Sublim : un systeme universel de bases lexicales multilingues et Nadia : sa specialisation aux bases lexicales interlingues par acceptions

Gilles Serasset

THÈSE

présentée par

Gilles SÉRASSET

pour obtenir le titre de

DOCTEUR DE L'UNIVERSITÉ JOSEPH FOURIER — GRENOBLE 1

(ARRÊTÉS MINISTÉRIELS DU 5 JUILLET 1984 ET DU 30 MARS 1992)

Spécialité

INFORMATIQUE

SUBLIM : un système universel de bases lexicales multilingues et **NADIA** : sa spécialisation aux bases lexicales interlingues par acceptions

8 décembre 1994

Composition du jury :

Président	Marie-France	BRUANDET
Rapporteurs	Helmut	SCHNELLE
	Jean	VÉRONIS
Examinateurs	Vincent	QUINT
	François	RECHENMANN
Directeur	Christian	BOITET

THÈSE PRÉPARÉE AU SEIN DU LABORATOIRE GETA (IMAG, UJF & CNRS)

Remerciements

Au moment d'amener les voiles, le marin sait qu'il ne doit son arrivée au port qu'aux vents qui ont su l'y mener.

Au vif zéphyr (Christian Boitet) qui, jubilant sur les kumquats du clown gracieux et non content d'avoir affrété le brigantin, a su gonfler ses voiles et le diriger sans abattre la mâture.

À l'aquilon (François Peccoud) qui m'a accueilli dans son aire et dont la force et la persévérance m'ont empêché d'affaler.

À Éole (Marie-France Bruandet) et aux quatres vents (Vincent Quint, François Rechenmann, Helmut Schnelle et Jean Véronis), qui ont su étudier les nœuds du mousse et y voir un travail de matelot.

À tous les vents de la rose, autan, mistral, khamsin, sirocco, harmattan, noroît, simoun, … (membres du GETA, équipe de B'VITAL, …) qui, par babord ou tribord, gonflant brigantine ou huniers, portant chant de sirènes ou arome de terre, rafraichissant cale et carré, réchauffant marins et matelots, ont aidé esquif et équipage à maintenir le cap vers de nouveau rivages.

Je ne peux oublier le brick (Mathieu Lafourcade) qui à quelques encâblures, cinglait, tout comme moi, dans le sillage de la goélette (Hervé Blanchon) qui a ouvert la marche.

Au foehn (Maria), qui a su prodiguer douceur et chaleur lorsque se profilaient sargasses et dérives dans le cœur du matelot.

À tous les vents que je ne peux citer ici, qu'ils me pardonnent et sachent que je n'oublie pas leur souffle enivrant.

Table des matières

Liste des figures

Introduction

Les besoins en ressources lexicales de grande taille sont de plus en plus importants. Les causes en sont multiples : le développement commercial de nombreux systèmes de Traitement Automatique des Langues Naturelles (TALN), le volume des textes à traduire, qui augmente dans de nombreuses institutions et en particulier à l'Union Européenne, l'apparition d'outils d'aide à la traduction humaine, qui nécessitent une couverture générale de la langue, etc.

Il faut donc développer des dictionnaires qui soient à la fois à usage humain et à usage "machinal". Différents outils existent déjà pour faciliter le développement de bases lexicales, comme par exemple le gestionnaire de dictionnaires de METAL (SNI-Germany) spécialisé dans le traitement des bases lexicales de systèmes de traduction automatique. Parallèlement, la TEI (Text Encoding Initiative) propose des standards de codage de dictionnaires imprimés (à usage humain). Entre les deux, de nombreux efforts ont été accomplis sur la définition de gestionnaires de bases lexicales indépendants des applications. Parmi ces efforts, le plus poussé est le projet ESPRIT MULTILEX. Néanmoins, ce projet présente quelques faiblesses (dictionnaires bilingues par transfert, obligation de coder les structures linguistiques sous forme de structures de traits typés…). Nous nous proposons de définir un système de gestion de bases lexicales multilingues (BDLM) en partant des acquis de projets tels que MULTILEX.

Le projet Sublim apporte des nouveautés par rapport aux différents systèmes de gestion de bases lexicales multilingues. Il permet de spécifier l'architecture lexicale d'une base particulière en utilisant des dictionnaires monolingues, bilingues ou interlingues dont la gestion globale (accès aux unités, structures squelettes…) est prise en charge par le système. Les unités des dictionnaires, ainsi que les informations qu'elles portent, ne sont pas contraintes. Cela permet d'utiliser SUBLIM pour implémenter des bases lexicales "fondées sur la connaissance" inspirées de projets tels que EDR (Japon) ou KBMT-89 (USA). On peut aussi implémenter des bases lexicales fondées sur une approche par transfert comme celles du projet Multilex ou comme les dictionnaires de METAL.

Le linguiste peut définir l'architecture linguistique des différents dictionnaires de sa base lexicale. Pour cela, il choisit les structures logiques servant de base à ses structures linguistiques parmi une importante collection (automates, graphes, arbres, structures de traits typés, ensembles, listes…). En combinant ces structures, on peut définir des structures

linguistiques complexes d'une manière naturelle. Cette approche universelle permet la création de bases lexicales pour des usages différents, automatiques aussi bien qu'humains.

Lorsque l'on veut construire des bases lexicales comportant de nombreuses langues, une approche interlingue semble s'imposer. Cependant, les projets EDR et KBMT, fondés sur la connaissance "extralinguistique" du domaine du discours, ont rencontré des problèmes théoriques complexes (raffinement des concepts, classification et exceptions, description des concepts…), avec pour conséquence un coût élevé de développement et des problèmes de cohérence.

C'est pourquoi, comme le projet ULTRA, nous avons choisi pour nos applications de privilégier une architecture lexicale interlingue fondée sur les connaissances linguistiques plutôt que sur les connaissances extralinguistiques. Le lexique "pivot" n'est alors plus formé de "concepts" (indépendants des langues), mais "d'acceptions interlingues" fonctions des langues en présence. Les bases utilisant cette architecture sont appelées "bases NADIA".

Nous avons développé autour de SUBLIM des outils facilitant la gestion de bases NADIA. Dans une base NADIA, on définit deux nouvelles classes de dictionnaires héritant des classes monolingues et interlingues de SUBLIM. On introduit aussi de nouvelles méthodes pour leur gestion, ainsi que des "unités dictionnairiques" (acceptions monolingues, acceptions interlingues…) dont le linguiste hérite pour définir son architecture linguistique.

Pour que SUBLIM (et a fortiori NADIA) puisse être utilisable pour définir, construire et maintenir des dictionnaires de grande taille, à usage "machinal" aussi bien qu'humain, il doit intégrer des outils conviviaux permettant d'éditer, de parcourir et de manipuler des structures complexes dans différents modes, textuels et graphiques.

Pour cela, nous utilisons Grif, un puissant éditeur de documents structurés. L'utilisation de différentes vues d'une même structure permet la création et le formatage d'un dictionnaire sous diverses formes (textes SGML, formulaires, graphiques, dictionnaires imprimables…). Pour la maintenance, il est également intéressant de visualiser une structure linguistique sous différentes formes, avec possibilité de cacher une partie de l'information.

Nous définissons aussi d'autres outils facilitant la gestion et l'exploitation des dictionnaires (vérification de cohérence, import/export, règles de valeurs par défaut…). Ces outils sont organisés au sein d'une architecture à trois niveaux séparant clairement les problèmes de stockage, de manipulation et de visualisation.

Dans la première partie de ce document, nous présentons les travaux qui ont été effectués dans le domaine. Cela nous permet ensuite d'analyser les problèmes des dictionnaires et d'évaluer les solutions qui ont été proposés par les auteurs des différents systèmes.

La seconde partie définit le projet SUBLIM de système universel de gestion de bases lexicales multilingues. Nous verrons comment on peut, avec ce système, définir une base lexicale multilingue (en définissant l'ensemble de ses dictionnaires et leurs architectures linguistiques). Nous étudierons ensuite l'architecture logicielle et les outils définis dans ce système.

Dans la troisième partie, nous spécialisons SUBLIM à l'approche interlingue par acceptions. Nous exposons les principes de cette architecture lexicale interlingue fondée sur une les connaissances linguistiques. Nous montrons ensuite que cette architecture lexicale est propice à l'utilisation de fonctions lexicales "à la Melčuk". Nous présentons enfin PARAX, la maquette d'une base lexicale utilisant cette approche, et dégageons les caractéristiques génériques de cette approche. Enfin, nous développons l'implémentation de NADIA, un gestionnaire de bases lexicales interlingues par acceptions, qui se présente du point de vue logique et informatique comme un spécialisation du système SUBLIM.

État de l’art et problèmes intéressants

I. Des dictionnaires sur papier aux bases lexicales

Les dictionnaires ont une longue histoire. Le mot lui-même est introduit dès 1539, avec le dictionnaire français-latin de R. Estienne. Auparavant, on parlait de “thesaurus” pour les dictionnaires en une seule langue.

Les dictionnaires électroniques sont apparus dès les débuts de l’informatique, et se développent aujourd’hui de façon spectaculaire.

Le besoin en dictionnaires électroniques est motivé principalement par deux mouvements :

- porter des dictionnaires existants sur un support électronique afin de simplifier leur accès ou pour les intégrer à des environnements informatiques de rédaction,
- construire des dictionnaires à usage informatique, c’est à dire utilisés comme données linguistiques par des processus automatiques.

Depuis quelques années, ces deux mouvements tendent à se rapprocher, notamment par l’apparition de travaux visant à la construction de bases lexicales à usage humain et/ou à usage automatique.

Bien que nos travaux se veuillent indépendants de l’information linguistique et lexicale associée aux différentes entrées du dictionnaire, il est important d’étudier les différents dictionnaires qui existent, afin d’en dégager les particularités, pour garantir la conservation de leurs fonctionnalités.

Au cours de ce “voyage au pays des lexiques”, le lecteur doit garder à l’esprit quelques questions, dont les réponses donnent les particularités de chacun des dictionnaires observés :

- *Comment accède-t-on à l’information ?*
- *Quelle est l’unité d’information (l’unité lexicale) ?*
- *À quelle information accède-t-on ?*
- *Comment l’information est-elle codée ?*

Dans la suite de ce chapitre, nous tâcherons de répondre à ces questions pour les différents dictionnaires étudiés, obtenant ainsi une idée de l’étonnante diversité du monde des dictionnaires (électroniques ou non).

Nous développons ce tour d’horizon en suivant la chronologie, c’est à dire en évoquant tout d’abord les dictionnaires papier afin d’avoir une illustration de la richesse des informations qui peuvent être présentes dans un dictionnaire. Nous continuons en étudiant les dictionnaires sur support électronique et nous terminons en étudiant les bases lexicales spécialisées, qui sont des dictionnaires construits pour un usage automatique.

1. Dictionnaires sur papier

Le petit Robert définit un dictionnaire comme étant un *recueil de mots rangés dans un ordre convenu qui donne une définition ou des informations sur les signes.*

Nous verrons que sous cette définition se cache une incroyable diversité de types d'information (définition, étymologie…), de moyen d'accès (dictionnaires de kanjis, de lemmes, par unités lexicales complexes…) de formatage, d'usage, de sélection des mots, etc.

1.1. Dictionnaires monolingues

Les dictionnaires monolingues sont les plus répandus. Qui n'a pas un petit Robert ou un Larousse, voire un Hachette dans sa bibliothèque ? Nous connaissons tous le moyen de les utiliser. Et pourtant, ici aussi, on peut avoir des surprises.

1.1.1. Un dictionnaire classique : le petit Robert

Dans un dictionnaire classique, l'entrée se fait par le lemme et la catégorie. On trouve ainsi deux articles différents pour le lemme devant (devant - préposition adverbiale et devant - nom masculin). Une entrée peut être simple (devoir, constater…) ou composée (pomme de terre, deus ex machina…).

On peut accéder à une unité du lexique par plusieurs entrées (ex : DICO n.m. v. DICTIONNAIRE).

Une unité lexicale est ensuite découpée en différents sens. Notons qu'il y a autant de découpages qu'il y a de dictionnaires monolingues. Dans le Robert par exemple, les verbes sont découpés selon leur comportement syntaxique (verbes transitifs ou intransitifs) puis selon leur sens.

On trouve de nombreuses informations dans ces dictionnaires. Pour chaque lemme, on a sa prononciation, sa catégorie, son genre et son étymologie. Pour chaque sens, on a sa définition, des exemples, des références à d'autres entrées synonymes ou antonymes, un domaine voire un niveau de langue.

Composer v. (XII^{e}; lat. componere, d'apr. poser).

I. V. tr. ◊**1°** (1559). Former par l'assemblage, la combinaison de parties. V. **Agencer, arranger, assembler, constituer, disposer, faire, former, organiser.** *Composer un remède, un breuvage, un plat.* V. **Confectionner, préparer.** *Pièces qui composent une machine.*— Fig. «*Tu composes dans ta jeunesse l'homme mûr, le vieillard que tu seras*» (Mac Orlan). ◊ **2°** (v. 1480) Faire, produire (une oeuvre). V. **Bâtir, créer, écrire, produire.** *Composer un livre, un poème, des vers.* ◊ **3°** (1690). Écrire (une œuvre musicale). *Composer une sonate, un chœur.* — Absolt. *C'est un grand interprète, mais il ne compose pas.* ◊ **4°** Imprim. (1621). Assembler des caractères pour former (un texte). «*Il eut, le premier, fini de composer 4 lignes*» (Duham.). *Le texte est composé, on va commencer le tirage.* ◊ **5 °** (1559). Élaborer, adopter (une apparence, un comportement). V. **Affecter.** *Composer son attitude, son maintien* (se donner, prendre une contenance). *Composer son visage, ses paroles.* V. **Étudier.** *Se composer un visage de circonstance.*

II. *V. intr.* ◊ **1°** (xv^{e}) S'accorder (avec qqn ou qqch.) en faisant des concessions. V. **Accommoder (s'), entendre (s'), traiter, transiger.** *Composer avec ses créanciers. Composer avec l'ennemi.* V. **Pactiser.** *Composer avec sa conscience.* «*Je fus lâche, et je composai avec ma déception*» (Colette). ◊ **2°** Faire une composition. *Les élèves sont en train de composer.*

III. SE COMPOSER. *v. pron.* ◊ **1°** Être composé de . V. **Comporter, comprendre.** *La maison se compose de deux étages.* ◊ **2°** Se faire, Se former. «*Les choses de la vie, comme les ondes de l'océan, se composent et se décomposent sans cesse*» (Hugo). ◊ **3°** (*Récipr.*). Se mêler en s'organisant (de plusieurs éléments). ◊ **4°** (Réfl.). *Vieilli.* Composer son attitude. «*l'art de se composer*» (Beaumarch.). ◊ ANT. *Analyser, décomposer, défaire, dissocier.*

Figure 1.1 : Article "composer" du petit Robert (édition de 1970)

Cette information est destinée à un usage humain. Aussi, on utilise différents styles de caractères pour dénoter différents types d'information. Ces styles sont en nombre limité, d'où une surcharge de certains styles (qui deviennent donc ambigus). C'est par sa connaissance de

la langue que l'utilisateur peut restituer le type d'information présent dans un article du dictionnaire.

1.1.2. Un dictionnaire d'unités complexes : le LOGOS

D'autres dictionnaires monolingues se distinguent par le fait que leur unités lexicales sont différentes de celles des dictionnaires classiques évoqués plus haut. Parmi ces dictionnaires, on peut citer le LOGOS de Bordas.

Les entrées du LOGOS correspondent à des familles de lemmes dérivés du même lemme d'origine, conformément à des schémas dérivationnels réguliers. Cette unité lexicale est notée par son lemme d'origine. Par exemple, l'unité lexicale *produire* regroupe les lemmes *produire, producteur, productif, productivité, produit.*

L'entrée de ce dictionnaire se fait par le lemme origine de l'unité lexicale. Ainsi, si l'on cherche le lemme *producteur*, il faut savoir qu'il appartient à l'unité lexicale *produire* pour trouver ses informations associées.

L'information d'une unité lexicale du LOGOS est composée de deux parties :

- La liste des sens du lemme vedette,
- La liste des dérivations de l'unité lexicale.

Pour chaque sens et pour chaque dérivation du lemme vedette, on a les mêmes informations que dans un dictionnaire classique. De plus, ces informations apparaissent sous une forme analogue.

construire v. t. (latin *construere*, même sens).

❶ Édifier ou faire édifier : *construire une maison, un pont, une digue…* — (absolument) *On construit beaucoup actuellement dans les banlieues des grandes villes; j'ai acheté un terrain, car j'ai l'intention de faire construire.* — (*par extension*) Réaliser ou faire réaliser (une chose qui suppose un plan préalable et un travail considérable et complexe) : *les chantiers navals construisent les navires; construire un alternateur, des camions…*

❷ (*figuré*) Composer, former selon un plan ou un système plus ou moins rigoureux : *construire l'intrigue d'un roman, le plan d'un exposé; cette dissertation n'est pas construite* : n'a pas de plan logique et net; *construire une théorie philosophique.* — (*spécialement, grammaire*) Disposer les mots ou les propositions à l'intérieur d'une phrase, selon un ordre déterminé : *construire une phrase, une période oratoire. — Construire une phrase latine, grecque…* : disposer les mots qu'elle contient de manière à les ordonner selon l'ordre analytique (sujet, verbe, compléments…), avant de la traduire.

cf. *bâtir, disposer, dresser, édifier, élever, établir, fabriquer, réaliser; agencer, arranger, articulier, assembler, combiner, composer.* — *ANT. abattre, démolir, détruire.*

◆ **se construire** v. pron. **1°** (*sens passif*) Être en cours de construction : *un pont se construit actuellement à la sortie du village.* — Être construit : *un tel immeuble ne peut se construire en moins de six mois.* — (grammaire) *Se construire avec…*, s'employer avec… : *le verbe* apprendre *peut se construire avec* à *suivi de l'infinitif* (j'apprends à lire). — **2°** (*sens réfléchi indirect*) …

◆ **constructeur, trice** n. m. *ou* adj. **1°** n. m. Celui qui construit : *un constructeur de bateaux, d'avions; un constructeur-promoteur* (voir *promoteur*). — (*figuré*) *Alexandre le Grand fut le constructeur d'un immense empire.* — **2°** adj. (*néologisme*) Se dit des idées dont la réalisation permettrait un progrès quelconque : *une proposition constructrice.* • N. B. : mieux : constructif.

◆ **constructif, ive** adj. Qui est naturellement fait pour construire : *une intelligence constructive.* — Qui permet de réaliser un progrès : *un programme constructif.*

◆ **construction** n. f. **1°** action d'édifier, de construire : *une maison en construction; construction d'un navire; chantier de construction navale; …* — **2°** Manière dont une chose est construite : *la construction préfabriquée permet de réduire le prix de revient des immeubles;…* — **3°** Édifice : *raser une ancienne construction.* — **4°** (*figuré*) Action de composer, de réaliser selon un plan; la manière dont un ouvrage est composé, organisé : *la construction de l'intrigue d'une comédie;…* — **5°** (*spécialement, grammaire*) Ordre dans lequel les mots se présentent dans une phrase : *la construction allemande diffère de la construction française.* — …

Figure 1.2 : L'unité lexicale complexe "construire". Cette unité regroupe les lemmes *construire*, *constructeur*, *constructif* et *construction*

1.2. Thesaurii

Les thesaurii sont des dictionnaires de concepts. L'accès à ce type de dictionnaire ne se fait pas par une forme graphique. Les concepts sont classés selon leur sens dans une hiérarchie de concepts utilisée pour la recherche. Nous allons étudier deux de ces thesauri.

1.2.1. Le Rodget's Thesaurus of English Words and Phrases

Le Rodget thesaurus comporte 1000 unités. Ces unités sont classées selon 39 catégories réparties dans 7 classes. L'entrée de ce thesaurus est un nombre (de 1 à 1000). Pour accéder à ce dictionnaire, il faut savoir à quelle classe et à quelle catégorie appartient l'article recherché. On peut ainsi chercher l'article en question grâce à une table organisée hiérarchiquement.

Notons que, grâce à un index, on peut aussi accéder à un article par un mot qui le désigne.

Un article de ce lexique est représenté par un entier et par un mot vedette. L'article est ensuite décomposé en catégories linguistiques (nom, verbe, adjectif…). Pour chaque catégorie linguistique, on a un ensemble de mots (classés par signification) qui sont liés à ce concept. Certains de ces mots apparaissent avec une référence à un autre concept. Enfin, on a un ensemble de liens vers des concepts sémantiquement liés au concept courant.

Si on lit le texte d'un article, on trouve des mots sémantiquement liés. Le sens des mots trouvés évolue peu à peu vers des contextes différents. Pour rendre cette transition plus distincte, certains mots sont parfois ajoutés comme pointeurs du contexte plutôt que comme éléments de vocabulaire.

480 **Judgment**: conclusion

N. *judgment*, judging (see *estimate*); good judgment, discretion 463 n. *discrimination*; bad judgment, indiscretion 464 n. *indiscrimination*; power of judgment, deiscretionary judgment, arbitrement 733 n. *authority*; arbitration, arbitrage, umpirage; judgment of facts, verdict, finding; penal judgment, sentence 963 n. *punishment*; spoken judgment, pronouncement; act of judgment, decision, adjudication, award; order, ruling; order of the court 737 n. *decree*;…

estimate, estimation, view 485 n. *opinion*; assessment, valuation, evaluation, calculation 465 n. *measurement*; consideration, ponderation; comparing, contrasting 462 n. *comparison*; transvaluation 147 n. *conversion*; appreciation, appraisal, appraisement; criticism, constructive c. 703 n. *aid*; destructive criticism 702 n. *hindrance*; critique, crit, review, notice, press n., comment, comments, observations, remarks 591 n. *dissertation*;…

estimator, judge, adjudicator; arbitrator, umpire, referee; surveyor, valuer 465 n. *appraiser*; inspector, inspecting officer, referendary, reporter, examiner 459 n. *enquirer*; counsellor 691 n. *adviser*; censor, critic, reviewer, commendator 591 n. *dissertator*; …

Adj. *judicial*, judicious, judgmatic 463 adj. *discriminating*; unbiased, dispassionate 913 adj. *just*; juridical, juristic,…

Vb. *judge*, sit in judgment, hold the scales; arbitrate, referree; hear, try, hear the case, try the cause 955 vb. *hold court*; uphold an objection, disallow an o.;…

estimate, form an e., make an e., measure, calculate, make 465 vb. *gauge*; value, evaluate, appraise;…

Adv. *sub judice*, under trial, under sentence.

See: 147, 438, 449, 462…

Figure 1.3 : Un article du Rodget's thesaurus

Les mots en italiques en début de paragraphe représentent les sous-concepts. La virgule est utilisée pour séparer les mots de même sens. Le point-virgule sépare des sens légèrement différents.

Ce thesaurus se présente donc comme un réseau de mots, liés sémantiquement, et auxquels on accède grâce à une classification hiérarchique de concepts.

1.2.2. Le vocabulaire de traitement de l'information (IFIP & ICC)

Le vocabulaire de traitement de l'information de l'IFIP (International Federation for Information Processing) et l'ICC (International Computation Center) est un dictionnaire

terminologique qui se présente comme un thesaurus. Les articles de ce dictionnaire sont rangés selon 20 catégories, classées dans 6 sections principales.

Chacune de ces catégories est décomposée de manière hiérarchique en termes :

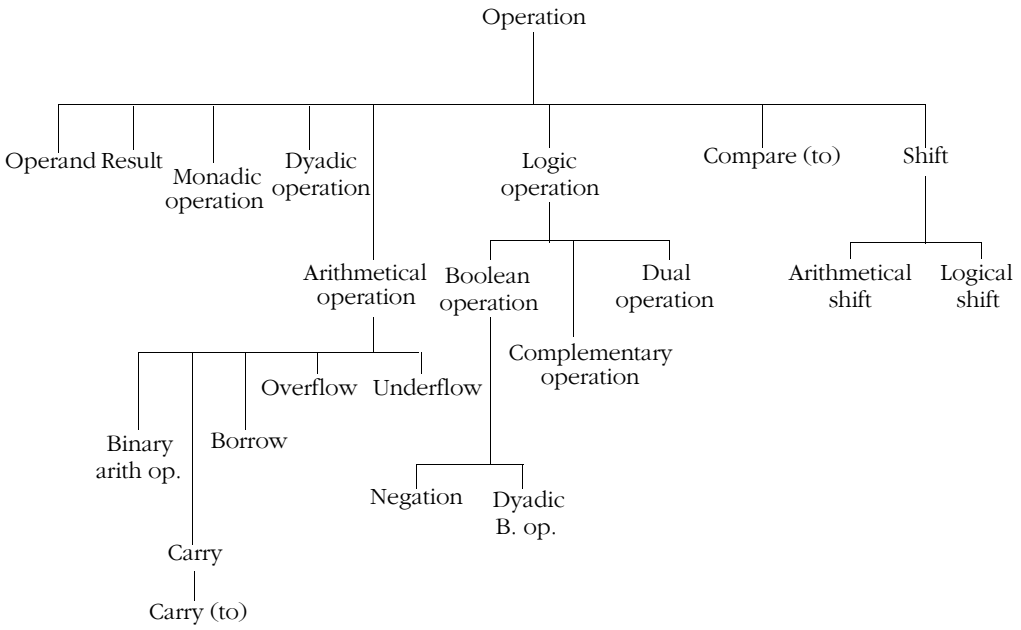


Figure 1.4 : Structure hiérarchique de la section G (Arithmetical and logic operations) du vocabulaire de traitement de l'information.

Chaque terme est associé à un symbole composé d'une lettre (la catégorie) et d'un chiffre (numéro du terme).

Les termes sont ensuite décrits, soit par une définition complète, soit en apparaissant à l'intérieur de la définition d'un autre terme. Ainsi, les termes *result* (G2) et *operands* (G3) apparaissent dans la définition de *operation* (G1) :

G1 G2 G3	OPERATION	A general term for any well-defined action, especially the derivation of a unit of data (the RESULT) from one or more given units of data (the OPERANDS), according to defined rules which specify the result for any permissible combination of values of the operands. Note: The term operands is sometimes used collectively for both operands and results since the results will often be operands in subsequent operations.

Figure 1.5 : Un exemple d'article du vocabulaire de traitement de l'information

Notons qu'un index permet de retrouver un article du dictionnaire à partir d'un terme.

Un article de ce dictionnaire contient une définition. On peut aussi trouver des notes ou des exemples illustrant le terme décrit. La catégorie linguistique des termes n'est pas indiquée.

Les définitions sont données de manière précise plutôt que concise. Les termes utilisés dans les définitions et définis par ailleurs dans le dictionnaire sont indiqués en italiques. Des alternatives au terme sont notées en souligné.

1.3. Dictionnaires bilingues

Les entrées des dictionnaires bilingues sont quasiment les mêmes que celles des dictionnaires monolingues classiques. La différence provient des entrées composées (pomme de terre…) qui ne sont plus considérées comme des entrées mais comme des parties d'un article (pomme…). Ces entrées (les lemmes) donnent accès à des traductions des différents sens du lemme dans la langue cible.

Ces dictionnaires sont bien souvent construits en collaboration entre deux éditeurs de dictionnaires monolingues. Ainsi, la décomposition en sens du dictionnaire bilingue est identique à la décomposition en sens du dictionnaire monolingue source. Par contre, la forme est assez différente, puisqu'elle est négociée entre les deux éditeurs. Ainsi, le petit Robert utilise des chiffres pour numéroter ses sens, alors que le Robert & Collins utilise des lettres.

composer 1 vt **(a)** (*confectionner*) *plat, médicament* to make (up); *équipe de football etc.* to select; *assemblée, équipe scientifique* to form, set up. **l'étalagiste compose une belle vitrine** the window dresser is arranging *ou* laying out *ou* setting up a fine display.

(b) (*élaborer*) *poème, lettre, roman* to write, compose; *symphonie* to compose; *tableau* to paint; *numéro de téléphone* to dial; *projet, programme* to work out, draw up; *couleurs, éléments d'un tableau* to arrange harmoniously; *bouquet* to arrange, make up.

(c) (*constituer*) *ensemble, produit, groupe* to make up; *assemblée* to form, make up. **pièces qui composent une machine** parts which (go to) make up a machine; **ces objets composent un ensemble harmonieux** these objects form *ou* make a harmonious group.

(d) (Typ) to set.

(e) (*frm: étudier artificiellement*) **~ son visage** to assume an affected expression; **~ ses gestes** to use affected gestures; **attitudes/allures composées** studied behaviours/manners; **il s'était composé un personnage de dandy** he had established his image as that of a dandy; **se ~ un visage de circonstance** to assume a suitable expression.

2 vi **(a)** (*Scol*) **~ en anglais** to sit (*surtout Brit*) *ou* take an English test; **les élèves sont en train de ~** the pupils are (in the middle of) doing a test ou an exam.

(b) (*traiter*) to compromise. **~ avec** *adversaire etc.* to come to terms with, compromise with.

3 se composer vpr (*consister en*) **se ~ de** *ou* **être composé de** to be composed of, be made up of, consist of, comprise; **la vitrine se compose** *ou* **est composée de robes** the window display is made up of *ou* composed of dresses.

Figure 1.6 : Un exemple d'article du Robert & Collins

Les sens du Robert monolingue se retrouvent assez facilement dans cet article de dictionnaire. On observe que ces sens sont raffinés à un niveau supplémentaire par différents contextes. Ces contextes servent au choix de la traduction anglaise désirée. On observe aussi de nombreux exemples qui, eux aussi, servent de contexte.

Ces dictionnaires ne peuvent être utilisés que dans un sens. Ils sont "mono-directionnels".

1.4. Dictionnaires multilingues

Des dictionnaires multilingues ont aussi été construits. Lorsqu'on aborde le multilinguisme, on peut envisager deux types de dictionnaires papier :

- les dictionnaires "1 vers n",
- les dictionnaires "n vers n".

Les dictionnaires "1 vers n" sont des dictionnaires bilingues "améliorés". Ils ont le même type d'entrées, et les informations classiques de plusieurs dictionnaires bilingues sont regroupées dans un même article.

Les dictionnaires "n vers n" sont plus intéressants puisqu'ils ne privilégient aucune langue source. Chaque langue peut être utilisée pour accéder aux équivalents dans les autres langues. Les contraintes linguistiques font que ce type de dictionnaire est limité à des domaines terminologiques.

Le dictionnaire des sciences de l'information (Moscou 1975) est un dictionnaire terminologique de 2235 termes en russe, bulgare, hongrois, espagnol, macédonien, allemand, polonais, roumain, serbo-croate, slovaque, slovène, tchèque, anglais et français. Ce dictionnaire comprend deux parties.

La première contient la liste des termes et de leurs synonymes en ordre alphabétique et de leur définition en langue russe. Chacun des termes est accompagné des termes équivalents

correspondants dans les autres langues. Chaque terme du dictionnaire terminologique comporte un numéro d'ordre ; ce numéro accompagne le terme et permet de retrouver la notion correspondante. Après chaque terme, un chiffre désigne la rubrique à laquelle appartient ce terme, conformément à une subdivision hiérarchique en rubriques thématiques.

1629	**редакционная коллегия** (08.3) Совещательный орган, включающий официально утвержденный состав специалистов, осуществляющих общее научное и организационное руководство по подготовке какого-либо издания и выполняющих определенные редакционные функции.	(б) редакционна колегия (в) szerkesztő bizottság (и) consejo de redacción (м) редакциси колегиум (н) Redaktioskollegium (п) kolegium redakcyjne (р) comitet de redacţie	(сх) redakcijski kolegijum (с) redakčná rada (сл) uredniški odbor (ч) redakční rada (а) editorial board, body of editors (ф) comité *m* de rédaction

Figure 1.7 : Une entrée du dictionnaire terminologique des sciences de l'information

La seconde partie du dictionnaire terminologique comporte les listes des termes en ordre alphabétique pour les autres langues, en tenant compte des diverses variantes possibles d'inversion des mots. Chaque terme équivalent est suivi du numéro d'ordre du terme correspondant en langue russe.

On peut accéder à ce dictionnaire par un lemme de n'importe laquelle des 14 langues considérées. On peut ainsi utiliser ce dictionnaire dans n'importe quel sens. Par contre, on n'a accès qu'a une définition russe des termes.

2. Dictionnaires sur support électronique

Le premier support qui a rendu les dictionnaires accessibles à la machine a été la bande de photocomposition. Ces bandes ne présentent pas un marquage descriptif des entrées en champs logiques (orthographe, catégorie, prononciation, définition…), mais un marquage typographique indiquant les changements de fontes, des caractères spéciaux ou des attributs de mise en page.

De nombreuses équipes se sont donc intéressées au problème de la transformation de ce format typographique en un format logique utilisable par la machine. Cet effort a abouti à l'apparition de dictionnaires (notamment le Longman Dictionary of Contemporary English, LDOCE) sur support informatique, sous un format logique.

Cet effort n'a pas pu se faire sans le choix d'un codage logique pour les informations lexicales. Nous présenterons donc un codage pour les dictionnaires, extrait des travaux de la Text Encoding Initiative (TEI).

Parallèlement, les éditeurs de dictionnaires se sont hâtés de prendre une place sur le marché des outils d'aide à la rédaction ou à la traduction, en proposant une version informatisée de leurs dictionnaires. Pour illustrer ce type d'outils, nous étudierons l'utilitaire Collins On-line sur Macintosh.

Dans le même temps, le travail des traducteurs s'effectuant de plus en plus sur ordinateur, on a développé des utilitaires permettant de créer, de gérer et de consulter des bases lexicales terminologiques. Nous nous intéresserons à l'un de ces outils : MTX Termex.

Enfin, l'utilisation du support informatique a permis d'étudier et de développer des dictionnaires électroniques ayant des fonctionnalités différentes de celles d'un dictionnaire papier. Cette étude a mené à la construction d'environnements de découverte lexicale dont nous étudierons un exemple : le Dicologique.

2.1. Un format de codage : SGML/TEI

La Text Encoding Initiative (TEI) étudie un standard de codage et d'échange de documents textuels. Ce standard se présente sous forme de recommandations et de structures de documents (pour la prose, les vers, les dictionnaire imprimés, les drames, les dictionnaires terminologiques…). Cette initiative est parrainée par l'Association for Computers and the Humanities (ACH), l'Association for Computational Linguistics (ACL) et l'Association for Literary and Linguistic Computing (ALLC).

Le format TEI est basé sur SGML (Standard Generalized Markup Language). SGML est un standard international (ISO 8879) pour la représentation de textes sous une forme électronique indépendamment de la machine et du système[1]. SGML utilise des étiquettes pour structurer les différents éléments d'un texte. Ces étiquettes sont notées entre chevrons (ex : `<paragraph>`) et agissent comme des parenthèses, l'étiquette fermante étant notée avec un "`/`" (ex : `</paragraph>`). Dans certaines conditions, l'étiquette fermante peut être omise.

Parmi les types de documents qui nous intéressent, la TEI a publié des standards pour le codage de dictionnaires imprimés et de bases terminologiques ([Sperberg-McQueen & Burnard 1994], chapitres 12 et 13).

La structure globale d'un dictionnaire est analogue à celle de textes usuels. On retiendra les éléments suivants :

`<text>`:	contient du texte de n'importe quelle sorte (structuré ou non),
`<front>`:	contient tout ce qui se trouve avant le début du dictionnaire lui même (Entêtes, page de titre, préface, dédicace…),
`<back>`:	contient tout ce qui se trouve après la fin du dictionnaire lui même (Annexes…),
`<body>`:	contient l'ensemble du texte du dictionnaire, sauf les parties *front* et *back*,
`<div>`:	contient une subdivision des parties *front*, *body* ou *back* du dictionnaire,
`<div0>`:	contient une subdivision (du plus haut niveau) du dictionnaire,
`<div1>`:	contient une subdivision (du niveau inférieur) du dictionnaire,
`<entry>`:	contient une entrée structurée du dictionnaire,
`<entryFree>`:	contient une entrée non conforme à la structure d'une entrée du dictionnaire,
`<superentry>`:	groupe les entrées d'un ensemble d'homographes.

Voici un exemple de structure de dictionnaire conforme aux recommandations de la TEI :

```
<body>
  <div0 type='dictionary'>
    <!-- English-French -->
    <entry>...</entry>
    <entry>...</entry>
    <!-- ... -->
  </div0>
  <div0>
    <!-- French-English -->
    <entry>...</entry>
    <entry>...</entry>
    <!-- ... -->
  </div0>
</body>
```

Les éléments *entry* et *entryFree* partagent les attributs **type** (entrée standard, homographe, référence croisée, affixe, abréviation…) et **clé** (une séquence de caractères reflétant la position alphabétique de l'entrée dans le dictionnaire).

---

[1] Une introduction à SGML est donnée en annexe A.

Le format de codage des dictionnaires permet aussi de coder de manière structurée les entrées d'un dictionnaire. La décomposition de l'entrée est effectuée grâce aux éléments suivants :

`<hom>`:	regroupe les informations propres à un homographe de l'entrée,
`<sense>`:	regroupe les informations propres à un sens,
`<form>`:	regroupe les informations sur la forme (orthographique et phonétique),
`<gramGrp>`:	regroupe l'information morpho-syntaxique d'une unité du dictionnaire,
`<def>`:	contient une définition,
`<trans>`:	contient du texte traduit et les informations associées,
`<eg>`:	contient un texte exemple contenant au moins une occurrence de l'entrée,
`<def>`:	contient une définition,
`<usg>`:	contient les informations sur l'usage,
`<xr>`:	contient des références croisées,
`<etym>`:	contient l'information étymologique,
`<re>`:	contient une entrée correspondant à une unité lexicale liée à l'entrée (expression, forme dérivée…),
`<note>`:	contient des annotations.

Le format de la TEI rend possible une décomposition plus fine de la structure d'une entrée de dictionnaire imprimé. Nous ne détaillerons pas cette décomposition ici. Le lecteur pourra se reporter à [Sperberg-McQueen & al. 1994].

2.2. Dictionnaires en ligne: le Collins On-Line

La plupart des éditeurs proposent aujourd'hui des versions informatiques de leurs dictionnaires. Ces versions reprennent les données et les fonctionnalités des dictionnaires papier.

Ces outils sont utilisés en complément d'un traitement de texte comme aide à la rédaction ou à la traduction de textes. Leurs fonctionnalités sont assez réduites, puisqu'elles ne permettent en général pas d'autre accès que l'accès alphabétique classique du dictionnaire papier.

Le Collins On-line, avec ses dictionnaires français-anglais et anglais-français, en est un exemple représentatif. Cet outil, avec ses dictionnaires, est le reflet du Robert et Collins, présenté plus haut.

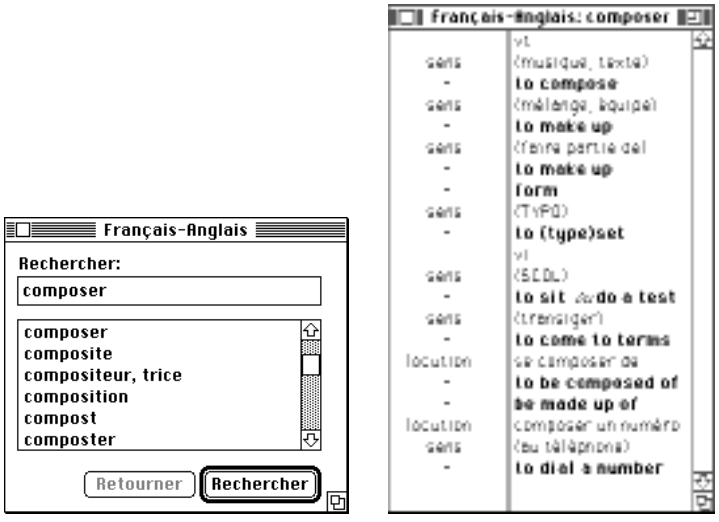


Figure 1.8 : L'entrée "composer" du Collins On-line, version Macintosh

La décomposition en sens et les équivalents sont les mêmes. Par contre, la forme est assez différente. Les contextes sont notés de manière plus succincte et les informations sont rendues sous forme de liste. En effet, les contraintes sont différentes. La présentation du dictionnaire papier est motivée par des contraintes de place, alors que la présentation du dictionnaire électronique est motivée par des contraintes de lisibilité à l'écran.

Des contraintes d'interface ont obligé les concepteurs à changer légèrement l'entrée du dictionnaire. En effet, le dictionnaire papier utilise le lemme et sa catégorie pour créer une entrée, alors que le dictionnaire électronique n'utilise que le lemme, les différentes catégories étant regroupées dans le même article.

Il est possible d'accéder rapidement à l'un des mots de la langue source apparaissant dans l'article, en le sélectionnant et en appuyant sur une combinaison de touches. Par contre, il n'est pas possible de faire la recherche rapide d'un mot de la langue cible dans le dictionnaire inverse.

2.3. Gestionnaires de dictionnaires terminologiques : MTX Termex

Le travail des traducteurs professionnels s'effectue maintenant en majorité sur support informatique. Il est donc crucial qu'ils disposent d'outils leur permettant d'avoir accès aux données lexicales sur leur machine, de manière intégrée à leur environnement de travail.

Ces traducteurs étant bien souvent spécialisés dans un domaine ou employés par des entreprises spécialisées, ils possèdent généralement leur propre terminologie. Des outils de gestion de dictionnaires terminologiques sont nécessaires, en plus des outils de consultation de dictionnaires classiques.

MTX Termex est un outil de ce genre. Il permet de consulter des dictionnaires du commerce, et aussi de créer son propre dictionnaire terminologique ou de modifier certaines entrées des dictionnaires du commerce.

Avec cet outil, une entrée est simplement une chaîne de caractères. Cette chaîne peut comporter des blancs et des symboles de ponctuation. L'information associée à cette entrée est un texte simple. On peut utiliser ce texte comme une structure "attributs-valeurs" en notant les attributs entre accolades.

Ainsi, on est libre de créer un dictionnaire ayant une structure quelconque.

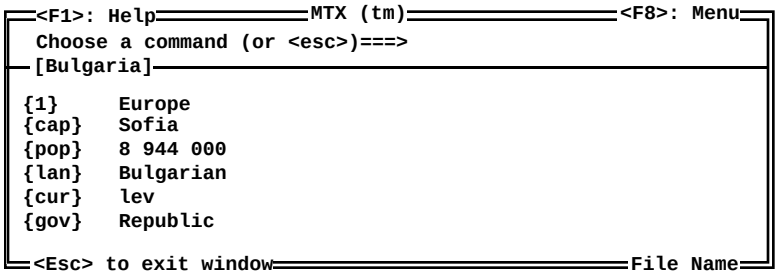


Figure 1.9 : Un exemple d'entrée d'un dictionnaire MTX Termex

Avec MTX 2, un traducteur peut très facilement créer un petit dictionnaire terminologique multilingue. Il paut aussi gérer les variantes orthographiques d'une entrée en les liant à l'entrée vedette.

La consultation et la navigation sont elles aussi assez simples (possibilité de créer des références croisées, de revenir aux fiches précédentes…). De plus, cet outil est intégré aux traitements de texte usuels (appel et consultation automatique de la sélection).

2.4. Un environnement original : le Dicologique

Le Dicologique est un produit commercialisé par la société MEMODATA (Caen) et fonctionnant sous DOS et Windows. Cet outil utilise véritablement les avantages de l'informatique pour renouveler les moyens d'accès au dictionnaire. En utilisant une approche ensembliste du lexique([Dutoit 1992]), il permet un accès par thèmes, analogies, idées…

Le Dicologique regroupe 120 000 entrées, 25 000 concepts et 350 000 relations. Les relations se font principalement au travers de :

thèmes : qui regroupent des éléments selon une approche analogique (ainsi, le thème PÊCHE regroupe une liste de pêcheurs, les techniques de la pêche, les bateaux de pêche…),

listes : qui regroupent des éléments proches par synonymie (ainsi la liste BATEAU DE MARCHANDISE regroupe péniche, cargo, méthanier, pétrolier…),

classes : qui regroupent des éléments de même classe dans une hiérarchie (ainsi, la classe NAVIRE ET BATEAU regroupe la classe des bateaux de pêche, la liste des bateaux de commerce…).

Les mots terminaux se trouvent au bout d'un graphe regroupant ces différents éléments :

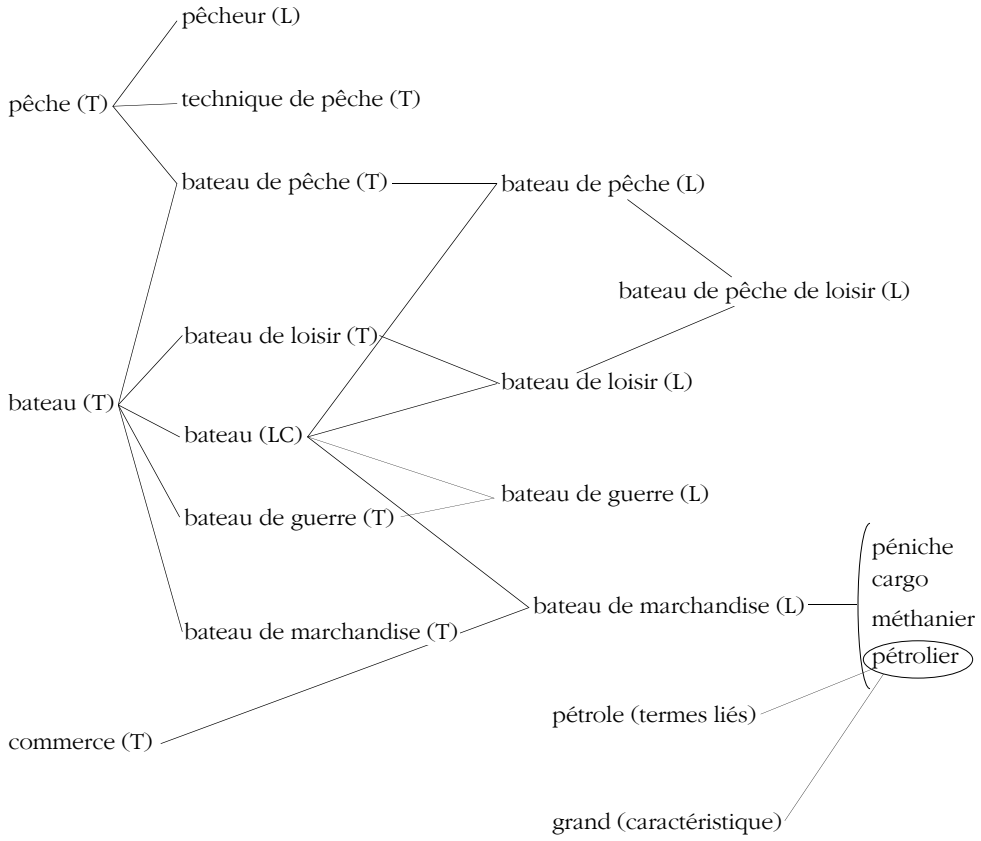


Figure 1.10 : Exemple de la structure des éléments du Dicologique

Il est possible d'accéder directement à un article en donnant son lemme. On obtient ainsi sa ou ses "quasi-définitions". Une quasi-définition est l'ensemble de classes, thèmes, listes… menant à ce mot. Ainsi, si l'on cherche le lemme "allège", on obtient l'information suivante :

```
1 : allège,                                    n.f.
  > appui, soutien, support (classe)           (448 Noms)
  > mur (description)                          (32 Noms)
  > fenêtre (termes liés)                      (15 Mots)
2 : allège
  > bateau (autre) (classe)                    (31 Noms)
  > chargement d'un navire (termes liés)       (21 Mots)
```

Il est possible, à partir de cette présentation de naviguer dans le Dicologique en sélectionnant l'un des éléments de la quasi-définition.

Le principal accès à ce dictionnaire reste un accès par idée. L'utilisateur peut construire des requêtes sous formes d'expressions booléennes. Les opérateurs sont ET, OU, SAUF, il y a une possibilité de parenthésage, et les opérandes sont des éléments (classe, liste, thème ou autre) du Dicologique.

Ainsi, la requête : `A ET B`, avec les sélections :

```
A = navire et bateau (liste)  (270 Noms)
B = hydrocarbure (thème)      (350 Mots)
```

produit la liste :

```
asphaltier                    n.m
bitumier                      n.m.
butanier                      n.m.
méthanier                     n.m.
pétrolier                     n.m.
propanier                     n.m.
supertanker                   n.m.
tanker                        n.m.
```

Le Dicologique ajoute à cet accès original d'autres petits outils tels que la recherche par une rime ou avec des caractères inconnus (pour les mots croisés). Notons aussi un accès par anagrammes.

3. Systèmes lexicaux spécialisés

Après l'étude des dictionnaires sur papier et des dictionnaires classiques informatisés, nous nous intéressons aux dictionnaires créés pour être utilisés par une machine. Ces dictionnaires comportent des informations hautement structurées nécessaires à certains traitements. Ils ont été créés de toutes pièces pour des traitements linguistiques particuliers sous une forme dictée par des problèmes informatiques (temps d'accès, encombrement…).

L'étude de l'histoire des lexiques dans le domaine du traitement des langues nous permet de dégager les tendances suivantes :

- Les dictionnaires étaient considérés uniquement comme la composante lexicale d'une application particulière, l'intérêt des chercheurs se portant plutôt sur les structures de données et les algorithmes. Peu de travaux ont été développés pour simplifier la création et la maintenance de ces composantes lexicales.
- Au fur et à mesure de la montée en puissance des systèmes de traitement des langues, la composante lexicale a présenté des problèmes de plus en plus importants. Le coût de développement de cette composante peut représenter plus de la moitié du coût d'un système. De plus, l'importance des dictionnaires rend difficile toute modification de leur structure (obligation de modifier toutes les entrées existantes).
- On essaye donc de dissocier la structure des informations lexicales de la structure utilisée par les algorithmes de traitement. On crée ainsi des dictionnaires indépendants (et des systèmes pour les gérer) qui servent de sources pour générer des dictionnaires d'applications. Pourtant, ces dictionnaires restent spécialisés pour un type de traitement donné. Nous appellerons donc ces dictionnaires (et leur systèmes de gestion) des *systèmes spécialisés*.

Les systèmes spécialisés sont des dictionnaires électroniques développés pour servir de ressources de connaissances lexicales pour certaines applications particulières. Certains ne sont utilisées que par une application. D'autres sont utilisés par plusieurs applications du même type.

Dans cette partie, nous étudierons d'abord des bases lexicales monolingues dont l'intérêt réside dans le type d'information et dans leur structure. Nous nous intéresserons ensuite à des systèmes qui illustrent différentes étapes vers la création d'une base lexicale multi-usages.

3.1. BDLex

BDLex est une base de données lexicales du français développée par l'IRIT (Institut de Recherche en Informatique de Toulouse). Cette base lexicale est monolingue. Elle est utilisée par de nombreux partenaires du PRC "Communication Homme Machine".

BDLex contient environ 25 000 lemmes (correspondant à environ 300 000 formes fléchies).

Les informations contenues dans BDLex sont morphologiques et phonologiques. L'unité lexicale de cette base est le lemme.

lemme	HG	PHON	FPH	HP	CL_PHON	NS	F	CS	GN	CF
nabab	11	/nA/bAb		11	/NA/DAD	2		N	Mn	01
nabi	11	/nA/bi		11	/NA/DI	2		N	Mn	01
nabot	11	/nA/bo	t"	11	/NA/DE	2		N	gn	01
nacelle	11	/nA/s&l	e	11	/NA/SEL	2		N	Fn	81

Figure 1.11 : Un exemple d'entrées de BDLex ; (HG : numéro d'homographe, PHON : phonétique, FPH : terminaison phonétique, HP : numéro d'homophone, CL_PHON : classe phonétique, NS : nombre de syllabes, F : fréquence, CS : classe syntaxique, GN : variation en genre et nombre et CF : classe flexionnelle)

Ce dictionnaire est typiquement à usage informatique. Les informations telles que le numéro d'homographe ou d'homophone sont particulièrement utiles pour gérer les problèmes d'ambiguïté lexicale, qui sont cruciaux lors du traitement automatique d'une langue.

De plus, les informations de ce dictionnaire sont codées et difficilement utilisables par un humain. On notera néanmoins que ce codage est relativement simple (structure en colonne, codes mnémotechniques), ce qui explique son succès auprès de nombreux laboratoires.

3.2. Dictionnaires du LADL

Le LADL (à l'Université Paris VII) a développé le dictionnaire DELAF, un dictionnaire contenant 600 000 formes fléchies du français. Ce laboratoire a aussi développé le dictionnaire DELACF, contenant 150 000 formes fléchies de mots composés français.

Outre leur taille (qui les place parmi les plus importants dictionnaires français), le DELAF et le DELACF présentent un aspect remarquable : leurs entrées sont représentées par des automates d'états finis. Cette structure d'automate est utilisée à différents niveaux. Appliquée au codage des informations morphologiques et syntaxiques d'une forme fléchie, elle permet de représenter les différentes ambiguïtés des formes fléchies.

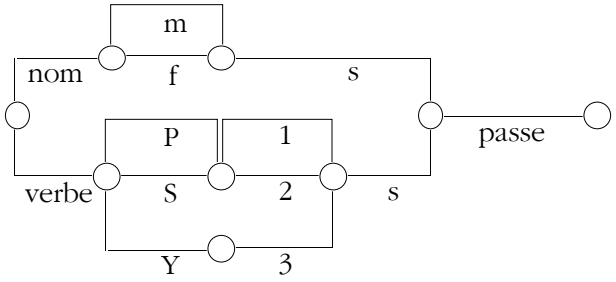


Figure 1.12 : Automate représentant la forme fléchie "passe". Les abréviations utilisées sont : m pour masculin, f pour féminin, s pour singulier, 1,2 et 3 pour dénoter la personne, P pour présent, S pour subjonctif, Y pour impératif.

Enfin, elle est utilisée pour représenter les différentes variantes d'un mot composé.

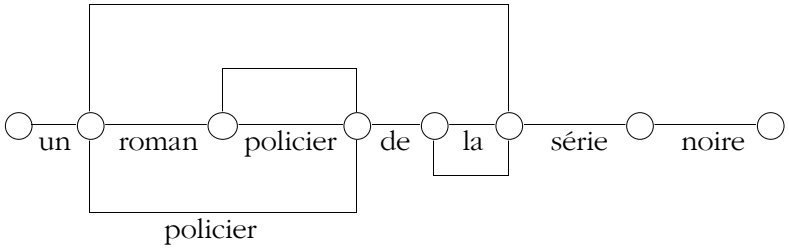


Figure 1.13 : Automate représentant les différentes variantes du mot composé "un roman policier de la série noire" qui peut se trouver sous les formes "un série noire", "un policier de série noire"…

3.3. Ariane

Le laboratoire GETA (Groupe d'Étude pour la Traduction Automatique) a développé un générateur d'applications de traduction automatique nommé ARIANE. En développant des applications de traduction, les chercheurs du GETA se sont vite rendu compte que le

problème de la gestion des dictionnaires électroniques était un problème crucial. Aussi, des outils ont été développés pour faciliter cette gestion.

Chaque application de traduction automatique développée sous ARIANE a son propre dictionnaire, éclaté en divers composants (pour l'analyse morphologique ou syntaxique, le transfert lexical ou structural, la génération syntaxique ou morphologique…). Chacun de ces composants est exprimé dans un format spécialisé du langage en charge de la phase de traitement en question.

L'éclatement du dictionnaire introduit des problèmes de cohérence des informations lexicales réparties dans les divers composants. Aussi, un outil (Visulex) a été développé pour réunir et visualiser les différentes informations lexicales d'une application de traduction automatique. Cet outil ne permet de manipuler qu'un couple de langues à la fois.

Dans une telle base, l'unité lexicale est une famille de lemmes liés par certaines fonctions lexico-sémantiques. Par exemple, les lemmes "construction", "constructif", "construire", "reconstruire" appartiennent tous à une même unité lexicale notée "construire-V".

Visulex génère deux niveaux de fichiers pour le linguiste. Le premier niveau donne l'ensemble des commentaires associés à une unité lexicale. Le second niveau donne le détail des informations linguistiques associées à une unité lexicale, sous la forme où on les trouve dans les dictionnaires d'applications. Commentaires et informations détaillées sont donnés de manière structurée. Un article de Visulex contient notamment un contexte, une morphologie, des détails sur les éléments du contexte et une liste d'équivalents (avec des conditions de choix).

```
------------------------------------------------------------
'GEHEN'
-------
                    --contextes--
   'GEHEN'
      --arbre: X(0,P1(1(V2)),P2(2),P3(3),P4(4))
      0:'GEHEN'
         --cmt?--
                    --morphologie--
   --cmt?--
      FORME COMPLETE DE PARTICIPE PASSE FLECHISSABLE
      GEGANGEN-
      PARADIGME 09 INCLUS DANS 01(LEG) SAUF POUR -T QUI NE ...
      PEUT FAIRE LE PARTICIPE PASSE
      GEH-
      PARADIGME 29, DESINENCES -0, -E, -EN, -EST, -ET, -ST, -T
      GING-
                    --expansions--
--arbre: X(0,P1(1(V2)),P2(2),P3(3),P4(4))
X:'VBPSP'
   --cmt?--
0:'GEHEN'
   --cmt?--
…
                    --equivalents--
                    ---------------
'ALLER'
   'ALLER'
      VERBE SANS AUCUNE RECTION DONNEE, EX: 'MARCHER'
      --cmt?--
      --si: --cmt?--

         '''AILL'
            BASE ACCEPTANT LES DESINENCES DE L'IMPARFAIT,DU...
             PRESENT SUB
      --si: --cmt?--
         '''IR'
            --cmt?--
      --si: --cmt?--
         'VAIS'
      --si: --cmt?--
         'VAS'
      --si: --cmt?--

         'VA'
      --si: --cmt?--
         'VONT'
      --si: --cmt?--
         '''ALLER'
            TABLE+S
      --sinon:
         '''ALL'
            COUP+ER,E,EUR,ANT
------------------------------------------------------------
```

```
------------------------------------------------------------
'GEHEN'
-------
                    --contextes--
   'GEHEN'
      --arbre: X(0,P1(1(V2)),P2(2),P3(3),P4(4))
      0:'GEHEN'
         IST:SUBV:=SEIN
                    --morphologie--
   VIDE:
      FCPPA:KMS-E-VB-U-ADJ,MT-E-PPA,SUBADJ-E-RSTA
      GEGANGEN-
      WSING:KMS-E-VB

      GEH-
      WFIEL:KMS-E-VB
      GING-
                    --expansions--
--arbre: X(0,P1(1(V2)),P2(2),P3(3),P4(4))
X:'VBPSP'
   VID:
0:'GEHEN'
   IST:SUBV:=SEIN
…
                    --equivalents--
                    ---------------
'ALLER'
   'ALLER'
      VB:
      R3:AX:=ETR,RFRUS:=SUPPR
      --si: SUBJ3:MOD-E-SUB-ET-TF-E-PRE-ET-NUM-E-SIN-OU-MOD...
            -E-SUB-ET-TF-E-PRE-ET-P-E-3
         '''AILL'
            VERBE:PGMV-E-VERBE

      --si: FUT:TF-E-FUT-OU-MOD-E-CDL
         '''IR'
            AUR:PGMV-E-FUTUR
      --si: SUIS:TF-E-PRE-ET-MOD-E-IND-ET-NUM-E-SIN-ET-P-E-1
         'VAIS'
      --si: DUEPRE:P-E-2-ET-TF-E-PRE-ET-NUM-E-SIN
         'VAS'
      --si: SAI:NUM-E-SIN-ET-TF-E-PRE-ET-MOD-E-IND-OU-MOD-E...
            -IMP-ET-NUM-E-SIN
         'VA'
      --si: SONT:TF-E-PRE-ET-MOD-E-IND-ET-NUM-E-PLU-ET-P-E-3
         'VONT'
      --si: NVBMAS:CPRD-E-VBACT-ET-CAT-E-CATN-ET-GNR-E-MAS
         '''ALLER'
            MOT:FLXN-E-MOT
      --sinon:
         '''ALL'
            V1AFE1:FLXV-E-AIMER,DRNV-E-FEME1
------------------------------------------------------------
```

Figure 1.14 : Exemple d'article généré par Visulex

Visulex rassemble des informations dispersées dans les nombreux dictionnaires des différents composants d'une application linguistique. Il est ainsi possible de consulter les informations liées à une unité lexicale pour retrouver la source d'éventuelles erreurs de traduction. Par

contre, la modification des dictionnaires ne peut se faire qu'au travers d'ARIANE, composant par composant. Néanmoins, l'outil Visulex est un premier pas vers la création d'une base lexicale spécialisée pour la Traduction Automatique.

3.4. BDTAO

B'VITAL utilise le système ARIANE pour construire des applications de traduction automatique industrielles. Pour cela, cette entreprise a développé une base lexicale réutilisable (BDTAO) à partir de laquelle il est possible de générer des composants du dictionnaire de l'application, au format ARIANE.

Cette base lexicale est spécialisée pour la traduction automatique, mais elle est indépendante d'une application de traduction particulière. Une même base peut être utilisée pour l'analyse et la génération. De plus, les entrées terminologiques sont réversibles en transfert.

BDTAO contient des dictionnaires "en fourche" (1 langue source -> n langues cibles).

Comme dans les bases Visulex, l'unité lexicale est une famille de lemmes liés par certaines fonctions lexico-sémantiques. Les informations linguistiques sont codées dans une structure "attribut-valeur" plate. Cette structure est séparée en deux sections. La première contient l'information monolingue, la seconde contient différentes traductions de l'entrée dans différentes langues, avec éventuellement des conditions guidant le choix.

```
*ADJECTIF
001 prochain
002 COU
101 S
103 ES
105 E
107 O
504 SXA
990  - prochainement = bientôt, dans un
990    proche avenir
99D 22/03/88
99A IM
$CODE
$UL PROCHAIN -A
$PH AM 1 29/09/88
$AM PROCHAIN  ADJ1 Z000154
```

Figure 1.15 : Une entrée lexicale de BDTAO

BDTAO est une véritable base lexicale spécialisée. En effet, toute modification ou création d'information lexicale se fera dans ce format. On peut ensuite générer des dictionnaires spécialisés pour chacun des composants d'une application linguistique sous ARIANE.

Un tel outil simplifie grandement la gestion des dictionnaires d'application puisque la consultation et la modification se font à un même endroit et dans un format simple. Les noms des attributs ne dépendent pas des noms des variables utilisées dans les grammaires des applications lexicales envisagées. Cette indépendance de BDTAO vis-à-vis d'une application particulière permet d'utiliser une base lexicale pour plusieurs applications différentes.

L'indexage de cette base lexicale est fait en utilisant des bordereaux d'indexage où le lexicographe répond à des questions simples sur l'entrée en question. De plus, B'VITAL a construit un outil permettant de récupérer les informations lexicales des applications de traduction développées antérieurement à BDTAO.

3.5. METAL

Pour son générateur de systèmes de Traduction Automatique (METAL 3.0), SIEMENS a développé des outils pour faciliter la manipulation des dictionnaires. Il y a deux types de dictionnaires sous METAL 3.0 : les dictionnaires monolingues et les dictionnaires de transfert.

Les structures des entrées de ces dictionnaires sont des structures de traits plates. Les traits des dictionnaires de transfert sont fixés pour chaque paire de langues. Les traits des dictionnaires monolingues sont définis pour chaque langue par un fichier de description.

Une entrée de dictionnaire METAL est un ensemble de traits avec leurs valeurs. Les valeurs de ces traits peuvent être de l'un des types suivants :

INTEGER	tout nombre entier entre -228+1 et 228-1.
STRING	une séquence de caractères entre guillemets, ex : “lexique”.
SYMBOL	un symbol Lisp
BOOLEAN	soit T, soit NIL
SET	une séquence d’INTEGERs, de STRINGs et/ou de SYMBOLs, non ordonnés et sans duplication.
LIST	une séquence ordonnée d’INTEGERs, de STRINGs, de SYMBOLs, et/ou de LISTs.

L’utilisateur peut définir ses propres structures linguistiques via un fichier de description. Ce fichier contient d’une part la description des traits (avec le type de ses valeurs) et d’autre part la définition des catégories lexicales. Une catégorie lexicale est une structure complexe regroupant plusieurs traits. La syntaxe de ces descriptions est une syntaxe LISP. Nous donnons en exemple les définitions d’un trait et d’une catégorie lexicale.

La macro `defeat` définit un trait nommé `a-cl` qui accepte une valeur de type ensemble sur 6 symboles de base. On n’autorise pas plus de 7 combinaisons sur l’ensemble du dictionnaire.

```
(defeat a-cl :set
             :values (FMNP1 FMNP2 FMNP3 FMNPS1 FMNPS2 FMNPS3)
             :allocate 7
             :pretty-name “Anaphoric class”)
```

La macro `defcat` définit une catégorie lexicale nommée `det` qui contient un trait obligatoire (`ca`) et 3 traits optionnels (`a-cl`, `abb` et `plc`). La valeur par défaut du trait `plc` est égale au symbole `NF`.

```
(defcat det :optional (A-CL FMNP1 FMNP2 FMNPS3)
            :optional ABB
            :required CA
            :optional PLC
            :default (PLC NF)
            :check-consistency t
            :pretty-name “Determiner”)
```

Ce langage de définition est d’un usage assez simple et il est effectivement utilisé par des linguistes. Par contre, on peut regretter que la définition d’un trait fasse intervenir des considérations linguistiques (contraintes sur le type du trait) et des considérations informatiques (espace alloué pour le stockage des informations).

Nous avons ici un premier pas vers la généricité dans les bases lexicales. Les structures linguistiques ne sont pas figées, mais peuvent être redéfinies par des linguistes. Par contre, la structure logique utilisée dans cette base lexicale est fixe : il s’agit de structures de traits plates utilisées par le système METAL.

II. Efforts en cours

Les systèmes spécialisés permettent un gestion assez aisée des données lexicales de différentes applications, mais sont peu adaptables à d'autres types d'application. En effet, non seulement leur formalisme est étroitement lié aux applications pour lesquelles ils ont été développés, mais aussi les types d'information linguistique qu'ils contiennent sont limités à ces applications. Par exemple, une base lexicale spécialisée à la TAO ne contient ni prononciation, ni éthymologie, ni exemples d'usage.

Ces systèmes présentent un avantage économique "local". En effet, ils permettent de réduire les coûts de développement des dictionnaires d'un type d'application particulier. Ils représentent un ensemble de sources assez bien organisés pour permettre une utilisation par des laboratoires ou groupe différents.

Ces sources ne sont hélas pas dans un format standardisé, ni facilement récupérables. La réutilisabilité de ces bases est donc un problème réel. Pour y remédier, on peut envisager différentes solutions :

- mettre au point un standard de codage des informations linguistiques,
- construire de nouvelles bases lexicales assez complètes pour être indépendante d'une application particulière et assez générales pour être indépendantes d'ue théorie linguistique particulière,
- regrouper des ressources lexicales en un même lieu, sans s'occuper de leur forme.

Cette dernière solution a été prise par le CLR (Consortium for Lexical Research) qui propose un serveur FTP (File Transfer Protocol) pour centraliser toutes les données disponibles. Une telle initiative ne règle aucun problème technique, mais facilite singulièrement la recherche de données lexicales, qui tenait, auparavant, du marathon.

De nombreux chercheurs se sont posé le problème de la construction de bases lexicales multi-applications.

Parmi les projets qui en sont résulté, quatre sont particulièrement intéressants, et nous les examinerons en détail. Il s'agit de Le Lexicaliste, EDR, GENELEX et MULTILEX.

Ces projets diffèrent en de nombreux aspects, comme leurs motivations, leurs choix technologiques, leurs moyens, etc., mais chacun est remarquable à plusieurs titres, et l'ensemble offre une vision assez juste des efforts actuellement en cours sur le domaine des bases lexicales.

1. Le Lexicaliste

Le Lexicaliste est un système de gestion de bases lexicales monolingues développé et commercialisé par la société SITE.

1.1. Vue générale du système

Le Lexicaliste s'appuie sur une description des entrées du lexique. Un article est un arbre décoré dont la racine correspond à l'entrée du dictionnaire (lemme) et les nœuds aux différents sens de l'article. Les décorations sont des structures attributs-valeurs simples portées par les différents sens de l'article.

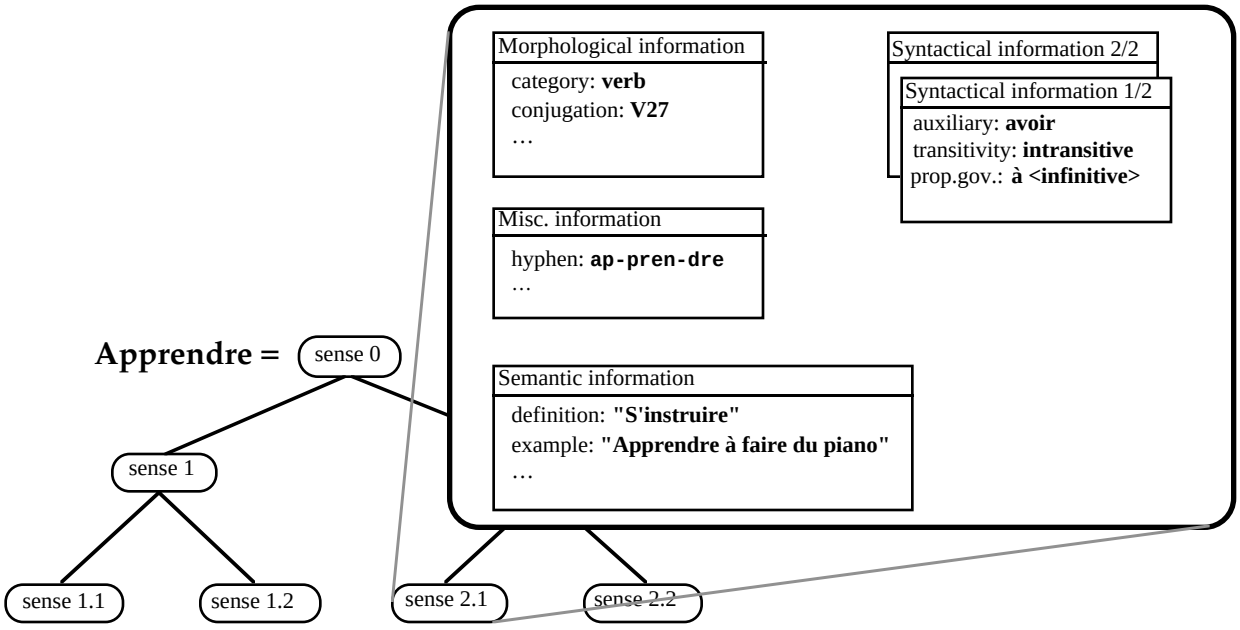


Figure 2.1 : Une entrée de dictionnaire créée par Le Lexicaliste

Le linguiste définit les attributs (et les valeurs) qui sont utilisés dans une base lexicale particulière. Cette description est appelée "référentiel". Il peut aussi donner des propriétés (attributs monovalués, multivalués, relations acycliques…) sur les attributs de la base. Ces propriétés sont contenues dans le "méta-référentiel".

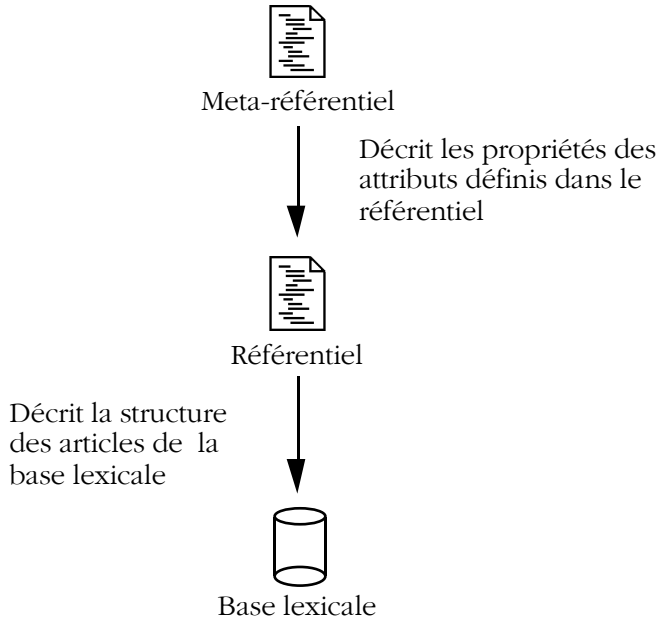


Figure 2.2 : Éléments de la définition d'un dictionnaire

Les attributs sont séparés en 5 catégories distinctes:

- attributs des lemmes (ex : `catégorie`),
- attributs des sens (ex : `transitivité, définition`),
- attributs des règles flexionnelles (ex : `nombre, genre`),
- relations lexicales (ex : `abréviation, dérivation`),
- relations sémantiques (ex : `hyperonymie, synonymie`).

Lorsque ce travail de définition a été accompli, les tables SQL et l'interface sont automatiquement générées par le système.

1.2. Réseau lexical et sémantique

Le système gère deux types de relations, qui définissent deux réseaux différents à l'intérieur du dictionnaire :

- les relations lexicales, définies sur un ensemble de sens de mots,
- les relations sémantiques, définies sur un ensemble de concepts.

Une relation lexicale relie deux sens de mots au niveau lexical. Par exemple :

- "appt" `est-abreviation-de` "appartement",
- "clef" `est-variante-orthographique-de` "clé"
- "apprentissage" `est-nominalisation-de` "apprendre".

Les relations lexicales relient les différents sens des lemmes, et non différents lemmes. Cela permet par exemple de relier les deux sens différents de "blanchir" à leur nominalisation "blanchissage" et "blanchiment"). On aura donc :

- "blanchissage" `est-nominalisation-de` "blanchir-1",
- "blanchiment" `est-nominalisation-de` "blanchir-2".

Une relation sémantique relie deux concepts au niveau sémantique (au sens de la référence au monde réel) :

- "chaise" `est-un` "meuble",
- "poisson" `a-connotation-avec` "mer",
- "malaria" `est-synonyme-de` "paludisme".

Chaque sens de mot peut être associé à un concept par un attribut particulier : le prédicat sémantique. Réciproquement, chaque concept peut être associé à un ou plusieurs sens de mot.

1.3. Vérifications de cohérence et valeurs par défaut

Pour simplifier la maintenance et l'indexage d'un dictionnaire, le linguiste dispose d'un langage spécialisé avec lequel il peut définir des contraintes sur certains des attributs d'un article.

Ces contraintes sont utilisées de deux manières pour :

- Vérifier la cohérence d'un article (déjà indexé dans le dictionnaire),
- Attribuer des valeurs par défaut à certains attributs d'un article en cours de création.

Avec ce langage, le linguiste peut par exemple exprimer les contraintes suivantes :

- un verbe pronominal prend l'auxiliaire "être",
- un verbe impersonnel n'a pas de forme passive,
- l'attribut `transitivité` prend la valeur `directe` pour les verbes se terminant en "iser" (cette valeur par défaut sera proposée),
- l'attribut `conjugaison` d'un verbe se terminant par "ger" prend la valeur `V1` ou `V2`.

Voici un exemple de ces contraintes :

```
DECL-MESSAGES

msg-aux "l'attribut auxiliaire n'est pas défini"
msg-transit "l'attribut transitif n'est pas défini"
```

```
msg-transObj2 "l'attribut transObj2 doit être défini"
msg-frmPassif "l'attribut frmPassif doit être défini"

DECL-CONTRAINTES
// Syntactic attributes for verbs
SI cat = verb ALORS
 // the default auxilary is “avoir” (to have)
 aux DEFINI DEFAUT {avoir} MESSAGE msg-aux

 // if the verb ends with “ter”, the conjugation is
 // V3 or V3H or V3Q (default V3).
 si cle = "*ter" alors
       mm dans { V3, V3H, V3Q } defaut V3
 FSI
 // An intransitive verb does not admit passive (by default)
 SI transit = intrans ALORS
       passiv DEFAUT non
 FSI
FSI
```

2. EDR

Le plus grand projet mondial de construction d'une base lexicale multilingue a débuté à Tokyo le 26 avril 1986. D'une durée de 9 ans, ce projet a impliqué 1200 hommes-années pour un coût total de 14 milliard de Yens (environ 750 MF).

Au terme de son contrat, EDR a atteint ses objectifs : la construction d'une base lexicale anglais-japonais utilisable par des systèmes de traduction automatique. Les dictionnaires anglais et japonais comportent 300 000 entrées chacun (200 000 en vocabulaire général et 100 000 en vocabulaire terminologique). EDR a aussi décrit 640 000 concepts correspondant à ces entrées. La base lexicale comporte enfin deux dictionnaires bilingues (1 par sens) de 300 000 entrées et deux corpus (anglais et japonais) de 250 000 phrases analysée. Ces corpus comportent aussi 20 millions de phrase Japonaises et 10 millions de phrases anglaises.

EDR est avant tout un projet industriel visant la construction d'une base lexicale de *grande taille*. Cet objectif n'a pu être atteint qu'au prix d'une simplification des structures linguistiques présentes dans les dictionnaires. Néanmoins, ces dictionnaires sont suffisamment complets pour être utilisés comme une source de données lexicales par de nombreux systèmes de traduction développés au Japon.

2.1. Architecture lexicale

Le projet EDR a délibérément adopté une approche mixte où l'on trouve un dictionnaire interlingue et des dictionnaires bilingues.

Les dictionnaires de mots (anglais et japonais) sont divisés en deux parties. La première contient environ 200 000 termes généraux, et la seconde contient environ 100 000 termes techniques (dans le domaine du traitement de l'information). Ces dictionnaires contiennent les informations grammaticales sur les entrées et les concepts auxquels elles sont associées.

Le dictionnaire de concepts regroupe 640 000 concepts, parmi lesquels, 100 000 proviennent des entrées terminologiques et sont communs aux deux langues. Par ailleurs, 60 000 concepts sur environ 300 000 concepts issus de chaque dictionnaire général sont communs aux deux langues (Ch. Boitet, communication personnelle).

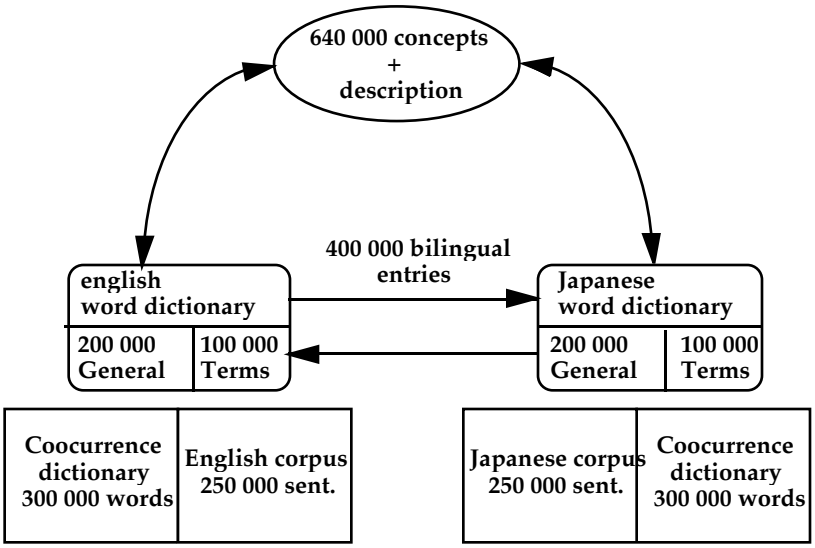


Figure 2.3 : Architecture lexicale du projet EDR

Les informations sur ces concepts sont séparées en deux parties : la classification des concepts et leur description.

2.2. Architecture linguistique

Le souci principal du projet EDR est la taille des dictionnaires. Aussi, ce projet a adopté une architecture lexicale assez simple. Les informations grammaticales contenues dans les dictionnaires s'expriment selon un ensemble fermé d'attributs. L'absence d'une véritable morphologie rend cette structure difficilement adaptable à des langues plus flexionnelles que l'anglais ou le japonais, comme le français ou l'allemand.

2.2.1. Dictionnaires monolingues

Les articles des dictionnaires monolingues EDR se composent de quatre parties (cf. figure 2.4.)

Entrée	**Informations grammaticales**	**Informations sémantiques**	**Informations supplémentaires**
Entrée Constituants Forme normale Attributs d'adjacence Division en syllabes Prononciation	Catégorie Arbre syntaxique Flexion Attributs grammaticaux Fonction (prép, articles…)	Identificateur de concept Illustration du concept	Usage Fréquence

Figure 2.4 : Structure d'une entrée de dictionnaire monolingue EDR

L'entrée du dictionnaire correspond à une occurrence. Lorsque cette entrée est complexe, elle est notée sous forme d'une liste de constituants, dont on connaît la forme normale et des contraintes sur leur forme dans l'entrée complexe. On dispose aussi de la division de l'entrée en syllabes et de la prononciation.

Word	**Headword**	
	Retrieval Entry	**Constituent Information**
study	stud(ELV1,ECV5)	
eat	eat(ELV2,ECV7)	
ate	ate(ELV2,ERV3)	
give up	give(ELV1,ECV9)	/w#suf(*,*)/(ELB1,ERB1) /up(ELW1,ERD5)

Figure 2.5 : Exemple d'entrée de dictionnaire monolingue EDR

Les informations grammaticales regroupent la catégorie de l'entrée, sa catégorie flexionnelle, un ensemble d'attributs grammaticaux et, pour les mots outils (prépositions, articles…), la fonction. Si l'entrée est complexe, un arbre syntaxique en détaille la structure.

Headword	**Part of Speech**	**Syntactic Tree**
mak/w#suf/ /*/ /use/ /of/ /*	EVP	EVP(EVE(EVE(mak)/EEV(W#suf))/ESY()/ ENP(ENP([EAP(*)]/EN1(use))/ESY()/ EPP(EPR(of)/ESY()/{ENP(*)}))
mak/w#suf/ /up/ /w#one's/ /mind	EVP	EVP(EVE(EVE(mak)/EEV(w#suf))/ESY()/ED3(up)/ ESY()/ENP(EP1(w#one's)/ESY()/EN1(mind)))

Figure 2.6 : Un exemple d'information grammaticale associée à une entrée

Les informations sémantiques regroupent un identificateur de concept, qui renvoie au dictionnaire de concept, et une explication en anglais et japonais permettant à un utilisateur d'identifier le concept dont il est question.

HeadWord	**Concept Illustration**	
	Primary Illustration	**Concept Explanation**
plane(ELN1,ECN1)	airplane(vehicle)	A vehicle called airplane
plane(ELN1,ECN1)	plane(tool)	A carpentry tool called plane

Figure 2.7 : Exemple d'information sémantique associée à une entrée

Enfin, un article du dictionnaire monolingue contient aussi des informations sur l'usage et la fréquence de l'entrée.

2.2.2. Dictionnaire bilingue

Les articles de dictionnaire bilingue comportent deux parties principales : l'entrée en langue source et les informations sur ses équivalents en langue cible.

Entrées	**Correspondances**
Entrée Notations supplémentaires	Correspondant Relation de correspondance Explications supplémentaires

Figure 2.8 : Structure d'un article de dictionnaire bilingue EDR

L'entrée est une chaîne de caractères (avec des espaces si elle est composée) donnant la forme canonique (infinitif des verbes…). Dans le dictionnaire japonais-anglais, les notations supplémentaires contiennent la prononciation de l'entrée (en katakana). Dans les deux dictionnaires bilingues, on trouve aussi une information sur les parties variables de l'entrée. Un point est placé après la partie de l'entrée qui n'est pas modifiée par les flexions.

Entrées	**Notations supplémentaire**
見る 美しい 私	ミ . ル ウツクシ . イ ワタシ
study name write wrote	stud.y name writ.e wrote

Figure 2.9 : Exemple d'entrées de dictionnaire bilingue

Le correspondant en langue cible d'une entrée en langue source est une liste d'entrées du dictionnaire de langue cible, à moins qu'une flexion particulière ne soit nécessaire lors de la traduction.

Les équivalences entre langues n'étant généralement pas parfaites (nuances entre les concepts recouverts par un mot et sa traduction…), on donne la relation de correspondance qui existe entre l'entrée et son équivalent. Il existe 5 relations de correspondance :

- équivalence : les concepts recouverts par l'entrée et sa traduction sont équivalents,
- sous-relation : le concept de l'entrée est plus large que le concept de sa traduction,
- super-relation : le concept de l'entrée est un sous-concept de celui de la traduction,
- synonymie : la différence entre les concepts est minime,
- remarque : le correspondant est une transcription de l'entrée. Dans ce cas, une explication est donnée en information supplémentaire.

Entrées	**Relations de correspondance**	**Correspondances**
犬	équivalence	dog
left	équivalence	左
annihilation	sous-relation	＜素粒子＞　消滅
迂遠	sous-relation	<explanation> circuitous
allege	super-relation	（証拠なしに）　断言する
探言	super-relation	(in) other words
abbey	synonymie	＜教会＞（以前修道院であつた）大教会
ameer	remarque	アミール［トルコ高官の称号］
烏帽子篭	remarque	ebosikago [bamboo hand-basket] ?

Figure 2.10 : Un exemple d'entrées de dictionnaire bilingues

L'explication supplémentaire est indiquée entre chevrons(<>), parenthèses (()) ou crochets ([]) suivant que l'on a affaire (respectivement) à une sous-relation, une super-relation ou une remarque.

2.3. Dictionnaire de concepts

Dans les trois phrases suivantes :

- Un éléphant apparaît,
- Un éléphant n'oublie jamais,
- L'éléphant est une espèce en danger,

le mot *éléphant* renvoie tour à tour à un individu de l'espèce des éléphants, à un prototype d'éléphant ou à l'espèce des éléphants. Néanmoins, ce mot réfère à *quelque chose* de commun à chacun de ces usages. Ce *quelque chose* est le contenu sémantique du mot *éléphant*. La même remarque s'applique au mot japonais 象 *zô*., qui a le même contenu sémantique que le mot *éléphant*.

Le contenu sémantique d'un mot, ainsi défini, est représenté par un élément du dictionnaire de concepts (un concept). Un concept (correspondant à un mot) a une identificateur unique. Ainsi, le concept correspondant à *éléphant* a l'identificateur <3bf246>. Néanmoins, comme il n'y a pas de possibilités de confusion (polysémie…), ce concept est aussi appelé "concept éléphant" et noté «éléphant».

Dans le dictionnaire de concepts, l'identificateur est accompagné d'une illustration (une phrase en anglais et en japonais) qui permet à l'utilisateur de connaître le concept qu'il manipule.

Le plus important travail de recherche, d'indexage et de maintenance a été effectué sur le dictionnaire de concepts. Ce dictionnaire est décomposé en deux parties. Dans la première partie, les concepts sont décrits par un ensemble de relations qu'ils entretiennent entre eux. Dans la seconde partie, les concepts sont classifiés hiérarchiquement.

2.3.1. Description des concepts

EDR a rejeté l'approche consistant à décrire un concept à l'aide de concepts de base plus simples. Aussi, chaque unité sémantique, correspondant à un mot ou une expression complexe, s'exprime par un concept (que l'on peut identifier et manipuler), et ce, même si la sémantique de ce concept peut se traduire par une description à base de concepts plus simples.

La description des concepts se base sur un ensemble de relations qu'ils entretiennent entre eux. EDR a retenu 24 relations et 50 attributs (relations unaires) pour cette description :

agent sujet d'une action volontaire. Les entités conscientes ou automatisées peuvent être de tels sujets.
"un animal mange"
«manger» — agent → «animal»

a-object attribut d'un objet
"les tomates sont rouges"
«rouge» — a-object → «tomate»

object objet affecté par une action ou un changement
"manger de la viande"
«manger» — object → «viande»

cause la cause
"mort de froid"
«mort» — cause → «froid»

implement instrument ou moyen dans une action volontaire
"couper avec un couteau"
«couper» — implement → «couteau»

material composant matériel ou structurel
"fait avec du lait"
«faire» — material → «lait»

source sujet d'un événement ou position ou condition initial d'un objet
"venir de Tokyo"
«venir» — source → «Tokyo»

goal sujet d'un événement ou position ou condition finale d'un objet
"aller à Tokyo"
«aller» — goal → «Tokyo»

place lieu où se tient un événement
"jouer dans la chambre"
«jouer» — place → «chambre»

scene contexte dans lequel un événement a lieu
"jouer dans un drame"
«jouer» — scene → «drame»

manner manière dont se passe une action ou un changement
"parler lentement"
«parler» — manner → «lentement»

time instant où se passe un événement
"se lever à l'heure"
«se lever» — time → «à l'heure»

time-from instant où débute un événement
"je travaille depuis le matin"
«travailler» — time-from → «matin»

time-to instant où se termine un événement
"je travaille jusqu'au soir"
«travailler» — time-to → «soir»

quantity quantité de chose, d'action ou de changement
"un kilo de pommes"
«pommes» — quantity → «kilo»

number nombre
"3 kilos"
«kilo» — number → «3»

condition relation de condition entre événements ou circonstances
"aller pleurer"
«aller» — condition → «pleurer»

cooccurrence relation simultanée entre événements ou circonstances
"partir en pleurant"
«partir» — cooccurrence → «pleurer»

purpose but d'une action
"aller voir"
«aller» — purpose → «voir»

sequence relation temporelle séquentielle entre événements ou circonstances
"sauter et frapper"
«sauter» — sequence → «frapper»

basis base de comparaison
"les roses plus belles que les lilas"
«roses» ← a-object — «beau»
← object — «plus» — basis → «beau»
—a-object → «lila»

and relation de conjonction entre concepts
"visiter Rome et Venise"
«visiter» — goal →
(«Rome» — and → «Venise»)

or	relation de disjonction entre concepts “visiter Rome ou Venise” «visiter» — goal → («Rome» — or → «Venise»)
modifier	autres relations

À ces 24 relations s’ajoutent 4 “pseudo-relations”, qui sont des relations définies en fonction des autres relations. Ainsi, la pseudo-relation *possessor* est définie comme suit :

possessor — possessor → ≡ ← object — «own» — agent →

Les pseudo-relations sont :

possessor	relation de possession “le chien de Tom” «chien» — possessor → «Tom»
beneficiary	bénéficiaire d’un événement ou d’une circonstance “utile aux personnes” «utile» — beneficiary → «personnes»
from-to	portée, chemin, … “un tiquet d’Osaka à Tokyo” «tiquet» — modifier → («Osaka» — from-to → «Tokyo»)
unit	l’unité “500 yens pour une douzaine” («1» ← number — «douzaine») ← unit — (« yens» — number → «500»)

Les attributs sont définis comme des relations unaires sur les concepts. EDR a défini 50 attributs répartis comme suit :

Attributs qualifiant l’objet :

all	Tous les objets
some	Un nombre non spécifié d’objets
specific	Des objets spécifiés
generic	Objets avec des caractéristiques générales

Attributs indicateur de temps :

past	Le point de vue est dans le passé
present	Le point de vue est dans le présent
future	Le point de vue est dans l’avenir

Attributs de relativité :

begin	Début d’une action ou d’un événement
end	Fin d’une action ou d’un événement
progress	Une action ou un événement est en cours
continue	Une action répétitive ou un événement répétitif est en cours
state	Une action ou un événement est terminé et un état ou résultat est atteint
complete	Toutes les actions ou tous les événements sont terminés
yet	N’a pas encore eu lieu
already	A déjà eu lieu
soon	Aura lieu bientôt
just	Vient d’avoir lieu
come	On approche du moment auquel pense le locuteur
go	On s’éloigne du moment auquel pense le locuteur

Attributs dénotant l’intention du locuteur :

imperative	Un ordre
interrogation	Une question
exclamation	Une exclamation
invite	Une invitation
rumor	Une rumeur
respect	Avec respect
polite	Avec politesse
require-agreement	Demande agrément ou confirmation
thought	Une pensée
conclude	Une conclusion
sure	Inférence à partir d’une situation
maybe	Inférence d’une éventualité
seem	Inférence ou supposition
recommend	Une recommandation
grant	Une permission
grant-not	Un refus de permission
underestimate	Une sous-estimation
duty	Une obligation
should	Une quasi-obligation
sufficiency	Suffisance
consent	Un consentement
pity	Désappointé
be-sorry	Avec remords
appearance	Circonstance ou comparaison
natural-result	Résultat naturel d’un événement ou d’une circonstance
advise	Avis ou recommandation donnée par l’auteur
natural-thing	Forme idéale, ce qui devrait être
blame	Un blâme
if	Spéculation sur quelque chose d’incertain
reality	La réalité
unexpected	Inattendu, imprévisible

Le dictionnaire de concepts de EDR contient un ensemble de descriptions. Une description de concept est composée de deux concepts liés par une relation, à laquelle est associée un facteur de certitude (0 ou 1). Ainsi, EDR introduit des relations positives (certitude 1) et négatives (certitude 0). Ainsi, le dictionnaire de concepts permet de déterminer s'il est pertinent d'établir une relation entre concepts dans des circonstances normales.

Cette organisation du dictionnaire EDR ne permet que des relations de concept à concept. Il est donc impossible à EDR d'utiliser des relations complexes pour décrire la sémantique d'une entrée. Aussi, le dictionnaire de concepts peut indiquer, pour le concept *joueur de tennis* :

«joueur de tennis» ← agent — «jouer»

mais pas :

«joueur de tennis» ← agent — («jouer» — object → «tennis»)

Les relations que l'on trouve dans le dictionnaire de concepts sont de deux types, suivant leur origine. Les relations du premier ordre sont basées sur l'intuition humaine, alors que les relations du deuxième ordre sont issues des résultats d'analyse des phrases du corpus EDR.

2.3.2. Classification des concepts

EDR a développé une classification de concepts afin de minimiser les relations à stocker (en faisant hériter aux concepts les relations de leurs super-concepts).

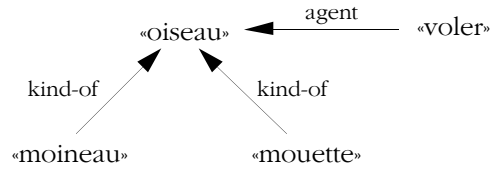


Figure 2.11 : Un exemple de classification de concepts

Pour cela, EDR a défini une relation particulière, nommée *kind-of*, qui relie un concept à son super-concept. Par cette classification, il est possible de déterminer les similarités entre concepts. Ainsi, grâce à la hiérarchie présentée en figure 2.11., on sait que les moineaux et les mouettes sont des oiseaux, et qu'ils volent.

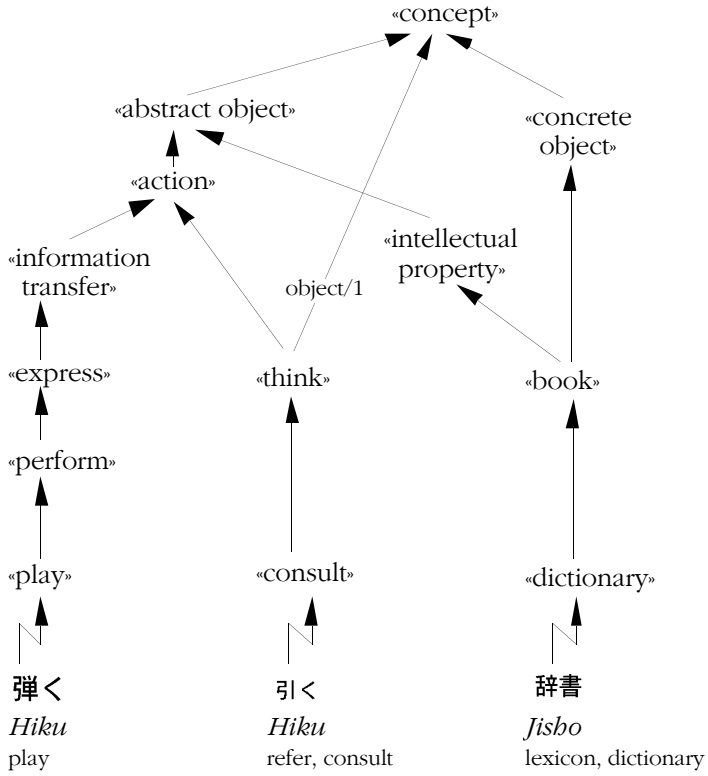


Figure 2.12 : Un extrait de la hiérarchie de concepts du dictionnaire EDR

Les relations du premier ordre (correspondant à l'intuition humaine) se trouvent plutôt dans les niveaux supérieurs de la hiérarchie, alors que les relations du second ordre (issues de l'analyse du corpus) se trouvent au niveau inférieur de cette hiérarchie.

Cette hiérarchie pose certains problèmes méthodologiques pour la création et la maintenance de la base lexicale. Il est en effet difficile de choisir les groupes de concepts ou de savoir sur quels critères les grouper. Pour cela, EDR a regroupé les concepts qui partagent un certain attribut, afin de repérer les groupes qui sont les plus représentatifs. De plus, un même concept peut appartenir à plusieurs groupes.

À l'intérieur de cette hiérarchie, les sous-concepts héritent des relations qu'entretiennent leurs super-concepts. Afin de dénoter des exceptions, comme *pingouin* qui est un oiseau mais ne vole pas, EDR a envisagé deux techniques.

La première consiste à ne pas dénoter l'appartenance de l'exception au groupe. Il n'y a donc plus d'exception, puisqu'on "oublie" qu'un pingouin est un oiseau.

Mais dans certains cas, où l'exception hérite de la grande majorité des relations du groupe, il est intéressant de garder l'exception à l'intérieur du groupe. Il faut alors indiquer la (ou les) relation(s) qui ne sont pas indiquées. Aussi, EDR a introduit des relations négatives entre concepts. Cette relation négative annule la relation positive héritée (voir figure 2.13.).

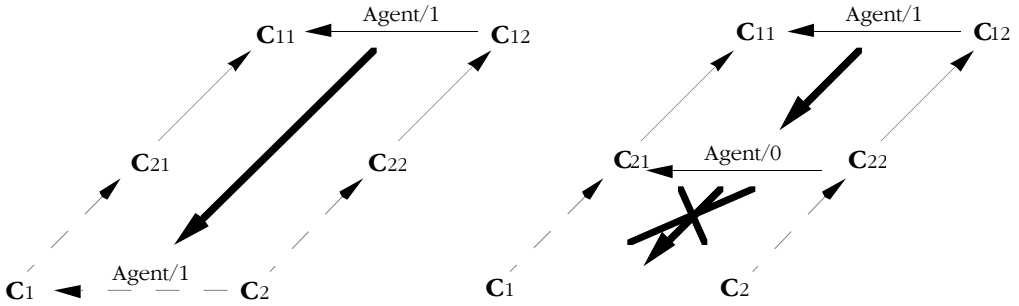


Figure 2.13 : Le mécanisme d'héritage et les relations négatives

Néanmoins, l'introduction de ces relations négatives entraîne des difficultés lorsqu'on obtient des schémas non prévus dans la définition de l'opération d'héritage. Aussi, ces relations ne sont introduites que par des lexicographes (et non par analyse de corpus).

3. Genelex

Un projet industriel nommé GENELEX a vu le jour grâce à une convention Européenne EUREKA. Ce projet regroupe des sociétés spécialisées en informatique linguistique, des universitaires et des éditeurs et utilisateurs de dictionnaires regroupés dans 3 pays (France, Italie et Espagne). Il implique environ 250 hommes-années.

L'objectif de GENELEX (GENEric LEXicon) est la construction d'un dictionnaire générique pour différentes langues européennes (pour l'instant, le français, l'italien et l'espagnol). Cela implique non seulement le développement d'un dictionnaire générique, mais aussi la mise au point d'une méthodologie qui garantira sa généricité au fur et à mesure de son évolution. Dans la suite, nous nous intéresserons aux travaux du consortium GENELEX France, sur les dictionnaires du français.

Le dictionnaire ainsi créé est considéré comme une grande base lexicale sans connexion directe avec un système de traitement des langues particulier. Les dictionnaires d'application seront générés par extraction des données nécessaires dans une forme adaptée aux besoins.

Pour GENELEX, une unité lexicale est le sens d'un mot, défini par les relations entre une unité morphologique, une unité syntaxique et une unité sémantique.

GENELEX a choisi de coder ses dictionnaires dans un format entités-attributs-relations. Ce choix permet de visualiser un unité lexicale comme un graphe. Cela permet aussi de placer chaque élément d'information sur un pied d'égalité (i.e. aucun nœud n'est privilégié lors des recherches, alors qu'une structure en arbre privilégierait la racine).

GENELEX a produit un dictionnaire public d'environ 3000 termes. Par ailleurs, les participants au projet ont transformé leurs dictionnaires privés au format défini par le consortium :

Hachette	55 000 termes
Notre temps	60 000 termes + 25 000 noms propres
SEMA Group	70 000 unités morphologiques
IBM	50 000 unités morphologiques
GSI-ERLI	68 000 unités morphologiques simples et 15 000 unités morphologiques composées

3.1. Le modèle conceptuel de GENELEX

Le modèle conceptuel de GENELEX définit le formalisme de haut niveau dans lequel un linguiste peut exprimer sa théorie linguistique. Ce formalisme de haut niveau doit être indépendant du stockage effectif des données.

GENELEX doit représenter les diverses descriptions existantes, compte tenu de ce qu'elles dépendront du modèle théorique, du degré de finesse et des critères discriminants retenus par le lexicographe, et ce, quelle que soit son école. Cela aura des répercutions :

- au niveau morphologique,
- au niveau du comportement syntaxique des unités lexicales,
- au niveau sémantique,
- au niveau des inter-relations entre les différentes informations codées.

Ces diverses descriptions apparaîtront soit dans plusieurs instances de dictionnaires, soit dans une seule et même instance construite par fusion de toutes les autres.

Le formalisme descriptif de Genelex est donc développé pour pouvoir contenir chacune des théories en usage. Il fonctionne donc comme un pont entre les différentes théories linguistiques.

Le modèle conceptuel GENELEX a été largement exprimé au travers de modèles entités-attributs-relations (Merise).

Beaucoup de contraintes d'intégrité sont exprimées dans ce formalisme : typage des objets, typage des relations, cardinalité des relations, etc. Cependant, ce modèle n'est pas fait pour exprimer des règles. Aussi, certaines contraintes ont dû être exprimées dans le document d'accompagnement (restriction sur les combinaisons de valeurs). Il s'ensuit que le modèle conceptuel de GENELEX combine l'utilisation du formalisme entités-attributs-relations et de commentaires en langage naturel.

Comme nous l'avons déjà mentionné, le formalisme de GENELEX se veut indépendant du stockage effectif des données. Cela permet à chaque membre du consortium GENELEX de construire ses propres outils sur la plate-forme de son choix. Un format d'échange est donc nécessaire pour assurer la compatibilité des différents outils.

Le consortium GENELEX a choisi SGML (Standard Generalized Markup Language) comme format d'échange. SGML est un langage de description de documents (cf. annexe A) qui

permet de décrire la structure et le contenu d'un ensemble de documents. La structure d'un document est donnée par une DTD (Document Type Definition).

GENELEX a donc traduit son modèle conceptuel en un modèle physique en construisant une DTD. Certaines des contraintes alors exprimées en langage naturel ont pu être exprimées formellement dans la DTD. Les autres contraintes apparaissent sous forme de commentaires.

3.2. Vue générale d'une unité du lexique

Comme l'ensemble des informations d'un dictionnaire Genelex, chaque unité du lexique peut être vue sous forme de graphe. Les unités sont organisées selon trois couches : unités morphologiques (UM), unités syntaxiques (USyn) et unités sémantique (USém).

Nous donnons en figure 2.14. l'articulation globale d'une unité du lexique.

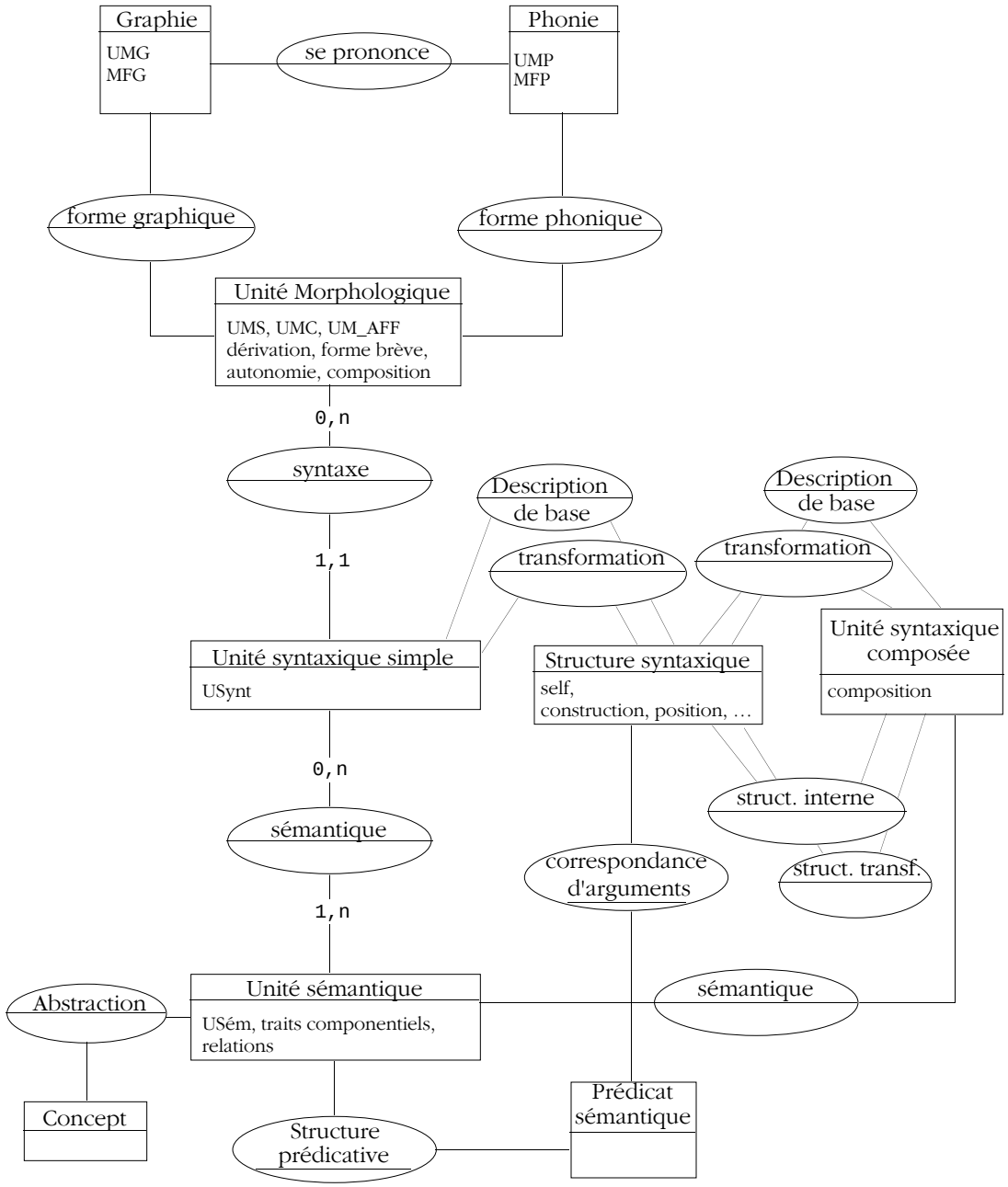


Figure 2.14 : L'articulation globale d'une unité du lexique

Une unité du lexique comprend donc une unité morphologique. Cette unité morphologique est reliés à 0, 1 ou plusieurs unités syntaxiques simples. L'unité syntaxique, par contre, n'est reliée qu'à une et une seule unité morphologique. Dans le cas d'expressions composées,

l'unité morphologique est reliée à plusieurs unités syntaxiques simples (celles des composants), elles-mêmes liées à une unité syntaxique complexe (décrivant le comportement syntaxique global du composé).

Une unité syntaxique simple est reliée à 0, 1 ou plusieurs unités sémantiques. L'unité sémantique est liée de manière biunivoque à un concept et à une structure prédicative. On connaît aussi la correspondance entre les arguments du prédicat et les éléments syntaxiques de l'unité.

Lorsqu'une unité sémantique est associée à plusieurs unités syntaxiques, celle-ci sont simples et proviennent de la même unité morphologique.

Avec ce schéma, il est possible de créer des dictionnaires n'ayant que des informations morphologiques, n'ayant que des informations morphologiques et syntaxiques ou ayant des informations morphologiques, syntaxiques et sémantiques.

3.3. Le modèle morphologique

Le modèle morphologique définit une unité morphologique (UM) et donne sa structure.

Une unité morphologique est le point d'entrée de la couche morphologique et le point de passage vers les autres couches.

Une unité morphologique est un regroupement de mots basé sur des propriétés morphologiques. Elle est identifiée par son lemme graphique et/ou par son lemme phonétique. La forme lemmatisée est la forme singulier s'il y a variation en nombre, masculin s'il y a variation en genre, et infinitif pour les verbes.

Dans certains cas, il est difficile de savoir si l'on a affaire à plusieurs unités morphologiques ou à une seule. Pour cela, GENELEX a défini des critères formels d'éclatement :

- si deux catégorie grammaticales peuvent être associées à une forme lemmatisée, on considère qu'on est en présence de deux lemmes distincts. Par exemple : *autiste* (nom) et *autiste* (adjectif). On note cependant la difficulté à distinguer certaines catégorie : nom/adjectif, participe passé/adjectif, participe présent/adjectif.
- si la variation en genre d'un nom reflète une variation sémantique (mise à part le changement de sexe), on considère que l'on est en présence de deux unités morphologiques distinctes. Par exemple : *un page/une page*, *un colonel/une colonelle* (la colonelle est la femme du colonel, pas un colonel féminin).
- si deux significations très distinctes (sans lien étymologique ou rhétorique) peuvent être associées à une forme lemmatisée, on considère que l'on est en présence de deux lemmes distincts. Par exemple : *fraise, poêle*. L'application de ce critère est laissée au lexicographe.

On ne peut malheureusement pas toujours déterminer si certains de ces critères s'appliquent ou non.

GENELEX a défini cinq types d'unités morphologiques :

- **UM simple** : une UM simple est associée à une graphie (plusieurs en cas de variantes) constituée d'une suite de caractères alphabétiques, de séparateurs (tiret, apostrophe, point) et de la marque éventuelle d'hyphénation. Par exemple : *demain*, *après-demain*, *aujourd'hui*.
- **UM affixes** : une UM affixe peut être de type préfixe, infixe ou suffixe, ou encore sans type dans le cas où elle ne prend son statut qu'en contexte de dérivation ou composition. Par exemple, *-tion* (suffixe), *re-* (préfixe), et *gyne* (sans type d'affixe), qui donne androgyne et gynécologue.
- **UM dérivées** : une UM dérivée est une unité morphologique simple qui entretient des liens de dérivation avec d'autres unités morphologiques (simples ou affixes). Ces unités

sont analysables, en ceci qu'elles sont généralement constituées de 0 à N préfixes, d'une base, et de 0 à N suffixes. Par exemple, ***ac****cultu****ation***.

- **UM composée** : La notion de composé étant très controversée, l'UM composée de Genelex est une expression complexe qu'un lexicographe a choisi de coder dans la couche morphologique. Ce choix repose sur un ensemble de critères linguistiques qui définissent la notion d'unité morphologique composée :
  - un des composants n'apparaît que dans cette expression complexe. Par exemple, *fur* dans *au fur et à mesure*.
  - particularité morphologique (changement de genre, nombre, mode flexionnel…) lors de la composition. Par exemple, *une deux-chevaux*, *un peau-rouge*.
  - particularité graphique (présence d'un séparateur graphique). Par exemple : *garde-malade*.
  - composé insécable (pas d'insertion autorisée). Par exemple : *à force de*. Les composés autorisant l'insertion tels que *mettre [qqch] en marche* relèvent de la composition au niveau syntaxique.
  - assimilable à une catégorie fonctionnelle terminale. Par exemple : *en vertu de* (assimilable à un préposition).
  - pas de composition sémantique. Par exemple : *une sage femme* (≠ femme qui est sage).
- **UM agglutinée** : permet d'enregistrer des phénomènes de contraction graphique de deux unités. Par exemple : *du* (= de + le).

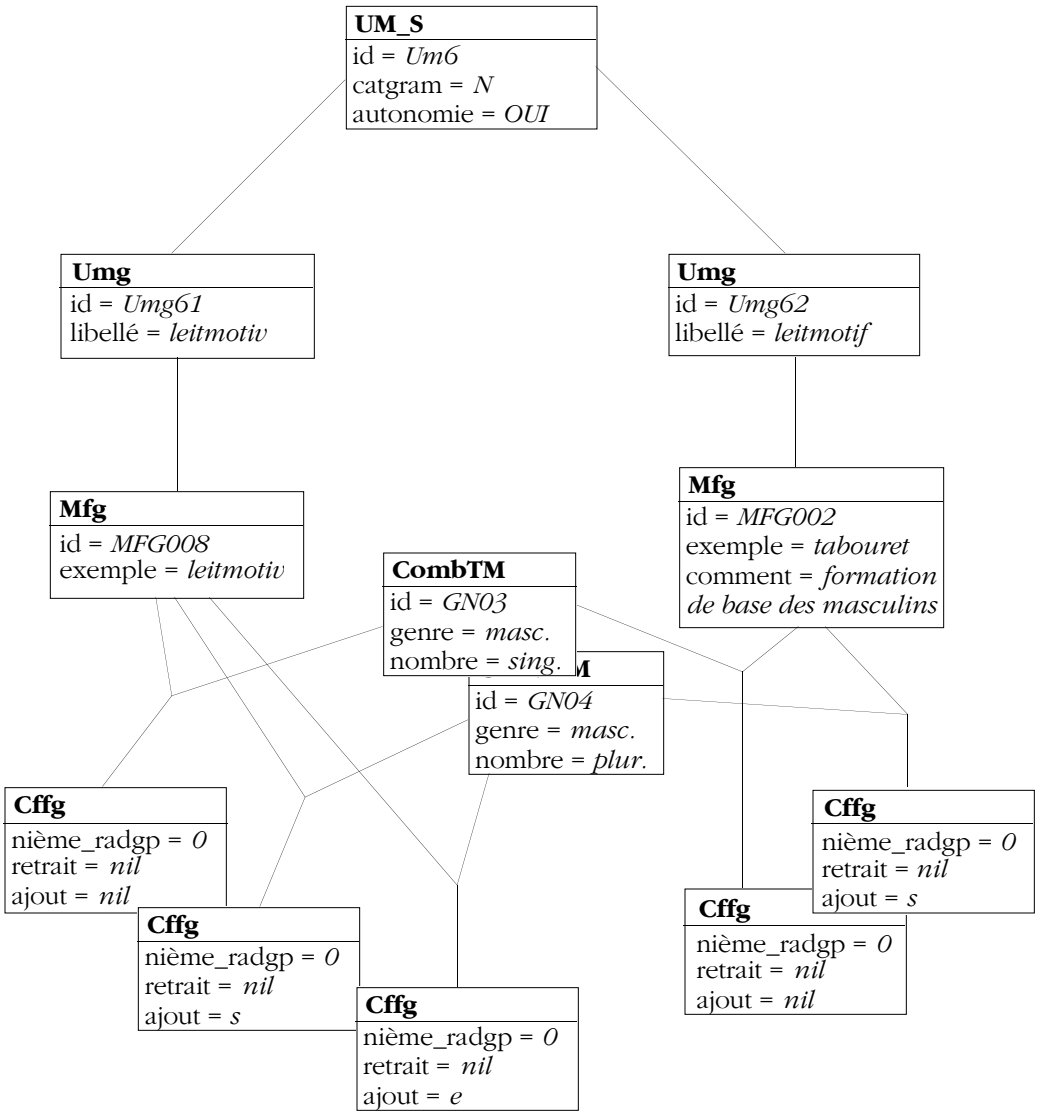


Figure 2.15 : Un exemple d'unité morphologique simple présentée sous forme graphique

Chacune de ces unités morphologiques possède une structure différente, définie par un schéma entités-attributs-relations. Le schéma entités-attributs-relations des unités morphologiques régit l'ensemble des interactions entres les différentes parties du graphe (l'unité morphologique, la graphie et la phonétique). Aussi, nous ne donnerons en figure 2.15. qu'un exemple concret d'utilisation du modèle Genelex pour une unité morphologique simple.

Dans cet exemple, on peut voir une unité morphologique ayant deux variantes graphiques, chacune avec un comportement morphologique différent. Cet exemple montre bien l'utilisation d'une structure graphique pour le codage d'informations linguistiques. Il permet aussi de voir les possibilités de factorisation des informations. Par exemple, le mode de flexion (Mfg) de la variante *leitmotif* est commun à l'ensemble des noms masculins réguliers.

3.4. Le modèle syntaxique

Le modèle syntaxique a pour but de décrire les comportements syntaxiques propres à l'unité lexicale décrite. Les comportements décrits sont ceux que l'appartenance de celle-ci à une catégorie et à une sous-catégorie ne suffit pas à prédire.

GENELEX a défini deux types d'unités syntaxiques (USyn) : les USyn simples et les USyn composées.

On associe à une unité morphologique autant d'USyn que l'on a identifié de comportements distincts. Une USyn simple décrit un comportement syntaxique pour exactement une UM. Elle est caractérisée par exactement une *description de base* décrivant un contexte syntaxique. On peut lui associer une ou plusieurs descriptions transformées.

Une description de base est l'association d'une *construction* décrivant un contexte syntaxique et d'un bloc d'information (appelé "self") enregistrant les propriétés propres à l'entrée décrite, lorsqu'elle apparaît dans ce contexte. Les constructions peuvent ainsi être largement partagées.

Une construction est définie par un ensemble de *positions* (Pi), avec indication d'optionnalité et, le cas échéant, de solidarité. La notion de position recouvre les notions de classes fonctionnelles, paradigmes distributionnels, actants, arguments, et rôles thématiques que l'on rencontre dans les différentes théories.

Une position se définit par un triplet :

- distribution (en terme de types de syntagmes, qui peuvent eux-mêmes être détaillés en termes de position s'ils sont non terminaux),
- fonction,
- rôle thématique.

Le formalisme permet donc des réécritures arborescentes, comme dans l'exemple de la figure 2.16. où l'on exprime une construction de l'unité lexicale *intéressant* (adjectif).

Comme nous l'avons vu dans le paragraphe précédent, les expressions courantes de la langue n'étaient codées sous formes d'UM que dans certains cas particuliers (présence de composant non autonome…). Ces expressions sont codées au niveau syntaxique par les unités syntaxiques composées.

Ces unités syntaxiques composées ne sont pas issues d'une unité morphologique. Leurs composants lexicalisés peuvent être soit des UM, soit des USyn. Leur comportement "externe" est décrit de la même façon que pour les unités syntaxiques simples. Leur structure interne est exprimée au moyen du même formalisme que les constructions.

L'interaction entre les composant ou les arguments et modifieurs est décrite. Les transformations possibles sont décrites (possessivation : `au grand dam de SN -> à son`

`grand dam`, `dans le but de P[mode: infinitif] -> dans ce but`). De plus, on peut mentionner les processus d'effacement : `fil de fer barbelé -> fil barbelé -> barbelé`.

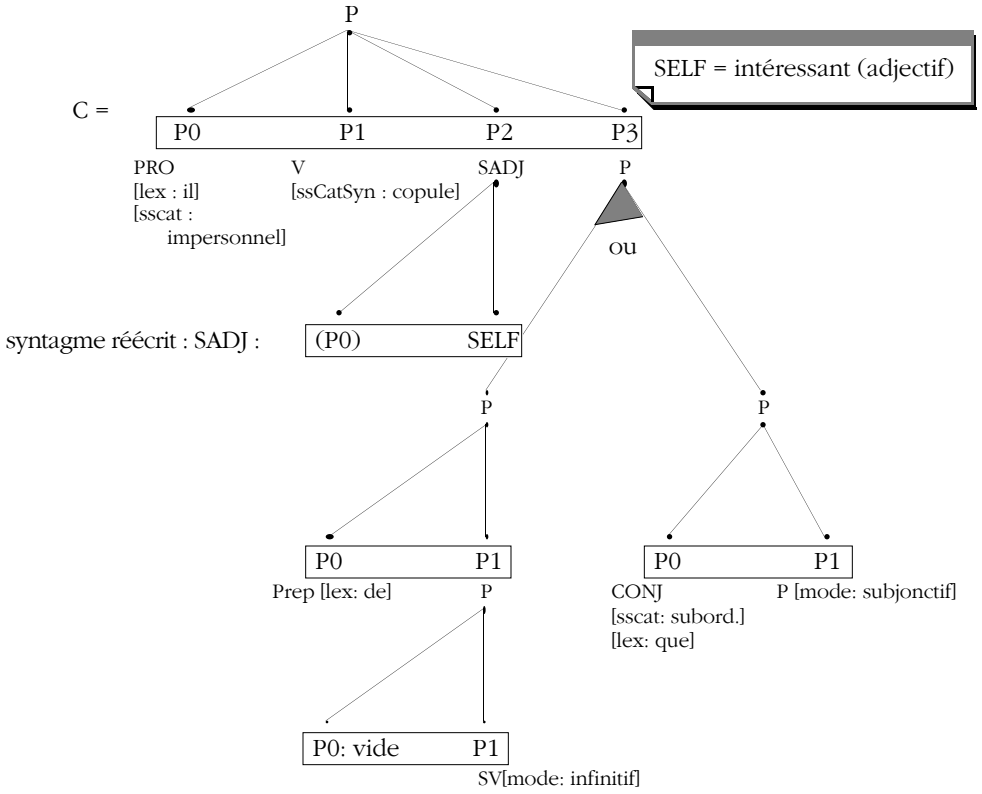


Figure 2.16 : Description de base associée à l'entrée "intéressant" (adjectif)

La figure 2.17. donne un exemple de construction syntaxique pour une USyn composée. L'interaction entre composants et arguments ou modifieur y est décrite (insertion d'un argument entre les composants, par exemple).

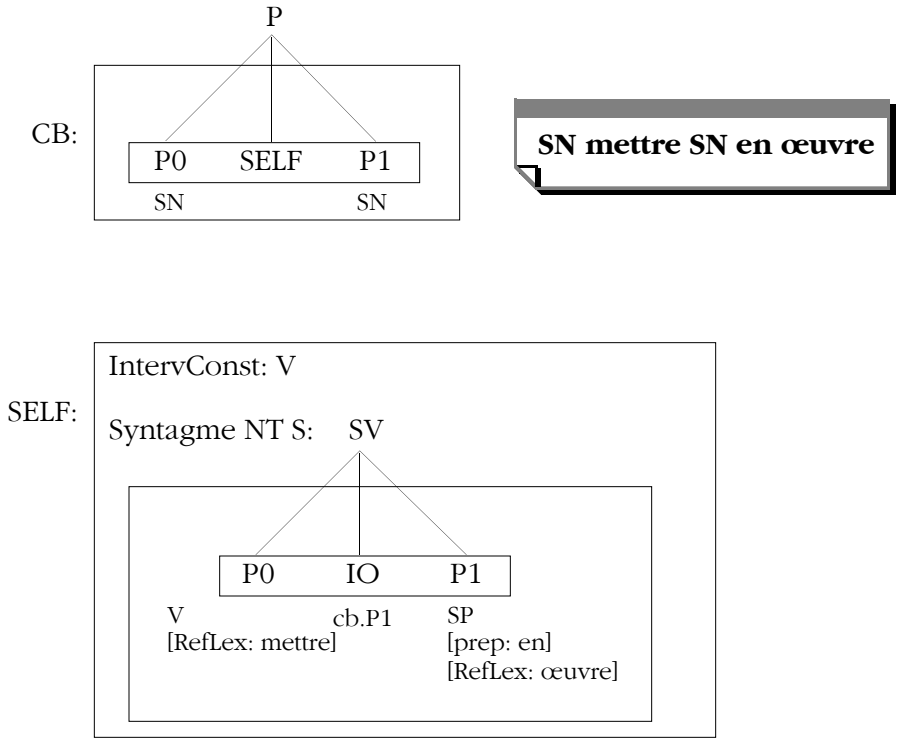


Figure 2.17 : Description de l'USyn composée "mettre en œuvre"

3.5. Le modèle sémantique

Vu le faible niveau de consensus dans la communauté scientifique en ce qui concerne la formalisation des connaissances sémantiques, GENELEX propose un méta-modèle à instancier plutôt qu'un modèle à implémenter directement sur les données lexicales.

Dans cet esprit, Genelex a clairement séparé la sémantique en deux niveaux :

- un niveau de **représentation sémantique linguistique** : cette représentation, très proche de la langue, est construite principalement à partir de l'observation du lexique en contexte et des relations sémantiques entre éléments du lexique. Ce niveau regroupe les informations sémantiques fines nécessaires à la traduction automatique de qualité ou à la génération, à la compréhension automatique de texte pour génération de résumé…).
- un niveau de **représentation sémantique conceptuelle** : cette représentation, issue des courants de l'intelligence artificielle, est d'une plus grande "abstraction". elle s'appuiera sur des primitives, associées à un formalisme de représentation des connaissances.

La profusion de théories que GENELEX souhaite pouvoir coder dans son modèle l'a mené à choisir une approche **plus multi-théorique que a-théorique**.

L'unité sémantique permet d'accéder à l'ensemble des informations sémantiques correspondant à une certaine acception d'une entrée.

L'unité sémantique décrit donc le sens d'une unité morphologique dans un contexte syntaxique donné. D'autre part, une unité syntaxique, et donc implicitement l'unité morphologique dont elle est issue, peut être associée à plusieurs unités sémantiques.

Les relations entre les différents modèles de GENELEX nous donnent donc une structure illustrée par l'exemple de la figure 2.18.

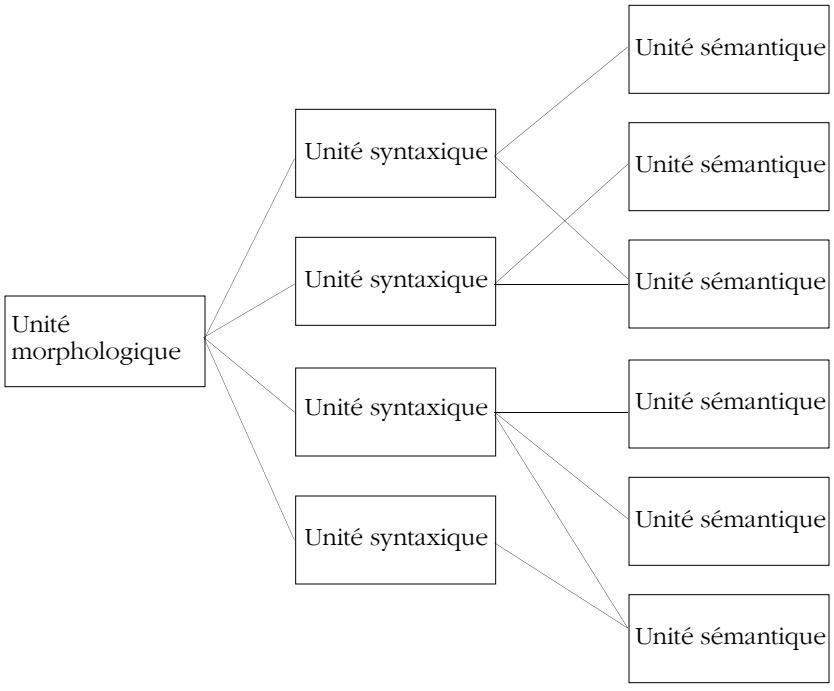


Figure 2.18 : Un exemple de relations entre les différents modèles GENELEX

Pour établir la correspondance entre syntaxe et sémantique, Genelex introduit la notion de prédicat. Toute unité sémantique peut-être associée à un prédicat. Il est ainsi possible de noter les liens entre les différentes positions (en syntaxe) et les arguments du prédicat (en sémantique). À chaque argument d'un prédicat est associée une information sémantique en terme de traits ou de rôle sémantique. On peut aussi lui associer une valeur par défaut, afin de traiter les cas où l'argument est absent en surface.

Enfin, il est possible d'établir des relations entre les différentes unités sémantiques du dictionnaire. Cette possibilité permet donc de coder un thesaurus, voire d'autres liens plus fins, comme les fonctions d'Igor Mel'čuk.

4. MULTILEX

MULTILEX est un projet ESPRIT (DG XIII), qui a débuté en décembre 1990 et s'est achevé en décembre 1993. Son but était de proposer des standards pour les bases lexicales multilingues. Les 3 années du projet ont été réparties en deux phases. La première phase, axée vers la recherche, a consisté en la définition des standards à adopter. La seconde, axée vers les applications, a expérimenté, corrigé et développé les standards ainsi proposés.

MULTILEX a proposé différents types de standards, portant :

- sur les **informations linguistiques** : quelles sont les informations que l'on met dans la base, sous quelle forme…) ?
- sur **l'architecture linguistique** : comment les entrées sont-elles organisées ?
- sur **l'architecture lexicale** : comment les dictionnaires sont-ils organisés ?
- sur **l'architecture informatiques** : comment sont organisés les différents outils ?

Enfin, MULTILEX a développé un prototype de dictionnaire quadrilingue (anglais, français, allemand et italien) selon ces standards.

L'architecture d'une base lexicale MULTILEX est basée sur des dictionnaires monolingues et bilingues.

Les unités lexicales sont représentées avec un formalisme basé sur les structures de traits typés. Un langage a été défini pour décrire la structure d'une unité lexicale. Un autre, défini par nos soins, permet de coder des règles de cohérence et d'intégrité. Chaque manipulation de l'information linguistique est faite dans ce formalisme. Ensuite, les entrées sont stockées dans une base de données relationnelle.

MULTILEX a aussi développé un standard linguistique pour la représentation d'unités lexicales de langues européennes. Ce standard code l'ensemble maximal d'informations linguistiques commun aux langues européennes.

4.1. Architecture lexicale

Une base de données lexicales MULTILEX est un ensemble de dictionnaires monolingues et bilingues. L'architecture lexicale prévoit un dictionnaire monolingue par langue, et deux dictionnaires bilingues unidirectionnels par couple de langues (voir figure 2.19.).

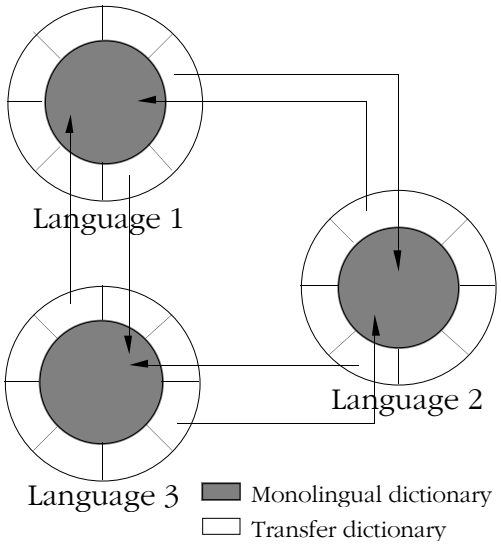


Figure 2.19 : L'architecture lexicale de MULTILEX

Un dictionnaire est un ensemble d'articles. On ne trouve qu'un type d'article par dictionnaire.

Un dictionnaire monolingue est un ensemble d'entrées monolingues (unités lexicales). La structure d'une unité lexicale est détaillée dans le paragraphe 4.2.1.

Un dictionnaire bilingue est un ensemble d'unités de transfert, reliant (de manière uni-directionnelle) des entrées d'une langue source aux entrées d'une langue cible.

4.2. Architecture linguistique

L'architecture linguistique définit les objets de base d'un dictionnaire et leurs relations. Le modèle Multilex ne restreint pas la liste des traits pour chaque langue. Aussi cette architecture linguistique régit-elle les relations entre des ensembles de traits linguistiques, plutôt qu'entre de simples traits.

Cette architecture garantit la consistance du modèle à haut niveau, et ce, quelle que soit la langue considérée.

Cette architecture fournit un squelette sur lequel les linguistes peuvent répartir les informations en utilisant des structures de traits. Un langage a été défini pour permettre au linguiste de spécifier les traits à ajouter à ce squelette pour un dictionnaire particulier.

4.2.1. Dictionnaires monolingues

Les articles d'un dictionnaire monolingue sont basés sur un modèle à deux niveau, qui garantit l'indépendance entre, d'une part, le comportement phonétique et graphique d'un mot, et, d'autre part, les sens exprimés par ce mot.

Le premier niveau de description d'une entrée est fourni par une GPMU (Graphic Phonologic Morphological Unit). Ce GPMU est un triplet regroupant les comportements orthographiques, phonologiques et morphologiques associés à une forme canonique. Dans le cas des variantes ("colineau", "colinot" en français, "yoghurt", "yogurt", "yoghourt" en anglais), chaque forme est représentée par une GPMU distincte.

Le second niveau de description est réalisé par les LU (Lexical Unit). Une LU identifie un sens de mot dans une langue donnée.

Ainsi, différentes LU peuvent renvoyer à une même GPMU (homonymie) et une LU peut être liée à différentes GPMU (variantes).

Les abréviations sont traitées comme des variantes d'une entrée. Les mots composés sont traités comme les entrées simples. Ainsi, le terme "Application Programming Interface" est représenté par une seule LU, liée à deux GPMUs (une pour l'entrée standard, l'autre pour son abréviation : "API"). La différence entre entrée composée et entrée simple se reflète dans les informations contenues dans la GPMU.

La GPMU d'une entrée complexe (appelée CGPMU) contient elle aussi des informations orthographiques, phonologiques et morphologiques, et de plus une référence aux constituants de l'entrée complexe (liens vers d'autres GPMUs). Si nécessaire, elle contient aussi des contraintes sur les différents constituants.

Ainsi, la CGPMU de "Application Programming Interface" renvoie aux GPMUs de "application", "programming" et "interface". De plus, elle indique que le pluriel ne s'applique qu'à "interface".

Une LU donne accès à différents types d'informations (sous forme d'ensembles de traits). Ainsi, une LU est reliée à une et une seule unité sémantique, une ou plusieurs unités syntaxiques et zéro ou plusieurs unités de transfert.

Un article de dictionnaire MULTILEX a donc la forme décrite par la figure 2.20.

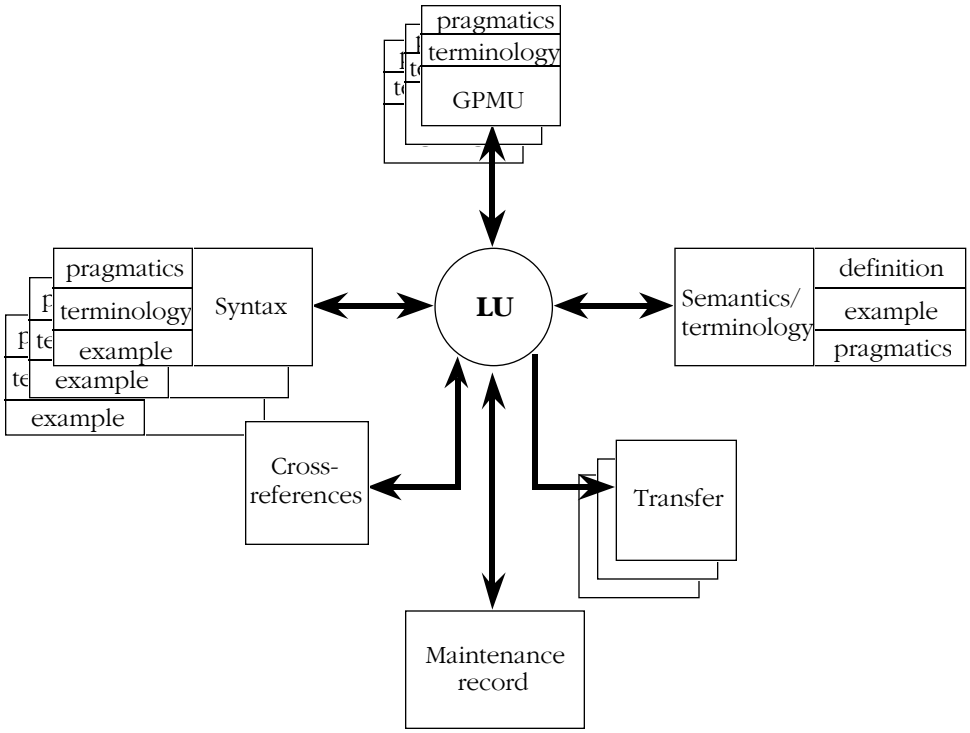


Figure 2.20 : Vue générale d'une LU (Lexical Unit) de MULTILEX

4.2.2. Dictionnaires de transfert

Une LU d'un dictionnaire représente un sens dans une langue particulière. Les LUs et leur organisation peuvent être différentes d'une langue à l'autre. Elle sont néanmoins utilisées comme point de départ et point d'arrivée pour des relations multilingues.

Dans l'organisation de MULTILEX, le multilinguisme est réalisé par une organisation multi-bilingue, par transfert, et unidirectionnelle.

4.2.2.1. Multi-bilinguisme

Pour chaque LU d'un dictionnaire monolingue, il est possible d'associer des LUs équivalentes de différents autres dictionnaires monolingues (voir exemple de la figure 2.21.).

Chaque relation bilingue est codée et stockée indépendamment. On peut donc associer des informations différentes à chacune des relations bilingues. De plus, si différents équivalents d'une LU existent dans une autre langue, on créé une relation bilingue par équivalent (voir exemple de la figure 2.21.).

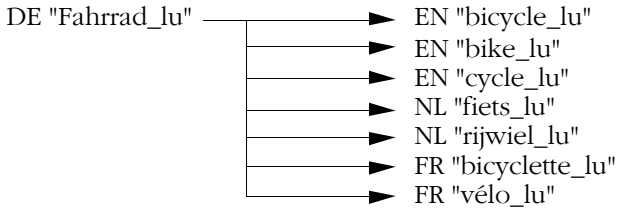


Figure 2.21 : Relations multi-bilingues de la LU allemande "Fahrrad"

4.2.2.2. Approche par transfert

Les relations multilingues sont basées sur un modèle par transfert. Chaque dictionnaire monolingue décrit ses LUs suivant des critères monolingues, et le transfert donne des moyens de passer d'informations issues du dictionnaires source à des informations utilisable pour la langue cible.

Les équivalences bilingues contiennent des informations contrastives entre la LU source et la LU cible. Ces informations contiennent notamment des indications sur les différence de sens entres les LUs, des conditions syntaxiques sur l'application de la correspondance et des transformations à appliquer pour effectuer le transfert.

4.2.2.3. Approche unidirectionnelle

Une relation bilingue est principalement constituée de deux parties : une condition syntaxique d'application et une transformation à appliquer aux informations de la langue source pour obtenir une structure valide en langue cible.

Il n'est pas possible, dans le cas général, d'inverser cette relation. En effet, s'il est possible (dans certains cas) d'inverser la transformation donnée, on ne peut déduire des informations présentes les conditions syntaxiques à appliquer à l'entrée en langue cible pour sélectionner le transfert inverse.

Le modèle de MULTILEX est donc fortement unidirectionnel.

4.3. Architecture logicielle

Parallèlement aux standards linguistiques, Multilex a développé des standards informatiques pour l'implémentation de bases lexicales multilingues. Ces standards régissent le format d'échange des données, l'architecture lexicale à adopter et l'ensemble des outils à construire pour la maintenance d'une base lexicale multilingue.

Les outils se conformant à ces standards peuvent le faire selon plusieurs niveaux, le minimum requis étant l'utilisation du format d'échange des données : MLEXd.

Cette architecture logicielle ressemble à l'architecture logicielle définie plus avant dans cette thèse, car, lors de ma première année de thèse, j'ai collaboré au projet MULTILEX. L'architecture logicielle que j'ai proposée a été retenue par le consortium. Celle proposée dans ce travail en est la version suivante.

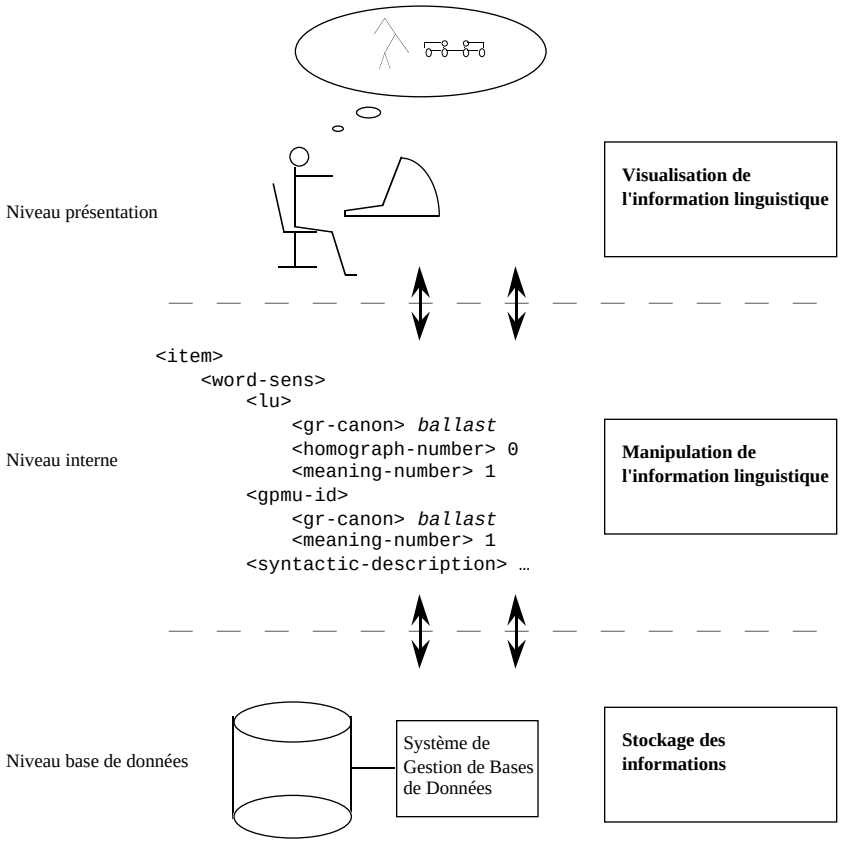


Figure 2.22 : Architecture logicielle d'un système de gestion de bases lexicales selon MULTILEX

L'architecture logicielle MULTILEX distingue fortement les problèmes de stockage, de manipulation, et de visualisation des données. Elle est basée sur trois niveaux :

- **niveau base de données** : ce niveau est en charge du stockage effectif des données. Différents systèmes relationnels de gestion de bases de données peuvent être utilisés à ce niveau, qui est invisible pour l'utilisateur.
- **niveau interne** : ce niveau est en charge des différentes manipulations sur les données des entrées de dictionnaires. C'est à ce niveau que les différents outils d'un système de gestion de bases lexicales opèrent. Pour ce niveau, MULTILEX conseille d'utiliser des structures de traits typés.
- **niveau présentation** : ce niveau est en charge de la présentation des informations à l'utilisateur. Cette présentation n'est pas nécessairement proche de la structure interne utilisée. De plus, il peut être possible de proposer différentes présentation d'une même information pour différents utilisateurs ou différents buts.

Cette architecture est illustrée par la figure 2.22.

Multilex a aussi défini un ensemble d'outils standard pour un système de gestion de bases lexicales multilingues. Ces différents outils manipulent les données linguistiques au niveau interne et interagissent avec l'utilisateur au niveau externe.

Les outils ainsi définis sont les suivants :

- **Éditeur** : l'éditeur permet d'éditer ou de modifier des entrées du dictionnaire. Pour cela, il doit offrir des fonction de navigation à l'intérieur de la base lexicale. Afin de permettre l'édition et la modification des entrées, l'ensemble des informations linguistiques doit être présenté à l'utilisateur.
- **Navigateur** : le navigateur permet de consulter la base lexicale. Il doit permettre différentes présentations suivant les motivations de l'utilisateur. Il n'est pas nécessaire que les présentations reflètent la totalité des informations linguistiques associées à une entrée. Éventuellement, l'outil donnera un moyen à l'utilisateur de spécifier sa propre présentation.
- **Vérificateur de cohérence** : cet outil permet de vérifier des règles de cohérence définies par l'utilisateur. Lorsque certaines de ces règles sont violées, le vérificateur le signale à l'utilisateur. Cet outil permet la vérification de cohérence à l'intérieur d'un article, la cohérences entre articles d'un même dictionnaire, et la cohérence entre articles de différents dictionnaires.
- **Défauteur** : cet outil permet de compléter des entrées incomplètes. Cet complétion peut avoir lieu a posteriori sur des entrées déjà présentes dans le dictionnaire, ou interactivement afin d'accélérer le processus d'édition. Cet outil utilise des règles de valeurs par défaut définies par un linguiste.
- **Import/export** : cet outil génère et accepte des fichiers MLEXd reflétant la structure linguistique des entrées du dictionnaire. Le format MLEXd est un format basé sur SGML, codant des structures de traits.

III. Les problèmes intéressants

Chacun des systèmes étudiés précédemment présente des particularités très intéressantes et il serait sans doute vain de prétendre proposer un n+1ème standard à ajouter à la collection. Notre but n'est donc pas d'introduire un nouveau formalisme révolutionnaire, ou de donner de nouvelles méthodes de codage d'informations linguistiques.

Il nous semble plus intéressant de tenter d'unifier ou de généraliser certaines solutions existantes ou originales, dans un domaine où les coûts incitent à un partage du travail.

Cette volonté d'unification signifie qu'à terme il doit être possible de construire un outil qui permette au moins le codage de chacun des projets étudiés précédemment. Dans ce contexte, les problèmes les plus intéressants se situent au niveau de l'architecture lexicale, de l'architecture linguistique, et de la présentation de l'information.

1. Architecture lexicale

L'architecture lexicale d'une base lexicale définit l'ensemble des dictionnaires qu'elle contient, leurs types et leurs relations. Elle régit donc, notamment, la manière dont est réalisé le multilinguisme.

Lors de l'étude des projets passés ou en cours, nous avons distingué plusieurs architectures lexicales. Pour MULTILEX, le multilinguisme est réalisé par la combinaison de dictionnaires monolingues et bilingues. Pour EDR, le multilinguisme est réalisé à la fois par des dictionnaires bilingues et par un dictionnaire interlingue de concepts.

En étudiant non seulement les projets de bases lexicales, mais aussi les projets de traduction automatique, on s'aperçoit que les approches bilingues et interlingues s'opposent bien souvent.

L'approche bilingue (figure 3.1.) se base sur un ensemble de dictionnaires bilingues pour établir les liens entre des entrées de deux langues différentes. Ces dictionnaires bilingues se présentent comme un ensemble de liens entre des entrées des langues source et cible. Chacun de ces liens porte des informations linguistiques qui permettent de coder des phénomènes particuliers entre les langues sources et cibles.

Le principe général de cette approche découle des travaux sur les systèmes de traduction automatique basés sur le transfert.

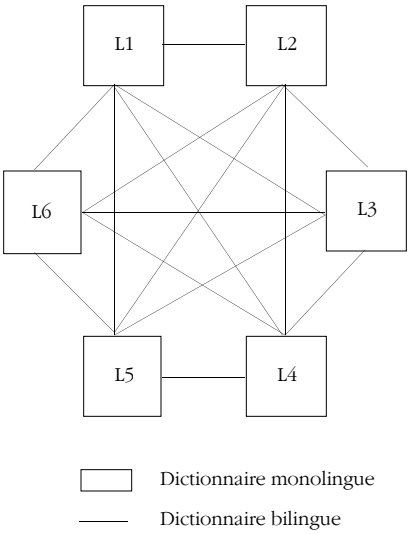


Figure 3.1 : Une base lexicale basée sur l'approche bilingue

L'approche interlingue (figure 3.2) utilise un langage artificiel intermédiaire (appelé *interlangue* et employé comme langage pivot) pour réaliser le lien entre les langues.

Les unités, généralement sémantiques, des langues de la base peuvent être représentées par cette interlangue, indépendamment de la langue de l'entrée. Aussi une interlangue doit-elle avoir son propre lexique et son propre ensemble d'attributs et de relations.

Une interlangue doit être définie en référence à un certain ensemble de langues naturelles, à moins qu'un univers de référence fixe (ontologie) ne soit représenté de manière autonome dans la machine.

Une interlangue consiste en deux parties distinctes : un lexique et un ensemble d'attributs et de relations.

La première partie d'une interlangue est le lexique, qui doit être suffisamment complet pour représenter les différents sens des mots trouvés dans l'ensemble des langues considérées.

Comme une interlangue est définie pour établir un lien entre les langues, ce lexique interlingue doit fournir un lien lexical entre les mots dans différentes langues. Aussi, deux sens équivalents de différentes langues doivent-ils être reliés à une seule unité interlingue.

Hélas, il n'y a pas nécessairement correspondance directe entre les sens des mots de différentes langues. Prenons l'exemple des mots français "fleuve" et "rivière" (dans leur sens concret le plus commun). Ces deux mots sont traduits en anglais par le mot "river" (dans son sens le plus commun). Les deux mots français ont deux sens différentsque l'anglais ne distingue pas. Un lien doit donc être établi entre ces sens dans le lexique interlingue. Par contre, cette distinction n'est pertinente que si l'on va de l'anglais vers le français. Dans un contexte de traduction anglais-japonais, cette distinction n'a pas lieu d'être, puisque le mot japonais "kawa" recouvre le même sens que le mot anglais "river".

La seconde partie de l'interlingue est l'ensemble de ses attributs et relations. Cet ensemble d'attributs et de relations doit être suffisamment complet pour permettre de coder les aspects linguistiques de toutes les langues considérées.

Cette partie n'est pas simple à définir, même si des études linguistiques fondamentales produisent de plus en plus de "microthéories" interlingues ou universelles (selon les termes de [Nirenburg & Defrise 1990a]) pour des phénomènes linguistiques, tels que l'aspect, le temps, la modalité, etc. qui, 20 ans plus tôt, semblaient ne pouvoir être décrits que par référence à une langue.

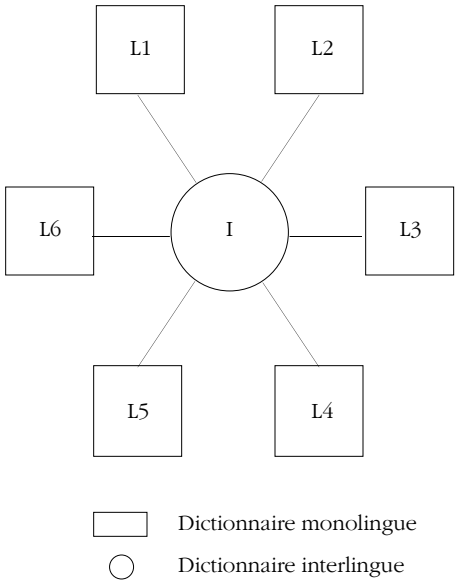


Figure 3.2 : Une base lexicale basée sur l'approche interlingue.

La critique la plus courante de l'approche par transfert porte sur le nombre de dictionnaires à définir : le nombre de dictionnaires bilingues théorique est fonction du carré du nombre de dictionnaires monolingues. Si n_m est le nombre de dictionnaires monolingues, et n_b le nombre de dictionnaires bilingues, on a:

$$n_b = \frac{n_m (n_m - 1)}{2}$$

Ce nombre est multiplié par deux lorsque l'on utilise des dictionnaires bilingues unidirectionnels.

Ce calcule se base sur l'hypothèse que tous les couples de langues doivent être présents et que la charge de traduction est répartie de manière uniforme entre les langues. Or, même à l'Union Européenne, qui est considérée comme l'organisme travaillant dans le contexte la plus multilingue (9 langues), l'effort de traduction n'est pas réparti de manière uniforme parmi les langues, les principales étant l'allemand, l'anglais, et le français.

Aussi, l'approche par transfert est la plus utilisée à l'heure actuelle. En effet, la plupart des projets ne se situent pas dans les hypothèses retenues pour effectuer le calcul précédent. Dans la plupart des cas, ces projets n'ont qu'une ou deux langues sources. Dans d'autres cas, il est possible de passer par un ensemble de langues intermédiaires qui agissent comme des pivots. Dans ce cas, le nombre de dictionnaire augmente de manière linéaire et non plus quadratique.

Pour sa part, la méthode interlingue permet de réduire le nombre de liens. En effet, il suffit de construire un lien entre chaque dictionnaire monolingue et le dictionnaire interlingue. On a ainsi une connectivité beaucoup plus réduite que dans l'approche bilingue.

Mais cet avantage se paie en complexité puisque ces liens sont beaucoup plus difficiles à gérer. En effet, l'ajout d'une entrée dans une langue peut remettre en cause les entrées du dictionnaire interlingue (et donc certains liens allant vers les autres langues). Il est très difficile d'évaluer la complexité qui se rajoute lorsque l'on utilise cette approche. De plus, on ne sait si la complexité de création d'une base interlingue n'augmente pas de manière quadratique avec le nombre de langue. Des expérimentations ont été conduites à une echelle suffisante au CICC (environ 50 000 mots de base et 25 000 termes techniques en japonais, chinois, malais,

indonésien et thai [Yaoliang & zhendong 1991]), mais on n'en trouve pas les résultats en termes de difficultés de développement de l'interlingue.

En partant de l'hypothèse selon laquelle le développement d'un dictionnaire interlingue n'augment pas de manière quadratique avec la nombre de langue et en se plaçant dans un contexte très fortement multilingue, on considère habituellement qu'une telle approche est justifiée lorsque l'on veut construire des bases lexicales comportant plus de huit langues ([Boitet 1988a, Boitet 1990a]).

À l'heure actuelle, la majorité des projets utilisant l'approche interlingue se basent sur une connaissance du monde (ontologie). Aussi, il est assez difficile de distinguer les problèmes entraînés par le choix d'une approche multilingue et ceux entraînés par une représentation des connaissances.

L'outil générique de gestion de bases lexicales multilingues (SUBLIM) défini dans cette thèse donne au linguiste le moyen de définir son architecture lexicale en déclarant explicitement l'ensemble des dictionnaires présents dans sa base.

La dernière partie de cette thèse propose dans ce cadre une nouvelle approche interlingue ne faisant pas appel à un modèle par connaissances.

2. Architecture linguistique

L'architecture linguistique d'une base lexicale définit la manière dont sont codées les entrées des dictionnaires qu'elle contient. Elle régit donc, notamment, les structures logiques qui seront utilisées dans le codage des informations linguistiques.

Lors de l'étude des efforts en cours, nous avons pu constater la multitude des choix qui se posent lorsque l'on veut représenter des informations linguistiques dans un dictionnaire.

De plus, on ne souhaite pas forcément représenter les mêmes informations. Une base lexicale voudra représenter toutes les informations morphologiques, syntaxiques et sémantiques alors qu'une autre se contentera de représenter la morphologie.

Les difficultés linguistiques ne s'arrêtent pas là. Dès le niveau morphologique, que l'on considère comme le plus simple et le plus mûr, on est confronté à des problèmes dus à l'organisation du dictionnaire et aux difficulté inhérentes de la langue (qu'est-ce qu'une entrée, puis-je avoir un espace dans une entrée, puis-je coder des expressions complètes, vais-je utiliser des tables pour coder la morphologie, ou bien un automate…). Ces mêmes problèmes se retrouvent à tous les niveaux.

À ces difficultés linguistiques s'ajoutent les problèmes dus à l'outil choisi pour gérer la base lexicale. En effet, si l'on souhaite réutiliser les outils de MULTILEX, il faut coder toutes les structures linguistiques avec la seule structure logique que propose MULTILEX : les structures de traits. De la même manière, l'utilisation des outils de Genelex oblige l'adoption d'une structure entités-attributs-relations. Certes, ces structures sont adaptées à certaines théories linguistiques, mais un outil ne doit pas préjuger des théories qui seront retenues par les linguistes.

Il existe une très importante variété de structures logiques utilisées dans certains dictionnaires. Parmi ces structures, on peut citer les ensembles d'attributs (EDR), les automates d'états finis (LADL), les graphes (GENELEX), les structures de traits (MULTILEX), les arbres étiquetés (Le Lexicaliste), les termes Prolog (ULTRA)…

Stuart M. Shieber [Shieber 1986] a défini trois critères d'évaluation des formalismes grammaticaux. L'un de ces critères est pertinent dans le contexte des bases lexicales :

- **Félicité linguistique** : le degré auquel les descriptions de phénomènes linguistiques peuvent être exprimées, directement ou indirectement, de la manière où le linguiste voudrait les exprimer.

Il est possible de coder une structure linguistique en utilisant n'importe quelle structure logique particulière. Néanmoins, ce codage n'est, bien souvent, ni naturel, ni pratique. De plus, il est souvent difficile de trouver une structure logique qui permette un codage naturel de la totalité des phénomènes linguistiques que l'on veut coder dans le dictionnaire. Aussi, il est difficile de satisfaire au critère de Shieber avec un système ne proposant qu'un type de structure logique.

L'outil générique de gestion de bases lexicales multilingues (SUBLIM) décrit dans cette thèse propose donc au linguiste une grande variété de structures logiques. Ainsi, il peut choisir la structure logique qu'il juge la plus adaptée à sa théorie linguistique. Il peut même coder des unités lexicales en utilisant différentes structures logiques.

3. Présentation de l'information

Les informations lexicales ont certaines particularités :

- cette information (parfois très complexe) est spécifiée, rentrée et utilisée par des linguistes non spécialisés en informatique,
- elle peut être utilisée à des buts très variés comme l'apprentissage d'une langue, la consultation d'une définition, la synthèse de parole, la Traduction Automatique…
- les performances des systèmes automatiques sont directement reliées aux informations lexicales qu'ils utilisent, ce qui rend cruciales les problèmes de maintenance et de correction,
- la taille d'une base lexicale est telle qu'elle nécessite plusieurs personnes pour l'indexage et la maintenance de cette information,

Tous ces facteurs demandent donc une présentation de cette information qui la rende facile d'accès à des fins de maintenance, alors que cette information est, en général, structurée pour un usage automatique.

Ce problème est crucial lorsqu'on laisse le linguiste libre dans le choix de ses structures. En effet, certaines structures, très appropriées pour une utilisation informatique, sont très difficiles à présenter, à lire et à comprendre. Ainsi, s'il est aisé de manipuler une structure d'automate représentée sous forme graphique, cela est beaucoup plus complexe si on souhaite la représenter sous une forme textuelle, en donnant par exemple la liste de ses arcs ou de ses nœuds.

Cet état de fait rend difficile le repérage d'erreurs éventuelles et leur correction. Il faut donc trouver un moyen de contrôler la manière dont est présentée l'information.

De plus, le mécanisme de présentation doit être suffisamment général pour permettre de masquer la structure interne de l'information lexicale.

Enfin, cette information peut être accédée pour différents usages. Dans certains cas, le lexicographe veut consulter l'ensemble des informations d'une entrée pour en vérifier la cohérence. Dans d'autre cas, il veut avoir une liste des entrées qui satisfont à un critère ou qui ont été classées par un tri. Dans ce cas, le lemme et la catégorie sont les seules informations qui l'intéressent. Il est donc important de pouvoir définir différentes présentations de l'information.

L'outil générique de gestion de bases lexicales multilingues (SUBLIM) décrit dans cette thèse propose au linguiste un moyen de définir différentes présentations de l'information. Cet outil permet donc :

- de contrôler la vue que l'on a de la structure interne,
- d'abstraire les informations linguistiques de leur codage informatique,
- de définir différentes vues d'une même structure, selon l'usage que l'on fait de la base lexicale.

Conception d'une base lexicale multilingue multiapplications

Introduction

Comme nous l'avons constaté dans la partie précédente, la diversité des solutions envisageables pour structurer une base lexicale, pour structurer une entrée du lexique ou pour présenter les informations linguistiques à l'utilisateur rend impossible le choix a priori d'une solution linguistique satisfaisant l'ensemble des utilisateurs.

Pourtant, les projets étudiés on souvent fait des choix restrictifs pour l'utilisation de leurs outils dans l'implémentation de bases lexicales variées.

Le projet MULTILEX a pourtant donné au linguiste un langage lui permettant de définir les traits qu'il souhaite coder dans son dictionnaire. Néanmoins, il ne peut utiliser que des structures de traits pour coder une théorie linguistique. Or, nous avons vu que de nombreuses autres structures logiques sont effectivement utilisées : graphes (GENELEX), automates (LADL), arbres (Le Lexicaliste)…

Certains diront qu'il est tout à fait possible de coder ces structures logiques avec des structures de traits. Cet argument est techniquement exact, mais ergonomiquement non fondé. En effet, ce type d'outil doit offrir au linguiste un niveau suffisant d'abstraction pour qu'il n'ait pas à se préoccuper d'un codage supplémentaire de l'information.

De plus, les projets existants ont figé l'architecture lexicale considérée. Il n'est donc pas possible d'utiliser leurs résultats pour expérimenter de nouvelles architectures lexicales.

Enfin, rares sont les projets qui se sont attachés au problèmes de la visualisation de l'information. Le langage de définition des traits linguistiques de MULTILEX, par exemple, permet de paramétrer le placement des éléments de la structures dans les écrans de saisie, mais il ne permet pas de visualiser la structure linguistique en faisant abstraction de sa représentation interne (structure de traits).

On veut donc construire un outil générique permettant de créer de nombreuses instances de bases lexicales qui ne seront pas forcément basées sur une même architecture lexicale, qui ne se fonderont pas sur la même théorie linguistique, qui n'utiliseront pas les mêmes structures logiques et où les informations ne seront pas visualisées de la même manière.

Aussi, nous avons choisi de travailler à la création de SUBLIM, un outil qui permettra au linguiste de créer et de gérer une base lexicale pour laquelle il a, au préalable, déclaré :

- les dictionnaires qu'elle contient : ces dictionnaires peuvent être monolingues, bilingues ou interlingues,
- pour chaque dictionnaire, quelles sont les structures qu'il contient : ces structures linguistiques peuvent être exprimées en utilisant différentes structures logiques de base (arbres, graphes, structures de traits, automates…) que l'on peut composer.
- pour chaque dictionnaire et pour chaque structure, la (ou les) manière(s) de les présenter.

Dans cette partie, nous présentons les principes et l'implémentation de cet outil. Nous commencerons par la partie concernant la définition des architectures lexicales, pousuivrons par celle concernant la définition de l'architecture linguistique, et finirons par l'architecture logicielle et les outils de gestion de la base lexicale.

IV. Définition d'une base lexicale multilingue

1. Définition de l'architecture lexicale

Lorsque l'on veut implémenter une base lexicale multilingue, la première décision à prendre concerne l'architecture lexicale. Cette architecture lexicale définit les dictionnaires de la base et le type de chacun. Ces dictionnaires peuvent être monolingues, bilingues ou interlingues.

L'information linguistique qui est associée aux unités lexicales n'a pas à être connue à ce niveau. Aussi, nous donnerons au linguiste un moyen de définir l'ensemble et le type des dictionnaires de la base. La définition formelle des unités du lexique se fera dans la section suivante.

Nous allons voir comment SUBLIM permet de déclarer l'architecture lexicale de trois bases différentes. Pour chacun de ces exemples, nous décrirons d'abord l'architecture lexicale en langue naturelle, puis nous la définirons formellement en utilisant le langage spécialisé LEXARD. Enfin, nous montrerons comment cette définition se traduit en termes d'implémentation.

1.1. Exemples

1.1.1. Une architecture bilingue

Le premier exemple montre comment définir l'architecture lexicale d'une base lexicale fondée sur l'approche bilingue. Pour cet exemple, nous nous inspirons de l'architecture lexicale des bases MULTILEX. La définition de cette architecture lexicale se ferra dans un fichier nommé `<nom de la base>.LEX`.

La base lexicale que nous souhaitons définir comprend 5 dictionnaires monolingues (anglais, français, italien, allemand et néerlandais). Ces 5 dictionnaires sont reliés par 20 dictionnaires bilingues unidirectionnels (voir figure 4.1.).

Les unités lexicales des dictionnaires monolingues sont des unités sémantiques. Les dictionnaires bilingues contiennent un ensemble de liens entre les unités du dictionnaire source et une ou plusieurs unités du dictionnaire cible.

Pour définir l'architecture lexicale de cette base, le linguiste commence par définir chacun de ses dictionnaires. Pour ce faire, il utilise un langage dont la syntaxe "noyau[2]" est en LISP.

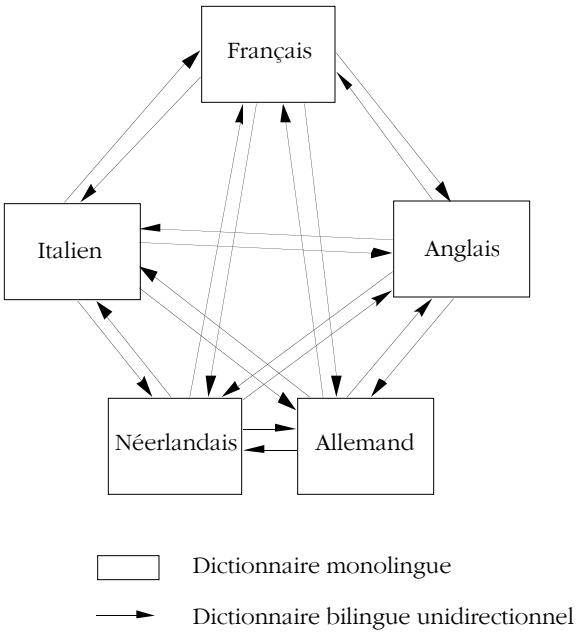


Figure 4.1 : Base lexicale quintilingue fondée sur une approche bilingue unidirectionnelle

La définition des dictionnaires monolingues se fait grâce à la fonction :

```
define-monolingual-dictionary name Keywords*
```

où *name* est un symbole définissant le dictionnaire de manière unique à l'intérieur de la base. Les *Keywords* définissent certains renseignements indispensables pour la gestion du dictionnaire. Ces mots-clés apparaissent comme une liste de couples attribut-valeur. L'attribut est noté avec un ":" au début du nom.

Voyons par exemple la définition du dictionnaire monolingue français de la base lexicale décrite précédemment :

```
(define-monolingual-dictionary french
     :language "Français"
     :owner    "GETA"
)
```

Ainsi, le dictionnaire monolingue français sera nommé de manière univoque par le symbole `french` dans l'ensemble de la base. L'attribut `:language` définit la langue du dictionnaire. Cette information est utile pour l'interface. L'attribut `:owner` définit le propriétaire du dictionnaire. Cet attribut est optionnel.

Les autres dictionnaires monolingues sont définis de la même manière.

Après avoir défini l'ensemble des dictionnaires monolingues, le linguiste peut définir, de manière analogue, les dictionnaires bilingues unidirectionnels, grâce à la fonction :

```
define-bilingual-dictionnary name Keywords*
```

où *name* est un symbole définissant le dictionnaire de manière unique à l'intérieur de la base.

---

[2] Mathieu Lafourcade propose une nouvelle définition du terme langage. Ses travaux sur les langages multi-dialectes [Gaschler & Lafourcade 1994a, Lafourcade 1994b] nous amènent à considérer un langage non pas par rapport à sa syntaxe (qui peut être multiple), mais par rapport à sa sémantique (qui, elle, est fixe). Ce genre de langage utilise une syntaxe privilégiée (appelée syntaxe noyau) qui sert à illustrer ses fonctionnalités. C'est cette syntaxe que nous utiliserons pour présenter nos travaux, sachant que cette syntaxe ne sera pas celle utilisée par le linguiste.

Ainsi, la définition du dictionnaire français-anglais se fera de la manière suivante (en admettant que le dictionnaire monolingue anglais a été nommé `english`):

```
(define-bilingual-dictionary french-english
     :type   unidirectionnal
     :source french
     :target english
     :owner  "GETA")
```

Ainsi, le dictionnaire bilingue français-anglais sera nommé de manière univoque par le symbole `french-english` dans l'ensemble de la base. L'attribut `:type` définit le type du dictionnaire bilingue. Le dictionnaire bilingue peut être unidirectionnel ou bidirectionnel. Les attributs `:source` et `:target` ne sont pertinents que dans le cas de dictionnaires bilingues unidirectionnels. Ils sont obligatoires dans ce cas. En effet, les structures linguistiques définies dans ce dictionnaire pourront utiliser ou se référer à des structures utilisées dans les dictionnaires source et cible. Dans le cas d'un dictionnaire bidirectionnel, ces attributs sont remplacés par l'attribut `:links` qui prend pour valeur une liste de dictionnaires.

Finalement, la définition de ces différents dictionnaires se fait dans le cadre de la définition d'une base lexicale multilingue particulière. Cette base lexicale est définie grâce à la fonction :

`define-lexical-database` *`name Keywords*`*

où *`name`* est un symbole définissant de manière unique la base lexicale.

Ainsi, notre base lexicale multilingue se définit par :

```
(define-lexical-database MULTILEX-like-database
     :owner   "GETA"
     :comment "Une base lexicale fondée sur une approche bilingue"
     :dictionaries
         (french english german dutch italian
          french-english french-german french-italian french-dutch
          ...
))
```

Notre base lexicale se nomme donc `MULTILEX-like-database`. Nous lui avons attribué un commentaire (une chaîne de caractères) et une liste de dictionnaires (ceux que nous avions définis auparavant).

1.1.2. Une architecture interlingue

Notre deuxième exemple montre comment définir l'architecture lexicale d'une base lexicale basée sur l'approche interlingue. Cette architecture lexicale est inspirée de l'architecture lexicale du système ULTRA [Farwell, Guthrie & Wilks 1993].

La base lexicale que nous souhaitons définir comprend 5 dictionnaires monolingues (anglais, chinois, japonais, espagnol et allemand). Ces 5 dictionnaires sont reliés par un dictionnaire interlingue (voir figure 4.2).

Les unités lexicales des dictionnaires monolingues sont des unités sémantiques. Le dictionnaire interlingue contient un ensemble d'unités interlingues appelées IR.

La définition des dictionnaires monolingues se fait exactement de la même manière que dans l'exemple précédent :

```
(define-monolingual-dictionary english
     :language "English"
     :owner    "CRL-NMSU")
```

La définition du dictionnaire interlingue se fait grâce à la fonction :

`define-interlingual-dictionary` *`name Keywords*`*

où *`name`* est un symbole définissant de manière unique le dictionnaire pour l'ensemble de la base.

Ainsi, la définition du dictionnaire interlingue se fera de la manière suivante :

```
(define-interlingual-dictionary IR
     :owner "CRL-NMSU"
     :links (english chinese japanese german spanish))
```

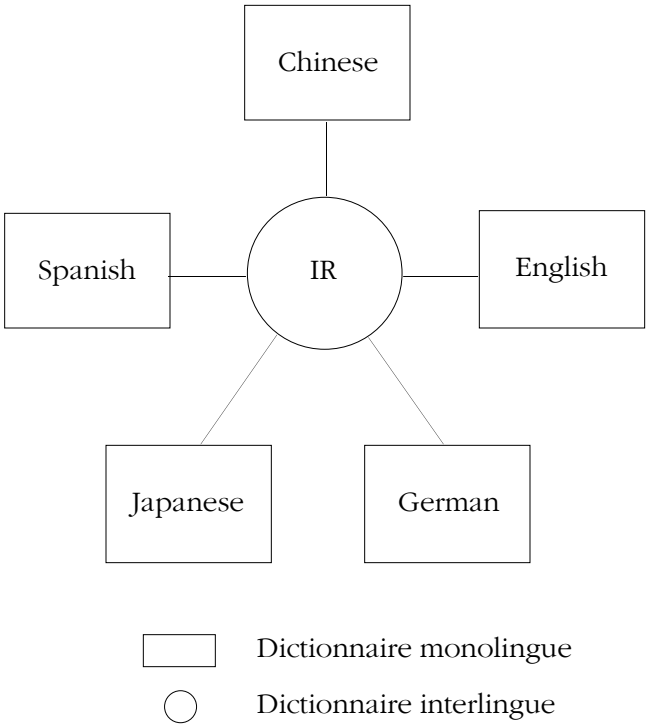


Figure 4.2 : Base lexicale quintilingue fondée sur une approche interlingue

Le dictionnaire interlingue sera nommé de manière univoque par le symbole `IR` dans l'ensemble de la base. L'attribut `:links` définit l'ensemble des dictionnaires monolingues reliés par le dictionnaire interlingue `IR`. La structure linguistique portée par les unités des dictionnaires interlingues peut donc utiliser ou se référer à des structures définies dans chacun des dictionnaires monolingues.

La base lexicale elle-même se définit comme auparavant :

```
(define-lexical-database ULTRA
     :owner   "CRL-NMSU"
     :comment "Une base lexicale fondée sur une approche interlingue"
     :dictionaries
         (english german spanish japanese chinese IR))
```

1.1.3. Une architecture mixte

Notre dernier exemple montre comment définir l'architecture lexicale d'une base lexicale basée à la fois sur une approche bilingue et sur une approche interlingue. Cette architecture lexicale est inspirée de l'architecture lexicale du projet EDR [EDR 1993].

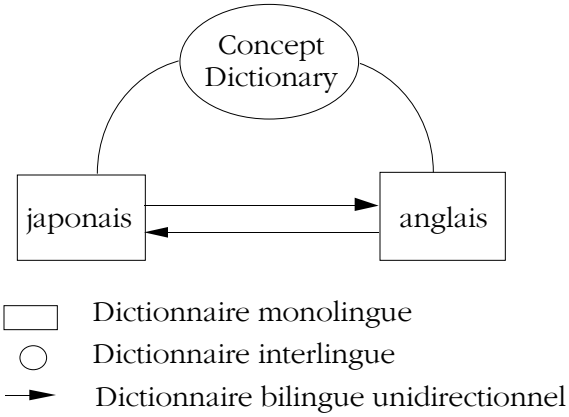


Figure 4.3 : Base lexicale anglais-japonais fondée sur une approche mixte

La base lexicale que nous souhaitons définir comprend 2 dictionnaires monolingues (anglais et japonais). Ces 2 dictionnaires sont reliés à la fois par deux dictionnaires monolingues unidirectionnels et par un dictionnaire interlingue (voir figure 4.3.).

Les unités lexicales des dictionnaires monolingues sont des unités sémantiques. Le dictionnaires interlingue contient un ensemble d'unités interlingues appelées concepts. Les dictionnaires bilingues contiennent un ensemble de liens reliant les unités du dictionnaire source à des unités du dictionnaire cible.

La définition des dictionnaires monolingues se fait exactement de la même manière que dans les exemples précédents :

```
(define-monolingual-dictionary english
     :language "English"
     :owner    "EDR")
```

Le dictionnaire interlingue se définit exactement comme dans l'exemple précédent :

```
(define-interlingual-dictionary concept-dictionary
     :owner "EDR"
     :links (english japanese))
```

Les dictionnaires bilingues se définissent exactement comme les dictionnaires du premier exemple :

```
(define-bilingual-dictionary japanese-english
     :type   unidirectionnal
     :source japanese
     :target english
     :owner  "EDR")
```

Et la base elle-même se définit comme dans les exemples précédents :

```
(define-lexical-database EDR
     :owner   "EDR"
     :comment "Une base lexicale fondée sur une approche mixte"
     :dictionaries
         (english japanese concept-dictionary))
```

1.2. Le langage de définition de l'architecture lexicale : LEXARD

Le langage LEXARD permet de définir une base lexicale. La définition d'une base lexicale passe par la création de ses éléments : les dictionnaires. LEXARD offre trois types de dictionnaires :

- dictionnaires monolingues,
- dictionnaires bilingues,
- dictionnaires interlingues.

La définition des dictionnaires monolingues se fait grâce à la fonction :

`define-monolingual-dictionary` *`name Keywords*`*

où *`name`* est un symbole définissant de manière unique le dictionnaire pour l'ensemble de la base. Les *`Keywords`* définissent des renseignements indispensables pour la gestion du dictionnaire :

`:owner` *`string`*	spécifie le propriétaire du dictionnaire.
`:language` *`string`*	spécifie la langue du dictionnaire.

La définition des dictionnaires bilingues se fait grâce à la fonction :

`define-bilingual-dictionnary` *`name Keywords*`*

où *`name`* est un symbole définissant le dictionnaire de manière unique à l'intérieur de la base.

`:owner` *`string`*	spécifie le propriétaire du dictionnaire.
`:type` *`type`*	spécifie le type (soit *unidirectionnel*, soit *bidirectionnel*) du dictionnaire.
`:source` *`symbol`*	spécifie le dictionnaire source (seulement si unidirectionnel).

`:target` *symbol* spécifie le dictionnaire cible (seulement si unidirectionnel).

`:links` *list* spécifie les dictionnaires liés par le dictionnaire bilingue (seulement si bidirectionnel). Cette liste a obligatoirement 2 éléments. Elle est donnée sous forme de liste de symboles, chacun correspondant à un dictionnaire défini par ailleurs.

La définition des dictionnaires interlingues se fait grâce à la fonction :

`define-interlingual-dictionary` *name Keywords**

où *name* est un symbole définissant le dictionnaire de manière unique à l'intérieur de la base.

`:owner` *string* spécifie le propriétaire du dictionnaire.

`:links` *list* spécifie la liste des dictionnaires liés par le dictionnaire interlingue. Cette liste est donnée sous forme de liste de symboles, chacun correspondant à un dictionnaire défini par ailleurs.

La définition de la base lexicale elle-même se fait grâce à la fonction :

`define-lexical-database` *name Keywords**

où *name* est un symbole définissant la base lexicale de manière unique. Les mots-clés admis sont :

`:owner` *string* spécifie le propriétaire de la base.

`:comment` *string* spécifie un commentaire sur la base lexicale.

`:dictionaries` *list* spécifie la liste des dictionnaires contenus dans cette base lexicale. Cette liste est donnée sous forme de liste de symboles, chacun correspondant à un dictionnaire défini par ailleurs.

Chacune de ces fonctions provoque la création d'instance des classes CLOS (Common Lisp Object System) prédéfinies : *lexical-database*, *monolingual-dictionary*, *bilingual-dictionary* et *interlingual-dictionary*.

La définition de l'architecture lexicale d'une base se fait dans un fichier dont le nom est composé du nom de la base lexicale, suivi de l'extension ".LEX" : `<nom de la base>.LEX`. La définition de l'architecture linguistique pour chacun des dictionnaires se fait dans un fichier dont le nom est celui du dictionnaire, suivi de l'extension ".LING" : `<nom du dico>.LING`.

2. Définition de l'architecture linguistique

De la même manière que l'on déclare comment sont organisés les différents dictionnaires d'une base lexicale, on doit, pour chaque dictionnaire, déclarer comment sont organisées les unités lexicales et leur information linguistique associée.

Pour cela, le système SUBLIM propose au linguiste un langage spécialisé pour la définition de structures linguistiques : LINGARD. Ces structures linguistiques seront représentées, dans le dictionnaire, avec différentes structures logiques (arbres, automates, structures de traits, graphes…). Le linguiste doit donc définir ses structures linguistiques en choisissant les structures logiques les mieux adaptées pour les coder.

Cette tâche est analogue à la définition des classes pour un langage à objets, à la définition de types de valeurs pour un langage algorithmique classique ou à la définition d'une structure de documents pour un éditeur de documents structurés (comme GRIF, LaTeX ou FrameMaker).

Dans cette section, nous commencerons par développer avec SUBLIM des exemples de structure que nous avons pu observer lors de notre étude du domaine. Nous commencerons par des structures simples, puis nous justifierons notre approche en codant des structures beaucoup plus complexes.

Nous décrirons ensuite l'ensemble des structures logiques de base connues du système, ainsi que leur comportement.

Enfin, nous étudierons l'implémentation de notre langage et montrerons comment il est possible de l'étendre en lui ajoutant de nouvelles structures logiques de base.

2.1. Exemples

2.1.1. Une structure simple : un dictionnaire ARIANE

La première structure que nous souhaitons implémenter est une structure simple contenant les informations nécessaires aux applications ARIANE. Cette structure peut être utilisée pour implémenter une base lexicale à partir de laquelle on générera les dictionnaires d'application pour un système de traduction développé en ARIANE.

Nous définissons la structure du dictionnaire français dans un fichier dont le nom est le nom du dictionnaire (tel qu'il est défini dans l'architecture lexicale) suivi de l'extension ".LING" : `<nom de dictionnaire>.LING`.

Une entrée de dictionnaire est un lemme associé à une catégorie. Cette entrée est la racine d'un arbre dont les nœuds sont des unités sémantiques (voir figure 4.4.).

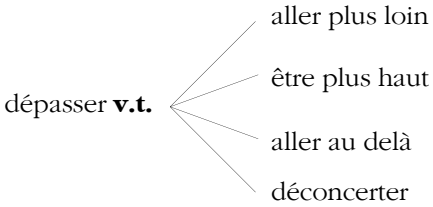


Figure 4.4 : Un exemple d'entrée de dictionnaire

Les unités sémantiques sont associées à une structure plus complexe constituée d'une simple structure attributs-valeurs (chaque valeur est atomique). Cette structure est illustrée par les tables données en figure 4.5.

catégorie	**attributs**	**valeurs possibles**
nc np	GNR NBR	mas, fem. sg, pl.
vb vbimp vbrefl	AUX RECIPROQUE VAL0, VAL1, VAL2, VAL3, VAL4 ASPECT	être, avoir. arg0/arg1, arg1/arg2. nom, à+nom, avec+nom, comme+nom, contre+nom, dans+nom, de+nom, en+nom, entre+nom, par+nom, parmi+nom, pour+nom, sur+nom, inf, à+inf, de+inf, adj, que+ind, que+subj, se-moy, se-pass, lieu-stat, lieu-dyn, manière, zéro. achevé, inachevé, début, fin, duratif, fréquent, instantané.
adj, adv, card, …		
TOUS	*DRVN* *DRVV* *DRVA*	*ncond, nlieu, ninstr, ncollect,* *nperson, adjrelat, adjqual, verbe;* *naction, nresult, nlieu, nagent,* *ninstr, adjact, adfpass, adjpotpas,* *adjresact, verbe*; *nabst, nperson, verbe.*

Figure 4.5 : Table des attributs et de leurs valeurs possibles

Les attributs donnés en italiques indiquent une dérivation. Cette dérivation a un type (un de ceux indiqués dans la colonne "valeurs possibles") et une valeur (une unité sémantique). DRVN dénote que l'unité sémantique courante dérive d'un nom.

Pour implémenter ce dictionnaire, le plus simple est d'utiliser les structures logiques suivantes :

- un **arbre** pour coder une entrée et ses raffinements de sens,
- des **structures de traits** pour coder l'information linguistique associée à chaque sens.

Pour cela, le linguiste définit un ensemble de classes. Chacune de ces classes hérite des classes prédéfinies (les structures logiques). Cette définition se fait grâce à la fonction :

```
define-linguistic-class name class-definition
```

où *name* est un symbole définissant de manière unique la classe définie à l'intérieur du dictionnaire. L'argument *class-definition* contient la définition d'une classe, notée de la manière suivante :

```
(logical-structure arguments*)
```

où *logical-structure* représente l'une des structures logiques dont on veut hériter le comportement. Les *arguments* dépendent de la structure logique spécifiée. Dans le cas d'un arbre, on notera les restrictions sur les classes linguistiques décorant les différentes parties de l'arbre. Dans le cas d'une structure de traits, on donne l'ensemble des traits et leurs valeurs possibles. Pour plus de détails, le lecteur se référera à la section suivante.

Pour coder ce dictionnaire, nous utiliserons les structures logiques d'arbre et de structure de traits. La définition d'une classe de structure logique *arbre* se fait comme suit :

```
(tree keywords*)
```

où les mots-clés restreignent les classes qui peuvent être valeurs de décoration des différentes parties de l'arbre. Les mots-clés possibles sont :

`:root class`	spécifie la classe acceptable pour la valeur de la décoration de la racine de l'arbre.
`:leaves class`	spécifie la classe acceptable pour la valeur de la décoration des feuilles de l'arbre.
`:nodes class`	spécifie la classe acceptable pour la valeur de la décoration de l'ensemble des nœuds de l'arbre (racine et feuilles comprises si elle ne sont pas définies par ailleurs).

La définition d'une classe de structure logique *structure de traits* se fait comme suit :

```
(feature-structure features)
```

où l'argument *features* est une liste de couples parenthésés représentant la liste des attributs définis dans la structure, avec la classe acceptable en valeur de chaque attribut.

La catégorie d'une entrée ne peut prendre qu'une valeur parmi un ensemble fini de valeurs. Pour cela, nous utiliserons la structure logique *one-of* qui prend comme argument la liste des valeurs possibles :

```
(one-of possible-values)
```

Ainsi, l'entrée du dictionnaire est définie comme un arbre dont la racine est décorée par une structure de traits simple (contenant une catégorie et une forme graphique) et dont les nœuds sont décorées par des unités sémantiques :

```
(define-linguistic-class entry
   (tree :root   (feature-structure
                     (graphic-form string)
                     (category     cat))
         :nodes  sem-unit))
(define-linguistic-class cat
   (one-of (nc np vb adj card deict repr sub coord)))
```

L'unité sémantique est définie comme une simple structure de traits. Les dérivations sont implémentées comme une structure de traits notant le type et la valeur de la dérivation:

```
(def-linguistic-class sem-unit
  (feature-structure
    ((category cat)
     ;; information de dérivation.
     (drvv (feature-structure
            ((deriv-kind
              (one-of (naction nresult nlieu nagent ninstr adject adjpass
                       adjpotpas adjresact verbe)))
             (deriv-from sem-unit))))
     (drvn (feature-structure
            ((deriv-kind
              (one-of (ncond nlieu ninstr ncollect nperson adjrelat
                       adjqual verbe)))
             (deriv-from sem-unit))))
     (drva (feature-structure
            ((deriv-kind (one-of (nabst nperson verbe)))
             (deriv-from sem-unit))))
     ;; information sur les valences
     (val0 valency)
     (val1 valency)
     (val2 valency)
     (val3 valency)
     ;; autres informations
     (gnr (one-of (masc fem)))
     (nbr (one-of (sg pl)))
     (aux (one-of (être avoir)))
     (reciproque (one-of (arg0-arg1 arg1-arg2)))
     (aspect (one-of (achevé inachevé début fin duratif fréquentatif instantané)
             )))))
```

La valence admet comme valeur un ensemble de valeurs prises parmi les valeurs de valences possibles. Pour cela, on utilise la structure logique *set-of* qui prend comme argument la liste des valeurs possibles :

```
(set-of possible-values keywords*)
```

où l'argument *possible-values* est la liste des classes acceptables pour les éléments de l'ensemble, et où les mots-clés définissent des contraintes de cardinalité sur l'ensemble :

`:min-elements number`	spécifie le nombre minimal d'éléments dans l'ensemble (par défaut : 0).
`:max-elements number`	spécifie le nombre maximal d'éléments dans l'ensemble (par défaut : pas de maximum).

Ainsi, les valences se décrivent comme suit :

```
(def-linguistic-class valency
  (set-of (nom à+nom avec+nom comme+nom contre+nom dans+nom de+nom en+nom
           entre+nom par+nom parmi+nom pour+nom sur+nom inf à+inf de+inf
           adj que+ind que+subj se-moy se-pass lieu-stat lieu-dyn manière
           zéro)))
```

2.1.2. Une structure "à la MULTILEX"

La seconde architecture linguistique que nous souhaitons définir est (librement) inspirée de l'architecture linguistique de MULTILEX.

Les détails de cette architecture sont donnés par la figure 4.6., reprise du chapitre II. Une unité lexicale de MULTILEX (LU) identifie un sens de mot. Elle est reliée à une ou plusieurs GPMU (unité graphique, phonologique et morphologique) qui identifie une forme canonique.

Une LU donne accès à différents types d'informations (sous forme d'ensembles de traits). Ainsi, une LU est reliée à une et une seule unité sémantique, une ou plusieurs unités syntaxiques et zéro ou plusieurs unités de transfert.

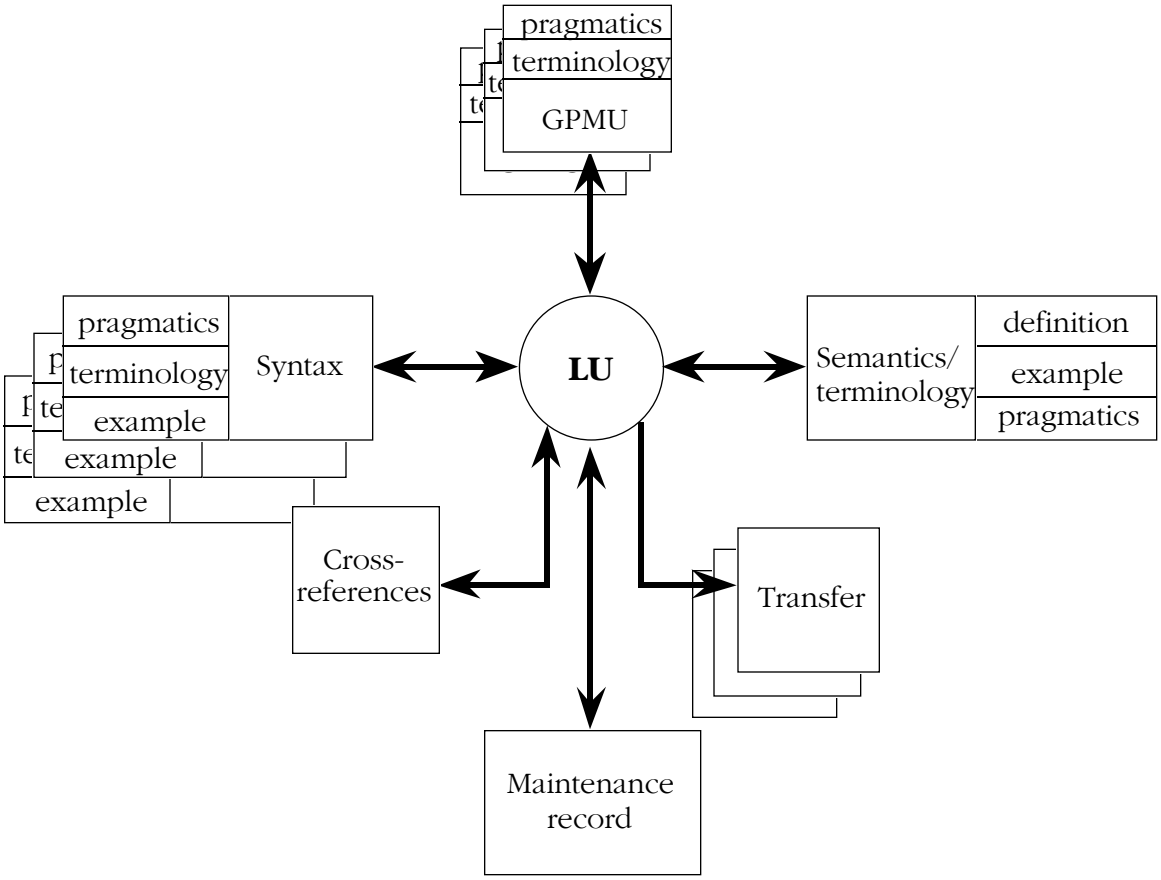


Figure 4.6 : Vue générale d'une LU (Lexical Unit) de MULTILEX

Nous noterons une unité lexicale comme une structure de traits où certains traits ont pour valeur un ensemble de liens reliant les différentes unités (GPMU, syntaxique, sémantiques et de transfert). Nous ne donnerons que la description d'une GPMU, et d'une unité syntaxique. Comme le projet Multilex, nous représenterons les informations associées à ces unités sous forme de structures de traits.

Pour définir une valeur de lien, nous utiliserons la structure logique prédéfinie *link* qui s'utilise comme suit :

```
(link keywords*)
```

où les mots-clés restreignent les classes qui peuvent être valeurs de décoration des différents éléments du lien. Les mots-clés possibles sont :

`:label class`	spécifie la classe acceptable pour la valeur de décoration du lien.
`:source [dict::]class`	spécifie la classe de l'instance de laquelle part le lien. Si le lien vient d'une structure d'un autre dictionnaire, on indique ce dictionnaire.
`:target [dict::]class`	spécifie la classe de l'instance vers laquelle pointe le lien.Si le lien va vers une structure d'un autre dictionnaire, on indique ce dictionnaire.
`:bidirectionnel boolean`	Le système doit-il gérer le lien inverse (qui pointera vers la structure ou est définie le lien d'origine) ?

Pour définir une valeur de type ensemble, nous utiliserons la structure logique *set-of* définie plus haut.

Une unité lexicale est reliée à une et une seule unité sémantique, une ou plusieurs unités de syntaxiques et une ou plusieurs GPMU. On la définit donc comme suit :

```
(def-linguistic-class MLX-UL
  (feature-structure
    ((GPMU        (set-of ((link :target MLX-GPMU
                                 :label preferred?))))
     (syntax      (set-of ((link :target MLX-SynU
                                 :label weight))))
     (semantic    (link :target MLX-SemU))
     (maintenance (link :target MLX-MaintU)))))
```

L'utilisation de liens nous a permis de repérer la GPMU préférée pour une unité lexicale, grâce à la décoration de classe `preferred?` (que l'on définit comme un simple booléen) sur les liens vers les GPMU.

```
(def-linguistic-class preferred? boolean)
```

On peut aussi vouloir associer un poids à l'un des liens. Ici, nous avons associé un poids au lien vers les structures syntaxiques, ce qui nous permet d'associer une probabilité plus forte à une des structures syntaxiques de l'unité lexicale :

```
(def-linguistic-class weight integer)
```

Les autres parties de cette structure seront implémentées comme des structures de traits. Nous ne les détaillerons pas, puisqu'elles ne présentent pas de difficultés.

2.1.3. Une structure "à la GENELEX"

La troisième architecture linguistique que nous souhaitons définir est (librement) inspirée de l'architecture linguistique de GENELEX. Nous ne définirons ici qu'une unité morphologique.

Rappelons qu'une unité morphologique est un regroupement de mots basé sur des propriétés morphologiques. Elle est identifiée par son lemme graphique et/ou par son lemme phonétique. La forme lemmatisée est la forme singulier s'il y a variation en nombre, masculin s'il y a variation en genre, et infinitif pour les verbes.

GENELEX a défini cinq types d'unité morphologiques :

- **UM simple** : une UM simple est associée à une graphie (plusieurs en cas de variantes) constituée d'une suite de caractères alphabétiques, de séparateurs (tiret, apostrophe, point) et de la marque éventuelle d'hyphénation.
- **UM affixe** : une UM affixe peut être de type préfixe, infixe ou suffixe, ou encore sans type dans le cas où elle ne prendrait son statut qu'en contexte de dérivation ou composition.
- **UM dérivée** : une UM dérivée est une unité morphologique simple qui entretient des liens de dérivation avec d'autres unités morphologiques (simples ou affixes).
- **UM composée** : l'UM composée de Genelex est une expression complexe qu'un lexicographe a choisi de coder dans la couche morphologique.
- **UM agglutinée** : permet d'enregistrer des phénomènes de contraction graphique de deux unités. Par exemple : *du* (= de + le).

L'ensemble des unités morphologiques (UM) a une structure analogue. À cette structure globale s'ajoutent d'autres liens dépendant de la nature de l'unité morphologique. Cela nous permettra d'introduire une notion fondamentale du langage de définition de l'architecture linguistique : la notion d'héritage.

Grâce à cette notion, il nous est possible de définir une classe linguistique correspondant à l'ensemble des UM. Nous pourrons ensuite définir les unités morphologiques particulières comme des sous-classes linguistiques de la classe UM. Ces structures hériteront donc de la structure générale des UM en lui rajoutant des éléments propres.

Cette notion d'héritage est rendue complexe par la diversité des structures logiques de base qui peuvent être utilisées dans les structures linguistiques. Aussi, nous détaillerons les

modalité de la notion d'héritage structure logique par structure logique dans le paragraphe suivant.

Une unité morphologique GENELEX a la structure illustrée par la figure 4.7.

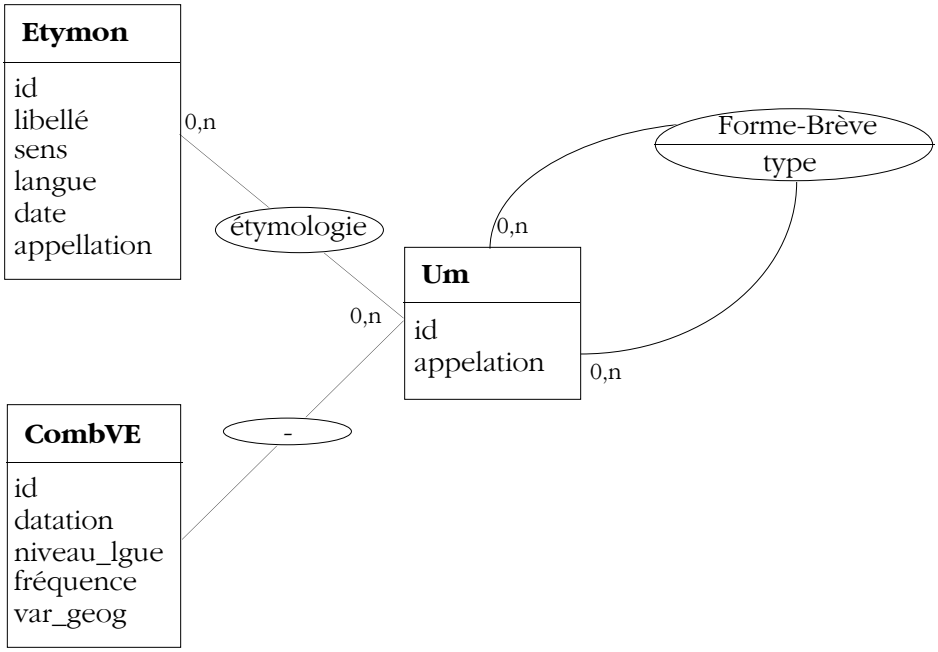


Figure 4.7 : Structure commune aux unités morphologiques de GENELEX

Nous la définirons comme une structure de traits dont certains traits ont une valeur qui est un ensemble de liens :

```
(def-linguistic-class UM
  (feature-structure
    ((formes-brèves (set-of ((link :target UM
                                   :label type-forme-brève))))
     (étymologie    (set-of ((link :target étymon))))
     (combVE        (link :target combVE))
     (appellation   string)
    )))
(def-linguistic-class type-forme-brève
  (one-of (abbéviation sigle accronyme)))
```

Un étymon est lui aussi une structure de traits. Ses traits acceptent une chaîne de caractères en valeur.

```
(def-linguistic-class étymon
  (feature-structure
    ((langue     string)
     (sens       string)
     (date       string)
     (appelation string))
  ))
```

De la même manière, le CombVE est défini comme suit :

```
(def-linguistic-class CombVE
  (feature-structure
    ((datation    (one-of (archaique vielli moderne)))
     (niveau_lgue (one-of (familier vulgaire argotique populaire litteraire
                           savant standard)))
     (fréquence   (one-of (rare courant)))
     (var_geog    string))))
```

Une fois qu'une unité morphologique est définie, nous allons définir les unités morphologiques particulières. Nous ne développerons que la définition de l'UM simple. Nous en donnons la structure graphique dans la figure 4.8.

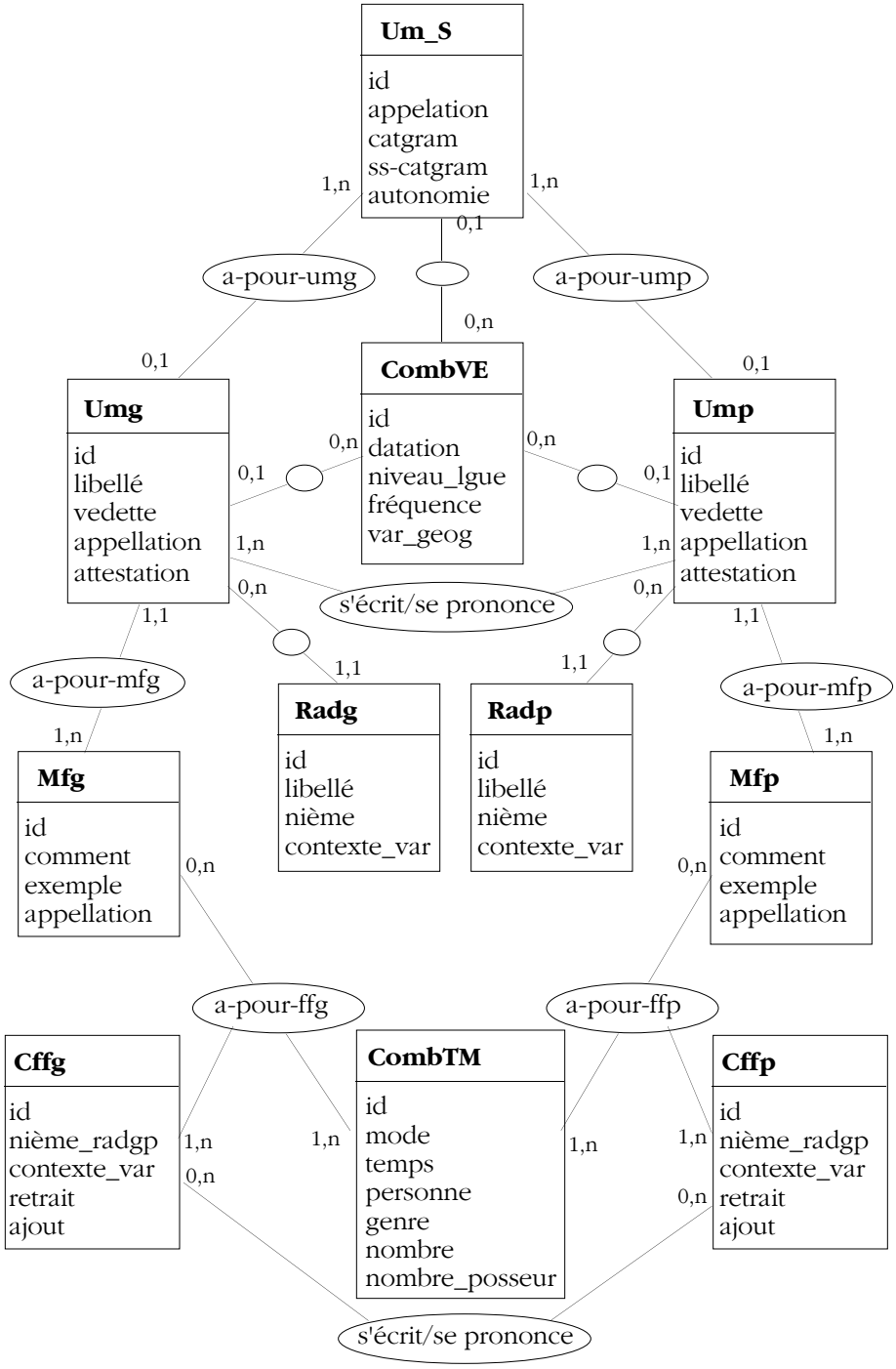


Figure 4.8 : Structure de l'UM simple

Une UM simple hérite de la structure d'une UM générale et rajoute des éléments particuliers. Pour dénoter de cet héritage, nous utilisons la syntaxe :

```
(parent-linguistic-class arguments*)
```

où *parent-linguistic-class* est le nom de la structure linguistique dont on veut hériter. Les *arguments* dépendent de la structure linguistique spécifiée. Le détail de ces arguments et la sémantique exacte de l'héritage seront exposés dans le paragraphe suivant.

Dans le cas d'une structure de traits, on hérite des traits déjà définis, que l'on rajoute aux traits spécifiés sur la sous-classe. Lorsqu'un trait de la sous-classe existe déjà sur la classe dont on hérite, deux cas se présentent :

- **héritage simple** : dans ce cas, la valeur spécifiée dans la classe dont on hérite est écrasée. Seule compte donc la valeur donnée dans la sous-classe.

- **héritage par unification** : dans ce cas, on fait l'unification des deux valeurs données dans la classe dont on hérite et dans la sous-classe que l'on définit. Si l'unification échoue, l'héritage simple est adopté. Si elle réussit, le trait litigieux prend pour valeur le résultat de l'unification. Cette opération d'unification sera détaillée dans le paragraphe suivant.

Le linguiste peut spécifier le type d'héritage désiré en utilisant le mot-clé *inheritance-type*, que l'on retrouve dans les *arguments* quelle que soit la structure de base. Ce mot-clé prend pour valeur *simple* ou *unification*. Le comportement par défaut est l'héritage simple.

Ainsi, la définition de l'UM simple de GENELEX est la suivante :

```
(def-linguistic-class UM_S
  (UM
    ((catgram    (one-of (nom adjectif adverbe verbe preposition conjonction
                          interjection determinant pronom particule)))
     (ss-catgram (one-of (propre commun possessif demonstratif partitif
                          defini indefini interrogatif cardinal ordinal
                          relatif personnel_fort personnel_faible impersonnel
                          exclamatif qualitatif coordination subordination
                          completif)))
     (autonomie  (one-of (oui non)))
     (usyn-l     (set-of (USyn)))
     (a-pour-Umg (set-of (Umg) :min-elements 1))
     (a-pour-Ump (set-of (Ump) :min-elements 1)))
  ))
```

Cette définition utilise les notions d'unités morphologiques graphiques (Umg, qui portent les informations sur l'écrit) et d'unités morphologique phonémiques (Ump, qui portent des informations sur l'oral).

```
(def-linguistic-class Umg
  (feature-structure
    ((vedette     (one-of (oui non)))
     (appellation string)
     (attestation string)
     (combVe      combVe)
     (a-pour-mfg  Mfg)
     (a-pour-radg (set-of (Radg)))
     (se-prononce (set-of (Ump) :min-elements 1))))
  ))
(def-linguistic-class Ump
  (feature-structure
    ((vedette     (one-of (oui non)))
     (appellation string)
     (attestation string)
     (combVe      combVe)
     (a-pour-mfg  Mfp)
     (a-pour-radp (set-of (Radp)))
     (s-écrit     (set-of (Umg) :min-elements 1))))
  ))
```

Ces structures utilisent un ensemble de radicaux graphiques et phonémiques (Radg, Radp) donnant l'ensembles des bases sur lesquelles se fondent les règles morphologiques.

```
(def-linguistic-class Radg
  (feature-structure
    ((nième        integer)
     (contexte_var string))))
(def-linguistic-class Radp
  (feature-structure
    ((nième        integer)
     (contexte_var string))
  ))
```

Les unités morphologiques sont associées à des informations morphologiques graphiques et phonémiques (Mfg, Mfp).

```
(def-linguistic-class Mfg
  (feature-structure
    ((comment     string)
     (exemple     string)
     (appellation string)
     (a-pour-ffg  (set-of ((link :target Cffg
                                 :label  CombTM)))))))
(def-linguistic-class Mfp
  (feature-structure
    ((comment     string)
     (exemple     string)
     (appellation string)
     (a-pour-ffg  (set-of ((link :target Cffp
                                 :label  CombTM)))))))
```

Ces informations de morphologiques utilisent des règles de calcul des formes fléchies pour les unités graphiques et phonémiques (Cffg, Cffp).

```
(def-linguistic-class Cffg
  (feature-structure
    ((nième-radgp  integer)
     (contexte_var string)
     (retrait      string)
     (ajout        string)
     (se-prononce  (set-of (Cffp)))))
(def-linguistic-class Cffp
  (feature-structure
    ((nième-radgp  integer)
     (contexte_var string)
     (retrait      string)
     (ajout        string)
     (se-prononce  (set-of (Cffg)))))
```

Chaque règle de calcul est associée à une combinaison temps/mode (combTM) identifiant la forme fléchie obtenue.

```
(def-linguistic-class CombTM
  (feature-structure
    ((mode           (one-of (indicatif subjonctif conditionnel imperatif
                              infinitif participe)))
     (temps          (one-of (present imparfait passe-simple futur passe)))
     (personne       (one-of (1 2 3)))
     (genre          (one-of (masculin feminin neutre)))
     (nombre         (one-of (singulier pluriel))
     (nombre-posseur (one-of (singulier-posseur pluriel-posseur))))))
```

On a ainsi défini une UM simple selon GENELEX.

2.1.4. Une structure complexe : le DEC de Mel'čuk

Jusqu'à présent, nous avons montré comment déclarer en SUBLIM des bases lexicales existantes. Nous avons pu constater que même si les personnes développant ces bases disent qu'elles ne font appel qu'à une structure de base (graphe ou structure de traits), il est utile de disposer de plusieurs autres structures pour les implémenter (notion d'ensemble, etc.).

L'exemple que nous allons étudier maintenant n'est pas une base lexicale à usage machinal. Il s'agit d'un dictionnaire (essentiellement papier) dont les informations sont assez complexes. Ce dictionnaire a été développé par Igor Mel'čuk et ses collègues, à Moscou, puis à Montréal.

Quelques exemples d'articles de ce dictionnaire sont donnés en Annexe C[3].

Une unité de ce dictionnaire est un sens de mot ou de locution (un sémantème). Cette unité lexicale est associée à une unité morphologique, à une définition, à d'éventuelles

---

[3] Je tiens à remercier Igor Mel'čuk qui me les a très gentiment communiquées.

connotations, à un régime, à des exemples, et à des fonctions lexico-sémantiques. Nous lui affectons de plus un numéro de sens qui l'identifie parmi les différents sens d'une entrée.

Un sémantème peut aisément être codé comme une structure de traits :

```
(def-linguistic-class sémantème
  (feature-structure
    ((numéro         numéro)
     (UMorph         UMorph)
     (définition     définition)
     (connotations   connotations)
     (régime         régime)
     (exemples       exemples)
     (lexico-sem-fns lex-sem-fns))
  ))
```

Une unité morphologique comprend une forme graphique et des informations morphologiques. Elle peut être reliée à plusieurs sémantèmes. Ces différents sémantèmes lui sont associés de manière arborescente :

CŒUR, nom, masc.

I.1a.	Organe principal de la circulation sanguine d'une personne… [*le cœur de Jean*]
1b.	Organe principal de la circulation sanguine d'un animal… [*le cœur de lion*]
2.	Produit alimentaire … [*le cœur de veau*]
3.	Partie de la poitrine d'une personne … [*Il a serré son fils sur son cœur*]
4a.	Organe imaginaire des sentiments … [*Le cœur espère toujours*]
4b.	Organe imaginaire de l'intuition … [*Son cœur le lui dit*]
5a.	… propriété de la personnalité … [*un cœur de glace*]
5b.	Personne possédant le cœur I.5a [*Vous devez la vie à un noble cœur, à un homme vaillant*]
II.1a.	Partie principale d'une unité fonctionnelle… [*le cœur du bateau*]
1b.	Élément principal [*le cœur du problème*]
2a.	Partie centrale d'un espace… [*le cœur du royaume*]
3.	Objet… ayant la forme du cœur I.1a [*un cœur en papier*]
4.	Une des quatre couleurs 2 des cartes à jouer… [*l'as de cœur*]
III.	Organe imaginaire des nausées … [*Cette senteur lui tournait le cœur*]

Aussi, nous définirons une unité morphologique comme un arbre portant des informations morphologiques à la racine et des sémantèmes sur les feuilles.

```
(def-linguistic-class UMorph
  (tree :root   Morphological-information
        :leaves sémantème))
```

L'information morphologique associée à la racine de cette arbre ne comporte qu'une graphie, une catégorie, un genre et un nombre.

```
(def-linguistic-class Morphological-information
  (feature-structure
     ((graph     string)
      (catégorie cat)
      (genre     gnr)
      (nombre    nbr))))

(def-linguistic-class cat
  (one-of (nom verbe adjectif adverbe)))
(def-linguistic-class gnr
  (one-of (masculin féminin)))
(def-linguistic-class nbr
  (one-of (singulier pluriel)))
```

Une définition du DEC n'est pas une simple chaîne de caractères :

I.1a. *Cœur de X* = Organe principal de la circulation sanguine d'une personne X qui se trouve dans la partie centrale du corps II.1d de X et qu'on représente symboliquement comme ayant la forme ♡ .

Mis à part le fait que l'on y trouve une image, on peut remarquer qu'elle se compose de deux parties principales. La première (indiquée en italiques) présente un usage du sémantème dans

une locution où les différents arguments du prédicat représenté sont indiqués sous forme de variable. La seconde est une explicitation du sens du sémantème. Cette explication réutilise les variables de la première partie. On remarque aussi qu'elle fait référence à des sémantèmes définis par ailleurs dans le dictionnaire (corps II.1d).

Nous simplifierons cette structure en la décomposant simplement en deux chaînes de caractères, l'une contenant la forme du prédicat, l'autre contenant sa définition :

```
(def-linguistic-class définition
  (feature-structure
    ((prédicat  string)
     (explicite string))))
```

Après cette partie de définition, on trouve éventuellement une partie consacrée aux connotations :

Connotations

1) Cœur I.1a est le siège des sentiments [voir CŒUR I.4a].
2) Cœur I.1a est le siège de l'intuition [voir CŒUR I.4b].
3) Cœur I.1a qui bat 1 représente la vie [voir les phrasèmes correspondants dans CŒUR I.1a].

Cette partie se présente comme une liste de connotations. Chacune est donnée sous forme de chaîne de caratères faisant référence à au moins un sémantème. Il est donc intéressant, dans une version informatique de ce dictionnaire, de conserver à la fois la connotation sous forme de chaîne de caractères et sous forme d'un ensemble de liens vers d'autres sémantèmes :

```
(def-linguistic-class connotations
  (set-of connotation))
(def-linguistic-class connotation
  (feature-structure
    ((texte    string)
     (réfère-à (set-of ((link :target sémantème)))))))
```

À la suite de ces éventuelles connotations, on trouve le régime du prédicat. Ce régime donne les informations sur les différentes réalisations syntaxiques des arguments du prédicat. Le régime est donné sous forme de tableau dont les colonnes correspondent aux arguments et les lignes aux différentes réalisations, certaines combinaisons ainsi établies étant non valides, on en reprend ensuite la liste, en indiquant leur impossibilité. On reprend aussi un certain nombre de ces combinaisons pour en donner des exemples (l'exemple suivant est tiré de enseigner 1) :

1. *X enseigne Y à Z* = X, censé avoir la qualification professionnelle dans le domaine Y, cause que Z apprenne III.1b Y en transmettant, méthodiquement et dans un cadre officiel, à Z des connaissances (portant sur) Y ou des techniques (portant sur) Y [$CausConv_{21}$(*apprendre III.1b*)].

Régime

1 = X	2 = Y	3 = Z
1. N	1. N 2. *à* V_{inf}	1. *à* N 2. **rare** N

1) $C_{2.2}$ sans $C_{3.1}$
2) $C_2 + C_{3.2}$
: impossible

$C_1 + C_2$: *Pierre enseigne la grammaire la couture à faire cela*

$C_1 + C_2 + C_3$: *Pierre enseigne la grammaire à ses élèves*

La structure correspondant à cette partie est beaucoup plus compliquée que celle des parties précédentes. En effet, cette présentation n'est que le reflet, imprimable, d'une structure

complexe où l'on retrouve l'ensemble des combinaisons possibles de réalisations d'arguments. On peut donc représenter cette partie de deux manières :

- en restant proche de sa forme papier. On a alors un tableau et une liste des combinaisons impossibles.
- en représentant cette structure de manière plus abstraite. On peut ainsi la représenter par un automate dont chaque chemin forme une combinaison valide.

Si l'on choisit la seconde solution, le régime donné en exemple sera donc représenté par l'automate donné en figure 4.9.

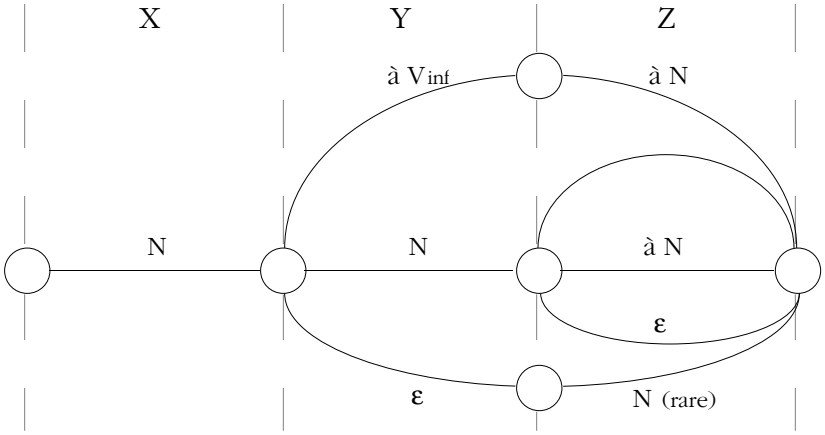


Figure 4.9 : Régime d'enseigner 1, sous forme d'automate

Pour exprimer cette solution, nous utiliserons la structure logique d'automate définie comme suit :

```
(automaton keywords*)
```

où les mots-clés définissent des contraintes sur les classes acceptables en décoration des différents éléments de l'automate :

`:arcs class`	spécifie la classe acceptable en décoration des arcs.
`:nodes class`	spécifie la classe acceptable en décoration de l'ensemble des nœuds.
`:starting-node class`	spécifie la classe acceptable en décoration du nœud d'entrée.
`:ending-nodes class`	spécifie la classe acceptable en décoration des nœuds de sortie.

Ainsi, cette structure s'exprimera sous forme d'une structure de traits dont l'un comportera l'automate, un autre donnera l'ordre dans lequel les arguments apparaissent dans le régime et un troisième donnera l'ensemble des exemples :

```
(def-linguistic-class régime
  (feature-structure
    ((automate       automate-régime)
     (argument-order (list-of (string)))
     (exemples       exemples-régime))))
(def-linguistic-class automate-régime
  (automaton :arcs réalisation-argument))
(def-linguistic-class exemples-régime
  (set-of ((feature-structure
             ((réalisations (list-of (string)))
              (exemple      string))))))
```

La partie la plus importante de ce dictionnaire réside dans l'ensemble des fonctions lexicales du sémantème. Leur meilleur définition est donnée, en première partie du DEC, par l'auteur, Igor Mel'čuk :

Les fonctions lexicales (FL) présentent l'ensemble de la cooccurrence lexicale restreinte intéressant le lexème considéré. Elles constituent une innovation lexicographique qui permet de décrire d'une façon

systématique un vaste ensemble de locutions plus ou moins figées qui ne sont quand même pas des expressions idiomatiques *stricto sensu.* Il s'agit, par exemple, des locutions comme une FERME intention, une résistance ACHARNÉE, un argument DE POIDS, un bruit INFERNAL, un désir ARDENT, une envie FOLLE, une règle STRICTE, une vérité INCONTESTABLE, où des adjectifs bien spécifiques doivent être employés avec les différents noms pour exprimer la même idée d'intensification. Comme autre exemple de locution de ce type, on peut citer les expressions DONNER une leçon, FAIRE un pas, COMMETTRE un crime, PORTER une accusation, etc., où des verbes sémantiquement vides (ou presque vides) différents doivent être choisis en fonction du nom d'action pour lier le nom d'agent en tant que sujet grammatical au nom d'action en tant que complément d'objet direct.

L'écriture générale d'une FL est de la forme : **f**(X) = Y, où **f** est la FL, X est son argument (un lexème ou bien une locution), et Y est la valeur de la FL **f** pour cet argument, c'est à dire l'ensemble des expressions linguistiques qui peuvent exprimer le sens ou le rôle syntaxique donné (noté par **f**) auprès de X.

Comme ce dictionnaire est imprimé, les expressions linguistiques sont données sous une forme linéaire :
MÉPRIS, nom, masc.

I. Attitude émotionnelle défavorable… [*le mépris pour ce corrupteur*]

[…]

Fonctions lexicales

$Caus_3Func_1$: engendrer [ART [illegible] N] [*La familiarité engendre le mépris*]

$Caus_{(3)}Func_1$: apprendre, inculquer [à N] [*Il inculque à ses étudiants le mépris de l'hypocrisie ; Son attitude envers ses employés apprend à ces derniers le mépris de leur chef*]

$Caus_{(2/3)}Func_1$: inspirer [ART à N] [*Cet événement inspire aux travailleurs le mépris de leur patron ; L'argent inspirait à ce philosophe un tel mépris qu'il a donné son héritage à son frère ; L'hypocrisie de Jean leur inspirait un profond mépris*]

Mais la structure interne de ces expression linguistique est un arbre syntaxique donnant la construction de cette expressions linguistiques et de l'argument X pour réaliser la fonction **f**. Ainsi, la structure interne de $Caus_3Func_1$(Mépris I) est l'arbre donné en figure 4.10.:

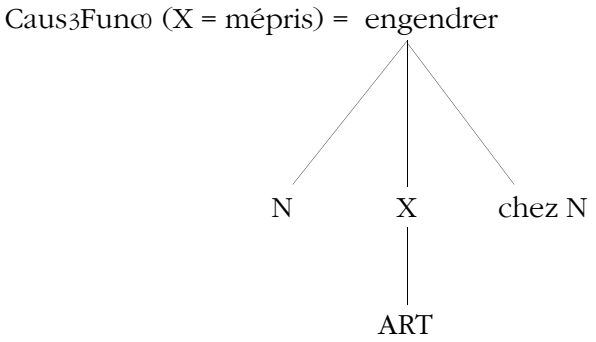


Figure 4.10 : Structure interne d'une expression linguistique, valeur de fonction lexicale

Une fonction lexicale représente donc un lien entre un sémantème et une expression linguistique complexe comportant d'autres sémantèmes. Aussi, la valeur de ces FL peut être représentée comme un ensemble d'arbres dont certains nœuds sont des variable, et d'autres sont des sémantèmes.

Il faut aussi représenter les fonctions lexicales. En effet, s'il y a un nombre limité de fonctions lexicales de base, on trouvera des fonctions composées dans les différents articles de dictionnaire.

Prenons un exemple : les fonctions **Oper$_1$**, **Oper$_2$**… ont pour valeur les verbes sémantiquement vides qui prennent le nom du premier, deuxième… actant comme sujet grammatical et C_0 (leur argument) comme complément d'objet principal :

Oper$_1$(*attention*) = *faire*
Oper$_2$(*attention*) = *attirer*
Oper$_1$(*conseil*) = *donner*
Oper$_2$(*conseil*) = *recevoir*
Oper$_1$(*aide*) = *prêter, accorder*
Oper$_2$(*aide*) = *recevoir*

La fonction Caus représente la notion : "faire en sorte que quelque chose ait lieu". Elle s'emploie le plus souvent en combinaison avec d'autres FL. Ainsi, si **Oper$_1$**(*désespoir*) = *éprouver, ressentir, avoir*, **CausOper$_1$**(*désespoir*) représente "faire en sorte que quelqu'un éprouve du désespoir". Donc **CausOper$_1$**(*désespoir*) = *pousser, réduire* [*qqn au désespoir*], *jeter* [*qqn dans le désespoir*], *frapper* [*qqn de désespoir*].

Il n'est donc pas possible de représenter chaque fonction lexicale comme un attribut dand une structure, puisque la possibilité de composition entraîne toute une combinatoire des fonctions lexicales. Nous les représenterons donc par la structure logique de base *function*. Cette structure logique s'exprime de la manière suivante :

```
(function keywords*)
```

où les mots-clés définissent des contraintes sur les classes acceptables pour les différents éléments de fonction :

`:label class`	spécifie la classe identifiant la fonction.
`:arguments class`	spécifie la liste des classes acceptable pour les arguments de la fonction.
`:value class`	spécifie la classe acceptable en valeur de la fonction.

Ainsi, la structure correspondant aux fonctions lexicales peut s'exprimer comme suit :

```
(def-linguistic-class lex-sem-fns
  (set-of (lex-sem-fn)))
(def-linguistic-class lex-sem-fn
  (function :label nom-FL
            :arguments (FL-arg)
            :value expression-linguistique))
```

Pour représenter la composition de fonctions, on peut autoriser l'utilisation d'une fonction lexicale en argument d'une fonction lexicale. Néanmoins, la valeur de la FL argument (si elle existe) n'est pas pertinente. Seule l'étiquette des fonctions composées est porteuse d'information. Aussi, le plus simple est d'autoriser une valeur complexe en label de la fonction. Nous définirons donc un label de fonction comme une liste (ordonnée) de noms de fonctions de base.

```
(def-linguistic-class nom-FL
  (list-of (nom-FL-base)))
```

Le nom d'une fonction de base est donné par un identificateur de la fonction (une chaîne de caractères) et par le numéro de l'actant sur lequel elle opère :

```
(def-linguistic-class nom-FL-base
  (feature-structure
    ((fonction string)
     (actant   integer))))
```

L'argument de la fonction est un sémantème. Le fait d'indiquer cet argument est redondant puisque cette fonction est définie à l'intérieur dans la structure même du sémantème.

```
(def-linguistic-class FL-arg sémantème)
```

L'expression linguistique valeur de la fonction est représentée sous forme d'arbre (comme nous l'avons indiqué plus haut). Les nœuds de cet arbre sont soit des sémantèmes, soit des variables. Pour simplifier, nous les noterons comme des chaînes de caractères.

```
(def-linguistic-class expression-linguistique
  (tree :nodes (one-of (sémantème string))))
```

La définition de la structure linguistique du DEC, même simplifiée, illustre parfaitement le besoin ressenti par les linguistes a de pouvoir mélanger différentes structures logiques dans une seule et même structure linguistique. Le fait de proposer différentes structures logiques permet au linguiste de manipuler des concepts proches de ceux utilisés dans sa théorie. Cela permet de simplifier le travail du linguiste en lui permettant de rester à un niveau d'abstraction très utile lorsqu'il souhaite implémenter une théorie complexe.

2.2. Le langage de définition de l'architecture linguistique : LINGARD

Dans la section précédente, nous avons donné des exemples d'utilisation des différentes structures logiques de base connues du système SUBLIM. Nous avons donné, de manière assez informelle, le moyen d'utiliser chacune de ces structures.

Nous exposons maintenant les principes de base du langage de définition de l'architecture linguistique.

2.2.1. Principes de base

La définition de l'architecture linguistique d'un dictionnaire se fait dans un fichier dont le nom est le nom du dictionnaire (tel qu'il est défini dans l'architecture lexicale) suivi de l'extension ".LING" : `<nom de dictionnaire>.LING`.

Le langage LINGARD a une syntaxe "noyau" écrite en LISP. Avec ce langage, il est possible de définir des structures linguistiques, à partir de structures logiques existantes. De plus, il est possible de nommer une structure linguistique particulière.

Ces structures linguistiques sont analogues à des *classes* (selon la terminologie des langages à objets). Un article de dictionnaire regroupe différentes *instances* de ces classes. Comme dans les langages à objets, il est possible d'hériter des classes déjà définies.

2.2.1.1. Nommage

Il est possible de donner un nom à une classe que l'on définit. Pour cela, on utilise la clause :

```
define-linguistic-class name class-definition
```

où *name* est un symbole définissant de manière unique la classe définie à l'intérieur du dictionnaire. *class-definition* est une clause LISP définissant une classe linguistique (voir paragraphe suivant).

Cette expression associe au symbole *name* la classe linguistique renvoyée par *class-definition*. Elle retourne la classe linguistique renvoyée par *class-definition.*

2.2.1.2. Définition

La définition d'une classe linguistique dépend de la structure logique que l'on souhaite utiliser. Néanmoins, cette définition se fait selon un schéma fixe :

```
(logical-structure arguments*)
```

où *logical-structure* représente la structure logique dont on se sert pour définir la classe linguistique. Les *arguments* dépendent de la structure logique utilisée. Ces arguments sont détaillés plus bas, structure logique par structure logique.

Certaines structures (fixes) sont associées à des symboles prédéfinis dans le système (boolean, integer…). Ces symboles peuvent être utilisés directement dans la définition d'une classe linguistique.

2.2.1.3. Héritage

LINGARD permet d'hériter du comportement d'une classe linguistique que l'on a définie auparavant. Bien que la sémantique de cet héritage dépende de la structure logique considérée, il est exprimé selon le schéma fixe :

```
(parent-linguistic-class arguments*)
```

où *parent-linguistic-class* représente la classe linguistique dont on hérite. Les arguments permettent de redéfinir les différents éléments de la classe linguistique mère selon deux schémas possibles :

- **héritage simple** : dans ce cas, les valeurs spécifiées dans la nouvelle classe remplacent les valeurs spécifiées pour la classe mère.
- **héritage par unification** : dans ce cas, on fait l'unification des deux valeurs données dans la classe dont on hérite et dans la sous-classe que l'on définit. Si l'unification échoue, l'héritage simple est adopté. Si elle réussit, le trait en cause prend pour valeur le résultat de l'unification. Cette opération d'unification (qui dépend elle aussi des structures logiques utilisées) sera détaillée dans les paragraphes suivants.

Le linguiste peut spécifier le type d'héritage désiré en utilisant le mot-clé *inheritance-type*, que l'on retrouve dans les *arguments* quelle que soit la structure de base. Ce mot-clé prend pour valeur *simple* ou *unification*. Le comportement par défaut est l'héritage *simple*.

Dans la suite, nous définirons cette opération d'unification en fonction de l'opération U', dite *unification faible* et définie comme suit :

$$U'(X, Y) = \text{si } U(X, Y) \neq \perp \begin{cases} \text{alors } U(X, Y) \\ \text{sinon } Y \end{cases}$$

U(X, Y) étant l'unification de X et Y.

2.2.1.4. Unification

Le mécanisme d'unification est utile sur les structures logiques, pour définir la sémantique de l'héritage. Notons que cette unification porte sur des classes de structures, et non sur des instances particulières de ces classes.

Afin de définir de manière cohérente cette opération sur l'ensemble des structures logiques, nous introduisons les notions suivantes :

L'ensemble des structures logiques et des classes linguistiques forme un ensemble nommé *ensemble des structures*, et noté Σ.

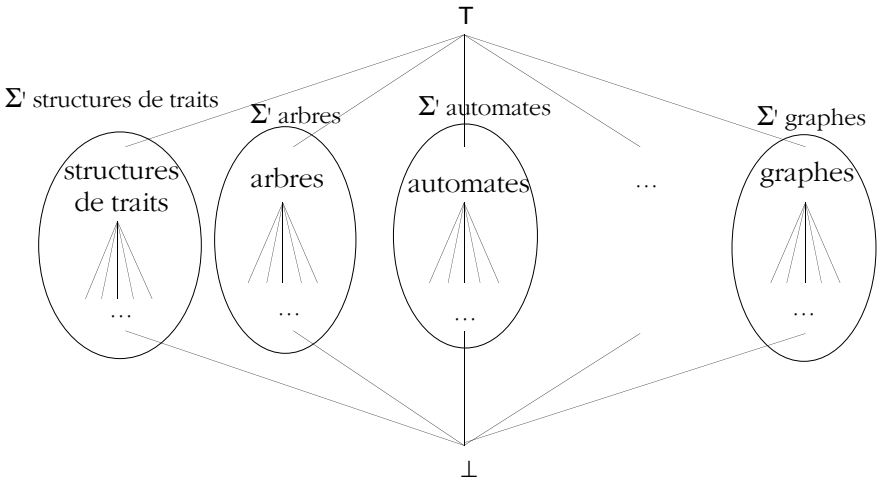


Figure 4.11 : Vue globale du treillis (Σ,<<)

Un ordre partiel est défini sur Σ et noté <<. Les structures logiques de base sont incomparables selon cet ordre. (Σ, <<) définit un treillis ayant l'aspect donné en figure 4.11.

Si la classe linguistique X est définie en fonction de la structure logique x, alors X se trouvera dans le sous-treillis Σ'_x. Si la classe linguistique Y est définie en fonction de X, figurant dans le sous treillis Σ'_x, alors Y figurera dans le sous-treillis Σ'_x. Deux classes linguistiques quelconques figurant dans deux sous-treillis différents sont donc incomparables par <<.

L'opération d'unification (notée U) est définie sur $\Sigma'_x \times \Sigma'_x$ pour toute structure logique x. Elle échoue (retourne ⊥) sur $\Sigma'_x \times \Sigma'_y$ ($x \neq y$).

Le symbole spécial T, situé au sommet du treillis (Σ,<<), est l'élément neutre de l'opération d'unification.

2.2.2. Arbres

La structure linguistique et informatique la plus classique est sans conteste l'arbre. Il est donc normal de proposer cette structure en premier. La structure définie ici représente la classe des arbres décorés.

2.2.2.1. Définition de la structure

La définition d'une classe linguistique ayant une structure d'arbre se fait de la manière suivante :

```
(tree keywords*)
```

où les mots-clés restreignent les classes qui peuvent être valeurs de décoration des différentes parties de l'arbre. Les mots-clés possibles sont :

`:root class`	spécifie la classe acceptable pour les valeurs des décorations de la racine de l'arbre.
`:leaves class`	spécifie la classe acceptable pour les valeurs des décorations des feuilles de l'arbre.
`:nodes class`	spécifie la classe acceptable pour les valeurs des décorations de l'ensemble des nœuds de l'arbre (racine et feuilles comprises si elles ne sont pas définies par ailleurs).

class est une classe linguistique quelconque. Si les mots-clés *root* et *leaves* ne sont pas définis, ils prennent la valeur associée à *:nodes*.

2.2.2.2. Héritage

Si parent-class est une classe linguistique ayant une structure d'arbre, on peut en hériter en définissant une nouvelle classe linguistique par :

```
(parent-class keywords*)
```

où les mots-clés redéfinissent les classes qui peuvent être valeurs de décoration des différentes parties de parent-class. Les mots-clés sont les mêmes que ceux utilisés dans la définition d'un arbre. On peut aussi spécifier le mot-clé *inheritance-type*, qui régit la sémantique de l'héritage :

- si *inheritance-type* a la valeur *simple :* pour chaque mot-clé, les valeurs spécifiées dans la définition de la sous-classe linguistique remplacent les valeurs de la classe mère, spécifiées pour le même mot-clé. Par exemple, si X est défini de la manière suivante :
  ```
  (define-linguistic-class X (tree :root Y :leaves Z :nodes W))
  ```
  et si X' est défini de la manière suivante :
  ```
  (define-linguistic-class X' (X :root A))
  ```
  alors X' correspond à la structure :
  ```
  (tree :root A :leaves Z :nodes W)
  ```

- si *inheritance-type* a la valeur *unification :* pour chaque mot-clé, les valeurs spécifiées dans la définition de la sous-classe linguistique s'unifient avec les valeurs de la classe mère, spécifiées pour le même mot-clé. Par exemple, si X est défini de la manière suivante :

```
(define-linguistic-class X (tree :root Y :leaves Z :nodes W))
```

et si X' est défini de la manière suivante :

```
(define linguistic-class X' (X :root A))
```

alors X' correspond à la structure :

```
(tree :root U'(Y,A) :leaves Z :nodes W)
```

2.2.2.3. Unification

L'unification sur $\Sigma'_{arbres} \times \Sigma'_{arbres}$ est définie de la manière suivante :

U: $\Sigma'_{arbres} \times \Sigma'_{arbres} \rightarrow \Sigma'_{arbres}$
$(T_1, T_2) \rightarrow T$

Si T_1est l'arbre dont les valeurs de décoration acceptables sont :
en racine : R_1
en feuilles : F_1
en nœuds : N_1.

et si T_2 est l'arbre dont les valeurs de décoration acceptables sont :
en racine : R_2
en feuilles : F_2
en nœuds : N_2.

alors T est l'arbre dont les valeurs de décoration acceptables sont :
en racine : $U'(R_1,R_2)$
en feuilles : $U'(F_1,F_2)$
en nœuds : $U'(N_1,N_2)$.

Lorsqu'une valeur de décoration acceptable n'est pas définie par le linguiste, elle a la valeur $\top$.

2.2.3. Graphes

La seconde structure que nous proposons est largement utilisée dans le domaine du Traitement Automatique des Langues Naturelles. Bien souvent, on interprète une structure en terme de graphes (les structures de traits avec réentrance notamment). Il est donc naturel de proposer cette structure de graphe parmi les structures logiques de base. Les graphes implémentés ici portent des décorations quelconques sur les nœuds et les arcs.

2.2.3.1. Définition de la structure

La définition d'une classe linguistique ayant une structure de graphe se fait de la manière suivante :

```
(graph keywords*)
```

où les mots-clés restreignent les classes qui peuvent être valeurs de décoration des différentes parties du graphe. Les mots-clés possibles sont :

`:nodes class`	spécifie la classe acceptable pour les valeurs des décorations des nœuds du graphe.
`:arcs class`	spécifie la classe acceptable pour les valeurs des décorations des arcs du graphe.

class est une classe linguistique quelconque.

2.2.3.2. Héritage

Si parent-class est une classe linguistique ayant une structure de graphe, on peut en hériter en définissant une nouvelle classe linguistique par :

```
(parent-class keywords*)
```

où les mots-clés redéfinissent les classes qui peuvent être valeurs de décoration des différentes parties de parent-class. Les mots-clés sont les mêmes que ceux utilisés dans la définition d'un graphe. On peut aussi spécifier le mot-clé *inheritance-type*, qui régit la sémantique de l'héritage :

- si *inheritance-type* a la valeur *simple :* pour chaque mot-clé, les valeurs spécifiées dans la définition de la sous-classe linguistique remplacent les valeurs de la classe mère, spécifiées pour le même mot-clé. Par exemple, si X est défini de la manière suivante :
  ```
  (define-linguistic-class X (graph :nodes Y :arcs Z))
  ```
  et si X' est défini de la manière suivante :
  ```
  (define-linguistic-class X' (X :nodes A))
  ```
  alors X' correspond à la structure :
  ```
  (graph :nodes A :arcs Z)
  ```
- si *inheritance-type* a la valeur *unification :* pour chaque mot-clé, les valeurs spécifiées dans la définition de la sous-classe linguistique s'unifient avec les valeurs de la classe mère, spécifiées pour le même mot-clé. Par exemple, si X est défini de la manière suivante :
  ```
  (define-linguistic-class X (graph :nodes Y :arcs Z))
  ```
  et si X' est défini de la manière suivante :
  ```
  (define linguistic-class X' (X :nodes A))
  ```
  alors X' correspond à la structure :
  ```
  (graph :nodes U'(X,A) :arcs Z)
  ```

2.2.3.3. Unification

L'unification sur $\Sigma'_{graphes} \times \Sigma'_{graphes}$ est définie de la manière suivante :

U: $\Sigma'_{graphes} \times \Sigma'_{graphes} \rightarrow \Sigma'_{graphes}$
$(G_1, G_2) \rightarrow G$

Si G_1est le graphe dont les valeurs de décoration acceptables sont :
en arcs: A_1
en nœuds : N_1.

et si T_2 est l'arbre dont les valeurs de décoration acceptables sont :
en arcs: A_2
en nœuds : N_2.

alors T est l'arbre dont les valeurs de décoration acceptables sont :
en arcs: $U'(A_1, A_2)$
en nœuds : $U'(N_1, N_2)$.

Lorsqu'une valeur de décoration acceptable n'est pas définie par le linguiste, elle a la valeur $\top$.

2.2.4. Liens

Il est souvent très utile de pouvoir établir un lien entre différentes unités d'une base lexicales. Certains travaux définissent grâce à de tels liens des graphes recouvrant l'ensemble des lexiques. De plus, cet élément est quasi indispensable dans une approche par transfert. La classe de liens définie ici porte une décoration quelconque.

2.2.4.1. Définition de la structure

La définition d'une classe linguistique ayant une structure de lien se fait de la manière suivante :

```
(link keywords*)
```

où les mots-clés restreignent les classes qui peuvent être valeurs de décoration des différents éléments du lien. Les mots-clés possibles sont :

`:label` *`class`*	spécifie la classe acceptable pour la valeur de décoration du lien.
`:source` *`[dict::]class`*	spécifie la classe de l'instance de laquelle part le lien. Si le lien vient d'une structure d'un autre dictionnaire, on indique ce dictionnaire.
`:target` *`[dict::]class`*	spécifie la classe de l'instance vers laquelle pointe le lien.Si le lien va vers une structure d'un autre dictionnaire, on indique ce dictionnaire.
`:bidirectionnel` *`boolean`*	indique si le système doit gérer le lien inverse (qui pointera vers la structure où est définie le lien d'origine).

Il n'est pas obligatoire de spécifier une classe source pour un lien défini comme valeur d'attribut d'une structure. Dans ce cas, la source du lien sera la structure où il est défini.

2.2.4.2. Héritage

Si parent-class est une classe linguistique ayant une structure de lien, on peut en hériter en définissant une nouvelle classe linguistique par :

```
(parent-class keywords*)
```

où les mots-clés redéfinissent les classes qui peuvent être valeurs de décoration des différentes parties de parent-class. Les mots-clés sont les mêmes que ceux utilisés dans la définition d'un graphe. On peut aussi spécifier le mot-clé *inheritance-type*, qui régit la sémantique de l'héritage :

- si *inheritance-type* a la valeur *simple :* pour chaque mot-clé, les valeurs spécifiées dans la définition de la sous-classe linguistique remplacent les valeurs de la classe mère, spécifiées pour le même mot-clé. Par exemple, si X est défini de la manière suivante :
  ```
  (define-linguistic-class X (link :label Y :target Z))
  ```
  et si X' est défini de la manière suivante :
  ```
  (define-linguistic-class X' (X :label A))
  ```
  alors X' correspond à la structure :
  ```
  (graph :label A :target Z)
  ```
- si *inheritance-type* a la valeur *unification :* pour chaque mot-clé, les valeurs spécifiées dans la définition de la sous-classe linguistique s'unifient avec les valeurs de la classe mère, spécifiées pour le même mot-clé. Par exemple, si X est défini de la manière suivante :
  ```
  (define-linguistic-class X (graph :label Y :target Z))
  ```
  et si X' est défini de la manière suivante :
  ```
  (define linguistic-class X' (X :label A))
  ```
  alors X' correspond à la structure :
  ```
  (graph :label U'(Y,A) :target Z)
  ```

2.2.4.3. Unification

L'unification sur $\Sigma'_{liens} \times \Sigma'_{liens}$ est définie de la manière suivante :

U: $\Sigma'_{liens} \times \Sigma'_{liens} \rightarrow \Sigma'_{liens}$
$(G_1, G_2) \rightarrow G$

Si G_1 est le graphe dont les valeurs de décoration acceptables sont :
en étiquette : L_1
en cible : C_1
en source : S_1
en bidirectionnel : B_1.

et si T_2 est l'arbre dont les valeurs de décoration acceptables sont :
en étiquette : L_2

en cible : C_2
en source : S_2.
en bidirectionnel : B_1.

alors T est l'arbre dont les valeurs de décoration acceptables sont :
en étiquette : U'(L_1, L_2)
en cible : U'(C_1, C_2)
en source : U'(S_1, S_2).
en bidirectionnel : U'(B_1,B_2).

Lorsqu'une valeur de décoration acceptable n'est pas définie par le linguiste, elle a la valeur T.

Notons que U'(B_1,B_2) est équivalent à B_2 (puisque l'unification sur les booléens ne réussit que sur des valeurs identiques).

2.2.5. Automates

Cette structure pourrait sembler très "informatique", pourtant, certains travaux purement linguistiques l'utilisent [Gross 1987]. Cette structure figure donc parmi les structure de base de SUBLIM. La classe d'automates définie ici porte des décorations quelconques sur les nœuds et les arcs.

2.2.5.1. Définition de la structure

La définition d'une classe linguistique ayant une structure d'automate se fait de la manière suivante :

```
(automaton keywords*)
```

où les mots-clés définissent des contraintes sur les classes acceptables en décoration des différents éléments de l'automate :

`:arcs class`	spécifie la classe acceptable en décoration d'un arc.
`:nodes class`	spécifie la classe acceptable en décoration d'un nœud.
`:starting-node class`	spécifie la classe acceptable en décoration du nœud d'entrée.
`:ending-nodes class`	spécifie la classe acceptable en décoration des nœuds de sortie.

class est une classe linguistique quelconque. Si les mots-clés *starting-node* ou *ending-nodes* ne sont pas définis, ils ont la valeur donnée au mot-clé *nodes*.

2.2.5.2. Héritage

Si parent-class est une classe linguistique ayant une structure d'automate, on peut en hériter en définissant une nouvelle classe linguistique par :

```
(parent-class keywords*)
```

où les mots-clés redéfinissent les classes qui peuvent être valeurs de décoration des différentes parties de parent-class. Les mots-clés sont les même que ceux utilisés dans la définition d'un automate. On peut aussi spécifier le mot-clé *inheritance-type*, qui régit la sémantique de l'héritage :

- si *inheritance-type* a la valeur *simple :* pour chaque mot-clé, les valeurs spécifiées dans la définition de la sous-classe linguistique remplacent les valeurs de la classe mère, spécifiées pour le même mot-clé. Par exemple, si X est défini de la manière suivante :
  ```
  (define-linguistic-class X (automaton :arcs Y :nodes Z :starting-node W))
  ```
  et si X' est défini de la manière suivante :
  ```
  (define-linguistic-class X' (X :arcs A :ending-nodes B))
  ```
  alors X' correspond à la structure :
  ```
  (automaton :arcs A :nodes Z :starting-node W :ending-nodes B)
  ```
- si *inheritance-type* a la valeur *unification :* pour chaque mot-clé, les valeurs spécifiées dans la définition de la sous-classe linguistique s'unifient avec les valeurs de la classe

mère, spécifiées pour le même mot-clé. Par exemple, si X est défini de la manière suivante :

```
(define-linguistic-class X (automaton :arcs Y :nodes Z :starting-node W))
```

et si X' est défini de la manière suivante :

```
(define linguistic-class X' (X :arcs A :ending-nodes B))
```

alors X' correspond à la structure :

```
(automaton :arcs U'(Y, A) :nodes Z :starting-node W :ending-nodes B)
```

2.2.5.3. Unification

L'unification sur $\Sigma'_{automates} \times \Sigma'_{automates}$ est définie de la manière suivante :

U: $\Sigma'_{automates} \times \Sigma'_{automates} \rightarrow \Sigma'_{automates}$
$(A_1, A_2) \rightarrow A$

Si A_1est le graphe dont les valeurs de décoration acceptables sont :
en arcs: Ar_1
en nœuds : N_1
en nœud initial : NI_1
en nœuds finals : NF_1.

et si A_2 est le graphe dont les valeurs de décoration acceptables sont :
en arcs: Ar_2
en nœuds : N_2.
en nœud initial : NI_2
en nœuds finals : NF_2.

alors A est le graphe dont les valeurs de décoration acceptables sont :
en arcs: $U'(Ar_1, Ar_2)$
en nœuds : $U'(N_1, N_2)$
en nœud initial : $U'(NI_1, NI_2)$
en nœuds finals : $U'(NF_1, NF_2)$.

Lorsqu'une valeur de décoration acceptable n'est pas définie par le linguiste, elle a la valeur $\top$.

2.2.6. fonctions

Les fonctions lexicales, comme celle introduites par Igor Melčuk, vont jouer un grand rôle dans les dictionnaires. Il est possible de simuler une telle structure avec un lien portant une certaine décoration. Néanmoins, nous avons choisi de l'inclure parmi les structures de base.

2.2.6.1. Définition de la structure

La définition d'une classe fonction se fait de la manière suivante :

```
(function keywords*)
```

où les mots-clés définissent des contraintes sur les classes acceptables pour les différents éléments de la fonction :

`:label class`	spécifie la classe identifiant la fonction.
`:arguments class`	spécifie la liste des classes acceptable pour les arguments de la fonction.
`:value class`	spécifie la classe acceptable en valeur de la fonction.

2.2.6.2. Héritage

Si parent-class est une classe fonction, on peut en hériter en définissant une nouvelle classe linguistique par :

```
(parent-class keywords*)
```

où les mots-clés redéfinissent les classes qui peuvent être valeurs de décoration des différentes parties de parent-class. Les mots-clés sont les même que ceux utilisés dans la définition d'une fonction. On peut aussi spécifier le mot-clé *inheritance-type*, qui régit la sémantique de l'héritage :

- si *inheritance-type* a la valeur *simple :* pour chaque mot-clé, les valeurs spécifiées dans la définition de la sous-classe linguistique remplacent les valeurs de la classe mère, spécifiées pour le même mot-clé. Par exemple, si X est défini de la manière suivante :
  `(define-linguistic-class X (function :label Y :arguments Z :value W))`
  et si X' est défini de la manière suivante :
  `(define-linguistic-class X' (X :label A))`
  alors X' correspond à la structure :
  `(function :label A :arguments Z :value W)`
- si *inheritance-type* a la valeur *unification :* pour chaque mot-clé, les valeurs spécifiées dans la définition de la sous-classe linguistique s'unifient avec les valeurs de la classe mère, spécifiées pour le même mot-clé. Par exemple, si X est défini de la manière suivante :
  `(define-linguistic-class X (function :label Y :arguments Z :value W))`
  et si X' est défini de la manière suivante :
  `(define linguistic-class X' (X :label A))`
  alors X' correspond à la structure :
  `(automaton :label U'(Y, A) :arguments Z :value W))`

2.2.6.3. Unification

L'unification sur $\Sigma'_{fonctions} \times \Sigma'_{fonctions}$ est définie de la manière suivante :

U: $\Sigma'_{fonctions} \times \Sigma'_{fonctions} \rightarrow \Sigma'_{fonctions}$
$(F_1, F_2) \rightarrow F$

Si F_1est une classe fonction dont les valeurs acceptables sont :
en étiquette: L_1
en arguments : A_1
en valeurs : V_1

et si A_2 est une classe fonction dont les valeurs acceptables sont :
en étiquette: L_2
en arguments : A_2
en valeurs : V_2

alors A est une classe fonction dont les valeurs acceptables sont :
en étiquette: $U'(L_1, L_2)$
en arguments : $U'(A_1, A_2)$
en valeurs : $U'(V_1, V_2)$

Lorsqu'une valeur de décoration acceptable n'est pas définie par le linguiste, elle a la valeur $\top$.

2.2.7. Structures de traits

Les structures de traits sont très utilisées par différents formalismes "fondés sur l'unification". Nous les incluons donc dans l'ensemble des structures logiques de base de SUBLIM. La classe définie ici est une généralisation des structures de traits, puisque n'importe quelle structure linguistique peut être valeur d'un trait.

2.2.7.1. Définition de la structure

La définition d'une classe linguistique de type structure de traits se fait de la manière suivante :
`(feature-structure features)`

où l'argument *features* est une liste de couples parenthésés représentant la liste des attributs définis dans la structure, avec la classe acceptable en valeur de chaque attribut.

Contrairement à certains langages d'unification [Aït-Kaci 1986, Emele & Zajac 1990b], apparentés aux langages à prototypes, nous établissons une différence entre les classes (ce que nous définissons ici) et leurs instances, qui serviront de briques de base aux articles de dictionnaires.

Aussi, lorsque nous définissons une classe linguistique de la manière suivante :

```
(feature-structure ((trait1 class1)
                    (trait2 class2)
                    (trait3 class3)))
```

les instances de cette classe linguistique ne peuvent pas contenir de traits non définis ici. Seuls *trait1*, *trait2* et *trait3* seront des traits valides pour ces instances.

Par contre, il est possible d'hériter d'une classe linguistique de type structure de traits en rajoutant de nouveaux traits.

2.2.7.2. Héritage

Si parent-class est une classe linguistique de type structure de traits, on peut en hériter en définissant une nouvelle classe linguistique par :

```
(parent-class features keywords*)
```

où l'argument *features* est une liste de couples parenthésés représentant la liste des attributs (re)définis dans la structure, avec la classe acceptable en valeur de chaque attribut. Les mots-clés permettent de spécifier le comportement de l'héritage.

Lorsqu'un attribut de la classe résultante est égale à l'union des attributs de la classe mère et des attributs de l'argument *features*, la valeur des attributs situés dans l'intersection des attributs de la classe mère et des attributs de l'argument *features*, dépend du mot-clé *inheritance-type*:

- si *inheritance-type* a la valeur *simple :* pour chaque mot-clé, les valeurs spécifiées dans la définition de la sous-classe linguistique remplacent la valeur de la classe mère, spécifiées pour le même attribut. Par exemple, si X est défini de la manière suivante :

  ```
  (define-linguistic-class X (feature-structure ((trait1 X)
                                                 (trait2 Y)))
  ```

  et si X' est défini de la manière suivante :

  ```
  (define-linguistic-class X' (X ((trait1 A)
                                  (trait3 Z)))
  ```

  alors X' correspond à la structure :

  ```
  (feature-structure ((trait1 A)
                      (trait2 Y)
                      (trait3 Z)))
  ```

- si *inheritance-type* a la valeur *unification :* pour chaque mot-clé, les valeurs spécifiées dans la définition de la sous-classe linguistique s'unifient avec les valeurs de la classe mère, spécifiées pour le même attribut. Par exemple, si X est défini de la manière suivante :

  ```
  (define-linguistic-class X (feature-structure ((trait1 X)
                                                 (trait2 Y)))
  ```

  et si X' est défini de la manière suivante :

  ```
  (define-linguistic-class X' (X ((trait1 A)
                                  (trait3 Z)))
  ```

  alors X' correspond à la structure :

  ```
  (feature-structure ((trait1 U'(A, X))
                      (trait2 Y)
                      (trait3 Z)))
  ```

2.2.7.3. Unification

L'unification sur $\Sigma'_{\text{structures de traits}} \times \Sigma'_{\text{structures de traits}}$ est définie de la manière suivante :

U: $\Sigma'_{\text{structures de traits}} \times \Sigma'_{\text{structures de traits}} \rightarrow \Sigma'_{\text{structures de traits}}$
$(S_1, S_2) \rightarrow S$

Soit t_1 l'ensemble des traits définis dans S_1 et t_2 l'ensemble des traits définis dans S_2. L'ensemble t des traits de S est : $t_1 \cup t_2$. Les valeurs associées aux trait de $t_1 \cap t_2$ sont égales au résultat de l'unification faible des valeurs associées aux traits de S_1 et des valeurs associées aux traits de S_2. Les traits qui ne sont pas communs aux deux structures prennent la valeur qui leur est attribuée à l'origine.

2.2.8. Ensembles

Parmi les structures de base de tout système, on trouve les ensembles. Les ensembles définis ici peuvent contenir des éléments homogènes (tous de la même classe) ou hétérogène (de classes différentes).

2.2.8.1. Définition de la structure

La définition d'une classe fonction se fait de la manière suivante :

```
(set-of possible-values keywords*)
```

où l'argument *possible-values* est la liste des classes acceptables pour les éléments de l'ensemble, et où les mots-clés définissent des contraintes de cardinalité sur l'ensemble :

`:min-elements` *number*	spécifie le nombre minimal d'éléments dans l'ensemble (par défaut : 0).
`:max-elements` *number*	spécifie le nombre maximal d'éléments dans l'ensemble (par défaut : pas de maximum).

2.2.8.2. Héritage

Si parent-class est une classe d'ensembles, on peut en hériter en définissant une nouvelle classe linguistique par :

```
(parent-class possible-values keywords*)
```

où l'argument *possible-values* est une liste de classes et où les mots-clés redéfinissent les contraintes de cardinalité sur parent-class.

Si le mot-clé *inheritance-type* n'est pas spécifié (ou vaut *simple*), l'ensemble des classes acceptables pour les éléments de la classe ainsi définie est égal aux classes acceptables indiquées dans *possible-values* .

Les contraintes de cardinalité, sont redéfinies si elles sont respécifiées.

Si le mot-clé *inheritance-type* est égal à *unification*, l'ensemble des classes acceptables pour les éléments de la classe ainsi définie est égal à l'intersection des classes acceptables indiquées dans *possible-values* et des classes acceptables pour parent-class.

Si les contraintes de cardinalité sont redéfinies, le minimum (resp. maximum) sera pris comme contrainte pour le nombre minimal (resp. maximal d'éléments).

2.2.8.3. Unification

L'unification sur $\Sigma'_{\text{ensembles}} \times \Sigma'_{\text{ensembles}}$ est définie de la manière suivante :

U: $\Sigma'_{\text{ensembles}} \times \Sigma'_{\text{ensembles}} \rightarrow \Sigma'_{\text{ensembles}}$
$(E_1, E_2) \rightarrow E$

Soit c_1 l'ensemble des classes acceptables définies dans E_1 et c_2 l'ensemble des classes acceptables définies dans E_2. L'ensemble c des classes acceptables pour E est égal à $c_1 \cap c_2$.

Le nombre minimal d'éléments d'une instance de E est égal au minimum des nombres minimaux d'éléments des instances de E_1 et de E_2.

Le nombre maximal d'éléments d'une instance de E est égal au maximum des nombres maximaux d'éléments des instances de E_1 et de E_2.

2.2.9. Disjonction

Lorsqu'on définit une structure linguistique, il est très intéressant de pouvoir définir une disjonction de classes.

2.2.9.1. Définition de la structure

La définition d'une classe disjonction se fait de la manière suivante :

```
(one-of possible-values)
```

où l'argument *possible-values* est la liste des classes acceptables pour une de ses instances.

Une instance de classe `(one-of (X Y Z))` est soit une instance de classe `X`, soit une instance de classe `Y`, soit une instance de classe `Z`.

2.2.9.2. Héritage

Si parent-class est une classe disjonction, on peut en hériter en définissant une nouvelle classe linguistique par :

```
(parent-class possible-values)
```

où l'argument *possible-values* est une liste de classes.

Si *inheritance-type* n'est pas spécifié (ou vaut *simple*), l'ensemble des classes acceptables pour les éléments de la classe ainsi définie est égal aux classes acceptables indiquées dans *possible-values* .

Sinon, l'ensemble des classes acceptables pour les éléments de la classe ainsi définie est égal à l'intersection des classes acceptables indiquées dans *possible-values* et des classes acceptables pour parent-class.

2.2.9.3. Unification

L'unification sur $\Sigma'_{disjonction} \times \Sigma'_{disjonction}$ est définie de la manière suivante :

U: $\Sigma'_{disjonction} \times \Sigma'_{disjonction} \rightarrow \Sigma'_{disjonction}$
$(D_1, D_2) \rightarrow D$

Soit c_1 l'ensemble des classes acceptables définies dans D_1 et c_2 l'ensemble des classes acceptables définies dans D_2. L'ensemble c des classes acceptables pour D est égal à $c_1 \cap c_2$.

2.2.10. Types de base

Le langage de définition de l'architecture linguistique permet aussi l'utilisation de structures logiques dont le comportement n'est pas raffinable. Ces structures de base sont :

- *boolean :* une classe admettant deux instances interprétées comme vrai et faux.
- *string :* une classe dont les instances sont des chaînes de caractères de longueur quelconque. Le codage de chaînes utilisant des scripts variés dépend de la plate-forme matérielle. Sur Macintosh, une telle chaîne peut être associée à un vecteur de style. Sur une station Unix, on peut utiliser différents codages du système, voire le codage UNICODE.
- *integer :* une classe dont les instances sont des entiers.
- *real :* une classe dont les instances sont des réels.
- *T:* une classe dont les instances sont quelconques (instances de n'importe quelle classe du treillis (Σ, <<)).

2.3. Implémentation

L'implémentation du noyau de SUBLIM repose sur le langage Décor, défini et implémenté par Mathieu Lafourcade [Lafourcade 1994b]. Décor est un langage de décoration implémenté en CLOS.

Ce langage permet de définir des types (analogues aux classes en CLOS). Il est possible de contraindre les type de valeurs possibles pour les attributs (analogues aux slots en CLOS). Enfin, on peut définir son propre mécanisme d'héritage et associer, aux attributs d'un type, des contraintes qui devront être vérifiés à tout moment.

2.3.1. Les structures de base

Les structures de base de SUBLIM sont implémentées comme des *types* en DÉCOR. Les éléments du dictionnaire seront des instances de ces types.

À titre d'illustration, nous allons donner l'implémentation des types structures de traits, arbres, et ensembles.

L'implémentation des structures de traits est immédiate dans un langage de décoration tel que Décor, ces structures étant déjà définies sous le nom `:aggregated`:

```
(define! feature-structure :type
    (:is-a :aggregated))
```

Le type arbre n'est pas défini dans DÉCOR. Nous allons donc le définir comme un type agrégat. Chaque arbre contient un trait donnant son père, la liste de ses fils et sa décoration :

```
(define! daughters :type
    (:is-a :list)
    (:allowed-types '(tree)))
(define! tree :type
    (:is-a :aggregated)
    (father (:type 'tree)
            (:obl t))
    (daughters (:type 'daughters)
               (:obl  t))
    (decoration (:type :top)))
```

Enfin, le type liste étant défini dans Décor, nous l'utiliserons pour l'implémentation des ensembles :

```
(define! set-of :type
    (:is-a :list))
```

2.3.2. Les classes linguistiques

La définition des classes linguistiques est faite par rapport aux structures de base ou à des classes linguistiques déjà définies. Cette définition se traduit par la création d'un nouveau type DÉCOR héritant du type correspondant à la structure de base.

Les expressions de déclaration de classes linguistiques sont des macros LISP qui se réécrivent sous forme de définitions de types.

Nous donnons un exemple d'une telle déclaration pour des structures de traits, des arbres et des ensembles.

Le premier exemple est une déclaration de structure de traits.

```
(define-linguistic-class morph
   (feature-structure
      (graphic-form string)
      (category     cat))
```

Cette déclaration se réécrit en :

```
(define! morph :type
    (:is-a 'feature-structure)
    (graphic-form (:type :lexical))
```

```
    (category (:type 'cat)))
```

Le second exemple porte sur la déclaration d'une classe linguistique basée sur une structure d'arbre :

```
(define-linguistic-class entry
   (tree :root   morph
         :leaves sem-unit))
```

Cette déclaration se réécrit en :

```
(define! entry :type
    (:is-a 'tree)
    (:root-decoration '(morph))
    (:leaves-decoration '(sem-unit)))
```

où les contraintes *root-decoration* et *leaves-decoration* ont été définies comme suit :

```
(define! root-decoration :constraint
    (:arguments (decoration-classes :list))
    (:object-category :decor)
    (:daemons :if-added)
    (:check-constraint-method (object object-category constraint-name args)
        (if (and (first args) (= (get-value object.father) nil))
          (or-list
             (mapcar #'(lambda (x)
                         (is-a-p (get-value object.decoration) x))
                     (first args)) ))))
(define! leaves-decoration :constraint
    (:arguments (decoration-classes :list))
    (:object-category :decor)
    (:daemons :if-added)
    (:check-constraint-method (object object-category constraint-name args)
        (if (and (first args) (= (get-value object.daughters) nil))
          (or-list
             (mapcar #'(lambda (x)
                         (is-a-p (get-value object.decoration) x))
                     (first args)) ))))
```

Le troisième exemple porte sur la définition d'une classe linguistique basée sur une structure d'ensemble :

```
(def-linguistic-class valency
  (set-of (nom à+nom avec+nom comme+nom contre+nom dans+nom de+nom en+nom
           entre+nom par+nom parmi+nom pour+nom sur+nom inf à+inf de+inf
           adj que+ind que+subj se-moy se-pass lieu-stat lieu-dyn manière
           zéro)))
```

Cette définition se réécrit en :

```
(define! valency :type
    (:is-a 'set-of)
    (:allowed-types
         '(nom à+nom avec+nom comme+nom contre+nom dans+nom de+nom en+nom
           entre+nom par+nom parmi+nom pour+nom sur+nom inf à+inf de+inf
           adj que+ind que+subj se-moy se-pass lieu-stat lieu-dyn manière
           zéro)))
```

L'héritage d'une structure linguistique définie auparavant se traduit exactement de la même manière qu'une définition, à partir d'une des structures de base prédéfinies. Si par exemple la classe linguistique UM est définie de la manière suivante :

```
(def-linguistic-class UM
  (feature-structure
    ((formes-brèves (set-of ((link :target UM
                                   :label type-forme-brève))))
     (étymologie    (set-of ((link :target étymon))))
     (combVE        (link :target combVE))
     (appellation    string)
    )))
```

Cela correspond en DÉCOR à :

```
(define! UM :type
    (:is-a 'feature-structure)
    (formes-brèves (define! nil :type
                       (:is-a 'set-of)
                       (:allowed-types
                             '((define nil :type
                                   (:is-a 'link)
                                   (target (:type 'UM))
                                   (label  (:type 'type-forme-brève)))))))
    (étymologie    (define! nil :type
                       (:is-a 'set-of)
                       (:allowed-types '((define nil :type
                                             (:is-a 'link)
                                             (target (:type 'étymon)))))))
    (combVE        (define nil :type
                       (:is-a 'link)
                       (target (:type 'combVE))))
    (appellation    :lexical))
```

Il sera possible d'hériter de cette structure, comme dans la définition suivante :

```
(def-linguistic-class UM_S
  (UM
    ((usyn-l     (set-of (USyn)))
     (a-pour-Umg (set-of (Umg) :min-elements 1))
     (a-pour-Ump (set-of (Ump) :min-elements 1)))
  ))
```

Ce qui correspond, de manière analogue, à :

```
(define! UM_S :type
    (:is-a 'UM)
    (usyn-l     (define! nil :type
                    (:is-a 'set-of)
                    (:allowed-types '(USyn))))
    (a-pour-Umg (define! nil :type
                    (:is-a 'set-of)
                    (:allowed-types '(Umg))
                    (:range 1)))        ;; seule la borne inférieure est notée
                                        ;; lorsqu'il n'y a pas de borne sup.
    (a-pour-Ump (define! nil :type
                    (:is-a 'set-of)
                    (:allowed-types '(Ump))
                    (:range 1)))
```

Lorsque l'interprétation de l'héritage est différente du mécanisme d'héritage standard, DÉCOR permet, grâce à des formules, de modifier cet héritage. Ainsi, on définit l'héritage d'une structure d'ensemble de la manière suivante :

```
(def-linguistic-class T2
  (T1 (x1 x2 x3)
      :min-elements 1))
```

se réécrit en :

```
(define! T2 :type
    (is-a 'T1)
    (:allowed-types (:value '(set-inheritance super-type '(x1 x2 x3)))
                    (:interpretation :formula))
    (:range (:value '(min (get-value T1.range 1)))
            (:interpretation :formula)))
```

où set-inheritance est une fonction LISP calculant l'héritage sur l'ensemble des classes possibles pour les éléments de l'ensemble.

2.3.3. Étendre SUBLIM

Il est possible d'étendre SUBLIM en lui ajoutant des structures logiques de base supplémentaires. Cette extension passe par la définition d'un type de base portant le nom de la structure que l'on ajoute.

Ce nouveau type *T* doit répondre au protocole SUBLIM, qui spécifie :

- la définition : on doit définir une macro ayant la forme générale du langage noyau SUBLIM et se réécrivant en une définition de sous-type de *T*,
- l'héritage : on doit définir une formule régissant l'héritage d'une classe linguistique basée sur *T*.

Il est ainsi assez facile de définir une nouvelle structure linguistique dans le noyau de SUBLIM. D'autre part, comme nous le verrons dans la suite, SUBLIM n'est pas restreint à un noyau de manipulation de structure, et comporte de nombreux outils qui sont développés dans le chapitre suivant.

Une extension de SUBLIM est donc bien plus complexe, puisqu'elle demande que la nouvelle structure de base réponde aux différents protocoles des différents outils. Notons que cette condition est nécessaire, mais non encore suffisante, puisque presque chacun de ces outils est paramétrable par un langage spécialisé qui lui est propre. La syntaxe de ces langages spécialisés devra donc éventuellement être modifiée pour prendre en compte la nouvelle structure de base.

V. Architecture logicielle et outils de gestion

1. Architecture logicielle

L'architecture logicielle utilisée dans ce projet a été définie lors de ma participation au projet MULTILEX. Cette architecture ayant été retenue par le consortium, elle est identique à l'architecture de MULTILEX. Par contre, sa mise en œuvre est plus complexe, car MULTILEX utilise un noyau fondé sur des structures de traits, alors que SUBLIM utilise un noyau fondé sur le multi-formalisme.

L'architecture logicielle de SUBLIM distingue fortement les problèmes de stockage, de manipulation et de visualisation des données. Elle est basée sur trois niveaux :

- **niveau base de données :** ce niveau est en charge du stockage effectif des données. Différents systèmes relationnels de gestion de bases de données peuvent être utilisés à ce niveau. On peut aussi vouloir utiliser des outils plus spécialisés (comme GENELEX qui a expérimenté une approche "tout en mémoire"). Ce niveau est invisible pour l'utilisateur.
- **niveau interne :** ce niveau est en charge des différentes manipulations sur les entrées de dictionnaires. C'est à ce niveau que les différents outils d'un système de gestion de bases lexicales opèrent. Ce niveau correspond aux structures que le linguiste a définies avec le langage LINGARD.
- **niveau présentation :** ce niveau est en charge de la présentation des informations à l'utilisateur. Cette présentation n'est pas nécessairement proche de la structure interne utilisée. De plus, il peut être possible de proposer différentes présentations d'une même information pour différents utilisateurs ou différents buts. Ce niveau de présentation peut être prototypé grâce à un éditeur de documents structurés, comme GRIF.

Cette architecture est illustrée par la figure 5.1.

Le fonctionnement de cette architecture est basé sur l'aller-retour entre les différents niveaux. Une requête sera formulée au niveau présentation, puis traduite en une structure du niveau interne. Cette structure sera elle même traduite en une requête de base de données. Le résultat sera transformé en un ensemble de structures du niveau interne, qui sera visualisé au niveau présentation.

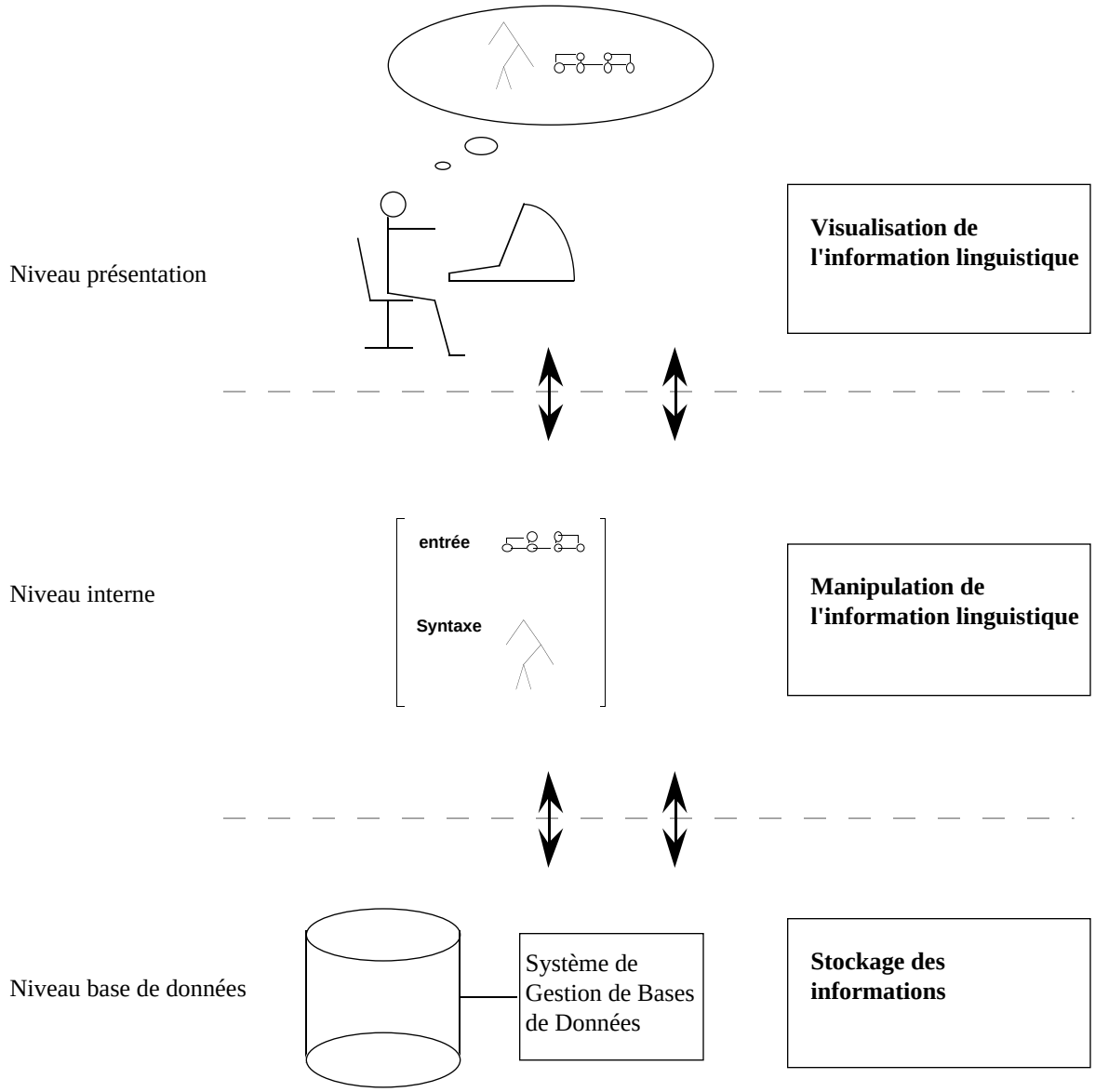


Figure 5.1 : Architecture logicielle du système SUBLIM

Cette architecture permet au système SUBLIM d'être neutre par rapport au type de base de données que l'on souhaite utiliser. Elle permet aussi de bien séparer structure interne et présentation, ce qui permet de bénéficier des avantages exposés dans la partie suivante.

2. Niveau Interne : manipulation des informations linguistiques

L'utilisation d'une base quelle qu'elle soit passe par la possibilité d'extraire des informations de cette base. Dans l'outil SUBLIM, les informations linguistiques ont deux caractéristiques essentielles :

- une structuration très forte,
- un mélange de structures logiques de base.

Il nous faut donc développer un moyen qui permette d'extraire de la base lexicale une structure, selon les critères que le linguiste souhaite définir.

Pour extraire une structure de la base lexicale, le linguiste doit donner l'ensemble des conditions minimales que doivent vérifier les structures à extraire. Pour cela, il spécifie une structure "patron". Le résultat de cette requête d'extraction est la liste des éléments de dictionnaire qui s'apparient avec la structure "patron".

Cette extraction peut être utilisée pour différents types d'utilisation (navigation, manipulation). Dans le cas où le linguiste souhaite faire des calculs sur les structures ainsi extraites, il est intéressant d'affecter à des variables des éléments de cette structure.

Pour définir une structure "patron", il faut pouvoir dénoter des instances particulières des différentes classes linguistiques. Notre langage le permet.

De plus, lorsqu'on sait dénoter une structure linguistique, il faut pouvoir lire et manipuler les différentes valeurs associées aux éléments de cette structure.

Dans cette partie, nous donnons un moyen de dénoter un ensemble de structures linguistiques et un moyen de manipuler ses différentes parties.

2.1. Dénoter un ensemble de structures

Pour dénoter un ensemble de structures, on définit un "patron". Ce patron est une structure partiellement définie et comportant des variables. Il désigne toutes les structures de la base auxquelles il s'apparie.

Les particularités du système Sublim font qu'un tel patron peut être complexe. Par exemple, on doit pouvoir désigner une structure d'arbre, dont la racine est un automate où l'état d'entrée est une structure de traits contenant au moins un trait nommé `cat` ayant la valeur X, et dont les feuilles sont des structures de traits ayant un trait nommé `cat` avec la même valeur X.

On le voit, le linguiste doit pouvoir spécifier un patron très complexe. Dans ce cas, le plus simple pour le linguiste est de manipuler une vue graphique. Il pourra ainsi visualiser sa structure linguistique de manière simple alors que la traduction de cette vue graphique en sa vue interne sera trop complexe pour être lisible.

Nous ne donnerons pas la syntaxe interne dans son ensemble, mais en montrerons des extraits dans les exemples de contraintes et de règles de valeurs par défaut que nous donnons plus loin.

2.2. Manipuler une structure linguistique

Pour manipuler une structure linguistique, il faut disposer de moyens d'accéder aux différentes composantes de cette structure. Dans cette section, nous donnons les différentes fonctions d'accès aux informations linguistiques.

2.2.1. valeurs de base

Les valeurs de base (boolean, string, integer, real) sont notées sous leur forme habituelle. Par exemple :

- *boolean :* `true`, `false`;
- *string :* `"ceci est une chaîne"`, `"cela aussi"`...
- *integer :* `1`, `2`, `3`...
- *real :* `1.32`, `2`...

2.2.2. Arbres

La manipulation d'une structure d'arbre passe par les primitives suivantes :

- *root :* retourne la racine de l'arbre ;
- *daughters:* retourne les sous-arbres de l'arbre ;
- *leaves :* retourne la liste des feuilles de l'arbre ;
- *leave? :* retourne vrai si l'arbre est une feuille ;

- *nodes :* retourne la liste des nœuds de l'arbre (cette fonction prend un argument supplémentaire indiquant si le parcours se fait en profondeur d'abord ou en largeur d'abord).

2.2.3. Graphes

La manipulation d'une structure de graphe passe par les primitives suivantes :
- *nodes :* retourne la liste des nœuds du graphe, sans duplication, sans ordre particulier ;
- *arcs :* retourne la liste des arcs du graphe, sans duplication, sans ordre particulier.

Sur un nœud d'un graphe, on peut utiliser les primitives :
- *entering-arcs :* retourne la liste des arcs menant à ce nœud ;
- *leaving-arcs :* retourne la liste des arcs partant de ce nœud ;

Sur un arc d'un graphe, on peut utiliser les primitives :
- *source :* le nœud d'où vient l'arc ;
- *target :* le nœud vers lequel pointe l'arc.

Et, indifféremment sur un nœud ou sur un arc :
- *decoration :* retourne la structure de décoration associée au nœud (ou à l'arc).

2.2.4. Liens

La manipulation d'un lien est identique à la manipulation d'un arc de graphe :
- *source :* le nœud d'où vient l'arc ;
- *target :* le nœud vers lequel pointe l'arc ;
- *decoration :* retourne la structure de décoration associée à l'arc.

2.2.5. Automates

La manipulation d'une structure d'automate passe par les primitives suivantes :
- *starting-node :* retourne l'état initial de l'automate ;
- *ending-nodes :* retourne la liste des états finals de l'automate ;
- *nodes :* retourne la liste des nœuds de l'automate ;
- *transitions :* retourne la liste des transitions de l'automate.

Sur l'état d'un automate, on peut utiliser les primitives :
- *entering-transitions :* retourne la liste des transitions menant à cet état ;
- *leaving-transitions :* retourne la liste des transitions partant de cet état.

Sur une transition, on peut utiliser les primitives :
- *source :* l'état d'où vient la transition ;
- *target :* l'état vers lequel pointe la transition.

Et, indifféremment sur un état ou sur une transition :
- *decoration :* retourne la structure de décoration associée au nœud (ou à l'arc).

2.2.6. fonctions

La manipulation d'une structure fonction passe par les primitives suivantes :
- *label :* retourne le label de la fonction (qui peut être une structure complexe) ;
- *arguments :* retourne les arguments associées à une instance de fonction (lorsque cette instance est associée à une structure particulière) ;
- *value :* retourne la valeur associée à une instance de fonction (lorsque cette instance est associée à une structure particulière) ;
- *apply :* retourne la valeur résultat de l'application de la fonction aux arguments passés en paramètres.

2.2.7. Structures de traits

La manipulation d'une structure de traits passe par les primitives suivantes :

- *get-value :* prend un chemin en paramètre, et retourne la valeur associée à ce chemin ;
- *features :* liste les traits ayant une valeur définie dans la structure ;
- *unify :* unifie les deux structures de traits passées en paramètres.

2.2.8. ensembles

- *union :* union ensembliste ;
- *intersection :* intersection ensembliste ;
- *subset? :* vérifie que le second argument (un ensemble) est sous-ensemble du premier (un ensemble) ;
- *element? :* vérifie que le second argument (un élément) est élément du premier (un ensemble) ;
- *card :* renvoie le cardinal de l'ensemble.

3. Éditeur, navigateur

Le niveau présentation défini dans l'architecture logicielle du système SUBLIM regroupe notamment un éditeur et un navigateur.

L'éditeur permet la création, la modification ou l'effacement d'une entrée de dictionnaire. L'éditeur doit proposer une vue d'ensemble de l'entrée de dictionnaire.

Le navigateur permet la sélection et la visualisation d'entrées de dictionnaire. Dans l'idéal, un tel navigateur devrait proposer différentes visualisations des informations, suivant l'utilisateur et suivant ses buts.

Néanmoins, il est souhaitable que le navigateur et l'éditeur soient confondus. En effet, c'est par la navigation que l'on détecte d'éventuelles erreurs dans un dictionnaire, et il est frustrant de ne pas pouvoir modifier l'entrée en question sans changer d'outil. Aussi, l'outil de navigation doit proposer des fonctions d'édition lorsque la visualisation le permet (lorsqu'il est possible de passer d'une modification sur la présentation à une modification sur la structure interne du dictionnaire).

C'est pour toutes ces raisons que nous avons choisi de réutiliser les résultats des recherches faites dans le domaine des documents structurés. Nous partons de la constatation qu'un dictionnaire est un document structuré. Dans cette section, nous commencerons par donner un aperçu de ce qu'est un document structuré. Nous donnerons ensuite les avantages de l'utilisation des documents structurés dans le contexte des bases lexicales multilingues.

3.1. Les documents structurés

Dans le domaine de l'édition de documents, on peut distinguer une typologie partielle des systèmes de production de documents :

- Traitements de texte : ces systèmes de production permettent l'édition de documents (principalement textuels), en affectant à certaines parties du texte des attributs "typographiques". Il permettent donc d'éditer le texte, et sa forme.
- Formateurs : ce sont des outils non-interactifs, ils prennent une description de la typographie d'un document (dans un format particulier) et en produisent une version formatée. Certains des formats utilisés permettent de s'abstraire de la présentation du document.
- Éditeurs et formateurs spécialisés : ces systèmes permettent d'éditer des éléments non-textuels, comme des formules ou des dessins…

À cette typologie, se rajoutent les éditeurs de documents structurés. Leur concepteurs partent d'une constatation simple : un document a une structure. Par exemple, un article de revue a un titre, une liste d'auteurs, un résumé, un ensemble de parties, de sous-parties, de paragraphes,

de figures, etc. Ces différents éléments se combinent entre eux pour former un document, de manière structurée (en effet, un paragraphe peut être élément d'une sous-partie, elle même élément d'une partie).

Partant de cette constatation, on a pu développer des systèmes de production et d'édition de documents structurés qui utilisent une description de la structure des documents qu'ils produisent. Ces documents, ou certains de leurs éléments, peuvent être très fortement structurés (comme une équation mathématique).

La structure que connait l'éditeur est une structure logique. Pour que l'utilisateur puisse créer (ou lire) un document, le système de production de documents doit pouvoir en offrir une visualisation (titre centré gras, auteurs centrés italique…). Cette visualisation est une structure physique de présentation reflétant tout ou partie de la structure logique du document. Ainsi, un article peut être visualisé de différentes manières : sous la forme habituelle, avec des styles différents, sous forme de plan…

Pour cela, on peut définir différentes "présentations" d'un même article. Le titre sera centré pour un article dans la revue X ou bien cadré à gauche dans la revue Y. Ces présentations régissent donc la forme globale du document. Parallèlement à ces présentations, on peut définir différentes "vues" qui agissent comme des filtres sur les informations à présenter. Il est ainsi possible de ne montrer que le plan d'un article :

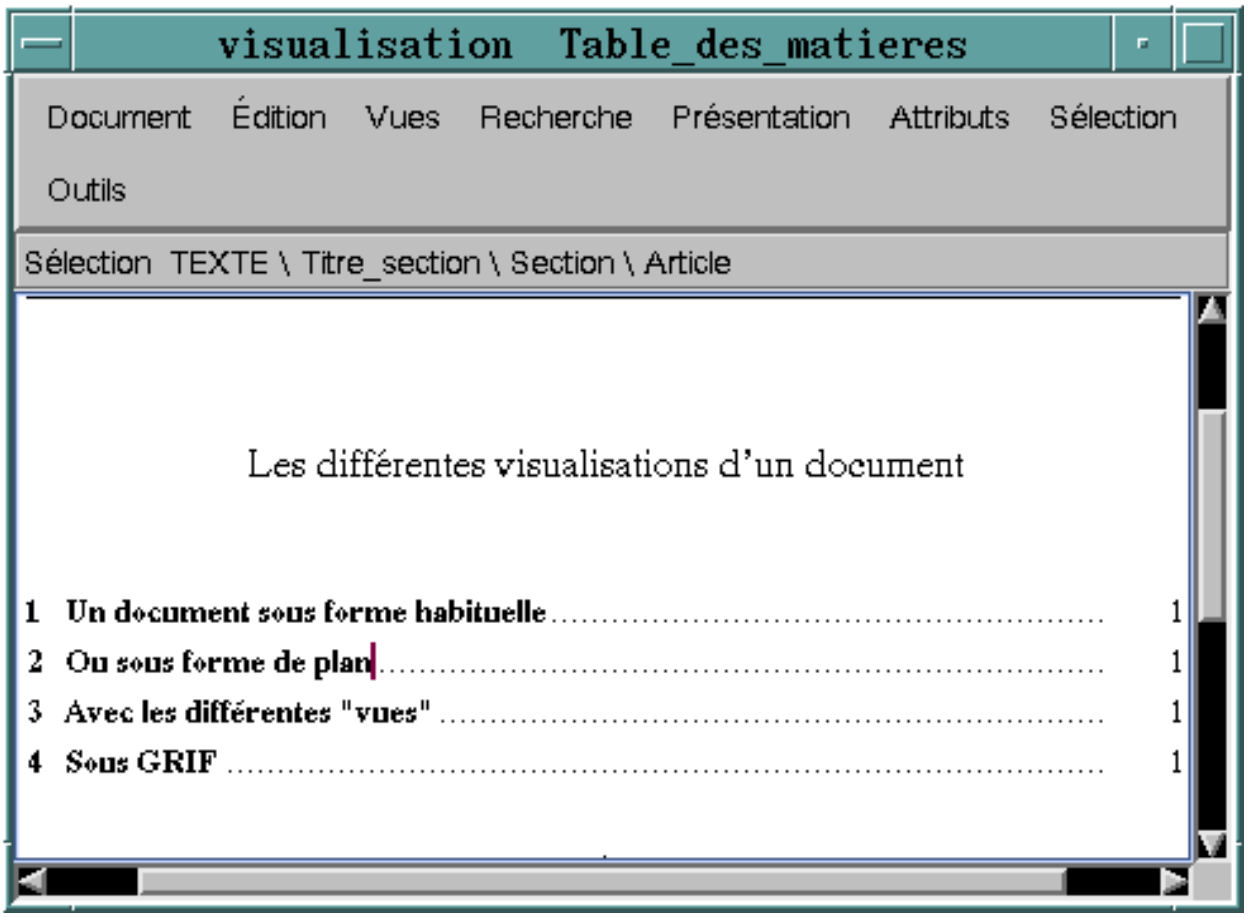


Figure 5.2 : Un article vu sous forme de table des matières

Ce document est un article dont on a simplement changé la vue. La vue standard du document le montrera dans son ensemble :

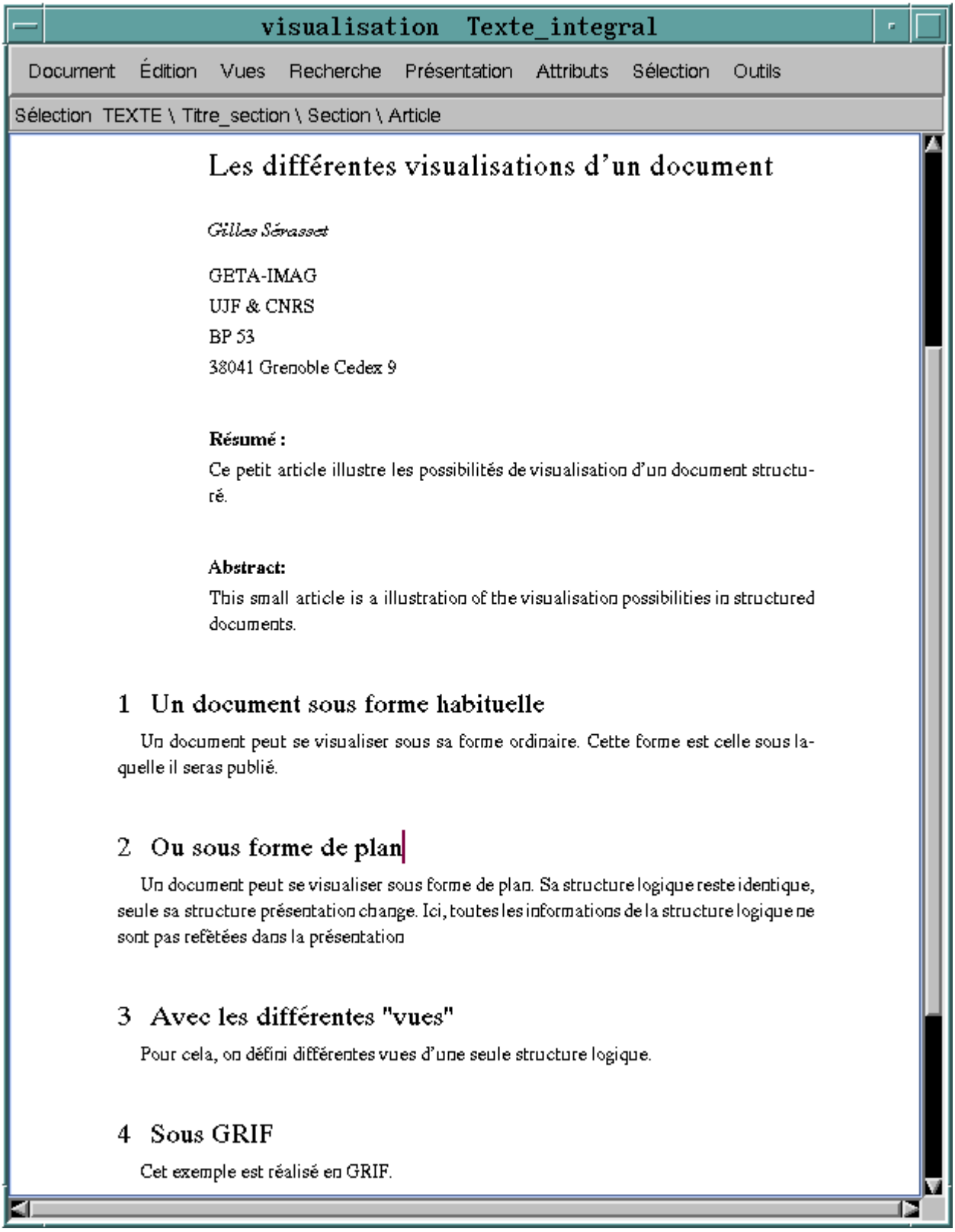


Figure 5.3 : Le même article vu sous la forme habituelle

Une telle méthode permet de s'affranchir des problèmes typographiques lors de l'édition d'un document.

3.2. Le dictionnaire, un document structuré

De la même manière qu'un document a une structure interne, un dictionnaire est un ensemble d'articles ayant une structure particulière. Dans le chapitre précédent, nous avons

exposé les moyens dont dispose le linguiste pour définir la structure d'une base lexicale et des dictionnaires qu'elle contient.

Lorsqu'on veut créer, remplir, gérer ou consulter un dictionnaire, on utilise une forme particulière reflétant cette structure interne. La problématique étant analogue à celle des documents structurés, nous proposons de réutiliser les outils produits dans ce domaine pour gérer l'interface entre un utilisateur et un dictionnaire.

Un dictionnaire est tout de même un document structuré particulier, par sa taille, par la complexité des structures qu'il peut contenir, et par le nombre élevé d'usages différents que l'on veut en faire.

Ces particularités rendent cruciale la possibilité de proposer différentes vues d'un même dictionnaire. En effet, les utilisateurs d'un dictionnaire souhaitent avoir une forme particulière à leur usage, soit parce qu'elle est mieux adaptée à leurs motivations, soit parce qu'ils ne souhaitent voir que les informations pertinentes à un usage particulier.

Pour illustrer cette possibilité, prenons l'exemple d'un dictionnaire bilingue que nous allons coder en GRIF. Ce dictionnaire a un titre, des commentaires, et une liste d'entrées regroupées en lettres.

Les entrées du dictionnaire se composent d'un lemme, d'une catégorie et d'une liste de sens. Un sens comprend un contexte ou synonyme (l'identifiant parmi les différents sens possibles) ainsi qu'une liste de constructions syntaxiques (un ensemble d'arbres), une liste de traductions (repérées par un contexte), et une liste d'exemples avec leurs traductions.

En GRIF, cette structure s'écrit de la manière suivante, dans le langage S[4]:

```
{Nom de la structure}
STRUCTURE bilingue;
{Nom de sa présentation principale}
DEFPRES bilingueP;
STRUCT
{Un dictionnaire bilingue a deux attributs (la langue cible et la langue source}
{Il a un nom, des commentaires et un ensemble d'entrées }
   bilingue (ATTR !Langue_source  = TEXT; !Langue_cible = TEXT) =
    BEGIN
         Nom_Dico = Text;
        ?Commentaire = Paragraphe_sequence;
         Entr\351e_sequence = LIST OF (Lettres_Entr\351e);
    END;
   Paragraphe_sequence = LIST OF (Paragraphe);

{ Les entrées sont regroupées par lettres }
   Lettres_Entr\351e (ATTR !Lettre = TEXT) = LIST OF (Entr\351e);
{ Une entrée comprend un lemme, une catégorie et une liste de sens }
   Entr\351e = BEGIN
        Lemme = TEXT;
        cat\351gorie = TEXT;
        Liste_sens = LIST OF (sens);
    END;

{ Le sens est indiqué par un contexte, suivi d'un ensemble d'arbres
syntaxiques, de traductions et d'exemples }
   sens = BEGIN
       Contexte_Global = TEXT;
      ?Syntaxes = LIST OF (Arbre);
      ?Traductions = LIST OF (Trad);
      ?Exemples = LIST OF (Exemple);
    END;
```

[4] L'annexe B donne une introduction à GRIF et présente brièvement ses différents langages.

```
   Trad = BEGIN
       Contexte_Source = TEXT;
       Traduction = TEXT;
    END;

   Exemple = BEGIN
       Exemple_Source = TEXT;
       Traduction_exemple = TEXT;
    END;
END
```

Lorsqu'on indexe ce dictionnaire, on veut avoir une vue où les informations sont complètes et clairement séparées. Ainsi, l'entrée *composer* du dictionnaire peut être créée sous la forme donnée par les figures 5.4 et 5.5.

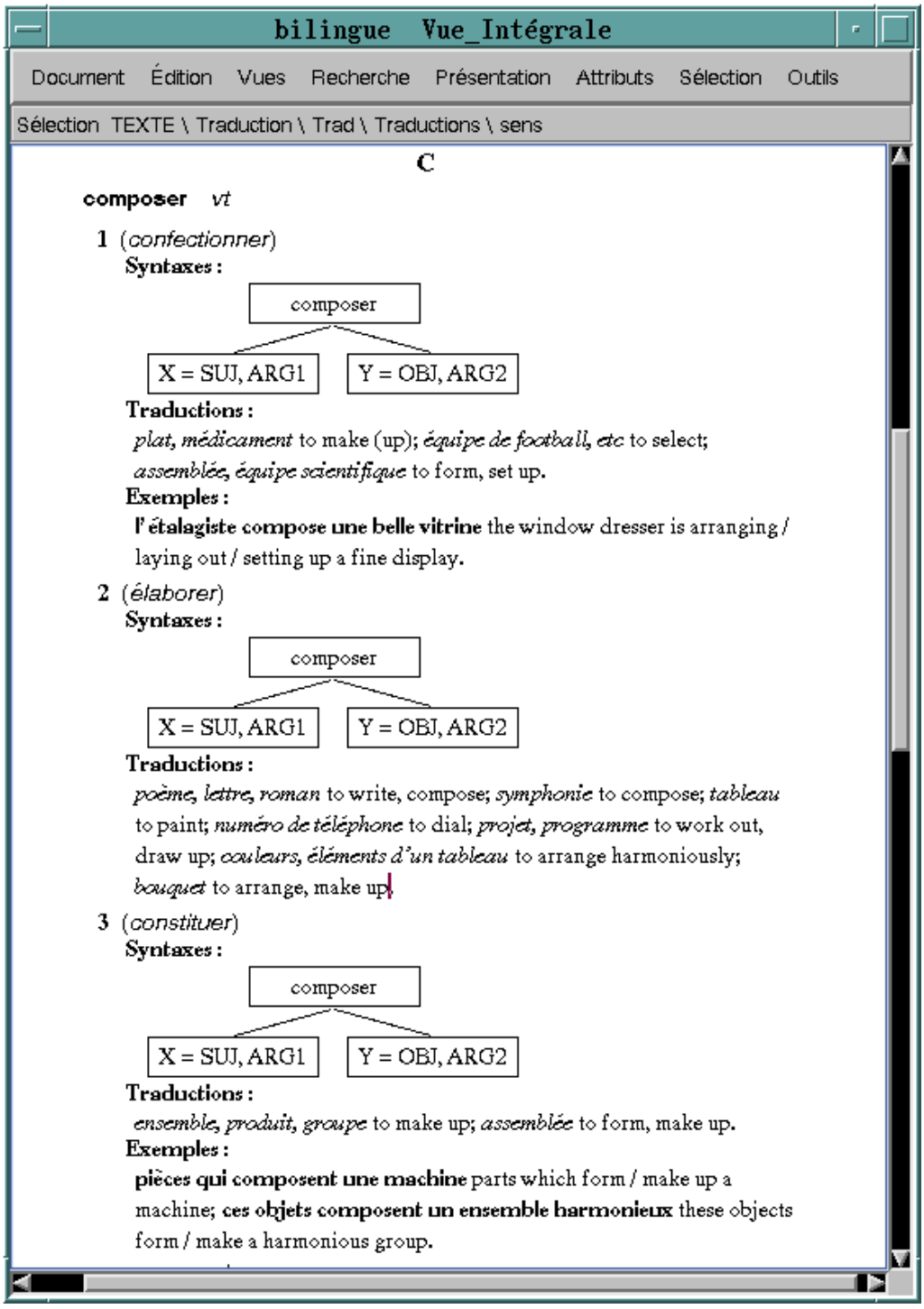


Figure 5.4 : Vue intégrale de l'entrée *composer* (transitif)

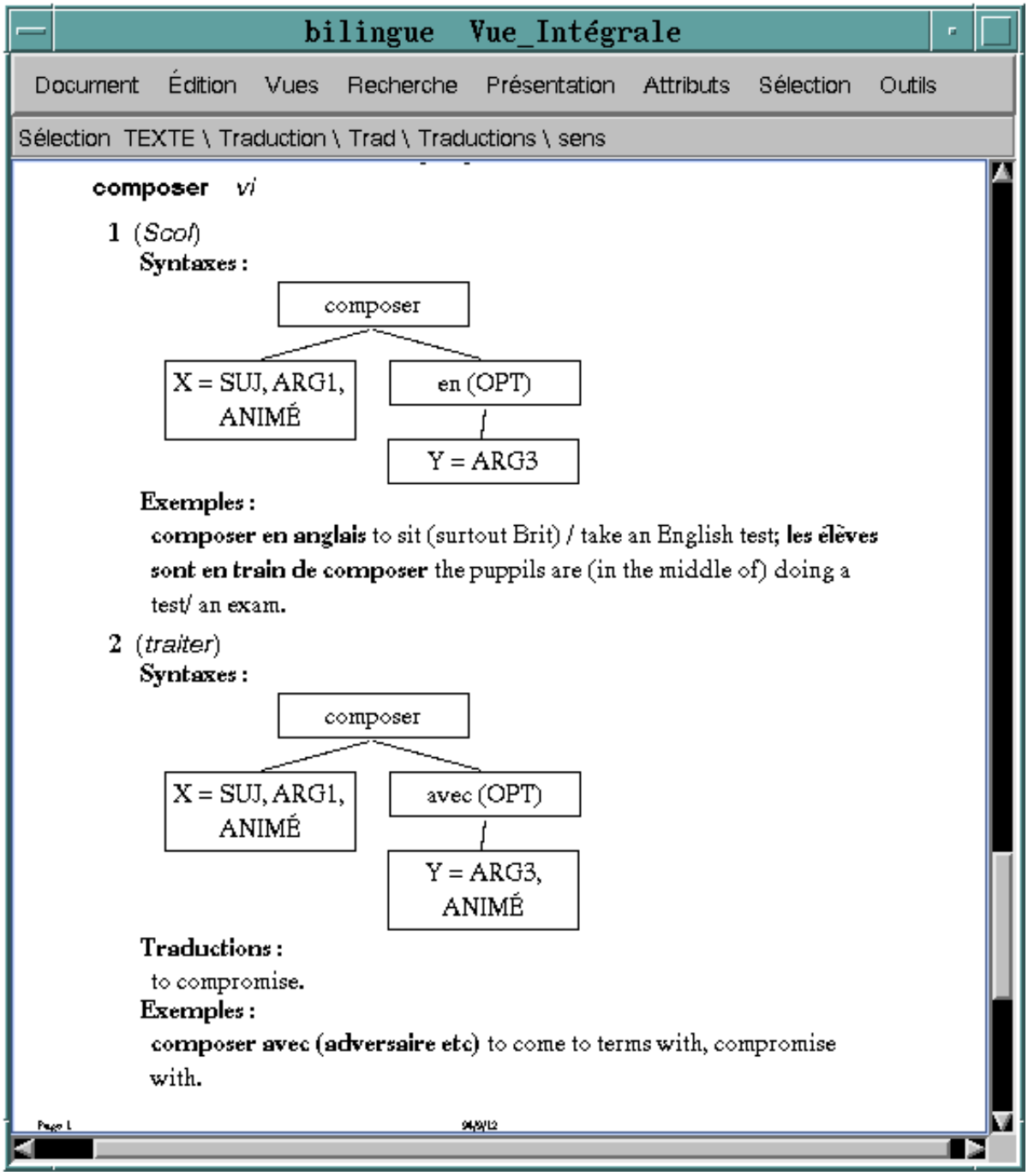


Figure 5.5 : Vue intégrale de l'entrée composer (intransitif)

Cette vue nous permet d'éditer chaque entrée en connaissant sa structure. On peut de plus éditer les arbres syntaxiques des différents sens.

Si l'on souhaite éditer ce dictionnaire sous une forme papier, on ne veut pas forcément voir l'ensemble de sa structure, mais uniquement les informations nécessaires à un humain. Une telle vue "éditoriale" est donnée dans la figure 5.6.

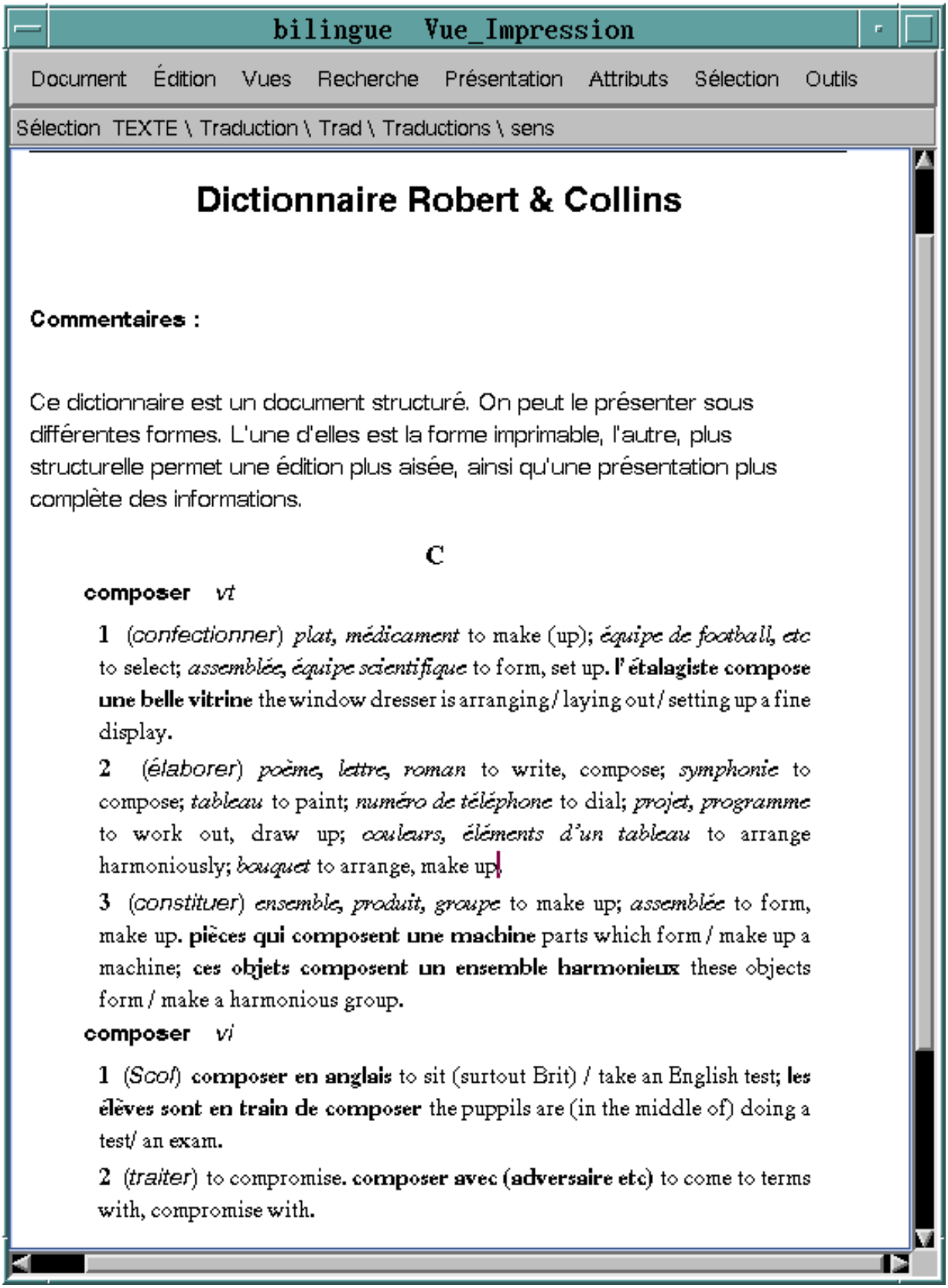


Figure 5.6 : Vue éditoriale du dictionnaire bilingue

Enfin, si l'on souhaite uniquement faire une étude des arbres syntaxiques des différents sens des entrées, on utilisera une vue permettant de masquer les informations non pertinentes. Cette vue est donnée en figure 5.7.

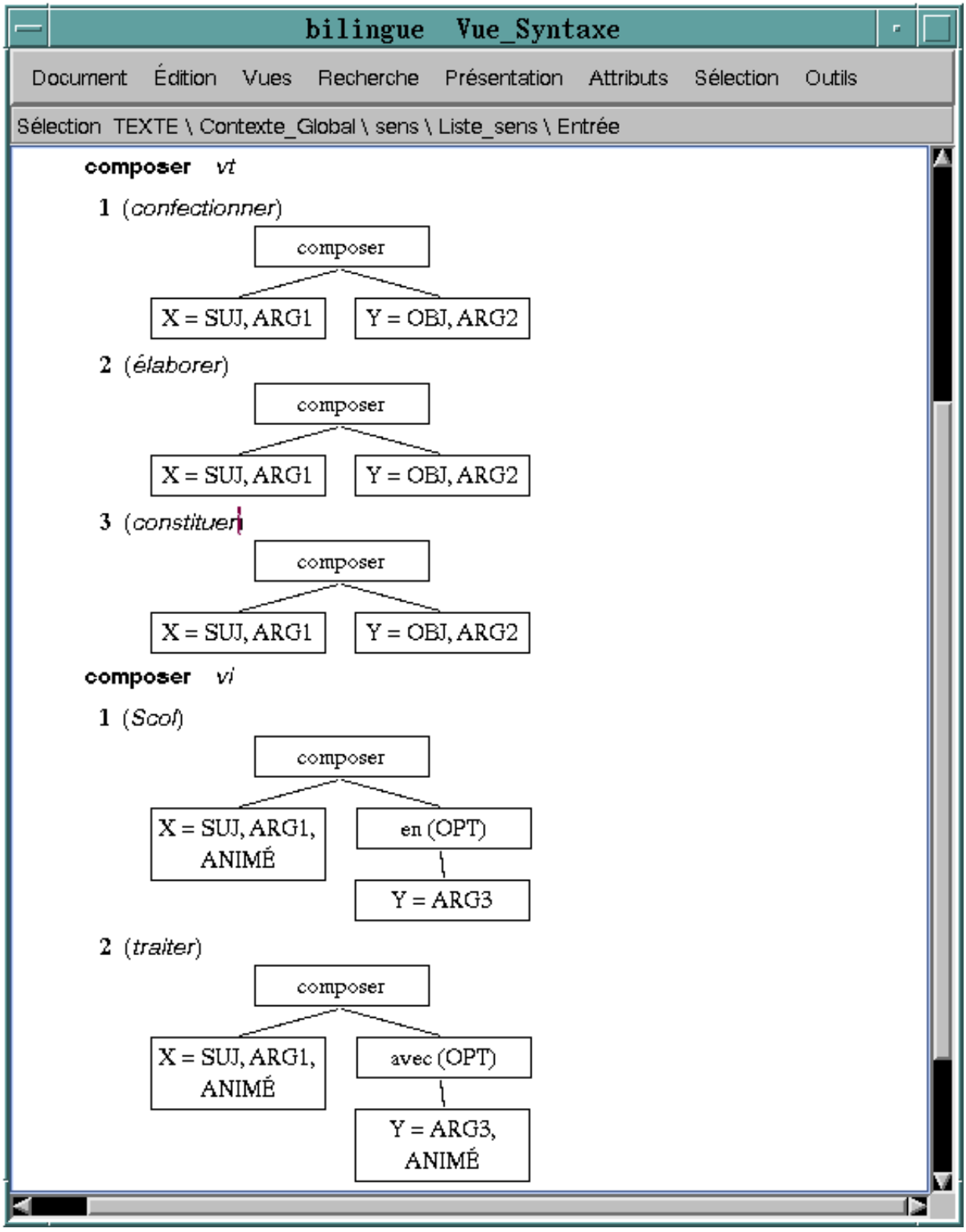


Figure 5.7. : Vue “syntaxique” du dictionnaire bilingue.

L’utilisation d’un système de production de documents structurés nous permet donc, pour un coût réduit, d’offrir de nombreuses vues d’un même dictionnaire. Par la création de vues “éditoriales”, on intègre l’ensemble de la chaîne de production d’un dictionnaire (conception, création, gestion, édition…) dans un seul outil.

L’utilisation du système GRIF, pour la présentation des données, est facilitée par son architecture. En effet, il ne se présente pas uniquement comme une application autonome et

fermée, mais comme une boîte à outils. Cet aspect permet de l'utiliser, non pas comme un éditeur indépendant du système SUBLIM, mais comme un composant intégré au système.

Néanmoins, GRIF présente actuellement deux limitations qui sont importantes dans le contexte des dictionnaires.

En premier lieu, GRIF est un outil mono-script. Cela signifie qu'il n'est capable de gérer que les langues dont l'écriture est basée sur l'alphabet romain. Ainsi, il est impossible de manipuler des dictionnaires russes, arabes, chinois ou japonais[5].

L'éditeur GRIF permet d'éditer et de manipuler du texte. Or, certaines structures de dictionnaire ont des attributs qui ont un nombre fini de valeurs atomiques possibles. Dans ce cas, on souhaiterait que l'éditeur ne permette pas la saisie d'une valeur non prévue. Cela peut se faire en associant l'attribut en question à un menu déroulant contenant l'ensemble des valeurs possibles. Mais, deuxième limitation, l'utilisation d'un tel menu n'est pas possible à l'intérieur d'un document GRIF.

Bien que GRIF soit utilisable pour construire une maquette de système, voire un prototype limité dans les langues qu'il accepte, il ne peut pas être utilisé dans son état actuel pour la création d'un système de gestion de dictionnaires multilingues.

Son utilisation dans une maquette a deux avantages importants :
- illustrer les avantages d'une approche du dictionnaire en tant que document structuré,
- inciter les constructeurs de système de production de documents structurés à généraliser leurs produits et à les étendre à de nouveaux scripts.

4. Vérificateur de cohérence

Le but du vérificateur de cohérence est de vérifier que les entrées d'un dictionnaire sont conformes à des contraintes spécifiées a priori. Ces contraintes sont définies en référence à la structure linguistique du dictionnaire.

Pour chaque dictionnaire, le linguiste peut définir un ensemble de contraintes et les vérifier sur l'ensemble du dictionnaire lorsqu'il le souhaite (contraintes statiques). Il peut aussi définir des contraintes qui seront vérifiées à chaque fois qu'une entrée sera créée ou modifiée (contraintes dynamiques).

Notons que certaines parties de la définition de l'architecture linguistique sont analogues à des contraintes (la cardinalité d'une liste par exemple). Ces contraintes, dites "structurelles" sont vérifiées dynamiquement à la modification des attributs sur lesquels elles portent.

Après avoir précisé les notions utilisées par le vérificateur de cohérence, nous donnerons quelques exemples de contraintes.

4.1. Notions

Une *contrainte* est une règle définie par un linguiste. Ces contraintes sont vérifiées lorsque le linguiste le souhaite, où à chaque fois qu'une entrée est créée ou modifiée.

Un *filtre* est un ensemble de contraintes.

On définit trois *niveaux* de contraintes :
- *Alerte :* lorsqu'une contrainte de ce niveau est invalide pour une entrée, un message est envoyé au linguiste. Tous les traitements restent autorisés sur cette entrée. L'alerte

---

[5] En effet, le travail de multilinguisation de GRIF effectué par Huy Khánh Phan [Phan 1991, Phan & Boitet 1992] a été réalisé sur une version de laboratoire figée et n'a pas (encore) été repris dans la version commerciale, ni dans les versions de recherche plus récentes du projet OPERA.

disparaît dès que le lexicographe valide l'entrée. Ce type de contrainte est utilisé pour détecter des erreurs potentielles.
- *Délai :* lorsqu'une contrainte de ce niveau est invalide pour une entrée, un message est envoyé au linguiste. L'entrée en question ne pourra pas être exportée. Les traitements interactifs (édition, navigation) ne sont pas changés. Ces contraintes sont utilisées pour la gestion d'entrées temporairement incomplètes.
- *Critique* : ce niveau de contrainte n'est pertinent que pour une contrainte dynamique. Lorsqu'une contrainte de ce niveau est violée par une transaction sur une entrée, cette transaction est annulée. Un message est envoyé au linguiste avec les renseignements nécessaires à la rectification de l'erreur.

On définit trois *types* de contraintes :
- *Intégrité :* une contrainte d'intégrité s'applique à un article d'un dictionnaire de la base lexicale. Elle assure qu'aucun article de la base lexicale ne présente une configuration illicite.
- *Cohérence locale :* un contrainte de cohérence locale s'applique à différents articles d'un même dictionnaire. Ces contraintes permettent de vérifier la cohérence d'un dictionnaire.
- *Cohérence globale :* une contrainte de cohérence globale s'applique à différents articles de différents dictionnaires dans une même base lexicale. Ces contraintes permettent de vérifier la cohérence globale de l'ensemble des dictionnaires dans une base lexicale.

Une contrainte contient trois parties principales :
- un patron qui spécifie l'ensemble des objets de la base de données qui sont concernés par cette contrainte,
- une expression booléenne qui doit être vérifiée par l'ensemble des objets concernés,
- une partie déclaration qui donne des informations supplémentaires sur la contrainte (message d'erreur, commentaire, niveau…).

Les contraintes d'intégrité et de cohérence locale sont associées à un dictionnaire. Les contraintes de cohérence globale sont associées à une base lexicale.

La définition d'une contrainte ne peut se faire que si l'on a auparavant défini l'architecture linguistique des différents dictionnaires. En effet, les expression d'extraction et les expressions booléennes portent sur des éléments des différentes structures linguistiques.

Avant de donner des exemples de contraintes, nous définissons donc l'architecture linguistique de la base lexicale auxquelles elles sont associées.

4.2. Structure de la base lexicale

La base lexicale sur laquelle portent nos exemples est basée sur une approche par transfert. Elle est composée de 3 dictionnaires monolingues (français, anglais, allemand) et des six dictionnaires bilingues correspondants :

```
(define-monolingual-dictionary french
     :language "Français"
     :owner    "GETA")

(define-bilingual-dictionary french-english
     :type   unidirectionnal
     :source french
     :target english
     :owner  "GETA")

(define-lexical-database example-database
     :owner   "GETA"
```

```
    :comment "Une base lexicale fondée sur une approche bilingue"
    :dictionaries
        (french english german
         french-english french-german ...))
```

L'entrée d'un dictionnaire monolingue est définie comme un arbre dont la racine est décorée par une structure de traits simple (contenant une catégorie et une forme graphique) et dont les feuilles sont décorées par des unités sémantiques (cette structure a été définie plus en détail dans le paragraphe 2.1.1. du chapitre I de la partie B):

```
(define-linguistic-class french-entry
   (tree :root   (feature-structure
                     (graphic-form string)
                     (category     cat))
         :leaves french-sem-unit))
(define-linguistic-class cat
   (one-of (nc np vb adj card deict repr sub coord)))
(def-linguistic-class french-sem-unit
  (feature-structure
    ((category cat)
     ;; information de dérivation.
     (drvv (feature-structure
            ((deriv-kind
              (one-of (naction nresult nlieu nagent ninstr adject adjpass
                       adjpotpas adjresact verbe)))
             (deriv-from sem-unit))))
     (drvn (feature-structure
            ((deriv-kind
              (one-of (ncond nlieu ninstr ncollect nperson adjrelat
                       adjqual verbe)))
             (deriv-from sem-unit))))
     (drva (feature-structure
            ((deriv-kind (one-of (nabst nperson verbe)))
             (deriv-from sem-unit))))
     ;; information sur les valences
     (val0 valency)
     (val1 valency)
     (val2 valency)
     (val3 valency)
     ;; autres informations
     (gnr (one-of (masc fem)))
     (nbr (one-of (sg pl)))
     (aux (one-of (être avoir)))
     (reciproque (one-of (arg0-arg1 arg1-arg2)))
     (aspect (one-of (achevé inachevé début fin duratif fréquent instantané)))
)))
(def-linguistic-class valency
  (set-of (nom à+nom avec+nom comme+nom contre+nom dans+nom de+nom en+nom
           entre+nom par+nom parmi+nom pour+nom sur+nom inf à+inf de+inf
           adj que+ind que+subj se-moy se-pass lieu-stat lieu-dyn manière
           zéro)))
```

L'entrée d'un dictionnaire bilingue est définie comme un lien reliant des entrées de dictionnaires monolingues, et décoré par une structure codant une condition et une action. Ces conditions sont codées comme de simples chaînes de caractères.

```
(def-linguistic-class french-english-transfer-link
  (link :source french::french-sem-unit
        :target english::english-sem-unit
        :label  french-english-transfer-info))
(def-linguistic-class french-english-transfer-info
  (feature-structure
   ((condition string)
    (action    string))))
```

4.3. Exemples de contraintes

On définit une contrainte par l'expression *define-coherence-rule* :

```
define-coherence-rule name
                      :applies-on    pattern
                      :verifies      boolean-expression
                      :error-message string
                      :level         level
```

Le patron (*pattern*) est la spécification d'une structure partielle contenant des variables. Ce patron désigne l'ensemble des structures du dictionnaire qui s'apparient avec lui, ainsi qu'un environnement où les variables définies dans la structure partielle sont valuées.

L'expression booléenne est une expression LISP, utilisant les variables définies dans l'expression d'extraction et retournant un booléen.

Le niveau est l'un des mot-clés : `:warning`, `:delay`, `:critical`.

Le message d'erreur est le message qui sera envoyé au linguiste si la contrainte est violée.

Pour spécifier le patron, qui désigne l'ensemble des structures sur lesquelles porte la contrainte, on dispose d'un moyen de dénoter une instance des classes linguistiques définies pour le dictionnaire. Nous verrons comment spécifier ce patron dans les différents exemples.

4.3.1. Contraintes d'intégrité

Les contraintes d'intégrité permettent de vérifier la bonne formation d'un article dans un dictionnaire.

Ce type de contraintes est défini au niveau du dictionnaire.

Dans l'exemple que nous donnons, l'attribut `category` est présent dans l'entrée et dans l'unité sémantique, et l'attribut `category` de l'unité sémantique doit avoir la même valeur que l'attribut `category` de l'entrée correspondante.

Cette contrainte d'intégrité s'applique à tous les types d'entrée. Le patron désigne donc l'ensemble des éléments de type *french-entry* dans la base. De plus, il nous faudra utiliser chacun de ces éléments. Donc, il nous faut associer une variable qui prendra pour valeur chacun des éléments tour à tour. Pour cela, on note le nom de la variable (précédé de @), suivi d'un patron représentant les éléments dont elle prendra la valeur.

L'expression booléenne doit vérifier, pour chacune des feuilles (les unités sémantiques) de l'entrée en cours de vérification, que l'attribut `category` de l'unité sémantique est égale à l'attribut `category` de l'entrée.

Cette contrainte s'exprime de la manière suivante :

```
(define-coherence-rule synchro-category
     :applies-on    (@Tree french-entry)
     :verifies      (let ((tree-decor (root Tree))
                          (cat (get-value tree-decor.category))
                          (result T))
                      (do-list (Usem (leaves Tree))
                        (setf result (and result
                                          (= cat (get-value Usem.category)))))
                      result)
     :error-message "Catégorie incompatible pour l'une des unités sémantiques"
     :level         :critical)
```

La seconde contrainte que l'on souhaite tester est liée aux attributs de dérivation de chaque unité sémantique. Ces attributs (`drvv, drvn, drva`) sont incompatibles (un seul d'entre eux peut être instancié à la fois).

Cette contrainte porte sur toutes les unités sémantiques.

L'expression booléenne doit vérifier qu'un seul parmi ces attributs est instancié. Lorsqu'un attribut n'est pas instancié, sa valeur est `:undef`.

Cette contrainte s'exprime de la manière suivante :

```
(define-coherence-rule only-one-drv
     :applies-on    (@Usem french-sem-unit)
     :verifies      (let ((drvv (get-value Usem.drvv))
                          (drvn (get-value Usem.drvn))
                          (drva (get-value Usem.drva)))
                      (cond ((not (= drvv :undef)) (and (= drvn :undef)
                                                        (= drva :undef)))
                            ((not (= drvn :undef)) (and (= drvv :undef)
                                                        (= drva :undef)))
                            ((not (= drva :undef)) (and (= drvv :undef)
                                                        (= drvn :undef)))
                            (T T)))
     :error-message "Deux dérivations pour une unité sémantique"
     :level         :critical)
```

La dernière contrainte d'intégrité que nous utiliserons porte aussi sur les dérivations. Le type de dérivation défini dépend de la catégorie de l'unité sémantique. Par exemple, un adjectif ne peut porter une information indiquant qu'il est produit par une dérivation de verbe vers nom.

Nous fractionnons cette contrainte en plusieurs contraintes simples selon la catégorie de l'unité sémantique de laquelle on dérive et de la catégorie de l'unité sémantique vers laquelle on dérive. Ainsi, le patron décrit les unités sémantiques ayant une même catégorie et dérivant d'une même catégorie d'unité sémantique. Nous donnons en exemple les adjectifs dérivant d'un verbe.

Il nous faut de plus manipuler le type de dérivation. Le patron spécifie donc une variable en valeur de l'attribut *deriv-kind*. Comme on n'impose pas de restriction sur la valeur de ce trait, seule la variable apparaît dans le patron (on ne dénote pas sa valeur).

L'expression booléenne vérifie que le type de dérivation est admis dans ce contexte :

```
(define-coherence-rule adj-drvv-coherence
     :applies-on    (french-sem-unit
                         [category : adj,
                          drvv : [deriv-kind : @kind]])
     :verifies      (or (= kind 'adject)
                         (= kind 'adjpass)
                         (= kind 'adjpotpas)
                         (= kind 'adjresact))
     :error-message "L'adjectif a une dérivation incompatible"
     :level         :critical)
```

4.3.2. Contraintes de cohérence locale

Les contraintes de cohérence locale permettent de vérifier la bonne formation de l'ensemble d'un dictionnaire. Ces contraintes portent donc sur plusieurs unités du dictionnaire.

Ce type de contrainte est défini de manière identique aux contraintes précédentes.

La contrainte que nous souhaitons définir vérifie que la catégorie de l'unité sémantique indiquée comme source d'une dérivation est compatible avec cette dérivation. En effet, si une

unité sémantique est le résultat d'une dérivation en provenance d'un verbe (resp. d'un nom, d'un adjectif), alors l'attribut *drvv* (resp. *drvn, drva*) sera spécifié et l'unité sémantique indiquée en valeur du trait *deriv-from* devra être un verbe (resp. d'un nom, d'un adjectif).

Nous définirons la contrainte vérifiant la cohérence pour l'attribut *drvv*.

Cette contrainte porte sur toutes les entrées qui ont une valeur pour l'attribut *drvv*.

L'expression booléenne suivant `:verifies` vérifie que l'unité sémantique indiquée en valeur de l'attribut *deriv-from* est un verbe :

```
(define-coherence-rule drvv-deriv-from-coherence
     :applies-on    (french-sem-unit
                         [drvv : [deriv-from : @source]])
     :verifies      (= (get-value source.category) 'vb)
     :error-message "Une dérivation verbale doit provenir d'un verbe."
     :level         :critical)
```

4.3.3. Contraintes de cohérence globale

Les contraintes de cohérence globale vérifient la bonne formation de l'ensemble de la base lexicale. Elles portent donc sur des unités de différents dictionnaires.

Ce type de contrainte est défini de manière analogue aux contraintes précédentes , mais porte sur différents dictionnaires. Le patron doit indiquer sur quel dictionnaire il s'applique. Pour cela, chaque classe linguistique indiquée sera notée, précédée du nom du dictionnaire et de "::".

Ce type de contrainte est défini au niveau de la base lexicale.

La contrainte de cohérence globale que nous souhaitons indiquer vérifie que l'unité sémantique, indiquée comme source sur un lien du dictionnaire *french-english*, existe bien dans le dictionnaire *french*.

Cette contrainte porte sur tous les liens de transfert du dictionnaire bilingue *french-english*.

L'expression booléenne vérifie l'existence de l'unité sémantique source dans le dictionnaire monolingue *french* :

```
(define-coherence-rule drvv-deriv-from-coherence
     :applies-on    (french-english::french-english-transfer-link
                         :source @french-sem-unit)
     :verifies      (exist? french::@french-sem-unit)
     :error-message "L'unité sémantique source du lien n'existe pas."
     :level         :critical)
```

5. Défauteur

Le but du défauteur est de donner des valeurs par défaut aux éléments des structures qui n'ont pas été renseignés par le lexicographe. Pour calculer les valeurs par défaut des différents éléments, le défauteur dispose de règles de calcul définies en faisant référence à la structure linguistique du dictionnaire.

Pour chaque dictionnaire, le linguiste peut définir un ensemble de règles de calcul produisant des valeurs probables pour différents éléments des structures linguistiques. Ces contraintes pourront être utilisées interactivement lors de l'édition d'une entrée (afin de faciliter le travail du lexicographe) ou bien être utilisées pour compléter des entrées importées ou partiellement indexées.

Après avoir précisé les notions utilisées par le défauteur, nous donnerons quelques exemples de règles de défaut.

5.1. Notions

Une règle de valeur par défaut contient trois parties principales :

- un patron qui spécifie l'ensemble des objets de la base de données qui sont concernés par cette règle,
- un test qui doit être vérifié pour que la règle s'applique,
- une expression qui associe une valeur à un des éléments de la structure linguistique. Cet élément ne prendra sa nouvelle valeur que s'il était indéfini auparavant (le linguiste a cependant un moyen de forcer l'affectation s'il le désire).

On définit une règle de valeur par défaut par l'expression *define-default-rule* :

```
define-default-rule name
                    :applies-on    pattern
                    :test          boolean-expression
                    :do            modifications
                    :redefine?     boolean
```

Le patron (*pattern*) a été défini dans la section précédente.

Le test est une expression booléenne. La règle ne s'appliquera que si cette expression est vérifiée.

La modification porte sur un et un seul élément de la structure linguistique.

Le mot-clé `:redefine?` indique si on force l'affectation lorsque la valeur est préalablement définie (par défaut, sa valeur est `false`).

5.2. Exemples de règles de valeurs par défaut

Les exemples suivants s'appliquent sur la structure utilisée dans la section précédente.

Notre premier exemple concerne le trait *category* de l'entrée *french-entry*. Pour calculer sa valeur par défaut, on utilise une règle heuristique, qui spécifie que les lemmes se terminant en "ence" sont probablement des noms.

Cette règle de défaut s'applique sur toutes les entrées du dictionnaire.

Le test porte sur la terminaison du lemme.

La modification porte sur le trait *category* de l'entrée.

```
(define-default-rule nominal-ending-ence
     :applies-on    (@entry french-entry)
     :test          (let ((decor (root entry)))
                       (suffix? (get-value decor.graphic-form) "ence"))
     :do            (assign decor.category 'nc))
```

Notre second exemple de règle de valeur par défaut permet de donner une valeur au trait *deriv-kind* d'une unité sémantique en s'appuyant sur le suffixe du lemme et sur sa catégorie. Ainsi, un nom dont le lemme se termine par "ement" est probablement un nom dérivé d'un verbe en tant que nom d'action.

Cette règle est complexe à écrire car elle manipule une entrée (afin de tester le lemme) et une des unités sémantiques qui lui sont associées. Les unités sémantiques sont les feuilles de l'arbre dont la racine est décorée par le lemme et la catégorie. Ces feuilles apparaissent à une profondeur variable dans les différentes entrées. Aussi, le patron doit sélectionner une feuille de l'arbre qui se trouve à une profondeur quelconque.

Pour cela, nous définissons un patron sur les arbres french-entry. Ce patron impose une contrainte sur la décoration de la racine (une décoration est indiquée entre accolades : "`{patron-sur-décoration}`"). Il sélectionne aussi une des racines de l'arbre french-entry. Pour cela, nous utilisons les notations suivantes :

- `@id` : dénote un arbre,

- `@*id` : dénote une forêt,
- `@/id` : dénote une multiforêt gauche,
- `@\id` : dénote une multiforêt droite,
- `@?id` : dénote un chemin dans un arbre (les nœuds d'un chemin sont séparés par des "."),
- `@!id` : dénote une feuille.

Si on ne souhaite pas conserver la valeur d'un de ces éléments, id sera la variable muette "-".

Le test vérifie que le lemme se termine par "ement".

La modification porte sur l'attribut *deriv-kind* de la dérivation verbale.

```
(define-default-rule noun+ement-naction
     :applies-on    (french-entry:(@root {[category : nc]}
                                   @?-.@!Usem))
     :test          (suffix? (get-value root.graphic-form) "ement"))
     :do            (assign Usem.drvv.deriv-kind 'naction))
```

La règle suivante donne une valeur par défaut à l'attribut *reciproque* de toutes les unités sémantiques. La valeur par défaut de cet attribut est *arg0-arg1*.

Cette règle porte sur toute les unités sémantiques.

Elle ne comporte pas de test.

Elle définit l'attribut réciproque s'il n'est pas défini.

```
(define-default-rule reciproque-default
     :applies-on    (@Usem french-sem-unit)
     :do            (assign Usem.reciproque 'arg0-arg1))
```

6. Import/Export

Un outil tel que SUBLIM ne peut exister sans un mécanisme lui permettant d'exporter les informations de sa base lexicale, ou d'importer des informations de dictionnaires existants.

Le mécanisme d'export doit permettre de générer un fichier export qui reflète la structure d'une base lexicale et qui soit utilisable par d'autres applications. Il doit permettre aussi de créer, à partir des informations d'une base lexicale, un structure d'export qui soit utilisable par d'autres applications.

L'utilisabilité d'un fichier d'export passe par deux points essentiels :
- le format du fichier doit être connu de l'application visée,
- la structure exportée doit être connue de l'application visée.

C'est pour pouvoir satisfaire ces deux points que nous avons choisi d'utiliser SGML (voir annexe A). SGML permet l'échange de documents dans des structures diverses. Afin d'utiliser un formalisme standardisé, nous utiliserons au maximum les entités et types de documents définis par la TEI (Text Encoding Initiative), notamment pour le codage des caractères. La TEI d'ailleurs offre des moyens standard de coder certaines des structures de données de base du système SUBLIM.

6.1. Notions

Nous définirons un export standard de chacune des structures de base du système afin de pouvoir exporter des données sans avoir à définir de structures particulières d'export. Ainsi, un mecanisme d'export des structures de traits sera fait vers le standard de codage des structures de traits de TEI.

Néanmoins, on peut prévoir que le linguiste voudra souvent générer une structure particulière, dépendant de l'application vers laquelle il exporte ses données. Cette structure particulière ne sera pas nécessairement le reflet de la structure (ou d'une partie de la

structure) du dictionnaire. C'est pourquoi il faut pouvoir disposer d'un mécanisme d'export assez sophistiqué.

Lors d'un processus d'export, on manipule deux structures différentes. La structure source est une structure linguistique définie en SUBLIM. La structure cible est une structure SGML. De même que l'on dispose d'une définition de la structure source (la définition de l'architecture linguistique de la base), on doit disposer d'une définition de la structure SGML. Cette définition est une Définition de Type de Document (DTD) SGML. Elle est le prérequis à toute opération d'export.

La définition d'une méthode d'export peut avoir différents aspects.

Dans certains cas, elle est présentée comme un moyen de *réécriture* d'une structure en une autre. Dans ce cas, on parcourt une structure source et, au fur et à mesure de ce parcours, on construit la structure cible.

Dans d'autre cas, elle est une *traduction* d'une structure source vers une structure cible. Dans ce cas, on définit des règles de traduction d'une structure vers une autre. Ici, ces règles sont données de façon déclarative, sans supposer aucun parcours particulier.

Nous considérons plutôt la définition d'une méthode d'export comme le *remplissage* d'une structure cible selon des informations prises dans la structure source. Dans ce cas, on parcourt la structure cible, et on calcule la valeur de chaque élément de la structure en fonction des valeurs trouvées dans la structure source. Cette méthode ne peut s'appliquer que si l'on a auparavant généré un squelette de la structure cible.

L'unité du lexique constituée par la structure d'export n'a pas de raison d'être la même que l'unité du lexique que l'on exporte.

Par exemple, le découpage en catégories des entrées du lexique défini et utilisé dans les sections précédentes est plus grossier que celui défini ici. Dans la structure d'export, un lemme comme "composer" a deux entrées (une pour le verbe transitif, l'autre pour le verbe intransitif) alors que le lexique source ne comporte qu'une entrée (puisqu'on ne fait pas la distinction entre verbe transitif et intransitif).

Pour cela, nous décomposons la procédure d'export en deux parties distinctes. La première étape permet la *création* du squelette formé par l'ensemble des unités du lexique d'export. Seules les informations nécessaires à la désignation d'une unité sont calculées à ce moment. Cette étape est réalisée par un ensemble de règles de réécriture.

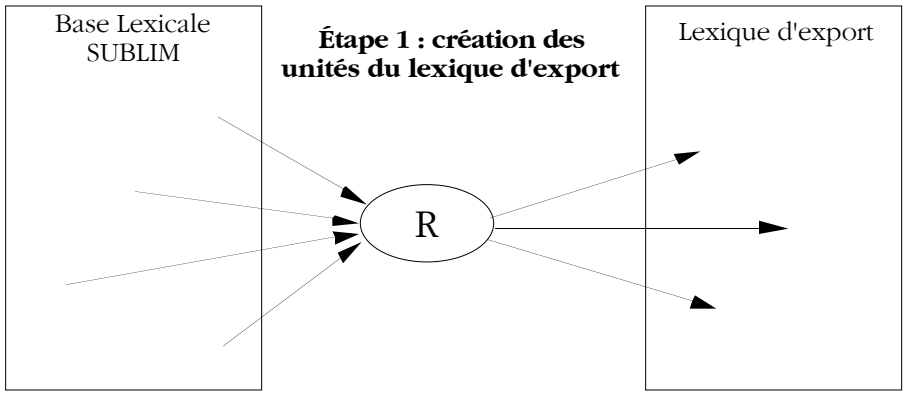


Figure 5.8 : Première étape du processus d'export

Pour la seconde étape, le point de départ est l'une des unités du lexique d'export. Cette étape a pour but le *remplissage* de chacune des unités du lexique d'export.

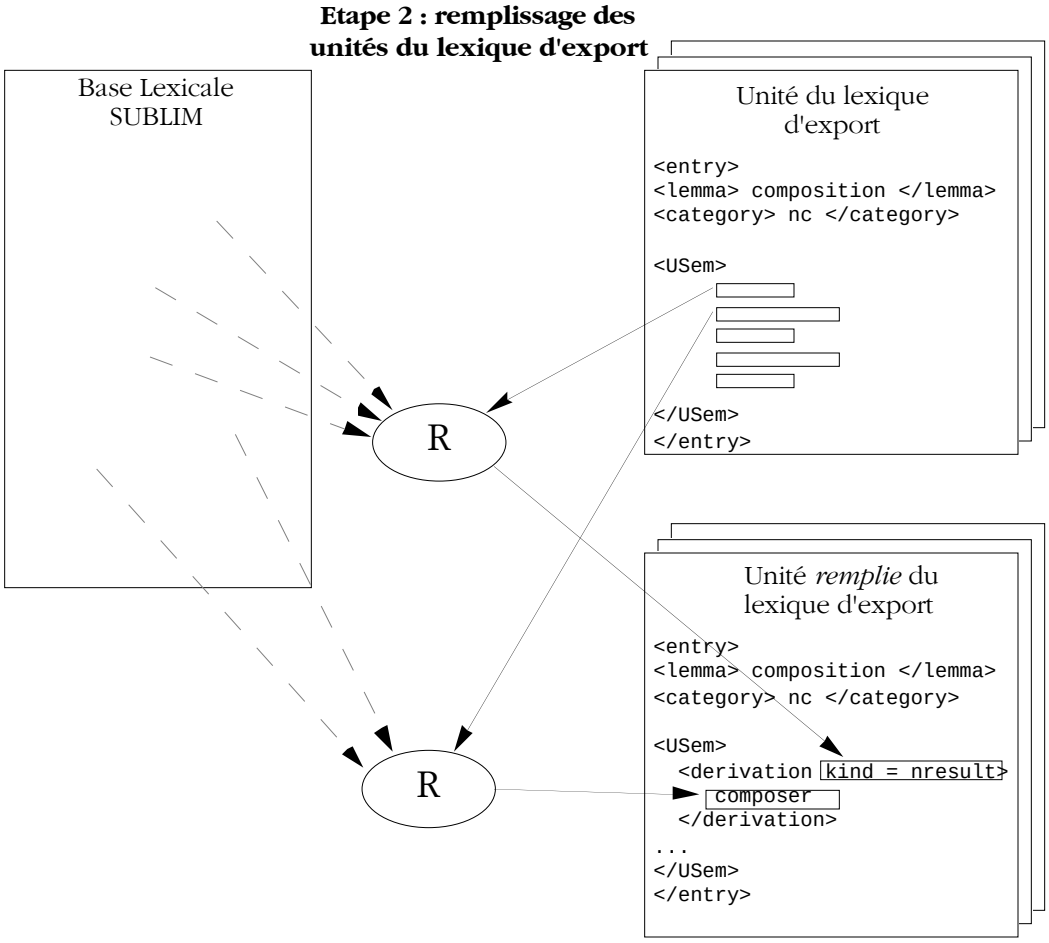


Figure 5.9 : Seconde étape du processus d'export

Ainsi, lorsqu'on définit ces règles de remplissage, on se situe dans le contexte d'une seule unité du lexique d'export. De plus, on se place dans le contexte d'un élément bien particulier dans la structure de cette unité. Ces règles sont associées à chaque élément de la structure d'export.

Il est possible de disposer d'une interface graphique pour définir les règles d'export. Cette interface permettra de visualiser la structure cible. Ainsi, on peut associer à chaque élément de la structure cible une méthode régissant la création et la forme de la valeur associée.

6.2. Exemple d'export

La structure de départ est la structure définie dans la section 4 de ce chapitre. La structure cible est définie par la DTD SGML suivante :

```
<!-- GETA-IMAG, 1994, export.dtd v.1.0 23/08/94 -->

<!-- DTD d'une structure d'export -->

<!--Un dictionnaire est une liste d'entrées. On lui associe aussi une langue-->
<!ELEMENT Dict       - -          entry* >
<!ATTLIST Dict       language     CDATA             #REQUIRED >

<!-- Une entrée est composée d'un lemme et d'une catégorie. -->
<!-- Elle est associée à une liste de sens -->
<!ELEMENT entry      - -          Usem* >
<!ATTLIST entry      lemma        CDATA             #REQUIRED
                     category     %cat              #REQUIRED >
<!ENTITY % cat "nc|np|vt|vi|adj|card|deict|repr|sub|coord">
```

```
<!-- Usem code une unité sémantique. -->
<!ELEMENT Usem       - -          (derivation & aux & reciproque) >

<!-- La dérivation donne le lemme source. 2 attributs sont définis -->
<!-- pour coder la catégorie du lemme source et le type de dérivation. -->
<!ELEMENT derivation - -          CDATA >
<!ATTLIST derivation source_cat   %cat              #REQUIRED
                     kind         %deriv_kind       #REQUIRED >
<!ENTITY % deriv_kind "naction|nresult|nlieu|nagent|ninstr|ncond|ncollect|
                       nperson|nabst|adject|adjpass|adjpotpas|adjresact|
                       adjrelat|adjqual|verbe">

<!ELEMENT aux        - -          EMPTY >
<!ATTLIST aux        value        %auxiliary        #REQUIRED >
<!ENTITY % auxiliary "être|avoir" >

<!ELEMENT reciproque - -          EMPTY >
<!ATTLIST reciproque value        %recipr           #REQUIRED >
<!ENTITY % auxiliary "arg0_arg1|arg1_arg2" >
```

Le but de la procédure que nous illustrons ici est d'exporter l'ensemble des noms et verbes d'un dictionnaire français dans la structure SGML définie ci-dessus.

La première étape de cette procédure consiste à créer les unités du lexique d'export. Ces unités sont désignées par leur lemme et leur catégorie. Ces deux attributs constituent l'ensemble minimal permettant de désigner une entrée de manière non ambiguë.

La création des unités du lexique source est faite selon les règles définies par :

```
create-export-unit name
                   :when    pattern
                   :if      boolean-expression
                   :create  creation
                   :context variable-list
```

où *name* identifie la règle de création. Le mot-clé *when* prend pour valeur un patron. Cette règle ne sera appelée que si une unité du lexique s'apparie avec lui. Le mot-clé *if* prend pour valeur une condition portant sur les variables définies dans le patron. La création n'aura lieu que si cette condition est vérifiée.

Le mot-clé *creation* prend pour valeur l'expression d'une structure d'export. Dans cette expression, on donne les élément SGML qui sont à créer. Les attributs SGML sont notés entre accolades à la suite de l'élément qui les porte et les valeurs sont notées sous forme de symboles ou de chaînes de caractères après les éléments et leurs éventuels attributs. Les sous-éléments sont notés entre parenthèses.

Le mot-clé *context* indique le contexte qui sera conservé pour l'étape suivante (remplissage).

L'attribut creation spécifie les valeurs qui identifient de manière unique une unité du lexique d'export. Si cette unité existe déjà, la création n'a pas lieu, mais le contexte de la règle s'ajoute au contexte de l'unité déjà existante.

La création des unités nominales se fait sans problème, puisqu'il y a correspondance directe entre les unités nominales des deux lexiques. La règle de création spécifie donc que, pour chaque unité nominale du lexique source, on crée une unité dans le lexique d'export :

```
(create-export-unit noun-creation
   :when    (@entry french-entry:(@- {[category : @cat (?or nc np),
                                       graphic-form : @lemma]}
                                    @*-))
   :create  (entry {lemma = @lemma, category = @cat})
   :context (@entry))
```

La création des unités verbales est plus complexe. En effet, on ne créera un verbe transitif que si l'une des unités sémantiques d'un verbe a une valence *val1* n'ayant que la valeur *nom*. Un verbe intransitif sera créé si l'une des unités sémantiques du verbe source considéré a une valence *val1* ayant une autre valeur que *nom*. Ces entrées ne sont créées que si elles n'existent pas encore.

```
(create-export-unit vt-creation
   :when    (@entry french-entry:(@- {[category : @cat vb,
                                       graphic-form : @lemma]}
                                  @?-.@!Usem))
   :if      (equal (get-value Usem.val1) '(nom))
   :create  (entry {lemma = @lemma, category = vt})
   :context (@entry @Usem))
(create-export-unit vi-creation
   :when    (french-entry:(@- {[category : @cat vb,
                                graphic-form : @lemma]}
                           @?-.@!Usem))
   :if      (not (equal (get-value Usem.val1) '(nom)))
   :create  (entry {lemma = @lemma, category = vi})
   :context (@entry @Usem))
```

Une fois que ces unités sont créées, il faut les compléter, à partir des informations présentes dans la base. Pour cela, on associe à chaque élément de la structure d'export, un ensemble de règles qui calculeront sa valeur, en se référant au contenu de la base lexicale.

Certains éléments doivent être créés avant d'être remplis. On aura donc, localement à une unité du lexique export, un ensemble de règles de création d'éléments. Ces règles sont définies grâce à :

```
create-export-element name
                      :on      pattern
                      :when    pattern
                      :if      boolean-expression
                      :create  creation
                      :context variable-list
```

où name est le nom de la règle.

Le mot-clé *on* prend pour valeur un patron qui s'appliquera sur l'unité d'export en cours de traitement. Ce patron sert à restreindre les unités sur laquelle porte la règle, aussi bien qu'à nommer des variables utilisées par la suite.

Le mot-clé *when* prend pour valeur un patron qui s'appliquera sur le contexte de l'unité du lexique d'export en cours de traitement. Cette règle ne sera appelée que si une unité du contexte s'apparie avec le patron. Le mot-clé *if* prend pour valeur une condition portant sur les variables définies dans le patron. La création n'aura lieu que si cette condition est vérifiée.

Le mot-clé *creation* prend pour valeur l'expression d'une structure d'export. Dans cette expression, on donne les élément SGML qui sont à créer. Les attributs SGML sont notés entre accolades à la suite de l'élément qui les porte et les valeurs sont notées sous forme de symboles ou de chaînes de caractères après les éléments et leurs éventuels attributs. Les sous-éléments sont notés entre parenthèses.

Le mot-clé *context* indique le contexte qui sera conservé pour le remplissage des sous-éléments de la structure cible.

Le premier élément à remplir est l'élément *Usem*. On associe donc à l'élément Usem certaines règles de création d'éléments.

```
(create-export-element Usem-filling
   :on      (entry {lemma = @lemma, category = nc})
   :when    (@entry french-entry:(@- {[category : @cat nc,
                                       graphic-form : @lemma]}
```

```
                                        @?-.@!Usem))
   :create  (entry (Usem))
   :context (@Usem @entry))
```

À l'intérieur d'une Usem, on remplit ensuite l'élément de dérivation.

L'élément dérivation de l'unité sémantique ne sera créé que si l'un des attributs *drvv*, *drva* ou *drvn* est présent dans l'unité sémantique correspondante dans le lexique source. On associe donc associer la règle suivante à l'élément dérivation :

```
(create-export-element derivation-filling
   :on      (Usem)
   :when    (@Usem french-sem-unit:[drvv : [deriv-kind : @kind,
                                            deriv-from : @from]])
   :create  (derivation {kind = @kind, source_cat = vb} (lemma @from)))
```

où lemma est une fonction (à définir) qui retourne le lemme associé à l'unité sémantique.

Le patron exprimé en valeur du mot-clé *when* s'unifiera dans le contexte qui aura été conservé à la création de l'unité *Usem*. La variable *@Usem* sera donc associée à l'unité sémantique du dictionnaire source qui aura motivée la création de l'Usem.

Cette règle illustre la difficulté de la définition d'un mécanisme d'export. En effet, elle comporte un problème qui ne peut être résolu avec le mécanisme d'export ainsi défini.

Ce problème porte sur l'attribut *source_cat*. Cet attribut prend une catégorie en valeur. Or les catégories des deux lexiques sont différentes. Dans le cas d'un verbe, on ne sait pas si la catégorie du lemme source de la dérivation est vi ou vt (puisque cette distinction n'est pas faite au niveau du lexique de départ). Pour connaître cette valeur, il faut savoir à quelle entrée du lexique d'export correspond l'unité sémantique indiquée en source de dérivation.

Dans le cas général, rien ne garantit qu'une telle correspondance existe.

Le mécanisme, tel qu'il est présenté ici, permet donc de réaliser des exports "guidés par la structure cible", ce qui simplifie l'écriture des règles lorsque les correspondances entre les deux structures ne sont pas aisées à établir. Ce mécanisme présente certains problèmes et mérite une étude beaucoup plus approfondie.

Spécialisation à l’interlingue par acceptions

Introduction

Le système SUBLIM précédemment décrit est générique. Il n'a donc pas d'a priori, ni sur l'architecture lexicale d'une base, ni sur l'architecture linguistique de ses dictionnaires. On peut donc l'utiliser pour créer des instances de bases lexicales ayant des fondements théoriques différents.

En contrepartie, il ne peut fournir qu'une aide générale pour la gestion de bases lexicales, qu'elles soient monolingues, multilingues par transfert, ou interlingues.

Cependant, son implémentation par objets le rend susceptible d'être spécialisé pour la gestion de bases ayant une architecture lexicale particulière.

Les recherches sur la traduction automatique fondée sur le dialogue [Blanchon 1992, Blanchon 1994, Boitet 1990b, Boitet & Blanchon 1993] se développent depuis quelques années. Dans cette approche, on demande à l'auteur d'interagir pour aider l'ordinateur à lever les ambiguïtés de son texte. Cette interaction n'a lieu qu'une fois pour le texte source, quel que soit le nombre de langues vers lesquelles on veut traduire. L'idée de base est que l'auteur acceptera de passer du temps à cette interaction s'il obtient des traductions dans plusieurs langues cibles et si le dialogue est assez ergonomique et compréhensible.

Dans le cadre de l'Union Européenne, qui reconnaît 9 langues officielles, les besoins en bases lexicales multilingues regroupant (au moins) ces 9 langues sont cruciaux.

Il est donc important de développer des bases lexicales multilingues regroupant de nombreuses langues. Dans ce contexte, une approche interlingue prend tout son sens.

Comme nous l'avons signalé dans la première partie, les approches interlingues sont souvent confondues avec les approches "ontologiques" (fondées sur la connaissance du ou des domaines de discours). Or, une approche fondée sur la connaissance rend difficile la gestion d'une base lexicale. En effet, aux difficultés inhérentes aux différentes langues de la base s'ajoutent les difficultés propres à la représentation des connaissances (description du sens, classification des concepts…).

Nous préférons une approche plus linguistique, défendue aussi bien par des linguistes "purs" (comme Igor Mel'čuk [Mel'čuk 1984, Mel'čuk 1988, Mel'čuk 1992]) que par des spécialistes

d'Intelligence Artificielle et du Traitement Automatique des Langues Naturelles (comme Yorick Wilks avec le projet ULTRA [Farwell, Guthrie & Wilks 1992, Farwell & al. 1993]). Pour nous, les éléments "pivots" sont alors des "acceptions interlingues".

Dans cette partie, nous définirons les principes de cette approche en la situant par rapport aux approches interlingues classiques. Nous montrerons ensuite que cette approche est propice à l'utilisation de fonctions lexico-sémantiques "à la Melčuk". Enfin, nous décrirons la maquette PARAX, une première expérimentation de cette approche. Nous conclurons en dégageant de cette étude les aspects génériques qui devront être implémenté dans NADIA, un système spécialisé dans la gestion de telles bases lexicales.

VI. L'approche par acceptions

1. Acceptions et concepts

L'approche par acceptions est fondée sur une architecture lexicale interlingue. Contrairement à la grande majorité des bases lexicales fondées sur un interlingue, cette architecture ne se fonde pas sur une représentation des connaissances.

L'aspect original de cette architecture lexicale repose sur le fait que les unités du dictionnaire interlingue sont des *acceptions*.

Une acception est un sens particulier d'un mot, admis et reconnu par l'usage. Il s'agit donc d'une unité sémantique propre à une langue donnée. Cette notion d'acception (appelée parfois sémantème), n'est pas nouvelle dans le domaine des dictionnaires monolingues. En fait, l'acception est bien souvent l'unité d'un lexique monolingue.

Ce qui est original, c'est l'utilisation de cette notion dans un contexte interlingue. Ce qui est difficile, c'est que rien ne garantit qu'une acception dans une langue corresponde à une acception identique (mais liée à un lemme différent) dans une autre langue.

1.1. Acceptions

1.1.1. Notion d'acception

Une acception monolingue est une unité sémantique d'une langue. Elle est locale à une langue de la base. Ainsi, les ensembles des acceptions monolingues de deux langues différentes sont différents, même si leur intersection n'est pas vide.

Le but essentiel de la base lexicale est de fournir un lien entre les acceptions monolingues des différents dictionnaires. Pour cela, nous définissons l'ensemble des acceptions interlingues comme étant l'union des ensembles d'acceptions monolingues des différents dictionnaires de la base, l'opération d'égalité sur les acceptions étant l'identité sémantique.

Ainsi, une acception monolingue correspond à une et une seule acception interlingue. Par contre, une acception interlingue correspond à une ou plusieurs acceptions monolingues de dictionnaires différents.

En faisant l'hypothèse (idéale) que les langues de la base utilisent un raffinement sémantique identique, on obtiendra une organisation illustrée par la figure 6.1.

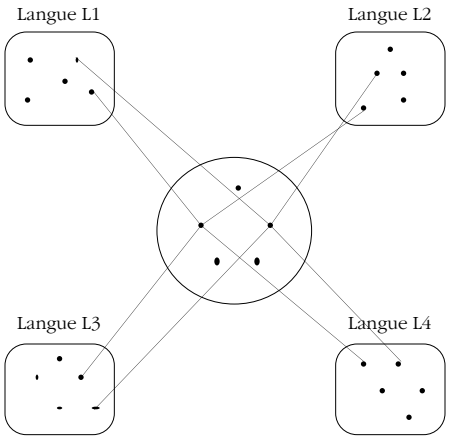


Figure 6.1 : L'interlingue par acception dans des conditions idéales

1.1.2. Relations entre acceptions

Hélas, ces conditions ne sont jamais réalisées, et les problèmes de raffinement de concepts entre différentes langues ne peut être ignorés. Les exemples sont nombreux. Par exemple, entre le français et l'anglais, on peut noter la différence de raffinement entre les acceptions de *rivière* et *fleuve*, d'une part et celle de *river*, d'autre part. En français, on fait une différence entre les cours d'eau se jetant dans la mer et les autres (*fleuve/rivière*), alors qu'en anglais, on ne la fait pas (*river*).

Les acceptions correspondantes n'étant pas sémantiquement identiques, nous obtiendrons une organisation illustrée en figure 6.2.

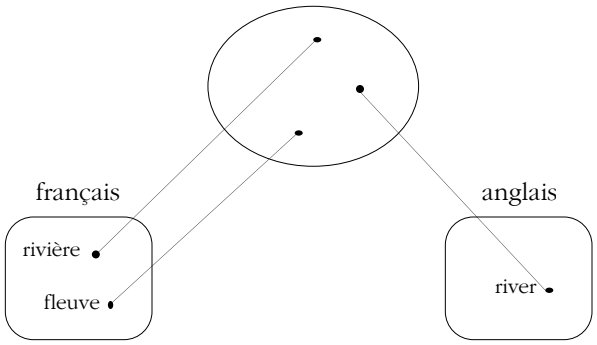


Figure 6.2 : Acceptions interlingues pour *rivière, fleuve* et *river*

Pour que le lien entre acceptions soit conservé (afin de pouvoir traduire ces termes), nous utiliserons un lien entre acceptions interlingues nommé lien de *raffinement.* Ainsi, l'organisation obtenue est illustrée en figure 6.3.

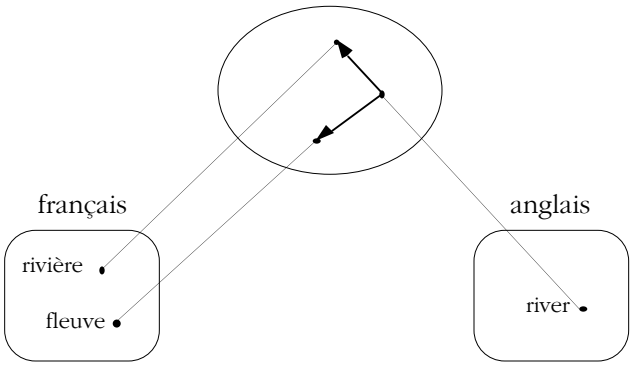


Figure 6.3 : Acceptions interlingues pour *rivière, fleuve* et *river*, avec liens de raffinement

Dans cet exemple, le lien de raffinement correspond à un raffinement sémantique. En effet, la sémantique de l'acception *river* correspond à l'union des sémantiques des acceptions *rivière*

et *fleuve*. Ce n'est pas toujours le cas. En effet, des problèmes contrastifs d'origine non sémantique peuvent se présenter. Prenons l'exemple en chinois de 工作 (gōngzuò). Certains dictionnaires (comme le "dictionnaire français de la langue chinoise" [Ricci 1986]) associent deux acceptions à cette entrée :

- travailler (de ses mains), travail manuel;
- travailler (en général), travail, occupation, besogne.

Comme nous le voyons, ces acceptions ont toutes deux à la fois un aspect nominal et un aspect verbal. Si on veut les relier aux acceptions françaises correspondantes, on doit choisir entre l'acception correspondante de *travail* (nominal) ou l'acception correspondante de *travailler* (verbal). On a donc bien un problème contrastif à résoudre. Les solutions à ce problème sont les suivantes :

- **On modifie le dictionnaire chinois** : cette solution ne doit être envisagée que si l'on considère que l'entrée, telle qu'elle apparaît dans le dictionnaire monolingue, est erronée.
- **On lie l'acception chinoise arbitrairement à l'une des acceptions du français (supposons l'acception verbale)** : on ne pourra donc traduire qu'à condition de disposer de relations de dérivation permettant de nominaliser l'acception française lorsque cela est nécessaire.

Chaque dictionnaire monolingue est indépendant des autres langues présentes dans la base. On ne peut donc envisager la première solution que dans le cas où l'on a détecté une erreur, ce qui n'est pas le cas général.

L'approche par acceptions ne peut faire des hypothèses a priori sur les informations que l'on trouve dans les dictionnaires monolingues. On ne peut donc pas présupposer l'existence d'un lien de dérivation dans ces dictionnaires.

Aussi, la solution de ce problème passe, dans le cas général, par le lien de raffinement. On obtiendra donc la configuration indiquée dans la figure 6.4.

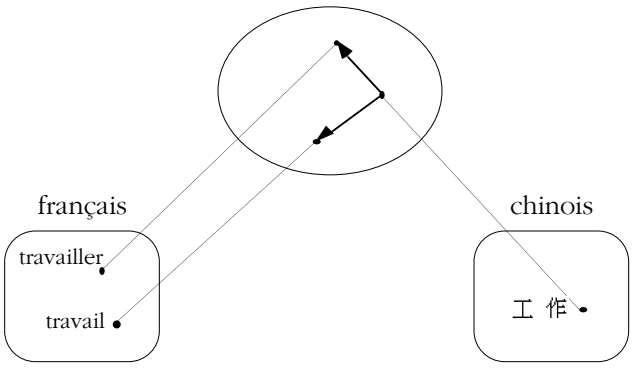


Figure 6.4 : Un exemple de lien de raffinement motivé par un phénomène contrastif non sémantique

Dans cette configuration, le lien de raffinement a une interprétation linguistique et non sémantique.

1.2. Concepts

Nous l'avons vu, les unités du lexique interlingue sont des acceptions, et non des "concepts", tels qu'on les rencontre en général dans les bases lexicales fondées sur la connaissance. Mais la différence entre les deux approches ne réside pas uniquement dans leur vocabulaire.

L'approche par connaissances se base sur l'hypothèse qu'il existe un niveau sémantique universel, indépendant des langues, et que les langues sont des reflets de ce niveau.

Ainsi, une base lexicale interlingue fondée sur la connaissance part d'un ensemble de dictionnaires de langues :

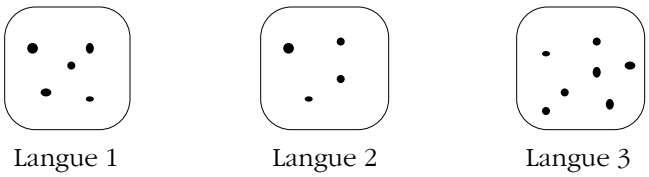


Figure 6.5 : Un ensemble de dictionnaires monolingues

À cet ensemble de dictionnaire, on ajoute un nouveau dictionnaire représentant un reflet du niveau sémantique universel (la connaissance) :

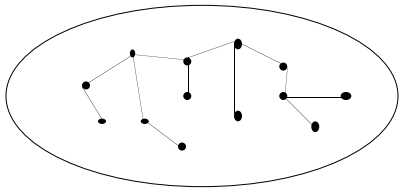


Figure 6.6 : Une base de connaissances

Le but est donc de relier les unités des différentes langues aux unités de la base de connaissances :

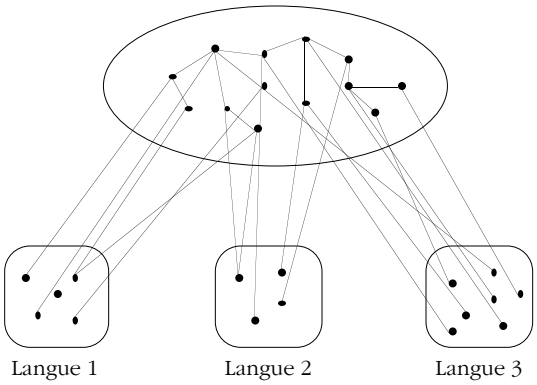


Figure 6.7 : Une base lexicale fondée sur la connaissance

Dans cette approche, la base interlingue est une modélisation du monde, suffisante pour permettre des calculs sur la sémantique des concepts manipulés dans le domaine.

Selon l'approche par connaissances, le dictionnaire interlingue est indépendant de l'ensemble des langues de la base. En théorie, l'ajout d'une langue ne devrait pas modifier son contenu.

L'exemple le plus représentatif de ce type d'approche a été donné par le projet KBMT [Goodman & Nirenburg 1991] développé au *Center for Machine Translation* de *Carnegie Mellon University*.

Ce projet utilisait une représentation du monde des ordinateurs personnels (appelée ontologie), et un outil d'acquisition et de maintenance des connaissances nommé ONTOS.

Le dictionnaire de concepts de KBMT est une représentation conceptuelle, indépendante des langues, des interactions entre un ordinateur personnel et son utilisateur.

Cette ontologie est représentée comme un réseau interconnecté et hierarchisé de *frames*, chacune représentant un concept en ONTOS.

1.3. Variantes et discussion

1.3.1. Différences entre approche par acceptions et approche par connaissances

Une base lexicale par acceptions part d'un état analogue : on dispose d'un ensemble de dictionnaires monolingues. Par contre, on ne dispose pas d'un dictionnaire interlingue. Il faut donc créer des liens entre les unités des différents dictionnaires de la base. Cette création se traduit par la construction d'une base interlingue par acceptions.

Cette base interlingue n'ayant aucune prétention de représentation des connaissances, on n'a donc pas à créer et à gérer un dictionnaire supplémentaire reflétant un langage abstrait.

Les unités et les relations entre acceptions dépendent des langues de la base. L'ajout d'une langue modifie donc quasi-obligatoirement le contenu du dictionnaire interlingue. De plus, la méthodologie de création, et notamment l'ordre dans lequel on établit les correspondances, peut influer sur le contenu du dictionnaire interlingue.

Cette approche n'apportant pas d'information sémantique, elle se justifie lorsque le nombre de langues de la base est suffisamment important pour que le coût de développement d'un lexique interlingue soit moins élevé que le coût de développement d'un grand nombre de dictionnaires bilingues.

1.3.2. Variantes de l'approche par acceptions

L'approche par acceptions est utilisée dans certains projets de bases lexicales. Le premier de ces projets, le projet ULTRA, est mené au Computing Research Laboratory de la New Mexico State University.

ULTRA (Universal Language TRAnslator) est un système de traduction interlingue qui implique actuellement cinq langues (anglais, allemand, chinois, espagnol et japonais). Ce système manipule un vocabulaire d'environ 10 000 acceptions (avec 6 000 à 7 000 mots dans les différents dictionnaires monolingues).

Les dictionnaires d'ULTRA contiennent des acceptions interlingues (appelés INTER TOKEN, et représentées dans la dernière partie de la figure 6.8) et des entrée monolingues (représentées dans les cinq premières parties de la figure 6.8). Les acceptions interlingues sont réparties en catégories (entités, relations, spécifieurs d'entité, spécifieurs de relation, etc.).

Chaque entrée des dictionnaires monolingue est une clause Prolog où le prédicat correspond à la catégorie de l'entrée, le premier argument est généralement la forme et le dernier, l'identificateur de l'acception correspondante.

Ainsi, le mot *novela* (roman) en espagnol a la forme suivante :

```
se_form(novela, ts, f, _Case, novel2_0)
```

Cette forme espagnole est donc associée à l'acception *novel2_0*. Cet identificateur est lui même associé à une clause du dictionnaire interlingue. De plus, cette acception est associée à un sens du dictionnaire LDOCE.

ULTRA utilise donc une variante de la notion d'acceptions que nous avons définie, dans le sens où ses acceptions interlingues sont les acceptions du dictionnaire anglais LDOCE, et non l'union des acceptions des langues de la base.

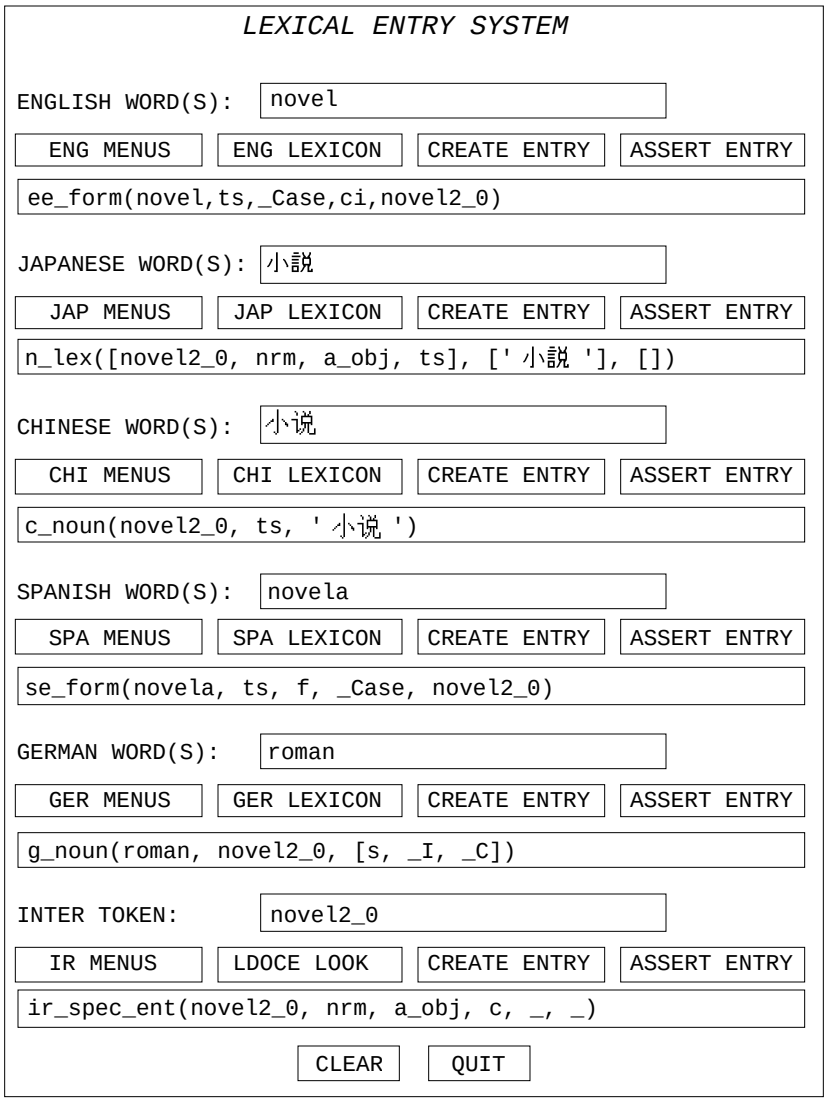


Figure 6.8 : Le système de gestion lexicale d'ULTRA

Un deuxième projet utilisant la notion d'acception est le projet EDR. Cela semble paradoxal pour un projet qui utilise un "dictionnaire de concepts" et représente des informations sémantiques par des liens entre les concepts (voir chapitre II).

Pourtant, les unités du lexique interlingue produit par EDR sont plus proches des acceptions que des concepts. En effet, ces unités sont créées en référence à un sens de mot dans l'un des dictionnaires monolingues de la base, et non en référence à une notion sémantique du monde.

La classification des "concepts" n'a pas de caractère encyclopédique, mais est motivée par des considération de factorisation de certaines relations entre unités du lexique interlingue.

De plus, EDR ne présente pas de véritables liens sémantiques entre les unités du lexique interlingue. Les liens utilisés ont des aspect plus lexicaux que sémantiques.

Par exemple, les liens *object*, *a-object*, *agent*, *cause*... portent bien une information sémantique, mais dans le contexte d'une base interlingue, ils sont plus utiles en étant interprétés en tant que liens de collocation. Ainsi, la relation «manger» — agent → «animal» s'interprète comme une relation de connotation entre les acceptions «manger» et «animal».

1.3.3. Méthodologie de création

Le dictionnaire d'acceptions interlingues est un moyen de lier les différentes unités des dictionnaires monolingues. La création de ce dictionnaire ne se base pas sur une représentation du monde. Il nous faut donc créer une méthodologie de création et de gestion des unités de ce dictionnaire.

La méthodologie que nous proposons se base sur la définition de liens bilingues.

La construction d'une base lexicale multilingue se passe en deux temps. Dans un premier temps, le lexicographe définit les informations monolingues pour un certain nombre d'entrées du dictionnaire. Ensuite, il donne différentes traductions pour ces entrées.

La première étape dans la construction du dictionnaire d'acceptions est la création des acceptions interlingues correspondant aux acceptions monolingues d'un dictionnaire particulier (voir figure 6.9.). Ce dictionnaire particulier sera appelé dictionnaire de référence.

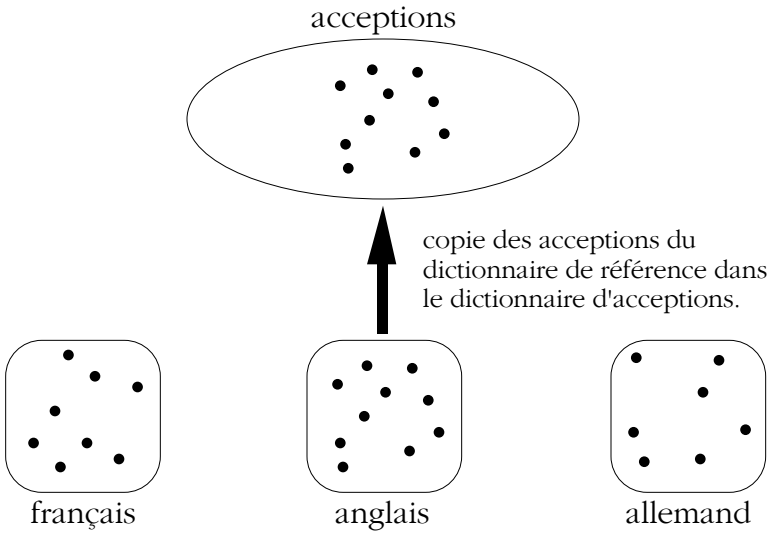


Figure 6.9 : Création du dictionnaire d'acceptions, première étape.

On relie ensuite les acceptions monolingues des autres dictionnaires aux acceptions interlingues ainsi créées. Pour cela, un lexicographe fournit des informations bilingues reliant les unités de deux dictionnaires monolingues. L'une au moins des unités liées doit être auparavant associée à une acception interlingue.

Prenons l'exemple d'une base lexicale français-anglais-allemand, et choisissons le dictionnaire anglais comme dictionnaire de référence. L'ensemble des acceptions du dictionnaire anglais correspond donc à des acceptions interlingues (voir figure 6.9.).

Le lexicographe donne une traduction de l'acception courante du mot français *rivière* en anglais. Il donne donc le mot anglais *river* comme traduction de *rivière*.

Le système lui demande de choisir parmi les acceptions de *river* celle qui correspond au sens de *rivière*. Le système établit donc un lien entre l'acception française de *rivière* et l'acception interlingue correspondant à l'acception anglaise de *river*.

Ainsi, l'information bilingue donnée par le linguiste a permis d'établir un lien interlingue. Après cette étape, la base de données présente la structure illustrée par la figure 6.10.

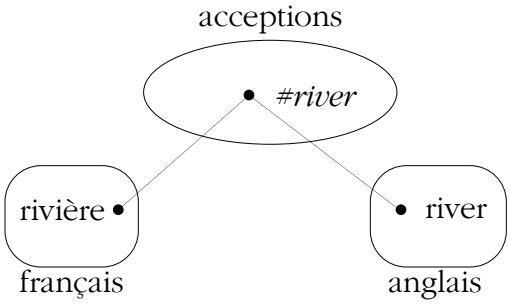


Figure 6.10 : État de la base lexicale après avoir lié *rivière*, avant d'avoir lié *fleuve*

Cette structure, bien qu'erronée (puisque les acceptions de *rivière* et *river* ne recouvrent pas la même sémantique), est cohérente avec le contenu de la base. En effet, tant que le mot *fleuve* n'a pas été introduit dans le dictionnaire ou lié à l'interlingue, il n'y a aucune raison de scinder l'acception associée à *rivière* et *river,* puisque ces mots sont toujours traduction l'un de l'autre.

Ce problème contrastif apparaîtra à l'introduction du mot français *fleuve*. Lorsque le lexicographe indique que *fleuve* se traduit par *river*, le système devra détecter que deux acceptions françaises sont associées à la même acception interlingue. Cela n'est possible que si ces deux acceptions sont parfaitement synonymes.

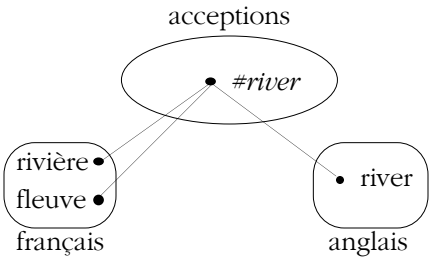


Figure 6.11 : Configuration illicite détectée par le système

Or, les acceptions de *rivière* et de *fleuve* ne recouvrent pas le même sens (en fait, les cas de synonymie parfaite son rares). Le système doit détecter ce genre de configuration et demander au lexicographe de résoudre le problème. Dans cet exemple, il y a quatre solutions possibles :

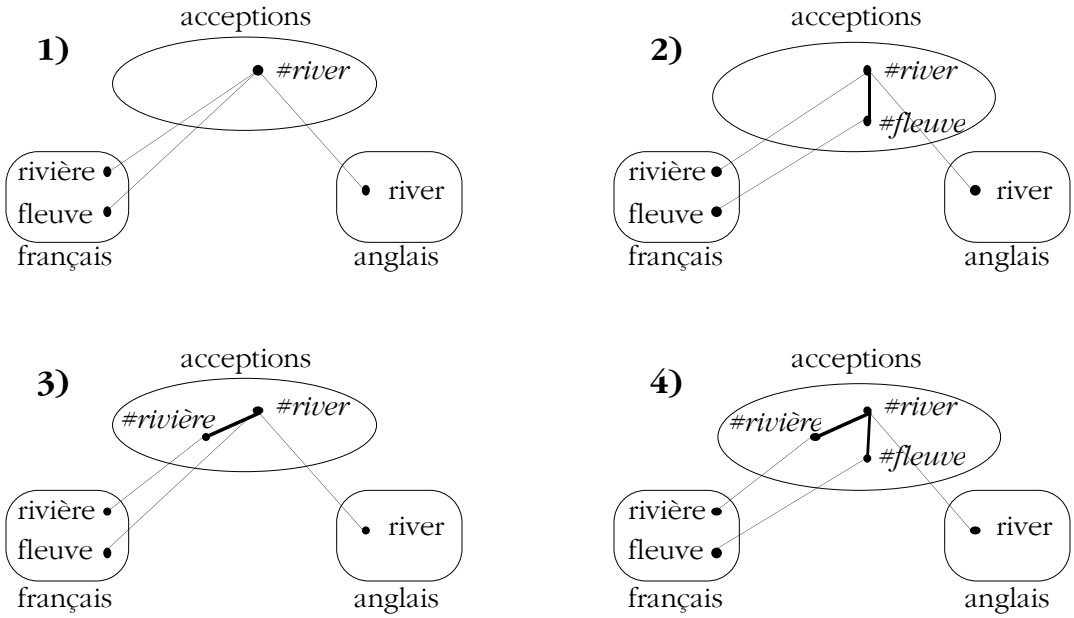


Figure 6.12 : Les différentes solutions aux problèmes contrastifs.

Dans l'exemple considéré, les sens de *river*, *rivière* et *fleuve* sont distincts deux à deux. Le système créera donc deux nouvelles acceptions interlingues correspondant à *rivière* et *fleuve*, et reliées par un lien de raffinement à l'acception de *river* (solution n° 4).

2. Acceptions et fonctions lexicales

Notre approche est très bien adaptée à l'utilisation de fonctions lexico-sémantiques dans les dictionnaires monolingues, aussi bien que dans les dictionnaires bilingues.

Le principe des fonctions lexicales développées par Igor Mel'čuk à Moscou puis à Montréal repose sur la constatation que certains mots dans un texte n'ont pas une valeur dénominative, mais ne sont présents que pour modifier d'autres mots proches.

Ainsi, lorsque l'on parle d'une "forte fièvre", le mot *forte* apparaît comme intensifieur de *fièvre*. Pour pouvoir traduire un texte, il faut être capable d'identifier la fonction de ces mots. En effet, "forte fièvre" ne peut être traduit par "strong fever", mais par "high fever". On traduit donc la fonction du terme, plutôt que le terme lui-même.

Le Dictionnaire Explicatif et Combinatoire (DEC) indique que l'intensifieur de *fièvre* est *forte*. Cette indication est donnée sous forme d'une fonction (Magn), appliquée à *fièvre* et dont le résultat est *forte*.

Igor Mel'čuk et ses collègues ont recensé 52 fonctions lexicales. Certaines de ces fonctions peuvent même être modifiées par l'ajout d'indices ou d'exposants (pris parmi un ensemble fini).

Ces fonctions lexicales ont été étudiées sur de nombreuses langues (russe, polonais, français, anglais, espagnol, allemand, et moins systématiquement sur le japonais, tatare, hongrois, chinois) et elles apparaissent actuellement comme universelles.

L'approche par acceptions utilise les sens de mots. Elle manipule donc les mêmes unités que les dictionnaires utilisant les fonctions lexico-sémantiques. De plus, les fonctions lexicales sont des liens formant un réseau reliant les unités d'un dictionnaire. L'approche par acceptions passe par l'utilisation d'au moins un lien entre unités du dictionnaire : le lien de raffinement.

Cette approche ne restreint pas les informations linguistiques des dictionnaires monolingues. Néanmoins, elle est propice à l'utilisation de fonctions lexicales.

Les fonctions lexicales définies par Igor Mel'čuk et ses collègues sont instanciées au niveau des dictionnaires monolingues. Par l'utilisation d'un système de bases lexicales fondées sur une approche par acceptions, il est possible d'étudier les moyens permettant de reporter et/ou de refléter certaines de ces fonctions lexico-sémantiques au niveau du lexique interlingue.

3. PARAX, une expérimentation

Afin d'expérimenter l'approche par acceptions, Étienne Blanc a construit une maquette de base lexicale interlingue par acceptions. Cette maquette a été implémentée avec HyperCard™ sur Macintosh™, et une version a été portée sur le gestionnaire de bases de données 4D™.

Dans ce paragraphe, nous présentons la version HyperCard de cette maquette.

3.1. Les dictionnaires monolingues

Les dictionnaires monolingues sont accessibles par la liste des lemmes présents. Un lemme nous mène à un écran où se trouvent les différentes acceptions associées (figure 6.13.). Le dictionnaire monolingue se compose de deux parties :

- les informations linguistiques associées à chaque acception (dans la colonne de gauche),
- une recopie de l'acception interlingue correspondant à chaque acception (dans la colonne du milieu).

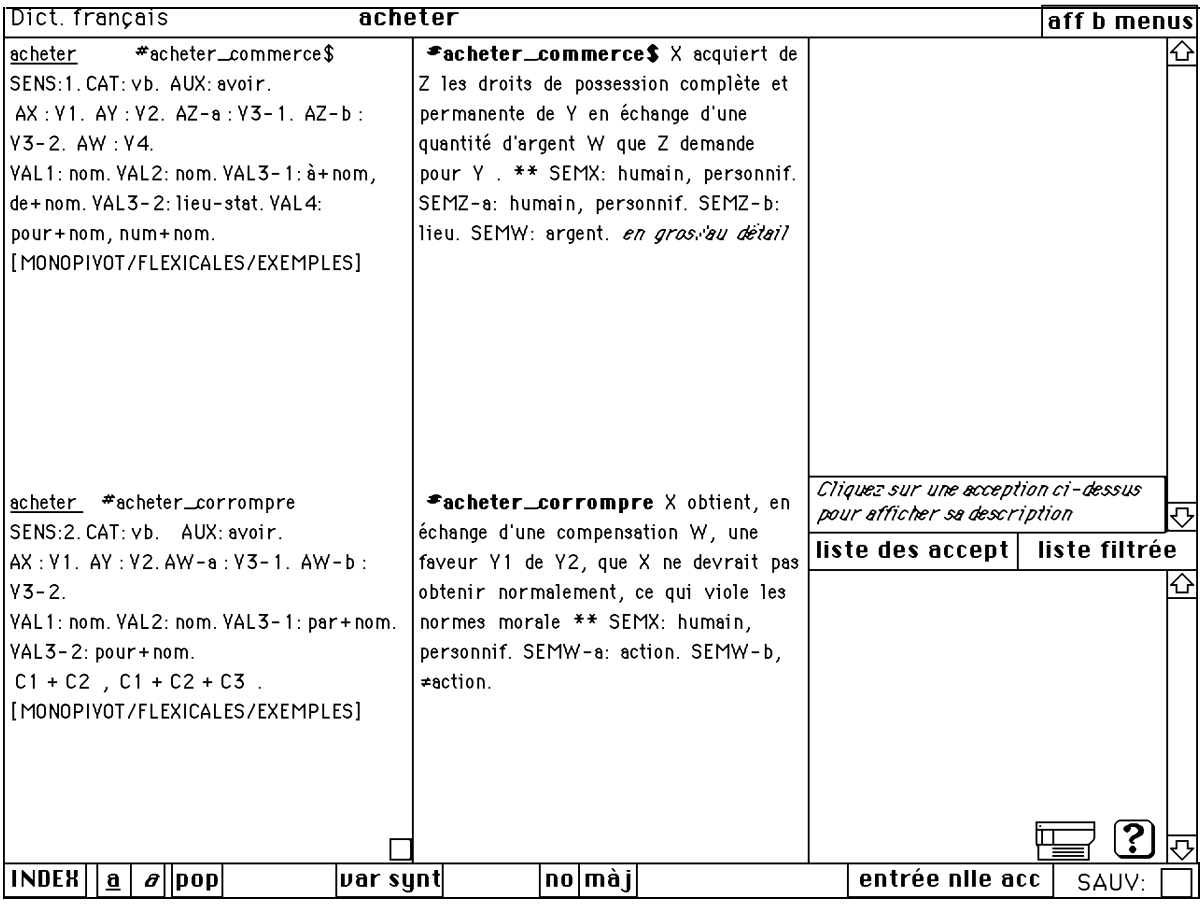


Figure 6.13 : Le dictionnaire monolingue de PARAX

L'acception est associée à une information linguistique qui indique sa structure argumentaire et le régime de ces arguments.

L'acception interlingue associée à une acception monolingue contient une définition dans la langue du dictionnaire. Elle contient de plus un ensemble d'informations sémantiques régissant la sémantique des arguments.

Il est possible de consulter le dictionnaire interlingue d'acceptions en cliquant sur le bouton `MONOPIVOT`, pour l'acception considérée.

3.2. Le dictionnaire interlingue

L'accès au dictionnaire interlingue se fait soit à travers la liste des acceptions définies, soit via un dictionnaire monolingue.

Lorsque l'on arrive à ce dictionnaire via un dictionnaire monolingue, on retrouve l'acception monolingue par laquelle s'est fait l'accès (colonne de gauche de la figure 6.14.). Dans la colonne centrale se trouve l'acception interlingue avec ses éventuelles sous-acceptions. Ainsi,

l'acception "#acheter_commerce" a une sous-acception issue du chinois. En effet, le chinois introduit une acception particulière correspondant à "acheter en gros".

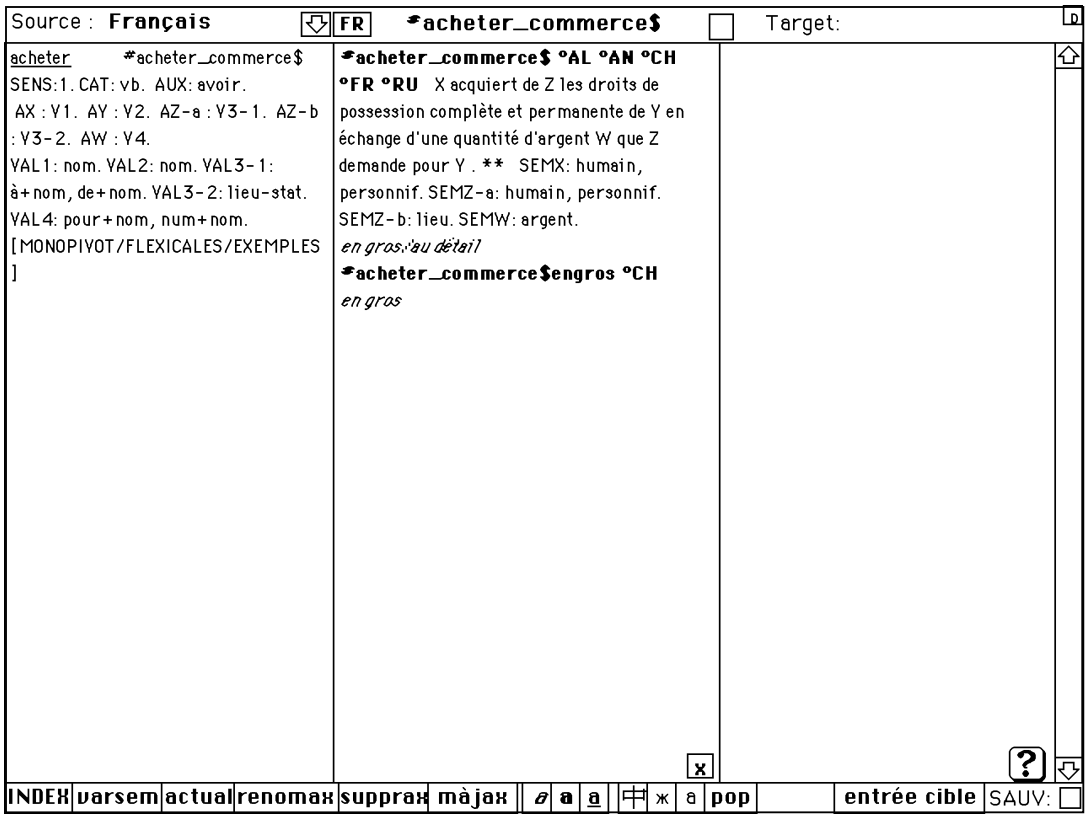


Figure 6.14 : L'acception interlingue "#acheter_commerce"

En regard de chaque acception se trouvent des liens vers les acceptions monolingues correspondantes. En cliquant dessus, on obtient ces acceptions dans la colonne de droite.

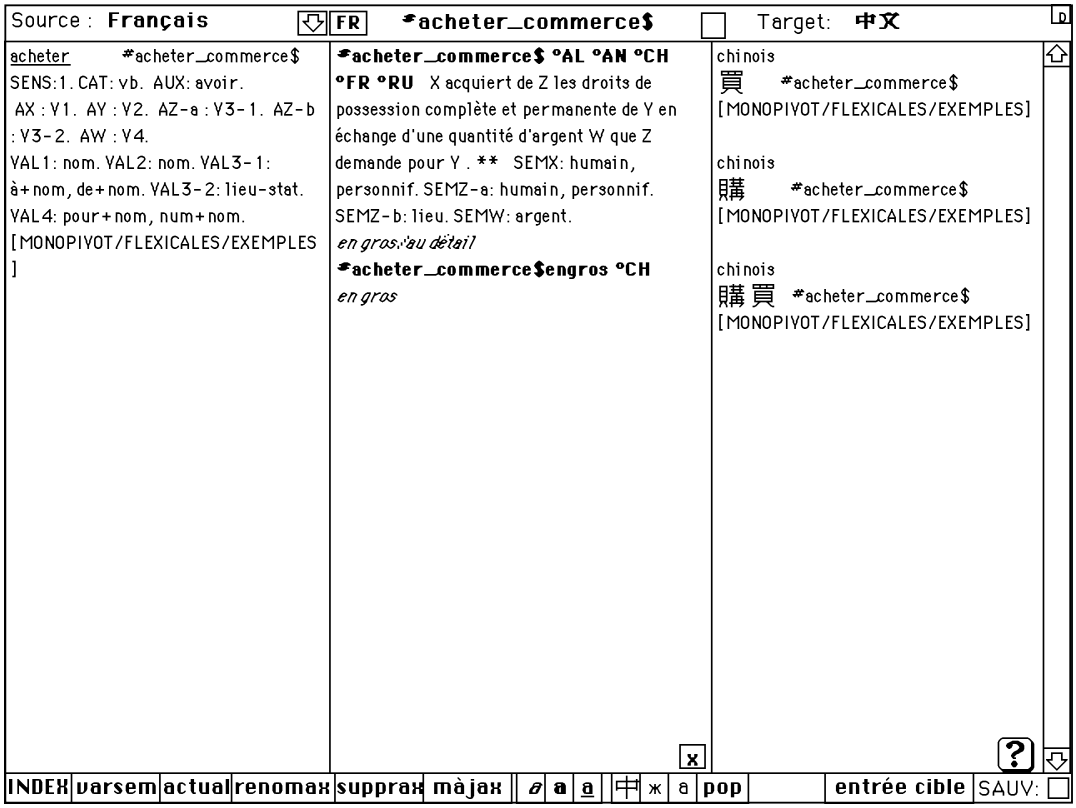


Figure 6.15 : L'acception interlingue "#acheter_commerce" et ses traductions en chinois

On procède de la même manière pour obtenir les correspondants des sous-acceptions.

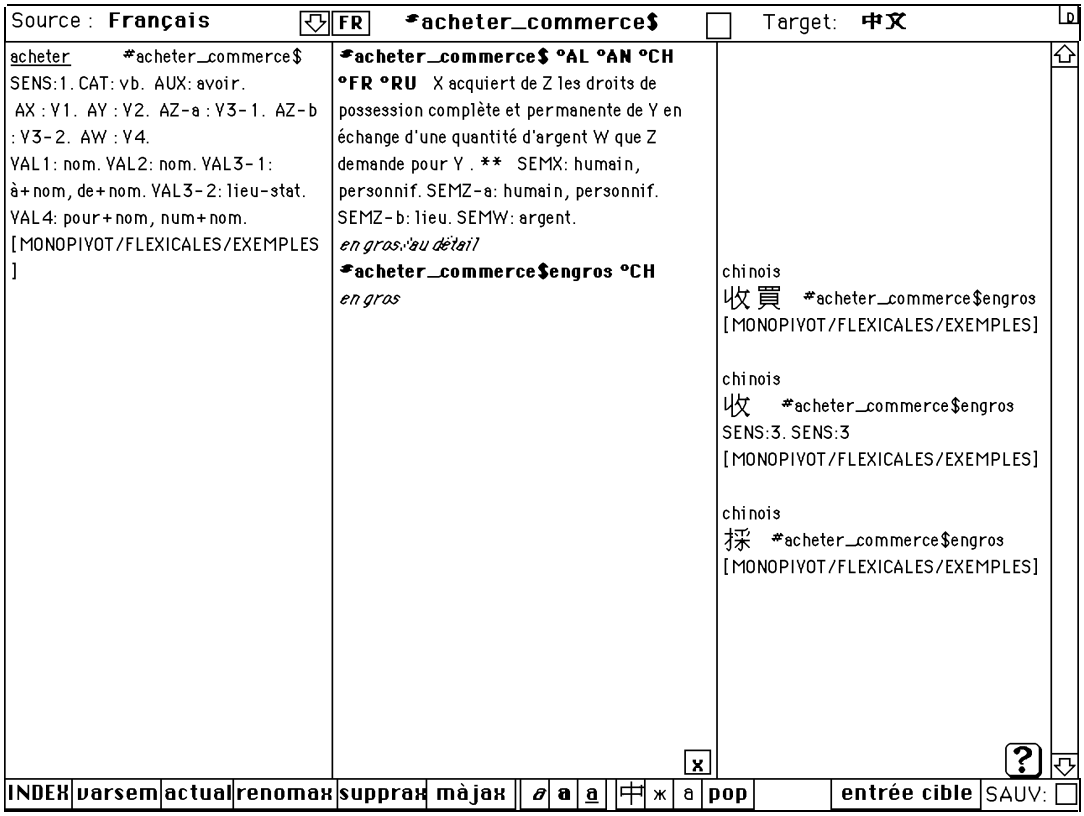


Figure 6.16 : L'acception interlingue "#acheter_commerce$engros" et ses traductions en chinois

Enfin, en cliquant sur `MONOPIVOT` pour l'acception cible considérée, on arrive au dictionnaire monolingue de la langue cible.

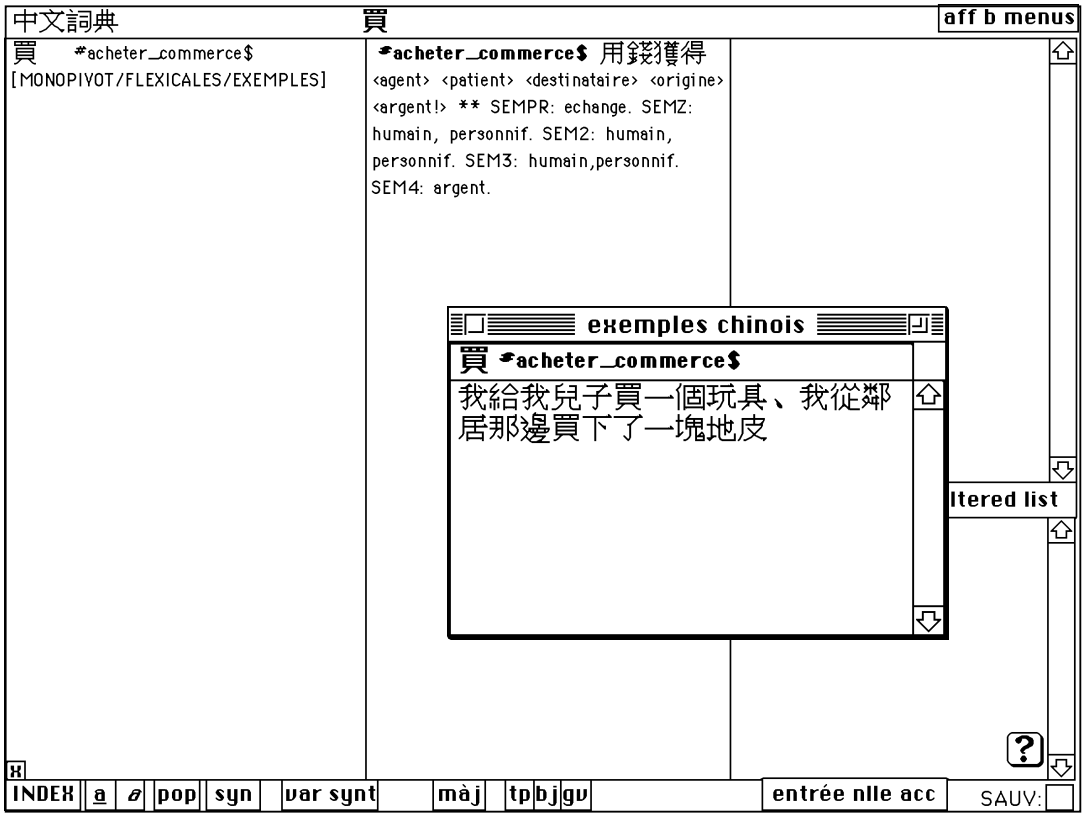


Figure 6.17 : Une entrée chinoise correspondant à l'acception "#acheter_commerce"

3.3. PARAX et les fonctions lexicales

Chaque acception monolingue est associée à un ensemble d'exemples et de fonctions lexicales telles qu'elles apparaissent dans le Dictionnaire Explicatif et Combinatoire d'Igor Mel'čuk. On obtient les exemples en cliquant sur EXEMPLE et les fonctions lexicales en cliquant sur FLEXICALES.

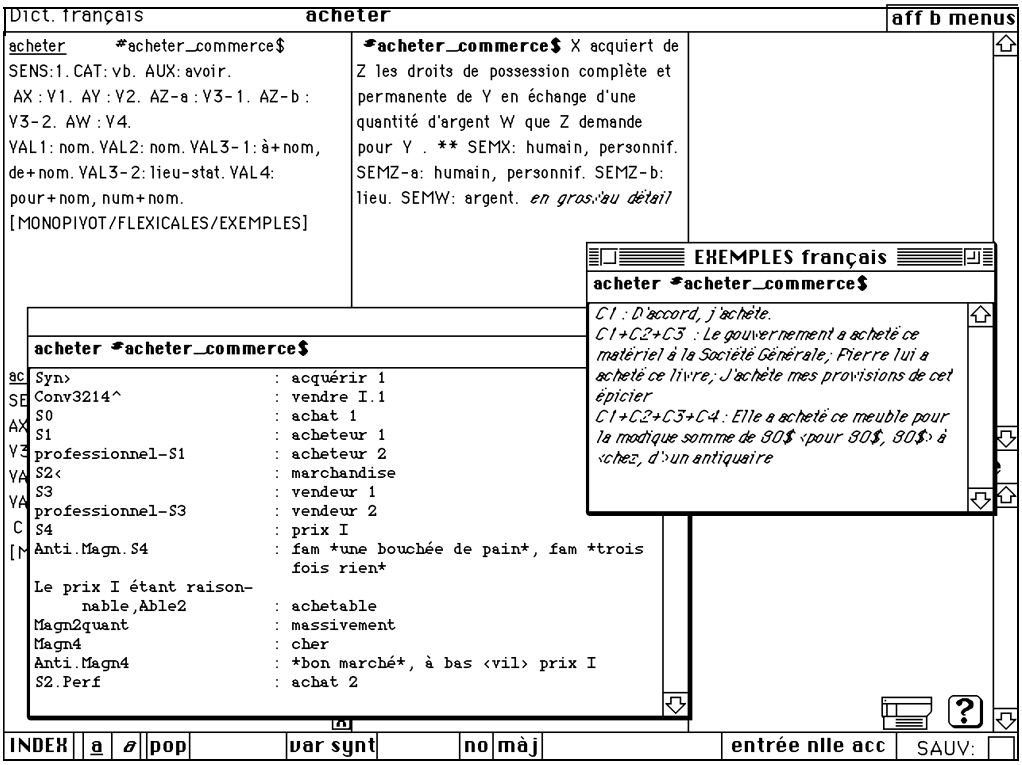


Figure 6.18 : Fonctions lexicales et exemples associés à l'acception monolingue Française "#acheter_commerce"

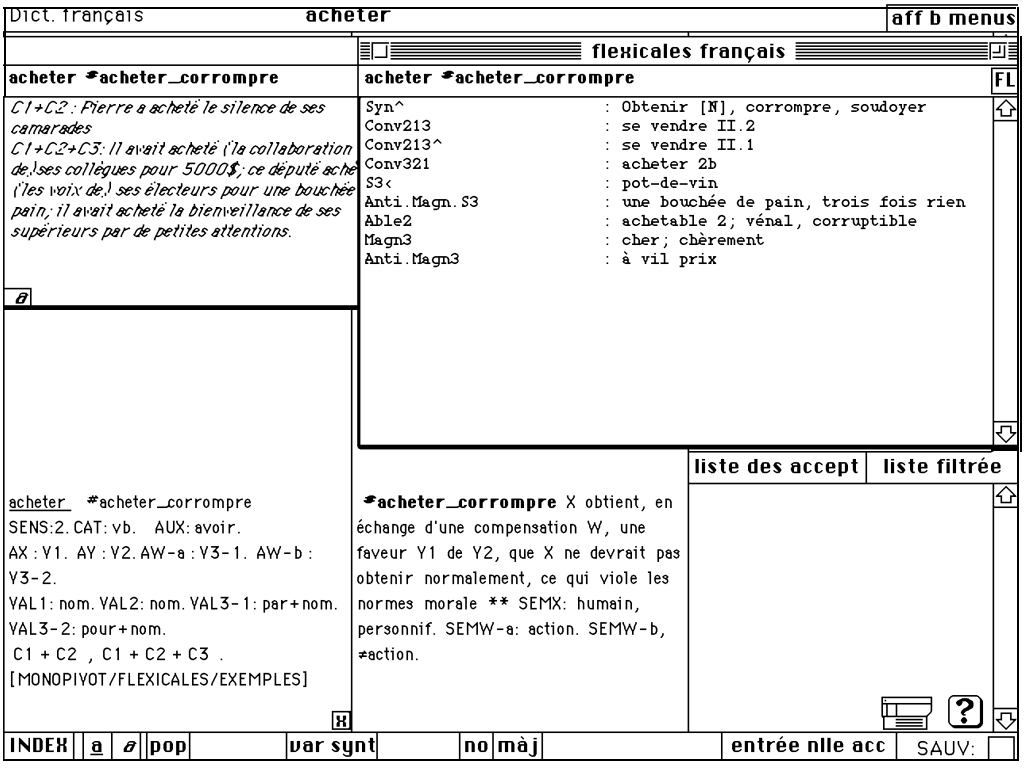


Figure 6.19 : Fonctions lexicales et exemples associés à l'acception monolingue Française "#acheter_corrompre"

3.4. Problèmes et limitations

L'utilisation du logiciel HyperCard ne permettra pas la construction de bases lexicales de grande taille, même si une expérimentation dans un logiciel de base de données commerciale (4D) n'a pas permis de conclure à la supériorité d'un tel système.

La création de nouvelles acceptions n'est pas pilotée par le système, mais reste à l'initiative du lexicographe. Cette méthodologie très ouverte oblige le linguiste à se poser de nombreuses questions lors d'une telle création. En conséquence, les motivations de cette création sont plus souvent dus à un raffinement naturel du linguiste (qui raisonne en termes de concepts) qu'a des besoin de codage d'un problème contrastif.

Avec une plate-forme de développement aussi générale, il est toujours possible de rajouter des fonctionnalités permettant l'indexage d'une entrée, la création d'une acception, son renommage, et la création d'un lien lexical. Par contre, il est difficile d'offrir une interface différente de celle présentée plus haut.

En particulier, il est impossible d'offrir au linguiste une vue graphique des différents réseaux lexicaux définis par les fonctions lexicales de Mel'čuk. Pourtant, une telle visualisation globale est utile pour la vérification des informations lexicales. Notons qu'un système se basant sur SUBLIM doit définir un outil de visualisation de graphe. En effet, une telle visualisation n'est pas très aisée à définir en GRIF.

Enfin, à cause de l'absence d'un mécanisme pratique de manipulation de structures complexes, HyperCard ne permet pas la définition et la vérification efficace de contraintes de cohérence.

HyperCard a donc permis de développer rapidement une maquette de base lexicale interlingue par acceptions. Il a aussi permis d'expérimenter les problèmes de maintenance rencontrés dans cette approche, et d'en déduire les comportements généraux souhaitables dans un système générique de gestion de bases lexicales interlingues par acceptions.

4. Conclusion de l'étude

L'approche par acceptions régit l'architecture lexicale et l'organisation des différentes unités du lexique. Elle n'impose pas de restriction sur les informations linguistiques associées aux unités de dictionnaires. On peut donc la considérer comme un cadre général dans lequel le linguiste est libre d'implémenter sa base lexicale, à condition de satisfaire à la condition suivante :

- **les unités du dictionnaire sont des acceptions**.

À cette condition, le linguiste dispose donc de toute la flexibilité de SUBLIM pour le codage des ses informations linguistiques.

De plus, il est possible d'associer des informations aux acceptions interlingues.

Enfin, l'approche par acceptions n'est donc pas antinomique d'une information linguistique interlingue.

En particulier, on peut imaginer de "glisser" vers l'approche conceptuelle, dans le cas de bases restreintes/dédiées à des langues de spécialité.

Enfin, nous pouvons dégager trois aspect génériques à l'approche interlingue par acceptions :

- quelle que soit la forme de l'information linguistique contenue dans les dictionnaires, les unités des lexiques monolingues et bilingues sont des acceptions ;

- l'approche par acceptions utilise un lien de raffinement. De plus, cette approche est propice à l'utilisation de fonctions lexicales. Aussi, nous proposons une structure de base correspondant à ces fonctions lexicales, et représentons de manière analogue le lien de raffinement ;
- l'utilisation de liens lexico-sémantiques entre unités du lexique permet (et incite à) une détection des schémas illicites dans un grand réseau lexical. Un système générique doit donc fournir des moyens de vérifier la cohérence de ce type d'informations.

L'application de l'approche par acceptions à un gestionnaire de bases lexicales multilingues se concrétise par le système NADIA, qui est une spécialisation du système SUBLIM.

Le système NADIA permet la création et la gestion de bases lexicales multilingues fondées sur les acceptions. Il permet au linguiste de définir les informations linguistiques associées aux unités des dictionnaire monolingues ou interlingue et lui propose les structures de base de SUBLIM et une structure plus particulièrement dédiée à la définition de fonctions lexicales. Il propose de plus un moyen de vérifier la cohérence d'un réseau lexical.

VII. Implémentation

1. L'acception, une structure logique supplémentaire

La première partie de la spécialisation de SUBLIM vers NADIA porte sur son noyau. Afin de pouvoir gérer des acceptions monolingues et interlingues, on introduit deux nouvelles structures de base : *monolingual-acception* et *interlingual-acception*.

1.1. Acceptions monolingues

L'acception monolingue est une structure pouvant accueillir n'importe quelle information linguistique. De plus, elle doit contenir un lien vers une et une seule acception interlingue se trouvant dans le dictionnaire interlingue. Nous la définissons comme une sous-classe de la classe acception, définie comme un agrégat avec un seul trait contenant n'importe quelle information linguistique.

Ainsi, la définition en DÉCOR de la nouvelle structure de base est la suivante :

```
(define! acception :type
    (:is-a :aggregated)
    (linguistic-information T))
(define! link-to-interlingua :type
    (:is-a 'link)
    (target (:type 'interlingual-acception)))
(define! monolingual-acception :type
    (:is-a 'acception )
    (interlingual-acception 'link-to-interlingua))
```

Pour définir une classe linguistique basée sur la structure d'acception, on utilise l'expression suivante :

```
(monolingual-acception keywords*)
```

où les mots-clés restreignent les classes qui peuvent être valeurs de décoration des différents éléments de l'acception monolingue. Le mot-clé possible est :

`:information class` spécifie la classe acceptable pour l'information linguistique associée à l'acception.

Ainsi, la définition :

```
(define-linguistic-class french-acception
   (monolingual-acception
      :information (feature-structure
                     (id         string)
                     (definition string))))
```

se réécrit en :

```
(define! french-acception :type
    (:is-a monolingual-acception)
    (linguistic-information (define! nil :type
                                (:is-a 'feature-structure)
                                (id         (:type 'string))
                                (definition (:type 'string))))))
```

On peut ainsi définir une acception sans ce soucier des liens qu'elle entretient avec les unités du dictionnaire interlingue.

1.2. Acceptions interlingues

L'acception interlingue peut, elle aussi, accueillir n'importe quelle information linguistique ou sémantique. De plus, elle doit contenir un ensemble de liens vers des acceptions monolingues se trouvant dans les différents dictionnaires interlingues.

Ainsi, la définition en DÉCOR de la nouvelle structure de base passe par la définition d'un lien vers les acceptions monolingues. Cette définition est la suivante :

```
(define! link-to-monolingual-acception :type
    (:is-a 'link)
    (target (:type 'monolingual-acception))
    (source (:type 'interlingual-acception)))
(define! interlingual-acception :type
    (:is-a 'acception)
    (monolingual-acceptions (define! nil :type
                               (:is-a :list)
                               (:range 1)
                               (:allowed-types 'link-to-monolingual-acception)))
    (close-acceptions (define! nil :type
                               (:is-a :list)
                               (:allowed-types 'refinement-link)))
    (explanation            string))
```

On remarque dans cette définition qu'une acception interlingue contient une explication sous forme d'une chaîne de caractères (ce qui est nécessaire pour une bonne gestion de la base).

Cette définition utilise la classe `'refinement-link` définie plus bas.

Pour définir une structure basée sur la structure d'acception interlingue, on utilise l'expression suivante :

```
(interlingual-acception keywords*)
```

où les mots-clés restreignent les classes qui peuvent être valeurs de décoration des différents éléments de l'acception interlingue. Les mots-clés possibles sont :

`:information` *`class`*	spécifie la classe acceptable pour l'information linguistique associée à l'acception.
`:refinement-link` *`class*`*	spécifie la (ou les) classe(s) de lien de raffinement (par défaut, on aura la classe `'refinement-link`).
`:link-to-monolingual-acception` *`class*`*	spécifie la (ou les) classe(s) de lien vers les dictionnaires monolingues.

On remarque qu'il est possible de modifier les liens de raffinement afin de leur associer une information (pondération…). Il est aussi possible de changer la classe des liens vers les acceptions monolingues afin d'y ajouter une information quelconque.

Ainsi, la définition :

```
(define-linguistic-class my-acception
   (interlingual-acception
      :information (feature-structure
                      (id         string)
                      (definition string))
      :refinement-link 'my-link))
```

se réécrit en :

```
(define! my-acception :type
    (:is-a 'interlingual-acception)
    (linguistic-information (define! nil :type
                               (:is-a 'feature-structure)
                               (id         (:type 'string))
                               (definition (:type 'string)))))
    (close-acceptions       (define! nil :type
                               (:is-a :list)
                               (:allowed-types '(my-link)))))
```

1.3. Dictionnaires d'acceptions

Le noyau de SUBLIM comprend aussi des éléments dictionnaires qui sont les éléments de base de la définition d'une base lexicale. NADIA introduit donc une nouvelle classe de dictionnaire : le dictionnaire interlingue par acceptions.

La définition d'un dictionnaire interlingue par acceptions se fait grâce à la fonction :

`define-acception-dictionary` *`name Keywords*`*

où *`name`* est un symbole définissant de manière unique le dictionnaire pour l'ensemble de la base.

`:owner` *`string`*	spécifie le propriétaire du dictionnaire.
`:links` *`list`*	spécifie la liste des dictionnaires liés par le dictionnaire interlingue. Cette liste est donnée sous forme de liste de symboles, chacun correspondant à un dictionnaire défini par ailleurs.

Cette définition se traduit par la création d'une instance de la classe *acception-dictionary*.

La classe *acception-dictionary* définie dans Nadia comprend des méthodes spécialisées pour la création et la gestion des acceptions interlingues.

2. Le lien lexical

Comme nous l'avons indiqué dans le chapitre précédent, l'approche par acceptions se prête bien à l'utilisation de liens lexicaux analogues aux fonctions lexicales.

2.1. Lien général

Comme dans SUBLIM, un lien a une source, une cible et une étiquette. Dans le cas d'un lien lexical, la source et la cible sont des acceptions. Pour le lien général, on ne fait aucune hypothèse sur l'information portée par l'étiquette.

Ainsi, ce lien lexical est une nouvelle structure de base définie comme suit :

```
(define! lexical-link :type
    (:is-a 'link)
    (target (:type 'acception))
    (source (:type 'acception))
    (label  T))
```

Pour définir une structure basée sur la structure de lien lexical, on utilise l'expression suivante :

`(lexical-link` *`keywords*`*`)`

où les mots-clés restreignent les classes qui peuvent être valeurs de l'étiquette du lien. Les mots-clés possibles sont :

`:label` *`class`*	spécifie la classe acceptable pour l'étiquette du lien.
`:target class`	spécifie la classe de la cible (`'acception` par défaut).
`:source class`	spécifie la classe de la source (`'acception` par défaut).

Ainsi, la définition :

```
(define-linguistic-class my-link
   (lexical-link
      :label integer)
```

se réécrit en :

```
(define! my-link :type
    (:is-a 'lexical-link)
    (label 'integer))
```

2.2. Lien de raffinement

La gestion de l'interlingue par acceptions passe par le lien de raffinement. Ce lien relie deux acceptions interlingues et contient une étiquette constante contenant son nom.

```
(define! refinement-link :type
    (:is-a  'lexical-link)
    (target 'interlingual-acception)
    (source 'interlingual-acception)
    (label  'refinement-link))
```

Ce lien ne peut pas être redéfini par l'utilisateur.

2.3. Liens “à la Mel’čuk”

Une fonction lexicale, comme par exemple une FL de Mel'čuk, est un lien entre acceptions. Ce lien comprend un nom de fonction comme étiquette. Pour représenter la composition de fonctions lexicales, nous représentons ce nom comme une liste (ordonnée) de noms de fonctions de base.

La sémantique d'une fonction de base peut être modifiée par l'ajout d'un indice ou d'un exposant. Nous nous limiterons aux indices “productifs” qui sont restreints aux numéros d’argument (d’autres indices existent mais sont locaux à un type de fonction et seront intégrés dans le nom). Aussi, un nom de fonction de base est représenté par une structure complexe :

```
(define! base-function-name :type
    (:is-a  'aggregated)
    (name     'flname)
    (index    'possible-indexes)
    (exponent 'possible-exponents))
(define! flname :type
    (:is-a  :enumerated)
    (:allowed-values '(Syn Syn^ Syn< Syn> Conv Anti Anti^ Anti< Anti> Contr
                       Epit Gener Figur S V A Adv Sinstr Sloc Smed Smod Sres
                       Sing Mult Cap Equip Germ Centr Culm Able Qual Magn
                       Plus Minus Ver Bon Pejor Pos Instr Loc Locin Locab
                       Locad Propt Pred Oper Func Labor Involv Incep Cont Fin
                       Caus Liqu Perm Real Fact LabReal Manif Prepar Prox
                       Degrad Nocer Obstr Excess Son Imper Perf Result Sympt)))
(define! possible-indexes :type
    (:is-a  :enumerated)
    (:allowed-values '(1 2 3 4)))

(define! possible-exponents :type
    (:is-a  :enumerated)
    (:allowed-values '(actual usual qual quant temp color dim fulg motor
                       stat trem t° I II III)))
```

Un nom de fonction lexicale est représenté par une liste de fonctions de base :

```
(define! lexical-function-name :type
    (:is-a  :list)
    (:allowed-types '(base-function-name)))
```

Un lien à la Mel’čuk est représenté par un lien avec un nom de fonction lexicale en étiquette :

```
(define! lexical-function :type
    (:is-a :lexical-link)
    (label 'lexical-function-name))
```

3. Vérification de cohérence

L'un des problèmes de la maquette PARAX provient de la difficulté de vérifier la cohérence des informations linguistiques. Cette cohérence peut porter sur les informations linguistiques contenues dans une entrée. Dans ce cas, le vérificateur de cohérence de SUBLIM peut-être utilisé.

Lorsqu'on utilise de nombreux liens entre acceptions, comme c'est le cas avec les fonctions lexicales, il faut aussi détecter des schémas incohérents sur le réseau lexical formé par ces liens.

3.1. Exemples de schémas à détecter

Les schémas de cohérence à détecter peuvent s'appliquer sur le réseau lexical formé par un ensemble de liens particuliers ou sur l'ensemble du réseau lexical de la base.

Prenons l'exemple d'une base lexicale interlingue par acceptions utilisant les fonctions lexico-sémantiques de Mel'čuk. Certaines fonctions lexicales sont reflétées dans la base interlingue (les relations de synonymie…). Dans une telle base lexicale, on veut pouvoir définir les contraintes de cohérence suivantes :

Les relations de synonymie sur les acceptions interlingues doivent être reflétées dans les dictionnaires monolingues (figure 7.1) :

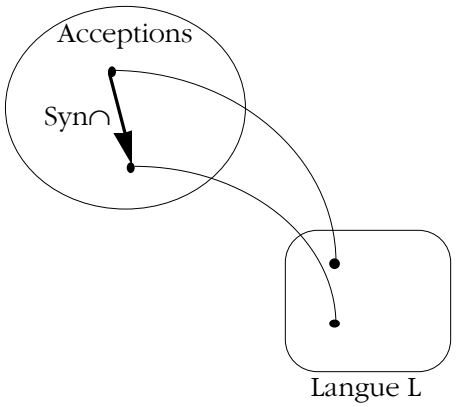


Figure 7.1 : Le lien de synonymie interlingue doit se refléter dans le dictionnaire monolingue

Cette contrainte de cohérence n'est pas une contrainte portant sur une configuration graphique, mais uniquement sur un élément de graphe. Elle peut donc s'exprimer avec le langage général de vérification de contraintes de SUBLIM.

Il s'agit d'une contrainte de cohérence globale, portant sur l'ensemble des liens de synonymie.

Un lien de synonymie du dictionnaire interlingue relie deux acceptions interlingues. Pour exprimer cette contrainte, il nous faut, à partir d'une acception interlingue, pouvoir connaître l'acception monolingue qui lui correspond dans un dictionnaire donné. Pour cela, on utilise la fonction prédéfinie dans NADIA :

```
corresponding-acception(A, D)
```

où `A` est une acception et `D` est un dictionnaire. Cette fonction renvoie une acception `A'` du dictionnaire `D`, correspondant à `A`. S'il n'existe pas d'acception correspondante, cette fonction retourne `nil`.

En voici la définition pour le dictionnaire *french* :

```
(define-coherence-rule interlingual-monolingual-syn
     :applies-on      (acception::lexical-function
                                  {lexical-function-name
                                      [flname : 'Syn]}
                                  [from: @source-acception,
                                   to:   @target-acception])
```

```
:verifies        (let ((source (corresponding-acception source-acception))
                       (target (corresponding-acception target-acception)))
                  (if (and source target)
                    (exist? (french::lexical-function
                                        {lexical-function-name
                                            [flname : 'Syn]}
                                        [from: @source-acception,
                                         to:   @target-acception]))
                      T))
 :error-message  "Un synonyme interlingue doit être reflété dans le
                  dictionnaire français"
 :level          :warning)
```

On ne peut avoir de cycle dans le sous-réseau des relations de synonymie englobante Syn∩ (voir figure 7.2.) :

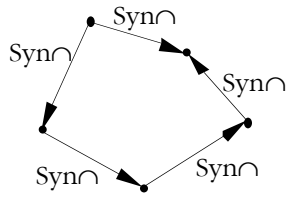


Figure 7.2 : Configuration illicite dans le sous-réseau de synonymie englobante Syn∩

Ainsi, dans chaque dictionnaire, il faut pouvoir spécifier que la relation de synonymie englobante Syn∩ n'admet pas de cycle. Cette relation permet de trouver des problèmes potentiels dans le réseau lexical d'une langue particulière.

Lorsqu'une relation est donnée dans différents dictionnaires, elle forme un réseau qui recouvre un ensemble de dictionnaires (voire la totalité de la base). Dans ce cas, il est intéressant de vérifier la cohérence entre les relations données sur les différents dictionnaires.

Dans ce cas, la contrainte définie plus haut correspond à la détection d'un schéma illicite ayant la forme donnée dans la figure 7.3.

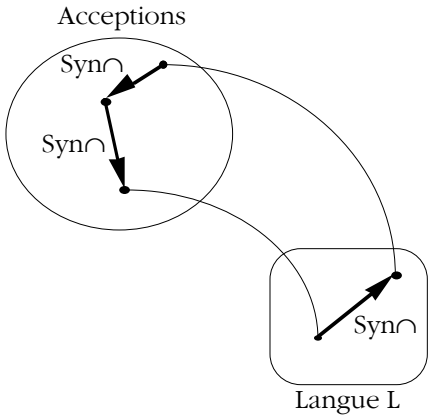


Figure 7.3 : Configuration illicite dans le sous réseau lexical interdictionnaire de synonymie englobante Syn∩

Ces contraintes portent sur l'ensemble d'un réseau lexical. On ne peut donc utiliser de manière efficace le moteur de vérification proposé par SUBLIM. On définit donc un moteur spécialisé dans la vérification de cohérence sur des réseaux lexicaux.

3.2. Déclaration d'une contrainte de cohérence sur le réseau lexical

La plupart des schémas à détecter passe par la détection d'un cycle dans un réseau lexical complexe. Cette détection de cycle ne peut se faire qu'avec des liens orientés.

Par contre, certains liens orientés forment des réseaux où les cycles sont valides. Aussi, il faut donner au linguiste un moyen d'exprimer quels sont les liens pour lesquels la cohérence doit être vérifiée.

Ces déclarations de contraintes comportent 2 parties principales :

- la déclaration du réseau lexical sur lequel portent les contraintes,
- la propriété qui doit être vérifiée par le réseau ainsi déclaré.

La déclaration d'un réseau lexical passe par la déclaration des liens qui le composent. On peut définir un réseau portant sur un dictionnaire (la définition est alors faite au niveau du dictionnaire) ou sur un ensemble de dictionnaires (la définition est alors faite au niveau de la base lexicale).

La déclaration d'un réseau portant sur un dictionnaire est faite en donnant l'ensemble des liens qui définissent ce réseau :

```
(lexical-network links*)
```

ou links est un suite de liens définis sur le dictionnaire.

Ainsi, le réseau formé par l'ensemble des liens de synonymie de Mel'čuk est défini par l'expression :

```
(lexical-network (lexical-function
                    {lexical-function-name
                        [flname : 'Syn]})
                 (lexical-function
                    {lexical-function-name
                        [flname : 'Syn^]})
                 (lexical-function
                    {lexical-function-name
                        [flname : 'Syn<]})
                 (lexical-function
                    {lexical-function-name
                        [flname : 'Syn>]}))
```

La déclaration d'un réseau portant sur un ensemble de dictionnaires est faite en donnant l'ensemble des liens qui définissent ce réseau. Ces liens seront notés en indiquant le dictionnaire sur lequel ils portent. Ainsi, un réseau de synonymie englobante Syn∩ portant sur trois dictionnaires monolingues (français, anglais et allemand) et le dictionnaire d'acceptions est-il défini au niveau de la base lexicale de la manière suivante :

```
(lexical-network (french::lexical-function
                    {lexical-function-name
                        [flname : 'Syn^]})
                 (english::lexical-function
                    {lexical-function-name
                        [flname : 'Syn^]})
                 (german::lexical-function
                    {lexical-function-name
                        [flname : 'Syn^]})
                 (acception::lexical-function
                    {lexical-function-name
                        [flname : 'Syn^]})
                 link-to-interlingua)
```

Rappelons que `link-to-interlingua` est la classe des liens qui relient une acception monolingue à son acception interlingue correspondante.

La propriété à vérifier sur un réseau lexical est l'une des propriétés suivantes :

- `acyclic` : vérifie qu'un réseau est sans cycle,
- `graph` : vérifie que le réseau (un multigraphe) est composé d'un seul graphe,
- `forest` : vérifie que le réseau (un multigraphe) est équivalent à une forêt (chaque graphe du réseau est un arbre),
- `tree` : vérifie que le réseau (un multigraphe) est équivalent à un arbre (le réseau est composé d'un seul graphe équivalent à un arbre).

Nous avons vu l'utilité de la propriété `acyclic`. Les propriétés `graph`, `forest` et `tree` sont utiles pour tester des relations définissant une hiérarchie (on ne doit avoir qu'un graphe sans cycle) ou une arborescence.

Ainsi, une contrainte de cohérence portant sur un réseau lexical s'exprime grâce à l'expression :

```
define-network-coherence-rule name
                              :applies-on    lexical-network-spec
                              :verifies      property
                              :error-message string
                              :level         level
```

La contrainte de cohérence vérifiant l'absence de cycle dans le réseau lexical de synonymie englobante d'un dictionnaire particulier est définie de la manière suivante (au niveau du dictionnaire) :

```
(define-network-coherence-rule acyclic-more-general-synonymy
    :applies-on    (lexical-network (lexical-function
                                       {lexical-function-name [flname : 'Syn^]}))
    :verifies      'acyclic
    :error-message "Détection d'un cycle dans le réseau de synonymie englobante"
    :level         :warning)
```

La contrainte vérifiant l'absence de cycle dans le réseau lexical de synonymie englobante sur plusieurs dictionnaires est définie de la manière suivante (au niveau de la base lexicale) :

```
(define-network-coherence-rule global-acyclic-more-general-synonymy
    :applies-on    (lexical-network
                      (french::lexical-function
                         {lexical-function-name
                             [flname : 'Syn^]})
                      (english::lexical-function
                         {lexical-function-name
                             [flname : 'Syn^]})
                      (german::lexical-function
                         {lexical-function-name
                             [flname : 'Syn^]})
                      (acception::lexical-function
                         {lexical-function-name
                             [flname : 'Syn^]})
                      link-to-interlingua)
    :verifies      'acyclic
    :error-message "Détection d'un cycle dans le réseau de synonymie global"
    :level         :warning)
```

Le système NADIA ainsi défini est en cours de réalisation. La première étape de son utilisation passe par la récupération de la maquette Parax dans ce système.

De plus, NADIA sera appliquée au développement d'une version informatique du DEC dans le cadre d'une action de recherche partagée entre le GETA et l'équipe d'Igor Mel'čuk à l'Université de Montréal.

Conclusion

Le premier système présenté dans ce document, SUBLIM, se place dans la continuité des efforts de généralisation d'outils de gestion de bases lexicales (MULTILEX, Le Lexicaliste…). Il part d'une volonté de disposer d'un outil générique de gestion de bases de données lexicales multilingues. Ce projet apporte des nouveautés par rapport aux différents systèmes de gestion de bases lexicales multilingues. Il permet de spécifier l'architecture lexicale d'une base particulière en utilisant des dictionnaires monolingues, bilingues ou interlingues dont la gestion globale (accès aux unités, structure squelette…) est prise en charge par le système. Les unités des dictionnaires, ainsi que les informations qu'elles portent, ne sont pas contraintes. Cela permet d'utiliser SUBLIM pour implémenter une des bases lexicales "fondées sur la connaissance" inspirées de projets tels que EDR (Japon) ou KBMT-89 (USA). On peut aussi implémenter des bases lexicales fondées sur une approche par transfert comme celles du projet MULTILEX ou comme les dictionnaires de METAL.

Ainsi, le projet SUBLIM dépasse les faiblesses du projet MULTILEX (dictionnaires bilingues par transfert, obligation de coder les structures linguistiques sous forme de structures de traits typées…), qui est le projet le plus poussé parmi ceux qui se sont attaqués à la définition de gestionnaires de bases lexicales indépendants des applications.

Le linguiste peut définir l'architecture linguistique des différents dictionnaires de sa base lexicale. Pour cela, il choisit les structures logiques servant de base à ses structures linguistiques parmi une importante collection (automates, graphes, arbres, structures de traits typés, ensembles, listes…). En combinant ces structures, on peut définir des structures linguistiques complexes d'une manière naturelle. Cette approche universelle permet la création de bases lexicales pour des usages différents, automatiques aussi bien qu'humains.

Le second projet présenté dans ce document, NADIA, ajoute au fonctionnalités générales de SUBLIM, des fonctionnalités particulières permettant la gestion de bases lexicales interlingues fondées sur les acceptions. L'architecture lexicale interlingue que nous privilégions pour nos applications est, comme celle du projet ULTRA, fondée sur des connaissances linguistiques plutôt que sur des connaissances extralinguistiques. Le langage pivot n'est plus formé de "concepts" (indépendants des langues), mais "d'acceptions interlingues" fonctions des langues en présence.

Cette étude présente une première étape dans la mise au point effective d'un système de gestion de bases lexicales interlingues. La définition d'une architecture à trois niveaux séparant clairement les problème de stockage, de représentation et de présentation des données garantit d'une part l'indépendance des outils vis-à-vis d'un système de stockage de données (SGBD relationnel, à objet, ou autres), et l'indépendance des données vis-à-vis de leur présentation.

L'utilisation d'une approche permettant de combiner des structures logiques de base pour définir, d'une manière naturelle, des structures linguistiques plus complexes donne au linguiste une liberté de choix et une expressivité qu'on ne trouve, à notre connaissance, dans aucun autre système de gestion de bases lexicales. Cette approche permet, avec une même plate-forme logicielle, de gérer des bases lexicales développées pour des besoins différents. Elle offre ainsi un premier pas vers la fusion ou la récupération de bases lexicales.

L'étude d'une approche interlingue fondée sur les acceptions a permis la spécialisation d'un système général. Néanmoins, cette spécialisation est générique et laisse au linguiste toute liberté de choix quant aux structures linguistiques portées par les "unités dictionnairiques" (acceptions interlingues, acceptions monolingues…).

L'approche par acceptions est bien adaptée à l'utilisation de liens lexicaux analogues aux fonctions lexicales définies par Igor Mel'čuk, puisque les unités des dictionnaires monolingues sont identiques aux unités du DEC. Cette approche permet l'étude du codage de certaines fonctions lexicales au niveau de la base interlingue.

La mise au point effective d'un tel système de gestion de bases lexicales multilingue ne peut être que longue et difficile. En effet, les développements et les problèmes de génie logiciel soulevés par l'implémentation de certains des outils définis ici sont des problèmes non triviaux. De plus, certains des outils définis dans ce document ont été simplifiés et peuvent faire l'objet d'étude séparées.

Dans le cadre d'une action de recherche partagée conduite entre le GETA et l'équipe d'Igor Mel'čuk à l'Université de Montréal, nous allons appliquer le système NADIA à la création d'une version électronique du DEC. Nous pourrons ainsi expérimenter différentes visualisations de la base lexicale qui sera créée. Nous pourrons de plus étudier en détail l'application des fonctions lexicales à un lexique interlingue. Enfin, cette expérimentation consistera en un premier test de la validité des choix pris pour les systèmes SUBLIM et NADIA.

Dans ce cadre, seule une certaine partie des outils de SUBLIM pourra être implémentées. Certains, comme le mécanisme d'import/export, pourront faire l'objet d'une étude approfondie, en vue d'une plus grande généralisation. De plus, l'import d'un dictionnaire quelconque dans une base SUBLIM passe par un prétraitement ("nettoyage" des données, standardisation en un fichier SGML…). Ce prétraitement constitue une opération difficile et nécessite certains outils particuliers. Dans cette optique, il est intéressant d'étudier la définition d'une plate-forme générique de manipulation de données linguistiques et lexicales qui permette de standardiser des documents (dictionnaires, corpus, bandes de photocomposition…) en des documents SGML, voire en des instances de documents TEI lorsque cela est possible.

Bibliographie

[Abeillé 1989] **Abeillé A. (1989)** *L'unification dans une grammaire d'arbre adjoints: quelques exemples en syntaxe française.* T.A. Information, **30**/1-2: pp. 69-112.

[Abeillé 1993] **Abeillé A. (1993)** *Les nouvelles syntaxes - Grammaires d'unification et analyse du français,* Armand Colin, Paris, 327 p.

[Adriaens & al. 1990] **Adriaens G. & Lemmens M. (1990)** *The Self Extending Lexicon : Off-line and On-line Defaulting of Lexical Information in the METAL Machine Translation System.* Proc. Coling-90, Helsinki, 20-25 August 1990, H. Karlgren ed. vol. 3/3: pp. 305-307.

[Aït-Kaci 1986] **Aït-Kaci H. (1986)** *An Algebraïc Approach to the Effective Resolution of Type Equations.* Theoretical Computer Science, **45**: pp. 293-351.

[Aït-Kaci & al. 1988] **Aït-Kaci H. & Lincoln P. (1988)** *LIFE : a Natural Language for Natural Language.* T.A. Information, **30**/1-2: pp. 37-67.

[Aït-Kaci & al. 1992] **Aït-Kaci H., Meyer R. & Roy P. V. (1992)** *Wild LIFE - A User Manual,* Available with the WILD-LIFE software, 81 p.

[Aït-Kaci & al. 1986] **Aït-Kaci H. & Nasr R. (1986)** *LOGIN : a Logic Programming Language with Built-in Inheritance.* Journal of Logic Programming, **3**: pp. 185-215.

[André & al. 1989a] **André J., Furuta R. & Quint V. (1989a)** *By way of an introduction. Structured Documents: What and Why?* In "Structured Documents", J. André, R. Furuta & V. Quint ed., Cambridge University Press: pp. 1-6.

[André & al. 1989b] **André J., Furuta R. & Quint V. (1989b)** *Structured Documents* P. Hammersley ed., The Cambridge Series on Electronic Publishing, Cambridge University Press, Cambridge, 220 p.

[Apple Computer Inc. 1992a] **Apple Computer Inc. (1992a)** *Macintosh Human Interface Guidelines,* Addison-Wesley Publishing Company, Inc., 384 p.

[Apple Computer Inc. 1992b] **Apple Computer Inc. (1992b)** *Text Services Manager*. In "Inside Macintosh: Text", Apple Computer Inc.: pp. 1-107 (section 7).

[Apple Computer Inc. 1989] **Apple Computer Inc. (1989)** *Hypercard Stack Design Guidelines,* Addison-Wesley Publishing Company, Inc., 230 p.

[Barnett & al. 1990] **Barnett J., Knight K., Mani I. & Rich E. (1990)** *Knowledge and Natural Language Processing*. Communications of ACM, **33**/8: pp. 50-71.

[Blanchon 1990] **Blanchon H. (1990)** *Ambiguity resolution and paraphrase selection*. Proc. DBMT-90, Post-COLING seminar on Dialogue-Based MT, Le Sappey, France, 26-28 August 1990, E. Blanc & C. Boitet ed. vol. 1/1: pp. 38-41 & 238-241.

[Blanchon 1991] **Blanchon H. (1991)** *Problèmes de désambiguïsation interactive en TAO personnelle*. Proc. L'environnement traductionnel : La station de travail du traducteur de l'an 2001, Mons (Belgique), 25-27 avril 1991, Actualités scientifiques, A. Clas: pp. 31-48.

[Blanchon 1992] **Blanchon H. (1992)** *A Solution to the Problem of Interactive Disambiguation*. Proc. Coling-92, Nantes, France, 23-28 juillet 1992, C. Boitet ed. vol. 4/4: pp. 1233-1238.

[Blanchon 1994] **Blanchon H. (1994)** *LIDIA-1 : une première maquette vers la TA interactive "pour tous"*. Thèse nouveau doctorat, Université Joseph Fourier (Grenoble 1), 319 p.

[Boitet 1988a] **Boitet C. (1988a)** *Hybrid Pivots using m-structures for multilingual Transfer-based systems*. Japanese Institute of Electronic Information and Communication Engineering, NLC, **88**/3: pp. 17-22.

[Boitet 1988b] **Boitet C. (1988b)** *Representation and computation of units of translation for Machine Interpretation of spoken texts*. Technical report, GETA - ATR, 1988, 20 p.

[Boitet 1990a] **Boitet C. (1990a)** *Multilingual Machine Traslation does not have to be saved by Interlingua*. Proc. MMT-90, Tokyo, 5-6 November 1990, 2 p.

[Boitet 1990b] **Boitet C. (1990b)** *Towards Personnal MT : general design, dialogue structure, potential role of speech*. Proc. Coling-90, Helsinki, 20-25 August 1990, H. karlgren ed., Hans karlgren, vol. 3/3: pp. 30-35.

[Boitet 1993a] **Boitet C. (1993a)** *Crucial open problems in Machine Translation & Interpretation*. Proc. BKK'93, Bangkok, Thailand, 17-20 March 1993 vol. 1/1.

[Boitet 1993b] **Boitet C. (1993b)** *Human-Oriented Design and Human-Machine-Human Interactions in Machine Interpretation*. Technical Report, ATR Interpreting Telecommunications Research Laboratories, 30 August 1993, 13 p.

[Boitet 1993c] **Boitet C. (1993c)** *Integration of Heterogeneous Components for Speech Translation: the "Whiteboard" Architecture and an Architectural Prototype*. Technical Report, ATR Interpreting Telecommunications Research Laboratories, 30 August 1993, 20 p.

[Boitet 1993d] **Boitet C. (1993d)** *La TAO comme technologie scientifique : le cas de la traduction automatique fondée sur le dialogue*. In "La traductique", P. Bouillon & A. Clas ed., Les presses de l'Université de Montréal, AUPELF/UREF: pp. 109-148.

[Boitet 1993e] **Boitet C. (1993e)** *Multimodal Interactive Disambiguation: first report on the MIDDIM project*. Technical Report, ATR Interpreting Telecommunications Research Laboratories, 30 August 1993, 16 p.

[Boitet 1993f] **Boitet C. (1993f)** *TA et TAO à Grenoble… 32 ans déjà !* T.A.L. (revue semestrielle de l'ATALA), **33**/1—2, Spécial Trentenaire: pp. 45-84.

[Boitet & al. 1990] **Boitet C. & Blanchon H. (1990)** *TAO personnelle et promotion des langues nationales : le projet LIDIA du GETA*. Proc. Les industries de la langue : perspectives des années 1990, Montréal, Canada, 22-24 novembre 1990 vol. 1/2: pp. 415-434.

[Boitet & al. 1993] **Boitet C. & Blanchon H. (1993)** *Dialogue-based MT for monolingual authors and the LIDIA project*. Proc. NLPRS'93, Fukuoka, Japon, 6-7 décembre 1993: pp. 208-222.

[Boitet & al. 1982a] **Boitet C., Guillaume P. & Quezel-Ambrunaz M. (1982a)** *ARIANE-78: an integrated environment for automatic translation and human revision*. Proc. COLING-82, Prague, July 1982: pp. 19-27.

[Boitet & al. 1982b] **Boitet C., Hue & Collomb Réd. (1982b)** *"DSE-2" — Spécification du système Ariane-X*. Projet ESOPE Contrat ADI/CAP-Sogeti/Champollion, GETA-Champollion - Cap Sogeti France, 24 juin 1982.

[Boitet & al. 1982c] **Boitet C. & Nedobejkine N. (1982c)** *Base lexicale : organisation générale et indexage*. rapport final, projet ESOPE ADI, partie D, GETA, Grenoble, 1982, 30 p.

[Boitet & al. 1986a] **Boitet C. & Nedobejkine N. (1986a)** *Toward Integrated Dictionary for M(A)T : Motivations and Linguistic Organisation*. Proc. COLING 86, Bonn, 25-29 aout 1986 vol. 1/1: pp. 423-428.

[Boitet & al. 1986b] **Boitet C. & Nedobejkine N. (1986b)** *Vers une base lexicale intégrée pour la T(a)O : motivations et organisation linguistique*. Proc. Journées francophones de l'informatique, bases de données et bases de connaissances, Grenoble, janvier 1986 vol. 1/1: pp. 151-169.

[Boitet & al. 1994] **Boitet C. & Seligman M. (1994)** *The "Whiteboard" Architecture: A Way to Integrate heterogeneous components of NLP Systems*. Proc. COLING-94, Kyoto, Japan, 5-9 August 1994, M. Nagao ed. vol. 1/2: pp. 426-430.

[Boitet & al. 1988] **Boitet C. & Zaharin Y. (1988)** *Representation trees and string-tree correspondences*. Proc. Coling-88, Budapest, 22–27 August 1988, D. Várgha ed.: pp. 59-64.

[Booch 1992] **Booch G. (1992)** *Conception orientée objets et applications*. Addison-Wesley ed., Addison-Wesley, 588 p.

[Brachman 1993] **Brachman R. J. (1993)** *Viewing Data Through a Knowledge Representation Lens*. Proc. KB&KS'93, Tokyo, Japan, December 1993, JIPDEC ed., JPDEC, vol. 1/1.

[Briscoe & al. 1993a] **Briscoe T. & Caroll J. (1993a)** *Generalized Probabilistic LR parsing Natural Language (Corpora) with Unification-Based Grammars*. C.L., **19**/1: pp. 25-59.

[Briscoe & al. 1993b] **Briscoe T., Paiva V. d. & Copestake A., ed. (1993b)** *Inheritance, Defaults, and the Lexicon*. Studies in Natural Language Processing, B. Boguraev ed., Cambridge University Press, Cambridge, 298 p.

[Broguraev & al. 1989] **Broguraev B., Briscoe E., Calzolari N., Carter A., Meijs W., Picchi E. & al. (1989)** *Acquisition of Lexical Knowledge for Natural Language Processing Systems*. technical annex for Esprit BRA,n° 3030, Acquilex, avril 1989, 30 p.

[Brown & al. 1989] **Brown R., Gates D. M., Goodman K., Kaufmann T., Kee M., Levin L. & al. (1989)** *KBMT-89*. project report, Center for Machine Translation, Carnegie Mellon University, April 1989.

[Brown & al. 1990] **Brown R. D. & Nirenburg S. (1990)** *Human-Computer Interaction for Semantic Disambiguisation*. Proc. COLING 90, Helsinki vol. 3/3: pp. 42-47.

[Calder & al. 1992] **Calder P. & Linton M. (1992)** *The Object-Oriented Implementation of a Document Editor*. Proc. OOPSLA'92, Vancouver, Canada, 1992, ACM ed., ACM, vol. 1/1: pp. 164-165.

[Calzolari 1988] **Calzolari N. (1988)** *The Dictionary and the Thesaurus Can Be Combined*. In "Relationnal Models of the Lexicon", M. W. EVENS ed., Cambridge University Press, Cambridge (Mass.): pp. 75-96.

[Calzolari 1989] **Calzolari N. (1989)** *Lexical Databases and Textual Corpora : Perspectives of Integration for a Lexical Knowledge Base*. Technical Report, Universita di Pisa, Dipartimento di linguistica, 1989, 6 p.

[Calzolari & al. 1990] **Calzolari N. & Bindi R. (1990)** *Acquisition of Lexical Information from a Large Textual Italian Corpus*. Proc. COLING 90, Helsinki, H. Karlgren ed. vol. 3/3: pp. 54-59.

[Calzolari & al. 1988] **Calzolari N. & Picchi E. (1988)** *Acquisition of Semantic Information from an On-line Dictionary*. Proc. COLING 88, Budapest, 22-27 August 1988, D. Várgha ed.: pp. 87-92.

[Carpenter 1992] **Carpenter B. (1992)** *The logic of Typed Feature Structures,* Cambridge University Press, 270 p.

[Chauché 1974] **Chauché J. (1974)** *Transducteurs et arborescences*. Thèse d'Etat, USMG - Grenoble I, 440 p.

[Date 1989] **Date C. (1989)** *Introduction au standard SQL,* Interéditions, 235 p.

[Defrise & al. 1990] **Defrise C. & Nirenburg S. (1990)** *Meaning Representation and Text Planning*. Proc. COLING 90, Helsinki, 20-25 August 1990, H. Karlgren ed. vol. 3/3: pp. 219-224.

[Delannoy 1990] **Delannoy J.-F. (1990)** *A Message Processing System with Object-Centered Semantics*. Proc. COLING-90, Helsinki, 20-25 August 1990, H. Karlgren ed. vol. 3/3: pp. 333-335.

[Delannoy 1991] **Delannoy J.-F. (1991)** *Un système fondé sur les objets pour le suivi de situations a partir de textes en languages naturel*. Thèse nouveau doctorat, Université d'Aix-Marseille III, 173 p.

[Delobel & al. 1982] **Delobel C. & Adiba M. (1982)** *Bases de données et systèmes relationnels,* Dunod, Paris, 450 p.

[DGT 1987] **DGT (1987)** *Version finale de la structure logique de la base de données lexicale*. rapport contrat DGT,n° 15, GETA, février 1987, 40 p.

[Domenig & al. 1992] **Domenig M. & Hacken P. t. (1992)** *Word Manager: A System for Morphological Dictionaries,* Georg Olms Verlag, Hildesheim, 211 p.

[Ducournau & al. 1989] **Ducournau R. & Habib M. (1989)** *La multiplicité de l'héritage dans les langages à objets*. Techniques & Science Informatiques, **8**/1: pp. 41-62.

[Dutoit 1992] **Dutoit D. (1992)** *A Set-Theoretic Approach to Lexical Semantics*. Proc. COLING-92, Nantes, July 23-28, 1992, C. Boitet ed. vol. 3/4: pp. 982-987.

[EDR 1993] **EDR (1993)** *EDR Electronic Dictionary Technical Guide*. Project report,n° TR-042, Japan Electronic Dictionary Research Institute Ltd., August 16, 1993, 144 p.

[Edvins 1993] **Edvins M. (1993)** *Objects Without Classes*. Frameworks, **7**/6: pp. 34-39.

[Emele & al. 1990a] **Emele M., Heid U., Momma S. & Zajac R. (1990a)** *Organising Linguistic Knowledge for Multilingual Generation*. Proc. COLING 90, Helsinki, 20-25 August 1994, H. Karlgren ed. vol. 3/3: pp. 102-107.

[Emele & al. 1990b] **Emele M. & Zajac R. (1990b)** *Typed Unification Grammars*. Proc. COLING 90, Helsinki, 20-25 August 1990, H. Karlgren ed. vol. 3/3: pp. 293-298.

[Farwell & al. 1992] **Farwell D., Guthrie L. & Wilks Y. (1992)** *The Automatic Creation of Lexical Entries for a Multilingual MT system*. Proc. COLING-92, Nantes, 23-28 July 1992, C. Boitet ed. vol. 2/4: pp. 532-538.

[Farwell & al. 1993] **Farwell D., Guthrie L. & Wilks Y. (1993)** *Automatically Creating Lexical Entries for ULTRA, a Multilingual MT System*. M.T., **8**/3: pp. 127-145.

[Fedder & al. 1991] **Fedder L., McNaught J. & Smith S. (1991)** *Typed Feature Logic and its role in MULTILEX*. Rapport Multilex, Centre for Computational Linguistics, UMIST, novembre 1991, 30 p.

[Furuta 1989] **Furuta R. (1989)** *Concepts and Models for Structured Documents*. In "Structured Documents", J. André, R. Furuta & V. Quint ed., Cambridge University Press: pp. 7-39.

[Gaschler & al. 1994a] **Gaschler J. & Lafourcade M. (1994a)** *A Case of Building and Manipulating a Dictionary with Very Simple Tools: the FEM Dictionary*. Proc. ICLA, Penang (Malaysia), 26-28 July 1994 vol. 1/1: pp. 34-37.

[Gaschler & al. 1994b] **Gaschler J. & Lafourcade M. (1994b)** *Manipulating human-oriented dictionaries with very simple tools*. Proc. COLING-94, Kyoto, Japan, August 5-9 1994, M. Nagao ed. vol. 1/2: pp. 283-286.

[Gates & al. 1989] **Gates D., Haberlach D., Kaufmann T., Kee M., McCardell R., Mitamura T. & al. (1989)** *Lexicons*. M.T., **4**/1: pp. 67-112.

[Gazdar & al. 1989] **Gazdar G. & Mellish C. (1989)** *Natural Language Processing in Lisp - An introduction to Computational Linguistics,* Addison-Wesley Publishing Company, 524 p.

[Genelex 1993] **Genelex (1993)** *Projet Eureka Genelex, modèle sémantique*. Rapport Technique, Projet Eureka Genelex, 4 mars 1994, 185 p.

[Genthial 1991a] **Genthial D. (1991a)** *Contribution à la construction d'un système robuste d'analyse du français*. Thèse nouveau doctorat, Université Joseph Fourier, 236 p.

[Genthial 1991b] **Genthial D. (1991b)** *Représentation des données lexicales : vers des traitements tolérants*. Proc. Deuxièmes journées nationales du GRECO-PRC Communication Homme-Machine, Toulouse, EC2 ed.: pp. 69-76.

[Genthial & al. 1990] **Genthial D., Courtin J. & Kowarski I. (1990)** *Contribution of a Category Hierarchy to the Robusteness of Syntactic Parsing*. Proc. COLING-90, Helsinki, 20-25 août 1990, H. Karlgren ed., Hans Karlgren, vol. 2/3: pp. 139-144.

[Goodman & al. 1991] **Goodman K. & Nirenburg S., ed. (1991)** *The KBMT project: a case study in Knowledge-Based Machine Translation.,* Morgan Kaufmann Publishers, San Mateo, California, 330 p.

[Gross 1987] **Gross M. (1987)** *The Use of Finite Automata in the Lexical Representation of Natural Language*. Proc. Electronic Dictionaries and Automata in Computational Linguistics- LITP Spring School on Theoretical Computer Science, St Pierre d'Oleron, M. Gross ed., Springer Verlag, Berlin,: pp. 34-50.

[Gross & al. 1985] **Gross M. & Tremblay D. (1985)** *Etude du contenu d'une banque terminologique.* Rapport technique, LADL Paris, mai 1985, 180 p.

[Hariè 1990] **Hariè S. (1990)** *Analyse automatique d'un dictionnaire en vue de la constitution d'une base de données lexicales*. Mémoire de DEA en Informatique et Automatique mention XIAO, Université d'Aix-Marseille III, GRTC,n° 371, septembre 1990, 68 p.

[Herwijnen 1990] **Herwijnen E. V. (1990)** *Practical SGML,* Kluwer Academic Publishers, Dordrecht(Nl.), 307 p.

[Hutchins 1986] **Hutchins W. J., ed. (1986)** *Machine Translation - Past, Present, Future.* Computers and their Applications, E. Horwood ed., Ellis Hordwood Limited, New York/Chichester/Brisbane/Toronto, 382 p.

[Hutchins & al. 1992] **Hutchins W. J. & Somers H. L. (1992)** *An introduction to Machine Translation,* Academic Press, Harcourt Brace Jovanovich, 362 p.

[Karttunnen 1984] **Karttunnen L. (1984)** *Features and Values*. Proc. COLING-84, Stanford University, California, 2-6 July 1984, ACL ed., Association for Computational Linguistics, vol. 1/1: pp. 28-33.

[Karttunnen 1991] **Karttunnen L. (1991)** *Finite-state Constraints*. Proc. CICL-91, USM, Penang, Malaysia vol. 1/1: pp. 1-18.

[Karttunnen 1993] **Karttunnen L. (1993)** *Finite-State Lexicon Compiler*. Research Report,n° ISTL-NLTT-1993-04-02, Xerox PARC, Avril 1993, 18 p.

[Karttunnen & al. 1992] **Karttunnen L. & Beesley K. R. (1992)** *Two-Level Rule Compiler*. Research Report,n° ISTL-92-2, Xerox PARC, October 1992, 15 p.

[Kay 1973] **Kay M. (1973)** *The MIND system*. In "Courant Computer Science Symposium 8: Natural Language Processing", R. Rustin ed., Algorithmics Press, New York: pp. 155-188.

[Kay 1980] **Kay M. (1980)** *The Proper Place of Men and Machines in Language Translation*. Research Report,n° CSL-80-11, Xerox, Palo Alto Research Center, octobre 1980, 20 p.

[Kay 1982] **Kay M. (1982)** *Machine Translation*. American Journal of Computational Linguistics, **8**/2: pp. 74-78.

[Keene 1989] **Keene S. E. (1989)** *Object-Oriented Programming in Common Lisp,* Addison-Wesley, 266 p.

[Kiczales & al. 1991] **Kiczales G., Rivières J. d. & Bobrow D. G. (1991)** *The Art of the Metaobject Protocol,* MIT Press, 335 p.

[Lafourcade 1992] **Lafourcade M. (1992)** *Le problème de l'accès au lexique dans les outils pour rédacteurs. ODILE, une approche.* Proc. Séminaire Lexique, Toulouse, Pôle langage naturel et parole du GDR-PRC CHM, vol. 1/1: pp. 81-89.

[Lafourcade 1993] **Lafourcade M. (1993)** *Geta-Browser.* GETA-IMAG, Grenoble, Common Lisp Object System (MCL - CLOS), Apple Macintosh, version 2.2.

[Lafourcade 1994a] **Lafourcade M. (1994a)** *Applying Pivot MT Techniques to Multi-dialectal Programming Language Editors.* rapport interne, GETA-IMAG, janvier 1994.

[Lafourcade 1994b] **Lafourcade M. (1994b)** *Génie logiciel pour le génie linguiciel.* Thèse nouveau doctorat, Université Joseph Fourier (Grenoble 1), 300 p.

[Lafourcade 1994c] **Lafourcade M. (1994c)** *ODILE: un outil personnel d'aide à la traduction.* Turjuman, **3**/1: pp. 13-21.

[Lafourcade 1994d] **Lafourcade M. (1994d)** *Re-Engineering with added Genericity of Specialized Languages for Linguistic Programming - A case study with the ATEF & LT SLLPs.* Proc. IACL'94, Penang, Malaysia, 26-28 July 1994: pp. 51-57.

[Lafourcade & al. 1992] **Lafourcade M. & Sérasset G. (1992)** *Geta-Strings.* Logiciel GETA, Grenoble, Common Lisp Object System (MCL - CLOS), Macintosh, version 1.0.

[Lafourcade & al. 1993a] **Lafourcade M. & Sérasset G. (1993a)** *DOP (Dictionary Object Protocol).* GETA-IMAG, Grenoble, Common Lisp Object System (MCL - CLOS), Apple Macintosh, version 2.0.

[Lafourcade & al. 1993b] **Lafourcade M. & Sérasset G. (1993b)** *Geta-Grapher.* GETA-IMAG, Grenoble, Common Lisp Object System (MCL-CLOS), Apple Macintosh, version 1.1.

[Lay & al. 1992] **Lay M.-H., Zaysser L. & Flores S. (1992)** *Projet Eureka Genelex, le modèle syntaxique.* Rapport technique, Projet Eureka Genelex, 10 juin 1992, 107 p.

[Lenat & al. 1990] **Lenat D. B., Guha R. V., Pittman K., Pratt D. & Shepherd M. (1990)** *CYC: Toward Programs with Common Sense.* Communications of ACM, **33**/8: pp. 30-49.

[Mel'čuk 1984] **Mel'čuk I. (1984)** *DEC : Dictionnaire explicatif et combinatoire du français contemporain, recherche lexico-sémantiques I,* Presses de l'université de Montréal, Montréal(Quebec), Canada, 172 p.

[Mel'čuk 1988] **Mel'čuk I. (1988)** *DEC : Dictionnaire explicatif et combinatoire du français contemporain, recherche lexico-sémantiques II,* Presses de l'université de Montréal, Montréal(Quebec), Canada, 332 p.

[Mel'čuk 1992] **Mel'čuk I. (1992)** *DEC : Dictionnaire explicatif et combinatoire du français contemporain, recherche lexico-sémantiques III,* Presses de l'université de Montréal, Montréal(Quebec), Canada, 323 p.

[Melby 1988] **Melby A. k. (1988)** *Lexical Transfert: Between a Source Rock and a Hard Target.* Proc. Coling-88, Budapest, 22-27 août 1988, D. Vargha ed. vol. 2/2: pp. 411-413.

[Melby 1991] **Melby A. K. (1991)** *Pour le traducteur : un poste de travail à trois niveaux d'assistance.* Proc. L'environnement traductionnel ; La station de travail du traducteur de l'an 2001, Mons, Belgique, 25-27 avril 1991 vol. 1/1: pp. 151-153.

[Meyer & al. 1990] **Meyer I., Onyshkevych B. & Carlson L. (1990)** *Lexicographic Principles and Design for Knowledge-Based Machine Translation.* Technical Report,n° CMU-CMT-90-118, Carnegie Mellon University, August 13, 1990, 66 p.

[Miike 1990] **Miike S. (1990)** *How to Define Concepts for Electronic Dictionaries*. Proc. international workshop on electronic dictionaries, Oiso Kanagawa, Japan: pp. 43-49.

[Morin 1991] **Morin J.-Y. (1991)** *Intégration des connaissances en génie linguistique : niveaux, dimensions, objets et contraintes*. Proc. L'environnement traductionnel - La station de travail du traducteur de l'an 2001, Mons, Belgique, 25-27 avril, AUPELF&UREF, Presses de l'Université de Montréal, vol. 1/1: pp. 109-133.

[Nagao 1993] **Nagao M. (1993)** *Current Status and Future Trends of Natural Language Processing*. Proc. KB&KS'93, Tokyo, Japan, December 1993, JIPDEC ed., JIPDEC, vol. 1/1: pp. 31-39.

[Nagao & al. 1985] **Nagao N., Tsujii J. & Nakamura J. (1985)** *Terminology dictionary for machine translation*. Proc. Second Infoterm Symposium on Terminology, Wien, 15-18 avril 1985.

[Nédobejkine 1990] **Nédobejkine N. (1990)** *Représentation des informations lexicales dans les dictionnaires électroniques*. T.A. Informations, **31**/1: pp. 5-15.

[Nédobejkine 1991] **Nédobejkine N. (1991)** *Dictionary Approach in Natural Language Processing*. Proc. Third International Conference on Translation, Kuala Lumpur (Malaysia).

[Nirenburg 1987] **Nirenburg S., ed. (1987)** *Machine translation*. Studies in Natural Language Processing, A. K. Joshi ed., Cambridge University Press, Cambridge, 350 p.

[Nirenburg 1989a] **Nirenburg S. (1989a)** *KBMT-89 Project Report.,* Center for Machine Translation, Carnegie Mellon University, Pittsburg, avril 1989, 286 p.

[Nirenburg 1989b] **Nirenburg S. (1989b)** *Knowledge-based machine translation*. M.T., **4**/1: pp. 5-24.

[Nirenburg & al. 1990a] **Nirenburg S. & Defrise C. (1990a)** *Lexical and Conceptual Structure for Knowledge-Based Machine Translation*. Proc. ROCLING III, Taipeh, 20-22 August 1990 vol. 1/1: pp. 105-130.

[Nirenburg & al. 1990b] **Nirenburg S. & Goodman K. (1990b)** *Treatment of Meaning in MT Systems*. Proc. ROCLING III, Taipeh, 20-22 August 1990 vol. 1/1: pp. 81-101.

[Nirenburg & al. 1989] **Nirenburg S. & Levin L. (1989)** *Knowledge Representation Support*. M.T., **4**/1: pp. 25-52.

[Norvig 1992] **Norvig P. (1992)** *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp,* Morgan Kaufmann Publishers, San Mateo - California, 948 p.

[Phan 1991] **Phan H. K. (1991)** *Contribution à l'informatique multilingue, extension d'un éditeur de documents structurés*. Thèse nouveau doctorat, Université des sciences et techniques de Lille, Flandres Artois, 231 p.

[Phan & al. 1992] **Phan H. K. & Boitet C. (1992)** *Multilinguization of an editor for structured documents. Application to a trilingual dictionary*. Proc. COLING 92, Nantes, 23-28 July 1992, C. Boitet ed., ACL, vol. 3/4: pp. 966 - 971.

[Qi 1991] **Qi Y. (1991)** *Research and Development of the Chinese Dictionary Used for Multilingual Machine Translation*. Proc. International Symposium on

Multilingual Machine Translation (MMT-91), Beijing, August 19-21, 1991 vol. 1/1: pp. 59-61.

[Quint 1987] **Quint V. (1987)** *Une approche de l'édition structurée des documents.,* Universtité scientifique, technologique et médicale de Grenoble, 274 p.

[Quint 1989] **Quint V. (1989)** *Systems for Manipulation of Structured Documents.* In "Structured Documents", J. André, R. Furuta & V. Quint ed., Cambridge University Press: pp. 35.

[Quint & al. 1994] **Quint V. & Vatton I. (1994)** *Making Structured Documents Active.* Electronic Publishing, **7**/1:.

[Ricci 1986] **Ricci, ed. (1986)** *Dictionnaire français de la langue chinoise.,* Institut Ricci - Kuangchi Press, Paris & Taipei, 185 p.

[Sabah 1988] **Sabah G. (1988)** *L'intelligence artificielle et le langage. Volume 1 : Représentation des connaissances* M. Borillo & F. Nef ed., Langue - Raisonnement - Calcul, Hermès, Paris, 352 p.

[Sabah 1989] **Sabah G. (1989)** *L'intelligence artificielle et le langage. Volume 2 : Processus de compréhension* M. Borillo & F. Nef ed., Langue - Raisonnement - Calcul, Hermès, Paris, 411 p.

[Sabah 1993] **Sabah G. (1993)** *Knowledge Representation and Natural Language Understanding.* AICOM, **6**/3/4: pp. 155-186.

[Schneider 1989] **Schneider T. (1989)** *The METAL System. Status 1989.* Proc. MT Summit II, Munich, Germany, 16-18 August 1989, C. Rohrer ed. vol. 1/1: pp. 128-136.

[Seligman & al. 1994] **Seligman M. & Boitet C. (1994)** *A "whiteboard" architecture for automatic speech translation.* Proc. International Symposium on Spoken Dialogue, Waseda University, Tokyo, 1-12 November 1993: pp. 4-8.

[Sérasset 1992a] **Sérasset G. (1992a)** *Defining a Database — An example.* Technical report, Multilex ESPRIT project, May 1992, 33 p.

[Sérasset 1992b] **Sérasset G. (1992b)** *Defining a database — The language.* Technical report, Multilex ESPRIT project, May 1992, 17 p.

[Sérasset 1992c] **Sérasset G. (1992c)** *Psi-termes et dictionnaires.* Proc. Séminaire Lexique des pôles langage naturel et parole du GDR PRC CHM, Toulouse, janvier 1992 vol. 1/1: pp. 35-45.

[Sérasset 1994a] **Sérasset G. (1994a)** *Approche œcuménique au problème du codage des structures linguistiques.* Proc. TALN-94 : Le traitement automatique du langage naturel en France aujourd'hui, Marseille, 7-8 avril 1994, P. Blache ed. vol. 1/1: pp. 109-118.

[Sérasset 1994b] **Sérasset G. (1994b)** *An Interlingual Lexical Organisation Based on Acceptions, From the Parax Mock-up to the NADIA System.* Proc. ICLA-94, Penang, 26-28 July 1994 vol. 1/1: pp. 21-33.

[Sérasset 1994c] **Sérasset G. (1994c)** *Interlingual Lexical Organisation for Multilingual Lexical Databases in NADIA.* Proc. COLING-94, Kyoto, 5-9 August 1994, M. Nagao ed. vol. 1/2: pp. 278-282.

[Sérasset 1994d] **Sérasset G. (1994d)** *Peut-on coder un dictionnaire avec des Ψ-termes.* Turjuman, revue de Traduction et d'Interprétation, École supérieure roi Fahd de traduction, Tanger, **3**/1: pp. 41-56.

[Sérasset 1994e] **Sérasset G. (1994e)** *Recent Trends of Electronic Dictionary Research and Development in Europe.* Technical Memorandum,n° TM-038, Japan Electronic Dictionary Research Institute Ltd., 16 March 1994, 89 p.

[Sérasset 1994f] **Sérasset G. (1994f)** *Software architecture and tools.* In "MULTILEX" (title to be decided), K. Ahmad ed., (to be published) Springer Verlag.

[Sérasset & al. 1993] **Sérasset G. & Blanc É. (1993)** *Une approche par acceptions pour les bases lexicales multilingues.* Proc. T-TA-TAO 93, Montréal, 30 septembre-2 octobre 1993, A. Clas ed. vol. 1/1: pp. (à paraître).

[Shieber 1986] **Shieber S. M. (1986)** *An Introduction to Unification-Based Approaches to Grammar,* CSLI Lecture Notes, Center for the Study of Language and Information, Menlo Park, 105 p.

[Sperberg-McQueen & al. 1994] **Sperberg-McQueen C. M. & Burnard L., ed. (1994)** *Guidelines for Electronic Text Encoding and Interchange.,* Text Encoding Initiative, Chicago, Oxford, 1290 p.

[St Clair 1991] **St Clair B. (1991)** *WOOD: a Persistent Object Database for MCL.* Apple, Avalaible in MCL CD-ROM & FTP (cambridge.apple.com), version 1.0.

[Steele 1990] **Steele G. L. Jr. (1990)** *COMMON LISP. The Language,* Digital Press, 1030 p.

[Tsujii 1986] **Tsujii J.-I. (1986)** *Future Directions of Machine Translation.* Proc. Coling-86, Bonn, 25-29 août 1986 vol. 1/1: pp. 655-668.

[Tsujii 1988] **Tsujii J.-I. (1988)** *What is a cross-linguisticaly valid interpretation of discourse?* Proc. New Directions in Machine Translation, Budapest, 18-19 août 1988, D. Maxwell, K. Schubert & T. Witkam ed., Distributed Language Translation, T. Witkam, Floris Publications,: pp. 157-166.

[Tsujii 1989] **Tsujii J.-i. (1989)** *Machine Translation in Natural Language Understanding.* Literary and Linguistic Computing, **4**/3: pp. 214-217.

[Tsujii 1990] **Tsujii J.-i. (1990)** *Why do we need man-machine interaction in MT?* Proc. RocLing III, Taipeh, August 1990 vol. 1/1: pp. 133-138.

[Tsujii & al. 1993] **Tsujii J.-i. & Ananadiou S. (1993)** *Knowledge-based Processing in MT.* Proc. KB&KS'93, Tokyo, Japan, December 1993, JIPDEC ed., JIPDEC, vol. 1/1: pp. 70-82.

[Tsujii & al. 1990] **Tsujii J.-i. & Fujita K. (1990)** *Lexical Transfer based on Bi-Lingual Signes - Toward Interaction during Transfert.* Proc. RocLing III, Taipeh, August 1990 vol. 1/1: pp. 141-157.

[Uchida & al. 1991] **Uchida H. & Zhu M. (1991)** *Interlingua.* Proc. International Symposium on Multilingual Machine Translation (MMT-91), Beijing, 19-21 August 1991 vol. 1/1: pp. 22-30.

[Ullman 1980] **Ullman J. D. (1980)** *Principles of Database Systems,* Computer Science Press, Rockville, 378 p.

[Véronis 1992a] **Véronis J. (1992a)** *Disjunctive Feature Structures as Hypergraphs.* Proc. COLING-92, Nantes, 23-28 July 1992, C. Boitet ed. vol. 2/4: pp. 498-504.

[Véronis 1992b] **Véronis J. (1992b)** *A Feature-Based Model for Lexical Databases.* Proc. COLING-92, Nantes, 23-28 July1992, C. Boitet ed. vol. 2/4: pp. 588-594.

[Véronis & al. 1989a] **Véronis J., Ide N. M. & Hariè S. (1989a)** *Construction automatique de grands réseaux de neurones pour la désambiguisation du langage naturel.* Proc. 10èmes journées Systèmes Experts et leurs applications, conférence spécialisée : le traitement des langues naturelles et ses applications, Avignon, 28 mai-1 juin 1990: pp. 105-117.

[Véronis & al. 1989b] **Véronis J., Ide N. M. & Wurbel N. (1989b)** *Extraction d'informations sémantiques dans les dictionnaires courants.* Proc. 7ème congrès Reconnaissance des Formes et Intelligence Artificielle, AFCET RFIA, Paris: pp. 1381-1395.

[Véronis & al. 1990] **Véronis J., Ide N. M., Wurbel N. & Harié S. (1990)** *Construction et exploitation d'une base de données lexicale Franco-Anglaise: le projet Vassar/GRTC.* Proc. Journées nationales du PRC Communication Homme-Machine, Toulouse, Octobre 1990: pp. 115-124.

[Wilks 1986] **Wilks Y. (1986)** *An Intelligent Analyzer and Understander of English.* In "Readings in Natural Language Processing", B. J. Grosz, K. Spark Jones & B. L. Webber ed., Morgan Kaufmann Publishers, Inc., Los Altos, California: pp. 193-204.

[Wilks & al. 1990] **Wilks Y. & Farwell D. (1990)** *A White paper on Research in Pragmatic-based Machine Translation.* Memorandan in Computer and Cognitive Science,n° MCCS-90-188, Computing Research Laboratory, New Mexico State University, Las Cruces, NM, 25 p.

[Wilks & al. 1993] **Wilks Y. & Nirenburg S. (1993)** *Toward Automated Knowledge Acquisition.* Proc. KB&KS'93, Tokyo, Japan, December 1993, JIPDEC ed., JIPDEC, vol. 1/1: pp. 54-58.

[Yaoliang & al. 1991] **Yaoliang J. & Zhendong D. (1991)** *As a CICC MMT (ODA) Project.* Proc. International Symposium on Multilingual Machine Translation (MMT-91), Beijing, 19-21 August 1991 vol. 1/1: pp. 13-15.

[Yokota 1990] **Yokota E. (1990)** *How to Organise a Concept Hierarchy.* Proc. International workshop on electronic dictionaries, Oiso Kanagawa Japan, Japan Electronic Dictionary Research Institute, Ltd., vol. 1/1: pp. 50-57.

[Zajac 1988] **Zajac R. (1988)** *Operations on Typed Feature Structures: Motivations and Definitions.* Internal Report, ATR Interpreting Telephony Research Laboratories, 1988, 32 p.

[Zampolli 1973] **Zampolli A. (1973)** *L'automatisation de la recherche lexicographique : état actuel et tendances nouvelles.* **18**/1/2: pp. 103-138.

[Zaysser & al. 1992] **Zaysser L., Laporte E., Lay M.-H., Vandenbussche C. & Francopoulo G. (1992)** *Projet Eureka Genelex, couche morphologique.* Rapport Technique, Projet Eureka Genelex, 2 juin 1992, 97 p.

Annexes

Annexe A : Introduction à SGML

SGML[6] (Standard Generalized Markup Language) est un standard international pour la définition de méthode de représentation de documents sous forme électronique.

SGML est un métalangage, c'est à dire un moyen de définir formellement un langage permettant la représentation d'un document électronique. Le langage ainsi défini est un langage d'étiquettes.

Une "étiquette" est un moyen de donner une information, ou une annotation sur le contenu d'un texte de document.

Un langage d'étiquette permet de définir l'ensemble des étiquettes autorisées, l'ensemble des étiquettes requises, comment ces étiquettes sont distinguées du texte et ce qu'elles signifient. SGML donne des moyen de définir les trois premiers points, le quatrième étant à la charge de l'utilisateur.

SGML est un formalisme descriptif permettant de définir un type de documents (Document Type Definiton ou DTD). Une DTD contient une description des étiquettes autorisées et requises. Chaque document SGML sera associé à un type. Ce type est la grammaire qui permettra de manipuler le document.

Un texte n'est pas une suite de mots que l'on ne peut différencier. Il peut être divisé en différentes unité textuelles. Un article par exemple peut être divisé en une suite de paragraphes, de chapitres, etc.

Le document est donc divisé en *éléments*. SGML ne permet pas la définition de la sémantique d'un élément, mais il permet de définir les relations qu'il entretient avec d'autres éléments.

À l'intérieur d'un texte étiqueté (une instance de document), chaque élément doit être explicitement repéré. Le moyen le plus couramment utilisé pour cela est l'insertion d'une étiquette au début et à la fin d'un élément. Ces étiquettes agissent donc comme des parenthèses sur le texte. Ainsi, une citation peut être indiquée comme suit :

```
... Rosalind's remaks <quote>This is the silliest stuff that ere I heard
of!</quote> clearly indicate ...
```

[6] Cette partie est inspirée du chapitre 2 de [Sperberg-McQueen & al. 1994].

Dans cet exemple, le début de l'élément est indiqué par l'étiquette `<quote>`. Sa fin est indiquée par `</quote>`. "quote" indique le nom de l'élément. L'utilisation des chevrons (< et >) et du slash (/) est la convention standard utilisée en SGML (cette convention peut être redéfinie).

Un élément peut être vide, contenir un simple texte, on contenir un ensemble d'autres éléments d'un type différent. Cela permet le codage de structure plus complexes.

Supposons que l'on souhaite coder une anthologie, qui contiendra un ensemble de poèmes dont on connaît le titre, et un ensemble de strophes (elles même constituées d'un ensemble de vers).

Un tel document apparaîtra sous la forme suivante[7]:

```
<anthology>
 <poem><title>The SICK ROSE</title>
   <stanza>
      <line>O rose thou art sick.</line>
      <line>The invisible worm,</line>
      <line>That flies in the night</line>
      <line>In the howling storm:</line>
   </stanza>
   <stanza>
      <line>Has found out thy bed</line>
      <line>Of crimson joy:</line>
      <line>And his dark secret love</line>
      <line>Does thy life destroy.</line>
   </stanza>
 </poem>
 <poem>

      <!-- more poems go here -->

</anthology>
```

Des blancs et retours à la lignes ont été introduits pour simplifier la lecture du document. Ils ne jouent aucun rôle dans la définition de la structure du document. De plus, la ligne :

```
      <!-- more poems go here -->
```

est un commentaire SGML qui n'est pas traité comme une partie du texte.

Cet exemple ne fait aucune hypothèses sur les règles qui gouvernent la structure d'une anthologie. Pourtant, on peut définir des règles qui permettront de simplifier l'étiquetage du document :

- une anthologie contient des poèmes, et rien d'autre,
- un poème a un seul titre qui précède la première strophe et qui ne contient pas d'autre élément,
- mis à part le titre, un poème ne contient que des strophes,
- une strophe ne contient qu'un ensemble de vers,
- seuls une strophe ou un autre poème peuvent venir à la suite d'une strophe,
- seuls un vers ou une strophe peuvent venir à la suite d'un vers.

À partir de ces règles, on peut inférer qu'il n'est pas nécessaire d'étiqueter explicitement la fin des vers et des strophes. La deuxième règle implique qu'il est inutile de marquer la fin d'un titre (qui est implicitement marquée par un début de strophe). De la même manière, il est

---

[7] Cet exemple est extrait de "Songs of innocence and experience" (1974) de William Blake.

inutile d'étiqueter explicitement la fin d'un poème. Ainsi, on peut représenter le même document de la manière suivante :

```
<anthology>
 <poem><title>The SICK ROSE
   <stanza>
      <line>O rose thou art sick.
      <line>The invisible worm,
      <line>That flies in the night
      <line>In the howling storm:
   <stanza>
      <line>Has found out thy bed
      <line>Of crimson joy:
      <line>And his dark secret love
      <line>Does thy life destroy.
 <poem>

      <!-- more poems go here -->

</anthology>
```

Les règles, telles celles décrites ci-dessus, sont la première étape du processus de création d'une spécification formelle de la structure d'un document SGML. Cette description est une "Définition de Type de Document" (DTD).

Ainsi, la DTD correspondant à l'exemple ci-dessus est définie comme suit :

```
<!ELEMENT anthology    - -      (poem+)>
<!ELEMENT poem         - O      (title?, stanza+)>
<!ELEMENT title        - O      (#PCDATA)>
<!ELEMENT stanza       - O      (line+)>
<!ELEMENT line         - O      (#PCDATA)>
```

On a ainsi défini les différents éléments du document. On a pu indiquer l'optionnalité ou l'obligation des étiquettes de début et de fin. Ainsi, dans la ligne :

```
<!ELEMENT title        - O      (#PCDATA)>
```

le "-" indique l'obligation de l'étiquette de début et le "O" indique que l'étiquette de fin peut être omise.

En troisième partie de la définition de chaque élément (entre parenthèses) apparaît la définition de son contenu. Le contenu peut être une suite de caractères `(#PCDATA)` ou un ensemble d'éléments `(title?, stanza+)`. Le "`?`" à la suite d'un élément indique sont optionnalité. Un "`+`" à la suite d'un élément indique que cet élément est obligatoire et peut être répété. Un "`*`"à la suite d'un élément indique que cet élément peut être répété, mais n'est pas obligatoire. Un "`,`" entre deux élément indique leur mise en séquence.

Il est possible d'associer des *attributs* à chacun des éléments d'un document. Il est ainsi possible d'associer un identificateur à un élément particulier ou de lui associer un numéro, un statut, etc. Ainsi, pour pouvoir associer un identificateur et un statut à un élément de type `poem`, on va ajouter la définition suivante dans la DTD `anthology`:

```
<!ATTLIST poem
          id        ID                                #IMPLIED
          status    (draft | revised | published)  draft     >
```

Dans cette définition, on indique à quel élément sont associés les attributs définis. Pour chaque attribut, on donne un nom, un ensemble de valeur, et une valeur par défaut. `ID` est un type de valeur spécial permettant de donner un nom unique à un élément du document. Sa

valeur est calculée automatiquement (`#IMPLIED`). L'attribut `status` peut avoir trois valeurs : `draft`, `revised`, `published`, avec `draft` comme valeur par défaut.

Les concepts exposés jusqu'alors portent sur la dénotation d'une structure dans le document. SGML propose aussi un moyen simple et souple de coder et de nommer des parties arbitraires du contenu d'un document, de manière portable. Cette possibilité est offerte par les *entités*.

Une entité est une partie nommée du contenu du document, indépendamment de sa structure. Par exemple, la déclaration suivante :

```
<!ENTITY tei "Text Encoding Initiative">
```

définit une entité de nom `tei` et dont la valeur est la chaîne "Text Encoding Initiative". On peut référer à ces entités à l'intérieur d'un document en insérant leur nom, précédé d'un "`&`" et terminé par un point-virgule. Ainsi, le texte "`La &tei; est une initiative`…" est équivalent au texte "`La Text Encoding Initiative est une initiative`…".

Ce mécanisme d'entité est utilisé notamment pour coder les caractères diacrités dans un texte que l'on souhaite pouvoir passer d'une machine à une autre.

Annexe B : Introduction à GRIF

GRIF[8] est un système interactif de production de documents structurés. Il permet de manipuler des documents complexes comportant des formules mathématiques, des tableaux, des schémas, etc., en mettant l'accent sur l'organisation logique des documents. Il est disponible sur station de travail Unix et utilise le système de fenêtrage X.

Tous les traitements que GRIF peut effectuer s'appuient sur un *modèle de document* de haut niveau. Pour GRIF, un document est d'abord *une structure logique*. Ainsi, un article scientifique est considéré comme une suite d'éléments typés : un titre, un ou plusieurs noms d'auteurs, un résumé et une suite de section, le résumé étant formé de paragraphes et chaque section comportant un titre, quelques paragraphes et une suite de sections de niveau inférieur. Cette organisation convient bien à un article, mais ne permet pas de représenter correctement un livre ou une lettre. C'est pourquoi il n'y a pas de modèle unique de document, mais un *méta-modèle* qui permet de décrire plusieurs modèles, un pour chaque *classe de documents*.

Grâce au méta-modèle, on peut définir une structure logique pour la classe de documents *Article*, une autre pour la classe de documents *Livre*, une autre pour la classe de documents *Lettre*… Un langage, appelé S, permet de spécifier des structures logiques génériques de ces classes de documents sous la forme de *schémas de structure*.

L'un des intérêts de l'utilisation d'une structure logique est qu'elle permet de produire automatiquement l'aspect graphique des documents. À chaque type d'élément de la structure logique générique est associé un ensemble de *règles de présentation* qui définissent l'aspect graphique de ce type d'élément. En appliquant ces règles aux éléments de la structure logique spécifique, on peut construire l'image du document. On débarrasse ainsi l'utilisateur du travail de mise en forme du document, on assure une bonne homogénéité de la présentation (tous

[8] Le nom Grif est associé à trois choses différentes : 1) un prototype en cours de développement à l'INRIA dans le cadre du projet Opéra (à Grenoble et à Rennes), 2) un produit commercial issu du précédent et 3) la société Grif SA, qui commercialise le produit précédent (Grif SA, 2, bd Vauban, BP 266, 78053 St Quentin en Yvelines Cedex). Nous ne parlons ici que du prototype 1.

les éléments de même type sont présentés par application des mêmes règles), et on peut assurer, lorsque c'est nécessaire, que le document est présenté selon un modèle imposé.

Un ensemble de règles de présentation définissant l'aspect graphique de tous les types d'éléments d'une classe de documents est appelé un *schéma de présentation*. Les schémas de présentation sont écrits dans un langage appelé P, qui permet aux utilisateurs de spécifier leurs propres présentations. Si l'on spécifie plusieurs schémas de présentation pour une classe de documents, on peut voir le même document sous différentes formes graphiques, sans affecter ni son contenu, ni sa structure logique.

L'impression peut se faire directement par GRIF (qui génère du Postscript™ et donne alors un comportement WYSIWYG) ou via un formateur. Cette seconde option fait intervenir un troisième langage, le langage T, qui permet d'exprimer des règles de traduction, regroupées dans des *schémas de traduction*. Un schéma de traduction spécifie le transcodage des caractères ainsi que des chaînes de caractères à engendrer pour chaque élément du document, en fonction de son type et de sa position dans la structure logique du document.

Un programme de traduction fait partie du système. Il lit un document produit par l'éditeur et, en suivant les règles exprimées dans un schéma de traduction, il produit un document traduit. Comme pour les schémas de présentation, on peut définir plusieurs schémas de traduction pour une même classe de documents, ce qui permet de traduire les documents d'une classe dans plusieurs formalismes différents.

La traduction peut être utilisée pour produire des documents acceptables par un formateur comme TeX ou LaTeX. Elle peut aussi être utilisée pour coder les documents selon un standard comme SGML.

Le langage S permet de décrire des structures logiques de classes de documents. Ces structures se présentent comme une grammaire hors contexte augmentée dont les terminaux sont les éléments textuels. On peut associer aux non-terminaux de la grammaire un ensemble d'attributs. Un attribut peut servir pour l'interfaçage (position d'un élément dans une page…), pour une référence (identificateur d'un paragraphe, d'une note de bas de page…) ou pour ajouter une information ne faisant pas partie du contenu d'un document (la langue d'un paragraphe, sa date de création…).

Pour illustrer GRIF, nous prendrons le même exemple que dans l'annexe précédente. Une anthologie contient un ensemble de poèmes dont on connaît le titre, et une suite de strophes (elles-mêmes constituées de vers).

Cette structure se traduit en S par :

```
STRUCTURE Anthology;
DEFPRES AnthologyP;

STRUCT
     Anthology (ATTR editor = TEXT) = BEGIN
             Anthology_title = TEXT;
             Poems = LIST OF (Poem);
     END;

     Poem    = BEGIN
             Title = TEXT;
             Author = TEXT;
             Stanzas = LIST OF (Stanza);
     END;
```

```
    Stanza  = BEGIN
              Lines = LIST OF (Line);
    END;

    Line    = TEXT;
END
```

Ici, l'élément *Anthology* a un attribut *editor* dont la valeur est un texte.

Étant donnée cette structure, on peut définir une présentation la reflétant dans un document. Cette présentation est définie en associant, à chaque élément de la structure, une boîte de présentation, dont la position et la taille sont définies en fonction de celle des boîtes voisines.

Par exemple, la boîte correspondant au titre de l'anthologie est centrée par rapport à sa boîte contenante (celle correspondant à l'anthologie), et son texte est écrit en Helvetica 14 gras :

```
Anthology_title:
BEGIN
   HorizPos: VMiddle = Enclosing . VMiddle;
   VertPos: Top = Enclosing . Top;
   Size: 14;
   Font: Helvetica;
   Style: Bold;
END;
```

Un poème est aligné à gauche, et se trouve 1 cm en dessous du poème précédent :

```
Poem:
 BEGIN
    HorizPos: Left = Enclosing . Left;
    VertPos: Top = Previous Poem . Bottom + 1 cm;
 END;
```

Son titre est présenté en Times 12 gras, aligné à gauche :

```
Title:
BEGIN
   VertPos: Top = Enclosing . Top;
   HorizPos : Left = Enclosing . Left;
   Size: 12;
   Font: Times;
   Style: Bold;
END;
```

L'auteur est présenté en Times 10 italiques, avec un retrait à gauche d'un demi centimètre :

```
Author:
BEGIN
   VertPos: Top = Previous Title . Bottom;
   HorizPos: Left = Enclosing . Left + 0.5 cm;
   Size: 10;
   Font: Times;
   Style: Italics;
END;
```

L'ensemble des strophes d'un poème est aligné à gauche et son texte est en Times 10. On laisse un espace de 0,7 centimètre après le nom d'auteur :

```
Stanzas:
BEGIN
   VertPos: Top = Previous Author . Bottom + 0.7 cm;
   HorizPos: Left = Enclosing . Left;
   Size: 10;
   Font: Times;
   Style: Roman;
END;
```

De la même manière, une strophe se trouve un demi-centimètre en dessous de la strophe précédente, et alignée sur la gauche, en Times 10 (la typographie a été héritée de la boîte contenante `Stanzas`):

```
Stanza:
BEGIN
    VertPos: Top = Previous Stanza . Bottom + 0.5 cm;
    HorizPos: Left = Enclosing . Left;
END;
```

Ainsi, on a défini une présentation pour la classe de document *Anthology*. Nous donnons une instance de cette classe de document dans la figure B.1.

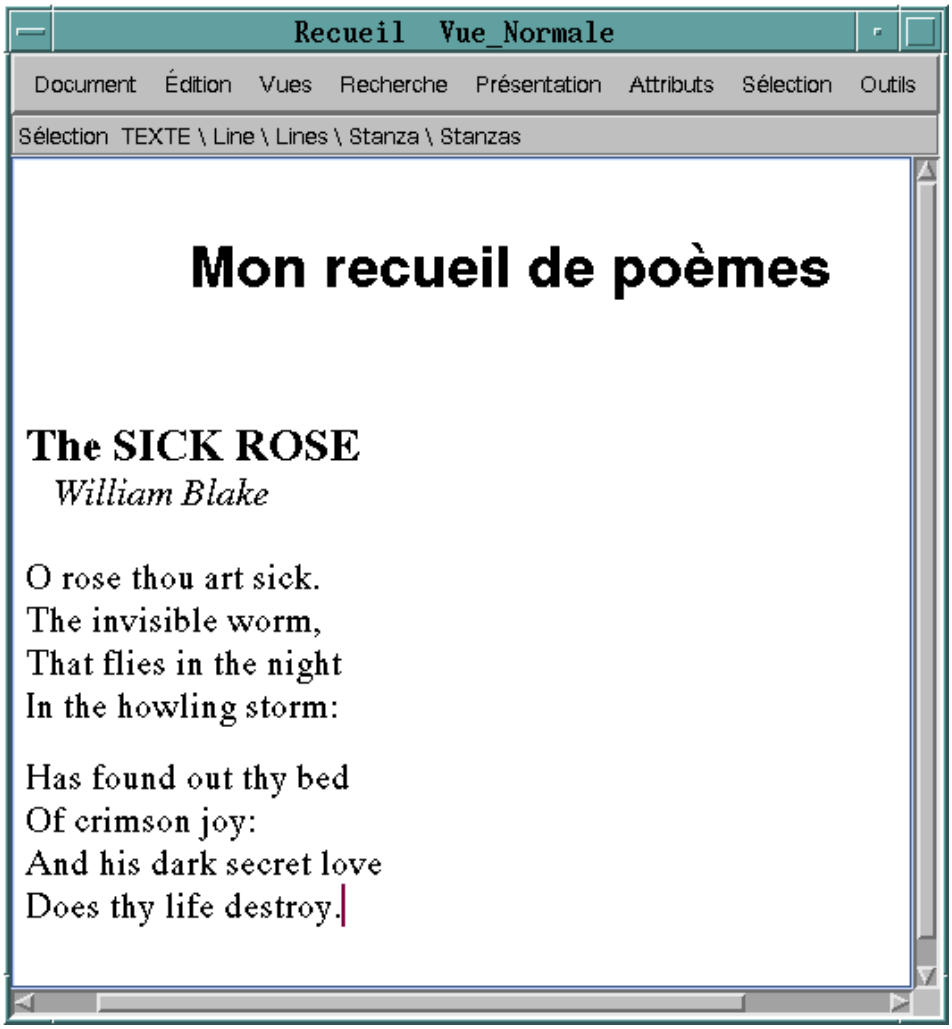


Figure B.1 : Une instance de document de la classe *Anthology*

Enfin, grâce au langage T, il est possible d'exporter les documents de cette classe sous certaines formes. Dans l'exemple proposé, nous allons exporter vers la forme SGML présentée dans l'annexe précédente.

Dans le format que l'on veut produire, on doit générer les étiquettes `<anthology>` et `</anthology>` autour du document :

```
Anthology:
BEGIN
   Create "<anthology>" before;
   Create "</anthology>" after;
END;
```

De plus, le titre de l'anthologie n'apparaît pas. On ne lui associe donc pas de règle de traduction.

Les étiquettes <poem> et </poem> apparaissent autour de chaque poème :

```
Poem:
BEGIN
   Create "<poem>" before;
   Create "</poem>" after;
END;
```

Le titre d'un poème apparaît entre les étiquettes <title> et </title>:

```
Title:
BEGIN
   Create "<title>" before;
   Create "</title>" after;
   Create content;
END;
```

Chaque strophe est entourée de <stanza> et </stanza>:

```
stanza:
BEGIN
   Create "<stanza>" before;
   Create "</stanza>" after;
END;
```

Enfin, chaque vers est entouré de <line> et </line>:

```
Line:
BEGIN
   Create "<line>" before;
   Create "</line>" after;
   Create content;
END;
```

D'autre mécanismes permettent la génération de formats d'export plus compliqués (conditions d'application de règles, sortie vers plusieurs fichiers, traduction des caractères spéciaux, utilisation de compteurs…).

Annexe C : Exemples d'articles du Dictionnaire Explicatif et Combinatoire du Français Contemporain

Nous donnons ici 5 articles du Dictionnaire Explicatif et Combinatoire du Français contemporain, extraits du volume I de ce dictionnaire.

Nous avons sélectionné deux noms, un verbe et deux adjectifs afin de donner une idée la plus exacte possible des structures de ce dictionnaire. Le lecteur souhaitant des renseignements plus approfondis sur la théorie sous-jacente à ce dictionnaire peut consulter les articles qui se trouvent au début de chaques volumes du DEC.

CŒUR, nom, masc.

I.1a.	Organe principal de la circulation sanguine d'une personne… [*le cœur de Jean*]
1b.	Organe principal de la circulation sanguine d'un animal… [*le cœur de lion*]
2.	Produit alimentaire … [*le cœur de veau*]
3.	Partie de la poitrine d'une personne … [*Il a serré son fils sur son cœur*]
4a.	Organe imaginaire des sentiments … [*Le cœur espère toujours*]
4b.	Organe imaginaire de l'intuition … [*Son cœur le lui dit*]
5a.	… propriété de la personnalité … [*un cœur de glace*]
5b.	Personne possédant le cœur I.5a [*Vous devez la vie à un noble cœur, à un homme vaillant*]
II.1a.	Partie principale d'une unité fonctionnelle… [*le cœur du bateau*]
1b.	Élément principal [*le cœur du problème*]
2a.	Partie centrale d'un espace… [*le cœur du royaume*]
3.	Objet… ayant la forme du cœur I.1a [*un cœur en papier*]
4.	Une des quatre couleurs 2 des cartes à jouer… [*l'as de cœur*]
III.	Organe imaginaire des nausées … [*Cette senteur lui tournait le cœur*]

I.1a. *Cœur de X* = Organe principal de la circulation sanguine d'une personne X qui se trouve dans la partie centrale du corps II.1d de X et qu'on représente symboliquement comme ayant la forme ♡.

Connotations

1) Cœur I.1a est le siège des sentiments [voir CŒUR I.4a].

2) Cœur I.1a est le siège de l'intuition [voir CŒUR I.4b].

3) Cœur I.1a qui bat 1 représente la vie [voir les phrasèmes correspondants dans CŒUR I.1a].

Régime

1 = X
1. *de* N
2. A_{poss}

C1 : *le cœur de Marie, s[illegible] cœ[illegible]*

[illegible]ns lexicale

Gener : organe[illegible]erne[illegible] *[illegible]eur, le foie, le[illegible]umons et autres organes internes viscères*

A_0 : de [ ] [*mala[illegible] cc[illegible]*, cardiaq[illegible] *[illegible]ardiaque, palpitations cardiaques [illegible]*]

Bon = Ver : bon | prépo[illegible] [illegible]issant[illegible]

AntiBon = AntiVer : mauvais | pr[illegible]

Loc_{in} : dans [le [illegible] *[illegible]e découvr[illegible] [illegible]ard[illegible]ans le cœur ; ress[illegible] un[illegible] [illegible]eur*]

Loc_{ad} : en [plein] [*Il a reçu la[illegible] plein cœur*]

$Oper_1$: avoir [ART] | C. a un dépendant [*avoir le cœur malade sain*]

S_1 (Oper$_1$ + AntiBon) : cardiaque 2

$Fact_0$: battre 1; se contracter

$SingS_0 Fact_0$: battement [de]

F_1 = $FinFact_0$: s'arrêter

F_1 comme conséquence de $Excess^{usual}$: céder

$S_0 FinFact_0$: arrêt [de ART]

de nouveau $CausFact_0$: ranimer [ART]

$CausFact_0^{ATTR}$ PlusBon :soutenir, stimuler [ART]

Son = $Fact_0$: cogner, battre 2 [*J'entends bat... cœur*]

$SingS_0$ Son : coup [de ART] [*J'... les coups rapides de son cœur*]

$S_0 AntiFact_{actual,0}$:// attaque, crise (cardiaque 1)

$AntiFact_{usual,0}$: **fam** battre la breloque, **fam** ... détraqués

$S_0 AntiFact_{usual,0}$: maladie 1a [de] ... insuffisance cardiaque 1

Degrad : faiblir, flancher

$Degrad^{actual}$: [Cl_{dat}] manquer [*À cette nouv... qui m...a*]

F_2 = $Caus_{usual,1}$Excess : se fatiguer [le]

$nonPerm_{usual,1}$Excess :**fam** soigner 2, ménager [A_{poss}]

$Excess^{actual}$: palpiter, accélérer

$S_0 Excess^{actual}$: // **spéc** palpitations (cardiaques 1), battements précipités [de ART]

AntiBon comme conséquence de $Excess^{usual}$: usé

Stop(C.) —
$Sympt_{13}$(*peur*, *émotion* ,…) :s'arrêter [(de N)]

Stop(C.) — $Sympt_{123}$ (émotion forte) : [Cl_{dat}] flanche [... *cœur lui flanche*]

F_3 = Stop(C.) — $Sympt_{13}$ (*chagrin*) : se rompre, se briser [(de N)]

$CausF_3$: rompre...

F_4 = Excess(C.) — $Sympt_{13}$(émotion forte) : **fam** battre la cha... (de N)]

$Adv_1 F_4$: le battant [*Nous l'attendons le cœur batta...*]

Excess(C.) — $Sympt_{13}$(*peur, horreur, effroi, chagrin, désespoir*) : se serrer, cogner, battre vite fort [(de N)]

Excess(C.) — $Sympt_{13}$(*joie ,amour*) : bondir, ... palpiter, frémir [(de N)]

Excess(C.) — $Sympt_{13}$(*pitié, chagrin, angoisse*) : se serrer [(de N)]

en forme de C. : en [ ] [*un ornement en cœur*]

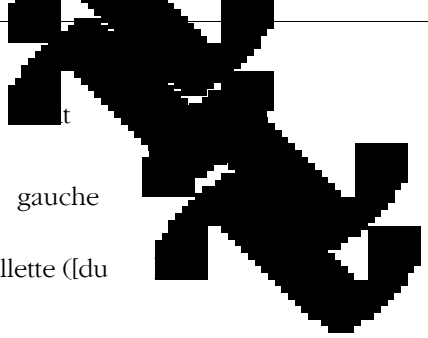
Parties du cœur

F_5 = moitié latérale
droite du C. : t

F_6 = moitié latérale
gauche du C. : gauche

cavité dans la partie
supérieure de F_5 et de F_6 :// oreillette ([du

cavité dans la partie
inférieure de F_5 et de F_6 :// ventricule ([du])



Affections du cœur

F_7 = syncope provoquée par un court arrêt du C. entraînant
une grande pâleur : syncope blanche

maladie 1a entraînant
de fréquentes F_7 : maladie 1a blanche

malformation du C.
chez les nouveau-nés : maladie 1a bleue

avoir une lésion au C. entraînant un souffle
[bruit anormal] : avoir un souffle [au]

hémorragie dans le C. : infarctus

syndrome caractérisé par des douleurs dans
la région du C. : angine de poitrine

personne qui a une
affection du C. : cardiaque 2

Traitement du cœur

discipline médicale
s'occupant du C. : // cardiologie

examen des bruits du C. : // ausculation cardiaque 1

étude des enregistrements graphiques des
mouvements du C. : // cardiographi

médicament pour le C. : // potion cordiale

opération sur le C. qui
continue à battre 1 : opération [à fermé]

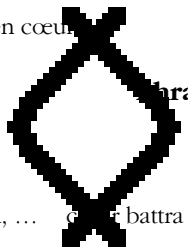
opération sur le cœur
qui est arrêté : opération [à ouvert]

pile électrique pour
stimuler le C. : stimulateur cardiaque

Exemples

Le cœur te flanche, ma beauté? [J. Giono]. Sous l'influence de causes diverses, en particulier une mauvaise nouvelle, ou un coup violent porté à l'estomac, le cœur peut s'arrêter pendant quelques instants, en même temps que la respiratio : la syncope est réalisée [P. Vallery-Radot]. Il montait s'arrêtant toutes les deux marches, reprenant souffle, attendant que se calment un peu les battements précipités de son cœur [A. Gide]. À l'annonce de cette nouvelle, il éprouva une vive douleur au cœur. Son pauvre petit cœur se mit à battre la chamade. S'il continuait malgré tout, c'est que son cœur était solide. Le cadavre était froid; le cœur avait cessé de battre depuis longtemps.

◊
(Faire) la bouche en cœur



rasèmes reliées à CŒUR I.1a **par la connotation**

'cœur I.1a qui bat 1 représente la vie'

Tant que ce mon, … battra

Ce son, … s'est arrêté

I.1b. *Cœur de X* = Organe principal de la circulation sanguine d'un animal X.

Régime

1 = X
1. *de* N
2. A_{poss}

C_1 : *le cœur d'un animal, son cœur*

Fonctions lexicales

Toutes les FL, sauf $nonPerm_{usual,1}$ Excess et les FL contenant Sympt : ∧ CŒUR I.1a

Exemples

La bête s'effondra, atteinte au cœur en plein cœur …

I.2. *Cœur de X* = Produit alimentaire — cœur I.1b d'un animal de boucherie X.

Régime

Le … d'veau … apprécié … le cœur de bœuf. Jeanne le prépare farci.

I.3. *Cœur de X* = Partie de la poitrine de la personne X, sous laquelle se trouve le cœur I.1a de X.

Employé dans quelques expressions signifiant 'X cause que Y soit en contact avec le cœur I.2 de X, ce qui manifeste l'affection de X pour Y' (comme, par ex., *serrer presser, étreindre quelqu'un contre sur son cœur, mettre porter quelque chose sur son cœur*).

Régime

1 = X
1. *de* N
2. A_{poss}

C_1 : *le cœur de Jean, son cœur*

Fonctions lexicales

Syn : poitrine I.1a, sein 2

Exemples

Après, elle s'est assise la main sur le cœur dans son fauteuil [É. Ajar]. Elle s'assit en prenant son fils entre ses deux genoux, et le pressant avec force sur son cœur, elle l'embrassa [H. de Balzac]. Quand un chanteur met la main sur son cœur, cela veut dire d'ordinaire : je t'aimerai toujours! [Ch. Baudelaire]. Il porte une croix sur son cœur. Il mit la lettre sur son cœur et

s'endormit. Il longeait le précipice, serrant l'enfant contre son cœur. Elle déposa des fleurs sur le cœur de Jean. Elle s'endormit, la tête sur le cœur de Jean.

I.4a. *Cœur de X éprouvant Y* [*à l'égard de Z*] = Organe imaginaire d'une personne X moyennant lequel X éprouve le sentiment Y (à l'égard de Z) [comme si cet organe se trouvait dans le cœur I.1a].

Régime

1 = X	2 = Y	(3 = Z)
1. *de* N 2. A_{poss}	1. *en* N 2. A	———

1) $C_{2.1}$: N = *joie, peine*,… [mais pas, par exemple, *admiration* ou *angoisse*]
2) $C_{2.2}$: A = *joyeux, haineux, effrayé*,… [mais pas, par exemple, *admiratif*]
C_1 : *le cœur de ma mère, son cœur*
Impossible : **Le cœur en angoisse, il marchait le long de la rivière* (1) [= *Le cœur angoissé*,…]
**Le cœur admiratif, il contemplait ce tableau* (2)

Fonctions lexicales

Syn : âme 1a
Contr : tête I.2, esprit; corps II.
Loc_{in} : en, dans [le/[illegible] C_1 /= [illegible] *cœur*]
Loc_{ab} : du plus pro[illegible]d [du[illegible] *profond du cœur, je vous en fais la promesse*]
Adv_1Real_1 : avec [le] [*Ce livre est écrit avec le cœur*]
$Real_2$: $Func_1(M_2(C.))$ [*La haine dévore son cœur*; *La joie dilate son cœur*; *L'amour enflamme son cœur*; *L'angoisse agite son cœur*,…]
$Fact_0$: $V_0(M_2(C.))$ [*[illegible] cœur souffre, aimait, regrette, s'inquiète, s'émeut*,… , mais pas *[illegible]r a honte*]
$SingS_0MagnFact_0$: coup [au]
$nonFact_0$: se taire
X^1 et X^2 ont les C. qui
$Fact_0$ de la même façon : ne former qu[illegible]'une âme
$Able_1Fact_0$: sensible, ten[illegible] jeun[illegible]
qui n'est plus jeune mais dont le C. est
$Able_1Fact_0$: jeune de
$Able_1MagnFact_0$: chaud, ardent
$nonAble_1Fact_0$: vide, aride
$nonAble_1Fact_0$ en conséquence de trop
$Fact_0$: blasé, lassé

façon de parler émotionnelle qui

$Caus_3Fact_0$: éloquence du

$Caus_3Fact_1$: [Cl_{dat}] touche, trouble, agite, [illegible]; [illegible] aller [illegible], parler [à] faire vibrer les cordes [du]

$Caus_3Fact_{12}$: [Cl_{dat}] remplir, [illegible] [N] [*Cette nouvelle me remplit le cœur d'espoir*]

Adv_2Fact_{13} : de [ ] [*catholique de cœur*]

$Fact_2$: éprouver, ressentir [N]

$MagnFact_2$: être plein, déborder, brûler, se consumer [de N] | Y est un sentiment fort [*Mon cœur est plein déborde [illegible]*]

$IncepFact_2$: se remplir [de N]

$Fact_3$: $v_0(M_2(C.))$ [(PREP) N] [*Son cœur s'attache facilement aux personnes* ; *Mon cœur espère cette rencontre*]

$Able_1Fact_3$: sensible [à N] [*Son cœur fut sensible à ces paroles*]

$IncepPredMinusAble_1Fact_3$: faiblir

$nonAble_1Fact_3$: inaccessible [à N]

$Labreal_{12}$: avoir [N au] [*Il [illegible] cœ*

F_1 = Z étant très important pour X,

$Labreal_{13}$: avoir, tenir [N à [illegible] *cette affaire [illegible] cœur*]

$IncepF_1$: prendre [N à [illegible] pour N]

Z étant important pour X,

$Labreal_{31}$: [Cl_{dat}] tenir [illegible] *me tient à cœur*]

$Conv_{21}Manif$: ven[illegible] du [illegible] [*Cette [illegible] vient du cœur*]

mots de X - spontanément

S_2Manif : cri [du]

F_2 = (en disant à W ses sentiments) ^{ATTR}

$Caus_1Manif$: ouvrir, découvrir, expliquer [A_{poss}] [illegible], livrer, montrer [A_{poss}] à nu [*Il mit son cœur à nu*]

un peu F_2 : ouvrir [un coin de A_{poss}]

volontairement $Caus_1Manif$:épancher, vider, déch[illegible]

deviner les sentiments de X sans que X

$Caus_1Manif$: lire [dans le/A_{poss}], sond[illegible] $_{poss}$]

$nonPerm_1Manif$: cacher [A_{poss}]

$Degrad$: vieillir

$Excess — Sympt_{23}(Y)$: frémir, tressaillir, tressauter, bondir [de N = Y]

$Excess — Sympt_{213}$(un fort sentiment Y) : [Cl_{dat}] sauter dans la gorge [de N = Y]

une partie du C. telle que son contenu est perçu ou admis par X : fond, replis, [illegible]oin | le plus souvent avec Loc_{in} [*Au fond [illegible] cœur Dans les replis de son [illegible] le secret de son cœur il [illegible] sentait encore de l'amour*]

avoir dans le C. de la sympathie pour la personne W : être [de avec N = W]

F_3 = faculté de X d'éprouver dans son C., en se souvenant des
qui avaient provoqué des sentiments forts, ces
mêmes sentiments : **litt** mémoire I.1 [du]

$IncepMagnReal_2(F_3)$: // se graver au fond

Y = am

$Real_3$: vivre [dans AR]

$Caus_3Fact_{actual,0}$: conquérir, gagner [ART]

$Caus_3Fact_{usual,0}$: attirer [ART]

$Fact_3$: être [à N] [*Mon cœur e*

$F_4 = S_{instr}Caus_3Fact_3$: chemin, clé [du

trouver F_4 : trouver [le chemin la clé

$Caus_3Fact_{actual,3}$: [se] aliéner [ART]
le fait que la personne Z aimée de X ne contacte plus X

$LiquFact_3$: **prov** Loin des du

$Labreal_{13}$: porter [N s

état des C. des personnes qui
s'aiment : uni [des

A_0 : de [ART drame, peine, sa
dame

$IncepReal_{13}$: donner [A_{poss} à

A_1Fact_0 : pris [*Elle a le cœur p*], épri

$Caus_3Fact_0$: posséder [ART], être maître [de ART]

$A_1nonFact_0$: libre, à prendre

$Fact_3$: appartenir [à N]
beaucoup de X +

$Caus_3Fact_3$: traîner tous [les a

homme —

$S_1Able_1Caus_3Fact_3$: bourreau [de s] au pl

femme —

$S_1Able_1Caus_3Fact_3$: charmeuse [d s] [

F_5 = dans le but que Z $Caus_3Fact_0$, $Caus_1Manif$
à Z : offrir [A_{poss} à N

F_6 = en réponse à F_5, Z
$Caus_1Manif$ à X : accorder [A_{poss} à N = X]

$AntiF_6$: refuser [A_{poss} à N = X]

Y = chagrin

$Fact_0$: saigner, pleurer

$CausFact_1$: [Cl_{dat}] blesser, b[illegible]er, br[illegible]ver, déchirer, [illegible], oppresser, percer, transpercer, serrer, ronger [le ~ : *Cette nouvelle lui brisa le cœur*]

$S_{res}CausFact_1$: blessure [au ~]

Y = dé[illegible]

$Real_1$: avoir [le ~ à V_{inf}] [*Il ava[illegible]* ~ *rire*]

$AntiAdv_1Real_1$: à contrecœu[illegible]

X refuse volontairement ce que son C. $Fact_0$: bouder [contre A_{poss} ~]

$Fact_1$: [le/A_{poss} ~ di[illegible]] [*Mon cœur me dit de me promener*]

= joi[illegible]

$Real_1$: avoir [du ~ à N] [= *ouvra[illegible] travai[illegible]*…]

Adv_1Real_1 : de bon ~ grand, tout ~ [illegible]gaieté de ~ [*Il accepta de bon cœur*]

Y = plaisir

$Real_3$: être [[illegible]n A_{poss} ~]

$Fact_0$: chanter

A_1Fact_0 : le ~ en fête

$Caus_3Fact_{actual,0}$: captiver [ART ~] [*Ce chant captivait [illegible]*]

$Caus_3Fact_1$: réchauffer, réjouir [ART ~]

$Fact_3$: y être [*Il faisa[illegible]vait[illegible]vec di[illegible]gence, mais le cœur n'y é[illegible]t pas*]

Y [illegible]

$F_7 = Real_1$: avoir [le ~ [illegible]voir un poignard ~ une épine [illegible]s le ~]

$AntiF_7$: avoir [le ~ [illegible]r]

à cause d'un amour 2 non partagé, $Real_1$: avoir [[illegible]ché[illegible]pe ~]

$Real_3$: [Cl_{dat} [illegible]r sur le ~]; rester en travers [du ~] [*Le silence de ce[illegible]mme[illegible]ta sur le cœur*]

$F_8 = A_1Fact_0$: le ~ gros lourd ~

$AntiF_8$: le ~ léger | postpos

$CausFact_1$: [Cl_{dat}] glacer, **litt** transir, oppresser [le ~ [illegible]pe[illegible] Cl_{dat}] pincer, ronger, empoisonner [le ~ (C.) = *haine, é[illegible]*]

$CausMinusFact_1$: [Cl_{dat}] soulager, apaiser, [illegible]mer[illegible]

$Fact_2$: être bourrelé [de re[illegible] (C.[illegible] *remords*

$Labreal_{13}$: avoir (fort) [N sur l[illegible]], en[illegible]co[illegible] V_{inf} sur le ~] [*J'ai ce soufflet fort sur le cœur* ; *Il en avait gros su[illegible]ur à leur dire adieu*]

$ContLabreal_{13}$: garder [N sur le ~]

Exemples

C'est moi qui suis le Seigneur qui sonde les cœurs, et qui éprouve les reins… [Bible]. Ce n'est pas de gaieté de cœur qu'il renonce aux certitudes métaphysiques [F. Maurois]. L'égalité est l'idéal de l'esprit de l'homme, et l'inégalité le penchant de son cœur [É. Bourges]. … pendant que la bouche accuse, le cœur absout [A. de Musset]. L'horrible silence qui y régnait me glaçait le cœur [A. France]. Un espoir immense me gonfle le cœur [G. Duhamel]. On n'a plus le cœur jeune impunément quand le corps a cessé de l'être [J.-J. Rousseau]. Et pourtant c'eût été si bon, au milieu de tant de deuils et de tristesse d'avoir un peu d'amour pour se chauffer le cœur! [A. France]. Il pleure dans mon cœur/ Comme il pleut sur la ville,/ Quelle est cette langueur/ Qui pénètre mon cœur? [P. Verlaine]. Le cœur a ses raisons que la raison ne connaît point [B. Pascal]. Jeune de cœur : c'est la vraie jeunesse [J. Giono]. D'un côté, c'est le cœur qui commande […], de l'autre, c'est votre cervelle et elle se sert librement de votre corps [J. Giono]. Cœur qui soupire n'a pas ce qu'il désire [proverbe]. Une chaumière et un cœur, c'était là toute son ambition. Cette attention délicate me va droit au cœur. Mais ils étaient de cœur avec la rébellion. L'œuvre de Tibulle est de celles qu'on ne peut comprendre qu'avec le cœur. Vous prenez la chose fort à cœur. Ce ue l'intelligence a acceptée doit se transplanter dans le cœur. Calvin insiste sur le fait que la religion chrétienne ne touche pas uniquement l'esprit, mais aussi le cœur et consiste dans la conviction inébranlable de l'esprit et du cœur. Je lui ai vidé mon cœur. Ne vous excusez pas : c'était le cri du cœur. J'ai à cœur de vous prévenir. C'est à contrecœur et bien malgré lui, que le duc d'Albe exécutait les instructions de Philippe II. Bude put alors se remettre à ses études; celles-ci lui tenaient tellement à cœur qu'il ne sut même pas s'en arracher le jour de son mariage. Elle garde, gravé au fond de son cœur, le jugement de Pâris, l'injure de sa beauté méprisée. Quelques mots mélancoliques dans une lettre nous ouvrent un coin de son cœur. Ils font le siège d'un même cœur de femme. Dès qu'on commençait à être gai, on avait le cœur qui s'ouvrait. Pour ce que vous m'avez révélé au sujet de cette pauvre Marie, j'en ai le cœur brisé. Marie s'est flattée qu'elle était la première et la seule à avoir ému son cœur. L'homme fort doit accepter d'un cœur égal les maux auxquels il ne peut rien.

◊
À cœur joie
À cœur ouvert
Avoir le cœur sur les lèvres
Cœur à cœur
Comme un cœur
De tout A$_{poss}$ cœur
En avoir le cœur net
Faire le joli cœur
Mon cœur [*Veux-tu, mon cœur?*]

I.4b. *Cœur de X* [*percevant Y*] = Organe imaginaire de l'intuition d'une personne X moyennant lequel X perçoit Y [comme si cet organe se trouvait dans le cœur I.1a].

Régime

1 = X	2 = Y
1. *de* N	———
2. A$_{poss}$	

C$_1$: *le cœur d'une mère, son cœur*

Fonctions lexicales

Syn : âme 1b
Real$_1$: écouter [son]
nonFact$_0$: se taire
AntiVer^{ATTR} Fact$_0$: se tromper
nonAble$_1$Fact$_0$: muet
F$_1$ = Fact$_1$: [Cl$_{dat}$] dire, souffler, pr
F$_2$ = S$_{instr}$F$_1$: voix [du]
Real$_1$(F$_2$) : écouter [la voix du]
AntiReal$_1$(F$_2$) : étouffer [la voix du]
Fact$_1$(F$_2$) : [Cl$_{dat}$] dire, souffler, prédire [N]
Fact$_2$: sentir, deviner [N]

Exemples

Je sais qu'il m'aime; mon cœur ne se trompe jamais. Malgré les serments répétés de son amant, son cœur lui disait qu'il lui était infidèle.

I.5a. *Cœur Y de X* = Cœur I.4a de X tel qu'il assure la propriété Y de la personnalité de X, cette propriété consistant en un fort penchant pour certains sentiments.

Régime

1 = X	2 =
1. *de* N	1.
2. A pos	2. A

$C_1 + C_2$: *un cœur de glace sensible, généreux, de pierre, son cœur d'or fidèle, juste,*

Fonctions lexicales

Syn : âme

S_1 : co

Oper$_1$: avoir

F_1 = C.^{2} bon : d'or en or

$A_1 + F_1$: de [*perso*

CausPredF$_1$: attend [PT]

C : av

Contr : il [re

C.^{2} génér : grand gra mai Oper$_1$

F_2 = C.^{2} no : bie haut é,

S_0F_2 : no

S_0Ant$_2$: ba se [it AR] [*ur*

F_3 = C.^{2} fi : d'aigle

F_4 = C.^{2} courageux : bien en cro , de fer

CausPredF$_4$: affermir, fo r [A

Oper$_1 + F_4$: avoir [du] au ventre

Imper(Oper$_{actual,1} + F_4$) : Haut les s! | C pl

dans les épreuves

Oper$_1 + F_4$: **prov** faire re m fo e bon

F_5 = Oper$_{actual,1}$ + si courageux que X peut

Z-er : avoir [le de = Z

ne pas Conv$_{21}$ F_5 : [Cl$_{dat}$] manquer

CausFunc$_1 + F_4$: donner [du à N] au ventre [*Les paroles de son chef lui donnèrent du cœur au ventre pour affronter l'ennemi*]

Caus$_1$Func$_0$ + F$_4$: se mettre [d[illegible]] a[illegible]

$_2$ bienveillant : plei[illegible]ordial 2 [*mot d'accueil cordial*]

A$_1$ [illegible]

$_1$enveillant : //[illegible]dia[illegible]

C.$_2$ h[illegible] : double[illegible]

insen[illegible]ble : s[illegible]airan[illegible]e, **litt** de granit, **vieilli** de br[illegible]

Caus1Pred$_6$F$_2$: [Cl$_{dat}$] durcir [A[illegible]*re[illegible]malheurs lui durcissent le cœur*]

méchant : noir, de v[illegible]

S$_0$(F$_7$) : noirceur du [illegible]

A$_1$ + F$_7$: sans-cœur [*ce voyou sans*[illegible]

PredF : être[illegible]manquer [illegible]

bien que X dis[illegible]es

ch[illegible]geables/Za F : **pr**[illegible]Soud[illegible]de fiel

C.$_2$ [illegible]reux : de poulet

C.$_2$ infidèle : d'artichaut

CausPred(C.$_2$ malhonnête) : dépraver, corrompre [ART]

Exemples

Les natures au cœur sur la main ne se font pas l'idée des jouissances solitaires de l'hypocrisie…[Barbey d'Aurevilly]. Comme un soldat qui prend la goutte à boire pour se mettre du cœœur au ventre… [J. Giono]. Mais je n'aurais jamais le cœur de pouvoir préférerl'un de vous deux à l'autre. Ce voyou sans cœur et sans honneur, ce bandit! S'il te reste un cœur, attends jusqu'à demain! Ceux qui avaient encore un peu de cœur l'ont perdu. Cet acte révèle la noirceur de son cœur.

I.5b. *Cœur Y* = Personne possédant le cœur I.5a Y [= S_1 (*cœur I.5a*)].

Régime

1 = Y
1. *de* N
2. A

C_1 : *un c[illegible]r de fer*

Fo[illegible]lexicales

Les FL Syn et celles de type C.$_2$ M_2 : /\ C[illegible] I.5a[illegible]

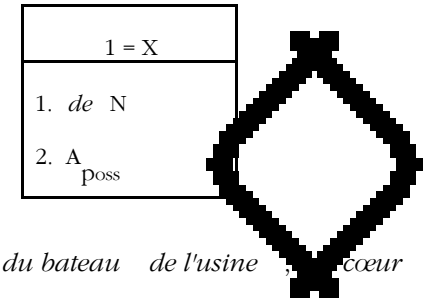
C. courageux peut tout réussir : **prov** À vaillant, rien d'impossible

Exemples

Quoi? dans leur dureté ces cœur d'acier s'obstinent [P. Corneille]. C'est un cœur de fer, indomptable. Vous devez de très humbles excuses à un noble cœur, votre fils.

II.1a *Cœur de X* = Partie principale d'une unité fonctionnelle X où l'activité caractéristique de X est la plus intense.

Régime



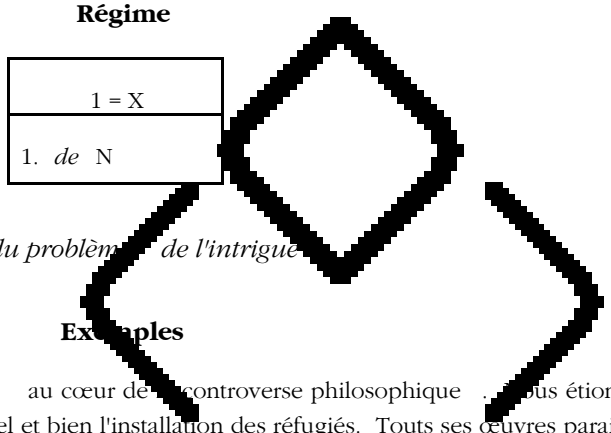
C_1 : *le cœur du bateau de l'usine cœur*

Exemples

La rue Sainte-Catherine est le cœur de Montréal. En un point qui n'est pas tout à fait le centre du terrier, mais quia été mûrement choisi pou un cas d'extrême péril, j'ai bâti le cœur de ma citadelle. La chaleur produite dans le cœur du réacteur est transférée par un fluide caloporteur à un circuit eau-vapeur.

II.1b. *Cœur de X* = Élément principal de X [= FL non standard «partie principale»(X) | X = *débat, sujet, question, problème, discussion, querelle, contrverse, document* ,…, mais pas **poème* , **roman* ,…].

Régime



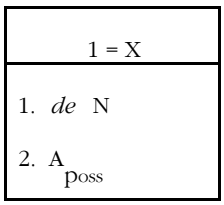
C_1 : *le cœur du problème de l'intrigue*

Exemples

Cette attitude nous mène au cœur d'une querelle au cœur de la controverse philosophique ... nous étions parvenus au cœur de sujet. Mais le cœur du document était bel et bien l'installation des réfugiés. Touts ses œuvres paraissent découler d;une conception centrale, du cœur mystérieux de sa philosophie.

II.2a. pas de pl. *Cœur de X* = Partie centrale d'un espace topographique X.

Régime



C_1 : *le cœur du royaume , son cœur*

Fonctions lexicales

Syn : centre, milieu

Loc_{in} : dans [A[illegible]]

$Magn_{['centrale']} + Loc_{in}$: en plein

Loc_{ab} : du [ ]

Exemples

C'est au cœur de cette forêt que se trouvaient les deux colonnes [G. de Sède]. Il s'agit pour lui à la fois de reculer ses frontières jusqu'au cœur de la Germanie [G. de Sède]. … les oasis les plus douces, les plus riches du cœur de l'Asie [J. Kessel]. Voilà l'ennemi dans le cœur du royaume! Pourquoi lui, Séjan, chef des 10 000 légionnaires qui gardaient le cœur de l'Empire romain, ne deviendrait-il pas le maître de cet Empire tout entier? Une source qui jaillit directement du cœur du rocher. Il faut au moins protéger le cœur de ce jardin. Des lianes moussues au cœur des buissons de lilas. Les manifestations se sont rendues en cortège au pied du tombeau en plein cœur de la capitale polonaise.

II.2b. *Cœur de X* = Partie centrale — en épaisseur — d'une plante X ou de la partie X d'une plante, qui est perçue comme distincte des autres parties de X.

Régime

⩚ CŒUR II.2a

C_1 : *le cœur de ce bouleau, son cœur*

Fonctions lexicales

Toutes les FL : ⩚ CŒUR II.2a

enlever le C. : // **spéc** décœurer [*décœurer une pièce de bois*]

Exemples

Ils devaient aussi se repasser le cœur de la salade, le blanc de la poularde et le foie du lapin! [M. Pagnol]. Les vieilles souches (de vigne0 sont pourries [illegible] au cœur, et le fruit n'en vaut guère [P.-L. Courier]. Ces troncs d'arbre échoués sur les plages […] et que le soleil et [illegible]ier ont desséchés jusqu'au cœur [S. Schwartz-Bart]. Le cœur du bouleau est malade. Un ver sortit du cœur de la pomme. Les deux [illegible]eds de laitue étaient aussi verts que la jeune herbe tendre ; leurs feuilles cachaient le cœur blanc et repliaient l[illegible]unes sur les [illegible]res.

◊

À cœur [*fromage fait [illegible] cœur*]

Au cœur En plein cœ[illegible] d[illegible]

Comme le cœur de la cheminée [*noir comme le cœur de la cheminée*]

II.3. *Cœur de X* = Objet en matière X ayant la forme symbolique du cœur I.1a.

Régime

1 = X
1. *en* N

C_1 : *un cœur en carton papier, sucre, tissu ,…*

Exemples

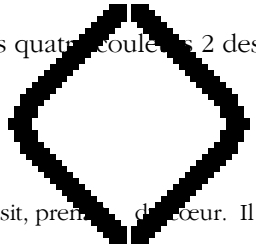
Elle portait au cou un cœur suspendu à une chaîne.

◊

Cœur de Marie de Jeannette [illegible]

Cœur-de-pigeon

II.4. pas de pl. Une des quatre couleurs 2 des cartes à jouer dont les points ont la forme symbolique du cœur I.1a de couleur 1 rouge.

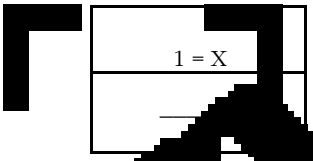


Exemples

As de cœur. Il jouait choisit, prem[illegible] de cœur. Il avait un beau jeu à cœur.

III. *Cœur* [*de X*] = Organe imaginaire des nausées d'une personne X, perçu comme étant proche du cœur I.1a de X.

Régime



Fonctions lexicales

$Real_1$: avoir mal [à son cœur] à l'envers

S_0Real_1 : mal de cœur

mouvement réflexe provoqué par S_0Real_1

avant que X vomisse : haut-le-cœur

$A_1CausFact_0$: écœurer

$Fact_1$: [Cl$_{dat}$] [le] soulever

$CausFact_1$: donner [à N] mal au cœur, lever, soulever, retourner, barbouiller [le cœur] / écœurer

être près de vomir : voir [le cœur] sur le bord des lèvres

F_1 = éprouver la sensation de pesanteur au C. à cause d'un repas Y : avoir [N = Y sur ART cœur] [*J'ai mon repas gâteau ,… sur le cœur*]

Co_{21} : N = Y [Cl$_{dat}$] rester, demeurer, peser [sur le cœur]

Exemples

Quelque mal de cœur que me causât le balancement de la voiture… [J.-F. Marmontel]. Les sachets de naphtaline dont la senteur poivrée lui tournait le cœur… [M. du Gard].

◊ Mal au cœur 2

♦ Par cœur

MÉPRIS, nom, masc.

I. Attitude émotionnelle défavorable… [*le mépris pour ce corrupteur*]

II. Opinion selon laquelle quelque chose n'a pas d'importance… [*le mépris du danger des convenances*]

I. *Mépris de X envers Y pour Z* = Attitude émotionnelle défavorable de X à l'égard de Y causée par le fait suivant : X croit que les actions, l'état ou les propriétés Z de Y causent que Y n'a pas de valeur morale ou sociale; cette attitude est celle qu'on a normalement dans de pareilles situations.

Régime

1 = X	2 = Y	3 = Z
1. *de* N	1. *de* N	1. *pour* N
2. A_{poss}	2. *pour* N	
3. A	3. *envers* N	
	4. *à l'égard de* N	

1) C_3 sans C_2
2) $C_{1.1}$ + $C_{2.1}$
3) $C_{2.2}$ + $C_{3.1}$
: impossible

4) $C_{1.2}$ + $C_{2.1}$: impossible si C_2 désigne une personne

5) $C_{1.3}$ + $C_{2.1}$: non souhaitable

C_1 : *le mépris de Paul, son mépris, le mépris populaire*

C_2 : *le mépris de Paul pour ce collègue, son mépris envers ce collègue, son mépris à l'égard de ce collègue, son hypocrisie*

$C_1 + C_2 + C_3$: *le mépris de Paul pour son mépris, le mépris populaire envers à l'égard de ce ministre pour son hypocrisie ses propos diffamatoires, le mépris de l'art pour son inefficacité*

Impossible : **son mépris pour l'hypocrisie* (1) [= *son mépris envers ce ministre pour son hypocrisie* ; le syntagme *son mépris pour l'hypocrisie* a un autre sens : *hypocrisie* correspond à Y et non à Z]
**le mépris de Paul de ce collègue* (2) [= *le mépris de Paul pour ce collègue*]
**le mépris pour ce collègue pour son hypocrisie* (3) [= *le mépris pour ce collègue à cause de son hypocrisie*]
**son mépris de Pierre* (4) [= *son mépris envers Pierre*]

Non souhaitable : ?*le mépris populaire de l'argent* (5) [= *le mépris populaire pour l'argent*]

Fonctions lexicales

Syn : dédain, irrespect, condescendance, arrogance, hauteur II, morgue, **litt** mésestime, **litt** superbe

Anti : respect I

Anti : considération, égard; d[illegible]érence, [illegible]

V_0 : mépriser I

A_0 : méprisan[illegible] 1 [*atti[illegible]nte*]

Gene[illegible] : attitude, s[illegible]tim[illegible] [*[illegible]t d'un profond mépris*]

S_1 : **litt** conte[illegible]r

$S_{2/3}$: objet [de [illegible] *Sa prise de p[illegible]tion [illegible] objet d'un grand m[illegible]ris*]

$Magn_{quant,1}$ + S_2 : paria

A_1 : plein [de [illegible] **Il s'est montré ple[illegible] de mépri[illegible]** *d'un mépris ins[illegible]* [illegible] / [illegible]éprisan[illegible] 2 [*Quand il voit [illegible] de malho[illegible]*

A_2 : couvert [de [illegible]

tel qu'il doit $PredA_{2/3}$: digne [de [illegible] *[illegible] méprisable*]

$PredAble_1$: enclin, porté [à ART], **fa**[illegible]e] facile | C_2 =

$PredAble_{2/3}$: mériter [ART]

$Qual_1$: dédaigneux, hautain, snob, a[illegible]nt, condescendant

$Qual_2$: abject, ignoble, infect, inf[illegible]e, bas, [illegible]isérable 2, sordide, dégoûtant, répugnant, vil, indigne, o[illegible]eux

Magn : grand, profond, absolu, sou[illegible] sans bornes; hautain, froid

PredMagn : ne pas connaître [illegible] avo[illegible] d[illegible]es

$Magn_{quant,1}$: collectif, public, g[illegible]rsel

IncepPredPlus : s'accroître [*Le mépris de l'a[illegible]oît de plus en plus dans certains groupes sociaux*]

$Caus_{(3)}$PredPlus : accroître [ART] [*La cupidité de leu[illegible]r n'a fait qu'accroître leur mépris de l'argent*]

IncepPredMinus : diminuer, s'atté[illegible]*n mépris s'[illegible] atténué après qu'il l'eût connu*]

Ver : justifié, (bie[illegible]dé, [illegible] m[illegible]e

AntiVer : injustifié [illegible] no[illegible]

Adv_1 : dans [A[illegible] = vivre, se[illegible] *vit dans un profond [illegible]ris p[illegible]e sont p[illegible] de s[illegible] social*]; ave[illegible] [ART [illegible]] | C_2(G) = [illegible]s co[illegible]es avec [illegible]ris sou[illegible]

Propt : par [ ] [*[illegible]a refusé de[illegible] [illegible]s juges*]

$Oper_1$: avoir, épro[illegible]] | C[illegible]

$Oper_2$: être en [illegible] ART [illegible] $_{ATTR}$ $Magn_{quant,1}$ [*Jean [illegible] m[illegible] tou[illegible]onde*], être vi[illegible]e [de ART], subir [ART]

$IncepOper_2$: tomber, **litt** sombrer [dans les] [*Le président La science [illegible] suite de ces événements, tomba dans le mépris général*]

$Magn_{quant,1}$ + $CausOper_2$: // mettre, clouer [N = Y] au pilori | Y désigne une personne

$FinFunc_0$: disparaître [*Son mépris des gens peu fortunés a disparu*]

$Caus_2Func_0$: encourir [ART] [*J[illegible]u le mépr[illegible]* atttirer [ART] | Y désigne une personne, et C_1 / [illegible] Ma[illegible] q[illegible] *l'est attiré le mépris général par son hypoc[illegible]*

$Caus_3Func_1$: engendrer [ART [illegible]N] [*La familiarité [illegible]e le mépris*]

$Caus_{(3)}Func_1$: apprendre, incul[illegible] [à N] [*[illegible]n inculque à ses étudiants le mépris de l'hypocrisie ; Son attitu[illegible]nvers ses e[illegible]és apprend à ces derniers le mépris de leur ch[illegible]*

$Caus_{(2/3)}Func_1$: inspirer [ART à N] [*C[illegible]ment ins[illegible]x tr[illegible]rs le mépris de leur patron ; L'argent [illegible]pir[illegible] [illegible] [illegible] l'il a donné son héritage à son frère ; L'hyp[illegible] [illegible] leu[illegible]nspi[illegible]profond mépris*]

$Caus_3Func_2$: attirer, valoir [A[illegible] | C_1 = / [illegible] A[illegible] [illegible]ant,1 *déclarations [illegible]ttirè[illegible] [illegible] d'u[illegible]on[illegible] de citoyens*]

$Labor_{12}$: tenir [N en [illegible]ndant/ da[illegible]RT [illegible][illegible]ndant] | Y désigne une personne [*Pierre[illegible]t Jea[illegible]*]

$Conv_{21}Manif$: dénoter, mont[illegible]

malgré X, $Conv_{21}Manif$: trahir [ART]

A_2Manif : plein, empr[illegible]nt [d[illegible]A[illegible] [illegible] / C_2 mépr[illegible]nt I.1 [illegible]

$Caus_1Manif$: manifester, r[illegible]er, m[illegible]ter, affi[illegible]icher, té[illegible]oign[illegible] [illegible], preuve [de [illegible]ART[illegible]

$SingS_0Caus_1Manif$: manifestation, signe, [illegible] preuve, geste 2 [de [illegible]ART[illegible]

(des paroles ou un comportement grossier en public)

$Caus_1Manif$: // conspuer, huer, baf[illegible] **li**[illegible]nd[illegible] **vieilli** honnir [N[illegible]

(des paroles ou un comportement grossier envers Y)

$Caus_1Manif$: [Cl_{dat}] cracher [A_{poss}] à la figure [*Je voudrais les écraser sous m[illegible] pieds ; leur cracher mon mépris à la figure*]; couvrir [N de ART [*El[illegible]e couvrit de son mépris en le toisant des pieds à la tête*]

F_1 = la personne pour laquelle le locuteur

$Caus_1Manif$: canaille, **fam** co[illegible] **fam** salaud, **fam** ordure, **fam** cochon, **fam** sale type, **pop** c[illegible]illon, **p[illegible]** [illegible]e, **pop** saligaud, **pop** fumier, …

$Mult(F_1)$: engea[illegible]ce

Mult(personnes ou choses pour lesquelles[illegible] lo[illegible]te[illegible]

$Caus_1Manif$) : ramass[illegible] [illegible] N]

Adv_1Caus_1Manif : avec [[illegible]ART[illegible] [illegible] *[illegible]e toisa avec mépris comme le fit Charles ce jour-[illegible]*

$Perm_1Manif$: déverser [ART

$nonPerm_1Manif$: refréner [ART

un sourire spécifique —

$Sympt_{23}$: sourire [de

S_0(sourire en grima-

çant — $Sympt_{23}$: rictus [de]

ricaner — $Sympt_{23}$: ricanet [de]

regarder d'une façon effrontée — $Sympt_2$: // toiser [N] (des pieds à la [illegible])

hausser les épaules — $Sympt_{23}$: hausser les épaules [de]

Exemples

L'Anglaise reconnut sa rivale et fut glorieusement anglaise; elle nous enveloppa d'un regard plein de son mépris anglais et disparut dans la bruyère avec la rapidité d'une flèche [H. de Balzac]. Je vais peut-être vous paraître vieux jeu, mais j'ai un mépris sans bornes pour ces femmes qui vont d'amant en amant, le plus souvent sans amour, pour des raisons de prestige ou de carrière [A. Maurois]. Rien ne m'a plus donné un absolu mépris du succès que de considérer à quel prix on l'obtient [G. Flaubert]. Le mépris âcre et froid des passants lui pénétrait dans la chair et dans l'âme comme une bise.

II. pas de pl. *Mépris par X de Y* = S_0 (*mépriser II*).

Régime

1 = X	2 = [illegible]
1. A_{poss}	1. *d*[illegible] N obl[illegible]

C_2 : *le mépris des lois de la liberté [illegible]resse, du danger*

$C_1 + C_2$: *son mépris de l'horaire de la décisio[illegible] de sa femme, des insultes*

Fonctions lexicales

Magn[illegible], IncepPred[illegible]us : /\ MÉPRIS I

Syn : dédain, indifférence, détachement

Anti : respect II.1; respect II.2

Anti : considération; attachement, inté[illegible] crainte

V_0 : mépriser II

Magn : grand, absolu, total; hautain

CausPredPlus : accroître, augmenter [ART] [*L'attitude du [illegible] accroissait son mépris des menac*[illegible]

IncepPredMinus : diminu[illegible]'a[illegible] [*Le [illegible] vie s[illegible]nue*]

Ver : justifié

AntiVer : injustifié, insensé[illegible]

Adv_1 : au [ ], avec [illegible]T [illegible] =/ [illegible]épris du [illegible]g[illegible], [illegible]ncé à la *poursuite de l'agresseur* [illegible] *ration avec un mépris total des insultes*]

$Oper_1$: avoir [ART] [*L[illegible] un mépris to[illegible] liberté de presse*]

$CausOper_1$: conduire, pouss[illegible] à A[illegible] [*Se[illegible]marades le conduisent au mépris des humiliations*]

$CausFunc_1$: inspirer [ART à N] [*Cet événement inspire aux citoyens le mépris du danger*]

Conv$_{21}$Manif : [N] montre, prouve [ART], témoigne [ART] *montre prouve, témoigne de* [*Pendant l'interrogation, cette femme a manifesté démontré* [ART] *grand mépris des humiliations ; Cet explorateur a fait preuve d'un mépris absolu du danger*]

Caus$_{1}$Manif : manifester, montrer, démontrer [ART], faire preuve [de [ART]] [*Pendant l'interrogation, cette femme a manifesté démontré* [ART] *grand mépris des humiliations ; Cet explorateur a fait preuve d'un mépris absolu du danger*]

Exemples

… poussant le mépris des scrupules presque aussi loin que le respect de l'étiquette [M. Proust]. Ce dont je suis sûr, c'est qu'on fait tuer les jeunes d'abord parce que les hommes très jeunes ont, plus que les autres, le hautain mépris de la vie [G. Duhamel]. Au mépris des réalités techniques, cette expression [*le petit écran*] assimile à un écran de cinéma la partie du tube cathodique où se forment les images de télévision. Elle a déploré leur mépris évident des valeurs démocratiques. Il avait toujours eu le mépris de son confort.

ENSEIGNER, verbe.

1. X cause que Z apprenne III.1b … [*X enseigne les mathématiques aux étudiants*]

2a. X énonce une affirmation qui fait partie de sa doctrine… [*Socrate enseignait à ses disciples que…*]

2b. X contient une affirmation qui fait partie d'une doctrine… [*La Bible enseigne que…*]

3a. Propriété ou action de X cause que Z apprenne Ia Y [*L'histoire nous enseigne que…*]

3b. Propriété ou action de X cause que Z apprenne IIa Y [*La servitude nous enseigne la ruse*]

1. *X enseigne Y à Z* = X, censé avoir la qualification professionnelle dans le domaine Y, cause que Z apprenne III.1b Y en transmettant, méthodiquement et dans un cadre officiel, à Z des connaissances (portant sur) Y ou des techniques (portant sur) Y [$CausConv_{21}$(*apprendre III.1b*)].

Régime

1 = X	2 = Y	3 = Z
1. N	1. N 2. *à* V_{inf}	1. *à* N 2. **rare** N

1) $C_{2.2}$ sans $C_{3.1}$
2) $C_2 + C_{3.2}$: impossible

$C_1 + C_2$: *Pierre enseigne la grammaire la couture à faire cela*

$C_1 + C_2 + C_3$: *Pierre enseigne la grammaire à ses élèves*

Impossible : **Pierre enseigne à danser (1) [= Pierre enseigne à danser aux enfants]*
**Pierre enseigne des enfants à lire (2) [= Pierre enseigne des enfants]*

Fonctions lexicales

Syn : apprendre III.2a, instruire 1a, **vieilli** professer 1

$Conv_{231}$: s'enseigner [*La chimie ne peut s'enseigner sans manipulations*]

S_0 : enseignement 1a

$S_{usual,1}$: enseignant I; maître II.2, instituteur, professeur; précepteur

S_2 : matière III

S_3 : élève, étudiant

livre-S_{instr} : manuel

$Able_2$: enseignable [*Le tact est difficilement enseignable*]

Exemples

Il comprenait et retenait aisément tout ce qu'on lui enseignait [A. Lesage]. Ces diverses sciences sont enseignées dans les écoles par des spécialistes. Quelle est ta profession? — J'enseigne. Il m'a enseigné à ne négliger aucun détail. On comprend aisément que pour enseigner aux enfants ayant des aspirations et des niveaux intellectuels si variés et surtout pour enseigner aux jeunes gens d'âge ingrat, il faut au maître une habilité pédagogique particulière.

2a. *X enseigne Y à Z* = X énonce une affirmation Y_1, qui fait partie d'une doctrine Y_2 proposée 4a par X, dans le but de causer que Z sache I le contenu de Y.

Régime

1 = X	2 = Y	3 = Z
1. N	1. N 2. *que* PROP	1. *à* N 2. **rare** N

1) C_2 : obligatoire s'il n'y a pas de $C_{3.2}$
2) $C_2 + C_{3.2}$: impossible

$C_1 + C_2$: *Les philosophes enseignent l'égalité entre les hommes que les hommes sont égaux*

$C_1 + C_2 + C_{3.1}$: *Socrate enseignait à ses disciples que la connaissance de soi est fondamentale*

$C_1 + C$: *Allez enseigner toutes les nations…* [Bible]

Fonctions lexicales

Syn : prêcher, professer 2
S_1 : maître II.5, gourou
S_2 : enseignement 2
S_3 : disciple

Exemples

Darwin enseignait que les espèces sont issues les unes des autres selon les lois de la sélection naturelle. Il faut toujours enseigner la vérité aux hommes. Pythagore enseignait qu'après la mort nous renaissions dans la nature.

2b. *X enseigne Y à Z* = X contient une affirmation Y_1 qui fait partie d'une doctrine Y_2 proposée 4a dans X [comme si X enseignait 2a Y à Z].

Régime

1 = X	2 = Y	3 = Z
1. N	1. N 2. *que* PROP obligatoire	1. *à* N

$C_1 + C_2$: *La Bible enseigne la transcendance de Dieu que Dieu est transcendant*

$C_1 + C_2 + C_3$: *La Bible nous enseigne que Dieu est transcendant*

Fonctions lexicales

S_2 : enseignement 2

Exemples

Le christianisme enseigne qu'il faut aimer son prochain comme soi-même.

3a. *X enseigne Y à Z* = Propriété ou action de X cause que Z apprenne I.a Y.

Régime

1 = X	2 = Y	3 = Z
1. N	1. N 2. *que* PROP obligatoire	1. *à* N

$C_1 + C_2$: *L'histoire enseigne le déclin de toutes les civilisations que toutes les civilisations sont appelées à disparaître*

$C_1 + C_2 + C_3$: *L'expérience nous enseigne que la guerre n'a jamais résolu les problèmes*

Fonctions lexicales

Syn : apprendre I.b

S_2 : enseignement 3

Exemples

Leur attitude au sage enseigne / Qu'il faut en ce monde qu'il craigne / Le tumulte et le mouvement [Ch. Baudelaire]. L'exemple de mes parents m'a enseigné le courage bien plus que ne l'auraient fait des discours.

3b. *X enseigne Y à Z* = Propriété ou action de X cause que Z apprenne II.a Y.

Régime

1 = X	2 = Y	3 = Z
1. N	1. N 2. *à* V_{inf} obligatoire	1. *à* N

$C_1 + C_2$: *La servitude enseigne la ruse à ruser*

$C_1 + C_2 + C_3$: *Mon père m'a enseigné la prudence par son exemple*

Fonctions lexicales

Syn : apprendre II.b

Syn : inculquer, éduquer

Exemples

Un bon maître a ce souci constant : enseigner à se passer de lui [A. Gide]. Le feu du soleil […] enseignait la patience [J. Kessel]. C'est sa mère qui lui a enseigné la coquetterie.

ÉTONNANT, adj.

1. X qui étonne 1 [

2. … qui frappe par son caractère remarquable [*un film étonnant,*]

1. [*X*] *étonnant* = X qui étonne 1 [= A_1 (*étonner 1*)].

Fonctions lexicales

Syn : surprenant

Magn : très, fort, bien // stupéfiant, ahurissant, ébahissant, **fam** épostouflant

2. [*X*] *étonnant* = [X] qui frappe par son caractère remarquable [comme si X était étonnant 1].

Fonctions lexicales

Syn : remarquable 2

Anti : ordinaire

Magn : // formidable, extraordinaire 2, merveilleux

Exemples

Un film étonnant, une femme étonnante

ÉTONNÉ, adj.

1. [X] qui s'étonne de Y [*Étonné devant ce spectacle inattendu, Jean s'est tû*]

2. … tel que Z manifeste l'étonnement de X [*des yeux étonnés]*

1. [*X*] *étonné de Y* = [X] qui s'étonne de Y [= A_1 (*s'étonner*)].

Régime

2 = Y
1. *devant* N
2. *de* V_{inf}

C_2 : *Étonné devant ce spectacle inattendu de voir son ami dans un tel état Jean s'est tû*

Fonctions lexicales

Syn : surpris 1

Magn : bien, fort, très // stupéfait, ahuri, ébahi, sidéré, abasourdi, bouche bée, **fam** soufflé, **fam** baba

Exemples

Une fille étonnée. Il lui écrivit une lettre et fut très étonné de recevoir une réponse. Tout le monde a été étonné de son comportement. Fort étonné devant la tournure des événements, Pierre se demandait quoi faire.

2. [*Z de X*] *étonné* = [Z de X] tel que Z manifeste l'étonnement de X [= A_2Manif(*s'étonner*)].

Fonctions lexicales

Syn : surpris 2

Exemples

Les yeux étonnés.