Peut-on coder un dictionnaire avec des Ψ-termes ?

Gilles Sérasset

▶ **To cite this version:**

Gilles Sérasset. Peut-on coder un dictionnaire avec des Ψ-termes ?. Turjuman, 1994, 3 (1), pp.41-56. hal-00966405

HAL Id: hal-00966405
https://hal.science/hal-00966405v1

Submitted on 26 Mar 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Peut-on coder un dictionnaire avec des Ψ-termes ?

Gilles Sérasset

GETA-IMAG, UJF & CNRS

BP 53
F-38041 Grenoble Cedex 9
Tél : 76 51 48 17
Fax : 76 51 44 05
e-mail : Gilles.Serasset@imag.fr

Extended Abstract

The development of a multilingual lexical database poses the problem of the choice of a formalism to represent informations. Unification-based formalisms are increasingly advocated as offering all the necessary qualities. In order to test this claim, we have chosen one of the most powerful unification based formalism, that of Ψ-terms (also known as "Typed Feature Structures"). We have used it to represent a dictionary article.

The dictionary article we have incoded is complex enough to make this test significant. Moreover, we wanted it to be linguistically meaningful, and as complete as possible. That is why we have chosen to code an article extracted from the "Dictionnaire explicatif et combinatoire du français contemporain" [Mel'čuk 1984, Mel'čuk 1988].

The paper begins with a brief definition of lexical elements, followed by a concise introduction to Ψ-terms formalism.

We then show how to code lexical informations, considering three levels: morphological, syntactic (with, in particular, derivations and constructions) and semantic (definitions and Mel'čuk's lexico-semantic functions).

This exercise allows us to highlight some problems inherent to the Ψ-terms formalism. In particular, we stress the difficulty to code characters strings (a definition for example), and the impossibility to express contraints such as incompatibility between features.

To conclude, we suggest a distinction between two representational levels. The first level should be readable by lexicographers and users. The second, more abstract and coded using an appropriate formalism, should be more oriented towards computer processing.

Keywords

Electronic dictionaries, lexico-semantic functions, Ψ-terms.

Résumé

Nous désirons étudier l'utilisation éventuelle du formalisme des Ψ-termes (les "structures de traits typés") pour représenter un dictionnaire monolingue. Pour cela, nous traitons un article extrait du "Dictionnaire explicatif et combinatoire du français contemporain" [Mel'čuk 1984, Mel'čuk 1988]. Cela nous permet de préciser quelques avantages et inconvénients de ce formalisme.

Mots clés

Dictionnaires électroniques, Fonctions lexico-sémantiques, Ψ-termes.

Introduction

La longue expérience du GETA en matière de Traduction Assistée par Ordinateur (TAO) a montré la nécessité de disposer d'une source d'informations lexicales unique et générale. Nous imaginons une telle source comme une base lexicale contenant l'ensemble des données disponibles. Chacune des applications linguistiques en utilise une partie. Le développement d'une telle base de données pose le problème du choix d'un formalisme de représentation des informations.

Depuis un certain temps, un attrait grandissant est apparu pour les formalismes fondés sur l'unification. Nous avons choisi l'un d'eux, les Ψ-termes (ou Typed Feature Structures), et nous l'avons utilisé pour représenter un article de dictionnaire, dans le but d'évaluer ce type de formalisme. Dans cet article, nous visons la représentation des informations lexicales. La priorité est donnée à la lisibilité et non aux traitements.

Après quelques définitions concernant les éléments du lexique, nous introduisons très brièvement le formalisme des Ψ-termes. Nous présentons ensuite les trois niveaux de représentation (morphologique, syntaxique et sémantique). Pour chacun de ces niveaux, nous donnons une représentation générale et un exemple.

I. Éléments du lexique

Pour toute application du traitement automatique des langues naturelles se pose immédiatement le problème de notation et de représentation des unités du lexique ou des mots. Le terme *mot* étant ambigu, nous préférons utiliser les termes *occurrence* et *lemme*.

Cette partie a pour but de définir, de façon volontairement succincte les éléments du lexique que nous manipulerons. Pour plus de détails, le lecteur pourra se reporter à [Nédobejkine 1991].

Occurrence, base

Une *occurrence* est toute forme apparaissant dans un texte (ex : *mettra*) entre deux délimiteurs (blanc, ponctuation).

Une chaîne de caractères commune à un certain nombre d'occurrences d'un lemme est une *base* (ex : *mir-* pour *mirer*, ou *mett-*, *met-*, *mi-* pour *mettre*).

Lemme, lexème

Un *lemme* (ou *lexème*) est une famille d'occurrences générée par le même paradigme à partir d'une base généralement commune et appartenant à la même classe syntaxique. Un lemme est habituellement noté par une occurrence (ou forme) canonique (verbe à l'infinitif, …).

Dérivation

Un lemme est dit *dérivé* s'il est formé à partir d'un autre, le plus souvent au moyen d'une affixation, et si son sens peut en être déduit grâce à un schéma formel (ex : *Français* -> habitant, citoyen de la *France*).

Unité lexicale, sémie

Une *unité lexicale* est un ensemble de lemmes dérivés d'un même lemme origine conformément aux schémas dérivationnels productifs retenus. On la note par son lemme origine. (ex : *appareiller -> appareiller*, *appareillage*, *appareillement*, *appareillade*, *appareilleur*).

On appelle *sémie* un sous-ensemble de lemmes d'une unité lexicale qui recouvrent les mêmes sens. Dans l'exemple précédent, l'unité lexicale *appareiller* compte 5 sémies.

Valence syntaxique

Les *valences syntaxiques* sont les contraintes syntaxiques sur les arguments des formes d'un lemme donné, pour chacune de ses constructions.

Trait sémantique

Un *trait sémantique* est une unité de classification du vocabulaire adoptée par l'auteur d'une théorie linguistique (*animé*, *inerte*, …). Les traits sémantiques permettent de cerner les sens des termes, voire à les raffiner pour des buts de traduction.

Il existe des traits sémantiques universels, mais chaque langue utilise à l'intérieur de son propre système certains traits sémantiques particuliers.

Valence sémantique

Les *valences sémantiques* d'un lemme (le plus souvent prédicatif) sont les contraintes sur les traits sémantiques des arguments, pour chacune des constructions "logiques". Une construction logique correspond souvent à plusieurs constructions syntaxiques synonymes (ex : actif, passif).

Sens ou Acception

Le sens peut être défini comme le sous-ensemble d'une sémie munie d'un trait sémantique, d'une construction syntaxique et d'une valence sémantique pour chacun de ses arguments

Usage

L'usage d'un lemme peut être défini comme son domaine d'emploi.

Locution et Mot composé

C'est une association de plusieurs lemmes pour exprimer une notion élémentaire.

II. Les Ψ-termes

Nous introduisons de façon succincte le formalisme des Ψ-termes. Pour plus d'informations, le lecteur peut se reporter à [Aït-Kaci 1986, Aït-Kaci et Lincoln 1988, Aït-Kaci et Nasr 1986, Genthial 1991a, Genthial 1991b].

De manière informelle, on peut voir un Ψ-terme comme la définition d'un type complexe (tel que "record" en Pascal) ou comme la dénotation d'un ensemble. Un Ψ-terme consiste en :
- une *racine*, qui dénote une classe d'objets,
- des *attributs* associés à d'autres Ψ-termes (ses sous-termes directs),
- une relation de *coréférence* entre attributs et sous Ψ-termes. Cette relation permet de noter qu'une même valeur (sous-structure) est partagée par différents attributs.

Ainsi, si l'on fait une analogie entre le formalisme des Ψ-termes et un mécanisme de définition de types comme les "record" en Pascal, la racine représente le nom du type construit et les attributs nomment les champs de ce "record". Par contre, la relation de coréférence n'a pas d'équivalent en Pascal ou en C. Introduire la coréférence permet de passer des structures arborescentes décrites par les types classiques à des structures de graphes orientés.

À l'aide de la définition d'un treillis de types, on obtient un ordre partiel sur les Ψ-termes qui permet de généraliser convenablement l'opération d'unification. On généralise ainsi l'unification classique sur les structures de traits non typés et sur les termes (du calcul des prédicats du premier ordre).

Nous utiliserons, en plus du formalisme initial défini par [Aït-Kaci 1986], la notion de *disjonction* (une disjonction de Ψ-termes est aussi un Ψ-terme).

Dans la suite de l'article, nous représentons les Ψ-termes sous forme textuelle (et non graphique), avec les conventions suivantes :
- les types sont écrits en minuscules,
- les traits sont écrits en majuscules,
- les étiquettes (utilisées pour marquer la coréférence) sont précédées d'un "@",
- la disjonction est présentée entre accolades ("{" et "}"),
- le séparateur entre trait et valeur est : "=>"

III. Codage

L'article du "Dictionnaire explicatif et combinatoire du français contemporain" de Mel'čuk, que nous avons pris comme exemple est présenté en annexe.

Informations morphologiques

Une entrée de dictionnaire est un "identificateur d'article", formé de la forme canonique du lemme représenté, sa classe et son numéro d'homographe. On associe à une entrée les informations morphologiques du lemme représenté (base(s), flexion, genre pour les noms,…).

Le Ψ-terme utilisé pour représenter un article à ce niveau est donné figure 1.

```
lemme ( ENTREE => idf_article ( FORM_CANON    => chaîne ;
                                CLASSE        => classe ;
                                NO_HOMOGRAPHE => numéro )
       MORPHO     => infos_morph ( MODELE =>
                                      { flex ( BASE    => chaîne ;
                                               FLEXION => code_flexionnel ),
                                        flex ( BASE    => chaîne ;
                                               FLEXION => code_flexionnel ),
                                        ... };
                                   ... ) ).
```

Figure 1 : vue générale d'une entrée de dictionnaire.

Ainsi, si on développe l'exemple du nom *assistance*, on obtient le Ψ-termes de la figure 2

```
lemme (       ENTREE => idf_article ( FORM_CANON    => assistance ;
                                      CLASSE        => nom )
              MORPHO     => infos_morph (
                                       MODELE =>
                                            flex (  BASE    => assistance ;
                                                    FLEXION => pas_de_pluriel );
                                       GENRE  => féminin
                                      )
    ).
```

Figure 2 : l'entrée de dictionnaire pour le nom *assistance* (selon Mel'čuk).

On remarque dans cet exemple que le numéro d'homographe n'a pas été précisé. En effet, *assistance* n'a pas d'homographe. Par contre, on l'aurait précisé pour d'autres articles, par exemple pour le verbe *ressortir* (*l'homme ressortait par la porte opposée, ce procès ressortissait à la Cour d'appel*) ou le nom *mode* (*le mode de cuisson, la mode d'hiver*).

La valeur de l'attribut `FORME_CANON` est, dans notre exemple, `assistance`. Or, la valeur d'un attribut doit être un Ψ-terme. Aussi, pour rester cohérent avec le formalisme de base, `assistance` doit être considéré comme un Ψ-terme simple (un type sans attributs). Cela implique de gérer autant de Ψ-termes simples qu'il y a d'entrées dans le dictionnaire. Il faut donc définir chacune de ces entrées dans le treillis des types, ou de convenir que tout type non déclaré dans ce treillis est une "feuille" (de prédécesseur unique, le "bas" du treillis).

L'article de dictionnaire est ensuite raffiné en acceptions auxquelles sont rattachées les informations syntaxiques et sémantiques, avec parfois des contraintes ou informations morphologiques spécifiques. De manière générale, on repère une acception de manière univoque en donnant l'identificateur d'article et le numéro d'acception. Ce raffinement est représenté par la figure 3.

```
lemme (       ENTREE => ...;
              MORPHO     => ...;
              ACCEPTIONS => { acception ( ENTREE => idf_article ( ... ) ;
                                          NUMERO => numéro_acception1    ),
                              acception ( ENTREE => idf_article ( ... ) ;
                                          NUMERO => numéro_acception2    ),
                              ...
                            }
    ).
```

Figure 3 : raffinement de l'entrée en acceptions.

Ainsi, le raffinement de l'exemple du nom *assistance* est donné figure 4.

```
lemme (       ENTREE => @a : idf_article ( FORM_CANON    => assistance ;
                                           CLASSE        => nom )
              MORPHO     => ...;
              ACCEPTIONS => {
                 acception ( ENTREE => @a ;
                             NUMERO => 1 ),
                 acception ( ENTREE => @a ;
                             NUMERO => 2.1 ),
                 acception ( ENTREE => @a ;
                             NUMERO => 2.2 ),
                 acception ( ENTREE => @a ;
                             NUMERO => 2.3 )
                           } ).
```

Figure 4 : raffinement de l'entrée *assistance*.

Informations syntaxiques

Les informations syntaxiques (dérivations, constructions) sont rattachées aux acceptions. Nous utilisons ici les notations adoptées par Mel'čuk dans [Mel'čuk 1984, Mel'čuk 1988].

Les informations syntaxiques sont séparées en deux parties distinctes :
- dérivations,
- constructions.

Les informations dérivationnelles sont séparées en deux parties :
- unités lexicales sources,
- unités lexicales cibles.

Pour chaque source et chaque cible, on a le type de dérivation associé.

Les constructions possibles sont données dans un Ψ-terme disjonctif, sous forme d'une liste de constructions pour chacun des actants. Comme toutes les constructions ne sont pas compatibles, on donne la liste des restrictions, en utilisant la coréférence.

Ainsi, la représentation des informations syntaxiques d'une acception pourrait être celle donnée figure 5.

```
acception ( ENTREE  => idf_article ( ... ) ;
            NUMERO  => numéro_acception ;
            SYNTAXE => infos_synt (
               DERIVATION    => dérivations (
                           VIENT_DE => { source (
                                          UL => acception (
                                               ENTREE => idf_article ;
                                               NUMERO => numéro_acception
                                                         ) ;
                                          PAR => type_de_dérivation
                                             ) ; ... } ;
                           VA_VERS  => { cible (
                                          UL => acception (
                                               ENTREE => idf_article ;
                                               NUMERO => numéro_acception
                                                         ) ;
                                          PAR => type_de_dérivation
                                            ) ; ... } ) ;
               CONSTRUCTIONS => régime (
                                    ACT1 => { @Ci : construction , ... } ;
                                    ACT2 => { @Cj : construction , ... } ;
                                    ... ;
                                    RESTRICTIONS => { impossible ( 1 => @Ci ;
                                                                   2 => @Cj ) ; ... }
                                        ) ) ).
```

Figure 5 : informations syntaxiques d'une acception.

A la place de la coréférence, on aimerait mieux disposer d'une notation plus fonctionnelle (*impossible (construction1,construction2)*) ou d'une notation utilisant des graphes (qui représenteraient les combinaisons valides de constructions).

Si l'on développe l'une des acceptions de notre exemple, on obtient le Ψ-terme de la figure 6.

```
acception ( ENTREE => idf_article ( FORM_CANON    => assistance ;
                                    CLASSE        => nom ) ;
            NUMERO => 2.1 ),
            SYNTAXE => infos_synt (
               DERIVATION     => dérivations (
                           VIENT_DE =>
                                 source (
                                    UL => @assister : acception (
                                         ENTREE  => idf_article        (
                                                 FORM_CANON => assister ;
                                                 CLASSE     => verbe   ) ;
                                         NUMERO => 2.1          ) ;
                                    PAR => drv_VNact ) );
                CONSTRUCTIONS => régime (
                                    ACT1 => { de_N ,
                                              A_poss } ;
                                    ACT2 => à_N ;
                                    ACT3 => { dans_N ,
                                              pour_N ,
                                              pour_V_inf }
                                        ) ) ).
```

Figure 6 : informations syntaxique de l'acception II.1 du nom *assistance*.

Informations lexicales et sémantiques

Les informations lexicales et sémantiques sont, elles aussi, rattachées aux acceptions. Elles comprennent :
- la définition,
- les fonctions lexico-sémantiques de Mel'čuk (cf. [Mel'čuk 1984, Mel'čuk 1988]),

On note que les informations de définitions dépendent de la forme de celle-ci. On peut avoir une courte phrase (*Ensemble de personnes qui assistent à un événement, ...*) ou une fonction lexicale d'une autre acception

(*Mult(assistant I), S_0(assister II.1), ...*), voire les deux.

```
acception ( ENTREE     => idf_article ( ... ) ;
            NUMERO     => numéro_acception ;
            SYNTAXE    => ... ;
            SEMANTIQUE => infos_sem (
                            DEFINITION => définition (...) ;
                            FL => fonctions (
                                     SYN => acception (
                                              ENTREE => idf_article ( ... ) ;
                                              NUMERO => numéro_acception
                                                      ) ;
                                     SYN_INTER => acception (...) ;
                                     ANTI => acception (...) ;
                                     ...    )
                                   )
          ).
```

Figure 7 : informations syntaxiques d'une acception.

```
acception ( ENTREE => idf_article ( FORM_CANON    => assistance ;
                                    CLASSE        => nom ) ;
            NUMERO     => 2.1 ;
            SYNTAXE    => ... ;
            SEMANTIQUE => infos_sem (
                 DEFINITION => définition (
                          ARG_S0 => acception (
                               ENTREE => idf_article        (
                                     FORM_CANON    => assister ;
                                     CLASSE        => verbe ) ;
                               NUMERO => 2.1
                                                ) ;
                                          ) ;

                 FL => fonctions (
                       SYN_INTER => acception (
                            ENTREE => idf_article ( FORM_CANON => aide ;
                                                    CLASSE     => nom ) ;
                            NUMERO => 1b  ) ;
                       S1        => acception (
                            ENTREE => idf_article ( FORM_CANON => assistant ;
                                                    CLASSE     => nom ) ;
                            NUMERO => 2.1 ) ;
                       S1_ETROIT => acception (
                            ENTREE => idf_article ( FORM_CANON => assistant ;
                                                    CLASSE     => nom ) ;
                            NUMERO => 2.2 ) ;
                       OPER1     => acception (
                            ENTREE => idf_article ( FORM_CANON => apporter ;
                                                    CLASSE     => verbe ) ;
                            SYNTAXE => infos_synt (
                                 CONSTRUCTION => régime (
                                       ACT2 => A_poss;
                                       ACT3 => a_N      ) ;
                                                  ) ;
                                            ) ;
                       OPER2     => acception (
                            ENTREE => idf_article ( FORM_CANON => recevoir ;
                                                    CLASSE     => verbe ) ;
                            SYNTAXE => infos_synt (
                                 CONSTRUCTION => régime (ACT2 => art ) ;
                                                  ) ;
                                            ) ;
                                       )
          ).
```

Figure 8 : les informations syntaxiques de l'acception II.1 du nom *assistance*.

Le formalisme des Ψ-termes est mal adapté dans le cas d'une chaîne de caractères. En effet, un trait ne peut avoir pour valeur qu'un Ψ-terme; et une chaîne de caractères n'est pas un Ψ-terme. On peut, bien sûr, la coder (une liste de caractères), mais c'est assez lourd.

Un autre problème vient du fait que l'on voudrait pouvoir donner une définition non-ambiguë (*Ensemble I de personnes I.2 qui assistent I à un événement II, ...*). On ne peut représenter cela de manière simple et naturelle en utilisant le formalisme des Ψ-termes.

Enfin, les fonctions lexico-sémantiques définies dans [Mel'čuk 1984, Mel'čuk 1988] ne sont pas des fonctions simples. On peut avoir des fonctions composées (*Adv_2Real_2(assistance II.1) = avec [l'~]*), des fonctions indicées

(*Oper$_1$(assistance II.1) = apporter [A$_{poss}$ ~ à N], Oper$_2$(assistance II.1) = recevoir [ART ~]*) et des fonctions demandant un certain contexte pour pouvoir s'appliquer (*Tenter de Caus$_2$Func$_0$(assistance II.1) = demander [l'~ de N]*). Le formalisme des Ψ-termes n'est pas assez souple pour permettre une représentation aisée de ce genre d'information.

Ainsi, la représentation des informations sémantiques d'une acception est donnée dans la figure 7.

Ainsi, en développant notre exemple, on aurait le Ψ-terme de la figure 8.

Conclusion

Lors du codage d'un article de dictionnaire, au moyen du formalisme des Ψ-termes, étendu par la disjonction, nous avons rencontré de nombreux problèmes. Le formalisme des Ψ-termes semble pourtant être adapté pour des structures relativement simples et les mécanismes qu'il offre sont intéressants (nommage des attributs, typage, héritage, coréférence). Mais, il pose des difficultés dans des cas plus complexes quoique fréquents en pratique :

- composition de fonctions lexicales,
- représentation des chaînes de caractères,
- expression de contraintes, avec par exemple les cas où :
  - un attribut est obligatoire,
  - deux attributs sont incompatibles,
  - la présence (ou l'absence) d'un (ou de plusieurs) attribut(s) entraîne la présence (ou l'absence) d'un (ou de plusieurs) autre(s) attribut(s).

La sémantique même des Ψ-termes pose un problème théorique lorsqu'on veut l'appliquer à la définition de structures de dictionnaire. Lorsqu'un linguiste défini des structures de dictionnaires, il déclare quels sont les *attributs valides* dans un article. Lorsqu'un linguiste défini un Ψ-terme, il déclare quels sont les *attributs connus* de ce terme. Tout autre attribut est valide (puisqu'unifiable avec le terme défini). La sémantique même des Ψ-termes (telle qu'elle est défini par l'uniique opération d'unification) est incompatible avec l'action de définition de structures d'un dictionnaire.

Tel qu'il est implémenté dans TFS [Emele et al. 1990], le formalisme des Ψ-termes nous paraît donc insuffisant comme support d'une application réaliste. Que faire pour surmonter ces inconvénients ? D'abord, rien ne nous empêche de nous inspirer de ce formalisme à un niveau purement syntaxique (coréférence, héritage), tout en abandonnant l'idée d'utiliser l'unification comme seule opération. En effet, elle serait mal définie et insuffisante.

Ensuite, on peut étendre le formalisme. En travaillant sur le DEC et sur d'autres exemples, nous avons souvent été tenté de donner à certains traits des valeurs qui ne sont pas des Ψ-termes. Cela nous conduit à envisager un formalisme plus général, analogue à celui proposé par [Aït-Kaci et Lincoln 1988] dans le système LIFE.

Une telle extension résoudrait les problèmes de fond. Mais le lecteur aura certainement remarqué que les articles de complexité moyenne sont tout à fait illisibles. Or, il faudra bien présenter des formes lisibles, manipulables de façon conviviale et ergonomique. Nous suggérons donc de distinguer deux niveaux d'écriture des articles :

- au premier niveau, on trouverait une représentation fondée sur SGML, ce qui permettrait, à l'aide d'éditeurs structurés, la saisie, la visualisation et la modification dans des présentations adéquates (champs, typographie riche, …). Cette représentation comprendrait la définition de la structure des articles et des attributs utilisés, ainsi que des contraintes absolues d'intégrité (des articles) et de cohérence (d'une ou de plusieurs bases).
- au deuxième niveau, on trouverait la structure interne, ou abstraite, associée à l'article, qui pourrait être directement lue et traitée par LISP. Rien n'interdirait qu'elle soit un codage d'un Ψ-terme (ou mieux, d'une λπΨ expression en LIFE), d'un terme Prolog, d'un graphe conceptuel, etc. Cette méthode garantirait que l'on puisse programmer toutes les opérations désirables sur une base lexicale, sans devoir se limiter à la seule unification.

Remerciements

Une partie de ce travail a été réalisé dans le cadre du projet ESPRIT Multilex (n° 5403), je veux remercier les participants de ce projet, qui m'ont aidés en me fournissants conseils et documents. Je tiens à remercier Christian Boitet qui a aimablement corrigé cet article. Je suis bien sûr seul responsable de toutes les éventuelles erreurs qui subsisteraient.

Bibliographie

Aït-Kaci H. (1986). *An Algebraïc Approach to the Effective Resolution of Type Equations,* Theoretical Computer Science**, vol. 45** : pp. 293-351.

Aït-Kaci H. et Lincoln P. (1988). *LIFE : a Natural Language for Natural Language*, MCC, TR ACA-ST-074-88, février 1988 : 27 p.

Aït-Kaci H. et Nasr R. (1986). *LOGIN : a Logic Programming Language with Built-in Inheritance,* JLP**, vol. 3** : pp. 185-215.

Emele M. et al. (1990). *Organising linguistic knowledge for multilingual generation*, COLING 90, Helsinki**, vol. 3/3** : pp. 102-107.

Genthial D. (1991a). *Contribution à la construction d'un système robuste d'analyse du français*, Thèse, Université Joseph Fourier : 235 p.

Genthial D. (1991b). *Représentation des données lexicales : vers des traitements tolérants*, Deuxièmes journées nationales du GRECO-PRC Communication Homme-Machine, Toulouse, 29-30 janvier 1991 : pp. 69-76.

Mel'čuk I. (1984). *DEC : Dictionnaire explicatif et combinatoire du français contemporain*, Montréal(Quebec), Canada, Presses de l'université de Montréal.

Mel'čuk I. (1988). *DEC : Dictionnaire explicatif et combinatoire du français contemporain*, Montréal(Quebec), Canada, Presses de l'université de Montréal.

Nédobejkine N. (1991). *Représentation du lexique dans la théorie linguistique du GETA*, Deuxièmes journées nationales du GRECO-PRC Communication Homme Machine, Toulouse.

Annexe

Nous donnons en annexe, l'article de dictionnaire que nous avons choisi de coder.

ASSISTANCE, nom, fém, pas de pl.

I. Ensemble de personnes qui assistent I… [Sa conférence a charmé l'assistance]

II.1. S_0(*assister II.1*) [l'assistance d'un technicien pour évaluer le travail des employés]

2. S_0(*assister II.2*) [l'assistance financière aux sinistrés]

3. Organisme ou programme visant l'aide 1b à Y… [l'assistance sociale]

I. *Assistance* [*à Y*] = Ensemble de personnes qui assistent I à un événement Y [= Mult(*assistant I*)].

Régime

2 = Y

—

Fonctions lexicales

$Syn_{\cap}$	: auditoire, public
$Magn_2^{quant}$	: nombreuse
$AntiMagn_2^{quant}$	: peu nombreuse, clairsemée, maigre
Pos_2	: enthousiaste
$AntiPos_2$	: hostile
$IncepFunc_0$	: se rassembler, se réunir
$FinFunc_0$	: se disperser
$Caus_2Func_0$	: attirer, rassembler, réunir

Exemples

L'assistance tout entière était secrètement offensée comme d'une marque de mépris [A. France]. Le Cardinal entra, salua l'assistance avec ce sourire héréditaire des grands pour le peuple… [V. Hugo]. L'assistance applaudit à son discours. L'assistance semblait captivée par le conférencier.

II.1. *Assistance de X [à Y] pour U* = S_0(*assister II.1*).

Régime

1 = X	2 = Y	3 = U
1. *de* N 2. A_{poss}	——	1. *dans* N 2. *pour* N 3. *pour* V_{inf}

C_1 : *l'assistance d'une infirmière <d'un technicien>, son assistance*
C_3 : *l'assistance dans le travail <pour l'opération, pour évaluer le travail des employés>*
$C_1 + C_3$: *l'assistance d'une infirmière <son assistance> dans ce travail <pour l'opération>, l'assistance d'un technicien <son assistance> pour évaluer le travail des employés*

Fonctions lexicales

$Syn_{\cap}$	:	aide 1b
S_1	:	assistant II.1
$S_{1\subset}$	:	assistant II.2
$Oper_1$	:	apporter [Aposs ~ à N]
$Oper_2$	:	recevoir [ART ~]
tenter de $Caus_2Func_0$	:	demander [l' ~ de N]
Adv_2Real_2	:	avec [l'~] | C1 ≠ Λ

Exemples

L'assistance de tous ces techniciens est nécessaire pour le tournage d'un film. L'infirmière apporte son assistance au chirurgien. Ce genre d'opération requiert l'assistance d'une infirmière expérimentée. Le chirurgien a opéré le blessé avec l'assistance de son aide major.

II.2. *Assistance W de X à Y dans U* = (α) L'emploi par X de ses ressources W dans le but de causer que W aident 2a Y dans U ou (β) ces ressources W [= S_0 ou S_4(*assister II.2*)].

Régime

1 = X	2 = Y	3 = U	4 = W
1. *de* N 2. A_{poss}	1. *à* N	1. *dans* N	1. A

C_1 : *l'assistance de Paul, son assistance*
C_2 : *l'assistance à une famille éprouvée <à un ami>*
C_3 : *l'assistance dans une circonstance douloureuse <dans un grand malheur>*
C_4 : *l'assistance morale <financière>*
C_1+C_2+C_3+C_4 : *l'assistance morale de Paul à sa sœur dans cette dure épreuve*

Fonctions lexicales

Syn	:	aide 2
$Syn_{\cap}$	:	secours, réconfort, appui, soutien
$Anti_{\cap}$	:	abandon
V_0	:	assister II.2
$Oper_1$	:	apporter [ART ~ à N], prêter [ART/Ø ~ à N]| plutôt A.(α); donner, accorder [ART ~ à

N]| plutôt A.(β)

$Oper_2$	:	recevoir [ART ~]
tenter de $Caus_2Func_0$	:	demander [~ à N] [Il demanda assistance à son frère]
Adv_2Real_2	:	avec [l' ~]| C1 ≠ Λ

Exemples

… que le mineur émancipé pourra accepter avec l'assistance de son curateur [Code Civil]. Les époux se doivent mutuellement fidélité, secours, assistance [Code Civil]. Lorsque le gouvernement s'est assuré de la probité d'un étranger, il doit lui accorder protection et assistance, et se regarder comme remplaçant à son égard son gouvernement naturel, et même sa famille [Bonald, Législation primitive…]. Son assistance morale lui a été d'un grand secours dans une telle situation. Elle le suppliait de lui accorder son assistance spirituelle. Services canadiens d'assistance aux immigrants juifs.

II.3. *Assistance D* [*à Y*] = Organisme ou programme visant l'aide 1b à Y concernant D ou l'assistance II.2 à Y dans une situation difficile concernant D [= $S_{1\cap}$(*aider 1b*, *assister II.2*)].

Régime

2 = Y	3 = D
_____	1. -N 2. A obligatoire

1)$C_{3.1}$	:	*N = annuaire*
$C_{3.1}$	:	*l'assistance-annuaire*
$C_{3.2}$	:	*l'assistance sociale <éducative, judiciaire, médicale, psychiatrique, maritime>*

Fonctions lexicales

$Syn_{\cap}$	:	aide 3; service
tenter de $Caus_2Func_0$	:	faire appel, avoir recours [à ART ~]
$Real_2$	:	bénéficier [de ART ~]

Exemples

L'assistance sociale vient en aide aux personnes incapables de subvenir à leur besoins. Cette famille est sur l'assistance sociale.