'épaisseur du livre matérialise également à la fois le nombre de pages qu'il contient, son début, son milieu et sa fin.

Autant de sensations qui disparaissent avec l'électronique. L'exploration du contenu semble paradoxalement moins immédiate et, croyant faciliter la lecture, les promoteurs de cette technique ne voient pas que la tablette à page unique frustre le lecteur. Question d'habitude ? Difficile de le savoir aujourd'hui tant les premiers livres électroniques cumulent encore les défauts de jeunesse. »

« Heurs et malheurs du livre électronique »
Le Monde.fr, Sciences (13 novembre 2002)

Cependant, de nombreuses difficultés initiales sont actuellement en train d'être surmontées. Je prends le cas du *Glassbook Reader* [Glassbook] où la typographie est fort soignée[19] et le lecteur peut placer un signet, voire annoter une page. Le Glassbook propose ainsi un support assez proche du livre papier mais comportant aussi les avantages dus au support numérique, et il faudra attendre pour savoir s'il sera finalement accepté par les lecteurs.

4.3.3. Conclusion

L'arrivé de l'informatique et, en conséquence, l'apparition des phénomènes tels que le texte numérique, l'hypertexte ou les livres électroniques, ont complexifié encore davantage la conceptualisation du texte. Existera-t-il un jour un consensus sur ce qui est un texte ? Je reste persuadé que la définition de cet objet complexe est bien susceptible d'être un problème éternellement ouvert car il s'agit d'un objet en métamorphose permanente. Prenons par exemple le cas des *wikimedias* [Wikimedia] dont le plus reconnu est peut-être l'encyclopédie Wikipedia [Wikipedia]. L'idée capitale des wikimedias, fondés sur le principe du *wiki*[20] et

[19] Ce dispositif utilise des polices composées par des caractères dessinés de manière vectorielle, ce qui représente un avantage visuel important vis-à-vis des polices *TrueType*.

[20] Du mot hawaïen *wiki* qui signifie « rapide ».

très proches du système Xanadu envisagé par Theodore Nelson, consiste à offrir des médias contenant du savoir libre accessibles gratuitement par tous. Un wiki peut se concevoir en même temps comme :

- une collection d'hypertextes pouvant être visités et modifiés à volonté par un utilisateur quelconque[21] ;

- une application en informatique collective.

Les *weblogs*, dits aussi *blogs*, constituent un autre exemple de mutation. Un blog est un site Internet fréquemment actualisé où un ou plusieurs auteurs publient périodiquement des écrits. Habituellement, ces publications se compilent en ordre chronologique afin d'afficher les plus récents en premier.

Peut-on dire qu'un blog ou un wikimedia est un texte ? Sont-ils de nouveaux types d'objets textuels venant enrichir l'ensemble de types existants[22] ? Quelles sont les implications conceptuelles de ces technologies sur la conceptualisation de l'objet « texte » ?

Si nous acceptons qu'un wikimedia soit un texte (conçu comme une collection d'hypertextes qui entretiennent des liens entre eux), il s'agit alors d'une œuvre collective résultante des multiples apports en provenance, éventuellement, des personnes de diverses cultures, géographies, langues, et réalisés pendant une durée indéterminée, à priori non limitée. Un texte de telle nature peut susciter plusieurs réflexions sur la cohésion, la cohérence, l'homogénéité (signes linguistiques utilisés, mise en forme choisie, etc.), les référents culturels des différents auteurs, etc.

Il existe une dynamique entre technologie et ce que l'on pourrait nommer conceptualisation ou théorie. Au fil du temps, la technologie s'est appuyée sur des aspects théoriques, les poussant au-delà des limites pré-établis, comme c'est peut être actuellement le cas pour l'encre électronique[23]. En général, cette exploration ou montée en exigence a stimulé la réflexion théorique à propos des outils conceptuels, afin d'expliquer comment ces nouvelles technologies modifient la notion de l'objet « texte », ces explications servant de rétroaction et venant enrichir la technologie. Rappelons que dans les premiers systèmes informatiques de visualisation sur écran, la ligne constituait la seule unité manipulable et le défilement sé-

[21] Dans certains cas il est nécessaire de s'enregistrer comme utilisateur.

[22] Sous-entendu que l'on postule également qu'il existe des types divers d'objets textuels.

[23] Joseph Jacobson, du MIT, travaille sur cette idée innovante qu'il a trouvé dans un ouvrage de science fiction.

quentiel la seule opération de contrôle disponible. La notion de fenêtre a permis postérieurement d'introduire le contrôle spatial en deux dimensions à l'aide d'objets spécialisés que sont les barres d'ascenseur, renouant ainsi avec un support qui prévalait avant l'introduction du codex et qui est d'ailleurs particulièrement perturbateur d'un point de vue spatial [Baccino 2004]. Néanmoins, force est de constater que ces dernières années les logiciels de traitement textuel ont réintroduit le format page (Word, Acrobat Reader, etc.), bien qu'il s'agisse d'un construit éphémère, alors que les navigateurs utilisés pour explorer le Web cherchent au contraire à introduire de nouvelles compositions spatiales qui allient fenêtres, bandeaux, tableaux, listes, etc. Cela ne signifie pas que la publication sur Internet aille à contresens des logiciels de traitement textuel. Bien au contraire, certains sites Internet, notamment ceux des journaux, se structurent d'une manière très proche au format papier, en utilisant des colonnes, de la pagination, des tables de matières, etc. Comme le précise Vandendorpe [2000] : *« on est en train d'intégrer à l'hypertexte les repères qui ont fait le succès du livre. »*

4.4. Synthèse

Pour clore ce chapitre dédié au texte, la littérature autour ce sujet étant immense, je n'ai voulu que mentionner quelques éléments concernant les enjeux impliqués par son élucidation, sans aucune prétention d'exhaustivité. Comme je l'avais annoncé dans l'introduction du chapitre, je n'ai pas tenté ici une définition du texte, mais plutôt d'exposer certaines questions qui se présentent pour le conceptualiser, l'idée étant de montrer qu'il s'agit d'un objet complexe et que tout essai de le modéliser ou de le représenter, afin de le manipuler informatiquement, présuppose forcement une prise de position et, souvent, une simplification.

Dans le cadre de cette thèse, je m'intéresse aux textes écrits encodés sous format numérique (textes numériques) qui contiennent des informations servant de guide à la lecture. Je ne considère pas les textes représentés par des images (par exemple des images issues d'un traitement de *scanning*) mais seulement ceux dont le contenu matériel est constitué d'entités saisissables et, en conséquence, analysables (lettres, caractères de ponctuation, etc.). Cela signifie que je travaille avec une facette très particulière de l'objet « texte », assez proche d'une fusion de deux premières définitions de document numérique exprimées dans [Pédauque 2003], et qui ne prend certainement pas en considération toutes les nuances de cet objet complexe, malgré le vif intérêt que cela représente.

Chapitre 5

Représentation informatique des textes

5.1. Motivation des différentes représentations

Rappelons, comme nous l'avons exprimé au début de cette partie, que le traitement informatique d'un texte nécessite forcément une représentation de celui-ci afin d'être manipulé par un système. C'est le type de traitement à effectuer qui viendra déterminer habituellement la représentation la plus convenable selon des critères à choisir : rapport coût / performance, souplesse, exhaustivité, simplicité, etc. En conséquence, il existe des représentations d'un texte qui le conceptualisent comme une séquence de caractères, jusqu'à des représentations fortement imprégnées par des approches linguistiques, où la représentation de texte permet de modéliser certains phénomènes discursifs.

Je voudrais effectuer tout d'abord deux constats, le premier concernant le nombre de représentations gérées par un système. Je considère que, en général, il en existe trois : la représentation du texte source, une représentation intermédiaire créée en mémoire par le système, et une représentation du texte résultat, comme le montre la figure ci-dessous.

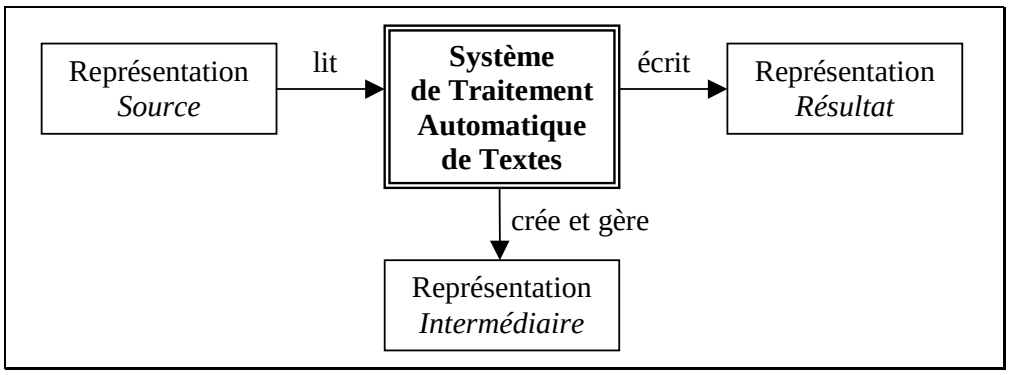

Figure 5.1 - *Possibles représentations d'un texte gérées par un système.*

Prenons l'exemple d'un système classique de construction automatique de résumés qui prend en entrée un texte plat (une suite de caractères), construit une représentation adéquate en mémoire (par exemple une arborescence selon un modèle déterminé), et affiche en sortie un autre texte, éventuellement avec des annotations (par exemple un texte annoté avec XML). Dans cette optique, ces logiciels de traitement automatique de textes fonctionnent de manière semblable à certains « traducteurs » [Aho *et al.* 1986] : ils traitent un texte source, construisant une représentation intermédiaire, puis ils génèrent un texte résultat.

Bien évidemment, les trois représentations (*source*, *intermédiaire* et *résultat*) ne sont nécessairement pas différentes pour tous les systèmes de traitement de textes, et l'on peut trouver des cas où la représentation source coïncide avec la représentation intermédiaire, ou cette dernière correspond aussi à la représentation du résultat.

Dorénavant, lorsque je mentionne la représentation d'un texte faite par un système, je ferai implicitement allusion à celle dont le système a besoin pour effectuer le traitement qu'il est censé réaliser. Dans certains cas celle-ci viendra directement de la représentation source (par exemple dans une représentation XML que le système peut reprendre tel quel) ; dans d'autres cas, celle-ci sera construite par le système (par exemple par un système de résumé automatique par extraction de phrases qui doit préalablement repérer celles-ci dans un texte plat).

Le deuxième constat a trait au type de représentation ; je propose d'en différencier deux : l'une *conceptuelle,* qui correspond à la notion de modèle, et l'autre *logicielle*, qui correspond à l'instance concrète (*i.e.* informatique) de la représentation. Une représentation conceptuelle est définie de manière abstraite et indépendante de toute implémentation informatique. Ainsi, l'on pourrait dire qu'un texte est constitué d'une suite de phrases, par exemple, sans renvoyer nécessairement à un format particulier ou à une façon d'encoder le texte. Cette représentation admet plusieurs représentations informatiques concrètes : un fichier texte ayant une phrase par ligne, un fichier texte plat à partir duquel il faut repérer les phrases, un fichier XML où les phrases sont des éléments structuraux, etc.

Notons que même une représentation aussi élémentaire que celle d'après laquelle un texte n'est qu'une séquence de caractères, utilisée entre autres, par les analyseurs lexicographiques des compilateurs, constitue une représentation valide de texte. Ce type de représentation, employée par les premiers éditeurs de texte comme *vi* en *Unix* ou *Wordstar* en *MS-DOS*, exige l'introduction de caractères spéciaux afin de donner des instructions (par exemple un

tabulateur, le renvoi à la ligne, une mise en forme en italique, etc.). Une généralisation de cette approche nous amène à des représentations tels que les formats RTF[24], PostScript, SGML[25], etc., où nous pouvons distinguer, dans un même contenu textuel, l'information textuelle proprement dit de la méta information incorporée au texte lui-même selon une syntaxe définie.

Pour les traitements stochastiques des textes pour lesquels il suffit généralement de représenter le texte comme une séquence d'unités lexicales (lexèmes). C'est le cas, par exemple, de plusieurs analyseurs morpho-syntaxiques ou de certains systèmes de reconnaissance de la parole, qui alimentent à partir de grands corpus des modèles n-grams afin de calculer la probabilité d'occurrence d'une unité en fonction des unités précédentes.

D'autres types de traitements, comme ceux destinés à la recherche d'information, représenteront le texte sous forme d'index inversés ou de vecteurs de suffixes, dans le but de répondre de manière performante à une expression de recherche [Baeza *et* Ribeiro 1999]. Le contenu du texte n'est pas entièrement pris en considération : le système élimine certains mots très fréquents (articles, prépositions, conjonctions, etc.), dits « stop-words », car ils ne sont pas intéressants en tant que termes de recherche. Quelquefois, l'information sur la structure du texte, si elle est disponible, est également encodée dans les index, permettant ainsi des recherches plus sophistiquées.

Pour certaines applications, comme celles de catégorisation de documents ou de résumé automatique, il est souvent important de distinguer des constituants textuels tels que les titres ou les paragraphes, en faisant l'hypothèse que l'occurrence d'un terme ou d'un syntagme, typiquement nominal, n'a pas la même valeur (poids) s'il apparaît dans un titre ou dans le corps du texte. De plus, la place du paragraphe dans le texte global peut être un indicateur de pertinence, de même que le rang d'un titre.

Dans les cas où la représentation d'un texte est orientée vers des analyses linguistiques fines, il est habituel de considérer tant des éléments structuraux (titres, sections, paragraphes, phrases, propositions, syntagmes, etc.) que des éléments discursifs (cadres organisationnels, relations rhétoriques, etc.). On reprendra ce type de représentations dans les sections suivantes.

[24] Rich Text Format.

[25] Standard Generalized Markup Language.

Enfin, la représentation des textes peut supposer que le texte n'est pas l'unité majeure mais qu'il fait partie d'une entité plus globalisante. C'est par exemple le cas des hypertextes, des systèmes de résumé multi-textes ou des systèmes de question-réponse où la réponse à une question se trouve généralement disséminée tout au long de plusieurs textes.

Il faut donc bien constater qu'en informatique la représentation d'un texte est fortement dépendante du type de traitement à effectuer.

5.2. Nature des objets à manipuler

Je propose une classification assez générale des types de représentation des textes, selon la nature des objets à manipuler, ce qui est lié aux différents niveaux de granularité.

5.2.1. Représentations « synthétisées »

Commençons par un certain type de représentation que l'on pourrait dénommer « synthétisées » au sens que celles-ci impliquent une perte d'information. C'est le cas, par exemple, des « sacs de mots » [Salton *et al.* 1975] : des représentations vectorielles d'après lesquelles un texte est représenté par un *vecteur* n-dimensionnel. Chaque coordonnée du vecteur correspond à la fréquence d'un terme dans le texte, et il existe différentes manières de pondérer un terme (cosinus de Salton, coefficient de Dice, coefficient de Jaccard…).

Ces types de représentations sont utilisés habituellement dans des applications de recherche d'information car ils permettent d'indexer des textes. Il est clair que, bien que les termes du texte font partie de la représentation, leurs instances concrètes et les relations créées par leur enchaînement (y compris la séquentialité) ne sont pas représentables, ce qui constitue une perte d'information.

5.2.2. Représentations plates

Les représentations plates de texte consistent à conceptualiser celui-ci comme une suite d'unités, ces unités pouvant être des caractères (dans le cas des analyseurs lexicographiques des compilateurs, par exemple), des lexèmes, etc. En guise d'exemple, je présente dans cette section la représentation proposée par Clarke *et al.* [1995] et celle proposée par Ben Hazez [2002].

5.2.2.1. Clarke

Clarke [1995] considère un texte comme une suite de symboles concaténés appartenant à un alphabet déterminé et à un alphabet de « *stopwords* ». La représentation étant proposée

dans le cadre de la recherche d'information, il existe une fonction d'index prenant en entrée un symbole du premier alphabet et renvoyant une liste de positions dans le texte où le symbole apparaît.

Le problème auquel s'attaque Clarke a trait au nombre de réponses à une requête. Sur la représentation de texte proposée, la réponse à une requête consiste en un ensemble de paires *(p,q)*, tels que *p* correspond à la position initiale du segment réponse et *q* correspond à la position finale. Puisque les paires peuvent se chevaucher et s'emboîter, Clarke définit une algèbre de requêtes, en proposant des opérateurs afin de réduire le nombre de réponses à une requête.

En s'inspirant de la représentation de Clarke, Slim Ben Hazez en a proposé une dont la finalité est le filtrage sémantique de textes.

5.2.2.2. Représentation utilisée par la plate-forme logicielle Semantext

Semantext [Ben Hazez 2002] est une plate-forme d'Exploration Contextuelle visant le filtrage sémantique de textes. La représentation de texte proposée par Ben Hazez, cherchant une alternative à celle utilisée au sein du projet ContextO, considère le texte comme une suite de caractères à partir desquels il est possible de créer des segments (définis par une position de début et de fin dans le texte) éventuellement étiquetés et de relier ceux-ci entre eux afin de construire une structure relationnelle. Ainsi, la structure textuelle construite prend la forme d'un graphe où les sommets sont des segments textuels étiquetés et les arcs représentent des relations nommées entre eux.

Cela implique que le texte, d'après cette représentation, est une entité ouverte quant à ses possibles éléments constitutifs. En effet, Ben Hazez ne restreignant pas l'utilisateur à un jeu d'unités textuelles prédéfini, il lui offre le choix de nommer à son gré les différents segments textuels.

Les critiques que l'on peut adresser à cette approche sont principalement deux. En premier lieu, Semantext fait un pré-traitement du texte en identifiant des différents éléments textuels (titres de section, paragraphes, phrases, etc.) présents dans le texte, cette structuration se traduisant en termes d'étiquettes et relations entre segments textuels. Il me semble, de fait, qu'il existe implicitement une représentation hiérarchique de texte, laquelle n'étant pas obligatoire, demeure présente. On peut alors se demander s'il ne s'agit pas, peut-être, d'une confusion entre représentation de texte et implémentation informatique de la représentation de texte.

En deuxième lieu, la représentation sous forme de segments textuels reliés entre eux de certaines structures textuelles, tels que les tables, les énumérations [Luc 2002] ou les puces, se révèle, de mon point de vue, assez complexe et le graphe résultant est susceptible de devenir non-maîtrisable par un encodeur humain.

5.2.3. Représentations structurées

Les représentations structurées de texte adoptent généralement une approche hiérarchique où des aspects syntaxiques et discursifs se mélangent souvent. Je présente dans cette section la représentation générique proposée par la TEI [TEI] et celle proposée par Gustavo Crispino [2003].

5.2.3.1. Représentation générique proposée par la TEI

La « *Text Encoding Initiative* » (TEI) est un projet visant la définition d'un ensemble de normes pour échanger des textes électroniques. Il s'agit d'une représentation générique fondée sur le SGML, offrant un ensemble de balises assez représentatif pour l'encodage des textes. Comme dans SGML, en TEI les éléments s'arrangent hiérarchiquement.

La TEI lite est une DTD prenant en considérations une sous partie des balises possibles. Le texte est divisé en trois niveaux :

- « *Front* » : contient les matières préalables au texte, tels que les en-têtes, le titre du texte, le préface, etc. ;

- « *Body* » : contient le corps du texte ;

- « *Back* » : contient des annexes, etc.

Il existe des éléments de division tels que le paragraphe « *p* », ou les génériques « *div* », qui peuvent s'emboîter (« div1 », « div2 », …, « div8 »). Ces éléments de division peuvent être typés et comporter un identifiant, utilisé postérieurement pour créer des liens entre les éléments.

Il faut souligner, d'une part, que la TEI offre des éléments spécifiques à certaines typologies textuelles, tels que « l » pour un vers en poésie. D'autre part, la TEI mélange des aspects de structure avec des aspects de mise en forme. Par exemple, une balise est prévue pour indiquer la fin de page.

Crispino [2003] s'est inspiré de cette représentation pour en proposer une, que je présente par la suite.

5.2.3.2. Représentation proposée par la plate-forme logicielle ContextO

La plate-forme ContextO, mentionnée au chapitre 1, utilise une représentation de texte spécifique, issue notamment du travail de thèse de Gustavo Crispino [2003] : « *Nous proposons pour notre travail une structure textuelle de base qui correspond à une organisation minimale commune à la plupart des textes. Une première approximation nous conduit à considérer que les textes que nous cherchons à représenter ont une hiérarchie, superficiellement apparente ou sous-jacente que nous organiserons en six niveaux : texte, section, sous-section, paragraphe, phrase et unité lexicale.* »

Bien que les éléments textuels sélectionnés soient bien connus, sous-jacente à cette représentation proposée se trouve la définition de LangTex [Crispino 2003], un puissant formalisme organisé en deux couches, dont la première, nommée *CBase*, permet la manipulation de la structure textuelle. Voici quelques exemples de la richesse offerte par LangTex :

1. *Sélectionner la première* ul *du quatrième paragraphe de la troisième section du texte* t *:* premiereUL(parNro(4, secNro(3, t))) 2. *Tester si la section qui contient l'unité lexicale* ul *est la dernière section de son niveau :* est-derniereSec(secParent(parParent(phParent(ul)))) 3. *Extraire les cinq dernières unités lexicales de la deuxième phrase du paragraphe* par *:* suivantes(ulNro((phNro(2, par).nombreUL) – 5, phNro(2,par)))

Tableau 5.1 – *Exemples d'utilisation de* LangTex *(extraits de [Crispino 2003])*

Nonobstant le bon rapport coût / performance de la représentation proposée et la puissance du langage offert, je partage les critiques de Crispino portant, d'une part, sur la rigidité de celle-ci due au fait d'offrir un ensemble fermé d'unités textuelles, et d'autre part, sur l'impossibilité de représenter des structures ne suivant pas la hiérarchie textuelle définie. Je voudrais y ajouter un point concernant les titres : dans de nombreuses représentations, y compris celle utilisée par ContextO, le titre (du texte et des sections) est généralement réduit au rôle d'un attribut des éléments textuels. Ce faisant, l'on interdit au titre d'avoir le statut d'unité textuelle, et l'une des conséquences la plus notoire, avec ContextO, est l'incapacité d'annoter sémantiquement un titre, bien qu'ils expriment souvent des valeurs sémantiques essentielles au traitement.

5.3. De quelques problèmes de représentation en linguistique textuelle

La linguistique textuelle ne se limite pas, contrairement aux analyses « traditionnelles », à la phrase comme élément d'étude. Bien au contraire, la linguistique textuelle travaille depuis une bonne trentaine d'années à décrire des marques linguistiques de la cohérence textuelle et à dégager les principes de la structuration des textes, en élaborant un éventail de concepts et de modèles d'interprétation textuelle, parmi lesquels il convient de mentionner – bien que les chercheurs soient loin d'être d'accord ni sur leur nombre, ni sur leur définition, ni sur leur appellation – les anaphores [Kleiber 1990], les connecteurs, les constructeurs d'espaces mentaux… [Fauconnier 1984] [Lundquist 1999].

Cette approche privilège l'étude du rapport existant entre les différentes unités textuelles, sous l'hypothèse que celui-ci dévoile certaines propriétés générales des textes tels que la *cohérence* et la *cohésion* [Halliday *et* Hasan 1976] [Charolles 1995]. Forçant le linguiste à regarder des séquences linguistiques qui vont au-delà de la phrase grammaticale, la linguistique textuelle a prouvé sa portée, par exemple, dans l'enseignement, que ce soit dans l'enseignement de la langue maternelle [Béguelin 2000], dans l'enseignement des langues étrangères [Lundquist 1990_a], dans l'enseignement de la littérature, dans l'enseignement des langues de spécialité et de la traduction [Lundquist 1989, 2000], pour ne mentionner que quelques disciplines auxquelles la linguistique textuelle présente un intérêt.

5.3.1. Les théories du discours

Il existe maintes théories du discours, parmi lesquelles on peut citer : l'Encadrement du discours [Charolles 1997], la Théorie des Structures Rhétoriques (*RST*) [Mann *et* Thompson 1988], la Théorie de la Représentation du Discours [Kamp *et* Reyle 1993], la Théorie du Centrage [Grosz *et al.* 1995] et le Modèle Linguistique du Discours [Polanyi 1987, 1988]. Généralement, les différentes théories partagent le fait d'établir une situation de dépendance entre deux unités du discours en terme de *coordination* ou de *subordination*, mais elles diffèrent dans leurs approches quant aux niveaux du discours étudiés (thématique, logique, rhétorique, etc.), les unités du discours considérées et les types de relations discursives supposés. Ainsi, pour certaines théories, l'unité discursive sera la proposition [Van Dijk *et* Kintsch 1983] [Mann *et* Thompson 1988], tandis que pour d'autres elle sera la phrase syntaxique [Polanyi 1988] [Choi 2002]. Si les différences ne sont pas très prononcées au niveau de l'unité du discours, elles sont notoires quant aux types de relations discursives considérées.

5.3.2. Les cadres de discours

Prenons le cas des *cadres de discours* [Charolles 1997] car il est important de montrer que les approches d'analyse du discours postulent des entités qui échappent à la hiérarchie plus ou moins accepté existant au niveau syntaxique. Michel Charolles propose dans [Charolles 1997] une théorie d'encadrement du discours qui se fonde sur l'hypothèse selon laquelle la cohérence, dans le discours, est révélée par l'existence de segments homogènes suivant un critère sémantique déterminé (par exemple un thème ou une localisation temporelle). Ces segments, dits « cadres de discours », peuvent se concevoir comme des unités informationnelles contribuant à la partition et à la structuration du discours, et ils sont annoncés par des expressions introductrices de cadres figurant généralement en position initiale du cadre. Les cadres de discours établissent ainsi un lien de cohésion textuelle que le lecteur reconstruit à partir de nombreux indices linguistiques et en particulier en s'appuyant sur ces introducteurs de cadres. La portée de ces marques, syntaxiquement non intégrées à l'énoncé où elles figurent matériellement peut s'étendre sur plusieurs phrases (voire paragraphes, pour certaines d'entre elles) créant ainsi une véritable unité textuelle, homogène sémantiquement et relativement autonome par rapport au contexte.

Charolles distingue quatre familles de cadres :

- les cadres **vérifonctionnels**, dont font partie les cadres **spatiaux** (« *À Chypre …*», « *En Europe… », etc.*), les cadres **temporels** *(« En 2003… », « Dans les années 1960… », etc.)*, les cadres **médiatifs** (« *Selon Piaget* »), les cadres **représentatifs** (« *Dans le film de Godard* »), les cadres **praxéologiques** (« *En chimie* ») ;

- les cadres **thématiques** (« *En ce qui concerne la partition de Chypre… », « À propos de la loi… », etc.*) ;

- les cadres **qualitatifs** (« *Par chance… », « Pour déranger son voisin … »*) ;

- les cadres **organisationnels** (« *En premier lieu …», « D'une part… », etc.*).

Divers systèmes de traitement automatique de textes ont utilisé la notion de cadre de discours afin d'améliorer, vis-à-vis des résultats, les traitements effectués. Par exemple, dans [Couto *et al.* 2004] les cadres thématiques [Porhiel 2001, 2003] et organisationnels sont calculés et utilisés comme support pertinent pour l'exploration des textes, dans le cadre du résumé automatique. Dans ce projet, l'une des hypothèses revendique que la structure thématique des documents offre une vue synthétique du contenu permettant de le parcourir afin

d'accéder aux différents niveaux de granularité thématique. Dans [Bilhaut *et al.* 2003] les aspects spatiaux et temporels (cadres vérifonctionnels) servent à développer une application de recherche d'information dans les documents géographiques, sous-entendu que ceux-ci permettent une indexation discursive assez fine de documents : « *En effet, la propriété caractéristique de l'information géographique est d'ancrer ce que nous désignons comme phénomène (le "quoi") dans une localisation spatiale (le "où") et, fréquemment, temporelle (le "quand").* » [Bilhaut *et al.* 2003]

Un point important a trait au phénomène d'indexation du aux marqueurs introducteurs de cadre. Tandis que pour Hallyday et Hamas [1976] les relations considérées sont les anaphores, c'est-à-dire des marques connectant avec une autre marque située en amont, dans les cadres de discours, les introducteurs de cadre indexent en aval toutes les propositions et phrases qui sont situés sous leur portée. Par exemple, lorsque l'on trouve un cadre de discours commençant par l'introducteur « *Dans les années 1960* », celui-ci indexe toutes les phrases qui suivent (jusqu'a la fermeture du cadre). C'est-à-dire que tous les événements relatés ont eu lieu dans les années 60.

5.3.3. Le repérage des introducteurs des cadres

L'un des problèmes les plus délicats d'une approche informatique (*i.e.* systématisé) de cadres de discours consiste à bien repérer la portée des expressions introductrices des cadres car, faute de cette précision, la structuration construite du discours risque d'être inexacte. Il est donc nécessaire de bien répertorier les indices susceptibles d'indiquer la finalisation d'un cadre. Malheureusement, ceux-ci ne sont habituellement pas explicites dans le texte. C'est-à-dire qu'ils ne sont pas concrétisés par l'auteur du texte sous forme d'expressions lexicales déterminées, signes de ponctuation, structuration syntaxique, etc. Il est alors nécessaire d'introduire des heuristiques, et l'on utilise parfois l'ouverture d'un nouveau cadre[26] ou le changement de temps verbal comme des indices de fermeture de cadre. Quelquefois, le changement de paragraphe constitue un indice de fin de cadre assez fiable [Couto *et al.* 2004] [Bilhaut *et al.* 2003]. Notons que cette heuristique présuppose une représentation textuelle prenant en considération les paragraphes. Il existe cependant certaines situations où la règle échoue, comme dans le cas des énumérations où les items peuvent correspondre à un paragraphe ou peuvent être composés de plusieurs paragraphes : un introducteur de cadre mis en amorce est susceptible d'englober tous les items qui le suivent [Bilhaut *et al.* 2003].

[26] Ce qui n'est pas toujours exacte car les cadres peuvent s'emboîter.

5.3.4. Les marqueurs d'intégration linéaire

Il faut attirer l'attention sur le fait que la structure suggérée par un cadre de discours peut transgresser la hiérarchie syntaxique du texte, conformé par des sections, paragraphes, phrases, etc. Je prends comme exemple les marqueurs d'intégration linéaire (*MIL*) [Jackiewicz 2002] [Jackiewicz *et* Minel 2003] introducteurs de cadres de discours ayant la propriété distinctive d'être indépendants des contenus sémantiques des cadres qu'ils introduisent, et qui conforment une série linéaire : *d'une part… de l'autre…* ; *en premier lieu … en second lieu…* ; *premièrement… deuxièmement… troisièmement…* ; etc. Notons que, chaque MIL introduisant un cadre organisationnel, l'ensemble de ceux-ci définit une structure organisationnelle plus complexe, introduite d'habitude par une amorce qui rend explicite le principe fédérateur des items de la série et en précise la longueur [Jackiewicz 2002], comme l'illustre l'exemple ci-dessous.

Quatre-vingts ans après, l'Union soviétique a fait naufrage, et le monde connaît une nouvelle grande mutation, que nous pourrions appeler la seconde révolution capitaliste. Elle résulte, comme la première, de la convergence d'un faisceau de transformations survenues dans trois champs.
***En premier lieu**, dans le domaine technologique. L'informatisation de tous les secteurs d'activités ainsi que le passage au numérique (…) bouleversent le travail, l'éducation, les loisirs, etc.*
***En deuxième lieu**, dans le domaine économique. Les nouvelles technologies favorisent l'expansion de la sphère financière. Elles stimulent les activités possédant quatre qualités: planétaire, permanente, immédiate et immatérielle. (…)*
***En troisième lieu**, dans le domaine sociologique. Les deux bouleversements précédents mettent à mal les prérogatives traditionnelles de l'État-nation et ruinent une certaine conception de la représentation politique et du pouvoir. (…) (Le Monde Diplomatique)*

Tableau 5.2 – *Exemple de MIL (extrait de [Couto et al. 2004]).*

Dans cet exemple la longueur de la série est indiquée par l'expression « trois champs » placée dans l'amorce de la structure globale (*i.e.* la série de trois cadres organisationnels). La

mise en texte d'une série n'utilise pas de disposition structurelle spécifique au sens de constituants textuels y présents. Dans l'exemple ci-dessus, la série est constituée d'une phrase amorce (où les anaphores dénotées par les référents « elle » et « la première » renvoient à la phrase précédente, où se trouvent les antécédents) suivie d'un ensemble de paragraphes, représentant chacun un cadre de discours introduit par un MIL. Si l'on veut définir une structure constituée de l'amorce et des trois introducteurs de cadre, clairement ce nouvel élément brise la hiérarchie établie. Néanmoins, si l'on veut définir un élément qui prenne en considération la structure des cadres, il est plausible de faire correspondre cet élément avec la structure hiérarchique du texte, mais d'autres combinaisons sont concevables.

5.3.5. Un exemple de cadres temporels

Il existe cependant des cas où même si les cadres sont contigus, la structure à créer ne peut pas être cadencée avec une hiérarchie. Je prends par exemple le morceau de texte suivant :

> ***Le 21 juin dernier**, chaleur et gris sur Paris. A Matignon, des ouvriers montent une estrade pour la Fête de la musique. Dans son bureau du 1er étage, Dominique de Villepin, en chemise et cravate bleue, long, mince, oppose un calme inédit à ceux qui tiennent son échec pour assuré : « Je voudrais prouver que le pouvoir n'est pas forcément le refuge du cynisme, du scepticisme et de l'inaction. » **Ce matin-là**, Thierry Breton, le ministre des finances, a parlé de la dette publique avec des chiffres abyssaux. **Trois semaines plus tôt**, la France a dit non à l'Europe, cette maison commune paradoxale, parce que non démocratique, dont les architectes semblent avoir au fil du temps oublié les origines et les fins.*
>
> *Une morosité durable affecte notre pays, dirigé **depuis longtemps** par des gouvernants paralysés par leur prudence. Dernier en titre : Jacques Chirac. Les Français, tentés de vivre en arrière, dans une quiétude provinciale, s'enferment avec leurs plaintes, s'adonnent à la peur et au repentir, sans savoir qu'ils sont enviés, leur volonté trébuche.*
>
> *[…]*
>
> *Deux styles. Villepin reprend avec eux une discussion assez vive. « Vous avez les chiffres que Breton a donnés ce matin. Il faudra trouver les moyens de faire des économies. C'est **juin 1940**, nous sommes le dos au mur. Est-ce que les gens s'en rendent comptent ? »*
>
> ***Depuis juillet 2004**, les trois hommes se sont préparés à ce qu'ils considèrent comme une mission de la dernière chance. La Place Beauvau, sous Villepin, est devenue une sorte de laboratoire clandestin de la société française, où le ministre de l'intérieur a beaucoup reçu, écouté, sans jamais rien en laisser savoir.*

Tableau 5.3 – *Exemple de cadres temporels (article paru dans l'édition du 4 Octobre 2005, Le Monde).*

Les introducteurs de cadre temporel ont été expressément mis en gras. Supposons que l'on veut définir une structure constituée des cadres temporels et que l'on veut que ses éléments (*i.e.* les cadres temporels) soient ordonnés chronologiquement. Il est clair que cette structure ne peut pas être cadencée à une hiérarchie car la séquentialité textuelle n'est pas la même que la séquentialité chronologique.

5.3.6. Critiques aux représentations arborescentes

Certains auteurs [Webber *et al.* 2003], [Wolf et Gibson 2005] ont critiqué le choix de représentations arborescentes pour modéliser les phénomènes discursifs. Par exemple, dans [Wolf et Gibson 2005], les auteurs montrent comment des structures de graphe sont nécessaires afin de représenter la cohérence discursive. Je prends l'exemple suivant de l'article cité :

0. Schools tried to teach students history of science.
1. At the same time they tried to teach them how to think logically and inductively.
2. Some success has been reached in the first of these aims.
3. However, none at all has been reached in the second.

Tableau 5.4 – *Exemple de texte pour la cohérence discursive (extrait de [Wolf et Gibson 2005]).*

Pour l'analyse du texte, l'ensemble de relations de cohérence correspond à celles développées dans [Hobbs 1985] et [Kehler 2002] :

Cause-effect	Because
Violated expectation	Although ; But
Condition	If…Then ; As long as
Similarity	(and) similarly
Contrast	But ; However
Elaboration	Also ; Furthermore
Attribution	…said ; According to…
Temporal sequence	Before ; Afterwards

Tableau 5.5 – *Relations de cohérence discursive considérées (extrait de [Wolf et Gibson 2005]).*

Pour le texte et les relations précédentes, l'analyse est la suivante :

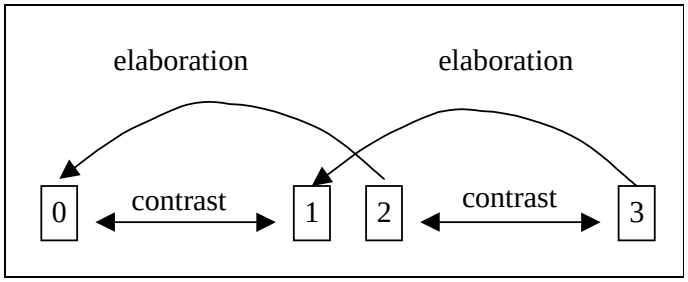

Figure 5.2 – *Analyse de cohérence discursive (extrait de [Wolf et Gibson 2005]).*

Dans l'exemple ci-dessus, la structure de cohérence discursive correspond à un graphe. D'autres exemples sont présentés dans l'article, qui montrent qu'il existe des phénomènes discursifs non représentables avec des structures hiérarchiques et qui doivent faire appel à des structures de graphe.

5.4. Synthèse

J'ai réalisé une analyse des différents types de représentations des textes, en identifiant trois types possibles : *synthétisées*, *plates* et *structurées*. Puisque le type de traitement à effectuer détermine la représentation la plus convenable, les représentations présentées s'avèrent insuffisantes pour la navigation textuelle.

Dans le cadre de cette thèse, je ne m'intéresse pas aux représentations synthétisées car il y a une perte d'information les rendant non utilisables pour la navigation textuelle, où l'information de chaque unité textuelle peut jouer un rôle important. De même en ce qui concerne les représentations plates car nous sommes intéressées à représenter la structure des textes. Les représentations structurées sont plus adéquates dans notre cadre, mais il manque la possibilité de représenter des structures discursives complexes qui ne suivent pas une hiérarchie. Excepté le cas de cadres de discours, où il existe un nouvel objet (le cadre), les autres propositions ne permettent pas de manipuler en tant qu'objets les nouvelles structures.

Aussi, je m'intéresse spécifiquement à une représentation permettant d'exprimer à la fois la structure syntaxique des textes, les structures discursives, et les valeurs sémantiques des constituants textuels, car toutes ces informations vont pouvoir servir de guide à la lecture dans le cadre de la navigation textuelle. L'intérêt de combiner la structure syntaxique avec des phénomènes discursifs, et le fait de souhaiter manipuler certaines structures complexes en tant qu'entités, motivent la définition d'une représentation spécifique pour la navigation textuelle, développée dans le chapitre suivant.

Chapitre 6

Proposition de représentation des textes pour la navigation textuelle

6.1. Introduction

Les insuffisances des représentations présentées au chapitre précédent ont motivé la définition d'une représentation des textes spécifique à la navigation textuelle. Pour ce faire, je me suis inspiré à la fois des propositions de [Crispino 2003] et de celles du modèle TEI Lite [TEI], qui sont structurées et relativement proches de la représentation souhaitée, en me fixant comme objectifs les points suivants :

- ne pas restreindre le type d'unités textuelles qui composent un texte à un ensemble prédéterminé ;

- offrir à la fois une organisation hiérarchique des unités textuelles et une autre permettant d'exprimer des relations non hiérarchiques ;

- considérer les titres comme des unités textuelles ;

- admettre que toute unité textuelle, y compris les titres et les relations non hiérarchiques, est susceptible d'avoir un nombre non limité d'annotations de nature quelconque.

Afin de satisfaire ces objectifs, ma proposition consiste alors à représenter un texte comme une hiérarchie d'unités textuelles de base, permettant la définition d'unités plus complexes susceptibles de ne pas suivre la hiérarchie établie. Toutes les unités de base sont typées, ce qui offre une souplesse non négligeable car au lieu d'avoir, par exemple, une unité textuelle *section* ou *paragraphe*, nous disposons d'une unité textuelle générique pouvant être d'un type quelconque, qui peut éventuellement instancier en un type *section* ou *paragraphe*. Il ne s'agit pas ici d'une nuance mais d'une décision conceptuelle d'après laquelle l'utilisateur[27] n'est

[27] Utilisateur au sens large, pouvant être, entre autres, un système générateur ou encodeur de textes.

pas restreint par un jeu prédéfini d'unités. Au contraire, celui-ci peut définir ces types d'unités à volonté.

Le texte est donc composé des trois éléments : son *titre*, un ensemble de relations non hiérarchiques, regroupés dans une unité nommée *tête*, et une hiérarchie d'unités textuelles, formulée dans une unité nommée *corps*.

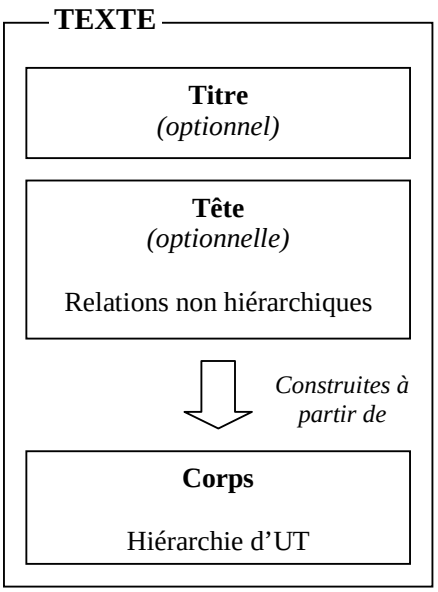

Figure 6.1 – *Composition d'un texte.*

Dans le diagramme ci-dessus, la flèche indique que les relations exprimées dans la *tête* sont construites à partir des UT existantes dans le *corps*. Je développe par la suite les trois composants principaux d'un texte.

6.2. Les titres

Pourquoi concéder un statut spécial aux titres au lieu, par exemple, de confier à l'utilisateur le soin de définir des unités textuelles de type « Titre » ? Les titres pouvant faire partie des relations entre constituants textuels (par exemple une relation rhétorique), leur légitimation comme une entité allant au-delà du simple attribut d'un constituant textuel est, de mon point de vue, justifiée. De ce fait, un titre est composé d'unités textuelles de base dont leur organisation dépendra de l'analyse faite par l'encodeur. Ainsi, le titre « *Outils dynamiques de fouilles textuelles* » peut se représenter comme une seule unité (un syntagme nominal),

une séquence d'unités (cinq mots), une hiérarchie d'unités (un syntagme nominal composé du syntagme nominal « *Outils dynamiques* » composé de deux mots, suivi du complément adjectival « *de fouilles textuelles* » composé de trois mots).

Malgré une diversité d'études sur les titres de presse [Ho-Dac *et al.* 2004], il en existe peu sur leur rôle dans des documents longs structurés hiérarchiquement en sections et sous-sections titrées [Jacquemin *et al.* 2005]. Intuitivement, même si les titres surpassent le simple rôle d'attribut, il est cependant certain que ceux-ci sont généralement reliés à un constituant textuel, celui-ci constituant leur empan. Si l'on accepte l'hypothèse selon laquelle une unité textuelle (par exemple une section) ne peut pas avoir plus d'un titre attaché directement, on peut considérer que toute unité textuelle est plausible d'en avoir au maximum un. Je considère important de manifester cette double nature : le titre relié à une unité textuelle et le titre composée d'unités textuelles. Le titre est donc une unité textuelle particulière formée, au sens le plus général, par une hiérarchie d'unités textuelles de base, et étant toujours affectée à une unité textuelle de base. Ces unités textuelles de base sont présentées par la suite.

6.3. Le corps du texte : les éléments hiérarchiques

L'unité textuelle de base (dorénavant *UT*) comporte différents attributs selon qu'elle s'agisse d'une *feuille* ou d'un *nœud* dans la hiérarchie d'UT. De plus, les UT possèdent des attributs fixes et d'autres dynamiques, pouvant avoir, optionnellement, un titre.

Une unité textuelle *feuille* correspond à une unité appartenant à la granularité la plus fine dans la hiérarchie définie, et c'est elle qui porte l'information lexicale, dénotée par un attribut fixe nommé *chaîne*. Une unité textuelle *nœud* sert pour représenter la hiérarchie textuelle, permettant l'emboîtement des différentes unités. En conséquence, elle ne porte pas directement d'information lexicale sinon qu'elle est composée des UT filles. Il est clair que l'information lexicale véhiculée par une UT nœud peut s'inférer à partir des unités filles, en faisant un parcours dans l'arborescence.

6.3.1. Notation graphique

J'utiliserai une notation graphique afin de montrer les différentes propriétés des UT et les rapports existants entre elles. Un exemple de cette notation est montré dans la figure 6.2.

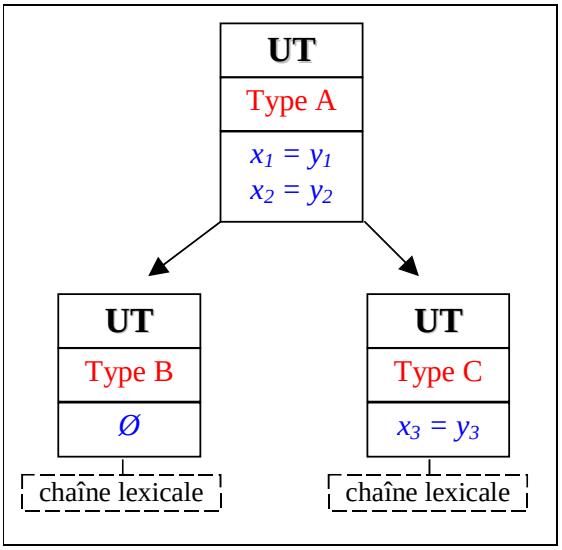

Figure 6.2 – *Exemple de diagramme d'UT.*

Les boîtes représentent des UT, étant chacune libellée avec la mention UT, le type d'UT et un ensemble d'attributs dynamiques (éventuellement vide) que l'on appellera *annotations* afin de les distinguer des attributs fixes. Dans cet exemple, une UT de type A contient deux UT filles étant à la fois des feuilles (cette inclusion est explicitée par des flèches) de type B et C respectivement. L'UT de type B, d'après l'arborescence exprimée, apparaît dans le texte avant que l'UT de type C. En outre, les chaînes lexicales des UT feuilles sont affichées à l'intérieur d'une boîte pointillée, et les UT de type A et C contiennent des annotations, indiquées par une paire nom-valeur sous forme d'affectation d'une variable. Dans l'exemple de la figure 6.2, l'UT de type A contient une annotation de nom x_1 et valeur y_1, et une autre de nom x_2 et valeur y_2, tandis que l'UT de type C contient une annotation de nom x_3 et valeur y_3 et celle de type B n'a aucune annotation.

Si une UT contient un titre, le diagramme est similaire mais celle-ci pointe de plus vers une boîte représentant le titre, qui contient les UT qui le composent, comme le montre l'exemple de la figure 6.3.

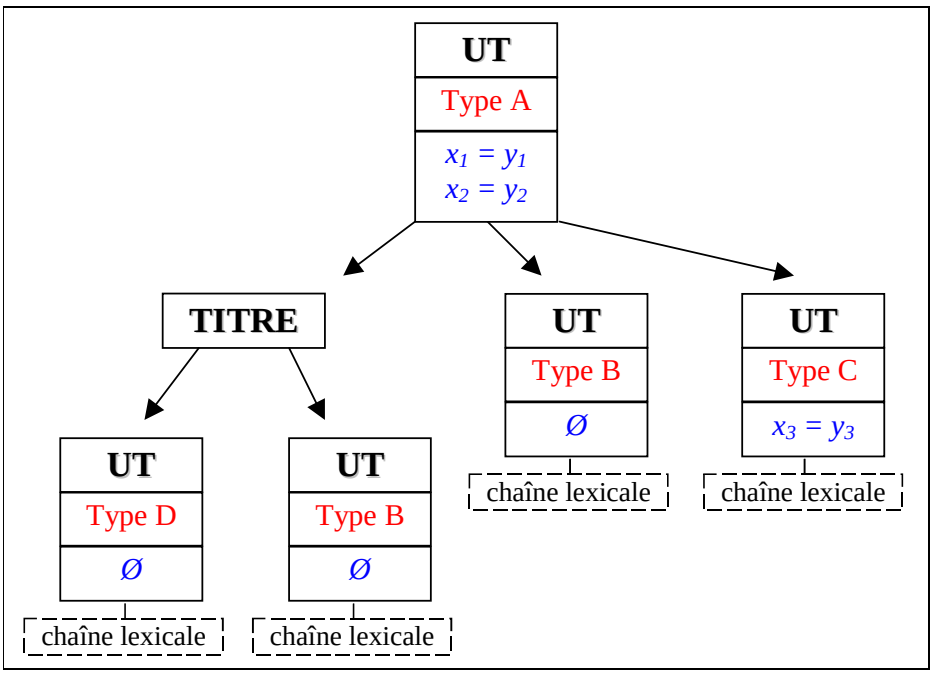

Figure 6.3 – *Exemple de diagramme d'UT avec un titre.*

6.3.2. Exemples de représentation

Prenons comme exemple un fragment d'un texte attesté (*cf.* tableau 6.1) emprunté au projet *NaviLire* (*cf.* chapitre 12).

Une carte de crédit utilisant la technologie biométrique
Deux compagnies danoises Scanecotech et Ewire, ont présenté le 7 mars 2005 le prototype d'une nouvelle carte de paiement qui utilise les empreintes digitales pour identifier son utilisateur. La carte biométrique ressemble à une carte de crédit et est conforme à toutes les spécifications imposant les dimensions physiques ainsi que les propriétés des cartes de paiement traditionnelles.
…

Tableau 6.1 – *Exemple de texte à représenter*[28].

[28] Extrait du « Bulletin Scientifique et Universitaire du Danemark », L'Ambassade de France au Danemark, Mars-Avril 2005, page 1.

Dans cet exemple, la première phrase (de fait un syntagme nominal) constitue le titre d'une section composé de plusieurs paragraphes, dont le premier est composé des deux phrases qui y sont affichées.

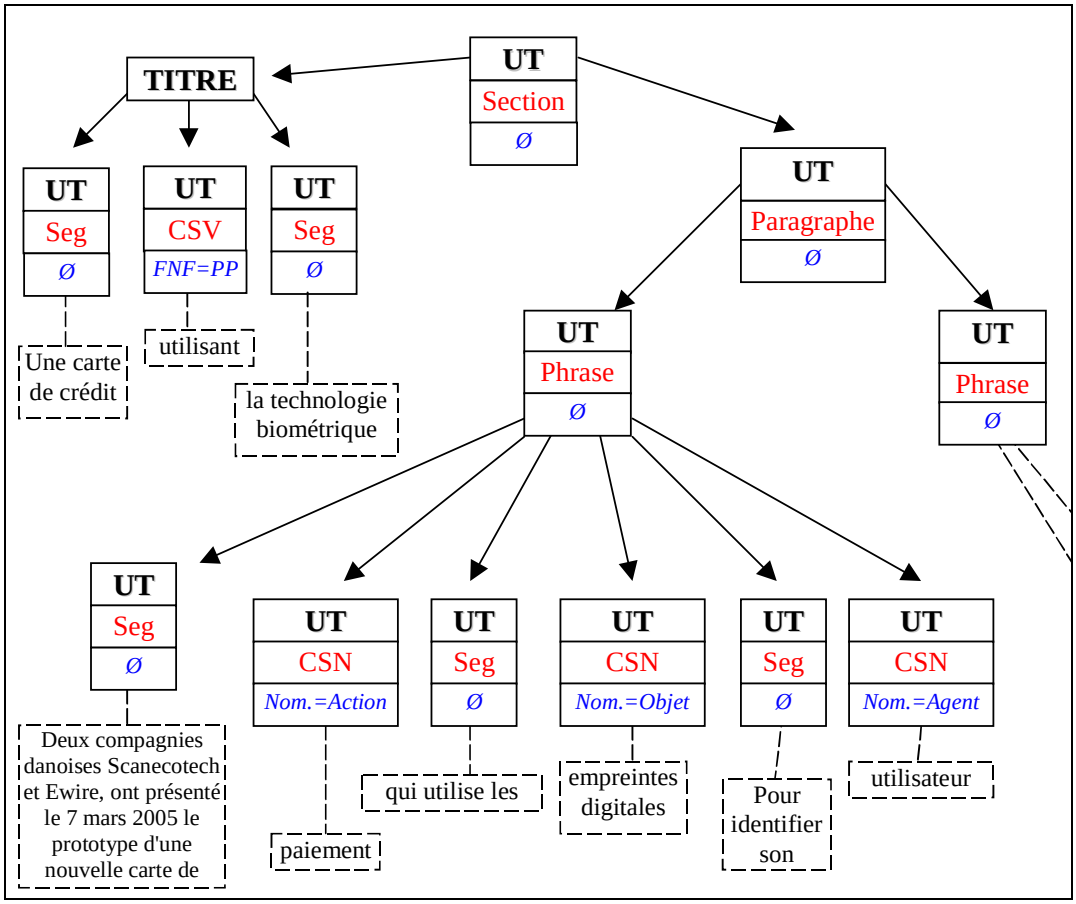

Figure 6.4 – *Diagramme d'UT pour le texte d'exemple.*

La figure 6.4 montre le diagramme correspondant à ce texte (sauf pour la deuxième phrase, dont la composition n'apparaît pas dans le diagramme). L'analyse linguistique, exhibée partiellement pour des raisons d'espace[29], à été effectuée par Lita Lundquist dans le cadre du projet NaviLire (*cf.* chapitre 12). Les UT qui y participent sont, selon leur types : *Section*, *Paragraphe*, *Phrase*, *Seg* (« Segment »), *CSV* (« Cohérence Statique Verbale ») et *CSN* (« Cohérence Statique Nominale »). Au niveau des annotations, on y trouve : *FNF* (« Formes non finies ») ayant la valeur *PP* (« Participe Présent »), et *Nom.* (« Nominalisation ») ayant les

[29] Certains libellés ont été raccourcis et quelques types d'UT enlevés.

valeurs *Action*, *Objet* et *Agent*. Notons que les titres, s'agissant d'UT, peuvent avoir des annotations.

Dans ce chapitre je laisserai de côté la question sur l'analyse effectuée (UT choisies, annotations définies, valeurs affectées, etc.), qui sera développée au chapitre 13.

6.3.3. Granularité des UT et notion d'héritage d'annotations

Examinons un deuxième fragment de texte (*cf.* tableau 6.2), emprunté à [Couto *et al.* 2004] et encodé par nos soins afin d'être utilisé par la plate-forme de navigation textuelle *NaviTexte*, qui sera présentée au chapitre 12. Le diagramme correspondant à l'analyse effectuée pour ce fragment est affiché dans la figure 6.5, où les UT ont été numérotées, selon un parcours en profondeur de l'arborescence, pour simplifier leur mention.

L'importance quantitative de l'investissement étranger est cependant moins significative de l'impact des firmes multinationales que le type de secteurs où elles se localisent. En Côte-d'Ivoire les firmes contrôlent pratiquement l'ensemble de l'industrie produisant pour le marché interne. Au contraire, l'accès à ce dernier leur est interdit dans la plupart des branches en Corée du Sud. Cette situation a des conséquences décisives, particulièrement sur trois variables stratégiques du processus de développement : l'allocation des ressources, le modèle de consommation et l'intégration en amont de l'activité industrielle.

Tableau 6.2 – *Exemple de texte à représenter (2).*

6.3.4. Notion d'héritage entre les UT

Plusieurs points sont à signaler dans l'exemple précédent. Le premier correspond à la granularité des UT : notons que l'encodeur du texte n'est pas obligé d'encoder toutes les UT de manière identique. Ainsi, il a choisi de dénoter l'UT numéro 2 comme une phrase, n'allant plus loin dans son analyse, tandis que les autres trois UT de type « Phrase » sont composées d'UT filles. Autrement dit, l'encodeur n'est pas obligé d'aller jusqu'au type d'UT le plus profond dans la hiérarchie (ce qui, de fait, n'existe pas dans la représentation proposée) pour pouvoir exprimer la chaîne lexicale.

Le deuxième point correspond à deux notions importantes : l'*héritage d'annotations* et la *synthèse de chaînes lexicales*.

Les annotations d'une UT peuvent être héritées ou bien par les UT descendants (*héritage descendant*), ou bien par les UT ascendants (*héritage ascendant*), ou bien dans les deux cas.[30] L'encodeur du texte est censé décider pour chaque instance d'annotation le *mode d'héritage*, choisissant parmi les valeurs suivantes : *aucun*, *ascendant*, *descendant* et *bidirectionnel*.

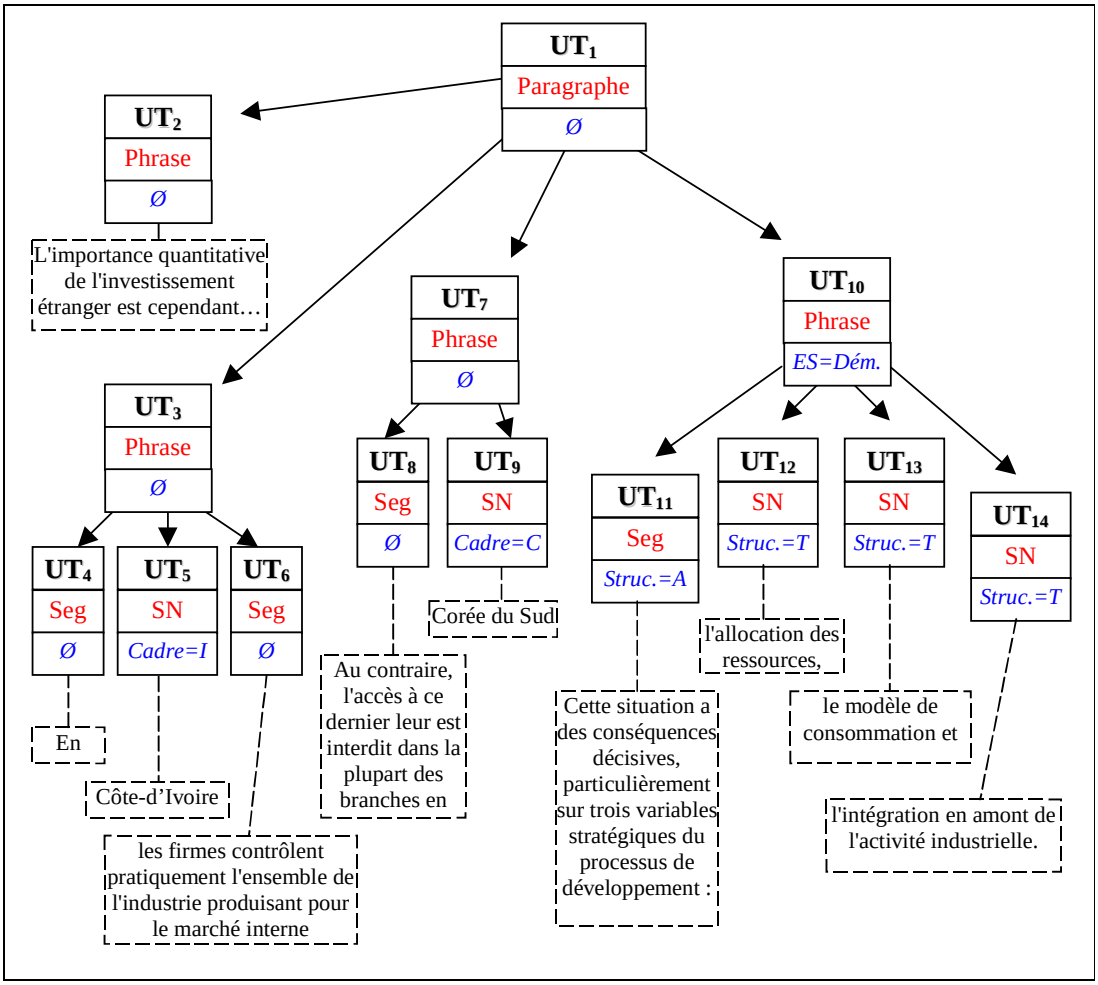

Figure 6.5 – *Diagramme d'UT pour le texte d'exemple (2).*

Dans l'exemple, l'UT numéro 10 est une phrase annotée comme une phrase démonstrative (*ES* étant « Étiquette Sémantique » et *Dém.* la valeur « Démonstration »), composée d'un segment et trois syntagmes nominaux. Chacune de ces quatre UT filles, en tant que consti-

[30] Je préfère utiliser les notions d'héritage ascendant et descendant d'annotations, au lieu d'utiliser les termes classiques aux grammaires attribuées qui sont *attributs hérités* et *attributs synthétisés* [Aho *et al.* 1986].

tuants textuels individuels, ne constituent pas une « Démonstration ». Par exemple, le syntagme nominal « l'allocation des ressources » n'est pas une « Démonstration ». En conséquence, de mon point de vue, l'annotation de la phrase ne devrait pas être propagée à celles-ci. C'est-à-dire qu'il ne devrait pas y avoir d'héritage descendant. De même pour l'héritage ascendant, car le fait d'avoir une phrase étiquetée en tant que « Démonstration » ne nous permet pas d'inférer que le paragraphe parent peut lui aussi être étiqueté comme « Démonstration ».

6.3.5. Discussion méthodologique

Il faut souligner que la notion d'héritage est intimement liée à la sémantique que l'on attribue aux annotations.

Ainsi, supposons qu'un utilisateur encode une annotation nommée *Thème*, ayant comme valeur le thème sur lequel porte un fragment textuel. Ce faisant, toutes les UT portant sur ce thème seront affectées directement par cette annotation. Néanmoins, si l'utilisateur interprète de plus qu'un fragment textuel discourt sur un thème déterminé si l'un de ses constituants le fait, alors l'héritage ascendant pour cette annotation est tout à fait adéquat.

L'héritage descendant peut être utilisé pour des raisons d'économie d'annotation : si toutes les UT filles d'une UT ont la même annotation avec la même valeur, et toutes les instances de cette annotation sont définies comme héritant de manière descendant, alors l'encodeur (un système ou un humain) peut supprimer cette annotation des filles et l'affecter à l'UT parente. D'autres combinaisons sont envisageables, pourvu que tous les descendants de l'UT parente (que ce soit par annotation directe ou par annotation héritée) partagent l'annotation.

Pour indiquer dans les diagrammes le mode d'héritage, chaque annotation est préfixée d'un symbole. Si l'annotation ne se propage pas (*i.e.* il n'y a ni d'héritage ascendant ni descendant), le préfixe est vide. Dans le cas contraire, les symboles « ↑ », « ↓ » et « ↕ » signaleront l'héritage ascendant, descendant et bidirectionnel, respectivement. Par exemple, pour indiquer qu'un UT porte une annotation sur le thème X à propager vers les UT ascendants, on noterait comme annotation : « ↑ Thème = X ».

Quant aux chaînes lexicales, bien qu'une UT nœud n'en possède pas une, force est de constater que celle-ci est facilement repérable à partir des UT descendants. Il suffit de *calculer* la chaîne lexicale de l'UT nœud comme l'enchaînement des chaînes lexicales appartenant aux UT feuilles du sous arbre dont cette UT est la racine. L'ordre d'enchaînement est calculé à partir d'un parcours en profondeur préfixé du sous arbre. Par exemple, la chaîne lexicale de

l'UT numéro 3 correspond à l'enchaînement des chaînes lexicales des UT numéro 4, 5 et 6, respectivement.

Certaines UT dont la mise en forme est très particulière peuvent être représentées dans cette représentation. Prenons l'exemple des tables dont un schéma de représentation est montré dans la figure 6.6.

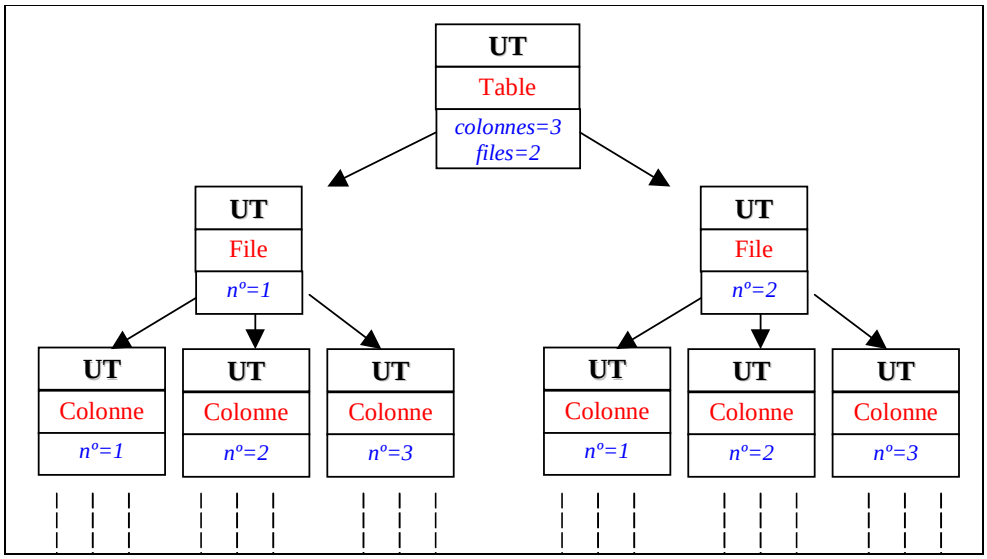

Figure 6.6 – *Schéma de diagramme d'UT pour représenter une table.*

Une table, parmi d'autres éléments textuels complexes comme les puces, les énumérations ou les schémas numérotés, peut ainsi être représentée, à condition qu'il existe une coordination entre codification et mise en forme.

6.3.6. Synthèse

En résumé, le *corps* du texte est formé par des *UT* typées s'agencent sous forme de hiérarchie. Le fait de permettre à l'encodeur de typer librement les UT constitue un bon compromis entre flexibilité et expressivité. D'autre part, une UT comporte certains attributs fixes : *type*, *titre*, *chaîne lexicale* (si elle est une feuille) et certains attributs optionnels dynamiques, nommés *annotations*. Ces annotations sont exprimées comme une liste de paires nom-valeur, ayant chacun un mode d'héritage parmi ces quatre valeurs : *aucun*, *ascendant*, *descendant* et *bidirectionnel*.

6.4. La tête du texte : les éléments non hiérarchiques

Afin de représenter des constituants textuels ne respectant pas la hiérarchie établie dans le *corps* du texte, l'encodeur dispose d'une suite de constructeurs de nouvelles unités textuelles. Les constructeurs possibles sont : *ensemble*, *séquence*, *référence* et *graphe*, correspondant à des objets mathématiques connus, qui disposent des propriétés prouvées, facilement manipulables. Il est clair que d'autres objets mathématiques auraient pu être introduits en tant que constructeurs de nouvelles unités textuelles. Je prends l'exemple des *tables n-dimensionnelles*, où pour n égal 1 on obtient une *tuple*. Bien qu'il s'agisse d'un objet valide et intéressant d'un point de vue formel, je n'ai pas trouvé d'exemples dans les textes motivant son utilisation. Il ne faut pas ici confondre un constructeur de type table avec une table que l'on peut trouver dans un texte, cette dernière étant un cas particulier de la première. En effet, les constituants d'une table de n dimensions peuvent être disséminés tout au long du texte, car ceux-ci ne sont pas obligés d'y apparaître dans un ordre déterminé[31]. Les seuls exemples trouvés justifiant l'introduction des tables en tant que nouveaux éléments textuels, correspondent au cas classique d'une table bidimensionnelle qui a une mise en forme explicite dans un texte, et qui peut être représentée dans le *corps* en utilisant une codification comme celle suggérée dans la section précédente.

6.4.1. Les constructeurs d'unités textuelles

Les nouvelles unités textuelles construites vérifient ces trois règles :

- elles sont construites exclusivement à partir d'UT de base existantes dans le *corps* (*i.e.* elles ne peuvent pas créer de nouvelles UT de base) ;

- elles peuvent avoir des annotations leur appartenant, ces annotations ne pouvant pas être héritées par les UT de base ;

- elles peuvent se combiner afin de construire des éléments plus complexes.

Intuitivement, la première et la troisième règle proposent une définition inductive des constructeurs. En guise d'exemple, restreignons cette définition au seul cas des ensembles :

[31] Ce constat est fait à priori car je n'ai pas trouvé d'exemples le confirmant.

- Cas de base : si u *est une UT, alors {*u*} est un ENSEMBLE.*
- Cas inductif : si E1 *et* E2 *sont des ENSEMBLES, alors* E1 ∪ E2 *est un ENSEMBLE.*
- Clause de clôture.

Tableau 6.3 – *Exemple d'une définition inductive des ensembles d'UT.*

J'omets le cas de base correspondant à l'ensemble vide car il n'est pas pertinent dans le cadre de la représentation des textes. Ces définitions deviennent intéressantes lorsque l'on combine différents types de constructeurs (par exemple : une séquence où les trois premiers éléments sont des UT de base, le quatrième est un ensemble, le cinquième une UT de base, et le sixième une référence). Je m'abstiendrai cependant d'introduire les nouveaux constructeurs de cette manière, privilégiant la notation graphique utilisée auparavant.

Je présente par la suite les différents constructeurs, en motivant leur définition à l'aide d'exemples concrets.

6.4.2. Ensemble

Un *Ensemble* permet de décrire un ensemble d'UT[32] entre lesquelles, selon le point de vue de l'encodeur, existe une relation d'équivalence dont sa nature peut être exprimée en utilisant des annotations. En guise d'exemple, des UT de type syntaxique différents peuvent exprimer un même thème ou *topic* ; le cas le plus fréquent est celui des verbes et des déverbaux, comme dans « Jean aime Marie » et « l'amour de Jean pour Marie ». Un ensemble permettrait de traiter ces UT comme une unité ayant un sens propre, ce sens étant indiqué par le biais d'annotations (par exemple une annotation nommée « *Sentiment* » avec une valeur « *Amoureux* »).

Le constructeur d'ensembles peut s'utiliser également pour rassembler des UT de même type, quoiqu'il ne soit pas très commun. Ce typiquement le cas où l'encodeur, au lieu de définir un attribut spécifique pour une certaine propriété, par exemple un attribut « Référent Discursif », définit le type de l'UT comme « Référent Discursif ». Ce faisant, il déplace la sémantique des attributs au type de l'UT, et ce déplacement peut avoir un sens si l'attribut dé-

[32] Les constructeurs permettant créer de nouvelles UT, l'utilisation de la sigle UT réfère désormais tant une UT de base qu'une nouvelle UT.

placé n'a qu'une valeur possible car, si ce n'est pas le cas, l'encodeur sera de toute manière obligé d'introduire un attribut pour exprimer les valeurs, ce qui est redondant.

6.4.3. Séquence

Une *Séquence* est une suite ordonnée d'UT à laquelle l'encodeur attribue une cohésion. Elle permet de décrire des éléments discontinus dans un texte. L'intérêt de ce type de structure peut être illustré sur différents exemples. Le premier est illustré par le besoin d'annoter un syntagme verbal dont la continuité est brisée par la négation. Ainsi, il n'est pas possible dans le *corps* d'indiquer que dans la suite « ne sont pas stockées », le syntagme verbal composé de « sont » et de « stockées» constitue une seule unité. Un palliatif possible dans le *corps* consiste à déclarer les UT indépendamment et de les inclure de manière à ce que « sont » et « stockées » appartiennent à une même UT parente (*cf.* figure 6.7). Mais d'une part, l'annotation « Verbe » au passif concerne les trois UT incluses et par conséquent le terme « pas » possédera cet attribut, ce qui n'est pas correct[33] ; d'autre part, les UT de numéro 4 et 6 ne sont pas liées entre elles.

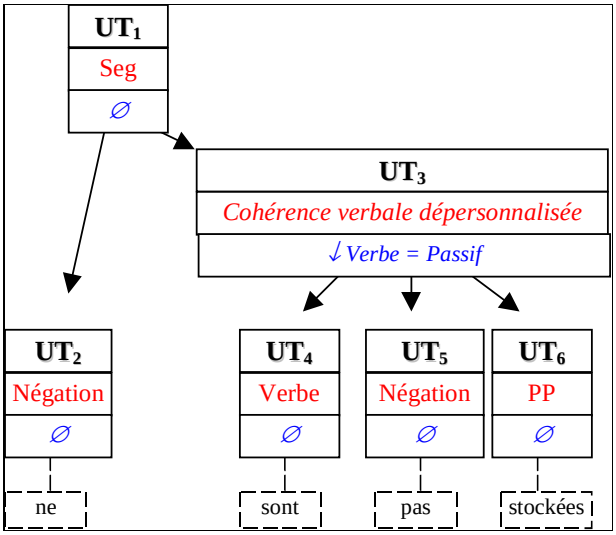

Figure 6.7 – *Diagramme d'UT incorrect pour représenter un syntagme verbal.*

[33] Clairement, cette annotation peut se propager aux UT filles pertinentes, mais le problème demeure car l'annotation concerne véritablement l'unité parente.

La déclaration d'une séquence (*cf.* figure 6.8) prend sens dans ce cas, à condition qu'il soit important d'indiquer l'ordre existant entre les éléments constituant le syntagme verbal car, dans le cas contraire, un ensemble suffirait. Notons que, dans le diagramme, l'ordre des flèches sortant de l'élément « Séquence » (de gauche à droite) indique l'ordre des UT dans la séquence.

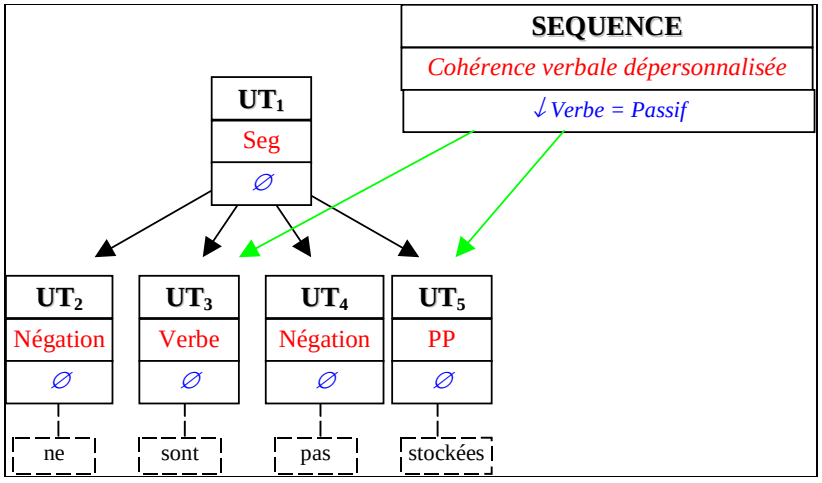

Figure 6.8 – *Création d'une* Séquence *pour représenter un syntagme verbal discontinu.*

Les cadres thématiques [Porhiel 2001, 2003] constituent un deuxième exemple de l'intérêt de cette structure : « *Les introducteurs thématiques constituent une classe cohésive dont les éléments, de nature abstraite, sont morphologiquement des prépositions* (en ce qui concerne, pour ce qui est de, à propos de, sur, etc.) *et des anaphores résomptives* (à ce sujet, à ce propos). *Ces unités lexicales assurent une cohérence thématique, le plus souvent au niveau local dans un texte. (…) Au niveau textuel, les introducteurs thématiques servent à introduire un élément dont il va être question dans la proposition et les phrases subséquentes : ils en précisent le thème. Ce sont des balises linguistiques identifiables dans tout type de texte : elles pointent sur une thématique et renforcent une segmentation thématique déjà opérée au niveau lexical (pronoms, répétitions, synonymie) et/ou orthographique (marques de paragraphes). Les introducteurs ont aussi pour fonction de séquencer explicitement des parties d'un texte : ils attirent l'attention sur un référent et le rendent saillant par rapport à d'autres choix possibles ; ils organisent l'information dans un texte …* » [Couto *et al.* 2004].

Le texte affiché dans la figure 6.9 exemplifie la fonction cohésive de ces marques. Dans cet exemple, une séquence afférente permettant de représenter les introducteurs de cadre est illustrée.

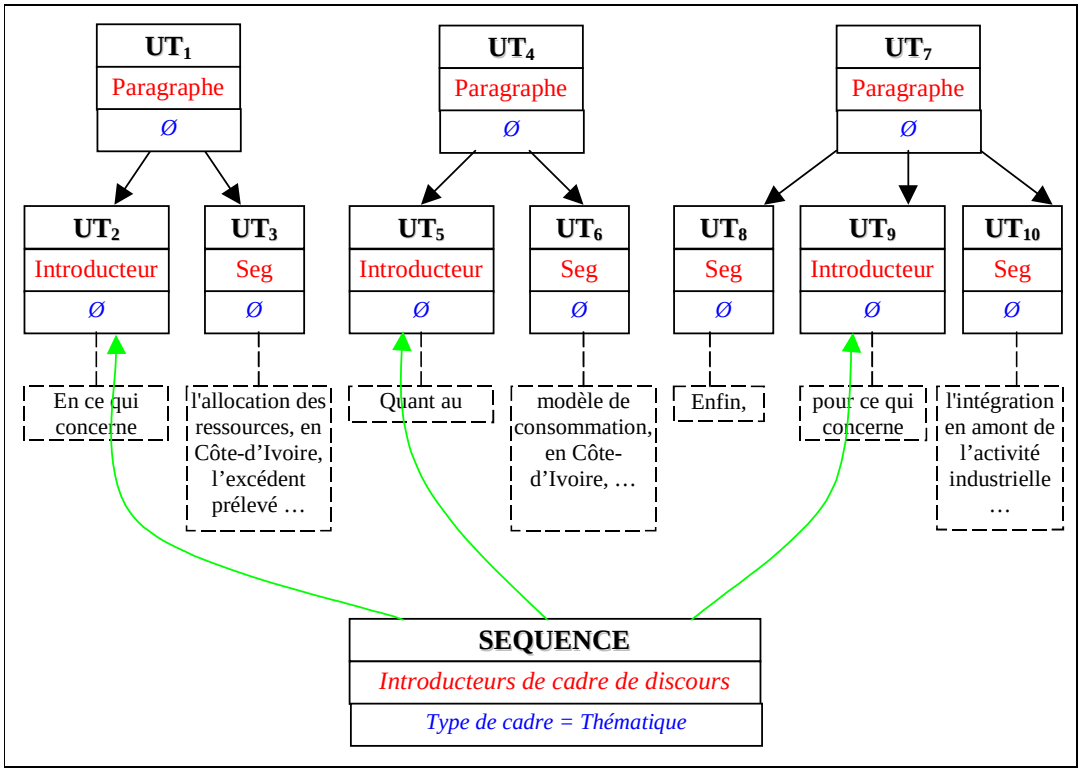

Figure 6.9 – *Création d'une* Séquence *pour représenter les introducteurs de cadres thématiques.*

Le dernier exemple concerne les chaînes de référence lexicales [Morris *et* Hirst 1991] [Barzilay *et* Elhadad 1997]. Une chaîne de référence lexicale est constituée par l'ensemble des syntagmes nominaux qui réfèrent à un même objet. Ainsi, dans un article de presse (Le Figaro, le 16 juillet 2004) sur l'amnistie fiscale[34], on trouve pour référer à « La taxe sur les fonds rapatriés en France », dix-sept corrélats linguistiques référant au même référent discursif dont par exemple « La taxe sur les fonds rapatriés en France », « une taxe sur les fonds placés à l'étranger et rapatriés en France », « une telle mesure », « elle », etc. La déclaration d'une séquence composée de toutes ces UT permet de concrétiser la chaîne de référence lexicale et offre ainsi les moyens de traiter simplement des phénomènes linguistiques très fréquents.

6.4.4. Référence

Les relations orientées de type 1 à 1 entre deux UT peuvent se décrire en utilisant le constructeur *Référence*, composé de deux éléments : l'UT *référente* et l'UT *référée*.

[34] Ce texte fait partie des textes recueillis et analysés par Lita Lundquist.

Un premier exemple d'utilisation (*cf.* figure 6.10) concerne la représentation d'une UT anaphore pronominale qui réfère au syntagme nominal. La déclaration d'une référence entre le syntagme nominal « une taxe sur les fonds placés à l'étranger » et l'anaphore pronominale « Elle » permet de représenter simplement la relation antre une anaphore et son référent. Dans le diagramme, la première flèche sortant de l'élément « Référence » indique l'UT référente et la deuxième, l'UT référée.

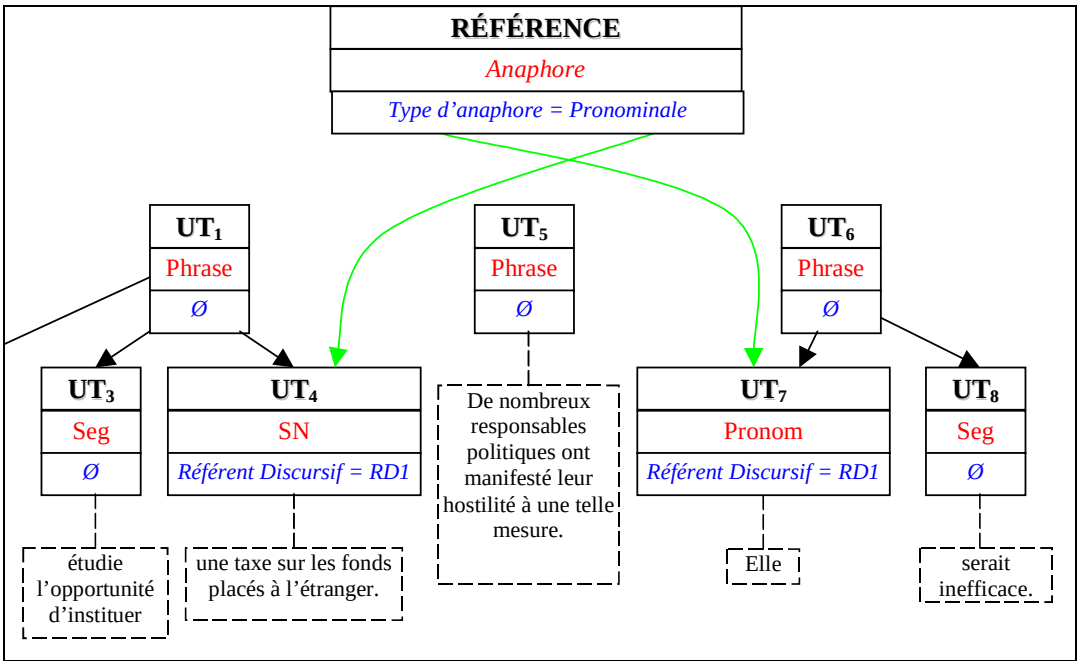

Figure 6.10 – *Utilisation d'une* Référence *pour les anaphores pronominales.*

Notons qu'il est également possible de représenter une cataphore ou une anaphore résomptive, pouvant ainsi lier des UT de type différents. D'autre part, il est possible de déclarer plusieurs références ayant la même UT référée, ce qui permet de lier toutes les anaphores à leur référent.

Un deuxième exemple est celui de la représentation des relations rhétoriques entre un noyau et un satellite proposées par la RST [Mann *et* Thompson 1988].

La différence avec une déclaration d'une séquence tient au fait que les UT qui référent n'ont pas de liens déclarés entre elles. Enfin, une référence n'étant pas une fonction, une même UT peut référer différentes UT. Au cas le plus général, toutes les références conforment un graphe, qui est le constructeur que nous considérerons par la suite.

6.4.5. Graphe

Un *Graphe* permet de décrire les relations entre différentes UT. Cette structure correspond exactement à la notion de graphe en mathématiques [Berge 1958] : on distingue les *sommets* du graphe, dans le cas présent ce sont des UT déclarées dans le *corps*, et les *arcs* orientés entre ces sommets, qui représentent les relations entre les UT. Graphiquement, un graphe peut s'afficher comme un ensemble de références.

Cette structure permet de représenter des phénomènes linguistiques complexes comme par exemple l'imbrication de différents cadres de discours [Charolles 1997], les différentes pistes de cohérence qui structurent un texte [Lundquist 1990_b], ou encore les expressions des sentiments d'un personnage dans un roman [Mathieu 2005]. Pour ce dernier exemple, dans le roman Madame Bovary, les sentiments de Emma vis-à-vis de son mari, son amant, etc., sont annotés avec différentes intensités. Cette annotation se représente sur un graphe comportant des liens gradués. Cet exemple d'application sera développé au chapitre 13.

6.5. Constat critique de la représentation proposée

Toute représentation de texte est en même temps une prise de position et un choix dont il convient de mesurer les conséquences. Celle proposée ici, bien que raisonnablement indépendante d'un type d'application déterminée, est focalisée sur la navigation textuelle, qui se fonde sur l'existence dans les textes des informations servant de guide à la lecture.

Parmi les aspects positifs de cette représentation, le fait de permettre à l'encodeur de typer librement les UT constitue un bon compromis entre flexibilité et expressivité. Certaines unités textuelles habituelles dans les textes, tels que les tables ou les énumérations, peuvent être en conséquence encodées. De plus, la possibilité de construire de nouvelles unités textuelles ne suivant pas une hiérarchie textuelle définie offre les moyens de traiter assez aisément des phénomènes linguistiques très fréquents.

Néanmoins, on peut adresser divers critiques à la représentation proposée, dont je mentionnerai trois. La première concerne le fait que la sémantique des constituants textuels (par exemple : « cette UT est une table ») est indépendante du texte et dépend de la dynamique existante entre encodeur et décodeur (*i.e.* la sémantique des UT n'est pas guidée par un constituant textuel déterminé mais elle doit être correctement décodée à partir des types et annotations des différentes UT). C'est le problème classique de trouver le bon compromis entre spécificité et généralité.

La deuxième critique vient du fait que les nouvelles UT construites dans la *tête* en utilisant les constructeurs *ensemble*, *séquence*, *référence* et *graphe*, risquent, d'une part, de devenir très complexes et, d'autre part, d'introduire des erreurs ou une certaine redondance dans la représentation. Un exemple de redondance peut être le fait d'encoder comme une séquence une suite d'UT étant contiguës dans le texte, et qui en conséquence sont déjà séquenciées. Il peut cependant s'agir d'une option de l'encodeur qui voudrait expliciter cette séquence pour une raison particulière. Un autre exemple d'erreur est illustré par le cas où l'encodeur exprime une relation de nature univoque entre deux UT, en utilisant le constructeur de références, mais il le fait deux fois, reliant la première UT avec la deuxième et vice-versa. A priori, la représentation n'offre pas de mécanismes de control de ce type d'erreurs.

La troisième critique concerne les éléments iconographiques et certains éléments de mise en forme tels que la numérotation des pages ou les entêtes : ils ne sont pas pris en considération par la représentation, quoiqu'ils soient habituellement présents dans les textes.

6.6. Synthèse

Puisque le traitement informatique d'un texte nécessite une représentation de celui-ci afin d'être manipulé pour un système, nous avons eu besoin de disposer d'une représentation de texte pour la navigation textuelle. Une représentation abstraite a été proposée. Quitte à devenir complexe dans certains cas, la représentation permet à l'encodeur de typer librement les UT, ce qui constitue un bon compromis entre flexibilité et expressivité. Ce faisant, certaines unités textuelles habituelles dans les textes, tels que les tables ou les énumérations, peuvent être encodées. En outre, la possibilité de construire de nouvelles unités textuelles ne suivant pas une hiérarchie textuelle définie offre les moyens de traiter assez aisément des phénomènes linguistiques très fréquents.

D'après la représentation abstraite proposée, nous pouvons conceptualiser le texte comme un objet comportant une couche de base (le *corps*) où les unités textuelles élémentaires s'organisent selon un régime hiérarchique. Des nouvelles couches manifestant différentes structures et organisations textuelles particulières peuvent se définir sur la couche de base. En outre la possibilité d'articuler ces nouvelles structures (indiquer ses constituants et le type de structure : ensemble, séquence, etc.), la représentation autorise l'encodeur à y ajouter des annotations spécifiques.

Troisième partie

Modélisation des connaissances

Sommaire

Les travaux réalisés au long de différents projets de recherche nous ont permis d'apprécier la pertinence des aspects de visualisation et d'interaction pour certains systèmes de traitement automatique de textes, notamment pour ceux concernant la fouille de textes. Après avoir constaté que, dans les différents systèmes étudiés, les représentations visuelles d'un texte, les opérations de visualisation et les opérations de navigation sont, de fait, des connaissances imbriquées dans ces systèmes, l'idée de formaliser celles-ci est apparue souhaitable. Nous avons, en conséquence, dégagé la notion de vue d'un texte et les différentes opérations de visualisation et de navigation, en proposant un langage permettant de les exprimer.

En premier lieu, je développe dans cette partie les différentes connaissances à modéliser (chapitre 7). Je présente ensuite, dans le chapitre 8, *Sextant*, un langage déclaratif de formalisation de connaissances de visualisation et de navigation. Dans le chapitre 9, une sémantique opérationnelle est proposée, afin de formaliser le sens des constructions syntaxiques de *Sextant*. Cette sémantique définit également une partie de la sémantique d'un système interpréteur du langage (*i.e.* une plate-forme de navigation textuelle).

Chapitre 7

Les connaissances à modéliser

7.1. Pourquoi un langage de modélisation des connaissances ?

Dans le cadre du projet REGAL (*cf.* chapitre 1), nous avons été confrontés à la problématique de fournir à un utilisateur différentes vues d'un même texte et de lui offrir des moyens de parcourir les informations présentes dans ces différentes vues. D'un point de vue informatique, le résultat du projet s'est concrétisé sous la forme d'une plate-forme de fouille de textes pouvant visualiser le texte selon trois types de vues :

- linéaire : le contenu du texte s'affiche de manière linéaire, l'information textuelle affichée correspondant aux informations lexicales des feuilles de la hiérarchie textuelle ;

- arborescente : le contenu textuel et l'organisation des constituants textuels s'affichent sous la forme d'une structure de type arbre ;

- mosaïque : la structuration (thématique dans le cadre du projet) du texte s'affiche à l'aide d'une structure du genre *Tilebar* [Hearst 1995] qui visualise les valeurs thématiques des constituants.

La plate-forme logicielle ContextO, utilisée dans le cadre du projet RÉGAL, coordonne automatiquement les différentes vues d'un même texte et propose également certaines opérations de navigation entre les éléments textuels, permettant à un lecteur de parcourir dynamiquement le texte en passant d'un type d'information à un autre. C'est ce parcours qui a été dénommé *navigation textuelle* [Minel 2002] [Couto *et* Minel 2004$_a$, 2004$_b$].

Bien que les recherches menées dans le projet RÉGAL nous aient permis, d'une part, d'abstraire la notion de vue par rapport à un document et, d'autre part, d'incorporer la notion d'opération de navigation, ces notions restaient des connaissances amalgamées à la plate-forme. En effet, tant les représentations visuelles du texte que les opérations de navigation étaient spécifiques et encodées dans l'applicatif.

Or, la plupart des systèmes de fouilles de textes (et plusieurs de systèmes de traitement automatique de textes) sont généralement confrontés à une problématique identique : mon-

trer un texte (résultat ou source d'une analyse) comportant des informations issues d'un traitement spécifique et parcourir ce texte. Dès lors, une partie de l'affichage des résultats et de l'interaction avec l'utilisateur peut être abstraite et isolée du système de fouille et, pour ce faire, des mécanismes d'abstraction doivent être proposés. D'un point de vue architectural, un système de traitement automatique de textes peut ainsi être divisé en deux parties :

- un sous-système moteur prenant en charge le(s) traitement(s) à effectuer par le système, et générant des résultats ;

- un autre sous-système, chargé des interfaces graphiques et de la gestion de l'interaction avec l'utilisateur.

De la même manière que ContextO propose une approche alternative à la construction automatique de résumés, un système de type question-réponse, par exemple, pourrait afficher les textes pertinents par rapport à une requête, en guidant l'utilisateur dans la fouille (la navigation) de ces textes par le biais d'opérations de navigation précises, afin de trouver dans ce parcours la réponse ou les réponses à sa question. L'identification des segments textuels faisant partie de la réponse peut se complémenter à l'aide d'opérations de visualisation. De plus, le fait de trouver une réponse peut susciter pour l'utilisateur la formulation d'une autre question. Pour illustrer cette hypothèse, j'ai choisi un échantillon de cas du *corpus de questions* du projet eVEIL[35], où nous avons pu constater ce type d'enchaînement de requêtes :

Question :

En cas de risque majeur, qui est le responsable de l'information aux citoyens ?

Réponse :

L'information : Qui : le M.A.T.E., le ministère de l'intérieur, le maire.

Comment : Le préfet établit, sur financement du M.A.T.E. : le dossier départemental des risques majeurs, le dossier communal synthétique. Le maire établit le document d'information communal.

[35] Le projet européen eVEIL (janvier 2002 – janvier 2004) a été une collaboration entre l'Université de Paris-Sorbonne (Paris IV), l'Université de René Descartes (Paris 5), Thales recherche (société anonyme), Projexion Netsoft (société anonyme) et Xistos Développement (société anonyme). L'objectif du projet a été de développer des méthodes et des outils pour assister un utilisateur dans sa recherche d'information dans des textes.

Question :

Qu'est-ce que le MATE ?

Réponse :

MATE (Ministère de l'Aménagement du Territoire et de l'Environnement)

Question :

Est-ce que le MATE maintient un (ou plusieurs) site Web ?

Réponse :

Ministère de l'Aménagement du Territoire et de l'Environnement (MATE) http://www.environnement.gouv.fr

Cet exemple montre une possible application de la navigation textuelle, où certains constituants de la réponse (par exemple la sigle *MATE*) font partie d'une nouvelle question.

Rappelons (*cf.* chapitre 3) qu'une approche systématisée à la navigation textuelle nécessite de quatre éléments :

- une représentation du texte ;

- la possibilité de pouvoir isoler les connaissances visuelles et navigationnelles ;

- un agent (une personne, une équipe d'experts, un système, etc.) capable d'encoder ces connaissances ;

- un système qui interprète ces connaissances.

Le système interpréteur sera chargé de lire, d'une part, les textes et, d'autre part, les connaissances, et de construire les vues décrites, comportant leurs opérations de visualisation et de navigation. Ce point sera développé au chapitre 12. Quant aux agents encodeurs des connaissances, ils seront traités à travers des exemples d'applications au chapitre 13. Je développe dans la suite de ce chapitre les mécanismes de représentation des connaissances.

7.2. Les connaissances visuelles et navigationnelles

Afin de pouvoir isoler les connaissances visuelles et navigationnelles, il est nécessaire de définir un formalisme pour représenter celles-ci car, faute de formalisation, il serait impossi-

ble, pour les différents agents encodeurs, de partager ces connaissances et, pour un système interpréteur, de les interpréter correctement.

La navigation textuelle se fonde sur l'hypothèse selon laquelle il existe dans les textes des informations qui servent, à un utilisateur, de guide à la lecture (*cf.* chapitre 3). Les informations guidant cette lecture correspondent aux résultats d'un ensemble de traitements automatiques ou semi-automatiques[36] d'un texte, qui se traduisent en la réécriture de ce texte selon la représentation proposée au chapitre 6. Du point de vue de cette représentation, cela peut signifier plusieurs choses. Un traitement de découpage automatique en phrases [Mourad 2001] ou en propositions [Wonsever 2004], peut construire la hiérarchie dans le *corps* du texte, en utilisant des UT de base. Des traitements comme ceux effectués dans le projet RÉGAL [Ferret *et al.* 2001] [Couto *et al.* 2004], peuvent, d'une part, annoter différentes UT du *corps* et, d'autre part, créer de nouvelles UT dans la *tête* afin de représenter des structures telles que les cadres thématiques ou les cadres organisationnels (*cf.* chapitres 1 et 5).

Le fait de pouvoir afficher un texte de manières différentes, et que chaque manière (*vue du texte*) comporte des indications précises sur les différentes options d'affichage (*opérations de visualisation*) et sur les interactions que l'utilisateur peut effectuer (*opérations de navigation*) constitue l'épine dorsale de notre approche. De plus, une vue d'un texte ne montre pas nécessairement tous les constituants d'un texte ; il peut s'agir d'une vue partielle se focalisant sur certains aspects spécifiques ou phénomènes présents dans celui-ci. Cela constitue, en quelque sorte, la vue d'un filtrage du texte. La gestion de la coordination, faite de manière automatique dans ContextO ou RÉGAL, peut être définie par l'encodeur en exprimant des *opérations de coordination*. Nous présupposons que deux vues sont coordonnables si et seulement si elles sont des vues différentes d'un même texte.

Je présente par la suite les concepts fondamentaux de notre approche et la manière dont ceux-ci se relient.

7.2.1. Les vues graphiques d'un texte

Un texte encodé selon le formalisme proposé au chapitre 6, est susceptible d'avoir plusieurs représentations graphiques. Chaque type de représentation correspond à un type de vue textuelle possible, comportant ses paramètres spécifiques. Il peut exister des paramètres

[36] Par exemple, l'étiquetage d'un corpus de test, réalisé à l'aide des outils logiciels, mais toujours avec une intervention humaine afin de faire la validation.

communs à toutes les vues (par exemple la taille de la police à utiliser), mais chaque type de vue définit un ensemble de paramètres particuliers.

Comme la notion de texte est étroitement liée à celle d'information lexicale, il faut souligner qu'une vue de texte peut ne pas prendre en considération les chaînes lexicales des unités textuelles, comme c'est le cas, par exemple, d'une de type *arborescente* où le *contenu* affiché correspond aux annotations (*cf.* figure 7.1).

Afin de présenter une approche systématisée des différentes vues, je propose une classification selon leur *type* et leur *contenu*. Les types possibles sont : *linéaire*, *arborescente* et *graphe* tandis que les contenus possibles sont : les *chaînes lexicales* et les *annotations*. Il en résulte qu'il existe six combinaisons possibles. Cependant, une vue de type linéaire où le contenu correspond aux annotations n'est semble pas particulièrement utile, et elle ne sera pas prise en considération. Les combinaisons valides s'affichent ci-dessous.

	linéaire	arborescente	graphe
chaîne lexicale	oui	oui	oui
annotations	non	oui	oui

Tableau 7.1 – *Différentes vues selon leur type et contenu.*

Certes, d'autres types de vues différents à ceux présentés ici sont envisageables, comme les vues basées sur la technique « Focus + Context » (*cf.* chapitre 2) [Lamping *et* Rao 1996] [Dieberger *et* Russell 2002], par exemple ; ou d'autres plutôt *ad-hoc* comme la vue « docball » [Crestani *et al.* 2002], qui montre la structure d'une hiérarchique d'un document en utilisant plusieurs cercles concentriques sectionnés. Néanmoins, le choix des types linéaire, arborescent et graphe correspond à la représentation de texte proposée, et constitue, de mon point de vue, un bon point de départ, pouvant s'enrichir des propositions et des développements postérieurs.

Les différentes vues sont présentées par la suite.

7.2.1.1. Vue linéaire

La vue la plus simple d'un texte est la vue *linéaire*, où celui-ci est représenté comme une suite de chaînes de caractères, cette suite étant le résultat d'un parcours en profondeur dans la hiérarchie d'UT de base présentes dans le *corps* du texte. Notons que si une UT comporte un titre, celui-ci est affiché en premier, selon une chaîne calculée également à partir d'un parcours en profondeur dans les UT constituant le titre.

Étant donné la nature plate de la représentation, il n'est pas possible de visualiser les unités textuelles complexes, décrites dans le chapitre 6, pouvant être exprimées dans la *tête* du texte.

Pour cette vue, il est important d'indiquer les éléments textuels (types d'UT) constituant des *séparateurs* visuels (*i.e.* des éléments après lesquels il faut introduire de sauts de ligne dans l'affichage).

7.2.1.2. Vue arborescente

Un deuxième type de vue montre le contenu du texte, c'est-à-dire les UT de base exprimées dans le *corps* du texte, de manière arborescente, cette arborescence reflétant la hiérarchie d'UT existante.

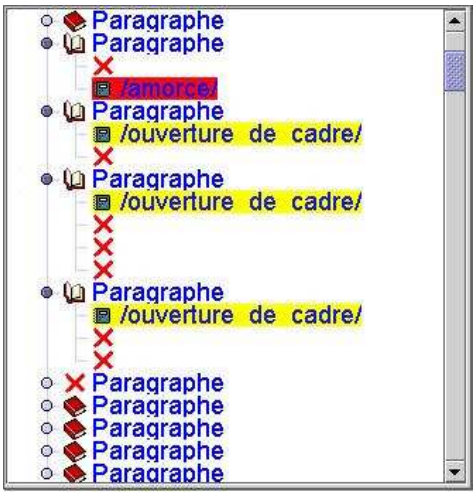

Figure 7.1 – *Exemple d'une vue de type arborescente.*

La nature hiérarchique de cette vue ne permet pas de représenter les unités textuelles complexes exprimées dans la *tête* du texte.

Selon le contenu, les libellés à afficher seront les chaînes lexicales des UT ou les annotations de celles-ci. La figure ci-dessous montre un exemple de vue de type arborescente où le contenu consiste aux annotations, extraite du projet RÉGAL (*cf.* chapitre 1). Notons que dans cette figure, toute UT comporte ou bien une annotation (dont la valeur s'affiche en utilisant un jeu de couleurs selon une carte de couleurs définie par l'utilisateur) ou bien aucune. En cas de plusieurs annotations, une UT pourrait s'étendre autant de lignes que d'annotations elle comporterait.

7.2.1.3. Vue graphe

Ce type de vue affiche toutes les UT d'un texte, en montrant les rapports entre elles. Ce rapport est déterminé à partir du rapport hiérarchique exprimé dans le *corps* du texte, et à partir des unités textuelles complexes exprimées dans la *tête* du texte. En conséquence, c'est cette vue qui montre la structuration globale du texte.

De même que pour la vue précédente, les libellés à afficher seront les chaînes lexicales des UT ou les annotations de celles-ci.

Notons, d'une part, que la vue de type arborescente est un cas particulier de cette vue. D'autre part, la vue graphe peut devenir très complexe à visualiser et la navigation s'avérer difficile[37].

7.2.1.4. Vues condensées

Une vue condensée montre de manière abstraite les rapports existants entre les UT concrètes d'un texte. Seuls le type d'UT et, éventuellement, le rang sont pris en considération dans une vue condensée. Ainsi, si une UT de type « Phrase » et numéro « 10 » est le parent d'une UT de type « SN » et numéro 42 (*i.e.* il existe deux nœuds reliés dans la représentation arborescente), la vue condensée créera un nœud pour une UT générique de type « Phrase » à laquelle reliera une UT générique fille de type « SN ».

Considérons une vue de type arborescente où le contenu correspond aux chaînes lexicales des UT, dont un exemple s'affiche à la figure 7.2.

[37] On peut trouver à l'adresse http://www.visualcomplexity.com/vc/index.cfm, un ensemble d'exemples de complexité de visualisation lorsque l'espace d'information est vaste. Il s'agit de représentations graphiques des réseaux de connaissances, et parmi ces exemples, on en trouve plusieurs correspondant à des vues de graphe en deux dimensions, similaires au type de vue proposé ici.

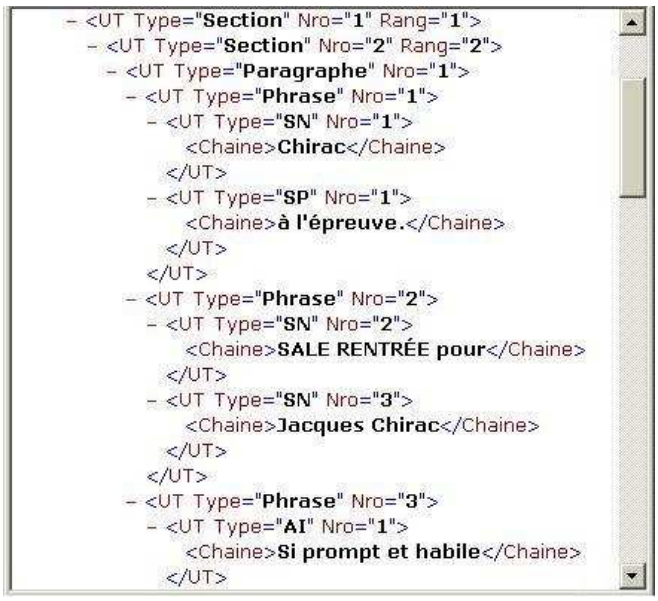

Figure 7.2 – *Exemple d'une vue de type arborescente.*

Une vue condensée de cette vue inférera les rapports entre les différentes UT et affichera ceux-ci de manière arborescente (*cf.* figure 7.3).

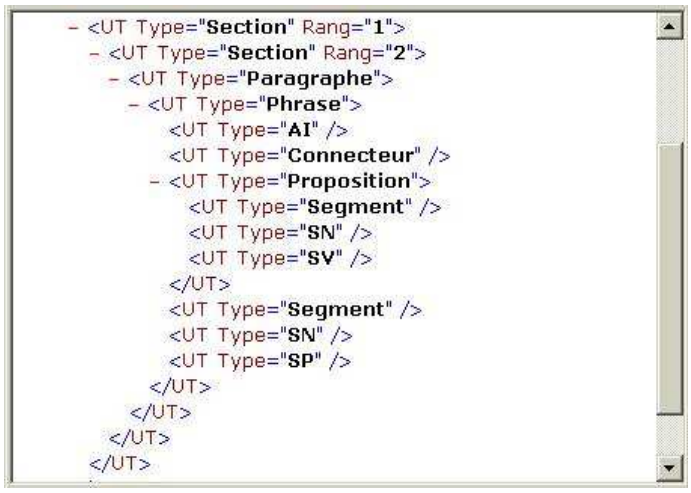

Figure 7.3 – *Exemple d'une vue condensée.*

A partir de la figure précédente, l'utilisateur comprend que dans le texte auquel la vue correspond, il existe, par exemple, au moins une instance d'une UT de type « Phrase », fille d'une UT de type « Paragraphe ». La répétition des UT de type « Segment », « SN » et « SV »

met en évidence que des instances concrètes de ces types d'UT occurrent dans le texte tant comme filles d'une UT de type « Proposition » que d'une UT de type « Phrase ».

Notons que, puisque la condensation est réalisée à partir des rapports provenant de la hiérarchie d'UT établie, une vue condensée n'a aucun sens si le type de vue est linéaire. En outre, ce type de vue est, d'une certaine manière, une vue créée à partir d'une autre vue sur laquelle une opération de condensation est appliquée. En conséquence, on pourrait considérer l' « état de condensation » (« oui » ou « non ») comme un paramètre d'une vue.

7.2.1.5. Vue de type « carte »

Dans le cadre de la navigation textuelle, la technique « *panning & zomming* » (*cf.* chapitre 2) peut s'utiliser pour positionner spatialement l'utilisateur dans un texte. Puisque pour un même texte plusieurs vues peuvent être créées, l'on peut envisager une vue particulière, nommée de type « carte », en vue d'y refléter les mêmes actions d'une autre vue. C'est-à-dire que si l'utilisateur se positionne dans une vue sur une UT, le positionnement est également réalisé sur la vue de type carte. De même, par exemple, pour des phénomènes de colorisation d'unités textuelles. La différence réside dans l'objectif de la vue de type carte, qui consiste à offrir à l'utilisateur une vision globale et manipulable, différemment d'autres vues (au sens des opérations de *zooming* et de *panning*), d'un texte ou d'une vue déterminée. Une vue de type carte pourrait se diminuer de sorte telle qu'elle affiche tout le contenu dans un espace rentrant dans le cadre d'un écran. Si le positionnement sur une UT est visuellement indiqué (par exemple en utilisant une mise en forme spécifique pour l'UT sélectionnée), dans cette vue ce positionnement donnera à l'utilisateur le placement de celle-ci par rapport aux autres UT. La colorisation de différentes UT permettra à l'utilisateur, dans cette vue, d'avoir une idée assez précise de la distribution des informations, à condition que les couleurs utilisées correspondent à une sémantique décodable par l'utilisateur.

Il faut souligner qu'il ne s'agit pas, au sens strict, d'un nouveau type de vue mais d'une manière d'afficher une vue déterminée, ayant son type et son contenu définis. Il en résulte que pour indiquer si une vue est de type carte ou non, on pourra utiliser un paramètre booléen.

7.2.2. Modules de connaissances et descriptions de vue

Les éléments constitutifs d'une vue sont spécifiés dans une *description de vue*. Plusieurs descriptions de vue peuvent être rassemblées dans une entité cohérente d'après l'encodeur des connaissances, nommée *module de connaissances*.

Nous pouvons concevoir la création d'une vue comme l'application d'une description de vue à un texte déterminé (*cf.* figure 7.4).

Figure 7.4 – *Création d'une vue d'un texte.*

Par analogie, l'application d'un module de connaissances (*i.e.* un ensemble de description de vues) à un texte implique la création d'un ensemble de vues, comme le montre la figure ci-dessous. En conséquence, toute vue est liée à un texte, à une description de vue et, indirectement, à un module de connaissances.

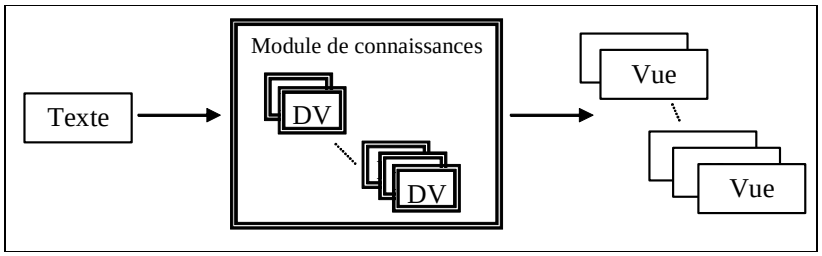

Figure 7.5 – *Application d'un module à un texte.*

Une description de vue est identifiée dans le module par son nom. Afin de la définir, l'encodeur doit indiquer :

- le type de vue selon les types de vues disponibles : linéaire, arborescente et graphe ;

- le contenu de la vue selon les contenus disponibles : chaînes lexicales et annotations ;

- ses paramètres, selon le type de représentation ;

- ses contraintes de création (*i.e.* des conditions d'appartenance à la vue, à vérifier par les unités textuelles du texte) ;

- un ensemble d'opérations de visualisation ;

- un ensemble d'opérations de navigation ;

- un ensemble d'opérations de coordination.

Le fait de pouvoir créer des vues partielles d'un texte introduit le besoin de contraintes. Il s'agit de conditions sur les UT, et la manière dont elles peuvent s'exprimer est présentée par la suite.

7.2.2.1. Notion de Condition sur les UT

Afin de filtrer les UT qui seront inclues dans une vue, un langage de conditions est proposé. Le but de ce langage est de permettre l'expression des conditions à vérifier pour une UT de base et, en conséquence, il porte sur les différents éléments de ces unités.

7.2.2.1.1. Conditions sur les attributs des UT de base

Pour les UT de base, on peut exprimer des conditions sur le *type*, le *numéro* et le *rang*. Ces conditions portent sur la *valeur*, pouvant exprimer des conditions d'égalité, d'inégalité, et d'ordre (inférieur et supérieur).

Par exemple :

- des UT de base de type « Section » dont le rang n'est pas égal à « 1 » ;

- des UT de base dont le numéro est supérieur à « 10 ».

7.2.2.1.2. Conditions sur les annotations

En ce qui concerne les annotations, on peut exprimer des conditions d'*existence sur une annotation* d'un nom déterminé, l'*existence d'une valeur* (*i.e.* une annotation quelconque ayant une valeur spécifiée) et l'*existence d'un couple* nom d'annotation - valeur.

Par exemple :

- des UT comportant une annotation de nom « Référent Discursif » ;

- des UT comportant une annotation dont la valeur est « Guerre en Irak » ;

- des UT comportant une annotation de nom « Relation RST » dont la valeur est « Élaboration ».

De même, des conditions de non-existence sur les annotations peuvent être exprimées.

7.2.2.1.3. Conditions sur la hiérarchie

En ce qui concerne la hiérarchie d'UT, plusieurs contraintes sont possibles. On peut exprimer qu'une UT comporte ou non un titre. De même, on peut exprimer qu'une UT comporte ou non une chaîne lexicale, un parent ou des fils. Quant à la chaîne lexicale d'une UT, on peut exprimer des contraintes sur la valeur de la chaîne complète ou sur un *préfixe*, un *suffixe* ou une *sous-chaîne*.

Ces conditions sont complémentaires aux conditions présentées plus haut. Ainsi, on pourrait exprimer, par exemple, qu'une UT comporte un titre où il existe une UT d'un type ou de rang déterminé. Les éléments structuraux participant aux conditions sont : le *parent*, les *fils*, les *ascendants*, les *descendants*, les *frères* et le *titre*. Il va de soi que le *parent* appartient aux *ascendants* et les *fils* sont inclus dans les *descendants*. Notons que ce type de conditions permet, en quelque sorte, de « parcourir » la hiérarchie.

Exemples de conditions sur la hiérarchie :

- des UT comportant une chaîne lexicale ;

- des UT comportant une suite non vide de fils ;

- des UT faisant partie d'un titre ;

- des UT comportant un frère de type « Section » ayant une annotation de nom « Thème » et valeur « Musique religieuse » ;

- des UT faisant partie d'un titre d'une UT de type « Section » ayant une annotation de nom « Thème » et valeur « Musique religieuse » ;

- des UT de type « SN » comportant une annotation de nom « Référent discursif », telles qu'il existe dans les ascendants une UT de type « Paragraphe » qui comporte une annotation de nom « Étiquette Sémantique » et valeur « Conclusion ».[38]

7.2.2.1.4. Conditions sur les UT complexes

Trois types de conditions peuvent s'envisager dans le cas des UT complexes. Premièrement, puisque celles-ci sont composées d'UT de base, la notion d'appartenance pourrait se tester. Ainsi, on pourrait exprimer qu'une UT de base appartienne à une UT complexe (par

[38] Cette condition peut se comprendre de la manière suivante : des syntagmes nominaux étant des référents discursifs, qui sont inclus dans de paragraphes conclusifs.

exemple à une séquence) ou qu'une UT complexe contienne une UT de base déterminée. Deuxièmement, puisque les UT complexes peuvent comporter des annotations, des contraintes sur celle-ci peuvent s'exprimer. Enfin, des contraintes reliant différentes UT complexes sont aussi possibles.

Néanmoins, pour des raisons de temps et d'un manque d'exemples réels d'utilisation, ce point n'a pas été abordé en profondeur dans le cadre de cette thèse. Des applications en cours (*cf.* chapitre 13) commencent à montrer l'intérêt d'exprimer des conditions sur les UT complexes.

7.2.2.1.5. Composition des conditions

Les conditions peuvent se combiner en utilisant les opérateurs classiques *OU*, *ET* et *NON*, de la logique.

7.2.3. Les opérations de visualisation

Les connaissances de visualisation modélisées sont considérées ou bien comme des opérations de transformation qui s'appliquent sur une UT ou bien comme des opérations d'affichage d'information relative à une UT. Elles s'inspirent des propositions de [Couto 2002] et peuvent se diviser en deux types principaux : des opérations de *mise en relief* et des opérations d'*aide contextuelle*.

7.2.3.1. Opérations de mise en relief

Ces opérations peuvent, d'une part, changer l'apparence sur l'écran de la chaîne textuelle (colorisation, taille de la police, etc.), et, d'autre part, permettent de modifier la chaîne textuelle, en ajoutant des informations lexicales. Une opération basique de mise en relief est définie par :

- l'élément textuel à mettre en relief ;

- une liste d'opérations de transformation.

L'UT à mettre en relief est indiquée en exprimant la condition qu'elle doit vérifier. Les opérations de transformation s'appliquent, en suivant l'ordre de le la liste, sur l'UT vérifiant la condition. Les différentes opérations possibles sont :

- *couleurPremierPlan*(couleur) et *couleurArrièrePlan*(couleur)

Ces opérations permettent de modifier la couleur en premier plan et arrière plan dont la chaîne lexicale est affichée.

- *taillePolice*(entier)

Cette opération définit la taille de la police dont la chaîne lexicale est affichée.

- *ajouter*(placement, chaîne)

L'opération d'ajout permet d'ajouter une chaîne lexicale à la chaîne lexicale d'une UT. Le placement peut être *préfixe* ou *suffixe*. Cet effet d'ajout est intéressant pour des UT appartenant à une structure discursive complexe comme par exemple les cadres de discours [Charolles 1997].

La cohérence des opérations de transformation décrites doit être respectée. Cela signifie que l'encodeur ne doit pas définir pour une même UT des opérations contradictoires. Par exemple, une opération de mise en relief basique contenant dans son ensemble d'opérations de transformation deux opérations *taillePolice* comportant des valeurs différentes. Du point de vue du langage, cela implique un contrôle sémantique sur les opérations de mise en relief (et, en conséquence, sur les modules de connaissances) qui est à la charge du système interpréteur.

Une liste d'opérations basiques de mise en relief peut se regrouper dans une opération complexe de mise en relief. Dans ce cas, les remarques sur la cohérence des opérations de transformation sont également valables.

Notons que les effets concrets de ces opérations sont fortement liés au type de vue, et ils seront bien différents s'il s'agit d'une vue de type linéaire, par exemple, ou s'il s'agit d'une vue arborescente. Dans le premier cas, les UT nœuds de la hiérarchie exprimée dans le *corps* du texte n'ont pas une représentation concrète dans la vue. En conséquence, une opération de colorisation sur une UT nœud, par exemple, affecte à toutes les chaînes lexicales des UT descendants qui sont des feuilles dans la hiérarchie, tandis que dans une vue arborescente, cette opérations affecte seulement au nœud correspondant puisqu'il à une représentation visuelle.

Si l'on considère que la mise en forme des UT d'une vue fait partie de l'état d'une vue, l'application d'une mise en relief peut changer cet état et c'est la sémantique du langage qui définira ce comportement.

7.2.3.2. Opérations d'aide contextuelle

Ce sont des opérations utilisant des techniques de type *infobulle* pour transmettre à l'utilisateur des informations sur les UT du texte. Une opération d'aide contextuelle est définie par :

- le type d'aide contextuelle ;

- l'élément textuel qui déclenche l'opération ;

- l'aide à afficher.

Nous avons défini deux types d'aide contextuelle : l'*affichage d'infobulles* et l'*affichage dynamique de vues*. L'UT déclenchant l'opération est signalée en exprimant la condition qu'elle doit vérifier. L'aide à afficher dépend du type d'aide contextuelle. Dans le cas d'affichage d'infobulles, il s'agit d'une *chaîne lexicale*. Dans le cas d'affichage dynamique de vues, il s'agit d'une *description de vue* ou du *nom* d'une description de vue existante dans le module.

Notons que l'aide contextuelle s'active au survol de la souris. Cela implique qu'une vue affichée dynamiquement sera « active » pendant que l'utilisateur positionne le curseur sur l'UT que l'a déclenchée. Le système interpréteur devra fournir des mécanismes spécifiques s'il souhaite offrir à l'utilisateur la possibilité d'interagir avec la nouvelle vue. De plus, s'il permet cette interaction, il devra considérer la possibilité d'avoir plusieurs déclenchements de vues.

7.2.4. Les opérations de navigation

La navigation est conceptualisée comme une opération reliant une UT *source* avec une UT *cible*. La manière dont ces deux UT sont reliées est fonction de quatre paramètres :

- la condition à vérifier par l'UT source ;

- la condition à vérifier par l'UT cible ;

- le type d'opération de navigation ;

- le rapport existant entre l'UT source et l'UT cible.

Une opération de navigation est définie comme une opération qui cherche l'UT cible à partir de l'UT source, en vérifiant les différentes conditions et en suivant l'orientation relative au type d'opération.

La source est définie en utilisant une condition sur les UT. Implicitement, une opération de navigation est *disponible* pour une UT déterminée si celle-ci vérifie la condition exprimée par la source.

La cible est déterminée à partir de deux paramètres: une condition à vérifier pour l'UT cible et le type d'opération de navigation. Une fois déterminée la source, plusieurs UT peuvent vérifier la condition de la cible, et c'est le type d'opération qui indique laquelle choisir d'entre elles. Chaque opération est donc typée avec une valeur qui appartient à l'ensemble {*premier, dernier, suivant[i], précédent[i]*}, étant *i* un nombre entier positif. Ces valeurs spécifient d'une part l'orientation, c'est-à-dire dans quel sens (avant ou après l'UT source) doit être effectué la recherche de l'UT cible, et d'autre part le référentiel, absolu (*premier, dernier),* ou relatif (*suivant[i], précédent[i]*), par rapport à la source. Dans le cas d'un référencement relatif, l'index *i* permet de spécifier le rang de la cible recherchée. Par exemple, le type « Suivant[3] » s'interprète comme la recherche, dans les UT vérifiant les conditions spécifiées pour la cible (*i.e.* les cibles potentielles), de la troisième unité textuelle située après l'unité textuelle source.

En ce qui concerne le rapport existant entre la source et la cible, il peut s'exprimer en utilisant des conditions sur deux UT spécifiques libellées *Source* et *Cible*, représentant la source et la cible respectivement. Selon la nature des conditions, cette approche sera ou non pertinente. Pour des conditions très simples, d'autres approches sont possibles. Par exemple, si l'on veut seulement tester l'égalité des valeurs d'annotations de nom identique, il suffira énumérer dans une liste les noms des annotations à vérifier. Ces aspects seront examinés de prés dans la proposition de la syntaxe du langage.

Du point de vue de la navigation textuelle, une vue a toujours une *UT active*, c'est-à-dire une UT sur laquelle l'utilisateur est positionné. Cela implique, d'une part, que la création d'une vue doit définir la première UT active de celle-ci. D'autre part, si l'exécution d'une opération de navigation réussit (*i.e.* une cible est déterminée pour une source donnée), l'impact sur la vue se concrétise par le changement de l'UT active, sous entendu que l'UT cible est différente de l'UT source. Si l'on considère que l'UT active fait partie de l'état d'une vue, l'exécution d'une opération de navigation peut changer cet état et c'est la sémantique du langage qui définira ce comportement. Notons que les exécutions successives des opérations de navigation produisent un historique de navigation de l'utilisateur. Cet historique pourrait être utilisé dans les conditions des opérations de navigation en faisant l'hypothèse que l'on peut inférer la prochaine navigation de l'utilisateur à partir des navigations précédentes.

7.2.5. Les opérations de coordination

Puisqu'un même texte peut avoir différentes vues, il est important de pouvoir coordonner celles-ci, c'est-à-dire que le fait de changer l'UT active d'une vue (que ce soit par la sélection explicite d'une UT ou par l'exécution d'une opération de navigation), implique le changement de l'UT active d'autres vues. Dans la plate-forme ContextO, cette coordination est réalisée automatiquement, étant sous entendu que les vues ont été créées à partir du même texte. Supposons que nous ayons une vue de type linéaire et une autre de type carte sur la vue linéaire. Si un changement de l'UT active dans la première vue se répercute automatiquement sur la deuxième, nous avons toujours, grâce à la deuxième vue, un repère visuel du positionnement global de l'UT. Néanmoins, si nous avons une vue partielle d'un texte (*i.e.* une vue contenant seulement un sous ensemble de toutes les UT du texte), il peut être intéressant de ne pas faire automatiquement la coordination avec une vue totale du même texte. L'utilisateur pourrait souhaiter parcourir (naviguer) la vue partielle (par exemple la vue d'un extrait) et, souhaitant mettre en contexte une UT dans la vue totale, demander manuellement la coordination. Notons que si la coordination est automatique, le système interpréteur est chargé de la réaliser, tandis que si elle est manuelle, le système doit fournir des mécanismes pour sélectionner les opérations de coordination à exécuter.

Comme une vue peut se coordonner avec plusieurs vues, l'encodeur peut définir alors un ensemble d'opérations de coordination pour une description de vue. Chaque opération de coordination doit indiquer :

- le nom de la vue cible avec laquelle il faut coordonner ;

- le mode de coordination : manuel ou automatique ;

Il faut souligner que l'exécution d'une opération de coordination peut échouer si l'UT de la vue source n'est pas présente dans la vue cible. Le système interpréteur ne devrait pas changer l'UT active dans la vue cible, faute d'une UT avec laquelle coordonner. Si la coordination est censée être automatique, ce problème engendrera alors un phénomène de non-coordination temporaire. Nous étudierons ce comportement au moment de donner la sémantique des opérations de coordination.

7.3. Synthèse

J'ai présenté dans ce chapitre différentes connaissances à modéliser dans le cadre de la navigation textuelle : la notion de vue d'un texte et les opérations de navigation, de visualisation et de coordination. De mon point de vue, les connaissances décrites dans le cadre de cette thèse constituent un bon point de départ pour développer la navigation textuelle, mais elles ne doivent pas être considérées comme exhaustives ou définitives. Dans le chapitre suivant je présente le langage Sextant, qui modélise les connaissances ici présentées.

Chapitre 8

Sextant, un langage de modélisation des connaissances de visualisation et de navigation

8.1. Introduction

Après avoir développé les différentes connaissances à modéliser, je présente dans ce chapitre la syntaxe qui définit *Sextant*[39], un langage de modélisation des connaissances de visualisation et de navigation. Afin de rendre cette présentation la plus claire possible, celle-ci est présentée en deux parties : d'abord, une vision globale des connaissances ; ensuite les opérations de visualisation, les opérations de navigation et les opérations de coordination. La notion de condition est fondamentale pour le langage Sextant, méritant une sous section spécifique préalable aux sections correspondant aux différents types d'opération.

Le langage proposé a pour finalité d'offrir des fonctionnalités à la fois suffisamment génériques tout en proposant une sémantique qui se focalise sur l'essentiel du processus de visualisation et de navigation, à l'inverse de langages de transformation ou de programmation comme, par exemple, XSLT ou XPATH. Notre langage est donc de type déclaratif et s'appuie sur des opérations prédéfinies. Bien que la nomenclature utilisée suggère généralement le sens des opérations, la sémantique complète du langage est définie au chapitre suivant.

8.2. Système de notation

Pour décrire la syntaxe, des grammaires hors-contexte sont utilisées, en respectant la restriction suivante : seul les éléments dont la police est ombrée correspondent aux éléments terminaux de la grammaire (*i.e.* ils correspondent aux expressions que l'encodeur écrira ef-

[39] Par analogie avec les navigateurs du XVIII siècle qui ont parcouru le monde en s'orientant sur les mers avec un sextant.

fectivement). Une variable (au sens des grammaires) nommée « valeur » représentant une valeur quelconque sera utilisée sans être définie en termes d'éléments terminaux.

Dans notre notation, l'*ensemble* est indiqué comme une suite d'éléments séparés par des virgules, entourée par des accolades d'ouverture et de fermeture. Par exemple, l'expression suivante :

$$\{e_1, e_2, e_3, e_4, e_5\}$$

dénote l'ensemble formé par les éléments e_1, e_2, e_3, e_4 et e_5.

Un *n-uplet* est dénoté comme une suite d'éléments séparés par des virgules, entourée par des parenthèses. Par exemple, l'expression suivante :

$$(x, y, z)$$

dénote le triplet formé par les éléments x, y et z à la première, deuxième et troisième position, relativement.

8.3. Modules et descriptions de vue

Une première approche de haut niveau à la syntaxe du langage est montrée dans le tableau 8.1. Un module de connaissances est un élément qui comporte un nom et un ensemble de descriptions de vue.

module	→	Module (nomModule , { ensembleDV })
nomModule	→	valeur
ensembleDV	→	descriptionVue | descriptionVue , ensembleDV
descriptionVue	→	DV (typeVue , contenuVue, nomVue ,
		première UT , { paramètres } , conditionFiltre ,
		{ opérationsVisualisation } ,
		{ opérationsNavigation } ,

		{ opérationsCoordination })
typeVue	→	linéaire | arborescente | graphe
contenuVue	→	chaînes lexicales | annotations
nomVue	→	valeur
premièreUT	→	condition
paramètres	→	(nomParamètre , valeurParamètre) , paramètres
		| ε
nomParamètre	→	valeur
valeurParamètre	→	valeur
conditionFiltre	→	condition | ε
opérationsVisualisation	→	opérationVisualisation , opérationsVisualisation | ε
opérationsNavigation	→	opérationNavigation , opérationsNavigation | ε
opérationsCoordination	→	opérationCoordination , opérationsCoordination | ε

Tableau 8.1 – *Grammaire de haut niveau du langage des connaissances Sextant.*

Chaque description de vue est un élément comportant :

- un type de vue selon les types de vues possibles ;

- le contenu selon les contenus possibles ;

- un nom de vue, unique dans le module ;

- la description de la première UT active de la vue, en termes de la condition à vérifier par cette UT [40] ;

- un ensemble de paramètres (couples nom – valeur) [41];

- une condition d'appartenance à la vue, à vérifier par les unités textuelles du texte ;

[40] Il faut souligner qu'une condition peut être formée par plusieurs conditions combinées en utilisant les opérateurs logiques ET, OU et NON.

[41] Pour une vue condensée ou une vue de type carte, des paramètres booléens sont utilisés. Ces deux types de vues peuvent se combiner. L'absence de paramètres est considérée comme une indication négative.

- un ensemble d'opérations de visualisation ;

- un ensemble d'opérations de navigation ;

- un ensemble d'opérations de coordination.

Dans le tableau 8.1, il manque les définitions des éléments suivants : *condition*, *OpVis*, *OpNav* et *OpCoord*, qui sont présentés par la suite.

8.4. Le langage de conditions

Une partie importante du langage Sextant est le langage de conditions, qui est utilisé par plusieurs constructions syntaxiques. Par exemple, on utilisera une condition pour exprimer des contraintes d'appartenance d'une UT à une vue. De même, pour indiquer les UT sur lesquelles une mise en relief s'applique, on utilisera une condition. À cause de la nature déclarative du langage, les conditions sont donc utilisées pour « signaler » des UT. Le cas le plus représentatif correspond à l'utilisation d'une condition pour la cible et d'une autre pour la source dans la description d'une opération de navigation.

Ce langage est présenté ci-après de manière progressive, en exemplifiant les possibilités décrites à la section 8.2.1.

8.4.1. Conditions simples : opérateur UT

Commençons par les *conditions simples*, qui sont celles portant sur les attributs et sur les annotations des UT. Pour ce type de conditions, nous utiliserons une notation proche de la notion de patron. On définit un opérateur *UT* comportant cinq opérandes qui correspondent aux propriétés suivantes d'une UT : le *Type*, le *Nro*, le *Rang*, les *Annotations* et la chaîne lexicale. Avec les trois premiers opérandes on dénote des contraintes d'*égalité*, d'*inégalité*, d'*ordre* (inférieur et supérieur), de *préfixe*, de *suffixe* et de *sous-chaîne* par rapport à des valeurs. De même pour le cinquième opérande. Le quatrième opérande est utilisé pour indiquer l'existence ou non-existence d'annotations, que ce soit un nom d'annotation, une valeur ou une paire nom d'annotation – valeur.

Par exemple, pour exprimer la condition suivante : des UT de type « Section » dont le rang n'est pas égal à « 1 », on écrit :

UT(Type = Section, *, Rang ≠ 1, *, *)

Ici, le symbole étoile « * » signifie qu'il n'y a aucune contrainte définie pour la propriété correspondante de l'UT. C'est-à-dire, dans l'exemple, que l'UT peut avoir un numéro et un ensemble d'annotations quelconques. La répétition des noms des attributs de l'UT (Type, Nro, Rang), inutile dans l'exemple précédent puisque les positions sont fixes et l'on peut les en déduire, sert, d'une part, à améliorer la clarté de la notation proposée, et d'autre part, pour indiquer l'ordre des arguments des opérateurs représentant des relations non symétriques. Constatons que pour les trois premières propriétés (et pour la cinquième aussi), l'opérateur de comparaison est dénoté de manière infixée. Ainsi, pour exprimer la condition suivante : des UT dont le type commence par « P » et le numéro soit supérieur à « 10 »[42], on doit écrire :

UT(P *estPréfixe* Type, Nro > 10, *, *,*)

L'expression « P estPréfixe Type » indique que la chaîne lexicale « P » doit être un préfixe du type de l'UT en question, ce qui dénote des UT de type paragraphe, phrase, proposition, préposition, etc. Les trois opérateurs possibles sont : *estPréfixe*, *estSuffixe* et *estSousChaîne*, et ils sont utilisés de manière infixe. Par exemple, pour dénoter une UT dont la chaîne lexicale contient le mot « président », on écrit :

UT(*, *, *, *, président *estSousChaîne* ChaîneLexicale)

En ce qui concerne les annotations, le quatrième opérande consiste en un ensemble de contraintes d'existence ou de non-existence. L'ensemble est indiqué comme une suite d'éléments séparés par des virgules, entourés tous par des accolades d'ouverture et de fermeture. Chaque élément est un couple où la première position contient le nom de l'annotation et la deuxième contient la valeur, l'encodeur pouvant utiliser le symbole étoile

[42] Notons que la comparaison avec le numéro 10 présuppose une comparaison numérique, qui est différente d'une comparaison lexicographique, d'après laquelle, par exemple, « 10 » est inférieur à « 5 ».

pour indiquer qu'il ne signale pas de contrainte sur le nom ou sur la valeur de l'annotation. Le couple est dénoté en utilisant des parenthèses, et il est préfixé par le type de condition : « ∃ » pour l'existence ou « ¬∃ » pour la non-existence. Il s'agit par défaut d'une condition d'existence et nous pouvons en conséquence omettre le type de condition dans ce cas.

Par exemple, pour exprimer la condition suivante : des UT comportant une annotation de nom « Référent Discursif », on écrit :

UT(*, *, *, {∃ (Référent Discursif, *)}, *)

Si nous sommes intéressés seulement à l'existence d'une valeur, comme dans la condition suivante : des UT de type Phrase comportant une annotation dont la valeur est « Guerre en Irak », on doit écrire :

UT(Type = Phrase, *, *, {(*, Guerre en Irak)}, *)

Plusieurs contraintes peuvent s'exprimer pour les annotations, comme le montre l'exemple pour la condition : des UT comportant une annotation de nom « Relation RST » dont la valeur est « Élaboration » et sans annotations de nom « Thème » :

UT(*, *, *, {(Relation RST, Élaboration), ¬∃ (Thème, *)},

Il faut souligner que l'encodeur est censé écrire des contraintes cohérentes. C'est-à-dire, par exemple, qu'il ne devrait pas décrire une condition sur une UT qui comporte une annotation de nom *X* et qui ne comporte pas une annotation de nom *X*. Du point de vue du langage, cela implique un contrôle sémantique sur les conditions et qui est à la charge du système interpréteur.

Sous jacent aux exemples ci-dessus, il y a la notion d'évaluation d'une condition. Pour évaluer une condition, il est nécessaire de connaître l'UT pour laquelle cette condition est évaluée. Le résultat de l'évaluation revient à décider si une UT déterminée vérifie ou non la

condition. Nous pouvons considérer cette évaluation comme l'application d'une fonction recevant deux paramètres : la condition à évaluer et l'UT sur laquelle évaluer celle-ci.

Pour l'instant nous laisserons de côté l'évaluation d'une condition car c'est la sémantique du langage qui définira le comportement de la fonction d'évaluation.

8.4.2. Conditions d'existence sur les éléments des UT

Pour les conditions qui vérifient si une UT comporte ou non un élément déterminé, un opérateur sans arguments est défini pour chaque élément (*cf.* tableau ci-après).

existeAnnotations *: teste si l'ensemble d'annotations d'une UT n'est pas vide ;*
existeChaîneLexicale[43] *: teste si la chaîne lexicale d'une UT est définie ;*
existeTitre *: teste si le titre d'une UT (*i.e. *la hiérarchie d'UT formant le titre) n'est pas vide ;*
existeParent *: teste si une UT a une UT parent ;*
existeFils *: teste si la suite d'UT filles d'une UT n'est pas vide.*

Tableau 8.2 – *Opérateurs d'existence sur les éléments des UT.*

Ces opérateurs peuvent se combiner avec l'opérateur de négation *NON* pour obtenir les conditions de non-existence. Ainsi, la condition « NON existeAnnotations » dénote les UT dont l'ensemble d'annotations est vide. Il faut souligner que si une UT vérifie la condition *existeFils*, alors cette UT est un nœud dans la hiérarchie d'UT définie dans le corps du texte, tandis que si elle ne la vérifie pas, il s'agit d'une feuille dans cette hiérarchie.

8.4.3. Conditions sur la hiérarchie

Pour les conditions où se joue le rapport entre les UT dans la hiérarchie, des opérateurs unaires spécifiques sont définis. Ces opérateurs prennent comme argument une condition

[43] Je voudrais remarquer la différence entre une UT qui ne contient pas de chaîne lexicale et une UT qui contient une chaîne lexicale vide. Dans le premier cas, l'élément Chaîne Lexicale, d'après la représentation de texte définie au chapitre 6, n'est pas défini pour l'UT, tandis que dans le deuxième cas, cet élément est défini mais il comporte une valeur de chaîne vide de caractères.

simple. Ainsi, pour exprimer la condition suivante : des UT parents des UT de type Phrase, on écrit :

estParent(UT(Type = Phrase, *, *, *, *))

Il convient aussi de pouvoir tester des conditions concernant le titre d'une UT. La première possibilité consiste à tester si une UT contient, dans les UT du titre, une UT avec certaines caractéristiques définies par une condition simple. Pour ce faire, on définit l'opérateur *contientDansTitre*. Une autre possibilité consiste à tester si une UT appartient au titre d'une UT vérifiant une condition simple, ce que l'on peut faire en utilisant l'opérateur *estDansTitreDe*.

Le tableau ci-dessous montre les opérateurs définis pour tester des conditions sur le rapport hiérarchique des UT.

estParent *: teste si une UT est le parent dans la hiérarchie d'UT d'une UT décrite en utilisant une condition simple ;*
estFils *: teste si une UT est le fils dans la hiérarchie d'UT d'une UT décrite en utilisant une condition simple ;*
estFrère *: teste si une UT est le frère dans la hiérarchie d'UT d'une UT décrite en utilisant une condition simple ;*
estAscendant *: teste si une UT est l'ascendant dans la hiérarchie d'UT d'une UT décrite en utilisant une condition simple ;*
estDescendant *: teste si une UT est le descendant dans la hiérarchie d'UT d'une UT décrite en utilisant une condition simple ;*
contientDansTitre *: teste si une UT contient dans les UT du titre une UT décrite en utilisant une condition simple ;*
estDansTitreDe *: teste si une UT appartient aux UT du titre d'une UT décrite en utilisant une condition simple ;*

Tableau 8.3 – *Opérateurs portant sur le rapport hiérarchique des UT.*

8.4.4. Exemples des conditions

Voici quelques exemples de conditions formulés en langage naturel et traduits au langage de conditions :

Exemple 1 : des UT comportant un frère de type « Section » ayant une annotation de nom « Thème » et valeur « Musique religieuse »

estFrère(UT(Type = Section, *, *, {(Thème, Musique religieuse)},*))

Exemple 2 : des UT faisant partie d'un titre d'une UT de type « Section » ayant une annotation de nom « Thème » et valeur « Musique religieuse » ;

estDansTitre(UT(Type = Section, *, *, {(Thème, Musique religieuse)},*))

Exemple 3 : des UT de type « SN » comportant une annotation de nom « Référent discursif », tel qu'il existe dans les ascendants une UT de type « Paragraphe » qui ne comporte pas une annotation de nom « Étiquette Sémantique » et valeur « Conclusion »

UT(Type = SN, *,*,{(Référent discursif, *)},*) ET estDescendant(UT(Type = Paragraphe,*,*,{ ¬∃(Étiquette Sémantique, Conclusion)},*))

Notons que tester l'existence d'une UT dans les ascendants correspond à tester que l'UT pour laquelle la condition doit vérifier soit une UT descendante de cette UT ascendante. La même logique s'applique aux opérateurs *estFils* et *estParent*, et aux opérateurs *contientDansTitre* et *estDansTitreDe*.

8.4.5. Syntaxe du langage des conditions

La syntaxe du langage des conditions est décrite par la grammaire ci-dessous.

condition	→	conditionSimple | conditionExiste | conditionHiérarchie | NON condition | condition ET condition | condition OU condition | (condition)
conditionExiste	→	existeAnnotations | existeChaîneLexicale | existeTitre | existeParent | existeFils
conditionHiérarchie	→	opérationCondHiérarchie (conditionSimple)
opérationCondHiérarchie	→	estParent | estFils | estFrère | estAscendant | estDescendant | contientDansTitre | estDansTitreDe
conditionSimple	→	UT (contrainteType , contrainteNro , contrainte-Rang , contraintesAnnotations , contrainteChaîne)
contrainteType	→	Type opérateur valeur | valeur opérateur Type | *
contrainteNro	→	Nro opérateur valeur | valeur opérateur Nro | *
contrainteRang	→	Rang opérateur valeur | valeur opérateur Rang | *
contrainteChaîne	→	ChaîneLexicale opérateur valeur | valeur opérateur ChaîneLexicale | *
opérateur	→	= | ≠ | < | > | ≤ | ≥ | estPréfixe | estSuffixe | estSousChaîne
contraintesAnnotations	→	{ listeContrainteAnnotation } | *
listeContrainteAnnotation	→	contrainteAnnotation |

		contrainteAnnotation listeContrainteAnnotation
contrainteAnnotation	→	valeurExistence (valeurNom , valeurAnnotation)
valeurExistence	→	∃ | ¬∃ | ε
valeurNom	→	valeur | *
valeurAnnotation	→	valeur | *

Tableau 8.4– *Grammaire du langage de conditions.*

8.5. Opérations de visualisation

Nous considérons ici la syntaxe des opérations de visualisation, qui sont divisées entre des opérations de *mise en relief* (MER) et des opérations d'*aide contextuelle*. De la même manière, les opérations de mise en relief sont divisées en opérations *basiques* et *complexes*, étant constituées ces dernières d'une liste d'opérations basiques. Le tableau ci-dessous montre la grammaire correspondant aux opérations de visualisation.

opérationsVisualisation	→	opérationVisualisation , opérationsVisualisation | ε
opérationVisualisation	→	miseEnRelief | aideContextuelle
miseEnRelief	→	merSimple | merComplexe
merComplexe	→	MER_C (nomMerComplexe , listeMerSimple)
nomMerComplexe	→	valeur
listeMerSimple	→	merSimple | merSimple , listeMerSimple
merSimple	→	MER_S (nomMerSimple , condition , { listeOpérationsTransformation })
nomMerSimple	→	valeur
listeOpérationsTransformation	→	opérationTransformation |

		opérationTransformation ,
		listeOpérationsTransformation
opérationTransformation	→	couleurPremierPlan (valeur) |
		couleurArrièrePlan (valeur) |
		taillePolice (valeur) |
		ajouter (placement , valeur)
placement	→	préfixe | suffixe
aideContextuelle	→	AideContextuelle (typeAideContextuelle ,
		condition , aMontrer)
typeAideContextuelle	→	infobull | vueDinamique
aMontrer	→	valeur | descriptionVue

Tableau 8.5 – *Grammaire correspondant aux opérations de visualisation.*

Pour une mise en relief, une condition dénote les UT à mettre en relief. Le système interpréteur doit fournir des mécanismes d'activation et de désactivation pour ce type d'opération, en utilisant éventuellement comme identifiant d'une mise en relief le nom défini pour celle-ci. L'activation d'une mise en forme consiste à l'application séquentielle de la liste d'opérations de transformation définies par celle-ci. La désactivation consiste à défaire le résultat obtenu de l'application des opérations de transformation. L'activation (désactivation) d'une mise en relief complexe implique l'activation (désactivation) séquentielle de la liste de mises en relief basiques formant celle-ci.

Notons que pour une opération de visualisation d'aide contextuelle, l'aide à afficher dépend du type d'aide. Cela implique que la valeur exprimée dans la production de la grammaire « aMontrer → valeur », sera une chaîne lexicale si le type d'aide consiste à l'affichage d'infobulles, tandis qu'elle sera le nom d'une description de vue (définie quelque part dans le module) dans le cas d'affichage dynamique d'une vue.

8.6. Opérations de navigation

En ce qui concerne la syntaxe des opérations de navigation, nous considérons d'abord une version simple d'opération de navigation, et ensuite nous examinerons des extensions possibles à l'opération de navigation.

D'après le chapitre précédent, une opération de navigation est constituée d'un nom, un type pourrant être : premier, dernier, suivant et précédent (ce deux derniers, éventuellement indexés); une source et une cible (*cf.* tableau 8.6).

opérationsNavigation	→	opérationNavigation , opérationsNavigation | ε
opérationNavigation	→	OpNav (nomOpérationNavigation , typeOpérationNavigation , source , cible)
nomOpérationNavigation	→	valeur
typeOpérationNavigation	→	premier | dernier | suivant | suivant [valeur] | précédent | précédent [valeur]
source	→	condition
cible	→	condition

Tableau 8.6 – *Grammaire correspondant aux opérations de navigation.*

8.6.1. Exemple d'opération de navigation

En guise d'exemple, je prends une opération de navigation extraite du projet REGAL. L'opération propose à l'utilisateur d'aller d'une phrase étiquetée comme annonce thématique vers l'annonce suivante. Celle-ci se formule en Sextant comme suit :

```
OpNav   (« Lire Thématique »,  suivant,
         UT(Type = Phrase,*,*,{(Étiquette Sémantique, Annonce Thématique)},*),
         UT(Type = Phrase,*,*,{(Étiquette Sémantique, Annonce Thématique)},*)  )
```

Tableau 8.7 – *Exemple d'opération de navigation.*

8.6.2. Extensions de l'opération de navigation

La notion d'opération de navigation peut s'étendre, au moins, de trois manières :

- en imposant des contraintes sur la source vis-à-vis de la cible et vice-versa ;

- en restreignant les UT sur lesquelles la recherche de la cible sera effectuée, ce qui correspond à la notion d'*empan* ;

- en permettant que les valeurs exprimées dans les conditions ne soient pas nécessairement encodées en dur par le concepteur, mais qu'elles puissent être fournies par le système interpréteur au moment d'exécuter l'opération de navigation, ce qui peut se réaliser en introduisant des *variables* dans les opérations de navigation.

Je présente par la suite ces trois possibles extensions de l'opération de navigation.

8.6.2.1. Relation Source-Cible

L'utilisation de conditions pour exprimer la source et la cible d'une opération de navigation a été examinée à la section 8.2.4. Considérons maintenant le fait de vouloir exprimer des contraintes sur la source vis-à-vis de la cible et vice-versa. Dans le cadre du projet RÉGAL, la plate-forme ContextO a été modifiée afin d'offrir à un utilisateur des opérations de navigation entre les éléments textuels ayant la même annotation (i.e. le même nom d'annotation avec la même valeur). Ce faisant, l'utilisateur peut se déplacer à travers des « conclusions », des « annonces thématiques », des « récapitulations », etc.

Supposons que l'on veuille exprimer une opération de navigation permettant de naviguer d'une conclusion vers la prochaine conclusion. La condition tant pour la source que pour la cible est qu'elle soit annotée en tant que « conclusion » et le type d'opération de navigation correspond à « suivant[1] ». En admettant que le type des UT annotées est « Phrase » et que le nom de l'annotation est « Étiquette Sémantique », la condition pour la source et pour la cible est :

UT(Type = Phrase, *, *, {(Étiquette Sémantique, Conclusion)}, *)

Une opération de navigation de type *suivant[1]* ayant cette condition sur la source et sur la cible, se déclenchera pour toutes les UT de type *Phrase* étiquetées sémantiquement comme « Conclusion ». Son exécution aura comme résultat le changement de l'UT active de la vue,

qui se traduit en un déplacement vers la prochaine UT de type *Phrase* (par rapport à l'UE source) étiquetée sémantiquement comme « Conclusion ».

Or, si l'on souhaite maintenant écrire une opération de navigation permettant de naviguer d'une « annonce thématique » vers la prochaine « annonce thématique », il faut créer une opération de navigation similaire à celle précédente, où la seule variation consiste à indiquer que la valeur de l'annotation est « Annonce Thématique » au lieu de « Conclusion ». En général, il sera nécessaire de créer autant d'opérations de navigation qu'il existe d'étiquettes sémantiques, tandis que le principe de navigation est le même : se déplacer d'une étiquette sémantique vers la prochaine comportant la même valeur. Autrement dit, on voudrait que la valeur de l'annotation exprimée dans la condition de la cible soit exactement la même que la valeur de celle exprimée dans la condition de la source.

Le même problème existe dans le cadre du projet NaviLire [Couto *et al.* 2005$_a$], [Couto *et al.* 2005$_b$], [Lundquist *et al.* 2006], où l'on souhaite exprimer des opérations de navigation pour les différents référents discursifs. Faute des contraintes entre la source et la cible, il faudrait écrire autant d'opérations qu'il existe de référents discursifs.

Différentes approches sont possibles afin d'exprimer des contraintes entre la source et la cible. Une possibilité consiste à utiliser le langage de conditions. La complexité de cette approche est liée à la sémantique des conditions : l'UT sur laquelle la condition est vérifiée est implicite. De plus, dans les conditions, telles qu'elles ont été définies, il n'est pas possible d'accéder à la valeur d'une annotation. Comment décrire alors que la valeur de l'annotation « Étiquette Sémantique » exprimée dans la condition de la cible soit exactement la même que la valeur exprimée dans la condition de la source ? D'abord, il est nécessaire de pouvoir accéder à la valeur d'une annotation, et pour ce faire, une opération nommée *valeurAnnotation* prenant comme arguments une UT et le nom d'une annotation pourrait se définir. Dans cette logique, pour référer la source et la cible, deux variables de nom *Source* et *Cible* sont introduites. La condition pourrait s'exprimer comme suit :

```
UT(Type=Phrase, *, *, {(Étiquette Sémantique, valeurAnnotation(Source, Étiquette Sémantique))}, *)
```

Or, il faudrait indiquer que cette condition doit se vérifier sur la cible, ce qui est fait si l'on écrit celle-ci comme condition de la cible. De même pour les contraintes sur la source où la cible est nécessaire.

Le point fort de cette approche consiste à offrir à l'encodeur une puissance significative au moment d'encoder des contraintes entre la source et la cible. Il pourrait ainsi utiliser des conditions sur la hiérarchie afin de demander, par exemple, que la source soit une UT fille de la cible, ou que la cible appartient aux ascendants de la source.

Le point faible est que nous sommes obligés d'élargir le langage de conditions afin de considérer un cas particulier, en obtenant ainsi des expressions qui ne pourront pas être utilisées dans un autre cas (par exemple pour exprimer les UT à mettre en relief dans une opération de visualisation). Cela complique les contrôles sémantiques à réaliser par le système interpréteur sur les constructions sémantiques.

En conséquence, dans un premier temps, l'approche adoptée pour tester l'égalité des valeurs d'annotations de nom identique consiste à énumérer dans une liste les noms des annotations à vérifier. Bien entendu, cette approche est correcte car la relation d'égalité est, en particulier, symétrique.

opérationNavigation	→	OpNav (nomOpérationNavigation , typeOpérationNavigation , source , cible , relationtSourceCible)
relationSourceCible	→	{ ensembleNomsAnnotations }
ensembleNomsAnnotations	→	nomAnnotation , ensembleNomsAnnotations | ε
nomAnnotation	→	valeur

Tableau 8.8 – *Grammaire correspondant aux opérations de navigation.*

Le tableau ci-dessus montre la grammaire correspondant aux opérations de navigation dans leur version simple.

8.6.2.2. Notion d'empan

Afin de restreindre l'espace de recherche des UT cibles, un *empan* de texte pourrait être spécifié. L'empan étant un ensemble d'UT, le langage permettrait de le définir en utilisant une condition. La question qui se pose est la suivante : comment calculer l'empan à partir d'une condition ? C'est-à-dire : comment choisir les UT pour lesquelles il faut évaluer la

condition afin de construire l'empan ? Il existe deux possibilités et l'on distinguera entre un *empan absolu* et un *empan relatif*.

Dans la définition d'un empan absolu, l'encodeur établit une condition à tester sur toutes les UT de la vue. Supposons que une UT cible, en plus de vérifier la condition de la cible, doive comporter un titre et appartenir à une section de rang supérieur à 2. L'encodeur peut dénoter cette contrainte en définissant comme empan absolu la condition suivante :

```
existeTitre  ET  estFils(UT(Type = Section, *, Rang > 2, *, *))
```

Le système interpréteur, au moment d'exécuter une opération de navigation, devra rechercher les cibles dans les UT vérifiant la condition de l'empan. Du point de vue des résultats, la définition d'un empan absolu consiste en conséquence à ajouter une condition à la condition de la source, et elle peut se concevoir comme l'application de l'opérateur logique ET à la condition de la source et à celle de l'empan :

```
condUTCibleRes = condUTCible  ET  condEmpanAbsolu
```

Dans la définition d'un empan relatif, l'encodeur établit une condition à partir de laquelle les UT cibles potentielles sont restreintes selon son rapport avec l'UT source (*i.e.* la condition est relative à la source). Par exemple, l'empan relatif d'une opération de navigation pourrait être les UT ascendants de l'UT source, ou les UT formant le titre de l'UT source. Nous revenons ainsi au problème concernant l'expression de la relation entre l'UT source et l'UT cible : pour exprimer que l'empan relatif consiste aux ascendants de l'UT source, l'encodeur pourrait dénoter :

```
estAscendant(Source)
```

Ce faisant, le système interpréteur, au moment d'exécuter une opération de navigation, devra rechercher les cibles dans les UT vérifiant la condition de l'empan, ce qui implique que

le terme *Source* est traité comme une variable calculée au moment de l'exécution de l'opération de navigation.

Une autre possibilité consiste à introduire des opérateurs et des opérations similaires a ceux définis par Crispino [2003] dans son langage LangTex, permettant de manipuler la structure textuelle de manière à ce qu'un empan relatif puisse être construit. Des conditions très complexes pourraient être exprimées. Par exemple, pour indiquer que l'empan relatif est formé par les UT frères de l'UT source ayant un titre où il y a au moins une UT qui comporte une annotation de nom « Thème », l'encodeur pourrait écrire comme condition :

```
existeNomAnnotation((selectTitre(selectFrères(Source))), « Thème »)
```

Dans l'exemple, l'opération *selectFrères* renvoie les UT frères de l'UT argument. De même, l'opération *selectTitre* renvoie les UT formant les UT du titre. L'opération *existeNomAnnotation* teste si une annotation d'un nom déterminé existe dans une UT appartenant à un ensemble d'UT.

Notons que cette condition implique, d'une part, un élargissement important du langage de conditions. D'autre part, celle-ci implique également un changement dans la sémantique du langage car il ne s'agit pas seulement de tester des conditions sur des UT mais aussi d'exécuter des opérations de sélection d'UT. De nouveau, il faut signaler que le langage de modélisation risque de devenir très complexe en vain.

Dans le cadre de cette thèse, la notion d'empan a été particulièrement étudiée sans avoir abouti, faute d'exemples réels d'utilisation, à une formalisation de celle-ci dans le langage Sextant.

8.6.2.3. Arguments dans les opérations de navigation

L'incorporation d'arguments à une opération de navigation incrémente la potentialité de celle-ci car elle permet à l'encodeur de modéliser des phénomènes pour lesquels il ne possède pas toute l'information de manière statique. Prenons comme exemple l'enchaînement de questions constaté dans le projet eVEIL. La question « Qu'est-ce que le MATE ? » est la conséquence de la réponse à la question : « En cas de risque majeur, qui est le responsable de l'information aux citoyens ? ». La réponse à une question de type « Qu'est-ce que *X* », dans le

cadre du projet eVEIL, peut se traduire comme la recherche d'une UT où le terme *X* est défini. Du point de vue de la navigation textuelle, étant donné que les textes ont été étiquetés sémantiquement par la plate-forme ContextO (*i.e.* les phrases conclusives ont une étiquette sémantique « Conclusion », celles définitoires une nommée « Définition », etc.), cette recherche consiste à définir une opération de navigation précisant comme cible une UT qui comporte une annotation de nom « Définition » avec la valeur *X*.

Sans la possibilité d'utiliser des variables dans les opérations de navigation, l'encodeur est obligé d'encoder dans la condition de la cible la valeur concrète de *X*. Ceci est, d'une part, assez incommode car plusieurs opérations similaires doivent être encodées individuellement. D'autre part, dans des cas d'utilisations réelles, il est impossible de prévoir toutes les valeurs possibles de X. L'encodeur pourra, au maximum, encoder un nombre d'opérations de navigation représentant la recherche des valeurs plus fréquentes.

L'encodeur pourra indiquer, pour chaque description de vue, un ensemble d'arguments. Les valeurs de ces arguments seront calculées par le système interpréteur au moment de l'exécution de l'opération de navigation. Les modifications à faire à la grammaire afin d'inclure la notion d'argument s'affichent dans le tableau ci-dessous.

opérationNavigation	→	OpNav (nomOpérationNavigation , typeOpérationNavigation , source , cible , relationSourceCible , { arguments })
arguments	→	nomArgument , arguments | ε
nomArgument	→	valeur

Tableau 8.9 – *Productions pour définir des arguments dans les opérations de navigation.*

La définition d'une opération de navigation précisant comme cible une UT qui comporte une annotation de nom « Définition » avec la valeur *X* s'affiche au tableau ci-dessous.

OpNav (« Rechercher Définition », {X}, suivant, UT(,*,*,*,*), UT(*,*,*,{(Définition, X)},*),{})*

Tableau 8.10 – *Exemple d'opération de navigation comportant des arguments.*

Notons que la source de cette opération de navigation peut être une UT quelconque et que le langage n'indique pas explicitement la manière dont les arguments sont calculés et chargés dans l'opérations de navigation. Constatons qu'il y a également un impact sur la grammaire du langage de conditions. En effet, là où l'on indiquait la possibilité d'avoir une valeur, en utilisant une variable spéciale nommée *valeur*, il peut y avoir maintenant tant une valeur (par exemple une chaîne de caractères ou un nombre entier) que le nom d'une variable.

L'encodeur doit-il toujours utiliser des variables pour exprimer des phénomènes comme celui qu'on vient de décrire ? Supposons que l'encodeur souhaite exprimer une opération de navigation recherchant la prochaine UT comportant une valeur pour un référent discursif. L'encodeur ne connaît pas la valeur du référent, mais il sait qu'elle est la même que celle comportée par l'UT source. Deux solutions sont plausibles. Dans la première, l'encodeur utilise la relation entre la source et la cible, comme le montre le tableau ci-dessous.

OpNav (« Rechercher RD », {}, suivant, UT(,*,*,*,*),* *UT(*,*,*,*,*), {Référent Discursif})*

Tableau 8.11 – *Recherche du même référent discursif (version sans arguments).*

Dans la deuxième (*cf.* tableau 8.12), l'encodeur utilise une variable pour indiquer la valeur de l'annotation recherchée.

OpNav (« Rechercher RD », {X}, suivant, UT(,*,*,*,*),* *UT(*,*,*,{(Référent Discursif, X)},*), {})*

Tableau 8.12 – *Exemple d'opération de navigation comportant des arguments.*

Quelle est alors la différence substantielle entre l'utilisation de variables et l'utilisation de la relation entre la source et la cible ? Principalement, lorsque l'encodeur utilise la relation entre la source et la cible, il est certain de l'origine de la valeur (*i.e.* il sait qu'elle vient de la

valeur d'une annotation et il connaît le nom de cette annotation). L'encodeur fait appel à l'utilisation de variables lorsqu'il ne connaît pas l'origine des valeurs et c'est le système interpréteur qui est chargé de les fournir. Il faut souligner qu'il s'agit de deux situations différentes et, en conséquence, les deux opérations de navigation proposées ne sont pas équivalentes.

8.7. Opérations de coordination

Toute description de vue comporte un ensemble, éventuellement vide, d'opérations de coordination. La définition d'une opération de coordination doit indiquer le *nom de la vue cible* et le *mode* de coordination (*cf.* tableau 8.13).

opérationsCoordination	→	opérationCoordination , opérationsCoordination | ε
opérationCoordination	→	OpCoord (nomVueCible , mode)
nomVueCible	→	valeur
mode	→	manuel | automatique

Tableau 8.13 – *Grammaire correspondant aux opérations de coordination.*

Il va de soi que l'encodeur est censé écrire des opérations de coordination cohérentes. Si, par exemple, il définit pour la vue de nom v_a une coordination manuelle avec la vue de nom v_b, et à la fois, il définit pour la même vue de nom v_a une coordination automatique avec la vue v_b, cela représente ou bien une incohérence ou, au moins, une redondance. Encore une fois, du point de vue du langage, cela implique un contrôle sémantique sur l'ensemble opérations de coordination, en particulier sur le rapport entre les différents ensembles appartenant aux descriptions de vue du module, et il reste à la charge du système interpréteur.

8.8. Synthèse

Dans ce chapitre j'ai présenté Sextant, un langage formel de modélisation des connaissances présentées au chapitre précédent. La syntaxe du langage a été développée de manière incrémentale en exemplifiant les différents constructeurs du langage. Le langage des condi-

tions joue un rôle fondamental dans Sextant. Afin de formaliser le sens des constructions syntaxiques du langage, une sémantique formelle est présentée au chapitre suivant.

Chapitre 9

Sémantique du langage Sextant

9.1. Introduction

Le sens des constructions syntaxiques du langage proposé est déterminé par une *sémantique opérationnelle*. A priori, un système interpréteur peut interpréter plusieurs textes et plusieurs modules en même temps. Néanmoins, du point de vue de la sémantique, je me suis intéressé à expliciter le comportement à vérifier par le système pour les actions suivantes :

- application d'un module à un texte ;

- évaluation d'une condition ;

- création d'une vue ;

- exécutions des opérations de visualisation, de navigation et de coordination.

L'application d'un module à un texte entraîne la création d'autant de vues de ce texte qu'il existe de descriptions de vues encodées dans le module. Il n'est pas intéressant d'indiquer la manière ou l'ordre dont le module ou le texte sont chargés par le système : on souhaite préciser que chaque description de vue crée une vue du texte, en décrivant comment la création est réalisée. Étant donné que le langage ne propose pas de constructions syntaxiques pour les actions mentionnées plus haut, chaque action est représentée par une opération fictive dont la sémantique est décrite. Une conséquence importante réside dans le fait que, faute de constructions syntaxiques pour les actions, le langage ne donne aucune indication sur des aspects concrets d'interaction avec l'utilisateur. Cela signifie, par exemple, que la manière dont une opération de visualisation peut être déclenchée ou dont une opération de navigation peut être exécutée par un utilisateur sera définie par le système interpréteur. Cette remarque est importante et elle est tout à fait cohérente avec la nature du langage, qui modélise des connaissances visuelles et navigationnelles et non pas des aspects d'interaction avec l'utilisateur.

Une autre conséquence, aussi importante, est que la sémantique offerte ne se limite pas à la sémantique du langage mais qu'elle définit également une partie de la sémantique d'un système interpréteur, cette dernière étant basée sur la première. En guise d'exemple, la création d'une vue n'est pas explicitée par le langage Sextant : elle est censée être le résultat de l'application d'une description de vue à un texte. C'est-à-dire que, du point de vue du langage, le seul élément de celui-ci participant à la création d'une vue est la description de vue. La sémantique étant définie pour la création d'une vue où les éléments participants sont un texte et une description de vue, cela nécessite la définition de la sémantique d'une description de vue, mais également la manipulation des constituants d'un texte. La situation est similaire en ce qui concerne les vues car celle-ci ne constitue pas un élément du langage Sextant. Il sera alors nécessaire de pouvoir manipuler de manière précise les textes et les vues, afin de définir la sémantique du système interpréteur et du langage Sextant. Nonobstant, il faut souligner que la sémantique donnée du système interpréteur sera partielle : plusieurs aspects, comme ceux concernant l'interaction avec l'utilisateur, ne sont pas pris en considération (*i.e.* sa sémantique n'est pas explicitée formellement).

Avant de décrire la sémantique de la création de vues, il est nécessaire de donner préalablement celle concernant l'évaluation d'une condition. De même pour les opérations de visualisation, de navigation et de coordination.

Excepté pour le cas de l'évaluation d'une condition, la sémantique des actions sera déterminée par le *changement d'état* que son exécution entraîne. L'état du système est fonction des états de toutes les vues actives. Notons que, une fois créée, une vue ne peut pas modifier son ensemble d'UT constituantes. L'utilisateur peut naviguer à l'intérieur de chaque vue (*i.e.* changer l'UT active), cette navigation impliquant des changements dans l'historique de navigation ; il peut également appliquer des opérations de mise en forme, ce qui changera les attributs de mise en forme pour les UT formant une vue. En plus de ses constituants, l'état d'une vue est alors fonction de l'UT active, de l'historique de navigation et de l'ensemble d'attributs de mise en forme.

L'exécution d'une opération de coordination entraîne généralement le changement d'état d'autres vues. Afin de modéliser cette situation, le nom d'une vue est considéré comme étant unique dans le système. Dans la pratique ceci n'est pas nécessairement exact car l'application d'un même module à deux textes différents crée, pour chaque texte, des vues qui s'appellent de la même manière. Néanmoins, le système interpréteur pourra nommer de façon univoque chaque vue créée.

Puisque les opérations de visualisation, de navigation et de coordination s'appuient sur des UT réelles appartenant au texte à partir duquel la vue a été créée, elles n'ont aucun sens dans les vues où les UT sont condensées, car celle-ci ne contiennent pas d'UT réelles. Pour ces types de vues, la sémantique ne sera pas explicitée. Nous pouvons conceptualiser celles-ci comme des vues plutôt « statiques » qui participent au processus de navigation textuelle en tant qu'éléments de support, mais qu'elles ne sont pas « navigables ».

Avant d'exprimer la sémantique du langage, des définitions permettant de manipuler les éléments qui n'appartiennent pas au langage sont essentielles. Cela correspond, du point de vue de la sémantique, à la définition des types sur lesquels celle-ci s'appuiera. D'une part, nous avons le texte, dont la représentation à considérer a été définie au chapitre 6. Nous avons besoin, au moment d'écrire la sémantique, de dénoter et de manipuler ses éléments constituants. D'autre part, il est également important de pouvoir dénoter et manipuler les vues.

Je présente par la suite l'ensemble de définitions préalables pour le texte et pour les vues. Ensuite, la sémantique du langage Sextant est développée.

9.2. Définitions pour le texte

Afin de définir la sémantique du langage Sextant, quelques définitions concernant la représentation de texte définie au chapitre 6 sont nécessaires. Ces définitions ont trait aux éléments constitutifs d'un texte et le rapport existant entre eux. A partir de la représentation proposée, il est possible de définir les différents éléments d'un texte sous forme d'objets comportant des attributs.

9.2.1. Attributs des éléments

Commençons par l'objet *Texte*, représentant un texte conforme à la représentation proposée au chapitre 6, qui contient les attributs suivants :

- *id* : un identifiant ;

- *titre* : liste ordonnée d'UT de premier niveau constituant le titre ; si le texte n'a pas de titre, alors cette liste est vide ;

- *tête* : ensemble d'UT complexes (ensembles, séquences, références et graphes) ;

- *corps* : liste ordonnée d'UT de premier niveau constituant le corps ;

En ce qui concerne les constituants de base du texte, une *UT de base* contient les attributs suivants :

- *Texte* : le Texte auquel cette UT appartient.

- *parent* : UT parent, ou *Texte* s'il s'agit ou bien d'une UT de niveau maximal dans la hiérarchie d'UT ou bien d'une UT de niveau maximal constituant le titre du texte ;

- *fils* : liste ordonnée d'UT filles ; si l'UT est une feuille dans la hiérarchie d'UT, alors cette liste est vide ;

- *titre* : liste ordonnée d'UT de premier niveau constituant le titre ; si l'UT n'a pas de titre, alors cette liste est vide ;

- *type* : chaîne de caractères indiquant le type de l'UT ;

- *nro* : chaîne de caractères indiquant le numéro de l'UT ;

- *rang* : chaîne de caractères indiquant le rang de l'UT ou ε si celui-ci n'a pas été indiqué ;

- *chaîne* : chaîne de caractères indiquant la chaîne lexicale de l'UT ou ε[44] si celle-ci n'existe pas ;

- *annotations* : ensemble de couples *(nom, valeur)* représentant les annotations affectées à l'UT. Chaque couple est préfixé d'un symbole indiquant le type d'héritage de l'annotation : vide, si l'annotation ne se propage pas ; « ↑ » pour signaler un héritage ascendant ; « ↓ » pour un héritage descendant et « ↕ » pour un héritage bidirectionnel.

Notation : Pour faire référence à un attribut d'un élément (*UT* ou *Texte*), j'utiliserai le point comme une fonction binaire infixe qui renvoie la valeur de l'attribut sollicité. Ainsi, par exemple, ut_1*.titre* renvoie la liste d'UT de premier niveau constituant le titre de l'UT ut_1 ; *utActive.parent* renvoie l'UT parent de l'UT *utActive* ; etc.

Remarquons que la manière dont est défini l'attribut *parent* exclut l'existence des éléments *titre* et *corps*, en les limitant à une nature d'attributs. Par exemple, parmi les UT appartenant à

[44] Afin de simplifier la sémantique, la chaîne de caractères vide, représentée par ε, signifie que la chaîne lexicale de l'UT n'a pas été définie. En conséquence, le fait qu'une UT ait une chaîne lexicale vide ne peut pas se distinguer du fait qu'une UT n'ait pas de chaîne lexicale. De même pour le rang. Ce choix aura des répercutions au moment de définir la sémantique du langage de conditions.

un titre, celles de premier niveau auront comme parent l'UT à laquelle le titre est affecté, ou *Texte* s'il s'agit du titre du texte. Cela implique que, quant à cette formalisation, les titres, c'est-à-dire les nœuds libellés comme TITRE dans la représentation proposée au chapitre 6, ne constitueront pas d'objets à manipuler. De même pour l'attribut *corps*.

Il est également important de remarquer la différence entre l'élément *Texte* et l'attribut Texte d'une UT de base. Le premier consiste à une valeur fictive à affecter afin de signaler que l'UT en question s'agit ou bien d'UT de niveau maximal dans la hiérarchie d'UT ou bien d'une UT de niveau maximal constituant le titre du texte. Le deuxième, par contre, contient comme valeur le texte auquel l'UT appartient. Cela implique que si l'on écrit :

$$ut_1.\text{Texte} = ut_2.\text{Texte}$$

on teste alors si les UT de base ut_1 et ut_2 appartiennent au même texte, tandis que si l'on écrit :

$$(ut_1.\text{parent} = ut_2.\text{parent}) \wedge (ut_1.\text{parent} = \textit{Texte})$$

on teste, par contre, si les UT de base ut_1 et ut_2 sont bien des UT de niveau maximal dans la hiérarchie d'UT ou bien des UT de niveau maximal constituant le titre du texte.

9.2.2. Types et opérations prédéfinis

Les chaînes de caractères sont considérées comme type de données élémentaire. De plus, je considère également comme types abstraits de données déjà connus les listes ordonnées, les ensembles et les dictionnaires. En conséquence, j'utiliserai sans définir formellement un ensemble d'opérations concernant chaque type, lesquelles sont mentionnées ci-après.

9.2.2.1. Opérations pour les listes ordonnées

Les opérations prédéfinies pour les listes ordonnées sont :

- *[]* : $\rightarrow$ liste

- *cons* : élément $\times$ liste $\rightarrow$ liste

- *insérer* : élément $\times$ liste $\rightarrow$ liste

- *supprimer* : élément × liste → liste

- *tête* : liste → élément

- *queue* : liste → liste

- *taille* : liste → N

- *position* : élément × liste → N

- *concat* : liste × liste → liste

Il faut signaler que l'opération *cons* insert un élément à la première position de la liste, tandis que l'opération *insérer* réalise l'insertion à la fin de la liste. L'opération *supprimer* supprime toutes les instances d'un élément dans une liste.

9.2.2.2. Opérations pour les ensembles

Les opérations prédéfinies pour les ensembles sont :

- *{}* → ensemble

- $\cup$: ensemble × ensemble → ensemble

- $\cap$: ensemble × ensemble → ensemble

- $\subset$: ensemble × ensemble → valeur de vérité

- $\in$: élément × ensemble → valeur de vérité

- *taille* : ensemble → N

9.2.2.3. Opérations pour les chaînes de caractères

Les opérations prédéfinies pour les chaînes de caractères sont :

- ε : → chaîne

- *concat* : chaîne × chaîne → chaîne

9.2.2.4. Opérations pour les dictionnaires de données

Les opérations prédéfinies pour les dictionnaires de données sont :

- *dico()* : → dictionnaire

- *insérer* : clé × élément × dictionnaire → dictionnaire

- *supprimer* : clé × dictionnaire → dictionnaire

- *existeClé* : clé × dictionnaire → valeur de vérité

- *existeÉlément* : élément × dictionnaire → valeur de vérité

- *taille* : dictionnaire → N

9.2.2.5. D'autres opérations

D'autre part, la notion d'appartenance d'un élément dans une liste (*i.e.* l'existence de, au moins, une occurrence de l'élément dans la liste) n'est pas prédéfinie, mais elle peut se définir de manière inductive comme suit :

$\in$: élément × liste → valeur de vérité

- $x \in [] =$ faux

- $x \in \text{cons}(x, l) =$ vrai

- $x \in \text{cons}(y, l) = x \in l$ (étant $x \neq y$)

De même, pour convertir une liste en un ensemble, une fonction spécifique peut se définir de manière inductive comme suit :

àEnsemble : liste → ensemble

- *àEnsemble*([]) = {}

- *àEnsemble*(cons(x, l)) = {x} ∪ àEnsemble(l)

La cardinalité de l'ensemble converti en liste différera de celle de la liste originale si celle-ci contient d'éléments répétés.

9.2.3. Définitions

Tout d'abord, nous définissons quelques concepts concernant le rapport existant entre les UT dans la hiérarchie. Cela comprend les notions d'ensemble d'ascendants et de descendants d'une UT, l'emboîtement d'UT, la notion d'ordre et distance entre UT, et la notion d'héritage d'annotations.

Définition 1 : ensemble d'ascendants

Soit *ut* une UT de base déterminée. L'ensemble d'ascendants *ASC* d'*ut* est défini comme une fonction de la manière suivante :

ASC : UT → ensemble

$$ASC(ut) = \begin{cases} \{Texte\} & \text{si } ut.\text{parent} = Texte \\ \{ut.\text{parent}\} \cup \text{ASC}(ut.\text{parent}) & \text{en cas contraire} \end{cases}$$

Notons, d'après la définition, que *Texte* appartient à l'ensemble d'ascendants de toute UT.

Définition 2 : ensemble de descendants

Soit *ut* une UT de base déterminée. L'ensemble de descendants *DESC* d'*ut* est défini comme une fonction de la manière suivante :

$DESC$: UT → ensemble

$$DESC(ut) = \text{àEnsemble(ut.titre)} \cup \text{àEnsemble(ut.fils)} \cup \bigcup_{t \in ut.titre} DESC(t) \cup \bigcup_{f \in ut.fils} DESC(f)$$

Notons d'après la définition que si *ut*.titre et *ut*.fils sont des listes vides, alors *DESC*(ut) est un ensemble vide. C'est-à-dire qu'une UT sans titre et feuille dans la hiérarchie ne comporte pas de descendants.

Nous pouvons étendre la définition des fonctions *ASC* et *DESC* afin de traiter des ensembles Ainsi, on obtient les définitions suivantes :

ASC_{ens} : ensemble → ensemble

$$ASC_{ens}(e) = \bigcup_{ut \in e} ASC(ut)$$

$DESC_{ens}$: ensemble → ensemble

$$DESC_{ens}(e) = \bigcup_{ut \in e} DESC(ut)$$

Pour prendre en considération l'élément *Texte*, on introduit deux cas particuliers pour *ASC* et *DESC*, selon lesquels l'élément *Texte* n'a aucun ascendant et les descendants correspondent à l'union des UT du titre du texte (ensemble $DESC_{titre}$) avec celles appartenant au corps du texte (ensemble $DESC_{corps}$) :

- $ASC(Texte) = \{\}$
- $DESC_{titre}(Texte) =$ àEnsemble($Texte$.titre) $\cup$ $DESC_{ens}(Texte$.titre)
- $DESC_{corps}(Texte) =$ àEnsemble($Texte$.corps) $\cup$ $DESC_{ens}(Texte$.corps)
- $DESC(Texte) = DESC_{titre}(Texte) \cup DESC_{corps}(Texte)$

Nous pouvons formaliser la notion d'adjacence d'UT présentée au chapitre 6 en introduisant la notion d'ordre d'UT. Informellement, cet ordre dérive d'un parcours en profondeur dans la hiérarchie d'UT, compte tenu que les UT du titre du texte précédent aux UT du corps, ainsi comme les UT du titre d'une UT précédent les UT filles de celle-ci. Définissons d'abord la notion d'emboîtement entre UT.

Définition 3 : emboîtement d'UT

Soient ut_1 et ut_2 des UT de base distinctes telles que ut_1.Texte = ut_2.Texte. L'UT ut_1 emboîte l'UT ut_2 si $ut_2 \in$ DESC(ut_1). L'élément *Texte* emboîte toute UT.

Intuitivement, si ut_1 emboîte ut_2, alors ut_1 précède ut_2, et vice versa. S'il n'y a pas d'emboîtement, l'ordre de préséance coïncide avec l'ordre de préséance des UT ascendantes de ut_1 et ut_2, qui sont frères dans la hiérarchie.

Dans la figure ci-dessous, le nœud numéro *2* précède le nœud numéro *9* car il existe une relation d'emboîtement, tandis que le nœud numéro *7* précède le nœud numéro *20* car le nœud numéro *4* « précède » (en tant que frère dans la hiérarchie) au nœud numéro *15*. La même idée s'applique à la hiérarchie d'UT, mais il faut prendre en considération l'existence des titres.

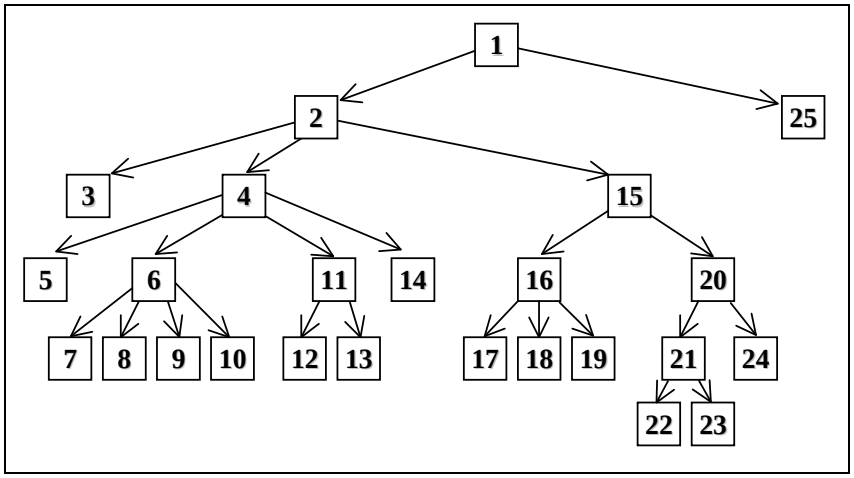

Figure 9.1 – *Ordre dans une arborescence.*

Afin de définir formellement un ordre des UT de base, on définit une relation d'ordre.

Définition 4 : relation d'ordre des UT de base

Soient ut_1 et ut_2 des UT de base distinctes telles que ut_1.Texte = ut_2.Texte. La relation $<_{UT} \subseteq UT^2$ est définie de la manière suivante :

- si ut_1 emboîte ut_2, alors $ut_1 <_{UT} ut_2$

- si ut_2 emboîte ut_1, alors $ut_2 <_{UT} ut_1$

- sinon

soient :

$T = ut_1$.Texte

$asc\_communs = \mathrm{ASC}(ut_1) \cap \mathrm{ASC}(ut_2)$

min_ac[45] $\in asc\_communs$ / $\forall y \in asc\_communs$, ($y \neq min\_ac \rightarrow$ y emboîte min_ac)

[45] Élément ascendant commun qui est emboîté par tous les autres éléments ascendants communs. Puisque l'élément *Texte* appartient à l'ensemble d'ascendants de toute UT, l'existence d'un ascendant commun est assurée et il est, « au pire », l'élément *Texte*.

$$ut\_pn^{46} = \begin{cases} \text{concat}(T.\text{titre}, T.\text{corps}) & \text{si } min\_ac = Texte \\ \text{concat}(min\_ac.\text{titre}, min\_ac.\text{fils}) & \text{sinon} \end{cases}$$

$utasc_1$ et $utasc_2$ / $(utasc_1 \in ut\_pn) \wedge (utasc_2 \in ut\_pn) \wedge (utasc_1 \neq utasc_2) \wedge$

$(utasc_1 = ut_1 \vee ut_1 \in \text{DESC}(utasc_1)) \wedge$

$(utasc_2 = ut_2 \vee ut_2 \in \text{DESC}(utasc_2)$

alors :

$ut_1 <_{UT} ut_2$ si position($utasc_1$, ut_pn) < position($utasc_2$, ut_pn)

$ut_2 <_{UT} ut_1$ en cas contraire.

Si l'on considère un ensemble d'UT (*i.e.* une collection d'UT où il n'y a pas d'UT doublonnées) la relation $<_{UT}$ introduit un ordre total dans les éléments et il est alors possible de définir les UT minimale et maximale (selon la relation $<_{UT}$) ainsi que la notion de distance entre UT.

Définition 5 : minimum et maximum selon la relation d'ordre des UT de base

Soit e un ensemble non vide d'UT. Les fonctions $MIN_{<UT}$ et $MAX_{<UT}$ renvoyant l'UT minimale et maximale, respectivement, selon la relation d'ordre $<_{UT}$ sont définies de la manière suivante :

$MIN_{<UT}$: ensemble → UT

$MIN_{<UT}(e) = \text{ut} \in e \ / \ \forall x \in e, (x \neq \text{ut} \rightarrow \text{ut} <_{UT} x)$

$MAX_{<UT}$: ensemble → UT

$MAX_{<UT}(e) = \text{ut} \in e \ / \ \forall x \in e, (x \neq \text{ut} \rightarrow x <_{UT} \text{ut})$

[46] Liste d'UT au premier niveau.

Afin de compléter les fonctions, pour un ensemble vide, celles-ci renvoient une valeur d'erreur, signalée par le symbole ⊥ :

$$MIN_{<UT}(\varnothing) = MAX_{<UT}(\varnothing) = \perp$$

Définition 6 : notion de distance selon la relation d'ordre des UT de base

Soit *e* un ensemble non vide d'UT. La fonction de distance entre deux UT appartenant à l'ensemble *e* est définie de la manière suivante :

$dist_{<UT}$: UT × UT × ensemble → Z

$$dist_{<UT}(ut_1, ut_2, e) = \begin{cases} \text{taille}(\{\text{ut} \in e, (ut_1 <_{UT} \text{ut}) \wedge (\text{ut} <_{UT} ut_2)\}) + 1 & \text{si } ut_1 <_{UT} ut_2 \\ 0 & \text{si } ut_1 = ut_2 \\ -\text{dist}(\text{ut2}, \text{ut1}, \text{e}) & \text{si } ut_2 <_{UT} ut_1 \end{cases}$$

Notons que, dans le premier cas, il faut additionner 1 au cardinal de l'ensemble des UT qui occurrent entre la première et la deuxième UT afin que deux UT contiguës aient une distance égale à 1. Signalons également l'existence de distances négatives et que la relation suivante est vérifiée :

$$dist_{<UT}(\text{MIN}_{<UT}(\text{e}), \text{MAX}_{<UT}(\text{e}), \text{e}) = \text{taille}(\text{e}) - 1$$

Définition 7 : héritage d'annotations d'UT

D'après la représentation de texte proposée, les annotations d'une UT peuvent être héritées ou bien par les UT descendants (*héritage descendant*), ou bien par les UT ascendants (*héritage ascendant*), ou bien dans les deux cas.

En conséquence, étant donné une UT, ses annotations héritées correspondent à l'union de celles héritées par héritage descendant et de celles qui le font par héritage ascendant. Deux fonctions, *AnnotationsHerAsc* et *AnnotationsHerDesc* sont définies afin de calculer, pour une

UT déterminée, ses annotations héritées par héritage ascendant et descendant, respectivement. Notons que les annotations héritées par héritage ascendant sont, de fait, les annotations des descendants dont le type d'héritage a été défini ou bien comme ascendant ou bien comme bidirectionnel. De même pour les annotations héritées par héritage descendant.

Soit *ut* une UT. Les fonctions *AnnotationsHerAsc* et *AnnotationsHerDesc* renvoyant les annotations héritées, respectivement, par héritage ascendant et descendant, sont définies de la manière suivante :

AnnotationsHerAsc : UT → ensemble

$$AnnotationsHerAsc\ (ut) = \bigcup_{ut_d \in DESC(ut)} \left\{ a \in ut_d.\text{annotations} \,/\, a = \uparrow (b,c) \wedge a = \updownarrow (b,c) \right\}$$

AnnotationsHerDesc : UT → ensemble

$$AnnotationsHerDesc\ (ut) = \bigcup_{ut_a \in (ASC(ut) - Texte)} \left\{ a \in ut_a.\text{annotations} \,/\, a = \downarrow (b,c) \wedge a = \updownarrow (b,c) \right\}$$

La fonctions *AnnotationsHéritées* renvoyant toutes les annotations héritées est définie de la manière suivante :

AnnotationsHéritées : UT → ensemble

$$AnnotationsHeritées\ (ut) = AnnotationsHerDesc\ (ut) \cup AnnotationsHerAsc\ (ut)$$

Toutes les définitions présentées dans cette section seront utilisées pour définir la sémantique du langage.

9.3. Définitions pour les vues

Une vue d'un texte est construite à partir d'une description de vue et elle est constituée par les UT vérifiant la condition d'appartenance (si celle-ci a été exprimée). La manière dont les UT d'une vue s'affichent dépendra du type de vue. Ainsi, pour une vue de type linéaire, la représentation des UT constitutives de la vue correspond à un enchaînement des chaînes

de caractères des UT feuilles dans la hiérarchie[47], tandis que pour une vue de type arborescente toutes les UT s'affichent sous forme de nœud libellé avec la chaîne lexicale. En conséquence, il peut exister des UT appartenant à une vue sans une représentation visuelle spécifique. Il y a une subtile différence entre une UT et la représentation de cette UT dans une vue déterminée. Dans une vue de type linéaire, une UT du texte n'est pas le même objet que la chaîne de caractères représentant cette UT. Dans une vue de type arborescente ou graphe, une UT du texte n'est pas le même objet que le nœud représentant cette UT. Évidemment, toute vue reflète, dans la mesure des ses possibilités, les phénomènes exprimés dans le texte. Le fait qu'une UT soit parent d'une autre UT, par exemple, se traduit, dans une vue arborescente, à deux nœuds liés par une relation de d'ascendance.

Afin de simplifier, je ferai référence indistinctement à une UT et à sa représentation dans une vue. Cela permet de définir les UT constituant une vue comme un ensemble d'UT[48], indépendamment du type de vue, et toute UT comportant les attributs définis à la section précédente (*i.e.* la hiérarchie d'UT peut s'inférer à partir des attributs des UT, en utilisant les définitions déjà présentées).

Une vue contient les attributs suivants :

- *nom* : le nom de la vue ;

- *type* : le type de vue selon les types déjà définis ;

- *contenu* : le contenu de la vue (*chaînes lexicales* ou *annotations*) ;

- *paramètres* : ensemble de paramètres sous forme de couples attribut – valeur ;

- *ensembleUT* : ensemble d'UT appartenant à la vue ;

- *utActive* : l'UT qui est active (*i.e.* sélectionnée actuellement) ;

- *visualisation* : ensemble d'opérations de visualisation ;

- *merActives* : liste d'opérations de mise en relief actives ;

- *navigation* : ensemble d'opérations de navigation ;

- *historique* : une liste d'UT représentant l'historique de navigation ;

---

[47] Notons que l'ordre de cet enchaînement est donné par la relation $<_{UT}$.

[48] Ce qui n'est pas tout à fait exact car une vue, en tant qu'élément visuel qui montre un texte mais qui n'est pas un texte, est constituée d'une collection de représentations d'UT, ces UT appartenant au texte à partir duquel la vue est créée.

- *coordination* : ensemble d'opérations de coordination.

Dans l'historique de navigation, la première UT de la liste coïncide avec la première UT sélectionnée au moment de créer la vue, celle-ci étant décrite en utilisant une condition dans la description de vue. Lorsque l'UT active change, la nouvelle UT active est insérée à la fin de l'historique.

Une vue enregistre toutes les mises en relief appliquées dans une liste de mises en relief. Toute mise en relief appartenant à cette liste est considérée *active*. On considère que toute UT a par défaut une mise en forme (libellé, couleurs de premier et arrière plan, et taille de la police) qui peut varier à cause de l'application des opérations de mise en relief. Comme toute opération de mise en relief est, à la fin, une liste d'opérations de mise en forme, au fur et à mesure que ces opérations de mise en forme sont exécutées, des « contradictions » sont possibles. Supposons que pour une UT déterminée, une mise en forme définit la taille de la police à 14, et que pour la même UT une autre mise en forme, appliquée postérieurement, définit la taille de la police à 10. Alors, il n'est pas possible que la chaîne de caractères de l'UT comporte à la fois une taille de police 10 et 14. Si l'on garde comme information seulement la dernière taille de la police, qui est 10, au moment de désactiver la dernière mise en forme, le système ne pourrait pas revenir à la taille 14. Le même raisonnement est valide pour les opérations de colorisation et d'ajout. En conséquence, il est nécessaire d'enregistrer toutes les opérations appliquées, mais aussi l'ordre dont elles ont été appliquées, ce qui justifie l'utilisation d'une liste comme type.

Afin d'exprimer dans la sémantique l'effet des opérations du langage, je considère qu'une vue offre l'ensemble des opérations suivantes :

- *désignerUTActive*(UT)

Cette opération indique visuellement à l'utilisateur l'UT active. Cette indication peut impliquer une mise en forme particulière pour l'UT (par exemple, un encadrement de la chaîne de caractères de l'UT dans une vue de type linéaire, ou une ligne foncée pour dessiner le nœud de l'UT dans une vue arborescente). De même, elle peut impliquer un changement dans la disposition spatiale de la vue (par exemple, en centrant dans la vue l'UT active).

- *désignerCouleurPremierPlan*(UT, couleur)

Cette opération définit la couleur en premier plan à utiliser pour afficher la chaîne de caractères d'une UT déterminée.

- *désignerCouleurArrièrePlan*(UT, couleur)

Cette opération définit la couleur en arrière plan à utiliser pour afficher la chaîne de caractères d'une UT déterminée.

- *désignerTaillePolice*(UT, entier)

Cette opération définit la taille de la police pour afficher la chaîne de caractères d'une UT déterminée.

- *désignerLibellé*(UT, chaîne de caractères)

Cette opération définit la chaîne de caractères à afficher pour une UT déterminée. L'utilité de cette opération est double : d'une part, elle permet de définir la chaîne initiale de caractères de la représentation d'une UT. D'autre part, elle permet de la redéfinir au fur et a mesure que le système interpréteur s'exécute. Pourquoi ne pas utiliser la chaîne de l'UT ? De fait, au moment de la création d'une vue la chaîne de la représentation coïncide avec celle de l'UT, mais dans le cas d'une mise en relief où des opérations d'ajout sont exprimées, c'est la chaîne de caractères de la représentation de l'UT qui change, celle de l'UT demeurant inchangée.

Les opérations ci-dessus sont considérées comme prédéfinies. L'implémentation de ces opérations ainsi que la sémantique spécifique seront prises en charge par le système interpréteur. Autrement dit, la sémantique décrite pour chaque opération sera traduite par la sémantique spécifique selon le type de vue. Par exemple, dans une vue de type linéaire, l'opération *désignerCouleurArrièrePlan* peut signifier, en termes de la vue, que le segment de caractères (indiqué par sa position de début et sa position de fin, par exemple) correspondant à la chaîne de caractères de l'UT spécifiée sera colorié selon la couleur indiquée, cette colorisation pouvant se concrétiser comme la définition d'un attribut de couleur affecté au segment.

De même que pour les définitions concernant le texte, pour faire référence à un attribut d'une vue, j'utiliserai le point comme une fonction binaire infixe qui renvoie la valeur de l'attribut sollicité. Ainsi, par exemple, *v.utActive* renvoie l'UT active de la vue *v* ; *v_par.coordination* renvoie l'ensemble d'opérations de coordination de la vue *v_par* ; etc.

Toutes les définitions présentées dans cette section seront utilisées pour définir la sémantique du langage.

9.4. Sémantique du langage Sextant

9.4.1. Introduction

L'intérêt de spécifier une sémantique formelle du langage consiste principalement à offrir les moyens afin que le système interpréteur implémente une *machine* qui *calcule* le langage. Comme il est intéressant d'indiquer la manière dont un effet est produit, car cela simplifie l'implémentation d'un interpréteur, une sémantique opérationnelle est définie [Nielson *et* Nielson 1992]. L'exécution des diverses opérations étant considérée comme des processus atomiques, la définition d'une *sémantique naturelle* est choisie, au lieu de choisir une *sémantique opérationnelle structurelle* [Plotkin 1982] [Hennessy 1990] qui permettrait de représenter des états intermédiaires dans l'exécution des opérations.

9.4.2. Notation

La sémantique des constructions du langage est spécifiée comme un système de transitions d'états, comportant deux types de configurations possibles :

$< I, e >$	qui représente que l'instruction I est exécuté dans l'état e
e	qui représente un état final

Dans la perspective d'une sémantique naturelle, nous sommes intéressés à la relation entre l'état initial et l'état final de l'exécution d'une instruction, ce qui est spécifiée par une *relation de transition*. On écrit une transition de la manière suivante :

$$< I, e > \rightarrow e'$$

Cette transition signifie que l'exécution de l'instruction I dans l'état e se termine et que le résultat est un nouvel état e'. La sémantique d'une instruction peut se construire (*i.e.* s'inférer) à partir des sémantiques d'autres transitions. Les transitions peuvent être alors combinées afin de construire une *règle* qui observe la forme générale suivante :

$$\frac{< I_1, e_1 > \rightarrow e_1', \cdots < I_n, e_n > \rightarrow e_n'}{< I, e > \rightarrow e'} \quad si \cdots$$

Les relations de transition exprimées au-dessus de la ligne séparatrice constituent les *prémisses* de la règle. La relation de transition au-dessous de la ligne constitue la *conclusion*. Une règle peut comporter un certain nombre de conditions, décrites à droite de la ligne, à vérifier au moment d'appliquer la règle. Les règles pour lesquels l'ensemble de prémisses est vide s'appellent *axiomes* et, afin de simplifier la lecture, la ligne séparatrice sera omise.

La sémantique des instructions peut ainsi se concevoir comme une fonction recevant un état et renvoyant un état. Si nous définissons :

$$\mathbf{S}_{SN} : \text{Instruction} \rightarrow (\text{État} \rightarrow \text{État})$$

Cela signifie que pour chaque instruction *I*, il existe une fonction $\mathbf{S}_{SN}[I] \in$ État → État, définit comme suit :

$$\mathbf{S}_{SN}[I]_e = e' \quad \text{si} \quad < I, e > \rightarrow e'$$

9.4.3. Instructions fictives

Comme notre langage est de type déclaratif et qu'il n'offre pas des instructions concrètes pour exprimer les actions du système interpréteur, les instructions fictives suivantes sont introduites pour les actions du système :

- *appMod* : applique un module de connaissances à un texte, obtenant comme résultat un ensemble de vues créées à partir des descriptions de vue exprimées dans le module ;

- *créerVue* : crée une vue d'un texte à partir d'une description de vue ;

- *sélUT* : sélectionne une UT déterminée comme UT active dans une vue ;

- *exécOpVis* : exécute une opération de visualisation dans une vue ;

- *exécOpNav* : exécute une opération de navigation dans une vue pour une UT source déterminée ;

- *exécOpCoord* : exécute une opération de coordination.

Il est important de souligner encore une fois qu'il en résulte que la sémantique à développer ne formalise pas seulement la sémantique des constructions du langage mais qu'elle explicite partiellement la sémantique d'un interpréteur à partir de la sémantique des constructions du langage.

Avant de décrire la sémantique opérationnelle du langage, il convient de préciser tout d'abord la notion d'état du système. Ensuite, le changement d'état résultant de l'exécution des actions possibles est développé pour les différentes actions.

9.4.4. Notion d'état

L'état du système est fonction des états de toutes les vues crées[49], l'état d'une vue correspondant à la valeur qui comporte chacun de ses attributs à un moment déterminé: nom, type, contenu, paramètres, ensemble d'UT, UT active, opérations de visualisation, opérations de mise en relief actives, opérations de navigation, historique de navigation, opérations de coordination. La différence, dans cette approche, entre *vue* et *état* d'une vue est alors très subtile, et je référerai indistinctement à une vue comme étant l'état d'une vue.

Une fois la vue créée, certains attributs seront considérés comme des attributs invariants, définis au moment de la création de la vue. Les attributs d'une vue pouvant varier après que la vue a été créée sont : l'UT active, la liste d'opérations de mise en relief actives et l'historique de navigation.

L'UT active d'une vue peut changer pour différentes raisons. En premier lieu, l'utilisateur peut choisir manuellement une UT et la signaler comme l'UT active. C'est le système interpréteur qui est chargé d'offrir les moyens pour réaliser cette opération, que nous décrivons comme l'exécution d'une opération nommée *sélUT*. En deuxième lieu, l'UT active d'une vue peut varier, c'est la conséquence de l'exécution d'une opération de navigation, qui définit comme UT active après l'exécution celle correspondant à la cible de l'opération. Enfin, l'exécution d'une opération de coordination (manuelle ou automatique) change l'UT active d'autres vues afin de la coordonner avec celle de la vue actuelle. Notons que le changement d'une UT active peut entraîner d'autres changements d'UT actives dans d'autres vues, à cause du déclenchement automatique d'opérations de coordination. Ce processus de déclenchement automatique est récursif et sa finalisation est assuré parce que, indépendamment du

[49] Comme l'action de détruire une vue n'est pas prise en considération, toutes les vues créées seront des vues actives et participeront à l'état du système.

nombre de cycles, il arrivera un moment où toutes les vues à coordonner auront comme UT active l'UT qui a déclenché initialement le processus.

Quant à la liste d'opérations de mise en relief actives, celle-ci variera au fur et à mesure que les opérations de mise en forme sont activées et désactivées. L'ordre d'activation est important car il permet préciser la sémantique des désactivations.

Bien que l'on puisse considérer que l'état d'une vue est fonction seulement des attributs variants, notons que les attributs invariants font également partie de l'état d'une vue puisqu'on veut spécifier la sémantique de la création d'une vue.

Définition 8 : état du système

L'*état du système* peut se concevoir comme un ensemble de vues, comportant chacune des valeurs spécifiques dans ses attributs. L'état initial du système est l'ensemble vide. Le changement de l'état du système est conséquence ou bien du changement de l'état d'une vue, ou bien de la création ou la destruction d'une nouvelle vue.

Nous modélisons l'état du système comme une fonction prenant le nom d'une vue (une chaîne de caractères) et renvoyant la vue correspondante :

$$\text{État} = \text{Nom de vue} \rightarrow \text{Vue}$$

Chaque état *e* spécifie alors une vue, dénotée *e nv*, pour chaque vue de nom *nv*. Notons que la destruction d'une vue est, en conséquence, très simple : il suffit de redéfinir la fonction d'état comme *non définie* pour la valeur correspondant au nom de la vue à détruire.

Définition 9 : changement de l'état du système

L'état du système étant l'état de ses vues actives, le changement de l'état du système est la conséquence ou bien du changement d'une de celles-ci, ou bien de l'ajout d'une nouvelle vue (*i.e.* la redéfinition de la fonction État afin d'expliciter la vue à renvoyer lorsque l'argument prend la valeur du nom de la nouvelle vue), ou bien de la destruction d'une vue existante. Afin de décrire le premier cas, où l'état d'une vue est remplacé par un nouvel état, on utilise la notion de *substitution* d'états, qui se dénote comme suit :

$$NomÉtat\ [\ NomVue \mapsto\ Vue\ ]$$

Cela signifie, par exemple, que *e* [*nv* ↦ *vue*] correspond à l'état identique à l'état *e* sauf que la vue de nom *nv* est définie comme *vue*. Formellement, on définit la substitution de la manière suivante :

$$(e\ [nv_1 \mapsto vue])\ nv_2 = \begin{cases} vue & \text{si } nv_1 = nv_2 \\ e\ nv_2 & \text{si } nv_1 \neq nv_2 \end{cases}$$

Définition 10 : changement de l'état d'une vue

L'état d'une vue étant les valeurs de ses attributs, le changement de l'état d'une vue est la conséquence du changement de la valeur d'une de ceux-ci. Afin de décrire ce changement de valeur, on utilise la même notion de *substitution*, en dénotant comme suit :

$$NomÉtat\ [\ NomVue\ .\ NomAttribut\ \mapsto\ Val\ ]$$

L'état sur lequel le changement est effectué est *NomÉtat*. Le changement consiste à redéfinir la valeur de l'attribut de nom *NomAttribut* dans la vue de nom *NomVue* à la valeur *Val*. Par exemple, pour définir, dans un état *e*, l'UT *ut* comme UT active de la vue de nom *nv*, on écrit :

$$e\ [\ nv\ .\ utActive\ \mapsto\ ut\ ]$$

Le résultat de cette application est un nouvel état *e'* contenant les mêmes vues que l'état précédent, mais où la valeur de l'attribut *utActive* de la vue de nom *nv* a été changée et affectée à la nouvelle valeur *ut*.

Formellement, on définit la substitution pour les vues de la manière suivante :

$$(e\,[nv_1.nAtt \mapsto v])\ nv_2 = \begin{cases} vue\ /\ (vue.nAtt = v) \wedge \\ \qquad (\forall nAtt_2, (nAtt_2 \neq nAtt) \rightarrow vue.nAtt = (e\ nv_1).nAtt_2)\ \text{si}\ nv_1 = nv_2 \\ \\ e\ nv_2 \quad \text{si}\ nv_1 \neq nv_2 \end{cases}$$

A partir de ces définitions on peut exprimer les règles et fonctions qui détermineront la sémantique du langage, ce qui est fait par la suite de manière progressive.

9.4.5. Méta-variables utilisées

La liste de méta-variables utilisées, correspondant aux types et constructions syntaxiques du langage (exception faite du langage de conditions, pour lequel celles-ci seront présentées postérieurement) est présentée ci-dessous.

T *est une méta-variable représentant un texte ;*

ut *est une méta-variable représentant une UT ;*

M *est une méta-variable représentant un module ;*

DV *est une méta-variable représentant une description de vue ;*

vue *est une méta-variable représentant une vue ;*

nom *est une méta-variable représentant un nom ;*

ens *est une méta-variable représentant un ensemble ;*

liste *est une méta-variable représentant une liste ;*

type *est une méta-variable représentant le type d'une vue ou le type d'une opération de navigation ;*

cont *est une méta-variable représentant le contenu d'une vue :* chaînes lexicales *ou* annotations *;*

opn, opc, mer et ot *sont des méta-variables représentant une opération de navigation, de coordination, de mise en relief et de transformation, respectivement ;*

v *est une méta-variable représentant une valeur quelconque ;*

Tableau 9.1 – *Liste de méta-variables utilisées.*

9.4.6. Application d'un module à un texte

Un module de connaissances contient un ensemble de descriptions de vue. L'application d'un module à un texte déterminé, consiste à la création d'une vue pour chaque description de vue, et elle est représentée par l'instruction suivante :

appMod (M, T)

Le module de connaissances *M* correspond à une construction suivant la syntaxe développée auparavant. Le texte *T* peut se concevoir comme une variable du type Texte définie à la section 9.2. Le résultat de cette instruction est la création d'un ensemble de vues, qui peut se concevoir également comme un ensemble de variables du type Vue définie à la section 9.3. Ces nouvelles vues causeront un changement dans l'état du système. En particulier, la sémantique de l'application d'un module consiste à ajouter ces nouvelles vues à celles déjà existantes, c'est-à-dire à l'état du système avant d'appliquer le module. Deux règles sémantiques sont définies dans le tableau ci-dessous.

$$[\text{appMod}_1] \quad < \text{appMod}(\text{Module}(\text{nom}, \{\}), T), e > \rightarrow e$$

$$[\text{appMod}_2] \quad \frac{< \text{créerVue}(DV, T), e > \rightarrow e', \quad < \text{appMod}(\text{Module}(\text{nom}, \text{ens}_{DV}), T), e' > \rightarrow e''}{< \text{appMod}(\text{Module}(\text{nom}, \{DV\} \cup \text{ens}_{DV}), T), e > \rightarrow e''} \quad \text{si } DV \notin \text{ens}_{DV}$$

Tableau 9.2 – *Règles sémantiques pour l'application d'un module de connaissances.*

La première règle, nommée [appMod$_1$], est un axiome et traite le cas où l'ensemble de descriptions de vue est vide. Dans ce cas, l'état du système reste inchangé. La deuxième règle, nommée [appMod$_2$], permet de déduire l'état résultant d'une application à partir de l'état résultant de la création d'une vue et de l'application d'un module qui ne contient pas la description de vue à partir de laquelle la vue a été créée.

Pour définir la sémantique de l'instruction *créerVue*, utilisée dans la règle [appMod$_2$], nous avons besoin préalablement d'expliciter formellement l'évaluation des conditions.

9.4.7. Évaluation d'une condition

La sémantique des conditions peut s'exprimer, au moins, de deux manières. La première consiste à définir une sémantique naturelle du langage de conditions. Pour ce faire, une nouvelle relation de transition $\rightarrow_{Cond}$ est créée, et les transitions observent la forme suivante :

$$< C, e > \rightarrow_{\text{Cond}} \textit{Valeur de Vérité}$$

Cela signifie que la condition *C* est évaluée pour l'état *e*, et la sémantique de cette évaluation correspond à la valeur de vérité de la condition pour l'état *e*. Étant donné qu'une condition est évaluée pour une UT déterminée et que dans la condition il peut exister des variables (dont la valeur est calculée par le système interpréteur avant d'évaluer la condition), la notion d'état pour les conditions peut se concevoir comme un couple formé par l'UT sur laquelle la condition doit s'évaluer et une fonction *Vars* prenant comme argument le nom d'une variable et renvoyant sa valeur.

Une deuxième approche consiste à définir une fonction sémantique chargée de calculer la valeur de vérité d'une expression de condition, étant donné la condition et l'état (couple UT – *Vars*). C'est cette approche, équivalente à la première, l'approche choisie dans le cadre des conditions car elle est à la fois, de mon point de vue, plus intuitive et plus lisible que l'autre.

Définition 11 : fonction *Vars*

Au moment d'évaluer une condition, le système interpréteur doit fournir une fonction *Vars* qui, pour chaque nom de variable utilisée dans la condition à évaluer, renvoie sa valeur. Le type de cette fonction est alors :

Vars : Nom de Variable → Valeur

Définition 12 : fonction sémantique **C**

La fonction sémantique **C** calcule la valeur de vérité d'une expression de condition, étant donné la condition et l'état (couple UT – *Vars*). Le type de cette fonction est alors :

$$\mathbf{C} : \text{Condition} \rightarrow \text{UT} \times (\text{Nom de Variable} \rightarrow \text{Valeur}) \rightarrow \text{Valeur de Vérité}$$

Il en résulte que pour chaque condition il existe une fonction sémantique spécifique prenant comme argument un état et renvoyant une valeur de vérité. En ce qui concerne la notation, on utilisera la notation suivante :

$$\mathbf{C}\ [\![c]\!]_{\mathrm{e}}$$

pour dénoter l'évaluation de la condition *c* dans l'état *e*. Un exemple de notation concret est le suivant :

$$\mathbf{C}\ [\![\text{UT}(*, *, *, *, x\ \textit{estSousChaîne}\ \text{ChaîneLexicale})]\!]_{(ut,\ Vars)}$$

qui dénote l'évaluation d'une condition simple sur l'UT ut, compte tenu du fait que les valeurs de variables sont définies par la fonction *Vars*. Les variables seront dénotées en italique, en utilisant les lettres *x*, *y* et *z*, éventuellement en y ajoutant des indices. Dans l'exemple ci-dessus, la valeur de la variable *x* s'obtient de l'application de la fonction Vars à cette variable ; c'ést-à-dire que la valeur de *x* est *Vars x*. La fonction **C** doit donc évaluer si *Vars x* est une sous-chaîne de la chaîne lexicale de l'UT *ut*.

Les lettres *x*, *y* et *z*, sont en conséquence des méta-variables représentant des variables. D'autres méta-variables seront utilisées (*cf.* tableau 9.3), correspondant aux constructions syntaxiques du langage de conditions.

cond *est une méta-variable représentant une condition ;*
c *est une méta-variable représentant une contrainte ;*
a *est une méta-variable représentant une annotation (un couple nom-valeur) ;*
op *est une méta-variable représentant une opération, et varie sur les valeurs suivantes : =, ≠, <, >, ≤, ≥, estPréfixe, estSuffixe et estSousChaîne ;*

Tableau 9.3 – *Liste de méta-variables utilisées pour le langage des conditions.*

La définition de la fonction sémantique **C** utilisera d'autres fonctions, qui sont progressivement définies. D'après la syntaxe du langage de conditions, il est nécessaire de définir la fonction **C** pour les conditions suivantes :

- composition des conditions (opérateurs NON, ET, OU).

- conditions simples (opérateur UT) ;

- conditions d'existence sur les éléments des UT (existeAnnotations, existeParent, etc.) ;

- conditions sur la hiérarchie (estParent, estDescendant, etc.) ;

Commençons par la définition de **C** dans le cas de composition de conditions. Le tableau ci-dessous montre les définitions concernant les opérateurs logiques NON, ET, OU.

$$\mathbf{C}\ [\text{NON } cond]_{(ut,\ Vars)} = \begin{cases} \textbf{vrai} & \text{si } \mathbf{C}\ [cond]_{(ut,\ Vars)} = \text{faux} \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

$$\mathbf{C}\ [cond_1 \text{ ET } cond_2]_{(ut,\ Vars)} = \begin{cases} \textbf{vrai} & \text{si } (\mathbf{C}\ [cond_1]_{(ut,\ Vars)} = \text{vrai}) \wedge (\mathbf{C}\ [cond_2]_{(ut,\ Vars)} = \text{vrai}) \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

$$\mathbf{C}\ [cond_1 \text{ OU } cond_2]_{(ut,\ Vars)} = \begin{cases} \textbf{vrai} & \text{si } (\mathbf{C}\ [cond_1]_{(ut,\ Vars)} = \text{vrai}) \vee (\mathbf{C}\ [cond_2]_{(ut,\ Vars)} = \text{vrai}) \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

Tableau 9.4 – *Fonction C pour la composition de conditions.*

Dans ce cas, la définition de la sémantique est assez simple : elle utilise les valeurs renvoyées par l'évaluation des conditions simples, et la sémantique des opérateurs de la logique classique.

Afin de ne pas faire une digression, la définition complète de la fonction sémantique **C** se trouve à l'annexe VII.

9.4.8. Création d'une vue

Une description de vue contient toutes les informations nécessaires pour créer une vue. Une vue est créée à partir d'une description de vue et un texte, ce qui peut se représenter par l'instruction suivante :

créerVue (DV, T)

La description de vue *DV* correspond à une construction suivant la syntaxe développée auparavant. Le texte *T* peut se concevoir comme une variable du type Texte défini à la section 9.2. Le résultat de cette instruction est un nouvelle vue, qui peut se concevoir également comme une variable du type Vue défini à la section 9.3.

Du point de vue de la sémantique, la création d'une vue entraîne la définition des attributs d'une nouvelle vue et la définition de la fonction d'état pour le nom de la nouvelle vue. Certains attributs sont copiés directement de la description de vue : type, ensemble de paramètres, ensemble d'opérations de visualisation, ensemble d'opérations de navigation, ensemble d'opérations de coordination. D'autres doivent être calculés en utilisant les attributs du texte et les valeurs indiquées dans la description de vue. Ainsi, par exemple, l'ensemble d'UT de la vue correspond à toutes les UT du texte vérifiant la condition d'appartenance exprimée dans la description de vue.

Intuitivement, la création d'une vue pourrait se considérer comme un axiome mais la sélection de la première UT dans la vue peut déclencher des coordinations avec d'autres vues, si des opérations de coordination automatique sont définies dans la description de vue. Afin de refléter cet effet (*cf.* tableau 7.16), on utilisera la sémantique d'une instruction auxiliaire *coordVues*, chargée de coordonner un ensemble de vues à partir d'un ensemble d'opérations de coordination.

[créerVue]

$$\frac{< \text{coordVues}(ut_{\text{Active}}, ens_{\text{Coord2}}), e > \rightarrow e', < \text{initLibellés}(nom, cont), e'' > \rightarrow e'''}{< \text{créerVue}(DV(type, cont, nom, cond_{\text{premUT}}, ens_{\text{Par}}, cond_{F}, ens_{\text{Vis}}, ens_{\text{Nav}}, ens_{\text{Coord}}), T), e > \rightarrow e'''}$$

si :

$e'' = e'[\, nom \mapsto vue\,] \wedge$

$(vue.type = type) \wedge (vue.contenu = cont) \wedge (vue.paramètres = ens_{Par}) \wedge$

$(vue.visualisation = ens_{Vis}) \wedge (vue.navigation = ens_{Nav}) \wedge (vue.coordination = ens_{Coord}) \wedge$

$(vue.ensembleUT = \{ut \in (DESC_{ens}(T.titre) \cup DESC_{ens}(T.corps)), \mathbf{C}\ [cond_F]_{(ut, \varnothing)} = vrai\}) \wedge$

$(vue.utActive = MIN_{<UT}(\{ut \in vue.ensembleUT, \mathbf{C}\ [cond_{premUT}]_{(ut, \varnothing)} = vrai\}) \wedge$

$(vue.merActives = []) \wedge (vue.historique = [vue.utActive]) \wedge$

$ut_{Active} = vue.utActive \wedge ens_{Coord2} = \{opc \in ens_{Coord}, opc.type = automatique\}$

Tableau 9.5 – *Règle sémantique pour la création d'une vue.*

Vu que plusieurs UT de la vue peuvent vérifier la condition de la première UT, celle sélectionné comme première UT active correspond à la première selon la relation d'ordre $<_{UT}$, renvoyée par la fonction $MIN_{<UT}$. Si l'ensemble d'UT vérifiant la condition premUT est vide, alors la valeur renvoyée par $MIN_{<UT}$ sera $\perp$. Cela signifie que l'UT active n'est pas définie, ou qu'elle est définie comme indéfinie. La sémantique de l'instruction *coordVues* indiquera le comportement du système lorsque l'UT avec laquelle coordonner est indéfinie.

Dans l'évaluation de la condition de filtre $cond_F$, la fonction *Vars* est dénotée comme étant vide. Cela veut dire qu'elle n'est pas utilisée dans la création de vues. Autrement dit, le type de condition que l'encodeur peut exprimer correspond à une condition sans variables. L'introduction de variables est réalisée afin de prendre en considération des phénomènes locaux aux UT, tel que l'enchaînement de questions dans le projet eVEIL. Il en résulte que la fonction *Vars* est calculée à partir d'une UT et fournie par le système interpréteur au moment, par exemple, d'exécuter une opération de navigation. L'utilisation d'une fonction *Vars* dans le processus de création d'une vue, bien que possible, implique en quelque sorte l'existence de variables globales. D'une part, nous n'avons pas trouvé d'exemples concrets d'utilisation. D'autre part, une telle situation pourrait se modéliser dans la sémantique en utilisant une notion d'*environnement*.

Quant à la liste initiale de mises en relief actives, celle-ci est vide, tandis que l'historique contient déjà la première UT sélectionnée.

Notons que les opérations de coordination exécutées sont seulement celles de type automatique. Cela est réalisé en faisant appel à une instruction auxiliaire *coordVues* dont la forme est :

coordVues (ut, ens)

Cette fonction sélectionne une UT *ut* déterminée comme UT active dans les vues indiquées dans les opérations de coordination formant l'ensemble *ens*. La sémantique de cette instruction est décrite ci-dessous.

$$[\text{coordVues}_1] \quad < \text{coordVues}(ut, \{\}), e > \rightarrow e$$

$$[\text{coordVues}_2] \quad \frac{< \text{exécOpCoord}(ut, opc), e > \rightarrow e', \quad < \text{coordVues}(ut, \text{ens}_{\text{Coord}}), e' > \rightarrow e''}{< \text{coordVues}(ut, \{opc\} \cup \text{ens}_{\text{Coord}}), e > \rightarrow e''} \quad \text{si } opc \notin \text{ens}_{\text{Coord}}$$

Tableau 9.6 – *Règles sémantiques pour la coordination d'un ensemble de vues.*

La première règle, nommée [coordVues$_1$], est un axiome et traite le cas où l'ensemble d'opérations de coordination est vide, l'état du système restant en conséquence inchangé. La deuxième règle, nommée [coordVues$_2$], permet de déduire l'état résultant d'une coordination à partir de l'état résultant de l'exécution d'une opération de coordination (*cf.* section 9.4.12) et de la coordination d'un ensemble d'opérations de coordination ne contenant pas l'opération de coordination exécutée.

Dans la création d'une vue, le libellé de chaque UT doit être déterminé. Cela se réalise en faisant appel à l'instruction auxiliaire *initLibellés*, qui reçoit le nom de la vue dont les libellés sont à initialiser et le type de contenu de la vue. Le libellé initial affecté à chaque UT dépend du contenu : si celui-ci correspond aux chaînes lexicales, on utilise directement la chaîne lexicale de l'UT ; si, par contre, il correspond aux annotations, on doit construire un libellé représentant toutes les annotations (*i.e.* toutes les couples nom – valeur). Il faut remarquer que

dans la sémantique de la création d'une vue, l'état pris en considération pour l'exécution de l'instruction *initLibellés* (e'') correspond à l'état résultant de la coordination des vues (e'), modifié (selon l'expression : $e'' = e'[\ nom \mapsto\ vue\ ]$). Cela est fait afin que la vue pour laquelle les libellés sont affectés soit déjà définie.

La sémantique de l'instruction *initLibellés* est décrite ci-dessous.

$$[\text{initLibellés}_1]\ \frac{< \text{initLibellés}(nom, cont, ens_{ut}), e > \rightarrow e'}{< \text{initLibellés}(nom, cont), e > \rightarrow e'} \quad \text{si } ens_{ut} = (e\ nom).\text{ensembleUT}$$

$$[\text{initLibellés}_2]\ < \text{initLibellés}(nom, cont, \{\}), e > \rightarrow e$$

$$[\text{initLibellés}_3]\ \frac{\begin{array}{l} < \text{désignerLibellé}(nom, ut, v), e > \rightarrow e', \\ < \text{initLibellés}(nom, cont, ens_{ut}), e' > \rightarrow e'' \end{array}}{< \text{initLibellés}(nom, cont, \{ut\} \cup ens_{ut}), e > \rightarrow e''}$$

si :

$$ut \notin ens_{ut} \wedge$$

$$v = \begin{cases} ut.\text{chaîne} & \text{si } cont = \text{chaînes lexicales} \\ \text{construireLibellé}(ut.\text{annotations}) & \text{si } cont = \text{annotations} \end{cases}$$

Tableau 9.7 – *Règles sémantiques pour l'initialisation des libellés.*

Comme on peut le constater, deux instructions *initLibellés* sont utilisées. La première, dont la règle sémantique est [initLibellés$_1$], correspond à celle présentée plus haut. Celle-ci fait appel à une deuxième instruction *initLibellés*, dont les arguments sont différents, chargée de traiter l'ensemble d'UT de la vue. Les règles de cette instruction, [initLibellés$_2$] et [initLibellés$_3$], reflètent un traitement inductif sur l'ensemble d'UT.

La sémantique de l'opération *désignerLibellé*, qui définit la chaîne de caractères effective à afficher dans une vue déterminée, est prédéfinie. Dans la règle [initLibellés$_3$] le libellé passé comme argument est ou bien la chaîne lexicale de l'UT ou bien un libellé construit par une fonction, considérée prédéfinie et nommée *construireLibellé*, selon le type de contenu de la vue.

9.4.9. Sélection d'une UT

Une UT peut être sélectionnée de trois manières différentes :

- directement par l'utilisateur (auquel cas, le système interpréteur doit fournir les moyens nécessaires pour le faire ; par exemple, en considérant un clic de la souris sur le libellé correspondant à l'UT comme une opération de sélection) ;

- comme résultat de l'exécution d'une opération de navigation (cela sera explicité par la sémantique de l'instruction *exécOpNav*) ;

- en conséquence du déclenchement des opérations de coordination automatique (ce qui a déjà été explicité dans la sémantique de l'instruction auxiliaire *coordVues* qui utilise la sémantique de l'instruction *exécOpCoord*).

L'action consistant à sélectionner une UT, quelqu'en soit la manière, peut se représenter par l'instruction suivante :

sélUT (ut, nom de vue)

La sélection de l'UT *ut* dans une vue est possible si et seulement si cette UT appartient à l'ensemble d'UT de la vue. Si l'UT n'y appartient pas, la vue reste inchangée. Il existe deux conséquences de la sélection d'une UT. En premier lieu, si la sélection est possible, l'UT sélectionnée devient la nouvelle UT active et l'historique de navigation est modifié afin de prendre en considération le changement d'UT [50]. En deuxième lieu, les opérations de coordination automatiques doivent être déclenchées (*i.e.* exécutées), toujours dans le cas où la sélection est possible.

Étant donné qu'une sélection peut déclencher des opérations qui auront comme résultat la sélection de la même UT dans d'autres vues, et ainsi suite, on peut se demander si ce processus de coordination s'arrête nécessairement. L'algorithme récursif sous-jacent à la sémantique souhaitée, c'est-à-dire l'algorithme assurant une condition d'arrêt, est le suivant :

```
sélUT (ut, nom) état
    Si (ut ∈ (état nom).ensembleUT)  et  (ut ≠ (état nom).utActive) alors
        état [ nom . utActive ↦ ut ]
        Pour chaque OpCoord(nom2, mode) ∈ (état nom).coordination faire
            Si mode = automatique alors
                sélUT(ut, nom2)
            FinSi
        FinPour
    FinSi
Fin sélUT
```

Tableau 9.8 – *Pseudo-code de l'algorithme de sélection d'une UT.*

L'arrêt de l'algorithme est assuré par la condition initiale que l'UT en entrée doit vérifier. D'une part, si celle-ci est ⊥ (*i.e.* une UT indéterminée ou « erreur »), qui n'appartient pas à l'ensemble d'UT de la vue, la sélection ne modifie rien de la vue et, en conséquence, ne propage rien non plus sur les vues avec lesquelles il faut se coordonner. De même dans le cas où il s'agit en effet d'une UT, mais que celle-ci n'appartient pas à l'ensemble d'UT de la vue. Le cas du ⊥ peut se produire lorsque le système doit sélectionner la première UT au moment de créer la vue, mais aucune UT ne vérifie la condition exprimée dans la description de vue. C'est la même situation lorsque l'exécution d'une opération de navigation ne renvoie aucune UT cible.

D'autre part, si l'UT en entrée est déjà l'UT active, que ce soit parce que pour cette vue une instance de *sélUT* s'est déjà appliquée ou parce que l'UT active correspondait déjà à l'UT en entrée au moment de faire le premier appel à *sélUT*, l'algorithme s'arrête.

En conséquence, si l'UT en entrée correspond à la valeur ⊥, la première instance de l'instruction *sélUT* s'arrête, sans faire un appel récursif. Sinon, puisque le nombre de vues n'est pas infini et puisque chaque exécution de *sélUT* pour une UT et une vue a comme résultat que l'UT active de la vue coïncide avec l'UT en entrée, le nombre maximale d'instances

[50] Il s'ensuit que l'historique est plutôt un historique d'UT sélectionnées, ce qui correspond à une vision où la sélection d'une UT faite par l'utilisateur ou résultante de l'exécution d'une opération coordination fait partie du phénomène de navigation textuelle.

exécutées possibles de l'instruction *sélUT* est égale au nombre de vues. Il en résulte que l'arrêt de l'algorithme est assuré.

Pour donner la sémantique, notons que la boucle de l'algorithme chargée de coordonner les vues pour lesquelles des opérations de coordination automatique sont définies, correspond tout à fait à l'instruction *coordVues* dont la sémantique a été exprimée plus haut.

$$[\text{sélUT}_1] \quad < \text{sélUT}(ut, nom), e > \rightarrow e \quad \text{si} \quad \begin{cases} (e\ nom).utActive = ut \\ ut \notin (e\ nom).ensembleUT \end{cases}^{\vee}$$

$$[\text{sélUT}_2] \quad \frac{< \text{désignerUTActive}(nom, ut), e > \rightarrow e' \quad < \text{coordVues}(ut, \text{ens}_{\text{CoordAut}}), e' > \rightarrow e''}{< \text{sélUT}(ut, nom), e > \rightarrow e'''}$$

si :

$(ut \in (e\ nom).ensembleUT) \wedge (ut \neq (e\ nom).utActive) \wedge$

$\text{ens}_{\text{CoordAut}} = \{opc \in (e\ nom).coordination, opc.type = automatique\}$

$e''' = e''\ [nom.utActive \mapsto ut]\ [nom.historique \mapsto insert(ut, (e\ nom).historique)]$

Tableau 9.9 – *Règles sémantiques pour la sélection d'une UT*

On utilise une règle nommée [sélUT$_1$] assurant le cas de base de la récursion, afin de représenter l'arrêt de l'algorithme, et une autre règle, nommée [sélUT$_2$], qui correspond à l'appel récursif qui utilise l'instruction *coordVues*, et qui modifie l'UT active et l'historique de navigation.

La sémantique de l'opération *désignerUTActive*, qui indique visuellement à l'utilisateur l'UT active, est considérée comme prédéfinie.

9.4.10. Exécution d'une opération de visualisation

La sémantique d'une opération de visualisation dépend du type d'opération. Dans le cas des opérations d'aide contextuelle, le résultat est plutôt associé à l'interaction avec l'utilisateur : le système lui affiche un « *infobull* » ou lui montre une vue crée dynamiquement. Étant donné que notre sémantique ne traite pas les phénomènes d'interaction et d'interfaces graphiques concrètes, la sémantique de ce type d'opération de visualisation pourrait s'appuyer, au mieux, sur la sémantique définie pour la création d'une vue.

La sémantique qui nous intéresse de définir est celle des opérations de mise en relief. Pour ce type d'opérations, la notion d'exécution est insuffisante : une mise en relief peut s'activer ou se désactiver, et chacune de ces actions peut se considérer comme une exécution de l'opération (puisque tant dans l'activation que dans la désactivation, tous les éléments décrits par l'opérations son mis en jeu).

Afin de simplifier la notation, j'utiliserai deux instructions auxiliaires (je laisse de côté l'instruction générique *exécOpVis*) : *activerMer* et *désactiverMer*, recevant l'opération de mise en relief, le nom de la vue sur laquelle cette opération dois s'activer ou se désactiver, et une fonction qui renvoie les valeurs des variables utilisées au sein de l'opération de mise en relief.

9.4.10.1. Activation d'une mise en relief

L'activation d'une opération de mise en relief dépendra du type d'opération : simple ou complexe. Dans le cas le plus général, l'idée consiste à parcourir les opérations de mise en relief simple composant une opération de mise en relief complexe. Pour chacune de ces opérations, on parcourt la liste d'opérations de transformation. L'application d'une opération de transformation dépendra du type de l'opération et il faut le faire pour toutes les UT de la vue qui vérifient la condition, ce qui implique un parcours dans les UT de la vue.

$$\frac{< \text{activerMer}_C(\text{MER}_C(\text{nom}, \text{liste}_{mer}), \text{nom}_{vue}, \text{Vars}), e > \rightarrow e'}{< \text{activerMer}(\text{MER}_C(\text{nom}, \text{liste}_{mer}), \text{nom}_{vue}, \text{Vars}), e > \rightarrow e''}$$

si :

[activerMer$_1$] $\text{MER}_C(\text{nom}, \text{liste}_{mer}) \notin (e\ \text{nom}_{vue}).\text{merActives} \wedge$
$e'' = e'[\text{nom}_{vue}.\text{merActives} \mapsto \text{liste}_{\text{merActives}}] \wedge$
$\text{liste}_{\text{merActives}} = \text{insérer}(\text{MER}_C(\text{nom}, \text{liste}_{mer}), (e\ \text{nom}_{vue}).\text{merActives})$

$$\frac{< \text{activerMer}_S(\text{MER}_S(\text{nom}, \text{cond}, \text{liste}_{OT}), \text{nom}_{vue}, \text{Vars}), e > \rightarrow e'}{< \text{activerMer}(\text{MER}_S(\text{nom}, \text{cond}, \text{liste}_{OT}), \text{nom}_{vue}, \text{Vars}), e > \rightarrow e''}$$

si :

[activerMer$_2$] $\text{MER}_S(\text{nom}, \text{cond}, \text{liste}_{OT}) \notin (e\ \text{nom}_{vue}).\text{merActives} \wedge$
$e'' = e'[\text{nom}_{vue}.\text{merActives} \mapsto \text{liste}_{\text{merActives}}] \wedge$
$\text{liste}_{\text{merActives}} = \text{insérer}(\text{MER}_S(\text{nom}, \text{cond}, \text{liste}_{OT}), (e\ \text{nom}_{vue}).\text{merActives})$

Tableau 9.10 – *Règles sémantiques pour l'activation d'une mise en relief.*

La sémantique de l'instruction *activerMer* est déterminée par celle des instructions *activerMer*$_C$ et *activerMer*$_S$, selon que la mise en relief à activer soit complexe ou simple, respectivement. L'instruction *activerMer*$_C$ active la liste de mises en relief simples qui composent une mise en relief complexe, comme le montre le tableau ci-dessous.

$$[\text{aciverMer}_{C1}]\ < \text{activerMer}_C(\text{MER}_C(\text{nom},[]),\text{nom}_{\text{vue}},\text{Vars}),\ e > \rightarrow e$$

$$[\text{activer-Mer}_{C2}]\ \frac{\begin{array}{l} < \text{activerMer}_S(\text{mer}_s,\text{nom}_{\text{vue}},\text{Vars}),\ e > \rightarrow e', \\ < \text{activerMer}_C(\text{MER}_C(\text{nom},\text{liste}_{\text{mer}}),\text{nom}_{\text{vue}},\text{Vars}),\ e' > \rightarrow e'' \end{array}}{< \text{activerMer}_C(\text{nom},\text{cons}(\text{mer}_s,\text{liste}_{\text{mer}}),\text{nom}_{\text{vue}},\text{Vars}),\ e > \rightarrow e''} \text{si } \text{mer}_s \notin \text{liste}_{\text{mer}}$$

Tableau 9.11 – *Règles sémantiques pour l'activation d'une opération de mise en relief complexe.*

Une opération de mise en relief simple qui est activé parce qu'elle fait partie d'une opération de mise en relief complexe, n'est pas insérée dans la liste d'opérations actives (*cf.* tableau 9.11) car c'est l'opération que la contient celle que sera y insérée. Cela évite la répétition de mises en relief simples dans la liste.

L'instruction *activerMer*$_S$ applique la liste d'opérations de transformation qui composent une mise en relief simple, comme le montre le tableau ci-dessous.

$$[\text{aciverMer}_{S1}]\ < \text{activerMer}_S(\text{MER}_S(\text{nom},\text{cond},[]),\text{nom}_{\text{vue}},\text{Vars}),\ e > \rightarrow e$$

$$[\text{activerMer}_{S2}]\ \frac{\begin{array}{l} < \text{appliquerOT}(\text{ot},\text{cond},\text{nom}_{\text{vue}},\text{Vars}),\ e > \rightarrow e', \\ < \text{activerMer}_S(\text{MER}_S(\text{nom},\text{cond},\text{liste}_{\text{OT}}),\text{nom}_{\text{vue}},\text{Vars}),\ e' > \rightarrow e'' \end{array}}{< \text{activerMer}_C(\text{MER}_S(\text{nom},\text{cond},\text{cons}(\text{ot},\text{liste}_{\text{OT}})),\text{nom}_{\text{vue}},\text{Vars}),\ e > \rightarrow e''} \text{si } \text{ot} \notin \text{liste}_{\text{OT}}$$

Tableau 9.12 – *Règles sémantiques pour l'activation d'une opération de mise en relief complexe.*

L'application des opérations de transformation est définie selon le type de transformation. Cette application doit se réaliser sur toutes les UT de la vue vérifiant la condition exprimée au sein de l'opération de mise en relief simple.

$$[\text{appliquerOT}_1] \quad \frac{< \text{appliquerOT}(ot, cond, nom, Vars, ens_{ut}), e > \rightarrow e'}{< \text{appliquerOT}(ot, cond, nom, Vars), e > \rightarrow e'}$$

si $ens_{ut} = (e\ nom).ensembleUT$

$$[\text{appliquerOT}_2] \quad < \text{appliquerOT}(ot, cond, nom, Vars, \{\}), e > \rightarrow e$$

$$[\text{appliquerOT}_3] \quad \frac{< \text{appliquerOTàUT}(ot, cond, nom, Vars, ut), e > \rightarrow e', \quad < \text{appliquerOT}(ot, cond, nom, Vars, ens_{ut}), e' > \rightarrow e''}{< \text{appliquerOT}(\{ut\} \cup ens_{ut}, cond, nom, Vars), e > \rightarrow e''} \text{ si } ut \notin ens_{ut}$$

$$[\text{appliquerOTàUT}_1] \quad \frac{< \text{désignerCouleurPremierPlan}(nom, ut, v), e > \rightarrow e',}{< \text{appliquerOTàUT}(\text{couleurPremierPlan}(v), cond, nom, Vars, ut), e > \rightarrow e'} \text{ si } \mathbf{C}\ [cond]_{(ut, Vars)} = \text{vrai}$$

$$[\text{appliquerOTàUT}_2] \quad \frac{< \text{désignerCouleurArrièrePlan}(nom, ut, v), e > \rightarrow e',}{< \text{appliquerOTàUT}(\text{couleurArrièrePlan}(v), cond, nom, Vars, ut), e > \rightarrow e'} \text{ si } \mathbf{C}\ [cond]_{(ut, Vars)} = \text{vrai}$$

$$[\text{appliquerOTàUT}_3] \quad \frac{< \text{désignerTaillePolice}(nom, ut, v), e > \rightarrow e',}{< \text{appliquerOTàUT}(\text{taillePolice}(v), cond, nom, Vars, ut), e > \rightarrow e'} \text{ si } \mathbf{C}\ [cond]_{(ut, Vars)} = \text{vrai}$$

$$[\text{appliquerOTàUT}_3] \quad \frac{< \text{désignerLibellé}(nom, ut, v_{libellé}), e > \rightarrow e',}{< \text{appliquerOTàUT}(\text{ajouter}(v_{placement}, v_{chaîne}), cond, nom, Vars, ut), e > \rightarrow e'}$$

si :

$\mathbf{C}\ [cond]_{(ut, Vars)} = \text{vrai} \quad \wedge$

$$v_{libellé} = \begin{cases} concat(v_{chaîne}, ut.chaîne) & si\ v_{placement} = préfixe \\ concat(ut.chaîne, v_{chaîne}) & si\ v_{placement} = suffixe \end{cases}$$

Tableau 9.13 – *Règles sémantiques pour l'application d'une opération de transformation.*

C'est *appliquerOTàUT* qui applique finalement l'opération de transformation à une UT, si celle-ci vérifie la condition imposée par l'opération.

9.4.10.2. Désactivation d'une mise en relief

La sémantique de la désactivation d'une mise en relief se définit en fonction de la sémantique d'initialisation de la mise en forme des UT et de la sémantique de l'activation des mises en relief restant de la suppression, de la liste de mises en relief actives, de la mise en relief à désactiver. L'état du système doit être équivalent à celui qu'on aurait obtenu si la mise en relief à désactiver n'avait jamais été activée.

$$[\text{désactiverMer}] \quad \frac{\begin{array}{l} < initLibellés(nom, cont), e > \rightarrow e', \\ < initTaille(nom), e'> \rightarrow e'', \\ < initCouleurs(nom), e''> \rightarrow e''', \\ < activerListeMer(liste_{mer}, nom, Vars), e'''> \rightarrow e^{IV} \end{array}}{< désactiverMer(mer, nom, Vars), e > \rightarrow e^{IV}}$$

$$\text{si :} \quad mer \in (e\ nom).merActives \wedge \quad liste_{mer} = sup\ primer(mer, (e\ nom).merActives) \wedge \quad cont = (e\ nom).contenu$$

Tableau 9.14 – *Règle sémantique pour la désactivation d'une mise en relief.*

La sémantique de cette instruction utilise celle d'une instruction nommée *ActiverListeMer* consistant à l'activation d'une liste d'opérations de mise en relief, décrite ci-dessous.

$$[\text{aciverListeMer}_1] \quad < activerListeMer([], nom, Vars), e > \rightarrow e$$

$$[\text{activerListeMer}_2]\ \frac{< \text{activerMer}(\text{mer}, \text{nom}, \text{Vars}), e > \rightarrow e',\ < \text{activerListeMer}(\text{liste}_{\text{mer}}, \text{nom}, \text{Vars}), e' > \rightarrow e''}{< \text{activerListeMer}(\text{cons}(\text{mer}, \text{liste}_{\text{mer}})), e > \rightarrow e''} \quad \text{si mer} \notin \text{liste}_{\text{mer}}$$

Tableau 9.15 – *Règles sémantiques pour l'activation d'une liste d'opérations de mise en relief.*

Notons que l'utilisation de la fonction *cons* assure l'ordre correct d'activations de différentes mises en relief, compte tenu que l'instruction est utilisée par *désactiverMer* en lui passant comme argument la liste d'opérations de mise en relief actives, cette liste ayant été construite en utilisant l'opération insérer, qui réalise l'insertion à la fin de la liste.

En outre, deux instructions, *initTaille* et *initCouleurs*, analogues à l'instruction *initLibellés*, sont utilisées, et leur sémantique est décrite ci-dessous.

$$[\text{initTaille}_1]\ \frac{< \text{initTaille}(\text{nom}, \text{ens}_{\text{ut}}), e > \rightarrow e'}{< \text{initTaille}(\text{nom}), e > \rightarrow e'} \quad \text{si ens}_{\text{ut}} = (\text{e nom}).\text{ensembleUT}$$

$$[\text{initTaille}_2]\ < \text{initTaille}(\text{nom}, \{\}), e > \rightarrow e$$

$$[\text{initTaille}_3]\ \frac{< \text{désignerTaillePolice}(\text{nom}, \text{ut}, v_{\text{tailleParDéfaut}}), e > \rightarrow e',\ < \text{initTaille}(\text{nom}, \text{ens}_{\text{ut}}), e' > \rightarrow e''}{< \text{initTaille}(\text{nom}, \{\text{ut}\} \cup \text{ens}_{\text{ut}}), e > \rightarrow e''} \quad \text{si ut} \notin \text{ens}_{\text{ut}}$$

Tableau 9.16 – *Règles sémantiques pour l'initialisation de la taille.*

$$[\text{initCouleurs}_1]\ \frac{< \text{initCouleurs}(\text{nom}, \text{ens}_{\text{ut}}), e > \rightarrow e'}{< \text{initCouleurs}(\text{nom}), e > \rightarrow e'} \quad \text{si ens}_{\text{ut}} = (\text{e nom}).\text{ensembleUT}$$

$$[\text{initCouleurs}_2]\ < \text{initCouleurs}(\text{nom}, \{\}), e > \rightarrow e$$

$$[\text{initCouleurs}_3]\ \frac{\begin{array}{l}< \text{désignerCouleurArrièrePlan}(\text{nom}, \text{ut}, v_{\text{coulAPParDéfaut}}), e > \rightarrow e',\\ < \text{désignerCouleurPremierPlan}(\text{nom}, \text{ut}, v_{\text{coulPPParDéfaut}}), e' > \rightarrow e'',\\ < \text{initCouleurs}(\text{nom}, \text{ens}_{\text{ut}}), e'' > \rightarrow e'''\end{array}}{< \text{initCouleurs}(\text{nom}, \{\text{ut}\} \cup \text{ens}_{\text{ut}}), e > \rightarrow e'''}\ \text{si ut} \notin \text{ens}_{\text{ut}}$$

Tableau 9.17 – *Règles sémantiques pour l'initialisation des couleurs.*

Les valeurs $v_{coulAPParDéfaut}$, $v_{coulPPParDéfaut}$ et $v_{tailleParDéfaut}$ sont considérées comme globales au système. Je l'utilise de manière peu rigoureuse ici, afin de ne pas définir la notion d'*environnement*, car il s'agit du seul exemple d'utilisation de variables globales, ce qui, a mon avis, ne justifie pas des définitions additionnelles.

9.4.11. Exécution d'une opération de navigation

Lorsqu'une opération de navigation est exécutée, le résultat est un changement de l'UT active de la vue où celle-ci s'est déclenchée. Du point de vue de la sémantique, on peut utiliser l'instruction *sélUT* définie auparavant, sous-entendu que selon le type d'opération de navigation on choisit l'UT cible à sélectionner. Si l'exécution de l'opération de navigation échoue (*i.e.* il n'existe pas de cible concrète), la valeur de l'UT à indiquer à *sélUT* sera ⊥.

L'exécution d'une opération de navigation peut se représenter par l'instruction suivante :

exécOpNav (opn, UT, nom de vue, vars)

Le quatrième argument consiste à une fonction renvoyant les valeurs des variables éventuellement utilisées dans l'opération de navigation à exécuter.

[exécOpNav]

$$\frac{< \text{sélUT}(\text{ut}_{\text{Cible}}, \text{nom}_{\text{vue}}), e > \rightarrow e'}{< \text{exécOpNav}(\text{OpNav}(\text{nom}, \text{type}, \text{cond}_S, \text{cond}_C, \text{ens}_{SC}), \text{ut}_{\text{Source}}, \text{nom}_{\text{vue}}, \text{Vars}), e > \rightarrow e'}$$

si :

$\mathbf{C}\ [\text{cond}_S]_{(\text{utSource, Vars})} = \text{vrai}\ \wedge$

$(\forall \text{nom}_A \in \text{ens}_{SC}, (\exists(\text{nom}_A, v) \in \text{ut}_{\text{Source}}.\text{annotations}) \wedge (\exists(\text{nom}_A, v) \in \text{ut}_{\text{Cible}}.\text{annotations}))\ \wedge$

$\text{ens}_{\text{Cibles}} = \{u \in (e\ \text{nom}).\text{ensembleUT}, \mathbf{C}\ [\text{cond}_C]_{(u, \text{Vars})} = \text{vrai}\}\ \wedge$

$$ut_{Cible} = \begin{cases} MIN_{<UT}(ens_{Cibles}) & \text{si type = premier} \\ ut_C \in ens_{Cibles} \,/ \\ \quad taille(\{u \in ens_{Cibles}, (ut_{Source} <_{UT} u) \wedge (u <_{UT} ut_C)\}) = n\text{-}1^{51} & \text{si type = suivant[n]} \\ ut_C \in ens_{Cibles} \,/ \\ \quad taille(\{u \in ens_{Cibles}, (ut_C <_{UT} u) \wedge (u <_{UT} ut_{Source})\}) = n\text{-}1 & \text{si type = précédent[n]} \\ MAX_{<UT}(ens_{Cibles}) & \text{si type = dernier} \end{cases}$$

Tableau 9.18 – *Règle sémantique pour l'exécution d'une opération de navigation.*

Bien que quatre règles sémantiques puissent s'écrire, chacune correspondant à un type d'opération de navigation, je préfère exprimer dans une seule règle les quatre cas possibles (*cf.* tableau 9.18).

La règle [exécOpNav] indique que l'état du système résultant d'exécuter une opération de navigation équivaut à l'état résultant de sélectionner l'UT cible dans la vue pour laquelle l'opération est exécutée. Le calcul de la cible est alors le point le plus important, mais il y a également certaines contraintes à vérifier. D'abord, l'UT pour laquelle cette opération de navigation s'exécute doit vérifier la condition de la cible. Ensuite, on teste l'égalité des valeurs d'annotation, pour chaque nom d'annotation indiqué dans l'ensemble d'annotations. Plus précisément, puisqu'à un nom d'annotation peuvent correspondre plusieurs valeurs, on teste l'existence, tant dans l'ensemble d'UT de la source que dans celui de la cible, d'une annotation de même nom et valeur, appartenant le nom à l'ensemble d'annotations à vérifier.

Afin de calculer la cible, une variable, nommée ens_{Cibles}, est utilisée pour des raisons de lisibilité. Il s'agit de l'ensemble de toutes les UT vérifiant la condition de la cible. A partir de cet ensemble on calcule la cible selon le type d'opération de navigation. Si le type est « premier », alors la cible est le premier élément de l'ensemble, d'après la relation d'ordre $<_{UT}$ dé-

[51] Notons que $n \in N^+$.

finie auparavant, ce qui est renvoyé par la fonction $MIN_{<UT}$. Par analogie,, si le type est « dernier », alors la cible est l'UT renvoyée par la fonction $MAX_{<UT}$. Si le type est « suivant[n] » ou « précédent[n] », on calcule l'UT cible comme une UT, appartenant à l'ensemble de cibles possibles, telle qu'il existe un nombre d'UT entre elle et l'UT source égal à l'argument *n* moins 1. La soustraction tient au fait que deux UT contiguës ont une distance égal à 1.

Pourquoi ne pas utiliser la fonction de distance ici ? Parce que l'UT source n'appartient pas forcément à l'ensemble d'UT cibles possibles. Si l'on veut utiliser la fonction de distance, on pourrait définir une fonction qui renvoie l'UT « plus proche » d'une UT dans un ensemble. L'application de cette fonction à l'UT source et l'ensemble d'UT cibles possibles, nous renverrait une UT (éventuellement la même UT source, si elle appartient à l'ensemble) que nous pourrions utiliser comme argument de la fonction distance.

9.4.12. Exécution d'une opération de coordination

Une opération de coordination contient le nom de la vue cible à coordonner. La coordination consiste à sélectionner dans cette vue cible une UT déterminée, et cette action peut se représenter par l'instruction suivante :

exécOpCoord (ut, opc)

Le résultat de cette instruction consiste en un changement éventuel de l'UT active dans la vue indiquée par l'opération de coordination. Si le type de coordination est manuel ou automatique n'est pas intéressant ici car on présuppose que l'opération doit s'exécuter quoi qu'il en soit son type. Le changement d'une UT active peut déclencher d'autres opérations de coordination, ce qui est fait par l'instruction *sélUT*. La sémantique de l'instruction *exécOpCoord* devient alors équivalente à celle de l'instruction *sélUT*. De ce fait, la seule règle sémantique nécessaire est définie comme suit :

$$[\text{exécOpCoord}]\quad \frac{< \text{sélUT}(\text{ut}, \text{nom}_{\text{vue}}), e > \rightarrow e'}{< \text{exécOpCoord}(\text{ut}, \text{OpCoord}(\text{nom}_{\text{vue}}, v), e > \rightarrow e'}$$

Tableau 9.19 – *Règle sémantique pour l'exécution d'une opération de coordination.*

9.5. Synthèse

Dans ce chapitre, une sémantique opérationnelle est proposée, afin de formaliser le sens des constructions syntaxiques de Sextant. Cette sémantique définit également une partie de la sémantique d'un système interpréteur du langage (*i.e.* une plate-forme de navigation textuelle). Bien que conscient de l'aspect un peu fastidieux de cette description, celle-ci offre une explication abstraite du langage et du comportement du système interpréteur, indépendante de toute implémentation informatique, où la notion d'état du système est explicitée, ainsi que la manière dont celle-ci change au fur et à mesure que l'utilisateur interagit avec le système. À partir de cette sémantique, des tests et des inférences de propriétés peuvent être réalisés sans avoir besoin d'une implémentation concrète. De plus, puisque le langage est un langage déclaratif à être utilisé dans un système interactif, montrer le changement d'état est, a mon avis, particulièrement utile. Par ailleurs, Cette sémantique montre un niveau de détail que généralement on ne trouve pas dans des sémantiques procédurales.

Enfin, la spécification formelle de la sémantique nous a permis de corriger la première implémentation de l'interpréteur du langage développé dans la plate-forme logicielle NaviTexte.

Quatrième partie

Une plate-forme de navigation textuelle et ses applications

Sommaire

Afin de tester notre approche de la navigation textuelle, une plate-forme logicielle, NaviTexte, a été développée. Cette implémentation implique divers composants. Le *chapitre 10* propose un encodage XML permettant d'encoder les textes selon la représentation informatique présentée au chapitre 6, afin d'être manipulés par la plate-forme logicielle NaviTexte. Au *chapitre 11*, une version XML du langage de modélisations de connaissances Sextant est définie. Le *chapitre 12* présente NaviTexte, la première plate-forme logicielle implémentant notre approche de navigation textuelle. Il faut souligner que NaviTexte est une plate-forme logicielle opérationnelle sur laquelle plusieurs applications concrètes ont été développées. Je présente dans *le chapitre 13* les applications créées en utilisant la plate-forme logicielle NaviTexte. Celles-ci sont assez hétérogènes, ce qui représente une preuve de la souplesse de NaviTexte en tant que plate-forme d'expérimentation.

Chapitre 10

Encodage XML de la représentation des textes

10.1. Constituants d'un texte et représentation XML

Il est important de souligner l'intérêt de bien distinguer entre la représentation informatique des textes présentée au chapitre 6 et l'encodage des textes, car la première admet plusieurs encodages informatiques concrets. Je développe par la suite l'encodage XML qui a été utilisé dans le cadre de cette thèse, en tant que support d'encodage des textes manipulés par la plate-forme logicielle NaviTexte.

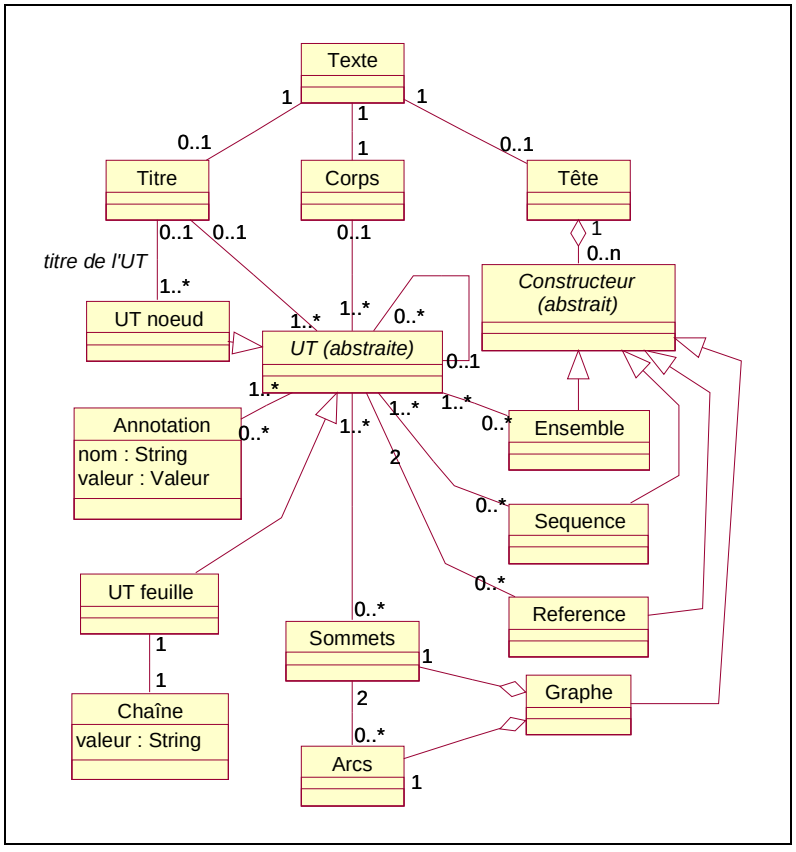

Figure 10.1 – *Diagramme UML illustrant le rapport existant entre les différents éléments de la représentation proposée.*

D'abord, nous pouvons inférer de la représentation abstraite les éléments suivants : *Texte*, *Titre*, *Tête*, *Corps*, *UT (nœud et feuille)*, *Chaîne*, *Annotation*, *Ensemble*, *Séquence*, *Référence* et *Graphe*. Le rapport existant entre ces différents éléments est affiché dans la figure 10.1 sous forme de diagramme de classes UML.

10.2. Grammaire de la représentation de texte

A partir de ce diagramme, une grammaire hors contexte peut être proposée (*cf.* tableau 10.1), offrant une première vue de haut niveau de la structuration des éléments. Notons qu'une hiérarchie d'UT est définie comme une liste d'UT (contenant éventuellement une seule UT) formant les UT au plus haut niveau dans la hiérarchie, ces UT pouvant être constituées d'autres UT.

Texte	→ *Titre* *Tête* *Corps*
Titre	→ *listeUT* | ε
Tête	→ *listeConst* | ε
Corps	→ *listeUT*
listeUT	→ *UT* | *listeUT* *UT*
listeConst	→ *Const* | *listeConst* *Const*
UT	→ *Titre* *listeAnnot* *Chaîne*
	| *listeAnnot* *Chaîne*
	| *Titre* *listeAnnot* *listeUT*
listeAnnot	→ *Annotation* | *listeAnnot* *Annotation*
Chaîne	→ *Valeur*
Annotation	→ *Nom* *Valeur*
Const	→ *Ensemble* | *Séquence* | *Référence* | *Graphe*
Ensemble	→ *listeUT*
Séquence	→ *listeUT*
Référence	→ *UT* *UT*
Graphe	→ *Sommets* *Arcs*
Sommets	→ *listeUT*
Arcs	→ *UT* *UT* | *Arcs* *UT* *UT*

Tableau 10.1 – *Grammaire de la représentation de texte.*

10.3. DTD de la représentation de texte

10.3.1. Structure globale du texte

Afin de définir une DTD à partir de la grammaire présentée plus haut, quelques aspects spécifiques concernant le formalisme XML doivent être résolus. Définir la hiérarchie d'UT en XML est simple car le formalisme est essentiellement récursif, permettant la définition de structures arborescentes. Définir la structure globale du texte (titre, tête et corps) est également simple (*cf.* tableau 10.2).

```
<!ELEMENT Texte (Titre?, Tete?, Corps)>
<!ELEMENT Titre (UT+)>
<!ELEMENT Tete ((Ensemble | Sequence | Reference | Graphe)*)>
<!ELEMENT Corps    (UT+)>
```

Tableau 10.2 – *Fragment de la DTD concernant l'élément* Texte.

10.3.2. Liens entre la tête et le corps : comment identifier les UT

Un problème se présente lorsque nous souhaitons exprimer en XML que les constructeurs dans la *tête* pointent vers des UT dans le *corps*, étant donné qu'ils ne peuvent pas créer de nouvelles UT de base. Il est nécessaire de pouvoir référencer une UT et pour ce faire il faut les identifier. Une possibilité consiste à utiliser les attributs de type ID offerts par XML. Ce faisant, les éléments de la tête utiliseraient des attributs de type IDREF pour référencer les UT selon leurs attributs ID. Si nous utilisons cette approche, toute UT doit comporter une valeur différente d'ID. Une numérotation séquentielle reflétant les positions des UT selon un parcours de l'arborescence doit être prévue par l'encodeur. De plus, les UT définissant une hiérarchie dans le corps, il peut apparaître intéressant de leur identifier conformément à sa position dans la hiérarchie. Ainsi, l'encodeur pourrait utiliser comme identifiants des chaînes telles que « *1* », « *2.6.2* », « *3.6.1.15* », etc. Or, ces deux manières d'identifier les UT se heurtent à deux inconvénients : la première ne prend pas en considération le type de l'UT comme élément différentiel d'une UT, et la deuxième peut devenir assez complexe dans le cas où l'encodage est fait par un encodeur humain. En effet, une numérotation séquentielle selon le type d'UT semble plus raisonnable et principalement plus lisible : l'encodeur numérote séquentiellement les UT en tenant compte de son type. Il y aura ainsi une séquence numérique,

par exemple, pour les UT de type « Paragraphe », une autre pour celles de type « Phrase », une autre pour celles de type « Proposition », etc. En outre, il n'est pas toujours intéressant de signaler la position dans la hiérarchie : il est généralement intéressant pour les UT de type « Section » ou pour celles conformant un schéma numéroté.

Aussi, je propose pour identifier une UT deux attributs additionnels au *type* : son *numéro séquentiel* (selon le type) et, optionnellement, son *rang* (*cf.* tableau 10.3). C'est-à-dire qu'une UT est identifiée par son type, son numéro et, s'il est présent, son rang. Il s'ensuit, par exemple, que le syntagme nominal numéro 15 n'est pas la même UT que le mot numéro 15, que la section numéro 1 de rang 1 n'est pas la même que celle de numéro 1 et rang 3, etc.

```
<!ELEMENT UT ((Titre?, Annotation*, UT+) | (Titre?, Annotation*, Chaine))>
<!ATTLIST UT    Type  CDATA  #REQUIRED
                Nro   CDATA  #REQUIRED
                Rang  CDATA  #IMPLIED>
<!ELEMENT Chaine   (#PCDATA)>
<!ELEMENT Annotation (#PCDATA)>
   <!ATTLIST Annotation  Nom   CDATA  #REQUIRED>
```

Tableau 10.3 – *Fragment de la DTD concernant l'élément* UT.

Par ailleurs, l'élément correspondant aux UT est défini en reflétant son rôle dans la hiérarchie : *nœud* ou *feuille*. Dans le premier cas, l'UT est constituée d'UT filles. Dans le deuxième, celle-ci a une chaîne lexicale. Notons que dans les deux cas, l'UT peut comporter un titre et un ensemble non limité d'annotations.

10.3.3. Référencement d'une UT

L'élément UT ainsi défini, il est nécessaire de préciser la manière dont les différentes instances d'UT seront référencées dans la tête par les constructeurs. L'utilisation des langages XLink [XLink] et XPointer [XPointer] a été écartée, d'une part, à cause de sa relative complexité et, d'autre part, parce que sa formalisation était encore instable au moment d'étudier la représentation de texte.

Une solution possible consiste à utiliser le même élément UT mais n'indiquant que son type, son numéro et éventuellement son rang. Du point de vue XML, il est possible de changer la définition de l'élément UT donnée au tableau 4.6, en désignant l'élément *Chaine* comme optionnel. Ce faisant, l'élément XML « <*UT Type*="*Phrase*" *Nro*="2" /> » ne constituerait pas une UT vide mais un pointeur vers une UT existante dans le *corps*.

Cette solution est néanmoins peu orthodoxe et nous pouvons, par contre, créer un nouvel élément XML représentant un pointeur vers une UT. Cet élément XML est nommé *UTP* et sa définition est décrite dans le tableau 10.4.

```
<!ELEMENT   UTP   EMPTY >
<!ATTLIST UTP  Type  CDATA  #REQUIRED
               Nro   CDATA  #REQUIRED
               Rang  CDATA  #IMPLIED>
```

Tableau 10.4 – *Fragment de la DTD concernant les pointeurs vers les* UT.

Constatons au passage que c'est le système de traitement automatique de textes qui est responsable d'interpréter le fait que les éléments UTP agissent, de fait, comme des pointeurs vers des UT.

10.3.4. Constructeurs de nouvelles UT

Quant aux constructeurs, j'utilise celui de *Séquence* pour exemplifier sa représentation XML. Une séquence peut s'exprimer comme une séquence d'UT ou, si elles sont « adjacentes », celle-ci peut s'exprimer en termes de la première et dernière UT[52]. Une séquence doit comporter un *numéro* afin d'être identifiée, et elle est susceptible d'avoir optionnellement un *type* et un *titre*[53]. De plus, une séquence peut avoir un nombre non limité d'annotations (*cf.* tableau 10.5).

[52] Le cas plus général où les constructeurs peuvent être créés à partir d'autres constructeurs n'a pas été entrepris dans le cadre de cette thèse car nous n'avons pas identifié suffisamment d'exemples justifiant cette représentation complexe.

[53] Les titres seront utilisés par la plate-forme *NaviTexte*.

```
<!ELEMENT Sequence ((Annotation*, Debut, Fin) | (Annotation*, UTP+))>
    <!ATTLIST Sequence Nro   CDATA #REQUIRED>
    <!ATTLIST Sequence Type  CDATA #IMPLIED>
    <!ATTLIST Sequence Titre CDATA #IMPLIED>
<!ELEMENT Debut (UTP)>
<!ELEMENT Fin   (UTP)>
```

Tableau 10.5 – *Fragment de la DTD concernant l'élément* Séquence.

Le tableau 10.6 illustre un exemple XML de la séquence définie par la figure 6.9 du chapitre 6.

```
<Sequence Type="Cadre" Titre="Introducteurs de cadre de discours" Nro="1")>
    < Annotation Nom="Type Cadre" >Thématique</Annotation>
<UTP Type="Introducteur" Nro="2">
<UTP Type="Introducteur" Nro="5">
    <UTP Type="Introducteur" Nro="9">
</Sequence>
```

Tableau 10.6 – *Exemple XML d'une séquence.*

10.3.5. Notion d'adjacence

Il faut préciser que la notion d'adjacence dans une arborescence correspond à une adjacence dans la structure linéaire résultante d'un parcours en profondeur, comme l'illustre la figure 10.2.

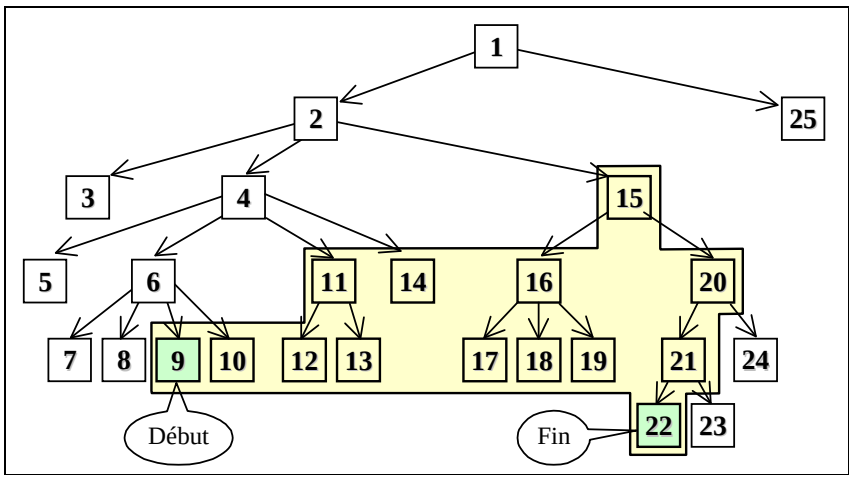

Figure 10.2 – *Adjacence dans une arborescence.*

Une séquence définie en indiquant le nœud numéro 9 comme début et le numéro 22 comme fin, comprendra tous les nœuds qui sont en gras dans la figure, entourés par l'encadrement coloré. L'ordre dans la séquence répond à l'ordre résultant d'un parcours en profondeur de l'arborescence. Notons que ni le début ni la fin de la séquence doivent être impérativement des feuilles dans l'arborescence.

10.4. Constat critique de la représentation XML définie

La représentation XML proposée est une parmi plusieurs possibles et constitue une couche au dessus du XML brut. La DTD complète se trouve à l'annexe II et plusieurs exemples de textes XML suivant cette représentation sont regroupés à l'annexe III.

À mon sens, cette représentation hérite des aspects positifs et négatifs provenant de la représentation abstraite (*cf.* chapitre 6). Il convient de remarquer que les nouvelles UT définies dans la tête sont légèrement plus complexes à cause du formalisme XML, et dans des longs textes celles-ci risquent de devenir assez lourdes. Il en résulte que des éditeurs XML spécifiques se révèlent nécessaires dans le cas d'un encodeur humain.

M'étant inspiré du modèle TEI Lite, la question qui se pose est pourquoi je n'ai pas utilisé directement la DTD proposée par celui-ci. Il existe à cela deux raisons principales. D'une part, les titres ne sont pas considérés comme étant composés d'unités textuelles mais comme des chaînes de caractères. D'autre part, le TEI Lite ne propose pas de constructeurs d'ensembles, de séquences et de graphes. Il est cependant possible de définir des références en utilisant des éléments comme *ptr*, *xptr* ou *ref*, mais celles-ci sont affectées aux éléments

mêmes, au lieu de le faire séparément. Cela signifie qu'il existe la possibilité d'exprimer des références mais que celles-ci ne sont pas traitées en tant qu'objets comportant des attributs (référent et référé) et des annotations sinon qu'elles doivent être inférées du contenu textuel. Autrement dit, la source n'est pas explicitée pour la référence car elle correspond à l'élément auquel la référence est affectée.

10.5. Synthèse

Ce chapitre propose un encodage XML permettant d'encoder les textes selon la représentation informatique présentée au chapitre 6, afin d'être manipulés par la plate-forme logicielle NaviTexte.

Trois cas de figure sont possibles pour utiliser l'encodage offert. En premier, un encodeur humain crée des textes à l'aide d'un éditeur XML. Ce cas correspond, par exemple, à l'utilisation de NaviTexte dans le cadre de l'apprentissage du français en tant que langue étrangère, application nommée *NaviLire* que sera présentée au chapitre 13. Actuellement, un projet pour développer un éditeur spécifique de textes NaviTexte est en cours à la Faculté d'Ingénierie de l'Université de la République (Uruguay).

Dans le deuxième cas, ce sont des systèmes de traitement automatique de textes qui annotent les textes et génèrent des textes résultants conformes à l'encodage XML proposée.

Dans le dernier cas, les textes issus d'un traitement sont automatiquement convertis vers la représentation proposée. ContextO est un bon exemple de cas de figure : un outil de transformation a été réalisé[54]. Du point de vue informatique, il s'agit d'une feuille XSLT prenant en entrée un texte conforme à la DTD ContextO et produisant en sortie un texte conforme à la DTD NaviTexte. Il va de soi que d'autres transformations analogues sont envisageables.

[54] Ce projet a été réalisé en 2005 par deux étudiants, Isabelle Ranque et Benoit Lamey, du DESS ILSI de l'Université Paris-Sorbonne.

Chapitre 11

Encodage XML du langage de modélisation des connaissances Sextant

11.1. Le choix de XML

Diverses implémentations du langage de modélisation des connaissances Sextant, développé au chapitre 8, sont envisageables. L'une d'entre elles consiste à définir une DTD représentant la grammaire définie pour le langage. Ce faisant, tout document XML conforme à cette DTD sera une instance du langage Sextant.

Le bénéfice d'utiliser XML comme langage de représentation concret du langage Sextant réside, au-delà du fait que XML constitue actuellement un standard, dans la construction d'un compilateur ou d'un interpréteur pour le langage. Au lieu d'utiliser des outils tels que JLex [JLex] et CUP [CUP] afin de construire, respectivement, les analyseurs lexicographique et syntaxique, ce qui correspond aux deux premières étapes d'un compilateur, on peut utiliser directement des outils tels que JDOM [JDOM], qui construisent un arbre d'analyse syntaxique à partir de l'analyse d'un document XML. Cela facilite notablement l'implémentation d'un système interpréteur.

11.2. La DTD

La correspondance entre la grammaire proposée et la DTD est assez immédiate. Un fragment de la DTD qui correspond à la grammaire de haut niveau présentée dans le tableau 8.1 du chapitre 8, est présenté ci-dessous.

```
<!ELEMENT Module (DescriptionDeVue+)>
    <!ATTLIST Module Nom CDATA #IMPLIED>
<!ELEMENT DescriptionDeVue (PremiereUT?, Parametre*, Condition?, Ops_Vis?,
                            Ops_Nav?, Ops_Coord ?)>
```

```
<!ATTLIST DescriptionDeVue Nom CDATA #REQUIRED>
<!ATTLIST DescriptionDeVue Type (Lineaire | Arborescente | Graphe) #REQUIRED>
<!ATTLIST DescriptionDeVue Contenu (Chaines | Annotations) #REQUIRED>
<!ELEMENT PremiereUT (Condition)>
<!ELEMENT Parametre (#PCDATA)>
<!ATTLIST Parametre Nom CDATA #IMPLIED>
<!ELEMENT Ops_Vis (MiseEnRelief | AideContextuelle)+>
<!ELEMENT Ops_Nav (Op_Nav+)>
<!ELEMENT Ops_Coord (Op_Coord+)>
```

Tableau 11.1 – *Fragment de la DTD du langage des connaissances.*

11.3. Le langage de conditions

En ce qui concerne le reste de la DTD, le cas spécifique des conditions, pour lesquelles une notation arborescente est proposée est remarquable. D'une part, cette notation est facilement exprimable en XML, dû à la nature hiérarchique du langage. D'autre part, celle-ci correspond à la notion d'arbre syntaxique [Aho *et al.* 1986] associé à une condition. Il faut souligner qu'il ne s'agit pas de l'arbre d'analyse syntaxique (résultant d'une analyse syntaxique de la condition, selon la grammaire du langage) mais d'une structure arborescente plus simple, qui condense l'information en indiquant les opérateurs comme nœuds internes et les opérandes comme feuilles.

Prenons comme exemple la condition suivante :

existeTitre OU (*estParent*(UT(Type = Phrase, *, *, *, *)) ET existeFils)

Cette condition vérifie si une UT contient un titre ou si elle comporte des UT filles et si elle est, en même temps, le parent d'une UT de type « Phrase ». L'arbre syntaxique correspondant à cette condition s'affiche ci-dessous.

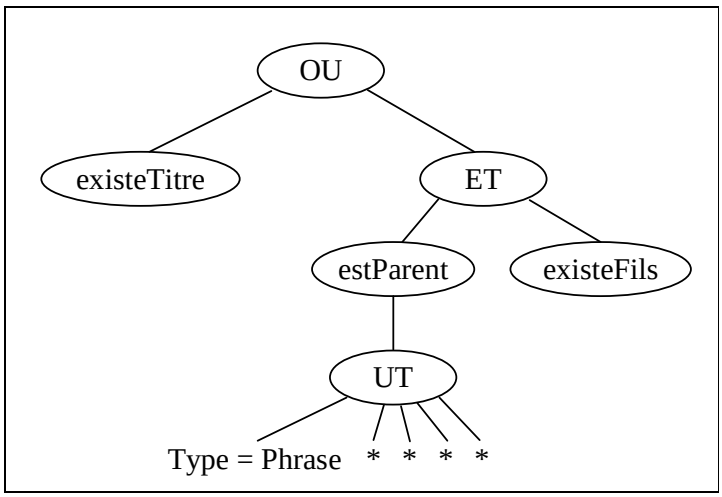

Figure 11.1 – *Exemple d'arbre syntaxique d'une condition.*

Le fragment de la DTD correspondant aux conditions est présenté ci-dessous.

```
<!ELEMENT Condition ((UT) |( Condition)+)>
   <!ATTLIST Condition Type (  Simple | existeAnnotations | existeChaîneLexicale |
                               existeTitre | existeParent | existeFils | estParent |
                               estFils | estAscendant | estDescendant |
                               contientDansTitre |  estDansTitreDe | ET | OU |
                               NON) #REQUIRED>
<!ELEMENT UT (Contrainte | ExisteAnnotation)*>
<!ELEMENT Contrainte EMPTY>
<!ATTLIST Contrainte arg1  CDATA  #REQUIRED
                     op  (= | ≠ | < | > | ≤ | ≥ | estPréfixe |  estSuffixe  | est
                           SousChaîne)[55]  #REQUIRED
                     arg2  CDATA  #REQUIRED>
<!ELEMENT ExisteAnnotation EMPTY>
```

[55] Il faut préciser que dans la DTD réelle, les opérateurs dénotés par des symboles sont remplacés, à cause des contraintes imposées par la syntaxe des DTD, par des opérateurs dénotés par des mots. Ainsi, par exemple, l'opérateur « = » devient l'opérateur « égal », etc.

```
<!ATTLIST ExisteAnnotation valeurExistence   (oui | non)   "oui"
                                nom  CDATA  #IMPLIED
                                valeur  CDATA  #IMPLIED>
```

Tableau 11.2 – *Fragment de la DTD correspondant au langage de conditions.*

En XML les différents opérateurs sont encodés comme des attributs nommés « Type » et quelques précisions sont nécessaires. D'abord, l'élément XML nommé UT ne représente pas une UT mais il s'agit de l'opérateur UT présenté au chapitre 8. Ensuite, une condition simple, exprimée par cet opérateur, n'a pas une structure de quintuple où les contraintes correspondant au type, au numéro, au rang, aux annotations et à la chaîne lexicale, respectivement, sont exprimées. Au contraire, les contraintes sur les attributs et les contraintes d'existence sur les annotations d'une UT correspondent à des éléments XML emboîtés dans l'élément UT. En conséquence, il n'y a pas d'ordre strict dans l'apparition des contraintes. De plus, le signe étoile n'est pas utilisé explicitement mais son utilisation est inférée de l'instance XML de la condition. Par exemple, si une condition simple, c'est-à-dire un élément UT, ne contient pas un élément *Contrainte* comportant « Rang » comme valeur pour l'attribut « arg1 » ou « arg2 », cela signifie qu'il n'existe pas de contrainte sur le rang, ce que le langage Sextant dénote en utilisant le signe étoile.

Enfin, il existe des restrictions ne pouvant pas être explicitées dans une DTD. Par exemple, si une condition est de type « Simple », alors elle doit comporter un seul élément XML, et cet élément ne peut être qu'un élément UT. Ou bien, si le type de la condition est « existeTitre », alors elle ne peut pas comporter d'élément XML. Ou bien, les contraintes d'une condition correspondant à l'opérateur UT ne doivent pas se répéter. Ce type de restrictions doit impérativement être contrôlée par le système interpréteur : cela correspond à une analyse sémantique d'une instance du langage, étape immédiatement suivante à celle d'analyse syntaxique.

En ce qui concerne les instances XML possibles d'une condition, en guise d'exemple, la condition dont l'arbre syntaxique est affiché à la figure 11.1, s'écrit en XML comme suit :

```
<Condition Type = "OU">
        <Condition Type = "existeTitre" />
```

```
<Condition Type = "AND">
    <Condition Type = "estParent" />
        <Condition Type = "Simple">
            <UT>
                <Contrainte   arg1 = "Type"
                              op = "="
                              arg2 = "Phrase" />
            </UT>
        </Condition>
    </Condition>
    <Condition Type = "existeFils" />
</Condition>
</Condition>
```

Tableau 11.3 – *Exemple d'une condition exprimée en XML.*

11.4. Un éditeur de modules de connaissances

Comme il arrive souvent, l'expression XML d'une construction quelconque du langage devient plus complexe et difficile à lire que dans le langage original. Ce type d'inconvénient peut être surmonté en utilisant des éditeurs spécifiques qui cachent à l'utilisateur la syntaxe concrète XML, en lui offrant des interfaces adaptées. Dans le cadre de cette thèse, un projet de DESS a été proposé à ces fins. Ce projet, réalisé par les étudiants Isabelle Ranque et Benoit Lamey, du DESS ILSI de l'Université Paris-Sorbonne, a abouti au développement d'un éditeur de modules de connaissances, qui a été de grande valeur tout au long du processus de création et test des modules utilisés par la plate-forme NaviTexte (*cf.* chapitre 12).

La figure 11.2 illustre le modèle général de l'écran principal du système. Dans le panneau situé à gauche, l'utilisateur définit les propriétés du module et il peut manipuler les différentes descriptions de vue inclues dans le module. Le panneau à droite contient tous les éléments d'une description de vue (en particulier, de celle sur laquelle l'utilisateur est positionné dans le panneau à gauche).

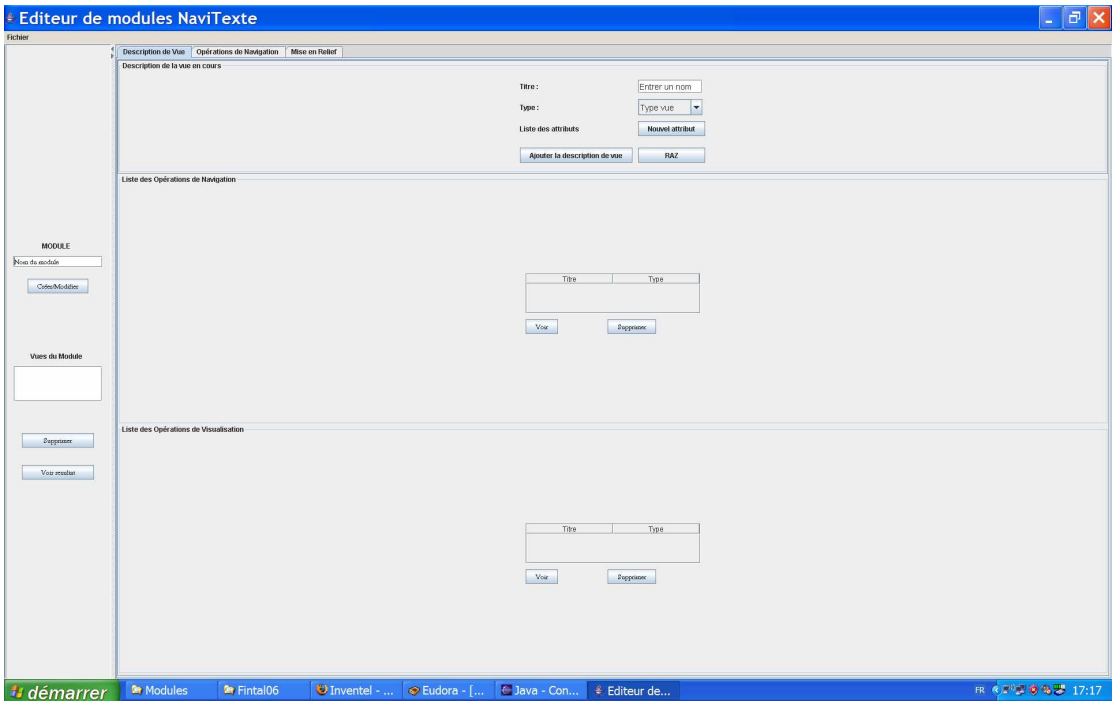

Figure 11.2 – *Écran principal de l'éditeur de modules de connaissances.*

Le panneau placé à droite est divisé en trois parties. La division supérieure contient les propriétés générales de la description de vue actuelle. La division au milieu contient toutes les opérations de navigation définies pour la description de vue. Enfin, la division inférieure contient les opérations de visualisation définies.

11.5. Synthèse

Une version XML du langage de modélisations de connaissances Sextant est définie. Le point fort de cette version réside dans la facilité, pour un système informatique, de manipuler les différentes instances du langage (des documents XML). Le point faible vient de la complexité des documents XML obtenus. Afin de pallier cette difficulté, un éditeur de modules de connaissances a été développé. Cet éditeur est actuellement opérationnel et il est utilisé dans le cadre de différentes applications de la plate-forme NaviTexte (*cf.* chapitre 11).

La DTD complète du langage de modélisation des connaissances Sextant se trouve à l'annexe V et plusieurs exemples de fichiers XML suivant cette représentation sont regroupés à l'annexe VI.

Chapitre 12

NaviTexte : une plate-forme de navigation textuelle

12.1. Objectifs de NaviTexte

De la même manière que l'efficacité de l'approche de l'hypertexte dépend de l'existence et de la qualité d'outils de lecture spécifiques (*i.e. browsers*), l'efficacité de la navigation textuelle est étroitement liée aux plates-formes logicielles implémentant ses concepts fondamentaux. Force est de constater qu'une approche « bonne à priori » peut échouer, faute d'outils concrets l'implémentant de manière adéquate. Pour n'en prendre qu'un exemple, on attend depuis quelques années un *ebook* capable d'être accepté par le grand public. Bien que le concept du livre électronique semble prometteur, le succès des premiers prototypes a été particulièrement mitigé, notamment à cause des problèmes que soulèvent la lecture sur ce type de support, tels que la typographie, la mise en forme ou l'ergonomie du dispositif [Baccino 2004].

Toutes proportions gardées, j'estime que le succès futur de notre approche dépendra du développement d'une plate-forme capable d'offrir aux utilisateurs les différents outils proposés par la navigation textuelle, d'une manière « acceptable » et « vraiment utile ».

NaviTexte est la première plate-forme logicielle implémentant notre approche. Cette plate-forme correspond au système interpréteur dont une partie de la sémantique est donnée dans le chapitre 9. Son implémentation comporte trois volets :

- l'implémentation informatique de la représentation des textes proposée au chapitre 6 ;

- l'implémentation informatique de *Sextant*, le langage de modélisation des connaissances proposé au chapitre 8 ;

- la conception et développement d'un environnement capable de traiter les textes, d'interpréter le langage Sexant et de gérer l'interaction avec l'utilisateur, en s'appuyant sur les réflexions du chapitre 2, concernant les IHM.

L'objectif de NaviTexte étant le développement d'un premier prototype opérationnel capable de tester les idées centrales de notre approche (*i.e.* un prototype à utiliser pour construire des applications concrètes et évaluer celles-ci), quelques restrictions ont été imposées. La stratégie sous-jacente à ces restrictions a été fixée en prenant en considérations les développements réalisés préalablement, notamment pour la plate-forme ContextO et pour le projet RÉGAL (*cf.* chapitre 1). Au moment d'implémenter NaviTexte, nous avons considéré comme secondaire le développement d'outils déjà développés dans ContextO, dont nous connaissons l'utilité vis-à-vis des utilisateurs, et que nous pourrions incorporer ultérieurement à NaviTexte. De ce fait, la vue de type arborescent, les opérations de coordination et la diversité d'opérations de mises en forme ont été mises en second plan, mettant l'accent sur les aspects de navigation tout en offrant des opérations basiques de colorisation[56].

12.2. Architecture conceptuelle et logicielle

12.2.1. Architecture conceptuelle

D'un point de vue conceptuel, la plate-forme NaviTexte est constituée par trois sous-systèmes principaux[57] (*cf.* figure 12.1) : le *système de gestion des textes*, le *système de gestion des modules*, et le *système de gestion des interactions*.

Le *système de gestion des textes* se charge d'offrir toutes les fonctionnalités concernant la représentation informatique du texte proposée au chapitre 6. Rappelons que d'après celle-ci, les textes sont encodés sous forme de fichiers XML, selon une DTD définie. Le chargement d'un fichier par le système entraîne la création en mémoire d'une instance du texte selon une représentation interne. Cette représentation prend en considération (*i.e.* interprète) des éléments XML ayant une sémantique particulière vis-à-vis du texte, tels que les éléments UTP ou les constructeurs déclarés dans la partie *tête* (*cf.* chapitre 6). De même pour le traitement des titres. C'est ce sous-système qui offre les fonctions nécessaires afin de parcourir la structure d'un texte : hiérarchie d'UT dans le corps, les UT « construites » dans la partie *tête*, les titres, etc.

Le *système de gestion des modules* permet de charger les modules de connaissances en mémoire et de manipuler ceux-ci. Pour ce faire, une représentation des modules en mémoire a

[56] Parmi les différentes opérations de visualisation (*cf.* chapitres 7 et 8), tels la définition de la taille, l'ajout de texte, les aides contextuelles, etc., seule la colorisation a été considérée.

[57] La notation UML est utilisée pour présenter les modèles statiques et dynamiques.

été définie, équivalente à celle d'un arbre, présentée au chapitre 8. La représentation en mémoire tient compte des différents éléments du langage et c'est ce sous-système qui s'occupe, entre autres, d'interpréter le langage de connaissances, y compris le langage de conditions (*cf.* chapitre 8).

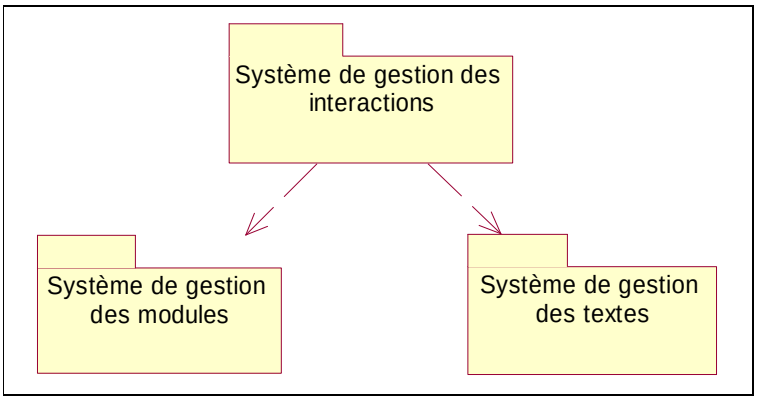

Figure 12.1 – *Architecture conceptuelle de NaviTexte.*

Quant au *système de gestion des interactions*, conceptuellement celui-ci se charge de tout ce qui concerne l'interaction avec l'utilisateur, et agit comme une interface entre les deux sous-systèmes précédents. En guise d'exemple, ce sous-système est chargé d'offrir les différentes opérations de navigation sur une UT (lorsque l'utilisateur clique sur celle-ci), de capturer le choix de l'utilisateur, de faire exécuter l'opération, d'appliquer le résultat de cette exécution (*i.e.* changer éventuellement d'UT active, modifier l'historique de navigation…), etc.

12.2.2. Architecture logicielle

A partir de l'architecture conceptuelle présentée, plusieurs architectures logicielles sont possibles. Nous en avons défini une (*cf.* figure 12.2) en nous inspirant de l'architecture Model-View-Controller (MVC) [Burbeck 1987]. Dans l'approche MVC, un modèle de données, les différentes vues de ce modèle et les contrôleurs (objets chargés de coordonner l'interaction entre les vues et le modèle) sont des entités séparées. Dans notre cas le texte joue le rôle du modèle, la vue du texte correspond au concept de *vue* et la partie du contrôleur correspond à la plate-forme logicielle.

La plate-forme est organisée en cinq packages principaux : *Vues*, *Textes*, *Modules* (ayant deux sous-packages : *Navigation* et *Visualisation*), *Configuration* et *Frames*.

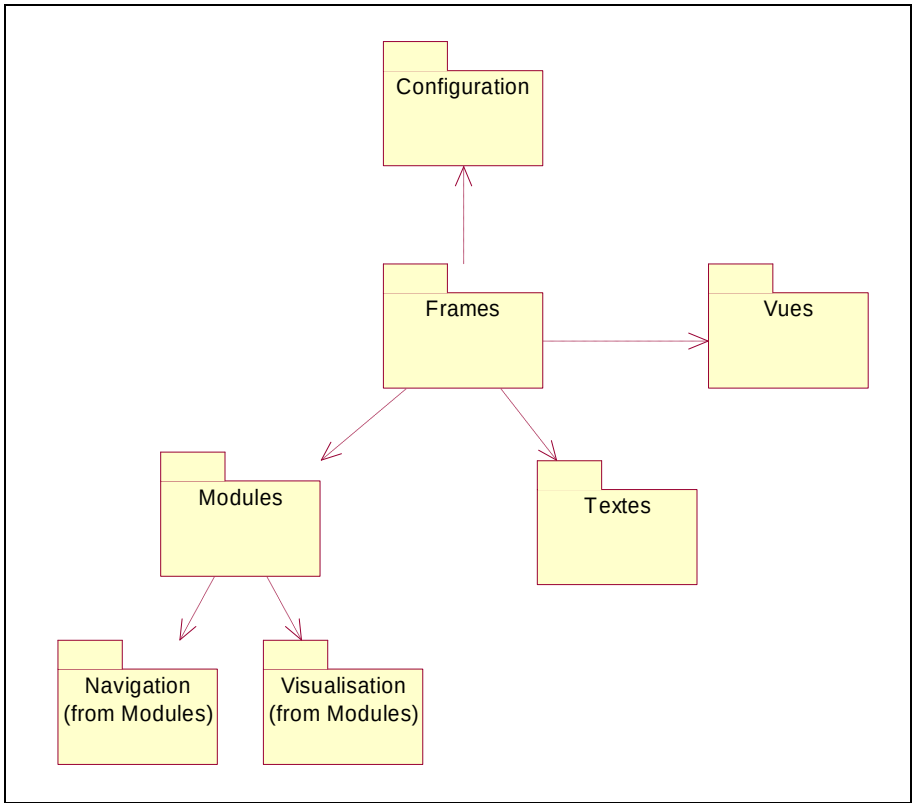

Figure 12.2 – *Architecture logicielle de NaviTexte.*

NaviTexte étant implémenté en Java, une architecture orientée objets a été prévue dès le départ. Cela atténue le choix d'avoir le contrôleur « appartenant à la plate-forme ». De fait, chaque classe contient toutes les connaissances nécessaires pour que ses instances (objets) puissent se gérer, et offre des méthodes spécifiques aux autres classes voulant interagir avec elle, cachant ainsi sa propre implémentation interne.

12.2.3. Les packages principaux : *Vues*, *Textes* et *Modules*

Les packages *Configuration* et *Frames* (*cf.* figure 12.2) sont, d'un point de vue conceptuel, assez simples : le premier contient les classes permettant la configuration de la plate-forme, et le deuxième contient les classes correspondant aux fenêtres du système. Il est intéressant d'analyser la conception des packages *Vues*, *Textes* et *Modules*, cœur de la plate-forme. Cela permettra d'aborder par la suite les aspects généraux de la conception de la plate-forme, notamment ceux qui concernent la création d'une vue et la gestion des évènements.

12.2.3.1. Le package Textes

Dans le package *Textes* (*cf.* figure 12.3) sont regroupées les classes ayant trait au chargement et à la gestion des textes. L'implémentation informatique est légèrement différente du modèle proposé au chapitre 10 (*cf.* figure 10.1) : une seule classe pour les UT a été définie ; le titre, le corps et la tête ont été définis comme des champs de la classe *Texte*, au lieu de les définir comme des classes séparées. La classe *UT_ID* correspond à l'identifiant d'une UT. Elle est utilisée à la fois pour identifier les UT et pour les référencer dans les éléments construits dans la tête du texte.

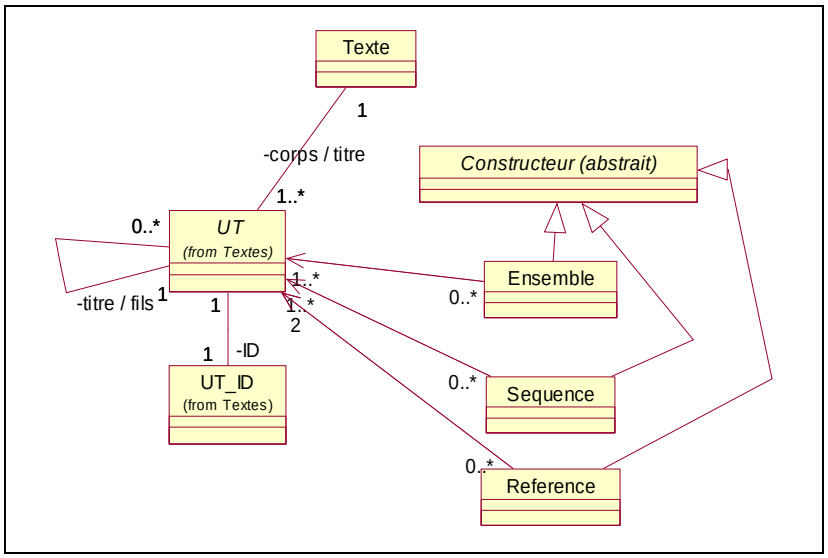

Figure 12.3 – *Package Textes.*

La classe *Texte* représente un texte, et ses méthodes principales sont :

- le constructeur *Texte* : charge un texte à partir d'un nom de fichier ;

- *getTitre* : renvoie une hiérarchie d'UT correspondant au titre du texte ;

- *getDFSUTs* : renvoie un vecteur d'UT correspondant à un parcours en profondeur préfixé dans la hiérarchie d'UT ;

- *getUTsPremierNiveau* : renvoie un vecteur d'UT correspondant aux UT de premier niveau dans la hiérarchie d'UT ;

- *getUTFeuilles* : renvoie un vecteur d'UT contenant les UT feuilles, selon un parcours en profondeur préfixé dans la hiérarchie d'UT ;

- *getEnsembles*, *getSequences*, *getReferences*[58] : renvoient les unités textuelles complexes exprimées dans la tête du texte ;

Il en résulte que la classe *Texte* prend en charge le chargement du texte et offre les moyens d'accéder à ses différents constituants.

12.2.3.2. Le package Modules

Les classes concernant le chargement et la gestion des modules sont regroupées dans le package *Modules* (*cf.* figure 12.4). La classe *Module* contient essentiellement un ensemble d'instances de la classe *DescriptionDeVue*, chargés par son constructeur à partir d'un nom de fichier. Les méthodes principales de la classe *DescriptionDeVue* sont :

- *creerVue* : crée une vue à partir d'un texte (*cf.* section XXX) ;

- *getOpsNav* : renvoie les opérations de navigation disponibles pour une UT ;

- *hasOpsNav* : indique si une UT dispose d'opérations de navigation.

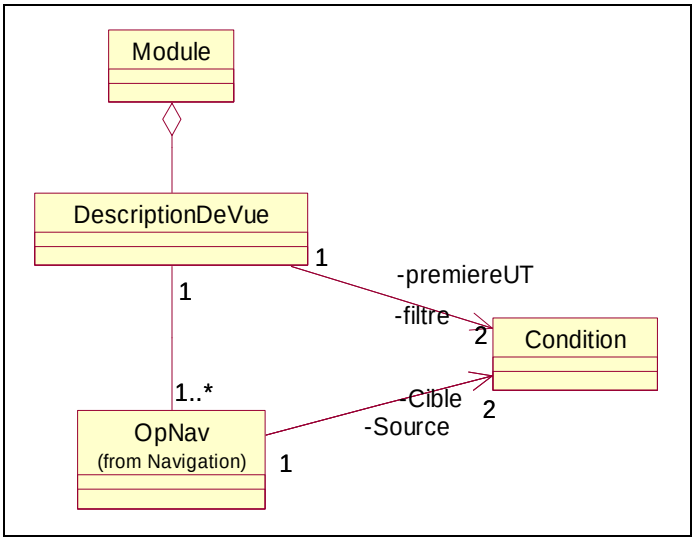

Figure 12.4 – *Package Modules et sous-package Navigation.*

La classe *Condition* offre une méthode nommée *vérifie*, qui vérifie si une UT déterminée vérifie la condition. C'est la classe qui implémente le langage de conditions (*cf.* chapitre 8). Rappelons que les conditions complexes sont formées à la base de conditions simples. Puis-

[58] Le cas des graphes a été écarté dans cette implémentation de NaviTexte.

qu'une condition simple est exprimée en fonctions des informations d'une UT, la méthode *vérifie* fait appel à la méthode *verifiePatron* de la classe *UT* (package *Textes*) lorsque la condition est de type simple.

Le sous-package *Navigation* contient la classe *OpNav*, qui correspond à la notion d'opération de navigation. Les méthodes offertes par cette classe sont :

- *estDisponible* : pour une UT déterminée, indique si l'opération de navigation est disponible (*i.e.* si la condition de la cible est vérifiée) ;

- *appliquer* : renvoie l'UT résultante de l'application de l'opération de navigation sur une UT déterminée ;

En ce qui concerne le sous-package *Visualisation*, il a été conçu mais partiellement implémenté. Sa conception (*cf.* figure 12.5) correspond exactement aux opérations de visualisation présentées dans le chapitre 8.

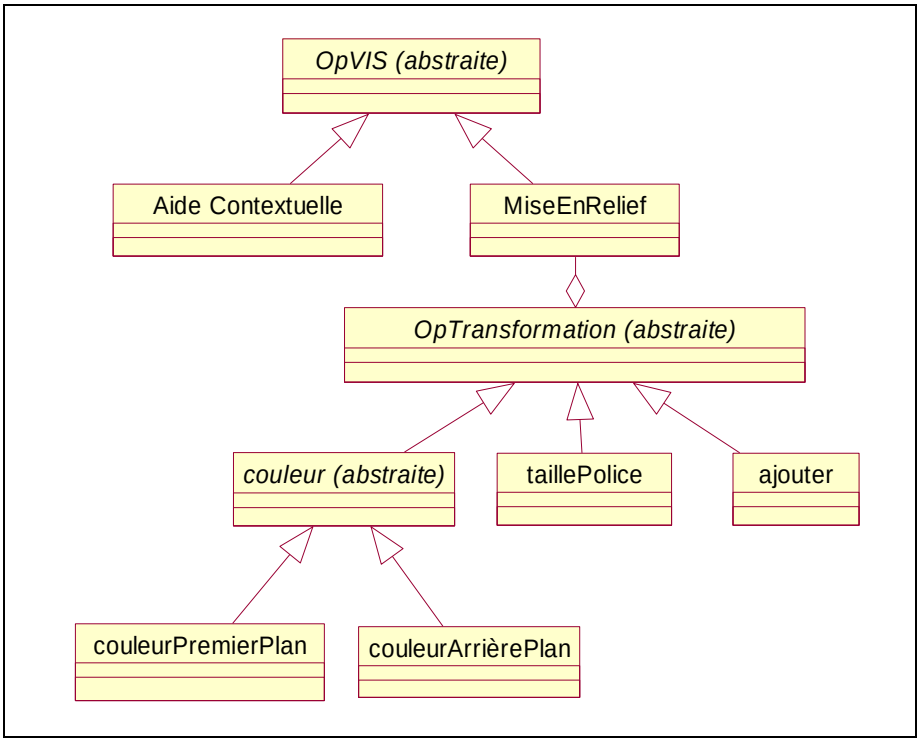

Figure 12.5 – *Sous-package Visualisation.*

La classe abstraite *OpVis* représente une opération de visualisation. Cette classe est étendue par deux classes : *MiseEnRelief* et *AideContextuelle*. Une mise en relief est constituée par

plusieurs opérations de transformation, représentées par la classe abstraite *OpTransformation*. Les classes *couleurPremierPlan*, *couleurArrierePlan*, *taillePolice* et *ajouter* correspondent aux opérations de transformation concrètes proposées par le langage Sextant.

12.2.3.3. Le package Vues

Le package *Vues* (*cf.* figure 12.6) est composé de la classe abstraite *Vue*, étendue par la classe *VueTextePlat*. Celle-ci utilise un *JTextPane* comme élément visuel et la classe *InfoUTs*, chargée de coordonner les UT avec leur représentation graphique dans le *JTextPane*. Enfin, L'interface nommée *VueListener*, définisse les méthodes à implémenter pour un écouteur de vues.

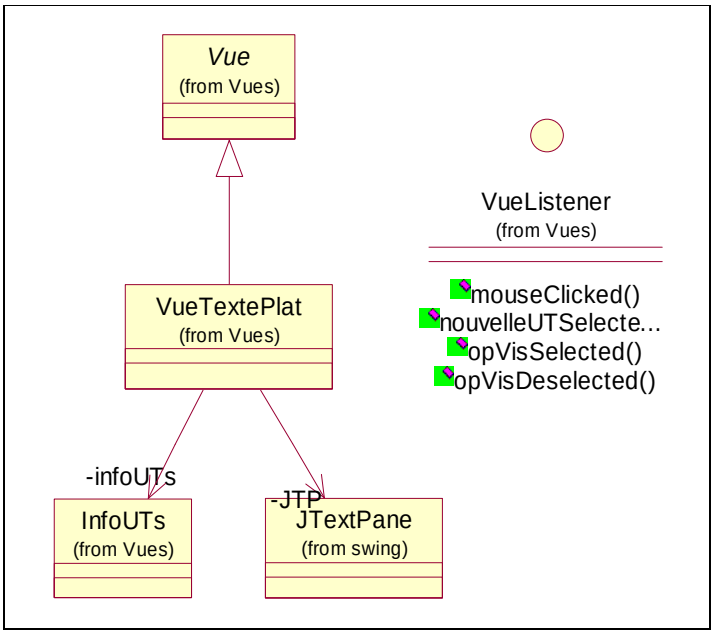

Figure 12.6 – *Package Vues.*

12.2.4. Conception générale de la plate-forme logicielle

Comme il a été souligné dans la section précédente, un très grand soin a été apporté à la conception générale : des classes abstraites ont été définies, des évènements spécifiques à certaines classes ont été définis sous forme d'interfaces (au sens Java), différentes couches concernant la complexité des interfaces graphiques ont été implémentées, un sous système de gestion d'erreurs a été développé, et la plate-forme est raisonnablement configurable.

En ce qui concerne les classes abstraites, je prends l'exemple des vues, pour lesquelles une classe abstraite nommée *Vue* existe, comportant les données et méthodes communes à toutes les vues : nom de la vue, clés du texte et du module à partir desquels celle-ci a été créée, accésseurs de l'UT active, insertion d'un titre ou d'une UT (*cf.* figure 12.7), etc. Ensuite une classe *VueTextePlat*, étendant la classe abstraite *Vue*, implémente le cas particulier d'une vue de type *texte plat.* C'est cette classe qui connaît effectivement l'implémentation interne de ce type de vue (par exemple, pour afficher le texte, la classe *JTextPane* du package *Swing* est utilisée). En conséquence, c'est celle-ci qui implémente réellement, par exemple, l'opération « insérer titre » à partir d'un titre sous forme de hiérarchie d'UT. Cela implique plusieurs choses :

- parcourir la hiérarchie des UT (pour ce faire, une classe *UT* appartenant au package *Textes* existe, offrant les méthodes nécessaires) ;

- demander, pour chaque UT, la chaîne lexicale correspondante ;

- insérer dans le *JTextPane* la chaîne de caractères de l'UT ;

- puisque le *JTextPane* travaille avec des segments de texte, au sens de segments linéaires de caractères, l'insertion d'une UT sauvegarde dans une structure de données spécifique (interne à la vue texte plat) la position initiale (dans le *JTextPane*) à laquelle cette UT se trouve, afin de pouvoir manipuler cette UT postérieurement ; cela signifie qu'il existe un « *mapping* » entre les UT et leur représentation visuelle (*cf.* section 9.3).

12.2.5. Création d'une vue

La figure 12.7 montre un diagramme UML de séquence décrivant la dynamique générale de traitement d'un texte qui se conclue notamment par la création d'une vue à partir de ce texte et d'une description de vue[59].

Rappelons que la création d'une vue est réalisée à partir d'un texte et d'une description de vue. Dans NaviTexte, le chargement d'un texte et le chargement d'un module de connaissances, déclenchent un processus automatique de création des vues : pour chaque texte et pour chaque description de vue existant dans chaque module, une vue est créée.

[59] De fait, autant de vues que de descriptions de vues dans le module de connaissances sont créées pour ce texte.

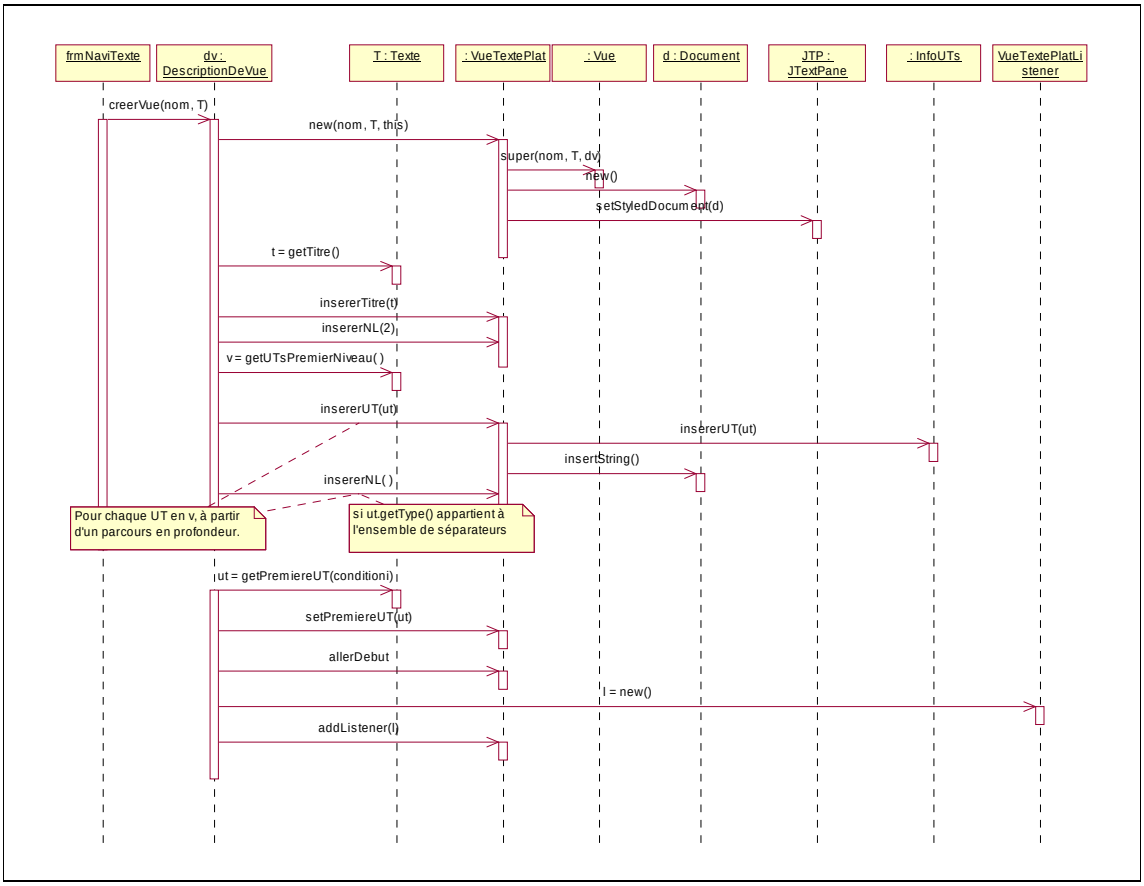

Figure 12.7 – *Création d'une vue de type texte plat.*

12.2.6. Gestion des évènements

Quant aux évènements spécifiques à certaines classes, prenons encore une fois l'exemple des vues. L'un des évènements définis pour une vue s'appelle *mouseClicked* indiquant que l'utilisateur a cliqué sur la vue avec la souris. Tous les évènements sont définis dans une interface Java nommée *VueListener* et la classe abstraite *Vue* offre une méthode nommée *addListener* permettant d'insérer un écouteur pour une vue déterminée. Cela signifie qu'une vue, par défaut, ne fait rien de spécifique lorsque l'utilisateur fait un clic sur elle. La seule action accomplie consiste à notifier à tous les écouteurs de cette vue que c'est sur celle-ci que l'utilisateur a fait un clic (en transmettant tous les information pertinentes). C'est, conceptuellement, le *système de gestion des interactions* qui interprète ce clic et qui en tant que contrôleur, au sens MVC, se charge d'exécuter les actions correspondantes. La figure 12.8 montre un diagramme UML de séquence décrivant la gestion d'un clic sur une vue de type texte plat.

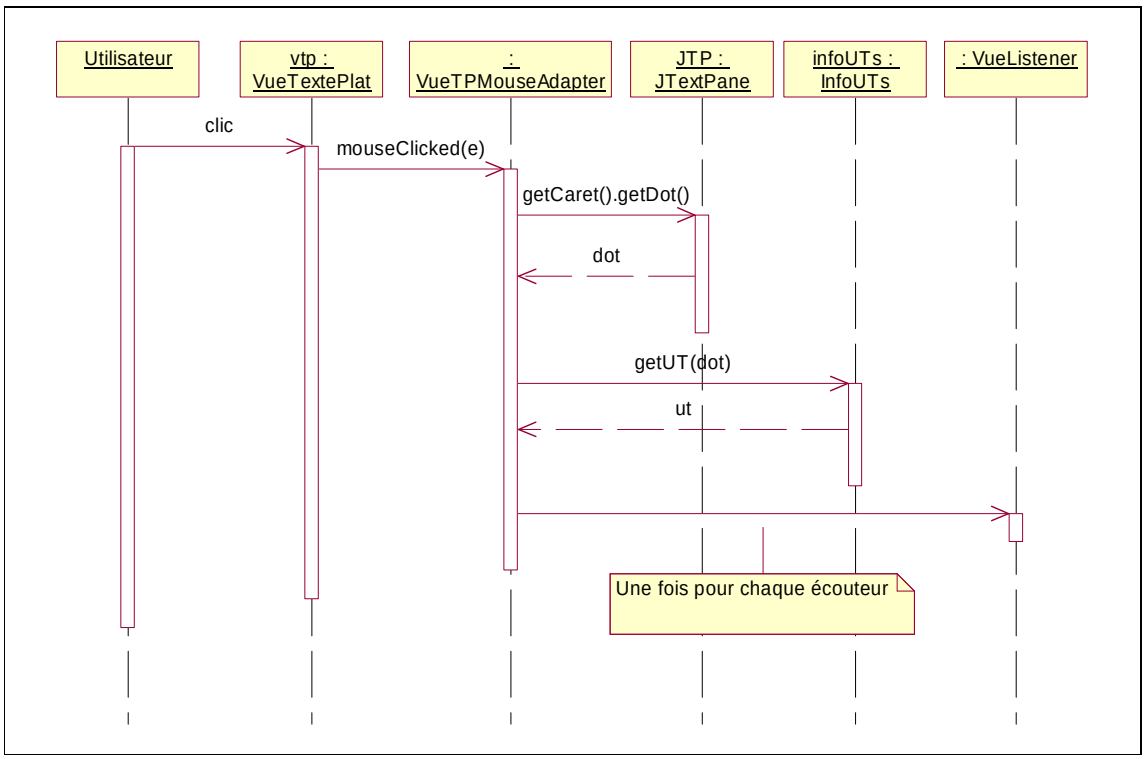

Figure 12.8 – *Gestion d'un clic sur une vue de type texte plat.*

Dans ce diagramme nous pouvons observer que le clic sur la vue déclenche une opération qui demande la position dans le *JTextPane* à laquelle le clic vient de se produire. A partir de cette position, un appel à un objet de la classe *InfoUTs* traduit cette position en termes d'UT cliquée. Enfin, pour chaque écouteur (inséré préalablement, par la méthode *addListener*, dans un tableau d'écouteurs), la méthode *mouseClicked* est déclenchée, en lui passant les arguments pertinents : la vue et l'UT sur laquelle le clic s'est produit et les coordonnées dans l'écran. Notons que, pour être précis, le clic est fait sur la représentation visuelle d'une UT. C'est pourquoi l'information sur la vue est importante : une même UT peut appartenir à plusieurs vues (*i.e.* peut avoir plusieurs représentations visuelles, une par vue la contenant).

A différents niveaux de la plate-forme, plusieurs écouteurs sont définis pour une vue, avec des objectifs différents : capturer si l'UT active a changée afin de modifier l'historique, capturer le clic sur une UT afin de déclencher les opérations de navigation disponibles… Je prend comme exemple ce dernier point, pour illustrer comment les opérations de navigation sont proposées à l'utilisateur. Au moment de créer une vue, un écouteur est créé et inséré dans la vue (*cf.* figure 12.7). Cet écouteur (une classe implémentant l'interface *VueListener*)

implémente la méthode nommée *mouseClicked*, déclenchée par le clic de la souris. L'instance de cette classe demande à la description de vue pertinente (*i.e.* celle à partir de laquelle la vue a été créée) les opérations de navigation disponibles pour l'UT en question (*cf.* figure 12.8). Ce point est important car seules les opérations de navigation dont cette UT vérifie la condition de la cible doivent être proposées à l'utilisateur. Une boucle est exécutée (*cf.* les critères d'optimisation de la performance), sauvegardant dans un tableau les opérations de navigation disponibles pour l'UT (appel à la méthode *estDisponible* de la classe *OpNav*, qui appelle à la méthode *vérifie* de la classe *Condition*). Le tableau étant construit, celui-ci est utilisé pour créer une instance de la classe *PaletteNavigation*.

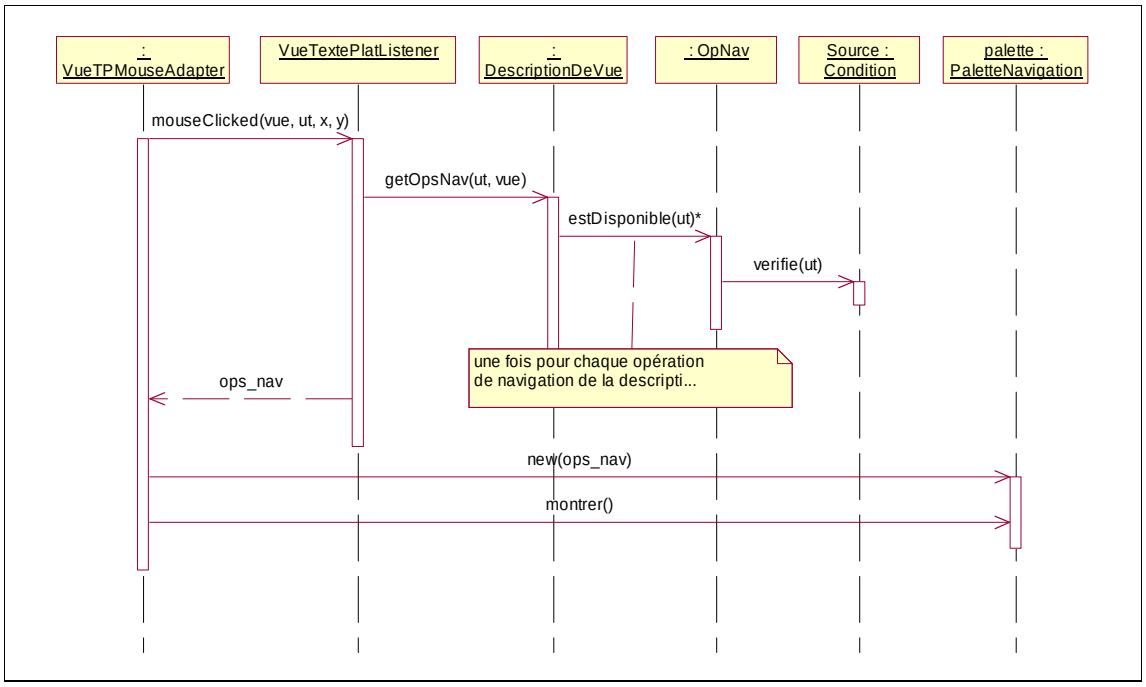

Figure 12.8 – *Proposition des opérations de navigation.*

12.2.7. Structures de données et algorithmes de parcours

En plus de la conception générale divisée en classes spécialisées, des structures de données spécifiques, notamment des dictionnaires (tables de *hashcode* java), ont été utilisées afin d'améliorer la performance du système. Bien que les textes et les modules des connaissances soient encodés sous forme de fichiers XML, lorsque ceux-ci sont chargés en mémoire, le système analyse (*parse)* le fichier correspondant mais il ne se limite pas à garder comme résultat un seul objet représentant le résultat de ce processus de *parsing* (par exemple un objet DOM ou JDOM contenant l'arbre d'analyse syntaxique sous forme d'éléments reliés entre eux). Au

contraire, plusieurs structures auxiliaires sont créées. Prenons le cas des textes. Le chargement d'un texte entraîne la création d'une structure comportant toutes les UT de haut niveau[60] appartenant au *corps*. De même, une structure représentant un parcours en profondeur de la hiérarchie est créée afin de simplifier l'exécution des opérations de navigation, où la notion de distance (*cf.* chapitre 9) est utilisée. D'autres structures concernant les nouvelles UT exprimées dans la *tête* sont également créées. Enfin, tout ce qui a trait à l'héritage des annotations est géré en utilisant des structures de données auxiliaires.

Un deuxième critère d'optimisation concerne les opérations de navigation. Étant donné que ni le texte ni le module de connaissances changent après la vue créée (ou bien si ceux-ci changent, ces changements ne sont pas répercutés sur les vues de manière dynamique tant que l'utilisateur ne redemande pas le chargement du module), l'ensemble du parcours reste figé au moment de la création d'une vue. Il en résulte que le système pourrait calculer, avant de présenter les vues à l'utilisateur, le résultat de l'application de toutes les opérations disponibles pour chaque UT appartenant à la vue. Néanmoins, ce calcul peut devenir complexe et potentiellement non nécessaire car il se peut que l'utilisateur n'exécute qu'un nombre très réduit de toutes les combinaisons possibles. D'autre part, le fait de calculer à chaque fois le résultat d'une opération de navigation devient redondant et non optimal (notons que comme les vues ne changent pas, pour une UT déterminée le résultat d'une opération de navigation reste toujours le même).

Afin d'optimiser ce calcul, un système de trace de résultats, s'inspirant des techniques de programmation dites paresseuses (*lazy)*, a été mis en place. Ce système commence avec une structure de données vide. Lorsque l'utilisateur clique sur une UT (*cf.* figure 12.7), les opérations de navigation disponibles sont calculées (seulement les opérations et non pas les résultats) et sauvegardées par ce système. Ce faisant, la boucle sur toutes les opérations de navigation exprimées dans une description de vue est réalisée, au maximum, une seule fois par UT. Lorsque l'utilisateur sélectionne une opération de navigation, le résultat (i.e. l'UT cible) est calculé et sauvegardé par le système de trace, évitant de la recalculer postérieurement. Au pire (*i.e.* lorsque l'utilisateur exécute, au moins une fois, toutes les opérations de navigation disponibles pour toutes les UT d'une vue), le système de traces crée une structure qui contient toutes les combinaisons, mais le temps total de calcul sera distribué au fil de l'utilisation et il passera inaperçu pour l'utilisateur.

[60] C'est-à-dire les UT dont le parent est Texte.

12.3. Composants visuels de NaviTexte

Les concepts et techniques présentés au chapitre 2 ont été appliqués afin de concevoir et développer la plate-forme. La figure 12.9 illustre le modèle général de l'écran principal de NaviTexte. Celle-ci est divisée en trois panneaux, une barre de menus, une barre d'outils et une barre d'état.

Le panneau placé à gauche en haut contient une vue condensée du texte où les éléments correspondent aux annotations (*cf.* section 12.3.3). Le panneau placé à gauche en bas contient l'information des textes et des modules de connaissances ouverts, ainsi que l'information des vues créées. Le panneau placé à droite contient les différentes vues des textes (*cf.* section 12.3.3).

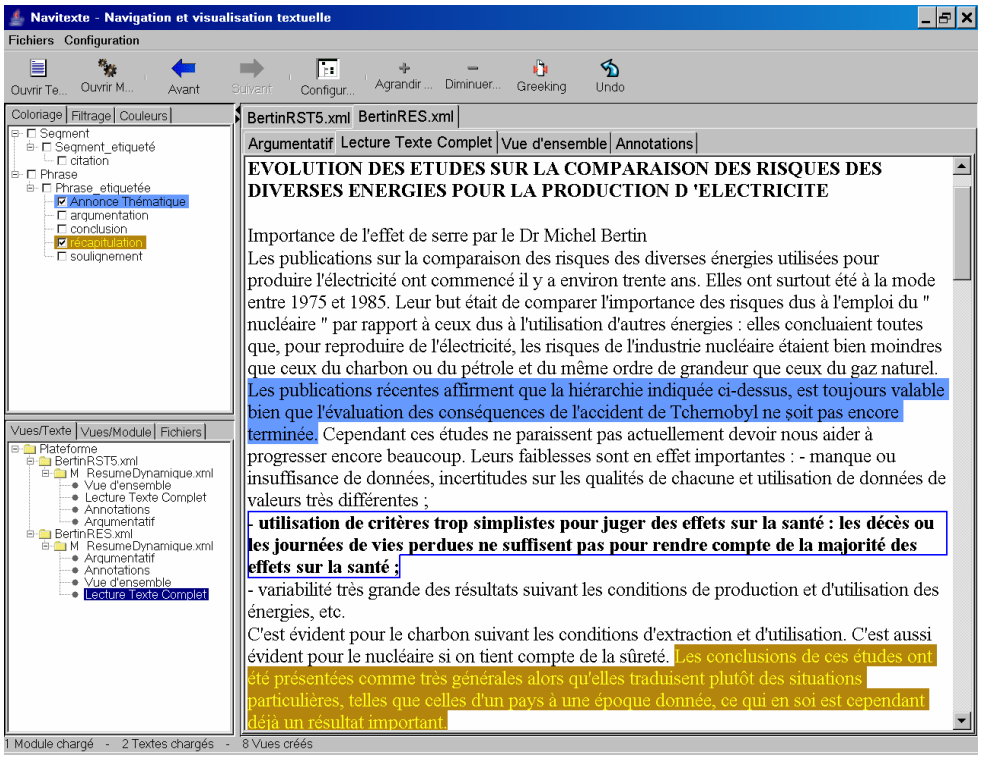

Figure 12.9 – *Écran principal de NaviTexte.*

En ce qui concerne la barre d'outils, celle-ci contient les opérations plus utilisées de la plate-forme :

- *Ouvrir Texte* et *Ouvrir Module* : ces opérations permettent de charger respectivement un texte et un module ; les répertoires par défaut sont configurables (*cf.* section 12.3.7) ; de plus, le système permet de définir des modules de connaissances à charger par défaut lorsque la plate-forme se lance ;

- *Avant* et *Suivant* : ces opérations permettent de naviguer à travers de l'historique de navigation (*cf.* section 12.3.6) ;

- *Configuration* : cette opération montre la fenêtre de configuration du système (*cf.* section 12.3.7) ;

- *Agrandir Police* et *Diminuer Police* : ces opérations permettent de modifier la taille de la police de la vue actuelle ;

- *Greeking* : cette opération modifie automatiquement la mise en forme d'une vue afin d'obtenir une vue « *greekée* » du texte (*cf.* section 12.3.3.1) ;

- *Undo* : cette opération défait la dernière opération exécutée.

Enfin, la barre d'état indique le nombre de modules et de textes chargés, aussi que le nombre de vues créées.

Je développe par la suite la dynamique d'exécution et la sémiotique définis par la plate-forme, les différents composants visuels et d'autres éléments tels que l'historique de navigation et les possibilités de configuration.

12.3.1. Dynamique d'exécution

Lorsque l'utilisateur déclenche l'exécution de la plate-forme NaviTexte, celle-ci charge les modules de connaissances par défaut (*cf.* section 12.3.7) et attend que l'utilisateur charge des textes et, éventuellement, des modules.

Dans l'exemple de la figure 12.9, la barre d'état montre que l'utilisateur a déjà chargé deux textes : *BertinRES.xml* et *BertinRST5.xml*, qui s'affichent, d'une part, dans le panneau en bas à gauche (panneau de fichiers ouverts et de vues créées) ; d'autre part, pour chaque texte il existe un onglet disponible libellé avec le nom du fichier, faisant partie du panneau à onglets principal. Chaque onglet contient à la fois un panneau à onglets où s'affichent les différentes vues du texte pour un module en particulier, à l'occasion le module *M_ResumeDynamique.xml*.

L'utilisateur peut ouvrir et fermer des textes et des modules au fur et à mesure que l'utilisation du système se développe. Il existe une option de rafraîchissement pour les mo-

dules (et pour les textes), simulant la fermeture de tous les textes (ou modules) ouverts et leur postérieur réouverture. Cette option est particulièrement utile dans de situations d'encodage de connaissances, afin de faire des tests progressifs.

12.3.2. Sémiotique de la plate-forme

Quelques signes spécifiques, liés au phénomène de navigation textuelle, sont offerts à l'utilisateur.

12.3.2.1. UT sélectionnée

Il existe toujours une UT sélectionnée, affichée en gras et encadrée d'un filet bleu. Dans l'exemple de la figure 12.9, l'UT sélectionnée est « - *utilisation de critères trop simplistes…* ». La sélection d'une UT peut se réaliser de trois manières :

- au moment de créer la vue : le système calcule, à partir de la description de vue, quelle UT sera la première UT sélectionnée ;

- par sélection directe : l'utilisateur clique sur l'objet visuel correspondant à l'UT (par exemple la chaîne lexicale dans une vue de type texte plat) ;

- comme résultat de l'exécution d'une opération de navigation.

La sélection d'une UT modifie l'historique de navigation (*cf.* section 12.3.6).

12.3.2.2. Souris sémantique

Dans une vue déterminée, une UT peut avoir trois états vis-à-vis de la navigation textuelle :

- il s'agit d'une UT sans annotations et, en conséquence, sans opérations de navigation déclenchables ;

- il s'agit d'une UT avec annotations mais qui ne vérifie pas la condition d'aucune des opérations de navigation décrites dans la description de vue correspondant à la vue ;

- il s'agit d'une UT avec opérations de navigation déclenchables (*i.e.* une UT pour laquelle au moins une condition des sources des opérations de navigation se vérifie).

La *souris sémantique* est une option configurable explicitant ces trois états possibles. Au survol de la souris, lorsque l'utilisateur se positionne sur l'objet représentant une UT, le caret de la souris change selon l'état de l'UT.

Figure 12.10 – *Différentes valeurs de la souris sémantique.*

12.3.2.3. Distinctifs des UT selon leur état

Il s'agit d'une option configurable similaire à la souris sémantique. La police dans laquelle une UT s'affiche varie selon l'état de l'UT (*cf.* figure 12.11).

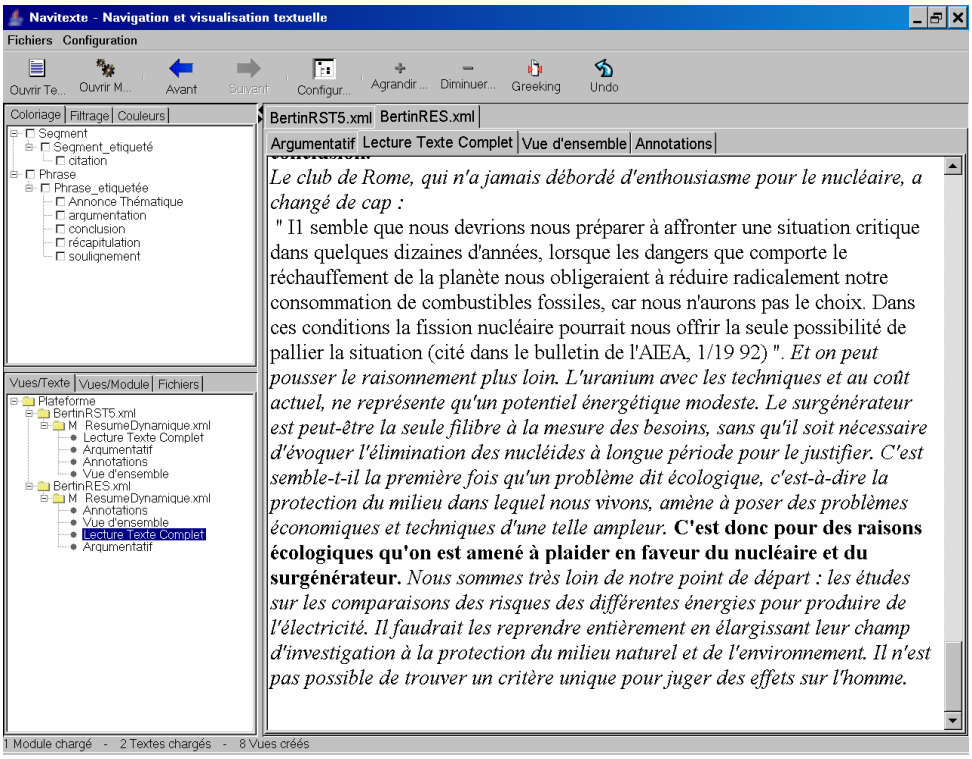

Figure 12.11 – *Affichage des UT selon leurs annotations et opérations de navigation disponibles.*

Nous faisons l'hypothèse suivante : une UT sans annotations est moins important qu'une UT avec annotations. En plus, une UT avec opérations de navigation est plus important qu'une UT avec annotations et sans opérations de navigation.

Aussi, s'il s'agit d'une UT sans annotations, la police est italique (puisqu'elle est plus difficile à lire). Dans le cas d'une UT avec annotations mais sans opérations de navigation, la police utilisée est normale. Enfin, les UT comportant des opérations de navigation sont mises en gras afin d'attirer l'attention du lecteur.

12.3.2.4. Trace des UT déjà « naviguées »

Optionnellement, l'utilisateur peut configurer la plate-forme de manière à ce que toutes les UT qui sont sélectionnées au fur et à mesure que la navigation se développe, changent de couleur. Ce faisant, l'utilisateur peut visualiser les UT qui ont été déjà « naviguées ». Cette option est particulièrement utile dans l'application de NaviTexte à l'apprentissage du français en tant que langue étrangère (*cf.* chapitre 13).

12.3.2.5. Distinctifs des opérations de navigation

Il existe trois états possibles pour les opérations de navigation :

- l'opération n'est pas exécutable ; c'est-à-dire que bien que l'UT vérifie la condition de la source de l'opération, il n'existe pas d'UT vérifiant la condition de la cible ;

- l'opération est exécutable et elle n'a pas été exécutée ;

- l'opération est exécutable et elle a été déjà exécutée.

Lorsque l'utilisateur clique sur une UT ayant des opérations de navigation déclenchables (*i.e.* une UT qui vérifie la condition de, au moins, une source de toutes les opérations de navigation définies pour la vue), un menu est offert à l'utilisateur, affichant les opérations de navigation associées à l'UT actuelle (*cf.* figure 12.12).

La mise en forme de chaque option correspondant à une opération de navigation indique son état. Dans l'exemple de la figure précédente, la première option, libellée « Lire_Thématique », est grisée car elle n'est pas exécutable. La deuxième et quatrième options, libellées respectivement « Lire_Récapitulation » et « Lire_Conclusion », s'affichent en gras car elles sont exécutables et n'ont jamais été exécutées. Enfin, la troisième option, libellée « Lire_Argumentation » et affichée en police normale, s'agit d'une opération de navigation exécutable qui a déjà été exécutée.

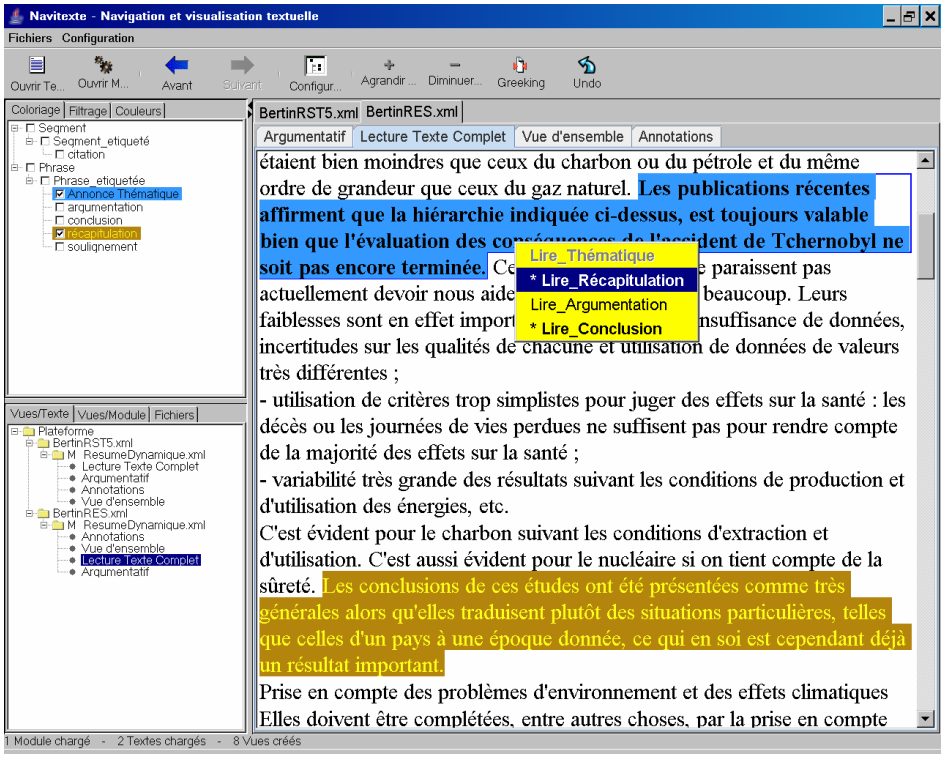

Figure 12.12 – *Distinctifs des opérations de navigation.*

12.3.3. Vues existantes

Rappelons les différentes vues graphiques d'un texte présentées à la section 7.2.1. Celles-ci sont classifiées selon leur *type* et leur *contenu*. Les types possibles sont : *linéaire (texte plat)*, *arborescente* et *graphe* tandis que les contenus possibles sont : les *chaînes lexicales* et les *annotations*. De plus, il existe des vues condensées et des vues de type « carte ».

Dans le cadre du développement de NaviTexte, l'accent a été mis sur les aspects de navigation, compte tenu de l'existence de développements préalables, notamment réalisés dans la plate-forme ContextO et dans le projet RÉGAL (*cf.* chapitre 1). En conséquence, la vue de type arborescent et contenu chaînes lexicales, les opérations de coordination et la diversité d'opérations de mises en forme ont été mises en second plan.

12.3.3.1. Vue linéaire des chaînes lexicales

La vue linéaire (texte plat) a été développée en tant que vue d'expérimentation afin de tester notre approche. Le contenu du texte s'affiche de manière linéaire à partir des informations lexicales des feuilles de la hiérarchie textuelle définie dans le *corps* du texte. Les titres

sont affichés en gras et séparés par un espace des UT « communes ». De plus, l'encodeur peut définir les types d'UT qui constituent des *séparateurs* visuels, c'est-à-dire des éléments après lesquels il faut introduire un saut de ligne dans l'affichage.

12.3.3.2. Vue arborescente condensée des annotations

La vue s'affichant dans le panneau placé à gauche en haut (*cf.* figure 12.9) constitue une vue condensée de type arborescent, où le contenu sont les différentes annotations du texte. Cette vue est calculée automatiquement par la plate-forme (*i.e.* le concepteur de modules ne doit pas la définir à l'aide d'une description de vue), en inférant les relations entre les différentes UT, les annotations disponibles et les valeurs possibles pour chaque annotation. La vue présentée présente un atout important car elle combine une visualisation condensée des annotations avec la possibilité de déclencher, par le biais des cases à cocher, des opérations de visualisation de mise en forme, en particulier celles correspondant aux opérations *couleurPremierPlan* et *couleurArrièrePlan* (*cf.* chapitre 8). À chaque valeur possible d'une annotation, la plate-forme attribue une couleur de premier plan et une autre d'arrière plan par défaut. Ces couleurs sont configurables (*cf.* section 12.3.7). Elles sont également sauvegardées par le système afin de les utiliser postérieurement.

12.3.3.3. Vue de type carte : utilisation du « greeking »

La technique de « *greeking* » présentée au chapitre 2 est l'une des techniques utilisées en visualisation d'informations pour afficher un texte dans un format réduit. L'utilisation de cette technique nous permet de visualiser un texte selon une vue de type « carte » (*cf.* chapitre 7). Bien que nous n'utilisons pas la technique « *panning* & *zomming* » (*cf.* chapitre 2), le « *greeking* » nous permet d'avoir une vision globale du texte (*cf.* figure 12.13).

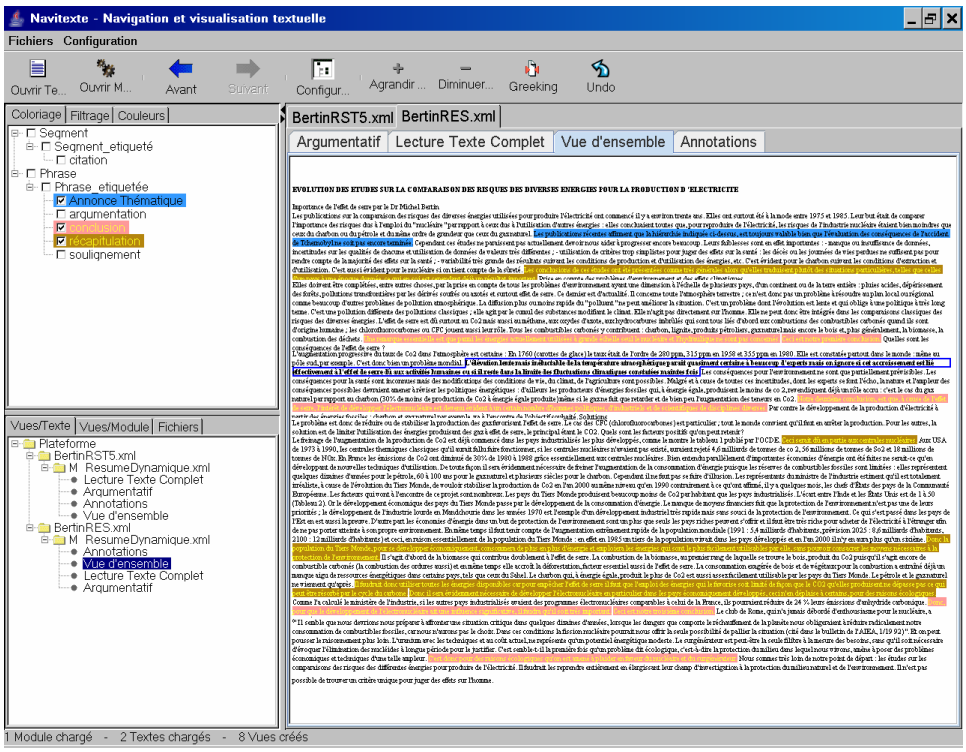

Figure 12.13 – *Exemple d'une vue de type carte, utilisant la technique de* « greeking ».

Il sera ainsi possible pour l'utilisateur d'évaluer très rapidement la position de l'UT active par rapport à la globalité du texte, aussi que la distribution des annotations.

12.3.3.4. Vue de graphe

Un premier essai d'implémentation de la vue de graphe a été réalisé. Cependant, ce type de vue pose plusieurs problèmes de mise en forme et d'interaction dûs à la taille du graphe. En raisons exclusivement de temps, j'ai laissé de côté ce type de vue. Son développement complet est proposé comme un travail futur.

12.3.4. Visualisation

Parmi les opérations de visualisation proposées par le langage (opérations de *mise en relief* et d'*aide contextuelle*), un ensemble a été choisi et implémenté.

En ce qui concerne la *mise en relief*, l'opération de transformation *ajouter* n'a pas été implémentée en raison de sa complexité technique car son implémentation complexifiait significativement la gestion de la coordination entre les UT et leur représentation graphique, assurée par la classe *InfoUTs* (*cf.* section 12.2.3.3). Les opérations *couleurPremierPlan* et *couleurArrièrePlan* ont été, dans un premier temps, incorporées à la plate-forme dans la vue arbo-

rescente condensée des annotations (*cf.* section 12.3.3.2). Leur traitement comme part du langage est en cours de développement. De même pour l'opération *taillePolice* : les options « Agrandir Police » et « Diminuer Police » font appel à son implémentation en interne. Il reste à intégrer ces éléments dans le langage.

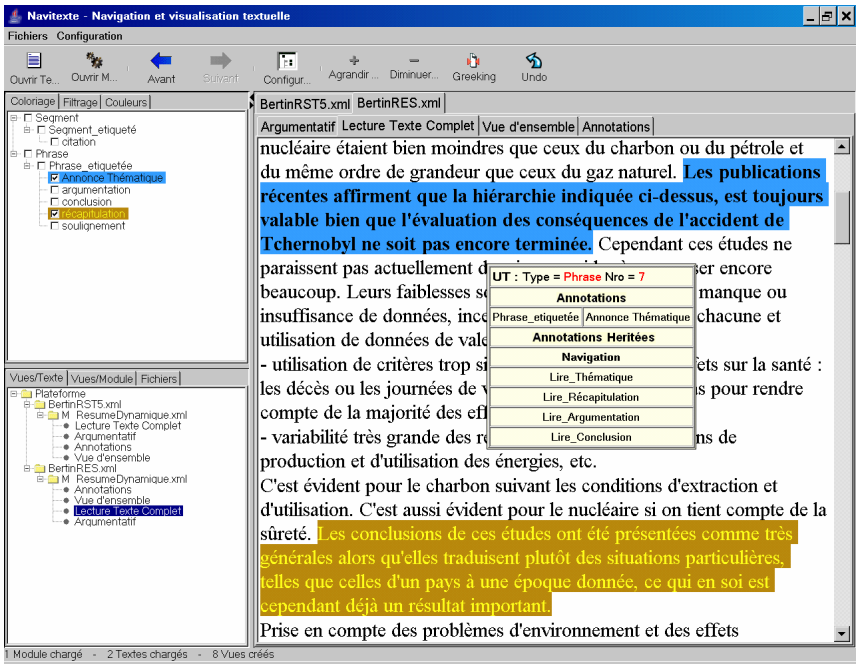

Figure 12.14 – *Aide contextuelle affichant une infobulle.*

En ce qui concerne l'*aide contextuelle*, une option configurable, de type affichage d'infobulles (*cf.* section 7.2.3.2) a été implémentée. Dans la figure 12.14, cette option a été activée. En conséquence, le survol de la souris sur l'UT « Les publications récentes… » déclenche l'affichage d'une infobulle comportant toutes les informations pertinentes de cette UT : les attributs *Type*, *Nro* et *Rang* ; les annotations directes et celles héritées ; les opérations de navigation potentiellement déclenchables. Cette aide s'est montrée particulièrement utile dans de situations d'encodage de textes et de connaissances.

12.3.5. Navigation

Une implémentation d'accès aux opérations de navigation a été développée. Lorsque l'utilisateur clique sur une UT ayant des opérations de navigation déclenchables, le système affiche un menu à l'utilisateur, avec toutes les opérations de navigation associées à l'UT ac-

tuelle (*cf.* figure 12.11). L'état des opérations est discernable d'après la police d'affichage, comme il a été décrit à la section 12.3.2.5.

La conception d'une palette de navigation plus ergonomique est en cours de développement.

12.3.6. Historique de navigation

Le système garde une trace des UT qui ont été sélectionnées, afin de construire un historique de navigation. Il existe trois manières de sélection d'UT (*cf.* section 12.3.2.1) : première UT de la vue, sélection directe et comme résultat de l'exécution d'une opération de navigation.

L'utilisateur dispose dans la barre d'outils de deux flèches afin de se déplacer à travers de l'historique de navigation. L'action consiste à changer l'UT actuelle, ce qui implique un changement visuel (déplacement dans le texte, encadrement de la nouvelle UT actuelle, etc.).

12.3.7. Configuration de la plate-forme

Cinq onglets de configuration de la plate-forme sont offertes à l'utilisateur. Premièrement, au niveau général, celui-ci peut configurer les répertoires de textes et de modules par défaut, ainsi que les panneaux à afficher dans la fenêtre principale de la plate-forme.

Deuxièmement, l'utilisateur peut spécifier une liste de modules à charger par défaut lorsque la plate-forme démarre.

Troisièmement, des options spécifiques aux vues sont configurables :

- activer la souris sémantique (*cf.* section 12.3.2.2) ;

- distinguer les UT selon leur état (*cf.* section 12.3.2.3) ;

- mettre en relief la trace des UT déjà naviguées (*cf.* section 12.3.2.4) ;

- montrer, au survol de la souris, l'information des UT (*cf.* figure 12.14).

Quatrièmement, en ce qui concerne la navigation, l'utilisateur peut choisir si le système montre toutes les opérations de navigation disponibles pour une UT ou si celles qui n'ont pas de cible sont cachées.

Enfin, l'utilisateur peut configurer les couleurs en premier et en arrière plan associées aux valeurs des annotations.

12.4. Limitations actuelles de la plate-forme

Les décisions prises au moment de développer NaviTexte entraînent des limitations de la plate-forme. En premier lieu, il reste des parties du langage Sextant à implémenter. En particulier, les vues de type arborescent et graphe, les opérations de coordination et les d'opérations de visualisation en tant qu'opérations interprétées par le système car actuellement quelques-unes ont été incorporées.

En deuxième lieu, la conception et développement d'une palette de navigation plus ergonomique que le menu offert à présent, se montre souhaitable. La combinaison d'aspects logiciels et matériels dans le cadre de la navigation textuelle est un objectif à long terme.

Enfin, il semble intéressant d'étudier la possibilité d'incorporer de nouveaux types de vues aux types existants. Cela peut impliquer la modification du langage Sextant.

12.5. Synthèse

Dans ce chapitre, j'ai présenté NaviTexte, la première plate-forme logicielle implémentant notre approche de navigation textuelle. Dans cette implémentation trois choses ont été développées :

- une représentation informatique des textes spécifique à la navigation textuelle, proposée au chapitre 6 et pour laquelle un encodage XML a été défini au chapitre 10 ;

- un interpréteur d'une version réduite de Sextant, le langage de modélisation des connaissances proposé au chapitre 8 et pour lequel un encodage XML a été défini au chapitre 11 ;

- un environnement capable de traiter les textes, d'interpréter le langage Sextant et de gérer l'interaction avec l'utilisateur.

Il faut souligner que NaviTexte est une plate-forme logicielle opérationnelle sur laquelle plusieurs applications concrètes ont été développées. De plus, l'aspect modulaire de la plate-forme, dû au très grand soin apporté à la conception générale, a permis d'adapter assez facilement NaviTexte à NaviLire, une application pour enseigner le français à des étudiants danois.

En ce qui concerne NaviTexte, deux choses sont à évaluer. D'une part, la plate-forme en soi-même, en tant que plate-forme de navigation textuelle implémentant notre approche. D'autre part, les applications créées en utilisant la plate-forme.

Je présente dans le chapitre suivant les applications créées en utilisant la plate-forme logicielle NaviTexte.

Chapitre 13

Applications de la navigation textuelle

13.1. Introduction

NaviTexte étant une plate-forme générique dédiée à la navigation textuelle, différentes applications profitant de notre approche peuvent se concevoir. La création d'une application sur NaviTexte implique de disposer d'un ensemble de textes encodés selon la DTD proposée au chapitre 10 et de définir un ensemble de modules de connaissances encodés selon la DTD proposée au chapitre 11.

Cependant, dans le cadre d'une collaboration avec Lita Lundquist (Département de français Institut F.I.R.S.T - Handelshøjskolen i København), nous avons développé un outil d'aide à la compréhension de textes complexes pour des étudiants FLE, nommé NaviLire[61], qui s'appuie sur la plate-forme NaviTexte. L'objectif a été de concevoir un outil qui permette à un apprenant de voir et de naviguer dans un texte entre des unités textuelles assurant la cohérence de ce texte.

Je présente dans ce chapitre trois applications différentes de la plate-forme logicielle NaviTexte. L'hétérogénéité de ces applications met en évidence la souplesse de NaviTexte en tant que plate-forme d'expérimentation.

13.2. Résumé automatique

Un grand nombre de systèmes de résumé automatique ont été proposés ces dernières années [Mani 2001], [Minel 2003]. Tous ces systèmes, fondés sur le principe de l'extraction de phrases, ont été confrontés à deux problèmes intrinsèques au procédé d'extraction. D'une part, à la rupture de la cohésion textuelle, comme par exemple la présence d'anaphores sans leur référent discursif. D'autre part, à l'adaptation du résumé aux besoins spécifiques d'un

[61] Cette application a reçu le soutien financier du département scientifique de l'ambassade de France au Danemark

lecteur. Jusqu'à présent ces problèmes n'ont pas reçu de solutions totalement satisfaisantes. Une autre approche consiste à considérer le processus de résumé comme un cheminement, plus exactement un parcours de lecture, dans le texte source qui soit propre au lecteur. Ainsi plutôt que de construire des fragments textuels, nous proposons des parcours de lecture spécifiques, concrétisés sous la forme d'opérations de navigation dans la plate-forme NaviTexte.

Cette première application développée est liée au point de départ de notre projet de recherche : le projet RÉGAL [Ferret *et al.* 2001] [Sabah 2002] [Couto *et al.* 2004], présenté au chapitre 1.

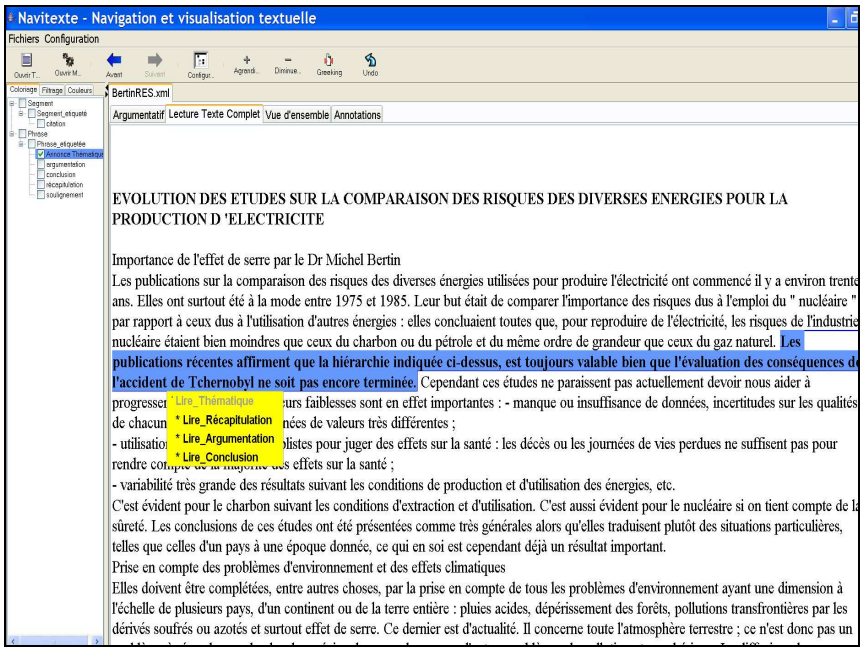

Figure 13.1 – *Exemple d'opérations de navigation pour le résumé automatique.*

Les textes étaient déjà encodés conformément à la DTD ContextO. Dans une première étape, nous avons converti ces textes afin de les rendre conformes à la DTD NaviTexte, à l'aide d'un outil de transformation réalisé dans le cadre d'un projet du DESS « *Ingénierie de la Langue pour la Gestion Intelligente de l'Information* » (*cf.* chapitre 10). D'autres transformations sont envisageables. Par exemple, des textes annotés par un système automatique comme LinguaStream [Bilhaut 2003] pourraient être convertis en entrées de NaviTexte.

Ensuite, un module de connaissances a été conçu en utilisant l'éditeur de modules présenté au chapitre 11. Dans ce module nous avons défini quatre descriptions de vue :

- une vue nommée « Lecture texte complet », où nous avons encodé toutes les opérations de navigation conçues au sein du projet RÉGAL (*cf.* figure 1.11) ;

- une vue nommée « Annotations », où seules les UT comportant des annotations s'affichent ;

- une vue « *greekée* » nommée « Vue d'ensemble », sans opérations de navigation ;

- une vue nommée « Argumentatif », ou seules les UT comportant des annotations qui révèlent une argumentation s'affichent.

Un exemple de parcours de lecture est illustré par la figure 13.1. Les annotations sont du type « Annonce Thématique », « Conclusion », « Soulignement Auteur », etc. On peut voir sur la figure les quatre parcours de lecture différents, suivant que celui-ci s'intéresse plutôt aux thèmes de l'article, à son argumentation ou à ses conclusions. Ainsi, dans la continuité de sa lecture du texte, le lecteur se voit proposer, par une signalétique spécifique, des parcours spécifiques sans rupture de la cohésion textuelle puisqu'il voit à tout instant le texte complet.

Cette application de NaviTexte au résumé automatique nous a permis de montrer, au moins en ce qui concerne le projet RÉGAL, la validité de notre hypothèse de départ : que les représentations visuelles d'un texte, les opérations de visualisation et les opérations de navigation, encodées dans l'applicatif, sont parfaitement isolables dans une entité (module de connaissances) à l'aide d'un langage (Sextant), et que cette entité peut être interprétée par un système (NaviTexte) afin de gérer l'interaction utilisateur.

13.3. Projet NaviLire, Modalités d'apprentissage en linguistique textuelle

13.3.1. Introduction

Le projet NaviLire a donné lieu à plusieurs publications [Couto *et al.* 2005$_a$], [Couto *et al.* 2005$_b$], [Lundquist *et al.* 2006]. Je reprends ci-dessous une partie de ces articles pour présenter de manière synthétique cette application. La linguistique textuelle permet, entre autres, de focaliser sur des structures textuelles et des réalisations linguistiques de la cohérence qui sont propres à des types de textes particuliers (texte narratif, texte argumentatif, etc.), voire propres à des langues différentes. En effet, en adoptant une perspective linguistique contrastive, il s'avère que les textes ne s'organisent pas de la même manière, même dans des langues apparemment proches, comme le français et l'anglais, ou comme le français et le danois. Ain-

si, les langues romanes réalisent souvent la reprise anaphorique par des syntagmes nominaux pleins et variés (nommés « anaphores infidèles » dans [Lundquist 2005]), comparées aux langues germaniques comme le danois, qui préfèrent l'anaphorisation pronominale; un signe parmi d'autres de stratégies totalement différentes pour présenter et organiser l'information dans les textes.

C'est pourquoi la linguistique textuelle, en plus de son intérêt purement linguistique, contient aussi des implications importantes pour l'enseignement des langues étrangères, surtout lorsque celles-ci sont vues sous leur aspect contrastif, et analysées à travers des textes authentiques, énoncés dans des situations réelles de communication. Si la cohérence textuelle constitue bien une « heuristique générale » d'interprétation textuelle [Charolles 1981], on peut la considérer comme quasi-universelle aussi, et par suite l'exploiter pour montrer, et enseigner, comment la cohérence se manifeste linguistiquement dans les textes en général, et dans des types de textes différents en particulier, de même que dans des langues différentes.

Il semble donc que l'enseignement de la linguistique textuelle contribue, dans de multiples contextes, à aiguiser l'attention des apprenants vers la réalisation de la « bonne formation » des textes, et à stimuler leur propre production de textes bien formés.

Sous cette hypothèse, il a été développé un outil, nommé NaviLire, qui s'appuie sur la plate-forme NaviTexte, permettant à l'élève/l'étudiant de **voir** et de **naviguer** dans le texte entre des unités textuelles assurant la cohérence, afin de réaliser tant la lecture que d'apprendre la production écrite de textes.

13.3.2. Problèmes cognitifs liés à l'apprentissage de la compréhension écrite des textes

Par le procédé selon lequel le lecteur apprend à naviguer dans un texte en suivant ses différentes pistes de cohérence – basées sur la référence, sur la prédication et sur les connecteurs – on attaque des problèmes cognitifs cruciaux pour lire, comprendre et interpréter correctement un texte, ainsi que pour apprendre par les textes. Le premier problème consiste à identifier les référents discursifs d'un texte et d'établir les relations correctes entre les SN qui y réfèrent. En d'autres termes, de décider s'il s'agit d'une relation de co-référence ou d'une disjonction référentielle. Cette compétence cognitive est primordiale pour arriver à établir une représentation mentale cohérente et correcte du texte en question, condition de toute compréhension par les textes : « *learning from text requires that the learner construct a coherent mental representation of the text* » [Kintsch 1998: 307].

Le second problème cognitif consiste à identifier le « où veut en venir l'émetteur » du texte. Cette orientation – expressive, argumentative, et d'autre –, qualifié de « programme

d'interprétation » par Lita Lundquist [1990$_b$], fonctionne d'abord du *général au particulier*, et ensuite *du spécifique au générique*, permettant d'identifier des marques dans le texte qui « vont dans le même sens » (voir *macrostructure* et *microstructure*, [Kintsch 1998: 50]. Cette identification de l'orientation, apportée entre autres par les prédications, est primordiale pour un déchiffrage correct de la cohérence sémantique et pragmatique du texte.

Enfin, les connecteurs soulignent les relations rhétoriques à établir entre des propositions ou autres séquences du texte, ce qui contribue, évidemment, de manière essentielle à établir les relations nécessaires pour construire la représentation mentale correcte du texte.

13.3.3. Représentation des unités textuelles nécessaires à la situation d'apprentissage

Pour naviguer dans le texte, nous avons isolé des unités textuelles qui permettent de spécifier des opérations de navigation, ce qui équivaut à établir des liens de cohérence entre des unités de même nature. Comme les éléments textuels appartiennent à des types différents, la navigation permet d'une part de suivre des pistes de cohérence différentes dans un même texte, et d'autre part d'en identifier les réalisations linguistiques dans une langue donnée (à l'occasion, le français). Plutôt que de manipuler des structures textuelles hiérarchiques [Couto *et* Minel 2004$_a$, 2004$_b$], nous distinguons ici des pistes parallèles de marques textuelles, chacune contribuant à un type particulier de cohérence.

Ces types de cohérence sont fondés, *grosso modo*, sur les principes exposés dans [Lundquist 1980, 1999], selon lesquels on peut distinguer dans les textes une cohérence référentielle, une cohérence prédicative et une cohérence pragmatique, fondée respectivement sur les trois actes de langage : la référence, la prédication et l'illocution qui entrent dans l'énonciation de chaque phrase [Searle 1969]. En fait, chaque phrase contient, en règle générale, un ou des syntagmes nominaux qui réfèrent à des entités extra-textuelles et instaurent des référents discursifs ; une ou plusieurs prédications qui d'une part prédiquent des propriétés et établissent des relations entre les référents discursifs et d'autre part peuvent porter des traces énonciatives, indicateurs avec d'autres de l'acte d'illocution, et de la cohérence pragmatique ; celle-ci se voit aussi signalée par les connecteurs qui témoignent d'un souci de faire ressortir les liens rhétoriques entre les propositions [Thompson *et* Mann 1988].

Retenant aussi l'unité phrase, nous organisons donc les pistes de cohérence autour de quatre types d'unités textuelles : les *phrases*, les *syntagmes nominaux*, les *prédications* et les *connecteurs*. Cet inventaire peut, à première vue, sembler trahir la multitude d'objets théoriques dégagés par la linguistique textuelle au cours des années, mais, correspondant à un

souci de simplification, il présente des unités textuelles qui peuvent, à notre avis, être le mieux exploitées pour naviguer dans la fouille textuelle de différentes pistes de cohérence.

13.3.4. Expérimentations d'usage

Rappelons (*cf.* section 4.3.1) que les résultats de différentes études sur l'intérêt de l'utilisation de dispositif d'apprentissage s'appuyant sur des techniques hypertextes sont contradictoires. Néanmoins, ces études ne sont pas pertinentes pour évaluer l'intérêt de notre approche car il convient de rappeler que notre dispositif de navigation est très différent. En effet, contrairement à l'hypertexte où la navigation est inscrite dans l'hyperlien, dans la plate-forme

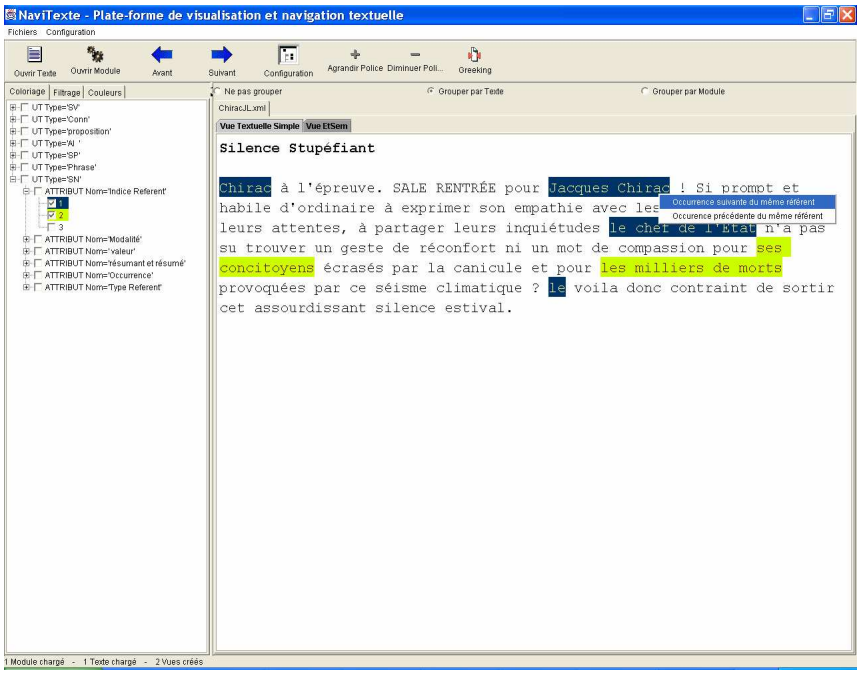

Figure 13.2 – *Exemple d'utilisation de NaviTexte pour l'enseignement en FLE.*

NaviTexte (et en conséquence dans la plate-forme NaviLire), la navigation est le résultat d'un processus qui exécute des connaissances qui ont été exprimées par l'auteur ou par l'enseignant et ces connaissances sont différentes pour chaque vue d'un même texte.

Afin de mesurer l'intérêt de notre dispositif, une expérimentation a été mise en œuvre. Dans une première étape, sur un groupe de quatorze étudiants danois en FLE, un test de lecture, utilisant la méthode classique de « papier et crayon », a été pratiqué, pour déterminer

leur compétence de lecture en français. Comme résultat, on a constaté que celle-ci était quasi similaire pour tous les sujets.

Dans une deuxième étape, la moitié des étudiants, un groupe de sept sujets nommé les « papiristes », a effectué une nouvelle expérimentation de lecture en utilisant la méthode « papier et crayon », tandis que l'autre moitié, les « navilistes » ont lu le même texte en utilisant NaviLire[62] (*cf.* figure 13.2). L'utilisation de NaviLire leur permettait d'avoir des navigations guidées au long des pistes sur :

- le thème principal du texte ;

- les marqueurs argumentatifs ;

- le pronom personnel *nous*.

NaviLire offrait également la possibilité de visualiser ces trois types d'information.

Après avoir étudié le texte, tous les sujets ont répondu au même questionnaire visant à tester leur compréhension du texte. Les questions adressaient différents points. Par exemple :

- des perspectives phénoménologiques (« combien difficile est le texte à lire/retenir, sur une échelle de 1-5 ? ») ;

- le nombre de termes retenus (« notez autant de termes dont vous vous souvenez pour référer X » ou « lesquelles des termes suivants sont apparus dans le texte : t_1, t_2, …, t_n ?») ;

- la compréhension générale du texte (« de quel type de texte s'agit-t-il ? » ou « comment vous argumentez cette caractérisation ? »).

13.3.5. Résultats de l'expérimentation

Les premiers résultats de l'expérimentation, affichés dans le tableau 13.1, sont calculés sur un total de quarante réponses, dont trente cinq concernent le contenu du texte. Ces résultats montrent que les « navilistes » ont une meilleure compréhension du texte que les « papiristes » dans quatorze questions, une compréhension égale dans seize questions, et une compréhension plus pauvre dans cinq questions.

[62] Ces étudiants ont effectué préalablement un entraînement avec NaviLire d'environ une heure et demi.

	Nombre de questions	Pourcentage
« *navilistes* » *mieux que* « *papiristes* »	14	40
« *navilistes* » *égal aux* « *papiristes* »	16	45,7
« *navilistes* » *pire que* « *papiristes* »	5	14,3
Total	35	100

Tableau 13.1 – *Comparaison des « navilistes » et « papiristes » (extrait de [Lundquist et al. 2006]).*

Les résultats quantitatifs affichés dans le tableau 13.2 montrent que, en ce qui concerne les questions phénoménologiques, il n'existe pas de différences entre « papiristes » et « navilistes » : les deux groupes jugent le texte facile à lire (respectivement 2,6 en moyenne versus 2,7) et facile à retenir (respectivement 2,7 en moyenne versus 2,9). En ce qui concerne le nombre de réponses correctes, prenons par exemple la question : « lesquels des termes suivants sont apparus dans le texte… » ; les « navilistes » ont répondu légèrement mieux que les « papiristes » (respectivement 32,3 réponses correctes en moyenne versus 29,9). De même pour la question « notez tous les marqueurs argumentatifs utilisés dont vous vous souvenez » : les « navilistes » ont signalé 4 en moyenne, versus 2,6 pour les « papiristes ».

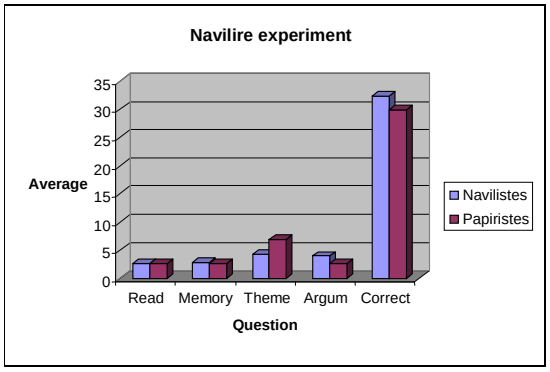

Tableau 13.2 – *Premiers résultats qualitatifs de l'expérimentation (extrait de [Lundquist et al. 2006]).*

Néanmoins, les « papiristes » ont été plus performants que les « navilistes » au moment de reproduire les termes référant le thème principal du texte, ayant en moyenne un nombre de 7 termes versus 4,3. Ce résultat est lié à une différence plus prononcée, trouvée dans l'analyse qualitative des réponses, qui a montré que les « papiristes » semblent avoir compris la structure générale du texte mieux que les « navilistes ». De fait, des sept « papiristes », quatre ont référé la structure globale du texte, qui était argumentative, en comparaison à zéro entre les « navilistes ».

13.3.6. Conclusions de l'expérimentation

Les différences constatées entre « papiristes » et « navilistes », au moins en ce qui concerne la compréhension générale du texte et sa structure, peuvent, prudemment, s'aligner avec certains constats préalables du domaine de la psycholinguistique concernant les logiciels de e-learning [Coirier *et al.* 1996], [Baccino 2004].

Les conclusions générales que l'on peut tirer de l'expérimentation sont, d'une part, que l'impact de la plate-forme logicielle NaviLire doit être évalué en menant de nouvelles expérimentations, plus raffinées que la première ici présentée. D'autre part, il s'avère souhaitable d'effectuer l'étude et conception de vues sémiotiques adaptatives pour pallier le problème de compréhension globale du texte.

13.4. En relisant Madame Bovary

13.4.1. Introduction

En collaboration avec Yannick Mathieu, du Laboratoire de Linguistique Formelle, UMR 7110, CNRS – Université Paris 7, il a été développé une application visant la lecture du roman Madame Bovary suivant les sentiments des personnages.

13.4.2. Annotations du roman

13.4.2.1. Termes d'intérêt

L'annotation des sentiments dans le roman est fondée sur un lexique d'environ 1200 termes dénotant sentiments, émotions et états psychologiques sous forme de verbes comme *aimer*, *effrayer*, etc., substantifs comme *amour*, *colère*, etc., et adjectifs comme *amoureux*, *jaloux*, etc. Ces termes sont regroupés en 38 classes sémantiques : *amertume*, *amour*, *amusement*, *déception*, *déprime*, *indifférence*, *pitié*, etc. [Mathieu 2000, 2004, 2005$_a$, 2005$_b$]. De plus, il existe trois catégories de termes : *négatif*, *positif* et *neutre*.

13.4.2.2. Les relations entre les classes sémantiques

Il existe des relations de *sens*, d'*intensité* et d'*antonymie* entre les classes sémantiques, lesquelles peuvent se représenter en utilisant des graphes (*cf.* figure 13.3).

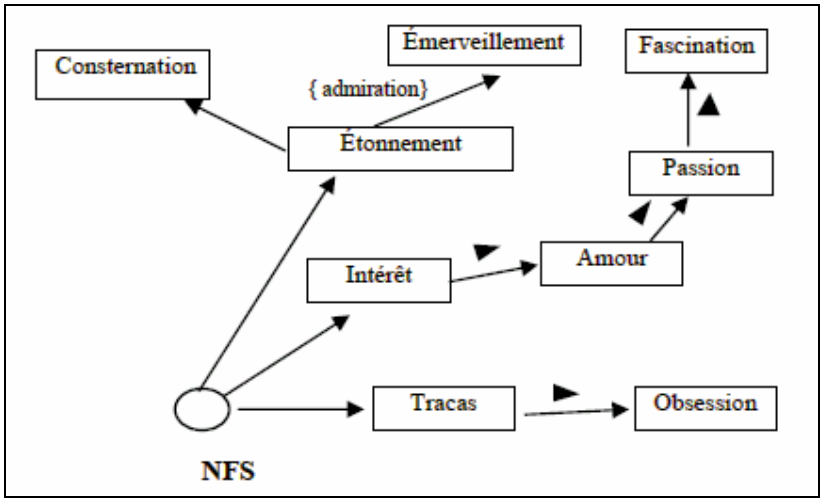

Figure 13.3 – *Les relations de sens et d'intensité entre les classes sémantiques (extrait de [*Mathieu 2005$_a$*]).*

L'orientation des arcs dans les graphes dépend de l'intensité du sentiment expérimenté. Deux concepts sont fondamentaux dans les relations : la notion d'*intensificateur* et la notion d'*expériencer et objet*.

L'*intensificateur* est une propriété dont la valeur par défaut est *neutre*, qui sert pour dénoter, en affectant sa valeur à *fort*, que dans une même classe sémantique il existe des termes dont l'intensité est plus marquée. Par exemple, pour les termes *exaspérer* et *irriter*, appartenant à la même classe sémantique, *l'intensificateur* du premier terme est affecté à fort tandis que celui du deuxième est affecté à neutre.

L'*expériencer* est l'individu qui éprouve un sentiment et l'*objet* est l'objet que concerne ce sentiment. En guise d'exemple, dans la phrase « Paul effraye Mary », l'*expériencer* est Paul et *l'objet* est Mary.

13.4.2.3. Les UT choisies

Les unités textuelles annotées sont des verbes, des substantifs et des adjectifs de sentiments identifiés semi-automatiquement. Une UT générique de type « Marque Sentiment » est utilisée, regroupant les trois catégories syntaxiques possibles. Les UT annotés dans le ro-

man contient l'information suivante sous forme d'annotations : catégorie grammaticale, classe sémantique, *intensificateur*, *expériencer* et *objet*.

13.4.3. Les opérations de navigation

Un ensemble d'opérations de navigation a été défini afin d'offrir des pistes de lecture au lecteur.

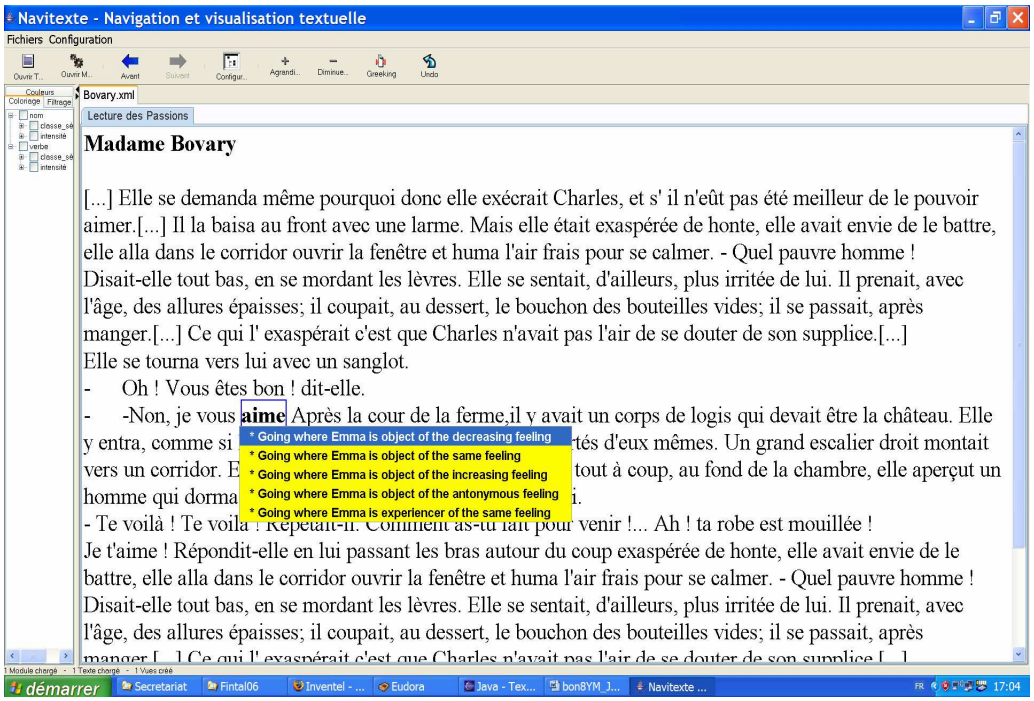

Figure 13.4 – *Exemple de pistes de lecture pour lire Madame Bovary.*

Actuellement, cinq pistes de lecture existent (*cf.* figure 13.4), pour chaque personnage principal du roman (Emma, Rodolphe, Charles et Léon) :

- même sentiment ;

- même sentiment avec intensité supérieure ;

- même sentiment avec intensité inférieure ;

- sentiment antonyme ;

- l'*experiencer* devient l'objet du même sentiment.

13.4.4. Expérimentations

Ce travail, en cours de développement n'a pas encore donné lieu à des expérimentations d'usage auprès d'utilisateurs autres que les chercheurs impliqués dans le projet.

Une expérimentation, auprès des étudiants de FLE est prévue pour la fin de l'année 2006.

13.5. Synthèse

Trois différentes applications de la plate-forme logicielle NaviTexte ont été présentées. Celles-ci sont assez hétérogènes, ce qui est pour nous une preuve de la souplesse de Navi-Texte en tant que plate-forme d'expérimentation. De plus, dans le cadre d'une collaboration pour enseigner le FLE, il a été développé NaviLire, un outil d'aide à la compréhension de textes complexes. L'aspect modulaire de la NaviTexte a permis d'adapter assez facilement celle-ci à NaviLire.

Il faut souligner que plusieurs aspects sont à considérer pour l'évaluation de NaviTexte. En premier lieu, NaviTexte, en tant que plate-forme logicielle, peut être évaluée vis-à-vis de la navigation textuelle. En deuxième lieu, les applications créées en utilisant la plate-forme peut être évaluées de manière indépendante. En fait, la création d'applications concrètes pour des cas d'utilisation réels, alimente des discussions quant à une méthodologie d'acquisition et expression de connaissances. D'une part, il est important de définir les moyens pour coordonner l'information encodée dans les textes avec les vues et opérations définies dans les modules de connaissances. Il existe des problèmes concernant le niveau de détail des connaissances dans les textes, la granularité des UT, la nomenclature, la manière dont les opérations de navigation peuvent être définies, etc. Un travail futur consiste à proposer, des « directives d'encodage de connaissances », tant dans les textes que dans les modules. Cela entraînera, sûrement, un enrichissement du langage Sextant et de nouvelles versions de la plate-forme logicielle NaviTexte.

En dehors des applications présentées dans ce chapitre, il y en a d'autres en cours actuellement. Je cite, par exemple, le projet « Parcours de lecture chronologique dans les textes de biographie », développé en collaboration avec Delphine Battistelli , Sylviane Schwer et Jean-Luc Minel qui vise à développer un outil logiciel qui assiste un lecteur dans sa lecture d'un texte, en exploitant les marques temporelles du texte. Cette assistance à la lecture se matérialisant sous la forme de propositions de déplacement dans le texte, l'utilisation de NaviTexte semble assez pertinente. Cette application de NaviTexte permettrait, par exemple, de naviguer en ordre chronologique un texte comme celui affiché dans le tableau 5.3, au chapitre 5.

Toutes ces applications, et les expérimentations menées avec elles, poussent à la réflexion, favorisant l'enrichissement général de notre approche de navigation textuelle.

Conclusions

Dans le cadre de cette thèse, l'aspect le moins exploré a été celui concernant les agents encodeurs. Les raisons en sont multiples, mais celles-ci peuvent se résumer à une question de stratégie : pour des contraintes de temps, il fallait choisir entre dédier une partie considérable de temps à chercher des experts et à travailler systématiquement avec eux afin de dégager une méthodologie d'acquisition et l'expression des connaissances, ou développer une première version de notre approche et raffiner celle-ci au vu des expérimentations. Essayant une couverture en ampleur de la thématique de cette thèse, j'ai choisi la deuxième option. A la lumière des différentes applications créées avec la plate-forme logicielle NaviTexte et des résultats partiels obtenus, je reste assez persuadé que le choix a été juste.

Je présente par la suite un bilan de ce travail de thèse ainsi que des éléments pour de possibles travaux futurs.

1. Bilan

Dans cette thèse j'ai proposé une approche systématisée à la navigation textuelle, en déterminant ses quatre éléments constitutifs : l'existence d'une représentation des textes ; la possibilité d'isoler les connaissances visuelles et navigationnelles ; l'existence d'un agent capable d'encoder ces connaissances ; et un système qui interprète ces connaissances.

En ce qui concerne la représentation des textes, nous avons vu que la conceptualisation de l'objet « texte » n'est pas triviale car il s'agit d'un objet complexe. En conséquence, tout essai de le modéliser ou de le représenter, afin de le manipuler informatiquement, présuppose nécessairement une prise de position et, souvent, une simplification. Mais, comme le traitement informatique d'un texte exige nécessairement une représentation de celui-ci afin d'être manipulé par un système, une analyse des différents types de représentations des textes a été réalisée. Trois types de représentations possibles ont été déterminés : *synthétisées*, *plates* et *structurées*. Puisque le type de traitement à effectuer détermine la représentation la plus convenable, les représentations présentées s'avèrent insuffisantes pour la navigation textuelle. Les représentations structurées sont plus adéquates dans notre cadre, mais il manque la possibilité de représenter des structures discursives complexes qui ne suivent pas une hié-

rarchie. Aussi, j'ai proposé une représentation permettant d'exprimer à la fois la structure syntaxique des textes, les structures discursives, et les valeurs sémantiques des constituants textuels, car toutes ces informations servent de guide à la lecture dans le cadre de la navigation textuelle. Un encodage XML des textes, conforme avec cette représentation, a été également proposé. L'intérêt de combiner la structure syntaxique avec des phénomènes discursifs, et le fait de souhaiter manipuler certaines structures complexes en tant qu'entités, ont motivé la définition de cette représentation spécifique pour la navigation textuelle. Celle-ci comporte les caractéristiques suivantes :

- le type d'unités textuelles qui composent un texte n'est pas restreint à un ensemble prédéterminé ;

- il est possible d'avoir une organisation hiérarchique des unités textuelles et une autre permettant d'exprimer des relations non hiérarchiques ;

- les titres sont considérés comme des unités textuelles ;

- toute unité textuelle, y compris les titres et les relations non hiérarchiques, est susceptible d'avoir un nombre non limité d'annotations de nature quelconque.

J'ai présenté ensuite différentes connaissances à modéliser dans le cadre de la navigation textuelle : la notion de vue d'un texte et les opérations de navigation, de visualisation et de coordination. De mon point de vue, les connaissances décrites dans le cadre de cette thèse constituent un bon point de départ pour développer la navigation textuelle, mais elles ne doivent pas être considérées comme exhaustives ou définitives. L'objectif de modéliser ces connaissances est double. D'une part, celles-ci sont généralement des connaissances amalgamées à plusieurs plates-formes de traitement automatique de textes. Or, on peut abstraire une telle plate-forme en termes de deux composants : un moteur chargé d'effectuer le traitement et un agent chargé des interfaces graphiques et de la gestion de l'interaction utilisateur. Pourquoi implémenter chaque fois les mêmes interactions ou les mêmes options de visualisation ? Pourquoi prétendre que les experts, pour ne mentionner qu'un exemple, en résumé automatique doivent être également experts en IHM ? Et si l'on fait appel à une équipe d'experts en IHM, pourquoi ne pas leur donner les moyens de créer des produits réutilisables, toujours dans le cadre du traitement automatique de textes ? D'autre part, la modélisation permet le développement progressif de bases de connaissances et le raffinement de celles-ci comme résultat d'une itération où l'encodeur de modules encode des connaissances qui sont ensuite testés sur des textes réels et avec des utilisateurs réels. Ce processus peut s'enrichir avec l'échange des connaissances réalisé par différents encodeurs.

Après avoir décrit les connaissances à modéliser, le langage formel de modélisation Sextant a été proposé. D'abord, la syntaxe du langage a été développée de manière incrémentale en exemplifiant les différents constructeurs du langage. Ensuite, une sémantique formelle du langage a été exprimée. Quoique celle-ci puisse être légèrement complexe, elle montre que le langage est bien défini car le sens des différents éléments du langage est précisé. De plus, une sémantique partielle du système interpréteur est également exprimée, en proposant des opérations abstraites dont la sémantique est explicitée. L'un des intérêts de cette sémantique opérationnelle, est de favoriser l'implémentation.

Une version XML du langage de modélisations de connaissances Sextant est définie. Le point fort de cette version réside dans la facilité, pour un système informatique, de manipuler les différentes instances du langage (des documents XML). Le point faible devient de la complexité des documents XML obtenus faiblesse qui a été en partie palliée par le développement d'un éditeur de modules de connaissances.

Il faut souligner que les connaissances exprimées dans les modules de connaissances s'appuient, d'une part, sur les informations déjà présentes dans les textes à traiter et, d'autre part, sur les connaissances que l'encodeur possède sur les attentes de l'utilisateur par rapport à l'information à lui offrir. Tout ce qui concerne le langage de conditions utilise la hiérarchie des UT et les annotations existantes dans le texte source. La manière dont ces informations sont utilisées et, en définitive, reliées, que ce soit par des opérations de visualisation, de navigation ou de coordination, dépend strictement de l'expertise de l'encodeur. Cela signifie que la qualité de la navigation textuelle repose sur la richesse de l'information présente dans les textes et sur la compétence de l'encodeur. En ce qui concerne le premier point, il existe de plus en plus de documents annotés avec des informations diverses. Or, bien que la compétence de l'encodeur puisse se concevoir initialement comme intuitive (par exemple, quelqu'un qui imagine comment l'utilisateur peut naviguer un texte, ou comment certaines unités textuelles devraient s'afficher), elle peut parfaitement être systématisée au long de différentes expérimentations dans le cadre de la navigation textuelle. Le développement d'une méthodologie d'acquisition de connaissances constitue actuellement l'un des objectifs les plus centraux de notre approche.

La plate-forme logicielle NaviTexte implémente un interpréteur d'une version réduite du langage Sextant. Il s'agit de la première plate-forme logicielle implémentant notre approche de navigation textuelle. Il faut souligner que NaviTexte est une plate-forme logicielle opérationnelle sur laquelle plusieurs applications concrètes ont été développées. De plus, l'aspect modulaire de la plate-forme, dû au très grand soin apporté à la conception générale, a per-

mis d'adapter assez facilement NaviTexte à NaviLire, une application pour enseigner le français à des étudiants danois. En dehors des applications présentées dans cette thèse, il y en a d'autres en cours actuellement. Toutes ces applications, et les expérimentations menées avec elles, poussent à la réflexion, favorisant l'enrichissement général de notre approche de navigation textuelle.

2. Perspectives

Plusieurs travaux futurs sont possibles, touchant les différents aspects abordés dans cette thèse.

Au niveau général, des questions de fond se posent sur la nature de notre approche de la navigation textuelle. Il reste des réflexions à faire sur ce qui peut être compris lorsque l'on parle de naviguer dans les textes. Autrement dit, et je reviens à la question qui amorce ce document : qu'est-ce que la navigation textuelle ? S'agit-il simplement de se déplacer parmi différentes unités textuelles d'un texte ? La recherche d'information ou un système de type question-réponse (*cf.* chapitre 7) représentent-ils des exemples de navigation textuelle ou d'applications de la navigation textuelle ? Notre approche de la navigation textuelle est-elle au final une généralisation de l'approche de l'hypertexte, assez proche de l'hypertexte dynamique, ou notre démarche est bien différente ? Puisque l'utilisateur a des besoins d'information plus ou moins spécifiques au moment d'interagir avec NaviTexte, serait-il intéressant de modéliser celui-ci et d'incorporer cette modélisation dans notre approche ? La modélisation de l'utilisateur est-elle implicite dans la codification de modules réalisée par l'encodeur ? Notre approche est-elle applicable lorsque le support change ; par exemple dans le cadre des livres électroniques ?

D'un point de vue plus spécifique, diverses améliorations peuvent être réalisées sur les quatre axes qui composent la navigation textuelle. En ce qui concerne la représentation des textes (*cf.* chapitre 6) et son encodage XML (*cf.* chapitre 10), une première piste de travail consiste à développer, d'une part, un éditeur spécifique qui permette à un utilisateur non expert d'encoder un texte NaviTexte à partir d'un texte plat. Un tel éditeur est indispensable dans une perspective de création manuelle ou sémi-automatique de corpus dans le cadre d'une application concrète. Actuellement, à la Faculté d'Ingénierie de l'Université de la République (Uruguay), un projet est en cours afin de développer un éditeur de textes NaviTexte. D'autre part, il convient d'étudier la faisabilité de convertir des fichiers issus d'autres traitements (par exemple, des textes annotés par un système automatique comme LinguaStream [Bilhaut 2003]) à des fichiers XML conformes à la DTD NaviTexte. Enfin, il semble in-

téressant d'évaluer la possibilité de proposer de nouveaux constructeurs d'UT (par exemple la notion de *tuple*) utilisés dans la *tête* du texte.

Quant au langage Sextant, d'autres types de vues que ceux proposés sont envisageables. Je prend l'exemple des vues basées sur la technique « Focus + Context » (*cf.* chapitre 2) [Lamping *et* Rao 1996] [Dieberger *et* Russell 2002], ou d'autres plutôt *ad-hoc* comme la vue « doc-ball » [Crestani *et al.* 2002], qui montre la structure hiérarchique d'un document en utilisant plusieurs cercles concentriques sectionnés. En outre, il faudrait étudier comment incorporer au langage le fait que les nouvelles UT *séquence* et *référence* introduisent des opérations de navigation inhérentes à elles-mêmes. La question qui se pose est la suivante : faut-il définir des opérations de navigation spécifiques à certains types de nouvelles UT (en conséquence l'encodeur doit les exprimer dans les modules de connaissances) ou suffit-il de considérer celles-ci comme prédéfinies et le système interpréteur est chargé de les offrir par défaut, sans nécessité d'une déclaration explicite d'opérations de navigation ?

Une autre possibilité consiste à permettre de manipuler l'historique de navigation afin d'écrire des conditions qui le prennent en considération. Ce faisant, des opérations de visualisation ou de navigation pourraient être proposées selon l'état total ou partiel de l'historique de navigation.

L'application à la lecture de Madame Bovary (*cf.* chapitre 13) nous a montré l'intérêt d'incorporer des ressources externes au langage, comme des thésaurus ou des ontologies. Ce faisant, on pourrait, par exemple, naviguer d'une classe de sentiments vers une autre ayant une relation avec la première, sans nécessiter de lexicaliser les noms de classes, mais en décrivant des conditions fondées sur la différence d'intensité entre les différentes classes de sentiments.

Enfin, la notion d'empan a été particulièrement étudiée sans avoir abouti, faute d'exemples réels d'utilisation, à une formalisation de celle-ci dans le langage Sextant. Cependant, je considère que la notion d'empan est importante car elle enrichit le concept d'opération de navigation. En conséquence, cette notion mérite une étude plus détaillée, peut-être à la lumière de futures applications qui montrent mieux son applicabilité.

La création d'applications concrètes pour des cas d'utilisation réels, alimente des discussions quant à une méthodologie d'acquisition et l'expression de connaissances. D'une part, il est important de définir les moyens pour coordonner l'information encodée dans les textes avec les vues et les opérations définies dans les modules de connaissances. Il existe des problèmes concernant le niveau de détail des connaissances dans les textes, la granularité des

UT, la nomenclature, la manière dont les opérations de navigation peuvent être définies, etc. Un travail futur consiste à proposer, des « directives d'encodage de connaissances », tant dans les textes que dans les modules. Cela entraînera, sûrement, un enrichissement du langage Sextant et de nouvelles versions de la plate-forme logicielle NaviTexte.

En ce qui concerne cette plate-forme, il reste des parties du langage Sextant à implémenter. En particulier, les vues de type arborescente et graphe, les opérations de coordination et les opérations de visualisation en tant qu'opérations interprétées par le système car actuellement quelques-unes ont été incorporées à la plate-forme sous forme d'options générales. En outre, la conception et développement d'une palette de navigation plus ergonomique que le menu offert à présent, se montre souhaitable. La combinaison d'aspects logiciels et matériels dans le cadre de la navigation textuelle est un objectif à long terme.

Enfin, je voudrais souligner que notre approche n'est pas contradictoire avec les recherches de base sur la sémantique ou la simulation de processus de compréhension réelle des textes. Bien au contraire, les résultats de ces recherches sont parfaitement exploitables par notre approche. D'une part, celles-ci peuvent enrichir les textes de deux manières : en encodant des phénomènes linguistiques complexes (par exemple des phénomènes discursifs) et en attribuant des annotations ou de méta-données. D'autre part, les recherches de base sur la compréhension réelle des textes peuvent alimenter la réflexion autour d'une méthodologie d'acquisition et expression de connaissances de visualisation et de navigation.

Annexes

Annexe I

Grammaire de la représentation des textes

Texte	→ Titre Tête Corps
Titre	→ listeUT | ε
Tête	→ listeConst | ε
Corps	→ listeUT
listeUT	→ UT | listeUT UT
listeConst	→ Const | listeConst Const
UT	→ Titre listeAnnot Chaîne
	| listeAnnot Chaîne
	| Titre listeAnnot listeUT
listeAnnot	→ Annotation | listeAnnot Annotation
Chaîne	→ Valeur
Annotation	→ Nom Valeur
Const	→ Ensemble | Séquence | Référence | Graphe
Ensemble	→ listeUT
Séquence	→ listeUT
Référence	→ UT UT
Graphe	→ Sommets Arcs
Sommets	→ listeUT
Arcs	→ UT UT | Arcs UT UT

Annexe II

DTD de l'encodage XML de la représentation des textes

```
<!ELEMENT Texte (Titre?, Tete?, Corps)>

<!ELEMENT Titre (UT+)>

<!ELEMENT Tete ((Ensemble | Sequence | Reference | Graphe)*)>
<!ELEMENT Ensemble (UTP+)>
  <!ATTLIST Ensemble Nro   CDATA #REQUIRED
                     Type  CDATA #IMPLIED
                     Titre CDATA #IMPLIED>
<!ELEMENT Sequence ((Debut, Fin, Annotation*) | (Annotation*, UTP+))>
  <!ATTLIST Sequence Nro   CDATA #REQUIRED
                     Type  CDATA #IMPLIED
                     Titre CDATA #IMPLIED>
<!ELEMENT Debut (UTP)>
<!ELEMENT Fin   (UTP)>
<!ELEMENT Reference (Source, Cible)>
  <!ATTLIST Reference   Nro   CDATA #REQUIRED
                        Type  CDATA #IMPLIED
                        Titre CDATA #IMPLIED>
<!ELEMENT Source (UTP)>
<!ELEMENT Cible  (UTP)>
<!ELEMENT Graphe  EMPTY >
<!ELEMENT UTP  EMPTY >
  <!ATTLIST UTP Type  CDATA  #REQUIRED
                Nro   CDATA  #REQUIRED
                Rang  CDATA  #IMPLIED>
```

```
<!ELEMENT Corps  (UT+)>
<!ELEMENT UT ((Titre?, Annotation*, UT+) | (Titre?, Annotation*, Chaine))>
  <!ATTLIST UT  Type  CDATA  #REQUIRED
                Nro   CDATA  #REQUIRED
                Rang  CDATA  #IMPLIED>
<!ELEMENT Chaine   (#PCDATA)>
<!ELEMENT Annotation (#PCDATA)>
  <!ATTLIST Annotation  Nom   CDATA  #REQUIRED>
```

Annexe III

Exemples de textes

BertinRST5

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE Texte SYSTEM "DocumentNaviTexte.dtd">
<Texte>
<Titre>
  <UT Type="Phrase" Nro="1">
    <Chaine> EVOLUTION DES ETUDES SUR LA COMPARAISON DES RISQUES DES DI-
    VERSES ENERGIES POUR LA PRODUCTION D 'ELECTRICITE</Chaine>
  </UT>
</Titre>
<Corps>
  <UT Type="Section" Nro="1" Rang="1">
    <Titre>
      <UT Type="Globale Titre" Nro="1">
        <Annotation Nom="Titre_etiqueté">Thématique</Annotation>
        <UT Type="Defaut" Nro="1">
          <Chaine> Importance de l'</Chaine>
        </UT>
        <UT Type= "SN" Nro="1">
          <Annotation Nom="Descripteur">Thème</Annotation>
          <Chaine>effet de serre</Chaine>
        </UT>
        <UT Type="Defaut" Nro="2">
          <Chaine>par le Dr Michel Bertin</Chaine>
        </UT>
      </UT>
  </Titre>
  <UT Type="Phrase" Nro="2">
    <Annotation Nom="Phrase_etiquetée">Préparation</Annotation>
```

```
   <Chaine> Les publications sur la comparaison des risques des diverses
   énergies utilisées pour produire l'électricité ont commencé il y a en-
   viron trente ans. Elles ont surtout été à la mode entre 1975 et 1985.
   Leur but était de comparer l'importance des risques dus à l'emploi du
   " nucléaire " par rapport à ceux dus à l'utilisation d'autres énergies
   : elles concluaient toutes que, pour reproduire de l'électricité, les
   risques de l'industrie nucléaire étaient bien moindres que ceux du
   charbon ou du pétrole et du même ordre de grandeur que ceux du gaz na-
   turel.</Chaine>

</UT>

<UT Type="Phrase" Nro="3">

   <Annotation Nom="Phrase_etiquetée">Démonstration</Annotation>

   <Chaine> Les publications récentes affirment que la hiérarchie indi-
   quée ci-dessus, est toujours valable bien que l'évaluation des consé-
   quences de l'accident de Tchernobyl ne soit pas encore termi-
   née.</Chaine>

</UT>

<UT Type="Phrase" Nro="4">

   <Annotation Nom="Phrase_etiquetée">Contraste</Annotation>

   <Chaine> Cependant ces études ne paraissent pas actuellement devoir
   nous aider à progresser encore beaucoup.</Chaine>

</UT>

<UT Type="Phrase" Nro="5">

   <Annotation Nom="Phrase_etiquetée">Interprétation</Annotation>

   <Chaine> Leurs faiblesses sont en effet importantes :</Chaine>

</UT>

<UT Type="Item" Nro="1">

   <Chaine>  - manque ou insuffisance de données, incertitudes sur les
   qualités de chacune et utilisation de données de valeurs très diffé-
   rentes ;</Chaine>

</UT>

<UT Type="Item" Nro="2">

   <Chaine>  - utilisation de critères trop simplistes pour juger des ef-
   fets sur la santé : les décès ou les journées de vies perdues ne suf-
   fisent pas pour rendre compte de la majorité des effets sur la santé ;
   </Chaine>

</UT>

<UT Type="Item" Nro="3">

   <Chaine> - variabilité très grande des résultats suivant les condi-
   tions de production et d'utilisation des énergies, etc.</Chaine>

</UT>

<UT Type="Phrase" Nro="6">

   <Annotation Nom="Phrase_etiquetée">Evaluation</Annotation>

   <Chaine> C'est évident pour le charbon suivant les conditions d'ex-
   traction et d'utilisation. C'est aussi évident pour le nucléaire si on
   tient compte de la sûreté. </Chaine>

</UT>

<UT Type="Phrase" Nro="7">
```

```
      <Annotation Nom="Phrase_etiquetée">Interprétation</Annotation>
      <Chaine> Les conclusions de ces études ont été présentées comme très
      générales alors qu'elles traduisent plutôt des situations particuliè-
      res, telles que celles d'un pays à une époque donnée, ce qui en soi
      est cependant déjà un résultat important.</Chaine>
    </UT>
  </UT>

  <UT Type="Section" Nro="2" Rang="1">
    <Titre>
      <UT Type="Globale Titre" Nro="2">
        <Annotation Nom="Titre_etiqueté">Thématique</Annotation>
        <Chaine> Prise en compte des problèmes d'environnement et des effets
        climatiques</Chaine>
      </UT>
    </Titre>
    <UT Type="Phrase" Nro="8">
      <Annotation Nom="Phrase_etiquetée">Préparation</Annotation>
      <UT Type="Defaut" Nro="1">
        <Chaine> Elles doivent être complétées, entre autres choses, par la
        prise en compte de tous les problèmes d'environnement ayant une di-
        mension à l'échelle de plusieurs pays, d'un continent ou de la terre
        entière : pluies acides, dépérissement des forêts, pollutions trans-
        frontières par les dérivés soufrés ou azotés et surtout </Chaine>
      </UT>
      <UT Type= "SN" Nro="1">
        <Annotation Nom="Descripteur">Thème</Annotation>
        <Chaine> effet de serre. </Chaine>
      </UT>
      <UT Type="Defaut" Nro="2">
        <Chaine> Ce dernier est d'actualité.</Chaine>
      </UT>
    </UT>
    <UT Type="Phrase" Nro="9">
      <Annotation Nom="Phrase_etiquetée">Evaluation</Annotation>
      <Chaine> Il concerne toute l'atmosphère terrestre ; ce n'est donc pas
      un problème à résoudre au plan local ou régional comme beaucoup d'au-
      tres problèmes de pollution atmosphérique. La diffusion plus ou moins
      rapide du " polluant " ne peut améliorer la situation. C'est un pro-
      blème dont l'évolution est lente et qui oblige à une politique à très
      long terme. C'est une pollution différente des pollutions classiques ;
      elle agit par le cumul des substances modifiant le climat. Elle n'agit
      pas directement sur l'homme. Elle ne peut donc être intégrée dans les
      comparaisons classiques des risques des diverses énergies.</Chaine>
    </UT>
    <UT Type="Phrase" Nro="10">
      <Annotation Nom="Phrase_etiquetée">Préparation</Annotation>
```

```
    <UT Type= "SN" Nro="2">
      <Annotation Nom="Descripteur">Thème</Annotation>
      <Chaine> L'effet de serre</Chaine>
    </UT>
    <UT Type="Defaut" Nro="3">
      <Chaine> est dû surtout au Co2 mais aussi au méthane, aux oxydes
      d'azote, aux hydrocarbures imbrûlés qui sont tous liés d'abord aux
      combustions des combustibles carbonés quand ils sont d'origine hu-
      maine ; les chlorofluorocarbones ou CFC jouent aussi leur rôle.Tous
      les combustibles carbonés y contribuent : charbon, lignite, produits
      pétroliers, gaz naturel mais encore le bois et, plus généralement,
      la biomasse, la combustion des déchets.</Chaine>
    </UT>
  </UT>
  <UT Type="Phrase" Nro="11">
    <Annotation Nom="Phrase_etiquetée">Démonstration</Annotation>
    <Chaine> Une remarque essentielle est que parmi les énergies actuelle-
    ment utilisées à grande échelle seul le nucléaire et l'hydraulique ne
    sont pas concernés. Ceci est notre première conclusion.</Chaine>
  </UT>
</UT>

<UT Type="Section" Nro="3" Rang="1">
  <Titre>
    <UT Type="Globale Titre" Nro="2">
      <Annotation Nom="Titre_etiqueté">Thématique</Annotation>
      <UT Type="Defaut" Nro="4">
        <Chaine> Quelles sont les conséquences de </Chaine>
      </UT>
      <UT Type= "SN" Nro="3">
        <Annotation Nom="Descripteur">Thème</Annotation>
        <Chaine>l'effet de serre ?</Chaine>
      </UT>
    </UT>
  </Titre>
  <UT Type="Phrase" Nro="12">
    <Annotation Nom="Phrase_etiquetée">Préparation</Annotation>
    <Annotation Nom="Phrase_etiquetée">Circonstance</Annotation>
    <Chaine> L'augmentation progressive du taux de Co2 dans l'atmosphère
    est certaine : En 1760 (carottes de glace) le taux était de l'ordre de
    280 ppm, 315 ppm en 1958 et 355 ppm en 1980. Elle est constatée par-
    tout dans le monde : même au pôle sud, par exemple.</Chaine>
  </UT>
  <UT Type="Phrase" Nro="13">
    <Annotation Nom="Phrase_etiquetée">Démonstration</Annotation>
```

```
    <Chaine> C'est donc bien un problème mondial.</Chaine>
  </UT>
  <UT Type="Phrase" Nro="14">
    <Annotation Nom="Phrase_etiquetée">Evaluation</Annotation>
    <UT Type="Defaut" Nro="5">
      <Chaine> L'élévation lente mais inéluctable de la température atmos-
      phérique parait quasiment certaine à beaucoup d'experts mais on
      ignore si cet accroissement est lié effectivement à</Chaine>
    </UT>
    <UT Type= "SN" Nro="4">
      <Annotation Nom="Descripteur">Thème</Annotation>
      <Chaine> l'effet de serre </Chaine>
    </UT>
    <UT Type="Defaut" Nro="6">
      <Chaine> dû aux activités humaines ou si il reste dans la limite des
      fluctuations climatiques constatées maintes fois. Les conséquences
      pour l'environnement ne sont que partiellement prévisibles. Les
      conséquences pour la santé sont inconnues mais des modifications des
      conditions de vie, du climat, de l'agriculture sont possi-
      bles.</Chaine>
    </UT>
  </UT>
  <UT Type="Phrase" Nro="15">
    <Annotation Nom="Phrase_etiquetée">Contraste</Annotation>
    <Chaine> Malgré et à cause de toutes ces incertitudes, dont les ex-
    perts se font l'écho, la nature et l'ampleur des conséquences possi-
    bles devraient amener à réviser les politiques énergétiques : d'ail-
    leurs les producteurs d'énergies fossiles qui, à énergie égale, pro-
    duisent le moins de co 2, revendiquent déjà un rôle accru : c'est le
    cas du gaz naturel par rapport au charbon (30% de moins de production
    de Co2 à énergie égale produite) même si le gaz ne fait que retarder
    et de bien peu l'augmentation des teneurs en Co2.</Chaine>
  </UT>
  <UT Type="Phrase" Nro="16">
    <Annotation Nom="Phrase_etiquetée">Démonstration</Annotation>
    <UT Type="Defaut" Nro="7">
      <Chaine> Notre deuxième conclusion, est que, à cause de l'</Chaine>
    </UT>
    <UT Type= "SN" Nro="5">
      <Annotation Nom="Descripteur">Thème</Annotation>
      <Chaine> effet de serre</Chaine>
    </UT>
    <UT Type="Defaut" Nro="8">
      <Chaine>, l'intérêt de développer l'électronucléaire est devenu évi-
      dent à un certain nombre d'hommes politiques, d'industriels et de
      scientifiques de disciplines diverses. Par contre le développement
      de la production d'électricité à partir des énergies fossiles :
```

```
        charbon et gaz naturel par exemple, va à l'encontre de l'objectif
        souhaité.</Chaine>
      </UT>
    </UT>
</UT>


<UT Type="Section" Nro="4" Rang="1">
    <Titre>
      <UT Type="Globale Titre" Nro="4">
        <Annotation Nom="Titre_etiqueté">Thématique</Annotation>
        <Chaine> Solutions</Chaine>
      </UT>
    </Titre>
    <UT Type="Phrase" Nro="17">
      <Annotation Nom="Phrase_etiquetée">Préparation</Annotation>
      <UT Type="Defaut" Nro="8.00">
        <Chaine> Le problème est donc de réduire ou de stabiliser la produc-
        tion   des gaz favorisant l'</Chaine>
      </UT>
      <UT Type= "SN" Nro="6">
        <Annotation Nom="Descripteur">Thème</Annotation>
        <Chaine> effet de serre</Chaine>
      </UT>
      <UT Type="Defaut" Nro="9">
        <Chaine>. Le cas des CFC (chlorofluorocarbones) est particulier ;
        tout le monde convient qu'il faut en arrêter la production. Pour les
        autres, la solution est de limiter l'utilisation des énergies pro-
        duisant des gaz à</Chaine>
      </UT>
      <UT Type= "SN" Nro="7">
        <Annotation Nom="Descripteur">Thème</Annotation>
        <Chaine> effet de serre</Chaine>
      </UT>
      <UT Type="Defaut" Nro="10.00">
        <Chaine>, le principal étant le CO2.</Chaine>
      </UT>
    </UT>
</UT>


<UT Type="Section" Nro="5" Rang="1">
    <Titre>
      <UT Type="Globale Titre" Nro="5">
        <Annotation Nom="Titre_etiqueté">Thématique</Annotation>
```

```
      <Chaine> Quels sont les facteurs positifs qu'on peut rete-
      nir?</Chaine>
    </UT>
</Titre>
<UT Type="Phrase" Nro="18">
    <Annotation Nom="Phrase_etiquetée">Préparation</Annotation>
    <Chaine> Le freinage de l'augmentation de la production de Co2 est dé-
    jà commencé dans les pays industrialisés les plus développés, comme le
    montre le tableau 1 publié par l'OCDE.</Chaine>
</UT>
<UT Type="Phrase" Nro="19">
    <Annotation Nom="Phrase_etiquetée">Interprétation</Annotation>
    <Annotation Nom="Phrase_etiquetée">Cause_Délibérée</Annotation>
    <Chaine> Ceci serait dû en partie aux centrales nucléaires.</Chaine>
</UT>
<UT Type="Phrase" Nro="20">
    <Annotation Nom="Phrase_etiquetée">Circonstance</Annotation>
    <Chaine> Aux USA de 1973 à 1990, Ies centrales thermiques classiques
    qu'il aurait fallu faire fonctionner, si les centrales nucléaires
    n'avaient pas existé, auraient rejeté 4,6 milliards de tonnes de co 2,
    56 millions de tonnes de So2 et 18 millions de tonnes de NOx.</Chaine>
</UT>
<UT Type="Phrase" Nro="21">
    <Annotation Nom="Phrase_etiquetée">Circonstance</Annotation>
    <Chaine> En France les émissions de Co2 ont diminué de 30% de 1980 à
    1988 grâce essentiellement aux centrales nucléaires.</Chaine>
</UT>
<UT Type="Phrase" Nro="22">
    <Annotation Nom="Phrase_etiquetée">Evaluation</Annotation>
    <Chaine> Bien entendu parallèlement d'importantes économies d'énergie
    ont été faites ne serait-ce qu'en développant de nouvelles techniques
    d'utilisation. De toute façon il sera évidemment nécessaire de freiner
    l'augmentation de la consommation d'énergie puisque les réserves de
    combustibles fossiles sont limitées : elles représentent quelques di-
    zaines d'années pour le pétrole, 60 à 100 ans pour le gaz naturel et
    plusieurs siècles pour le charbon.</Chaine>
</UT>
<UT Type="Phrase" Nro="23">
    <Annotation Nom="Phrase_etiquetée">Contraste</Annotation>
    <Chaine> Cependant il ne faut pas se faire d'illusion. Les représen-
    tants du ministre de l'industrie estiment qu'il est totalement irréa-
    liste, à cause de l'évolution du Tiers Monde, de vouloir stabiliser la
    production de Co2 en l'an 2000 au même niveau qu'en 1990 contrairement
    à ce qu'ont affirmé, il y a quelques mois, les chefs d'États des pays
    de la Communauté Européenne.</Chaine>
</UT>
<UT Type="Phrase" Nro="24">
```

```
   <Annotation Nom="Phrase_etiquetée">Justification</Annotation>

   <Chaine> Les facteurs qui vont à l'encontre de ce projet sont nombreux. Les pays du Tiers Monde produisent beaucoup moins de Co2 par habitant que les pays industrialisés. L'écart entre l'Inde et les États Unis est de 1 à 50 (Tableau 2).</Chaine>

</UT>

<UT Type="Phrase" Nro="25">

   <Annotation Nom="Phrase_etiquetée">Contraste</Annotation>

   <Annotation Nom="Phrase_etiquetée">Séquence</Annotation>

   <Chaine> Or le développement économique des pays du Tiers Monde passe par le développement de la consommation d'énergie. Le manque de moyens financiers fait que la protection de l'environnement n'est pas une de leurs priorités ; le développement de l'industrie lourde en Mandchourie dans les années 1970 est l'exemple d'un développement industriel très rapide mais sans souci de la protection de l'environnement.</Chaine>

</UT>

<UT Type="Phrase" Nro="26">

   <Annotation Nom="Phrase_etiquetée">Justification</Annotation>

   <Annotation Nom="Phrase_etiquetée">Séquence</Annotation>

   <Chaine> Ce qui s'est passé dans les pays de l'Est en est aussi la preuve.</Chaine>

</UT>

<UT Type="Phrase" Nro="27">

   <Annotation Nom="Phrase_etiquetée">Contraste</Annotation>

   <Annotation Nom="Phrase_etiquetée">Séquence</Annotation>

   <Chaine> D'autre part les économies d'énergie dans un but de protection de l'environnement sont un plus que seuls les pays riches peuvent s'offrir et il faut être très riche pour acheter de l'électricité à l'étranger afin de ne pas porter atteinte à son propre environnement.</Chaine>

</UT>

<UT Type="Phrase" Nro="28">

   <Annotation Nom="Phrase_etiquetée">Séquence</Annotation>

   <Chaine> En même temps il faut tenir compte de l'augmentation extrêmement rapide de la population mondiale (1991 : 5,4 milliards d'habitants, prévision 2025 : 8,6 milliards d'habitants, 2100 : 12 milliards d'habitants) et ceci, en raison essentiellement de la population du Tiers Monde : en effet en 1985 un tiers de la population vivait dans les pays développés et en l'an 2000 il n'y en aura plus qu'un sixième.</Chaine>

</UT>

<UT Type="Phrase" Nro="29">

   <Annotation Nom="Phrase_etiquetée">Démonstration</Annotation>

   <Chaine> Donc la population du Tiers Monde, pour se développer économiquement, consommera de plus en plus d'énergie et emploiera les énergies qui sont le plus facilement utilisables par elle, sans pouvoir consacrer les moyens nécessaires à la protection de l'environnement.</Chaine>

</UT>
```

```
<UT Type="Phrase" Nro="30">
  <Annotation Nom="Phrase_etiquetée">Elaboration</Annotation>
  <UT Type="Defaut" Nro="10.0004">
    <Chaine> Il s'agit d'abord de la biomasse qui contribue doublement à
    </Chaine>
  </UT>
  <UT Type= "SN" Nro="8.00">
    <Annotation Nom="Descripteur">Thème</Annotation>
    <Chaine>l'effet de serre</Chaine>
  </UT>
  <UT Type="Defaut" Nro="10.001">
    <Chaine>. La combustion de la biomasse, au premier rang de laquelle
    se trouve le bois, produit du Co2 puisqu'il s'agit encore de combus-
    tible carbonés (la combustion des ordures aussi) et en même temps
    elle accroît la déforestation, facteur essentiel aussi de
    l'</Chaine>
  </UT>
  <UT Type= "SN" Nro="8.001">
    <Annotation Nom="Descripteur">Thème</Annotation>
    <Chaine>effet de serre</Chaine>
  </UT>
  <UT Type="Defaut" Nro="11">
    <Chaine>. La consommation exagérée de bois et de végétaux pour la
    combustion a entraîné déjà un manque aigu de ressources énergétiques
    dans certains pays, tels que ceux du Sahel. Le charbon qui, à éner-
    gie égale, produit le plus de Co2 est aussi assez facilement utili-
    sable par les pays du Tiers Monde.Le pétrole et le gaz naturel ne
    viennent qu'après. Il faudrait donc utiliser toutes les énergies
    disponibles car pour empêcher l'</Chaine>
  </UT>
  <UT Type= "SN" Nro="8.002">
    <Annotation Nom="Descripteur">Thème</Annotation>
    <Chaine>effet de serre</Chaine>
  </UT>
  <UT Type="Defaut" Nro="12">
    <Chaine>il faut que l'emploi des énergies qui le favorise soit limi-
    té de façon que le CO2 qu'elles produisent ne dépasse pas ce qui
    peut être résorbé par le cycle du carbone. Donc il sera évidemment
    nécessaire de développer l'électronucléaire en particulier dans Ies
    pays économiquement développés, ceci n'en déplaise à certains, pour
    des raisons écologiques. Comme l'a calculé le ministère de l'indus-
    trie, si les autres pays industrialisés avaient des programmes élec-
    tronucléaires comparables à celui de la France, ils pourraient ré-
    duire de 24 % leurs émissions d'anhydride carbonique.</Chaine>
  </UT>
</UT>
<UT Type="Phrase" Nro="31">
  <Annotation Nom="Phrase_etiquetée">Démonstration</Annotation>
```

```
  <Chaine> Donc, pour que le développement de I 'electronucléaire ait une
  influence significative, il faudra qu'il soit très important. Ceci est
  notre troisième conclusion.</Chaine>

</UT>

<UT Type="Phrase" Nro="32">

  <Annotation Nom="Phrase_etiquetée">Justification</Annotation>

  <Chaine> Le club de Rome, qui n'a jamais débordé d'enthousiasme pour
  le nucléaire, a changé de cap :</Chaine>

</UT>

<UT Type="Segment" Nro="1">

  <Annotation Nom="Segment_etiqueté">Citation</Annotation>

  <UT Type="Phrase" Nro="33">

    <Chaine> "Il semble que nous devrions nous préparer à affronter une
    situation critique dans quelques dizaines d'années, lorsque les dan-
    gers que comporte le réchauffement de la planète nous obligeraient à
    réduire radicalement notre consommation de combustibles fossiles,
    car nous n'aurons pas le choix.Dans ces conditions la fission nu-
    cléaire pourrait nous offrir la seule possibilité de pallier la si-
    tuation (cité dans le bulletin de l'AIEA, 1/19 92) ".</Chaine>

  </UT>

</UT>

<UT Type="Phrase" Nro="34">

  <Annotation Nom="Phrase_etiquetée">Elaboration</Annotation>

  <Chaine> Et on peut pousser le raisonnement plus loin. L'uranium avec
  les techniques et au coût actuel, ne représente qu'un potentiel éner-
  gétique modeste. Le surgénérateur est peut-être la seule filière à la
  mesure des besoins, sans qu'il soit nécessaire d'évoquer l'élimination
  des nucléides à longue période pour le justifier.</Chaine>

</UT>

<UT Type="Phrase" Nro="35">

  <Annotation Nom="Phrase_etiquetée">Démonstration</Annotation>

  <Chaine> C'est donc pour des raisons écologiques qu'on est amené à
  plaider en faveur du nucléaire et du surgénérateur.</Chaine>

</UT>

<UT Type="Phrase" Nro="36">

  <Annotation Nom="Phrase_etiquetée">Solution</Annotation>

  <Chaine> Nous sommes très loin de notre point de départ : les études
  sur les comparaisons des risques des différentes énergies pour pro-
  duire de l'électricité. Il faudrait les reprendre entièrement en élar-
  gissant leur champ d'investigation à la protection du milieu naturel
  et de l'environnement. Il n'est pas possible de trouver un critère
  unique pour juger des effets sur l'homme.</Chaine>

</UT></UT></Corps></Texte>
```

Amnistie Fiscale

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE Texte SYSTEM "DocumentNaviTexte.dtd">
<Texte>
<Titre>
  <UT Type="Titre" Nro="2">
    <Annotation Nom="Titre_etiqueté"></Annotation>
    <Chaine>L'amnistie fiscale n'est pas immorale !</Chaine>
  </UT>
</Titre>
<Tete />
<Corps>
<UT Type="Paragraphe" Nro="1">
  <UT Type="segment" Nro="0">
    <Annotation Nom="annotation">Thème</Annotation>
    <UT Type="RD" Nro="1">
      <Annotation Nom="Référent Discursif">RD1</Annotation>
      <Chaine>La taxe sur les fonds rapatriés en France</Chaine>
    </UT>
  </UT>
</UT>
<UT Type="Paragraphe" Nro="1.1">
  <UT Type="segment" Nro="1">
    <Annotation Nom="annotation">Attitude Protagoniste</Annotation>
    <UT Type="segment" Nro="2">
      <Chaine>L'amnistie fiscale</Chaine>
    </UT>
    <UT Type="marqueur" Nro="1">
      <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
      <Chaine>n'est pas</Chaine>
    </UT>
    <UT Type="segment" Nro="3">
      <Chaine> immorale ! </Chaine>
    </UT>
  </UT>
  <UT Type="segment" Nro="4">
  <Annotation Nom="annotation">Thème</Annotation>
    <UT Type="segment" Nro="5">
```

```
        <Chaine>Le gouvernement de Jean-Pierre Raffarin étudie l'opportunité
        d'instituer </Chaine>
      </UT>
      <UT Type="RD" Nro="2">
        <Annotation Nom="Référent Discursif">RD1</Annotation>
        <Chaine>une taxe sur les fonds placés à l'étranger et rapatriés en
        France.</Chaine>
      </UT>
      <UT Type="RD" Nro="3">
        <Annotation Nom="Référent Discursif">RD2</Annotation>
        <UT Type="segment" Nro="6">
          <Chaine>Le produit de</Chaine>
        </UT>
        <UT Type="RD" Nro="4">
          <Annotation Nom="Référent Discursif">RD1</Annotation>
          <Chaine>cette taxe</Chaine>
        </UT>
      </UT>
      <UT Type="segment" Nro="7">
        <Chaine>financerait le plan que vient de dévoiler son ministre de la
        Cohésion sociale.</Chaine>
      </UT>
    </UT>
  <UT Type="segment" Nro="8">
    <Annotation Nom="annotation">Attitude Antagoniste</Annotation>
    <UT Type="segment" Nro="9">
      <Chaine>De nombreux responsables politiques ont</Chaine>
    </UT>
    <UT Type="marqueur" Nro="2">
      <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
      <Chaine>manifesté leur hostilité à</Chaine>
    </UT>
    <UT Type="RD" Nro="5">
      <Annotation Nom="Référent Discursif">RD1</Annotation>
      <Chaine>une telle mesure.</Chaine>
    </UT>
    <UT Type="RD" Nro="6">
      <Annotation Nom="Référent Discursif">RD1</Annotation>
      <Chaine>Elle</Chaine>
    </UT>
    <UT Type="marqueur" Nro="3">
      <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
```

```
      <Chaine>serait</Chaine>
    </UT>
    <UT Type="segment" Nro="10">
      <Chaine>inefficace et, surtout, immorale car </Chaine>
    </UT>
    <UT Type="RD" Nro="7">
      <Annotation Nom="Référent Discursif">RD1</Annotation>
      <Chaine>elle</Chaine>
     </UT>
     <UT Type="marqueur" Nro="4">
      <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
      <Chaine>blanchirait</Chaine>
    </UT>
    <UT Type="segment" Nro="11">
      <Chaine> les "criminels en col blanc".</Chaine>
    </UT>
  </UT>
  <UT Type="segment" Nro="12">
    <Annotation Nom="annotation">Thème</Annotation>
    <UT Type="segment" Nro="13">
      <Chaine>Les experts du gouvernement cherchent à déterminer les modali-
      tés optimales d'</Chaine>
    </UT>
    <UT Type="RD" Nro="8">
      <Annotation Nom="Référent Discursif">RD1</Annotation>
      <Chaine>une telle opération.</Chaine>
     </UT>
  </UT>
  <UT Type="segment" Nro="14">
    <Annotation Nom="annotation">Sous-Thème_1</Annotation>
     <UT Type="segment" Nro="15">
      <Chaine>Les cas des pays voisins</Chaine>
     </UT>
     <UT Type="RD" Nro="9">
      <Annotation Nom="Référent Discursif">RD1</Annotation>
      <Chaine>l'</Chaine>
    </UT>
    <UT Type="segment" Nro="16">
      <Chaine>ayant déjà pratiquée sont étudiés.</Chaine>
    </UT>
  </UT>
  </UT>
```

```
<UT Type="Paragraphe" Nro="2">
  <UT Type="segment" Nro="17">
    <Annotation Nom="annotation">Sous-Thème_1.1</Annotation>
    <UT Type="segment" Nro="18">
      <Chaine>En Allemagne, avec un prélèvement libératoire de 25 ou
      35%,</Chaine>
    </UT>
    <UT Type="RD" Nro="9.1">
      <Annotation Nom="Référent Discursif">RD1</Annotation>
      <Chaine>elle</Chaine>
    </UT>
    <UT Type="segment" Nro="19">
      <Chaine>rapporterait moins qu'espéré initialement : 1,5 milliard
      d'euros selon les instituts de conjoncture, contre les 5 milliards
      attendus.</Chaine>
    </UT>
  </UT>


<UT Type="segment" Nro="20">
  <Annotation Nom="annotation">Sous-Thème_1.2</Annotation>
  <Chaine> En Italie, taxés à 2,5%, 60 milliards d'euros ont été régulari-
  sés (et acquittée une taxe d'environ 1,5 milliard).</Chaine>
</UT>


<UT Type="segment" Nro="21">
  <Annotation Nom="annotation">Attitude Protagoniste</Annotation>
  <UT Type="marqueur" Nro="5">
    <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
    <Chaine>Comme on pouvait s'y attendre,</Chaine>
  </UT>
  <UT Type="segment" Nro="22">
    <Chaine> plus le taux de</Chaine>
  </UT>
  <UT Type="RD" Nro="13">
    <Annotation Nom="Référent Discursif">RD1</Annotation>
    <Chaine>la taxe libératoire</Chaine>
  </UT>
  <UT Type="segment" Nro="23">
    <Chaine>est bas, plus les montants rapatriés sont élevés et inverse-
    ment.</Chaine>
  </UT>
</UT>
```

```
<UT Type="segment" Nro="24">
  <Annotation Nom="annotation">Attitude Protagoniste</Annotation>
  <UT Type="marqueur" Nro="6">
    <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
    <Chaine> Plus surprenant est le fait que</Chaine>
  </UT>
  <UT Type="RD" Nro="2.1">
    <Annotation Nom="Référent Discursif">RD2</Annotation>
    <Chaine>le produit récolté</Chaine>
  </UT>
  <UT Type="segment" Nro="25">
    <Chaine>est le même en Allemagne qu'en Italie, pays du même ordre de
    grandeur que la France : soit environ 1,5 milliard d'euros.</Chaine>
  </UT>
</UT>
<UT Type="segment" Nro="26">
  <Annotation Nom="annotation">Attitude Antagoniste</Annotation>
  <UT Type="RD" Nro="2.2">
    <Annotation Nom="Référent Discursif">RD2</Annotation>
    <Chaine>Une telle somme</Chaine>
  </UT>
  <UT Type="segment" Nro="27">
    <Chaine> ne </Chaine>
  </UT>
  <UT Type="marqueur" Nro="7">
    <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
    <Chaine>contribuerait</Chaine>
  </UT>
  <UT Type="marqueur" Nro="8">
    <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
    <Chaine>, il est vrai</Chaine>
  </UT>
  <UT Type="segment" Nro="28">
    <Chaine>, que modestement au financement du plan de cohésion sociale
    dont le coût, en cinq ans, se monterait à 13 milliards d'eu-
    ros.</Chaine>
  </UT>
  <UT Type="marqueur" Nro="9">
    <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
    <Chaine> Mais</Chaine>
  </UT>
  <UT Type="segment" Nro="29">
```

```
      <Chaine>faut-il</Chaine>
    </UT>
    <UT Type="marqueur" Nro="10">
      <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
      <Chaine>pour autant</Chaine>
    </UT>
    <UT Type="segment" Nro="30">
      <Chaine>déclarer</Chaine>
    </UT>
    <UT Type="RD" Nro="14">
      <Annotation Nom="Référent Discursif">RD1</Annotation>
      <Chaine>la mesure envisagée</Chaine>
    </UT>
   <UT Type="segment" Nro="31">
      <Chaine>par M. Raffarin inefficace</Chaine>
    </UT>
    <UT Type="marqueur" Nro="11">
      <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
      <Chaine>?</Chaine>
     </UT>
</UT>
<UT Type="segment" Nro="32">
  <Annotation Nom="annotation">Attitude Protagoniste</Annotation>
   <UT Type="marqueur" Nro="12">
      <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
      <Chaine>Non</Chaine>
  </UT>
  <UT Type="marqueur" Nro="13">
      <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
      <Chaine>car</Chaine>
  </UT>
  <UT Type="segment" Nro="33">
      <Chaine>d'autres avantages</Chaine>
  </UT>
  <UT Type="RD" Nro="15">
      <Annotation Nom="Référent Discursif">RD1</Annotation>
      <Chaine>en </Chaine>
  </UT>
  <UT Type="segment" Nro="34">
      <Chaine>résulteraient. Des capitaux importés, qu'ils soient prêtés ou
      investis, créent des emplois. </Chaine>
   </UT>
```

```
</UT>
</UT>
<UT Type="Paragraphe" Nro="3">
  <Annotation Nom="annotation">Conséquence</Annotation>
  <UT Type="segment" Nro="35">
    <UT Type="marqueur" Nro="14">
      <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
      <Chaine>C'est pour cette raison que</Chaine>
    </UT>
    <UT Type="segment" Nro="36">
      <Chaine>le gouvernement subventionne des firmes étrangères pour
      qu'elles investissent en France. Avec</Chaine>
    </UT>
    <UT Type="RD" Nro="16">
      <Annotation Nom="Référent Discursif">RD1</Annotation>
      <Chaine>la mesure envisagée</Chaine>
    </UT>
    <UT Type="segment" Nro="37">
      <Chaine>,le Trésor ferait rentrer des capitaux tout en bénéficiant
      de</Chaine>
    </UT>
    <UT Type="RD" Nro="2.4">
      <Annotation Nom="Référent Discursif">RD2</Annotation>
      <Chaine>rentrées fiscales</Chaine>
    </UT>
    <UT Type="segment" Nro="38">
      <Chaine>. Il s'agit de la taxe directement prélevée mais aussi de
      l'impôt sur les revenus futurs générés par ces capitaux, de l'ISF
      acquitté par leurs possesseurs et, surtout, des droits de succession
      que les héritiers de ces derniers paieront nécessairement un
      jour.</Chaine>
    </UT>
    <UT Type="RD" Nro="2.5">
      <Annotation Nom="Référent Discursif">RD2</Annotation>
      <Chaine>Ces rentrées</Chaine>
    </UT>
    <UT Type="segment" Nro="39">
      <Chaine>représentent, au bas mot, une valeur d'au moins 30% des
      fonds rapatriés.</Chaine>
    </UT>
  </UT>
</UT>
<UT Type="Paragraphe" Nro="4">
  <UT Type="segment" Nro="40">
```

```
    <Annotation Nom="annotation">Sous-Thème_1.1</Annotation>
    <UT Type="marqueur" Nro="15">
      <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
      <Chaine>Si l'on en croit les expériences récentes</Chaine>
    </UT>
    <UT Type="segment" Nro="42">
      <Chaine>, pour collecter immédiatement un montant de l'ordre de 1,5
      milliard d'euros il est possible soit de fixer</Chaine>
    </UT>
    <UT Type="RD" Nro="17">
      <Annotation Nom="Référent Discursif">RD1</Annotation>
      <Chaine>la taxe</Chaine>
    </UT>
    <UT Type="segment" Nro="43">
      <Chaine>, comme en Allemagne, à 25% et faire rentrer</Chaine>
    </UT>
    <UT Type="RD" Nro="2.6">
      <Annotation Nom="Référent Discursif">RD2</Annotation>
      <Chaine>quelque 6 millions d'euros de capitaux,</Chaine>
    </UT>
  </UT>
  <UT Type="segment" Nro="44">
    <Annotation Nom="annotation">Sous-Thème_1.2</Annotation>
    <UT Type="segment" Nro="45">
      <Chaine>soit de</Chaine>
    </UT>
    <UT Type="RD" Nro="18">
      <Annotation Nom="Référent Discursif">RD1</Annotation>
      <Chaine>la</Chaine>
    </UT>
    <UT Type="segment" Nro="46">
      <Chaine> fixer, comme en Italie, à 2,5% et voir rentrer</Chaine>
    </UT>
    <UT Type="RD" Nro="2.7">
      <Annotation Nom="Référent Discursif">RD2</Annotation>
      <Chaine>60 millions d'euros</Chaine>
    </UT>
    <UT Type="segment" Nro="47">
      <Chaine>(avec, en France, l'existence d'un ISF qui risque de décou-
      rager certains candidats au retour). </Chaine>
    </UT>
   </UT>
```

```
<UT Type="segment" Nro="48">
 <Annotation Nom="annotation">Attitude Protagoniste</Annotation>
 <UT Type="segment" Nro="49">
   <Chaine>Compte tenu des autres dividendes fiscaux,</Chaine>
 </UT>
 <UT Type="marqueur" Nro="16">
   <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
   <Chaine>il est clair que</Chaine>
 </UT>
 <UT Type="segment" Nro="50">
   <Chaine>le choix du gouvernement devrait se porter sur</Chaine>
 </UT>
 <UT Type="RD" Nro="18bis">
   <Annotation Nom="Référent Discursif">RD1</Annotation>
   <Chaine>la taxe à 2,5%</Chaine>
 </UT>
 <UT Type="segment" Nro="51">
   <Chaine>qui maximise les entrées de capitaux. </Chaine>
 </UT>
</UT>
<UT Type="segment" Nro="52">
 <Annotation Nom="annotation">Conséquence</Annotation>
 <UT Type="segment" Nro="53">
   <Chaine>Le Trésor bénéficierait alors </Chaine>
 </UT>
 <UT Type="RD" Nro="2.8">
   <Annotation Nom="Référent Discursif">RD2</Annotation>
   <Chaine>d'une "manne" de l'ordre de20 milliards d'euros</Chaine>
 </UT>
 <UT Type="segment" Nro="54">
   <Chaine>, somme qu'il pourrait emprunter immédiatement sachant les
   capitaux revenus en France !</Chaine>
 </UT>
</UT>
<UT Type="segment" Nro="55">
   <Annotation Nom="annotation">Attitude Antagoniste</Annotation>
   <UT Type="marqueur" Nro="17">
     <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
     <Chaine> Reste la question</Chaine>
   </UT>
   <UT Type="segment" Nro="56">
```

```
        <Chaine>de l'immoralité de ce qui apparaît comme une amnistie fis-
        cale.</Chaine>
    </UT>
</UT>
<UT Type="segment" Nro="57">
    <Annotation Nom="annotation">Attitude Protagoniste</Annotation>
    <UT Type="marqueur" Nro="18">
        <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
        <Chaine>Rappelons que</Chaine>
    </UT>
    <UT Type="segment" Nro="58">
        <Chaine>tous les crimes et tous les délits bénéficient de périodes
        au-delà desquelles la justice renonce à poursuivre les coupa-
        bles.</Chaine>
    </UT>
</UT>
<UT Type="segment" Nro="59">
    <Annotation Nom="annotation">Spécification</Annotation>
    <Chaine>Seuls sont imprescriptibles les crimes contre l'humanité et...
    les délits associés aux sorties frauduleuses d'argent. </Chaine>
</UT>
<UT Type="segment" Nro="60">
   <Annotation Nom="annotation">Spécification</Annotation>
    <UT Type="marqueur" Nro="19">
        <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
        <Chaine>En effet</Chaine>
    </UT>
    <UT Type="segment" Nro="61">
        <Chaine>, l'argent sorti illégalement de France et placé à l'étran-
        ger est théoriquement assujetti chaque année au paiement d'impôts
        sur les plus-values et les bénéfices ainsi qu'à l'ISF. </Chaine>
    </UT>
</UT>
<UT Type="segment" Nro="62">
    <Annotation Nom="annotation">Concession</Annotation>
    <UT Type="marqueur" Nro="20">
        <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
        <Chaine> Même si</Chaine>
    </UT>
    <UT Type="segment" Nro="63">
        <Chaine> un fraudeur a "sorti" des capitaux il y a dix ou vingt ans
        et qu'il bénéficie de la prescription pour ce délit initial, il
        est</Chaine>
    </UT>
```

```
      <UT Type="marqueur" Nro="21">
        <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
        <Chaine>toujours</Chaine>
      </UT>
      <UT Type="segment" Nro="64">
        <Chaine>redevable au fisc français d'impôts dus pour les derniers
        exercices.</Chaine>
      </UT>
    </UT>
    <UT Type="segment" Nro="65">
      <UT Type="segment" Nro="66">
        <Annotation Nom="annotation">Conséquence</Annotation>
        <UT Type="segment" Nro="67">
          <Chaine>Il ne peut</Chaine>
        </UT>
        <UT Type="marqueur" Nro="22">
          <Annotation Nom="annotation">Marqueur Argumentatif</Annotation>
          <Chaine>donc</Chaine>
        </UT>
        <UT Type="segment" Nro="68">
          <Chaine>jamais bénéficier de la protection qu'offre la prescrip-
          tion à des auteurs de crimes plus importants.</Chaine>
        </UT>
      </UT>
    </UT>
    <UT Type="segment" Nro="69">
      <Annotation Nom="annotation">Attitude Protagoniste</Annotation>
      <UT Type="RD" Nro="19">
        <Annotation Nom="Référent Discursif">RD1</Annotation>
        <Chaine> L'opération qu'envisage M. Raffarin </Chaine>
      </UT>
      <UT Type="segment" Nro="70">
        <Chaine>ne ferait que réparer cette inégalité de traitement devant
        la loi et apporterait au Trésor des</Chaine>
      </UT>
      <UT Type="RD" Nro="2.9">
        <Annotation Nom="Référent Discursif">RD2</Annotation>
        <Chaine>ressources qui lui font cruellement défaut !  </Chaine>
      </UT>
    </UT>
  </UT></Corps></Texte>
```

Annexe IV

Grammaire du langage Sextant

module	→ Module (nomModule , { ensembleDV })
nomModule	→ valeur
ensembleDV	→ descriptionVue | descriptionVue , ensembleDV
descriptionVue	→ DV (typeVue , contenuVue, nomVue , première UT , { paramètres } , conditionFiltre , { opérationsVisualisation } , { opérationsNavigation } , { opérationsCoordination })
typeVue	→ linéaire | arborescente | graphe
contenuVue	→ chaînes lexicales | annotations
nomVue	→ valeur
premièreUT	→ condition
paramètres	→ (nomParamètre , valeurParamètre) , paramètres | ε
nomParamètre	→ valeur
valeurParamètre	→ valeur
conditionFiltre	→ condition | ε
opérationsVisualisation	→ opérationVisualisation , opérationsVisualisation | ε
opérationsNavigation	→ opérationNavigation , opérationsNavigation | ε
opérationsCoordination	→ opérationCoordination , opérationsCoordination | ε
condition	→ conditionSimple | conditionExiste | conditionHiérarchie | NON condition | condition ET condition | condition OU condition | (condition)

conditionExiste → existeAnnotations | existeChaîneLexicale | existeTitre | existeParent | existeFils

conditionHiérarchie → opérationCondHiérarchie (conditionSimple)

opérationCondHiérarchie → estParent | estFils | estFrère | estAscendant | estDescendant | contientDansTitre | estDansTitreDe

conditionSimple → UT (contrainteType , contrainteNro , contrainteRang , contraintesAnnotations , contrainteChaîne)

contrainteType → Type opérateur valeur | valeur opérateur Type | *

contrainteNro → Nro opérateur valeur | valeur opérateur Nro | *

contrainteRang → Rang opérateur valeur | valeur opérateur Rang | *

contrainteChaîne → ChaîneLexicale opérateur valeur | valeur opérateur ChaîneLexicale | *

opérateur → = | ≠ | < | > | ≤ | ≥ | estPréfixe | estSuffixe | estSousChaîne

contraintesAnnotations → { listeContrainteAnnotation } | *

listeContrainteAnnotation → contrainteAnnotation | contrainteAnnotation listeContrainteAnnotation

contrainteAnnotation → valeurExistence (valeurNom , valeurAnnotation)

valeurExistence → ∃ | ¬∃ | ε

valeurNom → valeur | *

valeurAnnotation → valeur | *

opérationsVisualisation → opérationVisualisation , opérationsVisualisation | ε

opérationVisualisation → miseEnRelief | aideContextuelle

miseEnRelief → merSimple | merComplexe

merComplexe → MER_C (nomMerComplexe , listeMerSimple)

nomMerComplexe → valeur

listeMerSimple → merSimple | merSimple , listeMerSimple

merSimple → MER_S (nomMerSimple , condition , { listeOpérationsTransformation })

nomMerSimple → valeur

listeOpérationsTransformation → opérationTransformation |
opérationTransformation ,
listeOpérationsTransformation

opérationTransformation → couleurPremierPlan (valeur) |
couleurArrièrePlan (valeur) |
taillePolice (valeur) |
ajouter (placement , valeur)

placement → préfixe | suffixe

aideContextuelle → AideContextuelle (typeAideContextuelle ,
condition , aMontrer)

typeAideContextuelle → infobull | vueDinamique

aMontrer → valeur | descriptionVue

opérationsNavigation → opérationNavigation , opérationsNavigation | ε

opérationNavigation → OpNav (nomOpérationNavigation ,
typeOpérationNavigation ,
source , cible , rapportSourceCible ,
{ arguments })

nomOpérationNavigation → valeur

typeOpérationNavigation → premier | dernier |
suivant | suivant [valeur] |
précédent | précédent [valeur]

source → condition

cible → condition

rapportSourceCible → { ensembleNomsAnnotations }

ensembleNomsAnnotations → nomAnnotation , ensembleNomsAnnotations | ε

nomAnnotation → valeur

arguments → nomArgument , arguments | ε

nomArgument → valeur

opérationsCoordination → opérationCoordination , opérationsCoordination
| ε

opérationCoordination → OpCoord (nomVueCible , mode)

nomVueCible	→ valeur
mode	→ **manuel** | **automatique**

Annexe V

DTD de l'encodage XML du langage Sextant

```
<!ELEMENT Module (DescriptionDeVue+)>
  <!ATTLIST Module Nom CDATA #IMPLIED>

<!ELEMENT DescriptionDeVue (PremiereUT?, Parametre*, Condition?,
                            Ops_Vis?, Ops_Nav?, Ops_Coord?)>
  <!ATTLIST DescriptionDeVue Nom CDATA #REQUIRED
                             Type (Lineaire|Arborescente|Graphe) #REQUIRED
                             Contenu (Chaines | Annotations) #REQUIRED>
<!ELEMENT PremiereUT (Condition)>
<!ELEMENT Parametre (#PCDATA)>
  <!ATTLIST Parametre Nom   CDATA  #IMPLIED>

<!ELEMENT Ops_Vis (MiseEnRelief | AideContextuelle)+>
<!ELEMENT MiseEnRelief EMPTY>
<!ELEMENT AideContextuelle EMPTY>

<!ELEMENT Ops_Nav (Op_Nav+)>
<!ELEMENT Op_Nav (Source, Cible, RelationSourceCible?, Arguments?)>
  <!ATTLIST Op_Nav   Type  (Premier|Suivant|Précédent|Dernier) #REQUIRED
                     Titre CDATA #REQUIRED
                     Rang CDATA "1">
<!ELEMENT Source (Condition)>
<!ELEMENT Cible (Condition)>
<!ELEMENT RelationSourceCible (RelationSC)+>
<!ELEMENT RelationSC (#PCDATA)>
<!ELEMENT Arguments (Argument)+>
```

```
<!ELEMENT Argument (#PCDATA)>

<!ELEMENT Ops_Coord (Op_Coord+)>
<!ELEMENT OpCoord EMPTY>

<!ELEMENT Condition ((UT) | (Condition)+)>
  <!ATTLIST Condition Type ( Simple | existeAnnotations |
                             existeChaîneLexicale | existeTitre |
                             existeParent | existeFils |
                             estParent | estFils | estAscendant |
                             estDescendant |  contientDansTitre |
                             estDansTitreDe | ET | OU | NON) #REQUIRED>

<!ELEMENT UT (Contrainte | ExisteAnnotation)*>
<!ELEMENT Contrainte EMPTY>
  <!ATTLIST   Contrainte arg1  CDATA  #REQUIRED
              op (EG | DF | INF | SUP | INFEG | SUPEG |
                   estPréfixe | estSuffixe | estSousChaîne) #REQUIRED
             arg2  CDATA  #REQUIRED>
<!ELEMENT ExisteAnnotation EMPTY>
  <!ATTLIST   ExisteAnnotation valeurExistence   (oui | non)   "oui"
              nom  CDATA  #IMPLIED
              valeur  CDATA  #IMPLIED>
```

Annexe VI

Exemples de modules de connaissances

M_ResumeDynamique

```
<?xml version="1.0" encoding="iso-8859-1"?>
  <!DOCTYPE Module SYSTEM "ModuleNaviTexte.dtd">
  <Module Nom="Module ContextO">
  <!-- Les descriptions de vues -->
  <DescriptionDeVue Titre="Synthèse" Type="Plat">
    <!-- Parametres -->
    <Parametre Nom="Separator">Paragraphe</Parametre>
    <Parametre Nom="Separator">Section</Parametre>
    <Parametre Nom="Separator">Titre</Parametre>
  </DescriptionDeVue>

  <DescriptionDeVue Titre="Lecture Texte Complet" Type="Plat">
    <!-- Parametres -->
    <Parametre Nom="Separator">Paragraphe</Parametre>
    <Parametre Nom="Separator">Section</Parametre>
    <Parametre Nom="Separator">Titre</Parametre>
    <Parametre Nom="Separator">Item</Parametre>
     <Parametre Nom="TaillePolice">30</Parametre>

    <!-- Ops. de visualisation de la vue textuelle simple   AUCUNE -->
    <!-- Operations de navigation de la vue textuelle simple -->
    <Ops_Nav>

   <!-- Expositif vers expositif               -->
   <Op_Nav Titre="Lire_Thématique" Type="Suivant">
     <Source>
      <Condition Type="Simple">
```

```
      <UT>
        <Contrainte arg1="Type" op="EG" arg2="Phrase" />
        <ExisteAnnotation nom="Phrase_etiquetée" valeur="Annonce Thémati
        que" />
      </UT>
    </Condition>
  </Source>
  <Cible>
    <Condition Type="Simple">
      <UT>
        <Contrainte arg1="Type" op="EG" arg2="Phrase" />
       <ExisteAnnotation nom="Phrase_etiquetée" valeur="Annonce Thémati
       que" />
      </UT>
    </Condition>
  </Cible>
</Op_Nav>

<!-- Expositif vers récapitulation   -->
<Op_Nav Titre="Lire_Récapitulation" Type="Premier">
  <Source>
    <Condition Type="Simple">
       <UT>
          <Contrainte arg1="Type" op="EG" arg2="Phrase" />
          <ExisteAnnotation nom="Phrase_etiquetée" valeur="Annonce Théma-
          ti que" />
       </UT>
    </Condition>
  </Source>
  <Cible>
    <Condition Type="Simple">
      <UT>
        <Contrainte arg1="Type" op="EG" arg2="Phrase" />
        <ExisteAnnotation nom="Phrase_etiquetée" valeur="Récapitulation"
        />
      </UT>
    </Condition>
  </Cible>
</Op_Nav>

<!-- récapitulation vers première récapitulation                -->
<Op_Nav Titre="Lire_Récapitualtion_Initiale" Type="Premier">
  <Source>
```

```
    <Condition Type="Simple">
      <UT>
        <Contrainte arg1="Type" op="EG" arg2="Phrase" />
        <ExisteAnnotation  nom="Phrase_etiquetée"  valeur="Récapitulation"
        />
      </UT>
    </Condition>
  </Source>
  <Cible>
    <Condition Type="Simple">
      <UT>
        <Contrainte arg1="Type" op="EG" arg2="Phrase" />
        <ExisteAnnotation  nom="Phrase_etiquetée"  valeur="Récapitulation"
        />
      </UT>
    </Condition>
  </Cible>
</Op_Nav>

<!-- récapitulation vers récapitulation                  -->
 <Op_Nav Titre="Lire_Récapitualtion_Suivante" Type="Suivant">
  <Source>
    <Condition Type="Simple">
      <UT>
        <Contrainte arg1="Type" op="EG" arg2="Phrase" />
        <ExisteAnnotation  nom="Phrase_etiquetée"  valeur="Récapitulation"
        />
      </UT>
    </Condition>
  </Source>
  <Cible>
    <Condition Type="Simple">
      <UT>
        <Contrainte arg1="Type" op="EG" arg2="Phrase" />
        <ExisteAnnotation  nom="Phrase_etiquetée"  valeur="Récapitulation"
        />
      </UT>
    </Condition>
  </Cible>
</Op_Nav>

<!-- récapitulation vers dernière récapitulation                 -->
<Op_Nav Titre="Lire_Récapitualtion_Finale" Type="Dernier">
```

```
  <Source>
     <Condition Type="Simple">
       <UT>
          <Contrainte arg1="Type" op="EG" arg2="Phrase" />
          <ExisteAnnotation nom="Phrase_etiquetée" valeur="Récapitulation" />
       </UT>
     </Condition>
  </Source>
  <Cible>
     <Condition Type="Simple">
       <UT>
         <Contrainte arg1="Type" op="EG" arg2="Phrase" />
         <ExisteAnnotation nom="Phrase_etiquetée" valeur="Récapitulation" />
       </UT>
     </Condition>
  </Cible>
</Op_Nav>

<!-- Conclusion  vers conclusion initiale    -->
<Op_Nav Titre="Première_Conclusion" Type="Premier">
  <Source>
    <Condition Type="Simple">
      <UT>
        <Contrainte arg1="Type" op="EG" arg2="Phrase" />
        <ExisteAnnotation nom="Phrase_etiquetée" valeur="Conclusion" />
      </UT>
    </Condition>
  </Source>
  <Cible>
    <Condition Type="Simple">
      <UT>
        <Contrainte arg1="Type" op="EG" arg2="Phrase" />
        <ExisteAnnotation nom="Phrase_etiquetée" valeur="Conclusion" />
      </UT>
    </Condition>
  </Cible>
</Op_Nav>

<!-- Conclusion vers conclusion suivante -->
<Op_Nav Titre="Conclusion_Suivante" Type="Suivant">
```

```
   <Source>
      <Condition Type="Simple">
         <UT>
            <Contrainte arg1="Type" op="EG" arg2="Phrase" />
            <ExisteAnnotation  nom="Phrase_etiquetée"  valeur="Conclusion"
            />
         </UT>
      </Condition>
   </Source>
   <Cible>
      <Condition Type="Simple">
        <UT>
           <Contrainte arg1="Type" op="EG" arg2="Phrase" />
          <ExisteAnnotation nom="Phrase_etiquetée" valeur="Conclusion" />
        </UT>
      </Condition>
   </Cible>
</Op_Nav>

<!-- Conclusion  vers conclusion finale -->
<Op_Nav Titre="Dernière_Conclusion" Type="Dernier">
   <Source>
      <Condition Type="Simple">
         <UT>
            <Contrainte arg1="Type" op="EG" arg2="Phrase" />
            <ExisteAnnotation  nom="Phrase_etiquetée"  valeur="Conclusion"
            />
         </UT>
      </Condition>
   </Source>
   <Cible>
      <Condition Type="Simple">
         <UT>
           <Contrainte arg1="Type" op="EG" arg2="Phrase" />
            <ExisteAnnotation  nom="Phrase_etiquetée"  valeur="Conclusion"
            />
         </UT>
      </Condition>
   </Cible>
</Op_Nav>

<!-- argumentation vers conclusion suivante  -->
```

```
  <Op_Nav Titre="Enchainement_Argumentation_Conclusion" Type="Suivant">
    <Source>
      <Condition Type="Simple">
        <UT>
          <Contrainte arg1="Type" op="EG" arg2="Phrase" />
          <ExisteAnnotation nom="Phrase_etiquetée" valeur="Argumentation"
          />
        </UT>
      </Condition>
    </Source>
    <Cible>
      <Condition Type="Simple">
         <UT>
            <Contrainte arg1="Type" op="EG" arg2="Phrase" />
            <ExisteAnnotation          nom="Phrase_etiquetée"          va-
            leur="Argumentation" />
        </UT>
      </Condition>
    </Cible>
 </Op_Nav>

<!-- Expositif vers argumentation                -->
<Op_Nav Titre="Lire_Argumentation" Type="Premier">
  <Source>
    <Condition Type="Simple">
       <UT>
         <Contrainte arg1="Type" op="EG" arg2="Phrase" />
        <ExisteAnnotation nom="Phrase_etiquetée" valeur="Annonce Thémati-
        que" />
       </UT>
    </Condition>
  </Source>
  <Cible>
    <Condition Type="Simple">
       <UT>
         <Contrainte arg1="Type" op="EG" arg2="Phrase" />
         <ExisteAnnotation  nom="Phrase_etiquetée"  valeur="Argumentation"
         />
       </UT>
    </Condition>
  </Cible>
</Op_Nav>
```

```
 <!-- Expositif vers argumentation                -->
 <Op_Nav Titre="Lire_Conclusion" Type="Premier">
   <Source>
      <Condition Type="Simple">
         <UT>
            <Contrainte arg1="Type" op="EG" arg2="Phrase" />
            <ExisteAnnotation nom="Phrase_etiquetée" valeur="Annonce Théma-
            tique" />
         </UT>
      </Condition>
   </Source>
   <Cible>
      <Condition Type="Simple">
         <UT>
             <Contrainte arg1="Type" op="EG" arg2="Phrase" />
            <ExisteAnnotation nom="Phrase_etiquetée" valeur="Conclusion" />
         </UT>
      </Condition>
    </Cible>
  </Op_Nav>
</Ops_Nav>
</DescriptionDeVue>

<DescriptionDeVue Titre="Annotations" Type="Plat">
  <!-- Parametres -->
  <Parametre Nom="Separator">Phrase</Parametre>
  <Parametre Nom="TaillePolice">38</Parametre>

  <!-- Contraintes de création-->
  <Condition Type="Simple">
    <UT>
       <ExisteAnnotation nom="Phrase_etiquetée" />
    </UT>
  </Condition>
  <!-- Ops_Vis / -->
  <!--Ops_Nav / -->
</DescriptionDeVue>

<DescriptionDeVue Titre="Argumentatif" Type="Plat">
  <Parametre Nom="TaillePolice">24</Parametre>
  <Parametre Nom="Separator">Phrase</Parametre>
```

```
  <!-- Contraintes de création-->
  <Condition Type="OU">
     <Condition Type="Simple">
        <UT>
          <Contrainte arg1="Type" op="EG" arg2="Phrase" />
          <ExisteAnnotation nom="Phrase_etiquetée" valeur="Argumentation" />
        </UT>
     </Condition>
     <Condition Type="Simple">
        <UT>
          <Contrainte arg1="Type" op="EG" arg2="Phrase" />
          <ExisteAnnotation nom="Phrase_etiquetée" valeur="Conclusion" />
        </UT>
     </Condition>
  </Condition>
</DescriptionDeVue>

<DescriptionDeVue Titre="Vue d'ensemble" Type="Plat">
  <!-- Parametres -->
  <Parametre Nom="Separator">Titre</Parametre>
  <Parametre Nom="Separator">paragraphe</Parametre>
  <Parametre Nom="Separator">section</Parametre>
  <Parametre Nom="Greeking" />
  <!-- Ops_Vis / -->
  <!--Ops_Nav / -->
</DescriptionDeVue>
</Module>
```

NavigationPrecedentCoherenceDescriptive

```
<?xml version="1.0" encoding="iso-8859-1"?>
  <!DOCTYPE Module SYSTEM "ModuleNaviTexte.dtd">
  <Module Nom="Module AMNISTIE">
  <DescriptionDeVue Titre="Navigation Cohérence descriptive" Type="Plat">
    <!-- Parametres -->
    <Parametre Nom="Separator">Titre</Parametre>
    <Parametre Nom="Separator">Paragraphe</Parametre>
    <Parametre Nom="TaillePolice">30</Parametre>
    <Parametre Nom="Annotation_Visible">Oui</Parametre>

    <Ops_Nav>
      <Op_Nav Titre="Cohérence descriptive: Vérifier le précédent syntagme
      nominal composé d'un adjectif ou d'un participe (passé ou présent)"
      Type="Précédent">
        <Source>
          <Condition Type="Simple">
            <UT>
              <Contrainte arg1="Type" op="EG" arg2="Cohérence descriptive"
/>
               <ExisteAnnotation nom="syntagme nominal à modificateur" va-
               leur="adjectif ou participe" />
            </UT>
          </Condition>
        </Source>
        <Cible>
          <Condition Type="Simple">
            <UT>
              <Contrainte arg1="Type" op="EG" arg2="Cohérence descriptive"
/>
               <ExisteAnnotation nom="syntagme nominal à modificateur" va-
               leur="adjectif ou participe" />
            </UT>
          </Condition>
        </Cible>
      </Op_Nav>

      <Op_Nav Titre="Cohérence descriptive: Vérifier le précédent syntagme
      nominal contenant une proposition relative " Type="Précédent">
        <Source>
          <Condition Type="Simple">
            <UT>
```

```
              <Contrainte arg1="Type" op="EG" arg2="Cohérence descriptive"
/>
                <ExisteAnnotation nom="syntagme nominal à modificateur" va-
                leur="proposition relative" />
              </UT>
            </Condition>
          </Source>
          <Cible>
            <Condition Type="Simple">
              <UT>
               <Contrainte arg1="Type" op="EG" arg2="Cohérence descriptive"
/>
                <ExisteAnnotation nom="syntagme nominal à modificateur" va-
                leur="proposition relative" />
              </UT>
            </Condition>
         </Cible>
       </Op_Nav>

     <Op_Nav Titre="Cohérence descriptive: Vérifier le précédent syntagme
     nominal composé de N (de) N" Type="Précédent">
        <Source>
           <Condition Type="Simple">
             <UT>
               <Contrainte arg1="Type" op="EG" arg2="Cohérence descriptive"
/>
              <ExisteAnnotation nom="syntagme nominal à modificateur" va-
              leur="N(de)N" />
             </UT>
          </Condition>
        </Source>
        <Cible>
           <Condition Type="Simple">
              <UT>
               <Contrainte arg1="Type" op="EG" arg2="Cohérence descriptive"
/>
                <ExisteAnnotation nom="syntagme nominal à modifica teur"
                valeur="N(de)N" />
             </UT>
           </Condition>
        </Cible>
      </Op_Nav>
  </Ops_Nav></DescriptionDeVue></Module>
```

Annexe VII

Sémantique du langage de conditions

La sémantique des conditions peut s'exprimer, au moins, de deux manières. La première consiste à définir une sémantique naturelle du langage de conditions. Pour ce faire, une nouvelle relation de transition $\rightarrow_{Cond}$ est créée, et les transitions observent la forme suivante :

$$< C, e > \rightarrow \textit{Valeur de Vérité}$$

Cela signifie que la condition *C* est évaluée pour l'état *e*, et la sémantique de cette évaluation correspond à la valeur de vérité de la condition pour l'état *e*. Étant donné qu'une condition est évaluée pour une UT déterminée et que dans la condition il peut exister des variables (dont la valeur est calculée par le système interpréteur avant d'évaluer la condition), la notion d'état pour les conditions peut se concevoir comme un couple formé par l'UT sur laquelle la condition doit s'évaluer et une fonction *Vars* prenant comme argument le nom d'une variable et renvoyant sa valeur.

Une deuxième approche consiste à définir une fonction sémantique chargée de calculer la valeur de vérité d'une expression de condition, étant donné la condition et l'état (couple UT – *Vars*). C'est cette approche, équivalente à la première, l'approche choisie dans le cadre des conditions car elle est à la fois, de mon point de vue, plus intuitive et plus lisible que l'autre.

Définition : fonction *Vars*

Au moment d'évaluer une condition, le système interpréteur doit fournir une fonction *Vars* qui, pour chaque nom de variable utilisée dans la condition à évaluer, renvoie sa valeur. Le type de cette fonction est alors :

Vars : Nom de Variable → Valeur

Définition : fonction sémantique **C**

La fonction sémantique **C** calcule la valeur de vérité d'une expression de condition, étant donné la condition et l'état (couple UT – *Vars*). Le type de cette fonction est alors :

$$\mathbf{C} : \text{Condition} \rightarrow \text{UT} \times (\text{Nom de Variable} \rightarrow \text{Valeur}) \rightarrow \text{Valeur de Vérité}$$

Il en résulte que pour chaque condition il existe une fonction sémantique spécifique prenant comme argument un état et renvoyant une valeur de vérité. En ce qui concerne la notation, on utilisera la notation suivante :

$$\mathbf{C}\ [\![c]\!]_e$$

pour dénoter l'évaluation de la condition *c* dans l'état *e*. Un exemple de notation concret est le suivant :

$$\mathbf{C}\ [\![\text{UT}(*, *, *, *, x\ \textit{estSousChaîne}\ \text{ChaîneLexicale})]\!]_{(ut,\ Vars)}$$

qui dénote l'évaluation d'une condition simple sur l'UT ut, compte tenu du fait que les valeurs de variables sont définies par la fonction *Vars*. Les variables seront dénotées en italique, en utilisant les lettres *x*, *y* et *z*, éventuellement en y ajoutant des indices. Dans l'exemple ci-dessus, la valeur de la variable *x* s'obtient de l'application de la fonction Vars à cette variable, c'ést-à-dire que la valeur de *x* est *Vars x*. La fonction **C** doit évaluer donc si *Vars x* est une sous-chaîne de la chaîne lexicale de l'UT *ut*.

Les lettres *x*, *y* et *z*, sont en conséquence des méta-variables représentant des variables. D'autres méta-variables seront utilisées (*cf.* tableau AVII.1), correspondant aux constructions syntaxiques du langage de conditions.

cond *est une méta-variable représentant une condition ;*
c *est une méta-variables représentant une contrainte ;*

a *est une méta-variable représentant une annotation (un couple nom-valeur) ;* op *est une méta-variable représentant une opération, et varie sur les valeurs suivantes :* =, ≠, <, >, ≤, ≥, *estPréfixe, estSuffixe et estSousChaîne ;*

Tableau AVII.1 – *Liste de méta-variables utilisées pour le langage des conditions.*

La définition de la fonction sémantique **C** utilisera d'autres fonctions, qui sont progressivement définies. D'après la syntaxe du langage de conditions, il est nécessaire de définir la fonction **C** pour les conditions suivantes :

- composition des conditions (opérateurs NON, ET, OU).

- conditions simples (opérateur UT) ;

- conditions d'existence sur les éléments des UT (existeAnnotations, existeParent, etc.) ;

- conditions sur la hiérarchie (estParent, estDescendant, etc.) ;

Commençons par la définition de **C** dans le cas de composition de conditions. Le tableau ci-dessous montre les définitions concernant les opérateurs logiques NON, ET, OU.

$$\mathbf{C}\ [\text{NON } cond]_{(ut,\ Vars)} = \begin{cases} \textbf{vrai} & \text{si } \mathbf{C}\ [cond]_{(ut,\ Vars)} = \text{faux} \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

$$\mathbf{C}\ [cond_1 \text{ ET } cond_2]_{(ut,\ Vars)} = \begin{cases} \textbf{vrai} & \text{si } (\mathbf{C}\ [cond_1]_{(ut,\ Vars)} = \text{vrai}) \wedge (\mathbf{C}\ [cond_2]_{(ut,\ Vars)} = \text{vrai}) \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

$$\mathbf{C}\ [cond_1 \text{ OU } cond_2]_{(ut,\ Vars)} = \begin{cases} \textbf{vrai} & \text{si } (\mathbf{C}\ [cond_1]_{(ut,\ Vars)} = \text{vrai}) \vee (\mathbf{C}\ [cond_2]_{(ut,\ Vars)} = \text{vrai}) \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

Tableau AVII.2 – *Fonction $\mathcal{C}$ pour la composition de conditions.*

Dans ce cas, la définition de la sémantique est assez simple : elle utilise les valeurs renvoyées par l'évaluation des conditions simples, et la sémantique des opérateurs de la logique classique. En ce qui concerne l'évaluation de conditions simples, le tableau ci-dessous montre la définition de la fonction **C** pour ces types de conditions.

$$\mathbf{C}\ [\mathrm{UT}(c_T, c_N, c_R, ens_{CA}, c_{Ch})]_{(ut,\ Vars)} = \begin{cases} \textbf{vrai} \text{ si } (\mathbf{C}_{\mathrm{Type}}\ [c_T]_{(ut,\ Vars)} = \mathrm{vrai}) \wedge \\ \qquad (\mathbf{C}_{\mathrm{Nro}}\ [c_N]_{(ut,\ Vars)} = \mathrm{vrai}) \wedge \\ \qquad (\mathbf{C}_{\mathrm{Rang}}\ [c_R]_{(ut,\ Vars)} = \mathrm{vrai}) \wedge \\ \qquad (\mathbf{C}_{\mathrm{Annots}}\ [ens_{CA}]_{(ut,\ Vars)} = \mathrm{vrai}) \wedge \\ \qquad (\mathbf{C}_{\mathrm{Chaîne}}\ [c_{Ch}]_{(ut,\ Vars)} = \mathrm{vrai}) \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

Tableau AVII.3 – *Fonction C pour les conditions simples.*

Il en résulte qu'une condition simple est vérifiée si et seulement si toutes les contraintes exprimées en utilisant l'opérateur UT sont vérifiées. Afin de vérifier les différentes contraintes, on utilise les fonctions auxiliaires $\mathbf{C}_{\mathrm{Type}}$, $\mathbf{C}_{\mathrm{Nro}}$, $\mathbf{C}_{\mathrm{Rang}}$, $\mathbf{C}_{\mathrm{Annots}}$ et $\mathbf{C}_{\mathrm{Chaîne}}$, qui sont définies dans les tableaux ci-dessous.

$$\mathbf{C}_{\mathrm{Type}}\ [\mathrm{Type}\ op\ v]_{(ut,\ Vars)} = \mathrm{Eval}_{\mathrm{op}}[ut.\mathrm{Type}\ \ op\ \ v]$$

$$\mathbf{C}_{\mathrm{Type}}\ [v\ op\ \mathrm{Type}]_{(ut,\ Vars)} = \mathrm{Eval}_{\mathrm{op}}[v\ \ op\ \ ut.\mathrm{Type}]$$

$$\mathbf{C}_{\mathrm{Type}}\ [\mathrm{Type}\ op\ x]_{(ut,\ Vars)} = \mathrm{Eval}_{\mathrm{op}}[ut.\mathrm{Type}\ \ op\ \ Vars\ x]$$

$$\mathbf{C}_{\mathrm{Type}}\ [x\ op\ \mathrm{Type}]_{(ut,\ Vars)} = \mathrm{Eval}_{\mathrm{op}}[Vars\ x\ \ op\ \ ut.\mathrm{Type}]$$

$$\mathbf{C}_{\mathrm{Type}}\ [*]_{(ut,\ Vars)} = \textbf{vrai}$$

Tableau AVII.4 – *Fonction d'évaluation des contraintes sur le type d'une UT.*

L'évaluation est toujours vraie si la contrainte imposée au type correspond à l'étoile, ce qui équivaut à indiquer qu'il n'y a pas de contrainte. Si une valeur a été signalée, l'évaluation fait appel à l'évaluation selon l'opérateur particulier utilisé, faite par la fonction $\mathrm{Eval}_{\mathrm{op}}$ recevant comme arguments deux valeurs et un opérateur. Enfin, si une variable a été signalée, la valeur transmise à la fonction $\mathrm{Eval}_{\mathrm{op}}$ est déterminée par la fonction Vars.

Cette approche est similaire pour les fonctions auxiliaires $\mathbf{C}_{\text{Nro}}$ et $\mathbf{C}_{\text{Rang}}$, définies ci-dessous.

$\mathbf{C}_{\text{Nro}}\ [\text{Nro}\ op\ v]_{(ut,\ Vars)} = \text{Eval}_{op}[ut.\text{Nro}\ \ op\ \ v]$

$\mathbf{C}_{\text{Nro}}\ [v\ op\ \text{Nro}]_{(ut,\ Vars)} = \text{Eval}_{op}[v\ \ op\ \ ut.\text{Nro}]$

$\mathbf{C}_{\text{Nro}}\ [\text{Nro}\ op\ x]_{(ut,\ Vars)} = \text{Eval}_{op}[ut.\text{Nro}\ \ op\ \ Vars\ x]$

$\mathbf{C}_{\text{Nro}}\ [x\ op\ \text{Nro}]_{(ut,\ Vars)} = \text{Eval}_{op}[Vars\ x\ \ op\ \ ut.\text{Nro}]$

$\mathbf{C}_{\text{Nro}}\ [*]_{(ut,\ Vars)} =$ **vrai**

Tableau AVII.5 – *Fonction d'évaluation des contraintes sur le numéro d'une UT.*

$$\mathbf{C}_{\text{Rang}}\ [\text{Rang}\ op\ v]_{(ut,\ Vars)} = \begin{cases} \text{Eval}_{op}[ut.\text{Rang}\ \ op\ \ v] \text{ si ut.Rang} \neq \varepsilon \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$$\mathbf{C}_{\text{Rang}}\ [v\ op\ \text{Rang}]_{(ut,\ Vars)} = \begin{cases} \text{Eval}_{op}[v\ \ op\ \ ut.\text{Rang}] \quad \text{si ut.Rang} \neq \varepsilon \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$$\mathbf{C}_{\text{Rang}}\ [\text{Rang}\ op\ x]_{(ut,\ Vars)} = \begin{cases} \text{Eval}_{op}[ut.\text{Rang}\ \ op\ \ Vars\ x] \text{ si ut.Rang} \neq \varepsilon \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$$\mathbf{C}_{\text{Rang}}\ [x\ op\ \text{Rang}]_{(ut,\ Vars)} = \begin{cases} \text{Eval}_{op}[Vars\ x\ \ op\ \ ut.\text{Rang}] \text{ si ut.Rang} \neq \varepsilon \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$\mathbf{C}_{\text{Rang}}\ [*]_{(ut,\ Vars)} =$ **vrai**

Tableau AVII.6 – *Fonction d'évaluation des contraintes sur le rang d'une UT.*

Notons que lorsque le rang n'a pas été indiqué pour une UT (*i.e.* sa valeur est égale à ε), toute contrainte, sauf l'étoile qui indique la non définition d'une contrainte, est évaluee en faux.

Pour évaluer les contraintes sur les annotations d'une UT, trois fonctions auxiliaires sont utilisées. Une première, nommée $\mathbf{C}_{\text{Annots}}$, qui évalue un ensemble de contraintes et qui fait

appel à une deuxième, nommée $\mathbf{C}_{\text{Annot}}$, qui évalue une contrainte sur une annotation. Une troisième fonction, nommée $\text{Eval}_{\exists}$, évalue l'existence d'une annotation pour une UT déterminée.

$$\mathbf{C}_{\text{Annots}}\ [c_A \cup ens_{CA}]_{(ut,\ Vars)} = \begin{cases} \textbf{vrai} \quad \text{si} \quad (\mathbf{C}_{\text{Annot}}\ [c_A{}^{63}]_{(ut,\ Vars)} = \text{vrai}) \vee \\ \qquad\qquad (\mathbf{C}_{\text{Annots}}\ [ens_{CA}]_{(ut,\ Vars)} = \text{vrai}) \\ \textbf{faux} \quad \text{en cas contraire} \end{cases}$$

$$\mathbf{C}_{\text{Annots}}\ [\{\}]_{(ut,\ Vars)} = \textbf{vrai}$$

$$\mathbf{C}_{\text{Annots}}\ [*]_{(ut,\ Vars)} = \textbf{vrai}$$

Tableau AVII.7 – *Fonction d'évaluation des contraintes sur les annotations d'une UT.*

La fonction $\mathbf{C}_{\text{Annots}}$ est définie de manière inductive. Elle évalue l'étoile à vrai et l'ensemble vide aussi à vrai (qui correspond au cas de base de l'induction). Le cas inductif évalue à vrai si et seulement si tant l'évaluation d'une contrainte sur une annotation que l'évaluation des contraintes restantes évaluent à vrai.

$$\mathbf{C}_{\text{Annot}}\ [a]_{(ut,\ Vars)} = \mathbf{C}_{\text{Annot}}\ [\exists a]_{(ut,\ Vars)}$$

$$\mathbf{C}_{\text{Annot}}\ [\exists(nom,\ v)]_{(ut,\ Vars)} = \text{Eval}_{\exists}\ [(nom,\ v)]_{\ ut}$$

$$\mathbf{C}_{\text{Annot}}\ [\exists(\text{x},\ \text{y})]_{(\text{ut},\ \text{Vars})} = \text{Eval}_{\exists}\ [(\text{Vars x},\ \text{Vars y})]_{\ \text{ut}}$$

$$\mathbf{C}_{\text{Annot}}\ [\exists(nom,\ *)]_{(ut,\ Vars)} = \begin{cases} \textbf{vrai} \text{ si } \exists v,\ \text{Eval}_{\exists}\ [(nom,\ v)]_{\ ut} = \text{vrai} \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$$\mathbf{C}_{\text{Annot}}\ [\exists(x,\ *)]_{(ut,\ Vars)} = \begin{cases} \textbf{vrai} \text{ si } \exists v,\ \text{Eval}_{\exists}\ [(Vars\ x,\ v)]_{\ ut} = \text{vrai} \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

63 Notons que la structure syntaxique d'une contrainte sur les annotations est différente de celle correspondant aux contraintes sur le type, le numéro, le rang et la chaîne lexicale. Afin de réduire le nombre de méta-variables, j'utilise la même méta-variable *c*, sous-indexée convenablement chaque fois, pour toutes les contraintes.

$$\mathbf{C}_{\text{Annot}}\ [\exists(*, v)]_{(ut,\ Vars)} = \begin{cases} \textbf{vrai} \text{ si } \exists nom, \text{Eval}_{\exists}\ [(nom, v)]_{ut} = \text{vrai} \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$$\mathbf{C}_{\text{Annot}}\ [\exists(*, x)]_{(ut,\ Vars)} = \begin{cases} \textbf{vrai} \text{ si } \exists nom, \text{Eval}_{\exists}\ [(nom, Vars\ x)]_{ut} = \text{vrai} \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$$\mathbf{C}_{\text{Annot}}\ [\exists(*, *)]_{(ut,\ Vars)} = \begin{cases} \textbf{vrai} \text{ si ut.annotations} \neq \{\} \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$$\mathbf{C}_{\text{Annot}}\ [\neg\exists a]_{(ut,\ Vars)} = \begin{cases} \textbf{vrai} \text{ si } \mathbf{C}_{\text{Annot}}\ [\exists a]_{(ut,\ Vars)} = \text{faux} \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

Tableau AVII.8 – *Fonction d'évaluation des contraintes sur une annotation d'une UT.*

La fonction $\mathbf{C}_{\text{Annot}}$ ci-dessus fait appel à la fonction $\text{Eval}_{\exists}$ (*cf.* tableau 9.11) lorsqu'il faut tester l'existence ou la non-existence d'une annotation comportant une valeur déterminée. Notons que pour l'évaluation de la contrainte $\exists(*, *)$ sur une UT, il est suffisant que l'ensemble d'annotations de l'UT (*i.e.* ut.annotations) ne soit pas vide.

$$\text{Eval}_{\exists}\ [(nom, v)]_{ut} = \begin{cases} \textbf{vrai} \text{ si } ((\text{nom}, \text{v}) \in \text{ut.annotations}) \vee \\ \qquad ((\text{nom}, \text{v}) \notin \text{ut.annotations}) \wedge \\ \qquad (\text{nom}, \text{v}) \in \text{AnnotationsHéritées(ut)}) \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

Tableau AVII.9 – *Fonction d'évaluation d'existence d'une annotation d'une UT.*

L'existence d'une annotation comportant une valeur est testée dans l'ensemble d'annotations directes ou, si elle n'y est pas présente, dans l'ensemble d'annotations héritées. Cette manière de tester tient ainsi compte des annotations héritées.

$$\begin{cases} \text{Eval}_{\text{op}}\ [ut.\text{chaîne}\ \ op\ \ v] \quad \text{si ut.chaîne} \neq \varepsilon \end{cases}$$

$$\mathbf{C}_{\text{Chaîne}}\,[\text{Chaîne } op\ v]_{(ut,\ Vars)} = \textbf{faux} \text{ en cas contraire}$$

$$\mathbf{C}_{\text{Chaîne}}\,[v\ op\ \text{Chaîne}]_{(ut,\ Vars)} = \begin{cases} \text{Eval}_{op}[v\ \ op\ \ ut.\text{chaîne}] & \text{si ut.chaîne} \neq \varepsilon \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$$\mathbf{C}_{\text{Chaîne}}\,[\text{Chaîne } op\ x]_{(ut,\ Vars)} = \begin{cases} \text{Eval}_{op}[ut.\text{chaîne}\ \ op\ \ Vars\ x] & \text{si ut.chaîne} \neq \varepsilon \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$$\mathbf{C}_{\text{Chaîne}}\,[x\ op\ \text{Chaîne}]_{(ut,\ Vars)} = \begin{cases} \text{Eval}_{op}[Vars\ x\ \ op\ \ ut.\text{chaîne}] & \text{si ut.chaîne} \neq \varepsilon \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$$\mathbf{C}_{\text{Chaîne}}\,[*]_{(ut,\ Vars)} = \textbf{vrai}$$

Tableau AVII.10 – *Fonction d'évaluation des contraintes sur la chaîne lexicale d'une UT.*

Notons que lorsque la chaîne lexicale n'a pas été indiquée pour une UT (*i.e.* sa valeur est égale à ε), toute contrainte, sauf l'étoile qui indique la non existence d'une contrainte, est évalue à faux.

La fonction d'évaluation auxiliaire Eval_{op}, utilisée par les fonctions auxiliaires $\mathbf{C}_{\text{Type}}$, $\mathbf{C}_{\text{Nro}}$, $\mathbf{C}_{\text{Rang}}$ et $\mathbf{C}_{\text{Chaîne}}$, est définie dans le tableau ci-dessous.

$$\text{Eval}_{op}[v_1\ =\ v_2] = \begin{cases} \textbf{vrai} & \text{si } v_1 = v_2 \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

$$\text{Eval}_{op}[v_1\ \neq\ v_2] = \begin{cases} \textbf{vrai} & \text{si } v_1 \neq v_2 \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

$$\text{Eval}_{op}[v_1\ <\ v_2] = \begin{cases} \textbf{vrai} & \text{si } v_1 < v_2 \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

$$\text{Eval}_{op}[v_1\ \leq\ v_2] = \begin{cases} \textbf{vrai} & \text{si } v_1 \leq v_2 \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

$$\text{Eval}_{op}[v_1\ >\ v_2] = \begin{cases} \textbf{vrai} & \text{si } v_1 > v_2 \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

$$\text{Eval}_{op}[v_1 \geq v_2] = \begin{cases} \textbf{vrai} & \text{si } v_1 \geq v_2 \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

$$\text{Eval}_{op}[v_1 \text{ estPréfixe } v_2] = \begin{cases} \textbf{vrai} & \text{si } \exists v_3, (v_1 \text{ concat } v_3) = v_2 \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

$$\text{Eval}_{op}[v_1 \text{ estSuffixe } v_2] = \begin{cases} \textbf{vrai} & \text{si } \exists v_3, (v_3 \text{ concat } v_1) = v_2 \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

$$\text{Eval}_{op}[v_1 \text{ estSousChaîne } v_2] = \begin{cases} \textbf{vrai} & \text{si } \exists v_3, \exists v_4, (v_3 \text{ concat } v_1 \text{ concat } v_4) = v_2 \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

Tableau AVII.11 – *Fonction d'évaluation d'un opérateur.*

Il faut souligner que les opérateurs « = », « ≠ », « ≤ », « < », « ≥ » et « > » utilisés à gauche par la fonction $Eval_{op}$ appartiennent au langage de conditions, tandis que les opérateurs équivalents « = », « ≠ », « ≤ », « < », « ≥ » et « > » utilisés à droite ne sont pas les mêmes, correspondant aux opérateurs prédéfinis sur les chaînes de caractères.

Pour l'évaluation des conditions d'existence sur les éléments des UT, on utilise les attributs d'une UT, comme le montre le tableau ci-dessous.

$$\mathbf{C}\,[\text{existeAnnotations}]_{(ut,\,Vars)} = \begin{cases} \textbf{vrai} & \text{si } ut.\text{annotations} \neq \{\} \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

$$\mathbf{C}\,[\text{existeChaîneLexicale}]_{(ut,\,Vars)} = \begin{cases} \textbf{vrai} & \text{si } ut.\text{chaîne} \neq \varepsilon \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

$$\mathbf{C}\,[\text{existeTitre}]_{(ut,\,Vars)} = \begin{cases} \textbf{vrai} & \text{si } ut.\text{titre} \neq \{\} \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

$$\mathbf{C}\,[\text{existeParent}]_{(ut,\,Vars)} = \begin{cases} \textbf{vrai} & \text{si } ut.\text{parent} \neq \text{Texte} \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

$$\mathbf{C}\,[\text{existeFils}]_{(ut,\,Vars)} = \begin{cases} \textbf{vrai} & \text{si } ut.\text{fils} \neq \{\} \\ \textbf{faux} & \text{en cas contraire} \end{cases}$$

Tableau AVII.12 – *Fonction $\mathbf{C}$ pour les conditions d'existence sur les éléments des UT.*

C'est-à-dire, par exemple, que l'UT *ut* vérifie la condition *existeFils* si et seulement si son ensemble de fils (*i.e. ut*.fils) n'est pas vide. L'existence de la chaîne lexicale est assuré par une chaîne lexicale non vide (*i.e.* distincte de la valeur ε). Notons que la fonction *Vars* n'est pas utilisée pour ce type de conditions.

On considère qu'une UT a une chaîne lexicale si la valeur de l'attribut chaîne de l'UT est différente de ε. D'autre part, toutes les UT ont un parent, que ce soit une UT parent ou Texte si l'UT est une UT de niveau maximale dans la hiérarchie d'UT. Néanmoins, on considère qu'une UT a un parent, si la valeur de l'attribut parent correspond à une UT. En conséquence, on considère pour le langage défini que les UT de niveau maximal n'ont pas de parent.

Quant à l'évaluation des conditions sur la hiérarchie, on utilise aussi bien les attributs d'une UT que les définitions présentées préalablement, comme le montre le tableau ci-dessous.

$$\mathbf{C}\,[\text{estParent}(cond)]_{(ut,\,Vars)} = \begin{cases} \textbf{vrai} \text{ si } \exists \text{utf} \in \text{ut.fils}, \ \mathbf{C}\,[cond]_{(utf,\,Vars)} = \text{vrai} \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$$\mathbf{C}\,[\text{estFils}(cond)]_{(ut,\,Vars)} = \begin{cases} \textbf{vrai} \text{ si } (\text{ut.parent} \neq \text{Texte}) \wedge (\mathbf{C}\,[cond]_{(ut.parent,\,Vars)} = \text{vrai}) \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$$\mathbf{C}\,[\text{estFrère}(cond)]_{(ut,\,Vars)} = \begin{cases} \textbf{vrai} \text{ si } \exists \text{utf}, \ (ut.\text{parent} = \text{utf.parent}) \wedge (\mathbf{C}\,[cond]_{(utf,\,Vars)} = \text{vrai}) \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$$\mathbf{C}\,[\text{estAscendant}(cond)]_{(ut,\,Vars)} = \begin{cases} \textbf{vrai} \text{ si } \exists \text{utd} \in \text{DESC}(ut), \ \mathbf{C}\,[cond]_{(utd,\,Vars)} = \text{vrai} \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$$\mathbf{C}\,[\text{estDescendant}(cond)]_{(ut,\,Vars)} = \begin{cases} \textbf{vrai} \text{ si } \exists \text{uta} \in (\text{ASC}(ut) - \text{Texte}), \ \mathbf{C}\,[cond]_{(uta,\,Vars)} = \text{vrai} \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$$\mathbf{C}\ [\text{contientDansTitre}(cond)]_{(ut,\ Vars)} = \begin{cases} \textbf{vrai} \text{ si } \exists \text{utt} \in ut.\text{titre}, \mathbf{C}\ [cond]_{(utt,\ Vars)} = \text{vrai} \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

$$\mathbf{C}\ [\text{estDansTitreDe}(cond)]_{(ut,\ Vars)} = \begin{cases} \textbf{vrai} \text{ si } \exists \text{utp}, \ (ut \in \text{utp.titre}) \wedge (\mathbf{C}\ [cond]_{(utp,\ Vars)} = \text{vrai}) \\ \textbf{faux} \text{ en cas contraire} \end{cases}$$

Tableau AVII.13 – *Fonction $\mathbf{C}$ pour les conditions sur la hiérarchie des UT.*

Références bibliographiques

[Aho *et al.* 1986] Aho A.V., Sethi R., Ullman J.D. – *Compilers: Principles, Techniques and Tools*, Addison Wesley, Massachusetts, USA, 1986.

[ATILF] Le Trésor Informatisé de la Langue Française – http://atilf.atilf.fr/.

[Baccino 2004] Baccino T. – *La lecture électronique*, Presses Universitaires de Grenoble, collection Sciences et Technologies de la Connaissance, 2004.

[Baecker *et al.* 1995] Baecker R., Grudin J., Buxton W., Greenberg S. – *Readings in Human-Computer Interaction: Toward the Year 2000,* Morgan Kaufmann Publishers, 1995.

[Baeza *et* Ribeiro 1999] Baeza R., Ribeiro B. – *Modern Information Retrieval*, Addison Wesley, 1999.

[Barzilay *et* Elhadad 1997] Barzilay R., Elhadad M. – « Using Lexical Chain for Text Summarization ». In *Workshop Intelligent Scalable Text Summarization,* EACL, Madrid, Espagne, 1997, pp. 11-17.

[Battistelli *et* Minel 2006] Battistelli D, Minel J.-L. (à paraître en 2006) – « Les systèmes de résumé automatique: comment assurer une continuité référentielle dans la lecture des textes », in Compréhension des langues et interaction sous la direction de Gérard Sabah, Paris, éditions Hermès, 37 p.

[Bederson et Holland 1994] Bederson B.B., Holland J.D. – « Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics », *in Proceedings of ACM User Interface Software and Technology Conference (UIST '94)*, pp. 17-26.

[Béguelin 2000] Béguelin M.J. (dir.) – *De la phrase aux énoncés, grammaire scolaire et descriptions linguistiques*, Bruxelles, De Boeck/Duculot, 2000.

[Ben Hazez 1999] Ben Hazez S. – « BDContext : un système de gestion de connaissances linguistiques orientées vers le filtrage sémantique de textes ». In *CIDE'99*, Damas, Syrie, 1999.

[Ben Hazez 2002] Ben Hazez S. – *Un modèle d'exploration contextuelle de textes : filtrage et structuration d'information textuelles, modélisation et réalisation informatique (système SEMANTEXT).* Thèse de doctorat, Université Paris-Sorbonne, Paris, 2002.

[Berge 1958] Berge C. – *Théorie des Graphes,* Dunod, 1958.

[Berri 1996] Berri J. – *Contribution à la méthode d'exploration contextuelle. Applications au résumé automatique et aux représentations temporelles. Réalisation informatique du système SERAPHIN*. Thèse de doctorat, Université Paris-Sorbonne, Paris, 1996.

[Berri *et al.* 1996] Berri J., Cartier E., Desclés J.-P., Jackiewicz A., Minel, J.-L. – « SAFIR, système automatique de filtrage de textes. ». *Actes de TALN'96*, Marseille, 1996, pp.140-149.

[Bilhaut 2003] Bilhaut F. – « The Linguastream Platfom », *Proceedings of the 19th Spanish Society for Natural Language Processing Conference (SEPLN)*, Alcalá de Henares, Spain, 339-340.

[Bilhaut *et al.* 2003] Bilhaut F., Ho-Dac M., Borillo A., Charnois T., Enjalbert P., Draoulec A.L., Mathet Y., Miguet H., Pery-Woodley M.-P., Sarda L. « Indexation discursive pour la navigation intra-documentaire : cadres temporels et spatiaux dans l'information geographique ». *Actes de Traitement Automatique du Langage Naturel (TALN)*, Batz-sur-Mer, France, 2003.

[Bodner *et* Chignell 1999] Bodner R., Chignell M. – « Dynamic hypertext: querying and linking », *ACM Computing Surveys, 31(4), 1999*.

[Boguraev *et al.* 1998] Boguraev B., Kennedy C., Bellamy R., Brawer S., Wong Y., Swartz J., « Dynamic presentation of document content for rapid on-line skimming », *AAAI Spring Symposium on Intelligent Text Summarisation*, Stanford, CA, pp. 109-118, 2001.

[Briet 1951] Briet, S. – *Qu'est-ce que la documentation*, EDIT, Paris, 1951.

[Brusilovsky 1994] Brusilovsky P. – « Adaptive Hypermedia: An attempt to analyse and generalise », *Proceedings of UM'94 Fourth International Conference on User Modelling*, 1994.

[Brusilovsky 1996] Brusilovsky P. – « Methods and techniques of adaptive hypermedia », *User Modeling and User-Adapted Interaction, 6, 2-3* pp. 87-129, 1996.

[Brusilovsky 2003] Brusilovsky P. – « From Adaptive Hypermedia to the Adaptive Web », *G. Szwillus, J. Ziegler (Hrsg.): Mensch & Computer 2003: Interaktion in Bewegung*. Stuttgart: B. G. Teubner, pp. 21-24, 2003.

[Brusilovsky 2004] Brusilovsky P. – « Adaptive Navigation Support: From Adaptive Hypermedia to the Adaptive Web and Beyond », *PsychNology Journal, Volume 2, Number 1*, pp. 7-23, 2004.

[Brusilovsky et Pesin 1998] Brusilovsky P., Pesin, L. – « Adaptive navigation support in educational hypermedia: An evaluation of the ISIS-Tutor », *Journal of Computing and Information Technology*, 1998.

[Brusilovsky et Eklund 1998] Brusilovsky P., Eklund J. – « A Study of User Model Based Link Annotation in Educational Hypermedia », *Journal of Universal Computer Science, vol n°4*, pp. 429-448, Springer Pub Co., 1998.

[Buckland 1997] Buckland M.K. – « What is a "document" », *Journal or the American Society of Information Science 48 - n° 9*, 1997, pp. 804-809.

[Burbeck 1987] Burbeck S. – « Applications Programming in Smalltalk-80 (TM): How to use Model-View-Controller (MVC) », http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html.

[Burns *et al.* 1986] Burns M., Warren D., Rudisill M – « Formatting space-related displays to optimize expert and nonexpert user performance », *Proc. ACM SIGCHI'86*, ACM, New-York, 1986, pp. 274-280.

[Card 1982] Card S.K. – « User perceptual mechanisms in the search of computer command menus ». *Proc. Human Factors in Computer Systems*, Washington D.C., ACM chapter, mars 1982, pp. 190-196.

[Card *et al.* 1999] Card S.K., Mackinlay J., Shneiderman B., *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann Publliishers, 1999.

[Charolles 1981] Charolles M. – « Coherence as a principle in the interpretation of discourse », *Text* 3, 1981, pp. 71-99.

[Charolles 1995] Charolles M. – « Cohésion, cohérence et pertinence du discours ». In *Travaux de linguistique, 29*, 1995, pp.125-151.

[Charolles 1997] Charolles M. – « L'encadrement du discours - Univers, champs, domaines et espace », *Cahier de recherche linguistique, LANDISCO*, 6, Université Nancy 2, 1997, pp. 1-73.

[Choi 2000] Choi F. – « Advances in domain independent linear text segmentation », *Actes de NAACL'00*, pp. 26-33, 2000.

[Clarke *et al.* 1995] Clarke C., Cormack G. V., Burkowski F. J. – « An algebra for structured text search and a framework for its implementation », The Computer Journal, 38(1) pp. 43-56, 1995.

[Coirier *et al.* 1996] Coirier P., Gaonac'h D., Passerault J.-M. – *Psycholinguistique textuelle. Approche cognitive de la compréhension et de la production des textes*. Armand Colin, Paris, 1996.

[Cotte 2002] Cotte D. – « L'approche néophyte de la page web. Ou " Mais où je clique là ?" », *Les cahiers du numérique 3 (3)*, pp. 17-32, 2002.

[Cotte 2004$_a$] Cotte D. – « Le concept de "document numérique" », *Communication et Langages n° 140*, pp. 31-42, 2004.

[Cotte 2004 $_b$] Cotte, D. – « Leurres, ruses, désorientation dans les écrits de réseau : la métis à l'écran. », *Communication & langages, n° 139*, Avril 2004, pp. 63-74.

[Couto 2002] Couto J. – *Los sistemas de exploración contextual de cara al usuario (les systèmes d'exploration contextuelle face aux utilisateurs)*, Mémoire de Master, Université de la République, Faculté d'Ingénierie, Uruguay, 2002.

[Couto *et al.* 2004] Couto J., Ferret O., Grau B., Hernandez N., Jackiewicz A., Minel J.-L., Porhiel S. – « RÉGAL, un système pour la visualisation sélective de documents. », *Revue d'Intelligence Artificielle*, Hermès, 2004, pp. 481-514.

[Couto *et* Minel 2004$_a$] Couto J., Minel J.-L. – « Outils dynamiques de fouille textuelle », *RIAO 2004,* Avignon, pp. 420-430.

[Couto *et* Minel 2004$_b$] Couto J., Minel J.-L. – « Interfaces dynamiques de fouilles textuelles », *Semaine du Document Numérique*. Université de la Rochelle, France, 2004.

[Couto *et al.* 2005$_a$] Couto J., Lundquist L., Minel J-L – « Navigation interactive pour l'apprentissage en linguistique textuelle », *Environnements Informatiques pour l'Apprentissage Humain (EIAH) 2005*, pp 45-56, Montpellier 2005 – France.

[Couto *et al.* 2005$_b$] Couto J., Lundquist L., Minel J.-L. – « Using NaviTexte to teach French as a second language », *Recent Research Developments in Learning Technologies 2005* pp. 269-273, Cáceres, España.

[Couto *et al.* 2005$_c$] Couto J., L. Lundquist, J.L. Minel. – « Naviguer pour apprendre », *TALN 2005*, Paris, France.

[Crestani *et al.* 2002] Crestani F., de la Fuente P., Vegas J. – « Experimenting with graphical user interface structured document retrieval », *SIGIR'02*, août 2002, Tampere, Finlande.

[Crispino *et al.* 1999] Crispino G., Ben Hazez S., Minel J.-L. « ContextO, un outil de la plate-forme d'ingénierie linguistique FilText ». In *VEXTAL 99*, Venise, Italie, 1999, pp. 361-367.

[Crispino 2003] Crispino G. – *Conception et réalisation d'un système informatique d'exploration contextuelle. Conception d'un langage de spécification de connaissances linguistiques*, Thèse de doctorat, Université Paris-Sorbonne, Paris, 2003.

[Crispino *et* Couto 2004] Crispino G., Couto J. – « Construction Automatique de résumés : une approche dynamique », *Traitement automatique des langues (TAL) Volume 45 - nº 1/2004, Résumé automatique de textes*, Hermès, 2004, pp. 95-120.

[CUP] CUP Parser Generator for Java. http://www.cs.princeton.edu/~appel/modern/java/CUP/

[Danielson 2002] Danielson D.R. – « Web navigation and the behavorial effects of constantly visible maps », *Interacting with Computers*, 14, 2002, p. 601-618.

[De Bra et al. 1999] De Bra P., Brusilovsky P., Houben G.-J. – « Adaptive hypermedia: from systems to framework », *ACM Computing Surveys 31 (4)*, 1999.

[Desclés *et al.* 1991] Desclés J.-P., Jouis C., Oh-Jeong H.-G., Maire Reppert D. – « Exploration Contextuelle et sémantique : un système expert qui trouve les valeurs sémantiques des temps de l'Indicatif dans un texte ». *Knowledge modeling and expertise transfer* (ed. D. Herin-Aime, R. Dieng, J.-P. Regourd, J. P. Angoujard), Amsterdam, 1991, pp. 371-400.

[Dieberger *et* Russel 2002] Dieberger A., Russell D.M. – « Exploratory navigation in large multimedia documents using Context Lenses », In *Proceedings of the 35th Hawaii International Conference on System Sciences*, Hawaii, 2002.

[DOM] Document Object Model – http://www.w3.org/DOM/

[DUC] Document Understanding Conferences – http://duc.nist.gov/

[Edwards *et* Hardman 1989] Edwards D.M., Hardman L. – « Lost in hyperspace: cognitive mapping and navigation in a hypertext environment », In R. McAleese (Ed.) Hypertext: Theory and Practice, 1989, Oxford: Intellect.

[Eick *et al.* 1992] Eick S.G., Steffen J.L., Sumner E.E. – « SeeSoft – A tool for Visualizing Line Oriented Software Statistics », dans [Card *et al.* 1999], pp. 419-430, 1992.

[Eklund et al. 1997] Eklund J., Brusilovsky P., Schwarz E. – « Adaptive Textbooks on the World Wide Web », *AusWeb97, Third Australian World Wide Web Conference*, 1997.

[Elm *et* Woods 1985] Elm W.C., Woods D.D. – « Getting lost: A Case Study in Interface Design. », *Proceeding of the Human Factors Society*, 1985, pp. 927-931.

[Ertzscheid 2003] Ertzscheid O. – « Comportements de navigation et documents électroniques : proposition d'invariants », *Colloque CIDE 6*, Caen, 24-26 Novembre 2003.

[Fauconnier 1984] Fauconnier G. – *Espaces mentaux,* Paris, Ed. de Minuit, 1984.

[Ferret *et al.* 2001] Ferret O., Grau B., Minel J.-L. , Porhiel S. – « Repérage de structures thématiques dans des textes », *TALN 2001*, Tours, pp. 163-172, 2001.

[Foucalt 1969] Foucault M. – *L'Archéologie du Savoir*. Gallimard, Paris, 1969.

[Fowler, 1998] Fowler S. – *GUI design handbook*. McGraw-Hill, 1998.

[Furnas 1986] Furnas G.W. – « Generalized Fisheye Views », *in Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pp. 16-23.

[García 1998] García D. – *Analyse automatique des textes pour l'organisation causale des actions. Réalisation du système informatique COATIS*. Thèse de doctorat, Université Paris-Sorbonne, Paris, 1998.

[Géry 2002] Géry M. – « Un modèle d'hyperdocument en contexte pour la recherche d'information structurée sur le Web », *Revue des Sciences et Technologies de l'Information,* 7/2002*,* Hermès, Paris, pp. 11-44, 2002.

[Glassbook] GlassBook Reader – http://www.glassbook.com

[Grosz *et al.* 1995] Grosz B., Joshi A.K., Weinstein S. – « Centering : A framework for modeling the local coherence of discourse. » *Computational Linguistics*, 21(2), pp. 203–225, 1995.

[Grudin 1991] Grudin J. – « *Interactive Systems: Bridging the Gaps Between Developers and Users* », dans [Baecker *et al.* 1995], pp. 293-303, 1991.

[Halliday *et* Hasan 1976] Halliday M., Hasan R. – *Cohesion in English*. Longman, New York, 1976.

[Harman 2002] Harman D. – « The Development and Evolution of TREC and DUC », *Third NTCIR Workshop on research in information Retrieval, Automatic Text Summarization and Question Answering*, Tokyo, October 2002.

[Hascoët *et* Beaudouin-Lafon 2001] Hascoët M., Beaudouin-Lafon M. – « Visualisation interactive d'information », *Revue I3 1(1)*, pp. 77-108, 2001.

[Hearst 1995] Hearst M.A. – « Tilebars: Visualization of term distribution information in full text information access. » In *Proceedings of Human Factors in Computing Systems, CHI'95*, pages 59-66, Denver, CO, 1995. ACM.

[Hennessy 1990] Hennessy M. – *The semantics of programming languages : an elementary introduction using structural operational semantics,* John Wiley and Sons, New York, N.Y., 1990.

[Hernandez *et* Grau 2002] Hernandez N., Grau B., « Analyse thématique du discours : segmentation, structuration, description et representation », *Actes de CIDE'2002*, Hammamet, Tunisie, pp. 277-285

[Herviou *et al.* 1996] Herviou M.-L, Quatrain R., Monteil M.-G. – « Construction de terminologies : une chaîne de traitement supportée par un atelier intégrant outils linguistiques et statistiques ». *Actes de TALN'96*, Marseille, 1996, pp.130-139.

[Ho-Dac *et al.* 2004] Ho-Dac M., Jacques M.-P., Rebeyrolle J. – « Sur la fonction discursive des titres », *In S. Porhiel et D. Klingler (Eds.), L'unité texte*, pp. 125-152, 2004.

[Hobbs 1985] Hobbs J.R. – « On the coherence and structure of discourse », *CSLI Technical Report 85-37*, Stanford, CA, USA, 1985.

[Jackiewicz 2002] Jackiewicz A. – « Repérage et délimitation des cadres organisationnels pour la segmentation automatique des textes ». In *CIFT'02 : Colloque International sur la Fouille de Textes*, Hammamet, Tunisie, 2002, pp. 95-107.

[Jackiewicz *et* Minel 2003] Jackiewicz A., Minel J.-L. – « L'identification de structures discursives engendrées par les cadres organisationnels ». In *TALN'03* , Batz-sur-Mer, 2003.

[Jacquemin *et al.* 2005] Jacquemin C. (Porteur du projet) – « Appréhender dynamiquement les textes à plusieurs niveaux de détail. » Présentation du projet COGNITIQUE dans le Colloque Société de l'information, *Bilan du programme interdisciplinaire de recherche du CNRS 2001 – 2005,* 19, 20 et 21 mai 2005, Lyon.

[Jeanneret 2001] Jeanneret Y. – « Informatic Literacy : manifestations, captations et déceptions dans le texte informatisé », *Spirales, Lille, n°28, "Nouveaux outils, nouvelles écritures, nouvelles lectures"*, mai 2001, p. 11-32.

[Jeanneret 2004] Jeanneret Y. – « Le procès de numérisation de la culture, un défi pour la pensée du texte », *Protée, Vol. 32, n° 2*, Sainte-Foy, Presses de l'Université du Québec, 2004, p. 9-18.

[JLex] JLex: A Lexical Analyzer Generator for Java. http://www.cs.princeton.edu/~appel/modern/java/JLex/

[Jouis 1993] Jouis C. – *Contributions à la conceptualisation et à la modélisation des connaissances à partir d'une analyse linguistique de textes : Réalisation d'un prototype (le système SEEK).* Thèse de doctorat, Université Paris-Sorbonne, Paris, 1993.

[Kamp *et* Reyle 1993] Kamp H., Reyle U. – « From discourse to logic : Introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory. » In *Studies in Linguistics and Philosophy, volume 42*. Kluwer Academic Publishers, London, Boston, Dordrech, 1993.

[Kehler 2002] Kehler A. – *Coherence, reference, and the theory of grammar*, Stanford, CA : CSLI Publications, 2002.

[Kintsch 1998] Kintsch, W. – *Comprehension. A Paradigm for Cognition*, Cambridge, Cambridge University Press, 1998/2003.

[Kleiber 1990] Kleiber G., Tyvaert J.-E. (éd), *L'anaphore et ses domaines*, Paris, Klincksieck, 1990.

[Lamping *et* Rao 1996] Lamping J., Rao R. – « The Hyperbolic Browser : A Focus + Context technique for visualizing large hierarchies », dans [Card *et al.* 1999], pp. 382-408, 1996.

[Landauer 1991] Landauer T. – « Let's Get Real: A Position Paper on the Role of Cognitive Psychology in the Design of Humanly Useful and Usable Systems ». Dans [Baecker *et al.* 1995], pp. 659-665, 1991.

[Laurière 1986] Laurière J.L. – *Intelligence Artificielle. Résolution de problèmes par l'Homme et la Machine*, Éditions Eyrolles, Paris, 1986.

[Le Priol 2000] Le Priol F. – *Extraction et capitalisation automatiques de connaissances à partir de documents textuels. SEEK-JAVA : identification et interprétation de relations entre concepts*. Thèse de doctorat, Université Paris-Sorbonne, Paris, 2000.

[Le Roux *et al.* 1994] Le Roux D., Minel J.-L., Berri J. – « SERAPHIN project ». *First European Conference of Cognitive Science in Industry*, Luxembourg, 1994, pp. 275-283.

[Lee *et* Tedder 2003] Lee M.J., Tedder M.C. – «The effects of three different computer texts on reder' recall : based on working memory capcity», *Computers in Human Behavior*, 19, 2003, p. 767-783.

[Luc 2002] Luc C. – « Une typologie des énumérations basée sur les structures rhétoriques et architecturales du texte», In *Inscription Spatiale du Langage : structures et processus (IISLsp 2002)*, Toulouse, 2002, pp. 153-164.

[Lundquist 1980] Lundquist, L. – *La cohérence textuelle, syntaxe, sémantique, pragmatique*, Copenhagen, Nordisk Forlag, 1980.

[Lundquist 1989] Lundquist L. – « Coherence in scientific texts ». Heydrich, Wolfgang (ed.) Connexity and coherence. Analysis of text and discourse, Berlin, de Gruyter, 1989.

[Lundquist 1990$_a$] Lundquist L. – *L'analyse textuelle. Méthode, exercices*, Copenhagen, Nordisk Forlag, 1990.

[Lundquist 1990$_b$] Lundquist L. – « Conditions de Production et Programmation Argumentative », *Verbum*, vol. 13, no 4, Sémantique et Société, p. 237-264, 1990.

[Lundquist 1999] Lundquist L. – « Le factum textus. Fait de grammaire, fait de linguistique ou fait de cognition? », *Langue française*, 1999, p. 56-75.

[Lundquist 2000] Lundquist L. – « Knowledge, events and anaphors in texts for specific purposes. » Lundquist, L. & R. Jarvella (eds.), *Language, text, and knowledge. Mental models of expert communication*, Mouton de Gruyter, Berlin, 2000.

[Lundquist 2005] Lundquist L. – « Noms, verbes et anaphores (in)fidèles. Pourquoi les Danois sont plus fidèles que les Français », *Langue française*, vol 145, pp. 73-92, 2005.

[Lundquist 2006] Lundquist L. – *Tekstkompetence på fremmedsprog*, Samfundslitteratur, 2006.

[Lundquist *et al.* 2006] Lundquist L., Minel J.L., Couto J. – « NaviLire, Teaching French by Navigating in Texts », *in Information Processing and Management of Uncertainty in Knowledge-Based Systems* (à apparaître), Paris, juillet 2006.

[Mani 2001] Mani I. – *Automatic Summarization*. John Benjamins Publishing Company, Amsterdam, 2001.

[McEneaney 1999] McEneaney J.E. – « Visualizing and Assessing Navigation in Hypertext », *Hypertext 1999*, pp. 61-70.

[Mann *et* Thompson 1987] Mann W.C., Thompson S.A. – « Rhetorical Structure Theory : A Theory of text organization ». In L. Polanyi (ed), *The Structure of Discourse.* Ablex Publishing Company, Norwood, 1987, pp. 85-96.

[Marcus 1990] Marcus A. – « *Principles of Effective Visual Communication for Graphical User Interface Design* », dans [Baecker *et al.* 1995], pp. 425-441, 1990.

[Marty 2005] « 76 définitions du signe de C.S. Peirce et leur analyse. » http://www.univ-perp.fr/see/rch/lts/marty/76-fr.htm

[Mathe et Chen 1994] Mathe N., Chen J. – « A User-Centered Approach to Adaptive Hypertext based on an Information Relevance Model », *Proceedings of the 4th International Conference on User Modeling (UM'94)*, Hyannis, MA, August 15-19, 1994. pp. 107-114.

[Mathieu 2000] Mathieu, Y. Y. – *Les verbes de sentiments. De l'analyse linguistique au traitement automatique*, Paris, CNRS éditions.

[Mathieu 2005$_a$] Mathieu, Y. Y. – « A Computational Semantic Lexicon of French Verbs of Emotion », in *Computing Attitude and Affect in Text: Theory and Applications James G. Shanahan, Yan Qu, Janyce Wiebe (Eds.)*. Springer, Dordrecht, The Netherlands. chapitre 10, pp. 109-123.

[Mathieu 2005$_b$] Mathieu Y. – « Annotations of Emotions and Feelings in Texts », Proceedings of *Affective Computing and Interaction (ACII2005)*, Beijing, Springer Lecture Notes in Computer Science.

[McDonald 1990] McDonald J.A. – « Painting Multiple Views of Complex Objects », *Proceedings of ECOOP/OOPLSLA'90*, pp. 245-257.

[Meunier 1998] Meunier J.-G. – « La gestion des connaissances et les intelliciels ». *Actes de RIFRA'98 – Rencontre Internationale sur l'Extraction le Filtrage et le Résumé Automatique*, Sfax, Tunisie, 1998, pp. 4-5.

[Minel *et al.* 1997] Minel J-L., Nugier S., Piat G. – «How to appreciate the Quality of Automatic Text Summarization », *Workshop Intelligent Scalable Text Summarization*, EACL, Madrid, Espagne, 1997, pp.25-30.

[Minel *et al.* 2001] Minel J.-L., Desclés J-P., Cartier E., Crispino G., Ben Hazez S., Jackiewicz A. – « Résumé automatique par filtrage sémantique d'informations dans des textes. Présentation de la plate-forme FilText ». *Revue Technique et Science informatiques*, Hermès, n° 3, pp. 369-395, 2001.

[Minel 2002] Minel J-L. – *Filtrage sémantique de textes. Problèmes, conception et réalisation d'une plate-forme informatique.* Habilitation à diriger des recherches, Université Paris-Sorbonne, Paris, 2002.

[Minel 2003] Minel J.-L. – *Filtrage sémantique de textes*. Editions Hermès, 2003.

[Morris *et* Hirst 1991] Morris J., Hirse G. – « Lexical cohesion computed by thesaural relations as an indicator of the structure of the text ». In *Computational Linguistics, 17(1)*, 1991, pp. 21-45.

[Mourad 2001] Mourad G. – *Analyse informatique des signes typographiques pour la segmentation de textes et l'extraction automatique des citations. Réalisation des applications informatiques SegATex et CitaRE.* Thèse de doctorat, Université Paris-Sorbonne, Paris, 2001.

[Mullet et Sano 1995] Mullet K., Sano D. – *Designing Visual Interfaces: Communication Oriented Techniques*, SunSoft Press, 1995.

[Murch 1985] Murch G. – « Color Graphics – Blessing or Ballyhoo ». Dans [Baecker *et al.* 1995], pp. 442-443, 1985.

[Myers 2000] Myers D.G. – *Psychology (6th edition),* Worth Publishing.

[Nelson 1965] Nelson T. – « A file structure for the complex, the changing and the interminate. », *In ACM 20th National Conference*, 1965.

[Nielsen 1994] Nielsen J. – *Heuristic evaluation.* In Nielsen, J., and Mack, R.L. (Eds.), Usability Inspection Methods, John Wiley & Sons, New York, 1994.

[Nielson *et* Nielson 1992] Nielson H.R., Nielson F. – *Semantics with application. A formal introduction,* John Wiley & Sons, 1992.

[Norman 1988] Norman D.A. – *The Design of Everyday Things*. Doubleday, New York, 1988.

[Otlet 1934] Otlet P. – *Traité de documentation*. Editiones Mundaneum, Brussels, 1934. Reprinted 1989, Liège: Centre de Lecture Publique de la Communauté Française.

[Otter *et* Johnson 2000] Otter M., Johnson H. – « Lost in hyperspace: metrics and mental models. », *Interacting with Computers 13(1)*, pp. 1-40, 2000.

[Pascual 1991] Pascual E. – *Représentation de l'architecture textuelle et génération de texte.* Thèse de doctorat, Université Paul Sabatier, Toulouse, 1991.

[Pédauque 2003] Pédauque R. – « Document : forme, signe et médium, les re-formulations du numérique. » – http://archivesic.ccsd.cnrs.fr/sic_00000511.html

[Peirce 1978] Peirce C.S. – *Ecrits sur le signe, rassemblés traduits et commentés par G. Deledalle*, Paris, Le Seuil (coll. L'ordre philosophique), 1978.

[Perlin et Fox 1993] Perlin K., Fox D. – « Pad: An Alternative Approach to the Computer Interface », *in Proceedings of 1993 ACM SIGGRAPH Conference,* pp. 57-64.

[Pery-Woodley 2005] Pery-Woodley M.-P. – « Discours, corpus, traitements automatiaques », *in A. Condamines (Ed.), Sémantique et Corpus*. Hermès, France, pp. 177-205.

[Pirolli et Rao 1996] Pirolli P., Rao R. – « Table Lens as a Tool for Making Sense of Data », dans [Card *et al.* 1999], pp. 597-615, 1996.

[Plotkin 1982] Plotkin G.D. – *A structural approach to Operational Semantics,* Lecture notes, DAIMI FN-19, Aarhus University, Danemark, 1982.

[Polanyi 1987] Polanyi L. – « Keeping it all straight : Interpreting narrative time in real discourse. » *WCCFL*, 6, pp. 229–245, 1987.

[Polanyi 1988] Polanyi L. – « A formal model of the structure of discourse. » *Journal of Pragmatics*, 12, pp. 601–638, 1988.

[Porhiel 2001] Porhiel S. – « Linguistic expressions as a tool to extract thematic information », *Corpus Linguistic 2001*, Lancaster University.

[Porhiel 2003] Porhiel S. – « Les introducteurs de cadre thématique », *Cahiers de Lexicologie 83, 2*, pp. 1-36, 2003.

[Rastier 2001] Rastier F. – *Arts et Sciences du texte*, Paris, P.U.F., collection «Formes sémiotiques ».

[RTP-DOC] « RTP (33) DOC- CNRS - Documents et contenu - Roger Pedauque » – http://rtp-doc.enssib.fr

[Sabah 2002] Sabah G. – « Rapport de fin de recherche ». *Projet Cognitique « Régal »* (sous la direction de Gérard Sabah), 2002.

[Saggion *et* Lapalme 1998] Saggion H., Lapalme G. – « Where does information come from ? Corpus Analysis for Automatic Abstracting », *RIFRA'98, Rencontre Internationale sur l'Extraction le Filtrage et le Résumé Automatique*, Sfax, Tunisie, 1998, pp. 72-83.

[Salton *et al.* 1975] Salton G., Wong A., Yang C. – « A vector space model for information retrieval », *Communications of the ACM*, **18** (11), pp. 613–620.

[Saussure 1995] Saussure F. de – *Cours de linguistique générale,* Paris, Payot, 1995 (Paris, Payot, 1916).

[Schmid *et* Baccino 2002] Schmid S., Baccino T. – « Perspective-Shift Effect and Text Format: An Eye-Tracking Study », *Current Psychology Letters: Behaviour, Brain & Cognition, 9*, pp. 73-87.

[Searle 1969] Searle, J. – *Speech Acts, An Essay in the Philosophy of Language*, Cambridge, Cambridge University Press, 1969.

[Shneiderman 1991] Shneiderman B. « A Taxonomy and Rule Base for the Selection of Interaction Styles », dans [Baecker *et al.* 1995], pp. 401-410, 1991.

[Shneiderman 1998] Shneiderman B. – *Designing the User Interface: strategies for effective Human-Computer interaction*, Addison-Wesley, 1998.

[Smith 1996] Smith P.A. – « Towards a practical measure of hypertext usability. », *Interacting with Computers 8 (4)*, pp. 365-381, 1996.

[Souchier 2002] Souchier E. – « Lorsque les écrits de réseaux cristallisent la mémoire des outils, des médias et des pratiques… », chapitre dans *Les défis de la publication sur le Web – Hyperlectures, cybertextes et méta-éditions*, coordonné par Jean-Michel Salaün et Christian Vandendorpe, presses de l'enssib, 2002.

[Souchier *et* Jeanneret 1999] Souchier E., Jeanneret Y. – « Pour une poétique de l'écrit d'écran », *Le multimédia en recherche, Xoana. Images et sciences sociales*, n° 6-7, 1999.

[Souchier *et* Jeanneret 2002] Souchier E., Jeanneret Y. – « Écriture numérique ou médias informatisés ? », *Pour la science-Scientific american*, Dossier n°33, *Du signe à l'écriture*, octobre-janvier 2002.

[Stanton *et al.* 2000] Stanton N., Correia A.P., Dias P. – « Efficacy of a map on search, orientation and access behaviour in a hypermedia system », *Computers & Education*, 35, 2000, p. 263-266.

[TEI] Text Encoding Initiative – http://www.tei-c.org

[Tullis 1981] Tullis T.S. – « An evaluation of alphanumeric, graphic and color information displays ». *Human Factors, 23*, 1981, pp. 541-550.

[Van Dijk *et* Kintsch 1983] Van Dijk T.A., Kintsch W. – *Strategies of discourse comprehension*. Academic Press, New York, 1983.

[Vandendorpe 1999] Vandendorpe C. – *Du papyrus à l'hypertexte. Essai sur les mutations du texte et de la lecture*. Éditions La Découverte, Paris, 1999.

[Vandendorpe 2000] Vandendorpe C. – « Livre virtuel ou codex numérique ? Les nouveaux prétendants. », *Bulletin des bibliothèques de France*, 45, 6, pp. 17-22, 2000.

[Virbel 1989] Virbel J. – « The contribution of Linguistic Knowledge to the Interpretation of Text Structures » In *J. André, V. Quint & R. Furuta (editors) Structured document,* Cambridge University Press, 1989, pp. 181-190.

[Ward 1994] Ward M. – « Xmdvtool: Integrating multiple methods for visualizing multivariate data », *in Proceedings of Visualization '94*, pp. 326–33.

[Webber *et* al. 2003] Webber B., Knott A., Stone M., Joshi A. – « Anaphora and Discourse Structure », *Computational Linguistics 29(4)*, pp. 545-588, 2003.

[Weinschenk *et al*. 1997] Weinschenk S., Jamar P., Yeo S. – *GUI design essentials*. Wiley Computer Publishing, 1997.

[Wikimedia] Wikimedia - Wikipedia, the free enciclopedia – http://en.wikipedia.org/wiki/Wikimedia

[Wikipedia] Wikipedia, the free encyclopedia – http://en.wikipedia.org/wiki/Main_Page

[Wolf *et* Gibson 2005] « Representing Discourse Coherence: A Corpus-Based Study », *Computational Linguistics, Vol. 31, No. 2*, pp. 249-288, juin 2005.

[Wonsever 2004] Wonsever D. – *Repérage automatique des propositions par exploration contextuelle*, Thèse de doctorat, Université Paris-Sorbonne, Paris, 2004.

[XLink] XML Linking Language (XLink) Version 1.0 – http://www.w3.org/TR/xlink

[XPointer] XML Pointer Language (XPointer) Version 1.0 – http://www.w3.org/TR/WD-xptr

[XSLT] XSL Transformations – http://www.w3.org/TR/xslt