# Chapitre 7

*"Cet écran où mon regard bute, tout en persistant à y voir de l'air, ne serait-ce pas plutôt l'enceinte d'une densité de plombagine ? Pour tirer cette question au clair j'aurais besoin d'un bâton ainsi que des moyens de m'en servir (...). Mais l'époque des bâtons est révolue."*
"L'innommable", Samuel Beckett, Ed. de Minuit.

# Des arbres aux prototypes : vers une classification dynamique

## 7.1. Acquisition de connaissances à partir d'un corpus

*"Le point de départ de toute description est la reconstruction des classes sémantiques au cours de la lecture du corpus."*
(Rastier 1994)

Nous nous plaçons ici dans le cadre de travail mis en oeuvre par (Habert & Nazarenko 1996). Dans ce cadre, il s'agit de simplifier des groupes nominaux complexes fournis par un outil d'extraction terminologique, LEXTER (Bourigault 1993), pour obtenir les arbres élémentaires montrant les relations syntaxiques fondamentales d'un corpus. Ce processus d'acquisition de connaissances est testé sur un corpus médical. Les savoirs extraits sont des arbres d'analyse fournis par un outil d'extraction terminologique (Lexter[1], Bourigault 1993), ces arbres étant ensuite simplifiés dans le but de déterminer les arbres de dépendance élémentaire qu'il est possible d'associer aux entrées lexicales (Cyclade[2], Habert & Nazarenko 1996). Ce travail vise à établir les fonctionnements d'une entrée lexicale dans le corpus, et en particulier à mettre en avant ce que cette entrée peut tisser comme liens avec les différentes entrées lexicales présentes dans le corpus. Il vise aussi à établir des classes de comportements syntactico-sémantiques que l'analyse du corpus peut révéler. Notre travail s'inscrit dans une reprise de l'approche harrissienne (Sager & al. 1987, Habert & Nazarenko 1996) : en fait la détermination de classes d'opérateurs et d'opérandes par le fonctionnement linguistique. Notre travail s'articule autour de trois éléments : un problème, un corpus et un outil informatique.

### 7.1.1. Un Problème

[1] Lexter est un outil d'extraction terminologique conçu et réalisé dans un environnement industriel: la Direction des Etudes et Recherches de EDF, pour aider à la mise au point de terminologie. Il prend en entrée des textes arbitrairement longs et produit des arbres d'analyses de séquences nominales en décomposant ces séquences en Tête et Expansion, de manière récursive.

[2] Le programme Cyclade (Habert & al. 1995) a pour tâche de simplifier les arbres d'analyse fournis par un analyse, ici par Lexter, et de mettre en évidence les arbres qui présentent des relations de dépendance entre mots pleins (vs. mots vides: prépositions, déterminants, conjonctions...).





Le problème à résoudre consiste à définir les fonctionnements lexicaux qu'il est possible d'attacher aux mots dans le cadre d'un analyseur à mots. Il s'agit d'une part de déterminer ces comportements lexicaux et d'autre part de les représenter. Notre démarche vise à mettre en lumière dans des domaines restreints les comportements des mots ainsi que les corrélations multiples qui existent entre les mots dans un flot continu de discours. Elle consiste en quelque sorte à "faire-émerger" ces comportements des mots puis à les représenter ou affiner les représentations existantes. L'hypothèse suivie ici étant qu'il y a peu de sens à vouloir faire de l'acquisition sémantique en dehors d'un sous-langage.

### 7.1.2. Un Corpus

La langue présente trop de variations pour pouvoir être appréhendée globalement. Ces variations se retrouvent le plus souvent au niveau des mots où elles sont plus directement accessibles. Notre travail de représentation et de classement s'appuie sur une recherche initiale au niveau des mots des régularités et des redondances d'utilisation dans des corpus donnés. Les processus de représentation et de classement commencent leur travail à partir de ces informations en sachant que des ajustements seront toujours possibles sur les résultats construits. Notre tâche vise donc à s'élever du particulier au général et à procéder à des regroupements pour y découvrir les faces cachées derrière les détails. Puisque la hiérarchie est un artifice, utile d'ailleurs pour classer les choses, nous ne nous imposons pas un classement prédéfini des savoirs à représenter, mais nous choisissons une démarche qui tente de reconstruire ce classement en tenant compte des informations délivrées par le travail d'extraction de savoirs à partir de corpus. La phase de représentation utilise des savoirs extraits à partir du corpus constitué dans le cadre du projet MENELAS (Zweigenbaum 1994, Zweigenbaum & al. 1995) pour la compréhension de textes médicaux. Ce corpus est utilisé par le Groupe de Travail Terminologique et Intelligence Artificielle (PRC-GDR Intelligence Artificielle, CNRS). L'unité thématique de ce corpus a trait aux maladies coronariennes. L'apprentissage sur corpus réalisé vise à repérer les arbres de dépendances élémentaires entre mots (relations opérateur-opérande) et les contraintes sur les combinaisons de ces arbres.

### 7.1.3. Un Outil Informatique

On a déjà vu que la PàP peut être considérée comme un outil de représentation privilégiée pour la représentation de domaines de savoirs pour lesquels on ne dispose pas de catégories bien précises. Si les savoirs à représenter ne sont pas connus de manière définitive, il est possible de commencer le processus de représentation en utilisant les savoirs déjà recensés puis d'affiner dynamiquement les objets construits dès que de nouveaux savoirs sont disponibles. A la limite, cette démarche de représentation n'a pas à se préoccuper de la forme terminale des objets à construire. Elle peut démarrer à partir d'une structure vide qui sera mise à jour dès que des informations sont disponibles. La représentation avec des prototypes n'est pas un processus qui capte globalement les savoirs sur un domaine donné et qui construit une structure définitive pour représenter ces savoirs. Il s'agit au contraire de mettre à jour progressivement tous les savoirs pertinents pour affiner le domaine à représenter. Cette redéfinition continuelle des objets à construire ne ressemble donc en rien à un programme qui serait stocké dans un répertoire d'alternatives potentielles, mais dépend des informations reçues "au fil de l'eau". Cette flexibilité de la PàP rend possible une mise en oeuvre du processus de représentation sur des domaines de connaissances qui peuvent évoluer. La démarche qui va être suivie vise à construire des structures de représentations d'unités lexicales initialement sous-déterminées et qu'il conviendra ensuite d'enrichir.

## 7.2. Situation effective par rapport au travail de Harris

Une approche à la Harris débouche sur la mise en évidence de classes d'opérateurs et de classes d'arguments, qui sont supposés fortement liées aux notions et relations du domaine considéré : peut-on y voir là une solution pour l'acquisition automatique de connaissances à partir de corpus et à termes pour la construction d'ontologies? Nous verrons infra qu'il convient de rester prudent sur ce point (Habert & al. 1996). Notre travail se situe dans un courant de linguistique expérimentale et utilise une approche symbolique, celle des sous-langages. Ce travail vise à caractériser syntaxiquement et sémantiquement certaines configurations nominales dans un domaine donné. En effet, il est souvent impossible de savoir si une configuration est acceptable ou non si on se place hors du domaine en cause (Habert & al. 1996).

### Mise en évidence des comportements des mots d'un corpus pour une automatisation de leur représentation et de leur classement dans un programme de TALN manipulant des prototypes

Notre travail s'inscrit dans une reprise de l'approche harrisienne et vise à automatiser les traitements (de représentation des mots et de leur classement) et à souligner les limites de cette induction de savoirs (un travail d'interprétation manuel semble indispensable).

- 1. Nous limitons notre étude à certaines configurations nominales et plus particulièrement aux séquences du type `NAdj` (Justeson & Katz 1995). Il s'agit en particulier de rechercher au niveau des mots des régularités et des redondances d'utilisation puis d'évaluer les regroupements obtenus (-> partie 2 chapitre 8 : expérimentations sur corpus).

- 2. Notre tâche vise à automatiser les traitements de représentation des mots (sous la forme de prototypes) puis leur classement : ces traitements sont conçus comme des processus évolutifs i.e. les résultats construits peuvent être affinés dès que de nouveaux savoirs permettent d'améliorer la qualité de ces résultats. Ces traitements sont précédés d'une phase d'extraction de savoirs à partir de corpus. La représentation inductive des mots s'appuie sur les résultats d'analyse fournis par les analyseurs Lexter (Bourigault 1993) et Cyclade (Habert 1995). Le but de ces outils est d'une part d'extraire des savoirs à partir de corpus (Lexter) et d'autre part de simplifier ces savoirs puis de caractériser leurs fonctionnements (Cyclade).

Il est important de souligner que les résultats construits ne sont pertinents que pour les corpus étudiés. Les relations syntactico-sémantiques mises en avant par le travail d'extraction de savoirs sont propres aux corpus examinés[3], le travail de représentation et de classement des mots sur la base des savoirs reçus est donc lui aussi complètement dépendant de ces corpus. Si on peut envisager d'élaborer une ontologie à partir des résultats construits (un travail d'interprétation sur les résultats construits reste à définir et à faire), celle-ci ne sera elle aussi pertinente que pour le corpus étudié (Zweigenbaum & al. 1997).

[3] *"Un corpus de spécialité ne permet pas de repérer toutes les relations, bon nombre d'entre elles font partie de l'implicite partagé par les locuteurs du sous-langage et n'effleurent pas"* (Habert & al. 1995).





## 7.3. Extraction des fonctionnements lexicaux : LEXTER et ZELLIG

### 7.3.1. Buts : retrouver les fonctionnements lexicaux

Pour construire une représentation des savoirs lexicaux, on utilise des résultats d'analyse fournis par des outils d'analyse : Lexter et Cyclade. Le but de ces outils est d'une part d'extraire des savoirs à partir de corpus (Lexter) et d'autre part de simplifier ces savoirs puis de caractériser leurs fonctionnements (Zellig).

### 7.3.2. LEXTER (Bourigault 1993)

Lexter prend en entrée des textes longs et produit des arbres d'analyses de séquences nominales en décomposant ces séquences en Tête (T) et Expansion (E), de manière récursive. Sur la séquence `"stenose de le tronc commun gauche"`, on obtient l'analyse suivante.

```
[T [T stenose] [E severe]][E de le [T tronc][E commun]][E gauche]
```

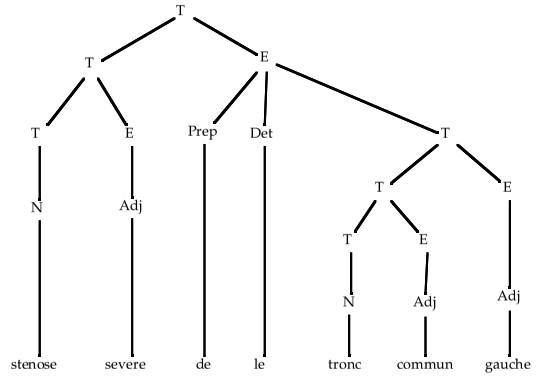


Figure 7.1 : "stenose severe de la tronc commun gauche".

Il est important de souligner que la démarche de Lexter est endogène c'est à dire que le travail d'analyse s'appuie uniquement sur les résultats d'analyse déjà construits pour analyser de nouvelles séquences.

Si le corpus contient la séquence `"angine de poitrine instable"`, on a deux analyses possibles pour cette séquence :

(1) `[angine de poitrine] instable`
(2) `angine de [poitrine instable]`

Lexter prend appui sur les séquences déjà analysées pour produire une analyse de cette séquence. Si on a la séquence `"angine de poitrine"` et que l'on n'a pas `"poitrine instable"` :

`angine de poitrine` existe

`poitrine instable` n'existe pas

Lexter produira l'analyse (1).

## 7.3.3. ZELLIG (Habert & al. 1996)

### 7.3.3.1. Buts

- Simplifier les arbres d'analyses fournis ici par LEXTER
- Mettre en évidence les relations élémentaires de dépendance entre mots pleins

### 7.3.3.2. Simplification d'arbres d'analyse

Le logiciel ZELLIG (Habert & al. 1996) a pour tâche de simplifier les arbres d'analyse fournis ici par Lexter et de mettre en évidence les arbres qui présentent des relations élémentaires de dépendance entre mots pleins. Pour le travail d'extraction des arbres élémentaires, un certain nombre d'arbres abstraits (sans terminaux) sont déclarés comme élémentaires. On crée une table d'arbres de ce type. Sont considérés comme élémentaires les arbres mettant en évidence une relation binaire entre deux mots pleins, nom ou adjectif, dans des schémas comme, par exemple, `N Prep N` ou `N Adj`. Ces dépendances associent à un élément gouverneur (nommé `tête`) soit un `argument` soit un `circonstant`."*Les dépendances élémentaires ainsi définies n'ont pas forcément de réalisation effective dans le corpus mais ils correspondent à des relations de dépendance vérifiées dans les arbres d'analyse, si l'on passe par une représentation logique de ces arbres et de ces dépendances élémentaires*" (Habert & al. 1996).

Sur notre exemple d'arbre fourni par Lexter, celui-ci est d'abord transformé en un arbre syntagmatique via le transducteur FRT (Habert & Bourigault 1996). Les Symboles non-terminaux sont numérotés pour pouvoir y faire référence.

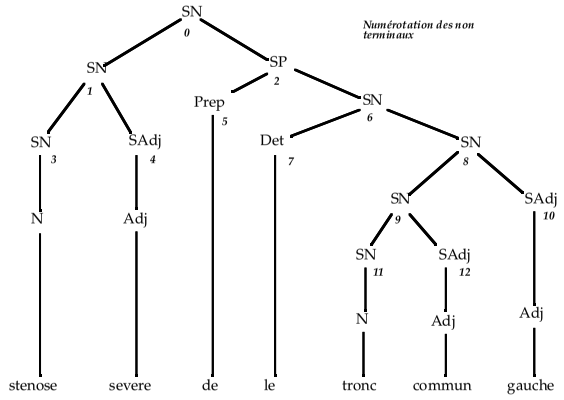


Figure 7.2 : Un arbre syntagmatique.

Le programme Cyclade (Habert & Nazarenko 1996) est ensuite chargé de déterminer les arbres élémentaires via un filtrage de quasi-arbres (Habert & Folch 1996). Il met à jour des arbres





élémentaires qui ne sont pas directement présents dans l'arbre de départ. Au total, les dépendances élémentaires mises à jour sont les suivantes :

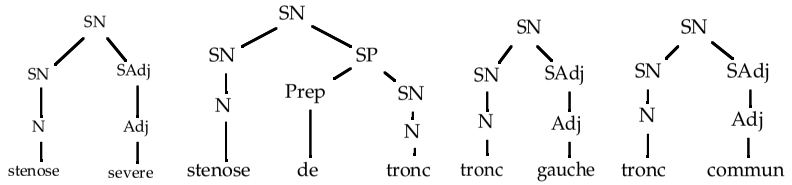


Figure 7.3 : Cyclade : simplifications des arbres d'analyse.

## 7.3.4. Difficultés pour extraire des arbres "élémentaires"

L'apprentissage à partir de savoirs extraits sur corpus et le travail de caractérisation de ces savoirs posent des problèmes majeurs pour la représentation des fonctionnements lexicaux.

### 7.3.4.1. Les arbres d'analyse produisent des arbres "élémentaires" non pertinents

Si on dispose dans le corpus des séquences,

`N coronaire droite` et `N coronaire`
(où N n'est pas un nom d'artère)

l'analyse proposée par Lexter est la suivante :

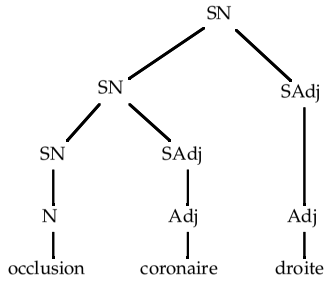


Figure 7.4 : [occlusion coronaire] droite

On obtient donc par élimination du modifieur l'arbre élémentaire :

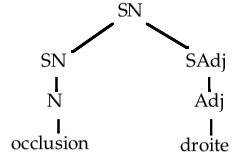


Figure 7.5 : occlusion droite

Or "droite" dans la séquence précédente est un modifieur de l'adjectif dénotant une artère. L'analyse à considérer est donc la suivante :

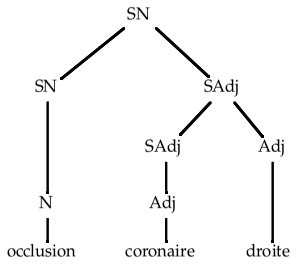


Figure 7.6 : occlusion [coronaire droite]

## 7.3.4.2. Imprévisibilité des contraintes sur les combinaisons d'arbres

- artère et infarctus entrent dans des relations de localisation qui se réalisent de manières distinctes :

  artere {coronaire circonflexe diagonal...}, infarctus {anterieur inferieur apical}

- Proximités de contextes entre adjectifs...

  coronaire et coronarien partagent des contextes :

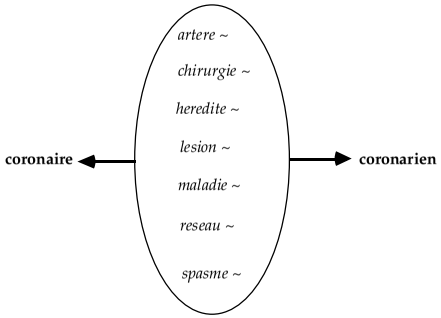


Figure 7.7 : coronaire-coronarien

- Mais ces deux mots ont des modes de combinaison divergents

  coronarien est associé à des adjectifs évaluatifs {severe, significatif, important}, coronaire ne l'est pas.

La figure 7.8 met en avant la diversité des combinaisons d'arbres réalisées sur une même (sous) famille de mots.

## 7.3.4.3. Imprévisibilité des comportements lexicaux





Le travail d'extraction confirme le fait que les comportements lexicaux ne suivent pas des parcours uniformes sur des entités de même catégorie. Et surtout qu'il est difficile de les prévoir. D'une part les mêmes comportements lexicaux ne se réalisent sur tous les mots d'une même catégorie, ils se distribuent sur des sous-familles particulières. D'autre part, ces comportements ne se réalisent de manière uniforme sur les différentes familles de mots d'une même catégorie. Sur telle famille de mots, un comportement se réalise d'une manière qui est différente de sa réalisation sur une autre famille.

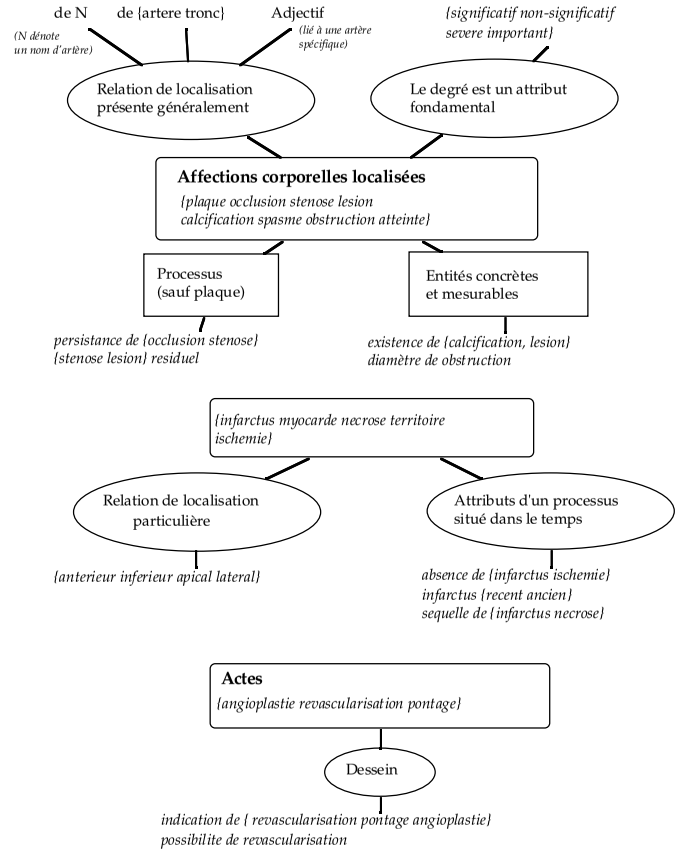


Figure 7.8 : Les comportements des mots sont imprévisibles

La figure précédente présente les comportements qui se réalisent sur certaines familles de mot (des noms) de notre corpus. Les relations rencontrées ne se réalisent pas de manière uniforme sur ces mots : c'est le cas de la relation de localisation qui est soit généralement présente, soit présente sous une forme particulière, soit non présente.

Le travail d'extraction de savoirs à partir de corpus permet d'identifier les valeurs particulières des fonctions lexicales (Mel'cuk 1988) (rencontrées généralement dans la langue) pour un domaine de discours donné. Ces réalisations varient semble-t-il selon les domaines. Ce

qu'il est important de souligner ici, c'est que le processus d'extraction puis de simplification des savoirs est conçu comme un processus évolutif. Si sur un état particulier du corpus on obtient des savoirs non pertinents, on peut penser que si le corpus est élargi, les nouvelles séquences analysées permettront d'affiner les savoirs déjà recensés et donc d'éliminer les savoirs non pertinents retenus à la phase précédente (-> partie 2 chapitre 8 : un système qui module les flux d'information).





# 7.4. Du lexique jaillit un réseau de prototypes

*"On remarquera que la convention, partagée par les linguistes, les informaticiens et les quelques mathématiciens qui s'y intéressent, amène à dessiner les arbres comme orientés vers le bas, racine en l'air, à l'envers en somme. Etrange!"*
Jacques Roubaud, Mathématique:, Ed. du Seuil 1997.

## 7.4.1. Définitions préliminaires

Dans la suite nous utiliserons les notations suivantes :

- On appellera **arbre élémentaire** un arbre associé à une entrée lexicale et qui traduit une relation binaire de dépendance entre mots pleins. Cette unité lexicale étant présente dans l'arbre au niveau de l'une des feuilles.

- On appellera **arbre dérivé** ou **arbre d'analyse** un arbre associé à un arbre élémentaire : cet arbre d'analyse est obtenu par des opérations effectuées à partir de l'arbre élémentaire (adjonction, substitution...).

- On appellera catégorie syntaxique ou mineure une des catégories grammaticales traditionnelles telles que `Nom, Verbe, Adjectif...`

- On appellera prototype d'**arbre élémentaire** l'objet informatique (le prototype) défini pour représenter un arbre élémentaire.

- On appellera prototype d'**arbre d'analyse** l'objet informatique (le prototype) défini pour représenter un arbre d'analyse.

## 7.4.2. Dégager les comportements syntaxiques d'une entrée lexicale

Nous travaillons ici à partir des résultats obtenus par (Habert & Fabre 1995).

On donne donc ci-dessous certains résultats issus du travail réalisé sur le corpus MENELAS et dont on a pu extraire les informations suivantes :

```
#LESION#

1#(SN (SN NOM) (SPREP PREP (SN NOM)))#
```

- LESION DE TRONC simplification
  arbre complexe lié: LESION DE LE TRONC GAUCHE

- LESION SUR CIRCONFLEXE
  arbre complexe lié: LESION SUR LA CIRCONFLEXE DISTALE

```
2#(SN (SN NOM) (SADJ ADJ))#
```

- LESION NON-SIGNIFICATIF

- LESION RESIDUEL

- LESION CORONARIEN
  arbre complexe lié: LESION CORONARIEN SEVERE

- LESION SIGNIFICATIF
  arbre complexe lié: PAS DE LESION SIGNIFICATIF

- LESION SEVERE
  arbres complexes liés:
  MONTRE DE LES LESION TRITRONCULAIRES SEVERE
  LESION ATHEROMATEUX SEVERE
  LESION CORONARIEN SEVERE
  LESION TRITONCULAIRES SEVERE

- LESION TRITRONCULAIRES
  arbres complexes liés:
  MONTRE DE LES LESION TRITRONCULAIRES SEVERE
  LESION TRITONCULAIRES SEVERE

- LESION COMPLEXE

- LESION DE DIFFUS

- LESION ATHEROMATEUX
  arbres complexes liés:
  EXISTENCE DE LESION ATHEROMATEUX
  EXISTENCE DE LESION ATHEROMATEUX SEVERE
  LESION ATHEROMATEUX SEVERE

- LESION IMPORTANT

- LESION MINIME

- LESION DIAGONAL
  arbre complexe lié: LESION CORONAIRE DIAGONAL

- LESION CORONAIRE
  arbres complexes liés:
  EXISTENCE DE UNE LESION CORONAIRE
  LESION CORONAIRE DIAGONAL

- LESION CIRCONFLEXE
  arbre complexe lié: LESION CIRCONFLEXE DISTALE

- LESION BITRONCULAIRES
  arbre complexe lié: EXISTENCE DE LES LESION BITRONCULAIRES

- LESION DISTALE
  arbre complexe lié: LESION CIRCONFLEXE DISTALE

- LESION VERTEBRAL

Ces premiers résultats vont permettre une première phase de représentation des informations ainsi révélées : le savoir attaché à cette première entité lexicale va servir de base de travail. Il est clair que nous devons garder à l'esprit qu'il conviendra à coup sûr de revenir sur cette première ébauche de représentation. Notons `ACL` la famille lexicale regroupant les noms désignant une affection corporelle localisée. `Lésion` appartient à cette famille. On décide donc de construire un prototype associé à `lésion` qui porte les attributs habituellement associés à cette entrée lexicale (<- représentations antérieures). On associe de plus à cette entrée la liste des comportements syntaxiques que le corpus a révélés, i.e. les arbres élémentaires et leurs arbres complexes liés. Ces comportements sont attachés à notre prototype via la délégation, l'attribut `parent` indiquant l'objet qui porte ces comportements. Pour ne pas figer cette représentation, cette assignation est réalisée via un attribut modifiable.





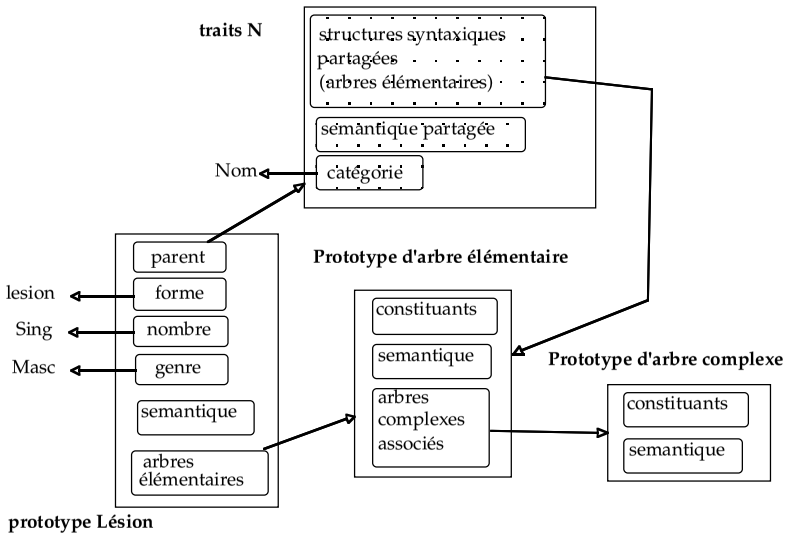


Figure 7.9 : Un prototype pour `lesion`.

## 7.4.3. Etendre la représentation en ordonnant les comportements

Si l'on considère maintenant une nouvelle entrée appartenant à la même famille lexicale, dont les arbres élémentaires sont de même nature que ceux associés à `lésion`, on peut se servir de cette première phase de représentation pour construire une représentation prototypique de cette nouvelle entité.

Famille des affections localisées :

On peut synthétiser les comportements syntaxiques des éléments de cette famille de la manière suivante :

$Comp(ACL_1)$ : $N_{ACL}$ + (1) DE {artere, tronc}
$Comp(ACL_2)$ : $N_{ACL}$ + (2) DE {nom d'artère}
$Comp(ACL_3)$ : $N_{ACL}$ + (3) SUR {nom d'artère}
$Comp(ACL_4)$ : $N_{ACL}$ + (4) ADJ {adjectif lié à un nom d'artère}

Les prototypes représentant des éléments de cette famille lexicale délègueront donc les comportement syntaxiques associés à cette famille dans l'objet (`traits`) que l'on peut noter :

$$\{i = 1,2,3, 4;\ Comp(ACL_i)\}$$

Si un des prototypes représentés ne possède pas tous les types de comportement associés à cette famille, il convient de remodeler cette représentation de la manière suivante :

- On factorise les comportements communs que l'on regroupe dans un objet `traits` donné, et on définit un nouvel objet `traits` qui porte les comportements résiduels.

- On modifie les liens parentaux entre les prototypes définis et leurs parents respectifs : un prototype donné peut déléguer ces comportements aux deux objets `traits` ainsi définis ou vers un seul de ces objets `traits`.

De même, si l'on considère une nouvelle famille d'entités lexicales, on peut, dans un premier temps, entreprendre une première phase de représentation de la même manière que celle que nous avons mise en place pour `lesion`. Si par la suite, la représentation adoptée met en évidence des parallèles comportementaux entre les deux familles, il reste possible de mettre en commun les comportements syntaxico-sémantiques visés. Il s'agit en quelque sorte de définir des catégories abstraites représentant des comportements partagés par un ensemble d'entités lexicales de manière transversale par rapport à leur famille catégorielle initiale.

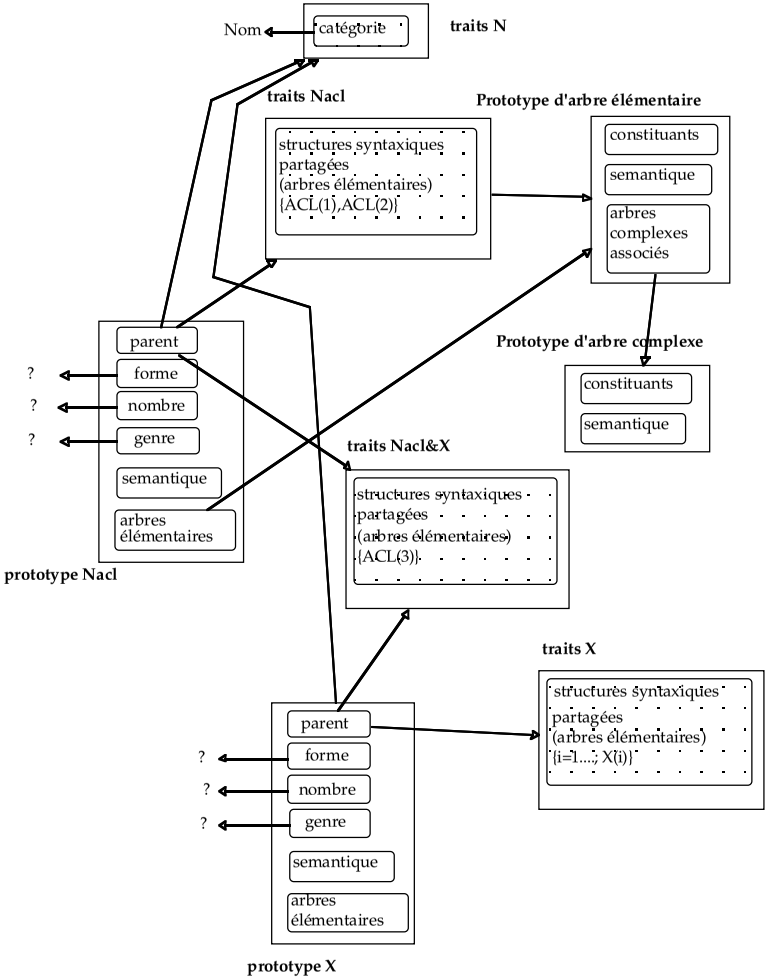


Figure 7.10 : Des Comportements et des Prototypes.





# 7.5. Esquisse d'un réseau de comportements syntaxico-sémantiques hiérarchisés

Dans le cadre de notre analyse automatique utilisant le langage à prototypes Self, nous allons donc utiliser des résultats d'analyses syntaxiques produits par les analyseurs Lexter et Cyclade sur des corpus de langages spécialisés pour associer aux mots (et donc à leurs représentations prototypiques) des contraintes. L'objectif à atteindre dans notre développement de l'analyse automatique avec des prototypes est de trouver, sur un corpus donné, les éléments de savoir linguistique qui vont permettre de définir un système d'interprétation cohérent et maîtrisable; c'est-à-dire les faits et les règles qui peuvent produire, à partir de ce corpus, des éléments utiles pour une analyse automatique de textes.

## 7.5.1. Une première phase de représentation des informations recueillies sur un corpus donné

On affecte désormais aux prototypes de mot construits leurs comportements syntaxico-sémantiques. On considère pour commencer que l'on récupère des informations sur certaines entités lexicales sous la forme suivante :

```
pontage Nom (Sn(Det Nom) NPrepN(Nom Prep Nom) AdjN(Adj Nom) NAdj(Nom Adj) PrepN(Prep Nom) ...)
```

Le travail d'extraction associe au mot `pontage` de catégorie `Nom` les arbres `Sn(Det Nom) NPrepN(Nom Prep Nom) AdjN(Adj Nom) NAdj(Nom Adj) PrepN(Prep Nom)`... Ces informations sont fournies par le traitement d'extraction de connaissances sur un corpus donné suivant le travail présenté supra. Ces informations vont permettre d'enrichir la représentation des prototypes lexicaux définis dans notre cadre de travail : ces derniers ayant à priori peu de savoir sur ce qu'ils sont capables de faire; ils ne disposent pas, par exemple, de savoir sur les structures syntaxiques dans lesquels ils peuvent être insérés. A la différence de ce que nous faisions précédemment, il ne s'agit plus de pré-déterminer et d'encoder "à la main" les savoirs syntaxiques que l'on peut attacher aux mots. Il s'agit au contraire d'utiliser les savoirs recueillis sur des séquences présentes dans un corpus et de les affecter aux entités prototypiques concernées.

### Association automatique des comportements syntaxiques aux prototypes lexicaux

Ces informations sont donc affectées aux prototypes visés par ces contraintes de la manière suivante :

- A partir de la description précédente du mot `pontage`, on affecte automatiquement au prototype représentant ce mot les comportements qui lui sont associés : les arbres `Sn`, `NPrepN`... sont associés au mot `pontage`. On associe donc au prototype lexical `pontage` les prototypes d'arbres `Sn`, `NPrepN`...

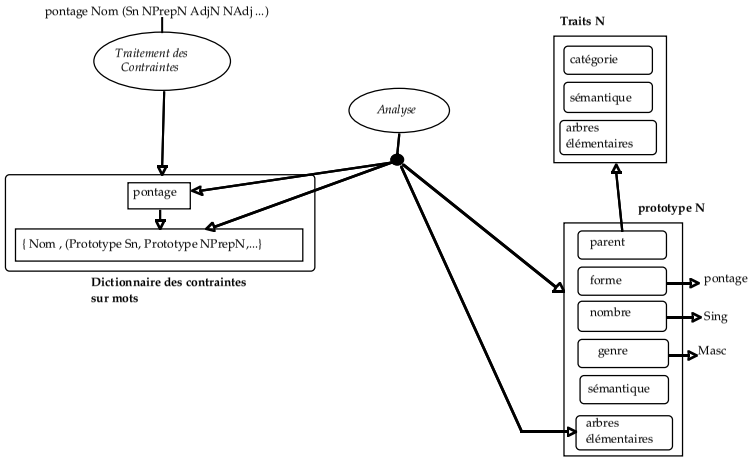


Figure 7.11 : Les prototypes apprennent peu à peu.

Il convient de préciser que cette première phase de représentation maintient pour le moment la prédéfinition des savoirs syntaxico-sémantiques pour des représentations prototypiques de catégories syntaxiques représentant des "groupes de mots" : les prototypes `Sn`, `NPrepN`... disposent d'un savoir syntaxico-sémantique qui a été encodé à l'image de ce que nous avons déjà présenté. Nous verrons infra comment la génération automatique de prototype d'arbre nous permettra de ne pas utiliser de savoirs préconstruits (-> automatisation des traitements de génération de prototypes de mots et d'arbres).

## 7.5.2. Construction d'une hiérarchie approximative

Le schéma précédent ne tient pas compte de ce que nous avons présenté supra concernant le partage des savoirs syntaxiques communs à un ensemble de prototypes donnés. Il manque pour cela une phase de travail capable d'évaluer les similitudes de comportements entre les entités représentées.

Dans l'exemple qui suit, on dispose au départ des descriptions textuelles des mots `pontage` et `lesion`. On affecte donc aux prototypes lexicaux concernés leurs comportements. On cherche ensuite si ces deux prototypes lexicaux possèdent des comportements communs. Si tel est le cas, les comportements partagés par ces deux prototypes lexicaux sont affectés automatiquement à l'objet existant qui porte déjà leurs comportements partagés (mise à jour de l'attribut `arbre élémentaire` dans l'objet `Traits N`).

- Phase 1 : Lecture des contraintes, vérification de l'existence des prototypes catégoriels ou création de ceux-ci.
- Phase 2 : Attribution des arbres élémentaires aux prototypes de mot.
- Phase 3 : Classement automatique des prototypes de mot en fonction de leurs comportements[4], mise en place d'un héritage local de comportements.

[4] On classe les objets représentant des mots sur la base des comportements réels associés aux mots ainsi modélisés. Les comportements informatiques qui ne décrivent pas les mots dans leur activité langagière ne sont pas pertinents pour ce classement.





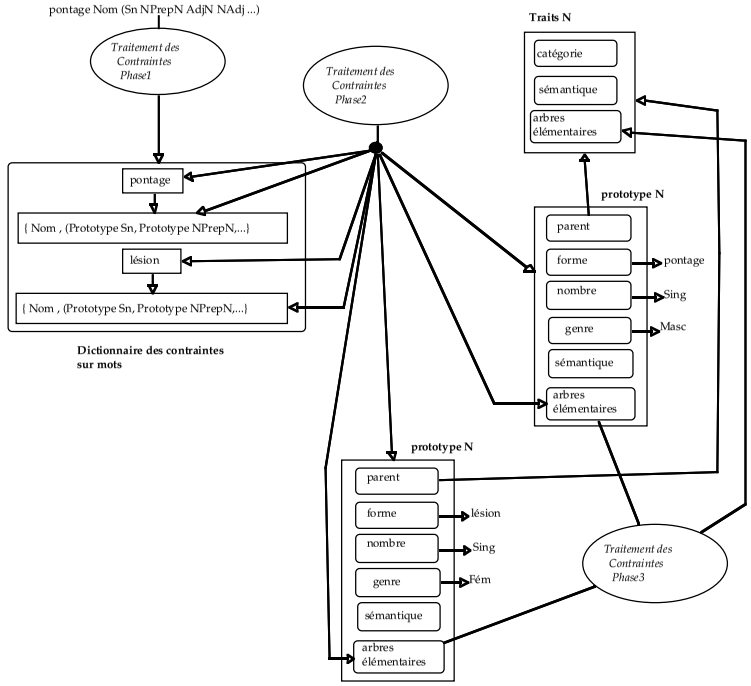


Figure 7.12 : Les prototypes s'organisent peu à peu.

## 7.5.3. Illustration avec Self

Nous présentons en annexe (-> annexe partie 2 chapitre 7) une trace de l'exécution des opérations décrites ci-dessus. Dans l'exemple donné, on exécute une réorganisation des savoirs associés aux prototypes représentant la catégorie des `Noms`. On fait tout d'abord une recherche de tous ces prototypes et de leurs contraintes respectives. Puis on détermine les comportements communs : une classe de comportements partagés par tous les prototypes qui disposent de contraintes. Cette classe pourra ensuite être associée à l'objet `Traits` commun à tous ces prototypes. Les prototypes qui ne disposent pas de contraintes à l'issue de la définition de celles-ci disposeront ainsi d'un savoir par défaut. Cela ne veut pas dire que ce savoir sera utilisé pour les entités qui ne disposent pas de contraintes particulières. Il s'agira simplement de marquer que sur une famille donnée, certains prototypes possèdent des contraintes et que ces savoirs sont disponibles mais ne sont pas utilisables pour tous les éléments de la famille.

## 7.5.4. Hiérarchie mouvante et apprentissage

Si la représentation prototypique d'une entrée lexicale n'est pas affectée par le travail d'extraction de connaissances sur le corpus donné, il va être possible de vérifier que les comportements hérités, via le partage défini dans cette phase d'apprentissage de comportements par défaut, sont valides ou non pour cette entité. Dans le premier cas, on peut avoir une confirmation d'un savoir construit par défaut, dans le second cas, on peut envisager que cet apprentissage implique une réorganisation des savoirs représentés : suppression du

partage (globalement ou localement) pour cette entité, mise en place d'une hiérarchisation plus fine des comportements qui tienne compte des particularités mises à jour pour cette entité.

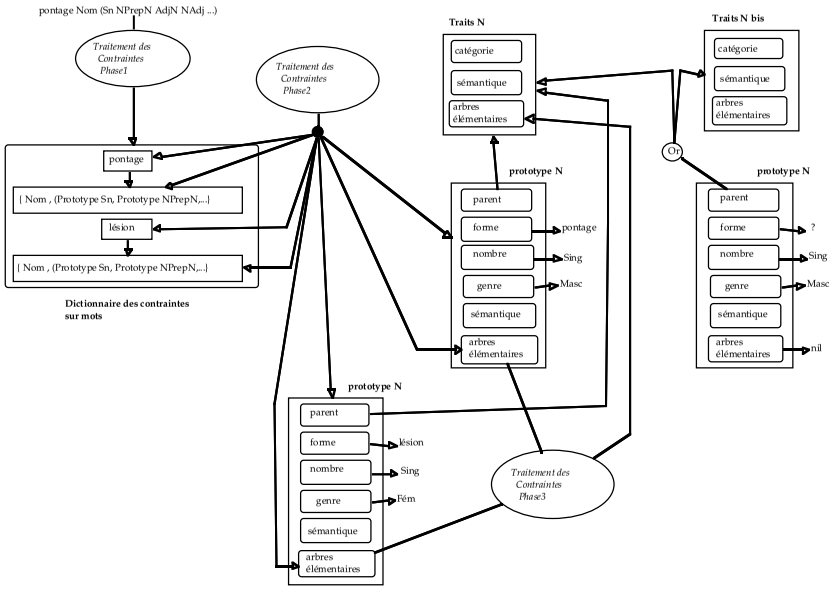


Figure 7.13 : Les représentations se développent par apprentissage et organisation des savoirs.





# 7.6. Automatisation des traitements de représentation des mots et des arbres

Approches d'un dispositif de génération automatique de prototypes de mot et de prototypes d'arbre à partir de savoirs extraits sur corpus.

## 7.6.1. Construire les représentations à la demande

Puisqu'il n'est pas possible de prédéterminer ce que peuvent être les informations fournies par la phase d'extraction de savoirs à partir de corpus, le dispositif doit permettre une génération automatique de représentations des informations reçues (mot, catégorie...). Le dispositif doit pouvoir désormais se passer des représentations prédéfinies d'unités de langue, il doit les créer de toutes pièces si nécessaire puis les multiplier.

En reprenant notre exemple précédent :

```
(1) pontage Nom (Sn(Det Nom) NPrepN(Nom Prep Nom) AdjN(Adj
Nom) NAdj(Nom Adj) PrepN(Prep Nom) ...)
```

la phase d'extraction d'informations sur le corpus va produire une suite de contraintes telles que celles associées à `pontage` dans `(1)`; ce qui va conduire ensuite à associer au prototype lexical représentant le mot `pontage` la liste des prototypes représentant les catégories syntaxiques présentes dans la liste des contraintes associées à `pontage` dans `(1)`. Cette association ne posera pas de problèmes si les prototypes associées à ces catégories syntaxiques sont déjà définis. Dans le cas contraire, on ne peut que produire une erreur. Pour pallier ce manque éventuel de prédéfinition de prototypes représentant une catégorie syntaxique donnée, il semble raisonnable de disposer d'un générateur de prototypes capable de produire à la volée un objet capable de représenter une catégorie syntaxique non encore représentée.

Sur ce point, on retrouve un point fondamental déjà énoncé concernant le problème fondamental de l'analyse automatique dans un univers prototypique et plus précisément la nécessité d'aller vers un effacement progressif de la notion habituelle de grammaire telle qu'elle est utilisée dans l'analyse automatique. Notre approche se situe en effet dans une perspective où l'analyse automatique reconstruit une grammaire en fonction des informations rencontrées; et non dans une perspective où l'analyse automatique se contente d'opérer la vérification d'une grammaire donnée. On part de rien (les représentations prototypiques initiales sont minimales) et on reconstruit des représentations ou on les affine en fonction des savoirs mis à jour. En particulier, il s'agit aussi de reconstruire une grammaire, en fait un prototype de grammaire qui, du fait de l'approche retenue ici, n'existera pas au démarrage de l'analyse, mais se révèlera en fin d'analyse.

## 7.6.2. Génération automatique de prototypes

On a donc mis en place dans les phases de traitements des contraintes telles qu'elles sont présentées dans les figures précédentes, une étape de création "à la volée" (i.e. automatique) de prototypes si on rencontre une catégorie syntaxique non encore représentée. Ces nouveaux objets seront créés soit en utilisant les représentations existantes, soit de toutes pièces.

### 7.6.2.1. Génération par création explicite d'un nouvel objet

On peut ainsi être contraint de construire de toutes pièces une représentation prototypique d'une catégorie donnée. On dispose pour cela d'un opérateur capable de produire une représentation qui tienne compte des informations fournies (nom de la catégorie, constituants éventuels...).

- Génération à la volée (i.e. automatique) de prototypes par création explicite d'un nouvel objet

Si la catégorie d'une unité lexicale n'est pas encore représentée, on définit les objets pour la représenter avec les attributs adéquats. Dans l'exemple qui suit, si le mot `pontage` ne possède pas encore de représentation prototypique et si la catégorie des noms n'est représentée par aucun prototype, on construit[5] de toutes pièces une nouvelle représentation prototypique de cette nouvelle famille catégorielle en utilisant les informations associées à ce premier élément de cette famille.

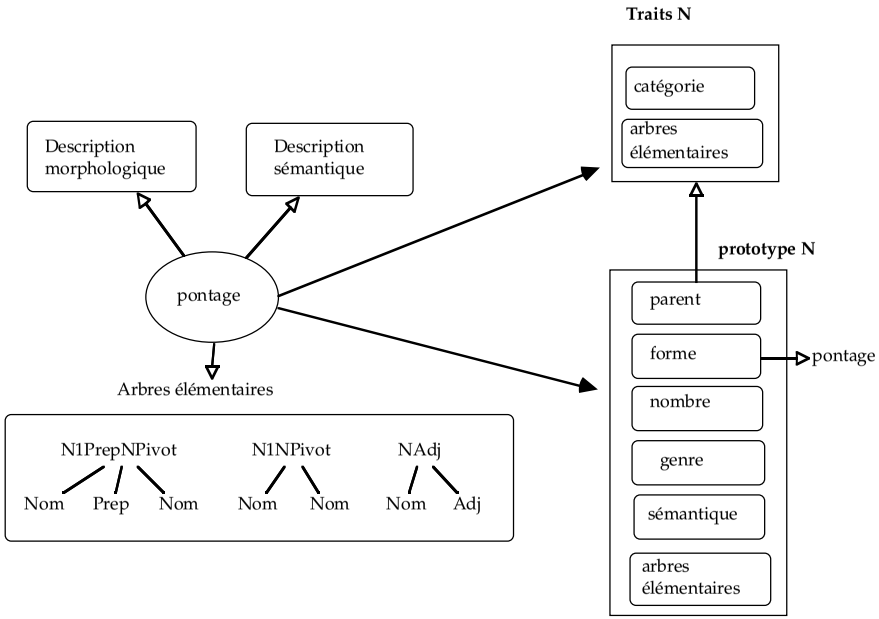


Figure 7.14 : Génération à la volée de prototype catégoriel.

[5] La structure construite pour représenter un mot contient un certain nombre d'attributs qui seront valués dès que les informations pertinentes pour ces attributs seront disponibles.





Si les arbres élémentaires associés à `pontage` ne possèdent pas encore de représentation prototypique (si la catégorie syntaxique tête d'un arbre élémentaire n'est pas encore représentée), le générateur d'arbre construit automatiquement des structures pour représenter ces arbres en tenant compte des informations fournies (nombre de composants, position des composants...).

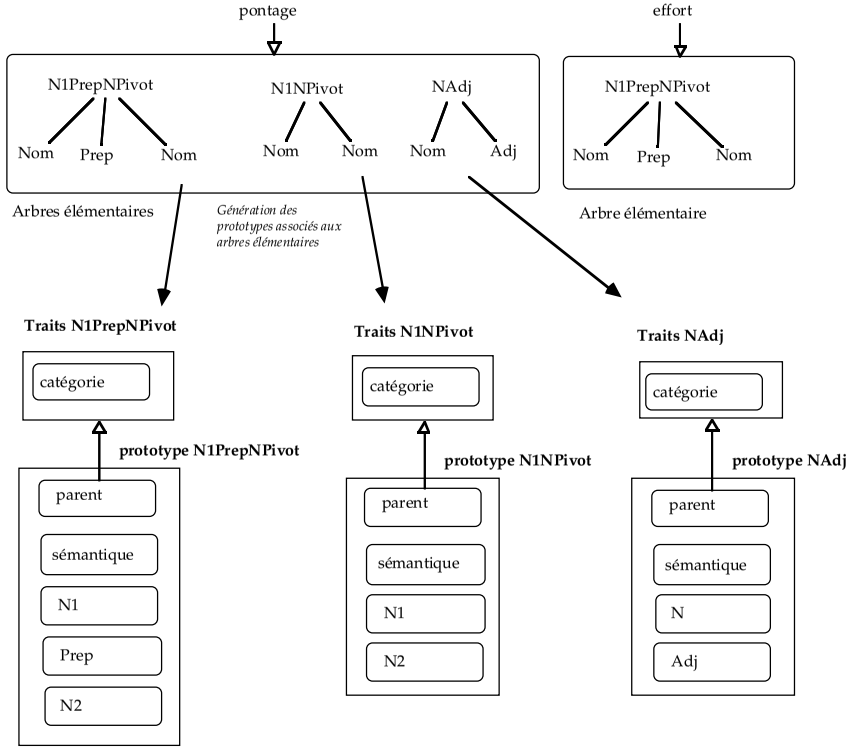


Figure 7.15 : Génération à la volée de prototypes d'arbre élémentaire.

Dans la figure précédente, le prototype `N1PrepNPivot` représente l'arbre `N1 Prep N2` dans lequel le `Nom` en position `N2` est le pivot de l'arbre à construire.

- Génération à la volée (i.e. automatique) de prototypes d'arbres d'analyse par création explicite d'un nouvel objet

De même, si les arbres d'analyse associés à `pontage` ne possèdent pas encore de représentation prototypique (si la catégorie syntaxique tête d'un arbre d'analyse n'est pas encore représentée), le générateur d'arbre construit automatiquement des structures pour représenter ces arbres en tenant compte des informations fournies (nombre de composants, position des composants...).

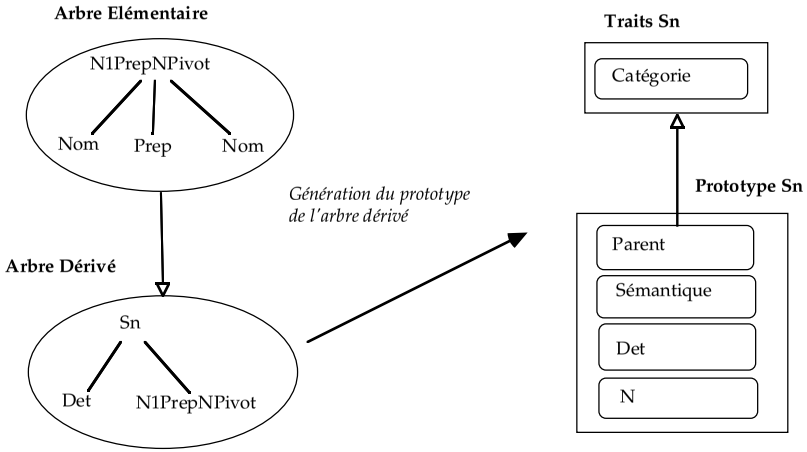


Figure 7.16 : Génération à la volée de prototype d'arbre d'analyse.





## 7.6.2.2. Génération par clonage et ajustements

On dispose aussi des outils habituels dans un environnement de programmation qui manipule des prototypes pour mettre en oeuvre la génération de nouveaux prototypes, à savoir l'opération de clonage pour dupliquer les objets concernés puis les opérations qui permettent d'ajuster les valeurs des attributs d'un objet donné (ajouter des attributs, en retirer, ajuster leurs valeurs...).

- Génération à la volée (i.e. automatique) de prototypes par clonage et ajustements :

Si le mot `"effort"` n'a pas encore de représentation prototypique, on utilise le clonage (sur une entité existante et de même catégorie) puis l'ajustement pour le représenter.

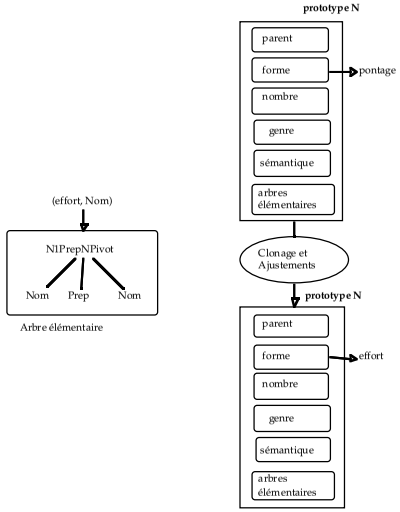


Figure 7.17 : Génération à la volée de prototype par clonage et ajustement.

On donne en annexe (-> annexe partie 2 chapitre 7) une trace de l'exécution de cette nouvelle phase de génération de prototypes.

## 7.6.3. Une construction inductive des savoirs et une classification dynamique de ces savoirs

La génération de prototypes s'appuie sur les savoirs recueillis pour construire des représentations prototypiques adéquates. Notre prototype de grammaire va donc se construire à partir des informations attachées aux mots. Tout le savoir associé aux formes lexicales va permettre de construire par induction un prototype de grammaire dépendant des informations attachées aux mots et donc au corpus initial qui a servi de base pour l'extraction des informations utilisées ici.

## 7.6.3.1. Construire un prototype de grammaire lexicalisée

La phase d'extraction de savoirs sur le corpus fournit des éléments d'information sur des entités lexicales déjà catégorisées[6]; ces éléments d'information peuvent d'ailleurs être des formes de savoir abstrait (on ne restreint pas le savoir attaché aux mots à de simples relations entre formes lexicales). Nous souhaitons en fait, grâce à ces éléments reçus en amont de la représentation, pouvoir tenir compte du fait que dans cette représentation, certains éléments peuvent se construire par introspection et d'autres non. Les informations initiales que nous manipulons ont pour le moment l'allure suivante :

```
#Dc
pontage
Nom
Sn
(('Det' & 'Nom') asList)
NPivotprepN2
(('NomPivot' & 'Prep' & 'Nom') asList)
N1prepV
(('Nom' & 'Prep' & 'Verbe') asList)
AdjN
(('Adj' & 'Nom') asList)
NAdj
(('Nom' & 'Adj') asList)
NAdj
(('Prep' & 'Nom') asList)
N1NPivot
(('Nom' & 'NomPivot') asList)
Fc#
```

Dans cet exemple le mot `pontage` de catégorie `Nom` est associé aux arbres élémentaires `Sn`, `NPivotprepN2`, `N1prepV`, `AdjN`, `NAdj`, `NAdj`, `N1NPivot`. A chacun de ces arbres élémentaires, on associe la liste de ses composants: `(('Det' & 'Nom') asList)`, une liste en Self, associée à `Sn`... Il est clair que ce savoir initial demande à être complété par des informations supplémentaires. On peut par exemple y adjoindre des informations morphologiques, sémantiques... sur les mots (<- génération automatique de prototypes) et tout type d'information pertinente pour la description du mot à représenter. On peut ainsi y ajouter les informations qui indiquent que tel arbre élémentaire est associé à tel(s) arbre(s) complexe(s), en indiquant l'historique de la dérivation permettant de passer de l'un à l'autre (Habert & Gaussier 1997).

```
#Dc
pontage
Nom
(Description morphologique)
Masc
Sing
...
(Informations sémantiques...)
...
Sn
(('Det' & 'Nom') asList)
(Arbre complexe lié et dérivation)
NPivotprepN2
(('NomPivot' & 'Prep' & 'Nom') asList)
(Arbre complexe lié et dérivation)
N1prepV
(('Nom' & 'Prep' & 'Verbe') asList)
(Arbre complexe lié et dérivation)
AdjN
```

[6] A la différence de (Bensch & Savitch 1995), notre approche n'a pas à se poser la question de la catégorisation des éléments lexicaux à représenter.





```
(('Adj' & 'Nom') asList)
(Arbre complexe lié et dérivation)
NAdj
(('Nom' & 'Adj') asList)
(Arbre complexe lié et dérivation)
PrepN
(('Prep' & 'Nom') asList)
(Arbre complexe lié et dérivation)
N1NPivot
(('Nom' & 'NomPivot') asList)
(Arbre complexe lié et dérivation)
Fc#
```

D'une part, nous souhaitons construire dynamiquement des représentations informatiques des formes lexicales. D'autre part, en utilisant les savoirs associés aux mots représentés, nous visons à une classification dynamique des mots et de leurs représentations. Nous ne limitons pas le savoir attaché aux prototypes représentant des mots à des associations de formes lexicales (tel mot est fréquement associé à tel(s) mot(s) dans tel(s) contexte(s) (cf. infra)), mais nous souhaitons construire des savoirs comportementaux des formes représentées qui puissent généraliser les savoirs incrémentalement recueillis et construits. On veut par exemple pouvoir repérer que tel groupe de mots est capable de se comporter de manière particulière dans une structure syntaxique donnée (par exemple construction d'une structure de type `N1PrepN2` en position `N1` avec `N2` de type(s) donné(s)). Notre approche de représentation dynamique à partir de savoirs extraits de corpus ne doit pas limiter les savoirs aux seules inter-relations entre les formes lexicales représentées, mais doit pouvoir nous permettre de classer et de prédire les nouvelles représentations à venir.

## 7.6.3.2. Une amorce de classement guidée par la syntaxe

Notre objectif est de tendre vers la détermination de classes sémantiques, de manière inductive, dans une approche symbolique[7], celle des sous-langages. Tout d'abord, il convient de souligner que le classement automatique opéré jusqu'ici s'appuie principalement sur des contraintes syntaxiques. Dans ce qui a été présenté supra, les comportements associés aux représentations prototypiques des entrées lexicales contraintes sont de simples squelettes syntaxiques dépourvus de contraintes particulières. Il est clair que cette affectation de comportements syntaxiques n'est pas suffisante pour une classification cohérente des éléments représentés : *"// des propriétés contextuelles permettent de rapprocher certains mots, mais la syntaxe seule ne permet pas d'identifier les liens sémantiques entre ces unités lexicales. Elle suggère des liens de hiérarchie (`générique` vs. `spécifique`) ou des relations conceptuelles (ex. `a pour partie` ) mais des connaissances extérieures sont nécessaires pour valider la nature de ces liens. Par exemple, la similitude des syntagmes `pontage coronarien` et `pontage saphène` n'indique pas que la mention du vaisseau joue dans un cas le rôle de localisateur (pontage* ***sur*** *la coronaire) et dans l'autre celui d'un instrument (pontage* ***à l'aide de*** *la veine saphène)"* (Habert & Nazarenko 1996).

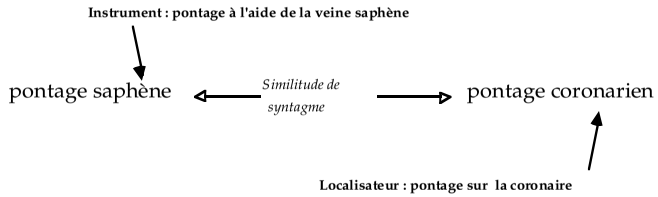


Figure 7.18 : Similitude formelle de syntagmes

[7] (Resnik 1993) présente une approche de construction de classes sémantiques de manière inductive dans une optique statistique.

La syntaxe doit en quelque sorte dégrossir la représentation mais ne permet pas à elle seule de classer les unités représentées; (Habert & Nazarenko 1996) montrent que la syntaxe est incapable à elle seule de délimiter des classes de mots reflétant une notion. On peut avoir un rapprochement de certaines unités lexicales suivant certains comportements syntaxiques (les noms d'affections), mais aucune classification ne se révèle directement à partir des comportements syntaxiques : *"le fonctionnement syntaxique des unités permet donc non pas de construire une ontologie du domaine mais de dégrossir le travail de définition de cette ontologie"* (Habert & Nazarenko 1996). A l'inverse des approches harrissiennes[8] et statistiques[9], notre approche ne conduit pas à la détermination de classes sémantiques satisfaisantes mais elle constitue une méthode d'amorçage pour l'élaboration de l'ontologie du domaine, nous suivons sur ce point la démarche suivie par (Habert & Nazarenko 1996) : la construction de l'ontologie du domaine étudié nécessite un part d'interprétation, *"il y a, entre le flou notionnel inhérent aux langues naturelles (Kayser 1992), y compris aux langues de spécialités (Dachelet 1994), et la stabilité conceptuelle qui est requise dans les ontologies construites, un seuil qui ne peut être franchi automatiquement"* (Habert & Nazarenko 1996).

On peut aussi envisager de projeter les résultats intermédiaires sur des classes sémantiques construites par ailleurs afin d'affiner les savoirs construits et donc leurs classements. Si les projections de savoir permettent d'affiner les arbres élémentaires en amont, en particulier sur l'arbre `NAdj` associé à pontage, on va affiner le travail de représentation des arbres élémentaires associés à pontage.

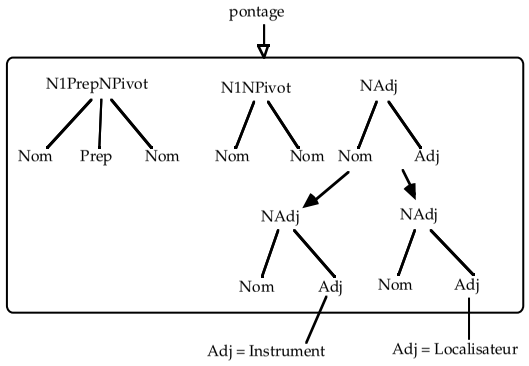


Figure 7.19 : Affiner les savoirs initiaux...

Si on dispose en amont d'informations supplémentaires sur les arbres élémentaires, on peut affiner la construction des représentations prototypiques de ces arbres élémentaires. Sur notre exemple précédent, si la différence dans la nature des liens sémantiques entre les composants de l'arbre élémentaire `NAdj` est précisée sur les arbres élémentaires recueillis en amont, on utilisera cette précision pour spécialiser les prototypes à construire associés à l'arbre élémentaire `NAdj` à représenter.

[8] Dans l'approche suivie par N. Sager, les classes de mots et les formules syntaxiques d'une grammaire de sous-langage sont données comme en correspondance étroite avec les classes d'objets du monde réel et à leurs relations, le travail d'interprétation des résultats construits semble sous-estimé.

[9] Dans les approches statistiques (Church & Hanks 1990, Hindle 1990), les classes de mots sémantiquement pertinentes sont obtenues statistiquement et induites du corpus à partir des seules mesures de similarités entre mots : *"En réalité, les listes de mots obtenues ne constituent pas de véritables classes de mots cohérentes et homogènes. Il est souvent difficile de leur attribuer une étiquette globale, et ce d'autant plus que les mesures de similarité constituent des résumés statistiques bruts dont les critères de construction sont effacés"* (Habert & Nazarenko 1996).





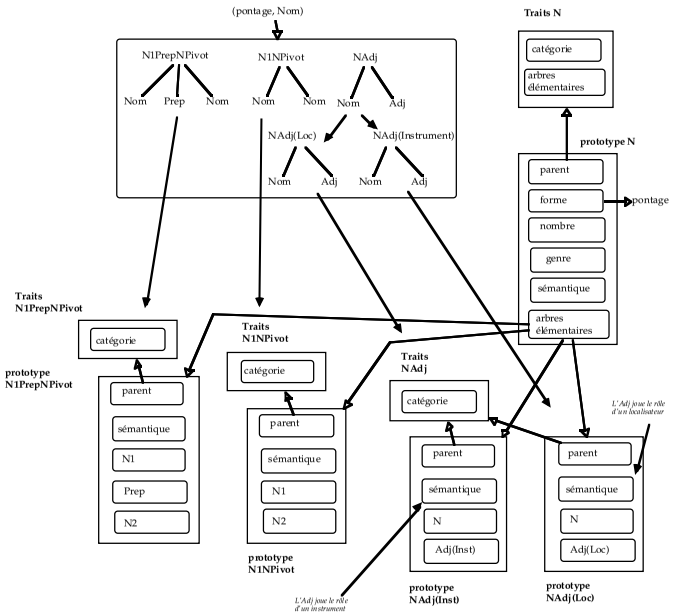


Figure 7.20 : Pour affiner les arbres "élémentaires" à construire.

De plus le classement présenté ne tient pas compte des arbres d'analyse associé aux arbres élémentaires et donc aux mots (-> partie 2 chapitre 8). Or si le classement sur les arbres élémentaires peut rapprocher des mots comme `effort` et `myocarde` sur la base d'un comportement partagé : ces deux mots entrent dans des séquences `N1 Prep N2` en position `N2`. Ce rapprochement ne dit rien de la différence de comportements de ces deux mots dans des arbres d'analyse : dans ces derniers, `effort` ne peut pas y être modifié, alors que `myocarde` l'est toujours par un déterminant.

## 7.6.3.3. Créer des pôles multiples de regroupements de comportements partagés, les abstraire et les organiser

Notre approche vise à travailler à partir du savoir attaché aux prototypes lexicaux représentés puis à repérer les partages possibles sur ces savoirs parmi tous les objets représentés. Il s'agit en fait de mettre en place et d'affiner l'équilibre qui existe entre des savoirs localement répartis et des savoirs partagés qui établissent des liens entre les éléments représentés. Pour mener à bien cette tâche, il est clair que nous devons pouvoir manipuler et assimiler assez d'informations dans le flot de savoirs extraits du corpus de travail. La difficulté est bien évidemment de construire des outils qui permettent au dispositif d'apprendre et de construire du savoir à partir de ce flot de savoirs initialement indépendants puisque répartis sur chacun des mots. En prenant appui sur un premier classement des unités lexicales sur la base de la micro-syntaxe qui leur est attachée, on va pouvoir construire un réseau de pôles de comportements partagés par des ensembles de prototypes (-> incise 1 : Self et l'héritage). Il reste ensuite à opérer un travail d'interprétation de ces différents pôles de regroupements de comportements. Il est sûr que ces regroupements ne permettent pas de classer les unités lexicales de manière suffisante : les rapprochements syntaxiques peuvent révéler des similitudes purement formelles et masquer la différence de nature entre les types conceptuels

en jeu dans ces structures. Dans la figure 7.21, les prototypes construits (représentant des noms) sont regroupés en sous-familles en tenant compte des arbres élémentaires associés à ces prototypes de mots (par délégation). Les prototypes d'une sous-famille délèguent les mêmes arbres élémentaires à un ou plusieurs pôles de comportements partagés.

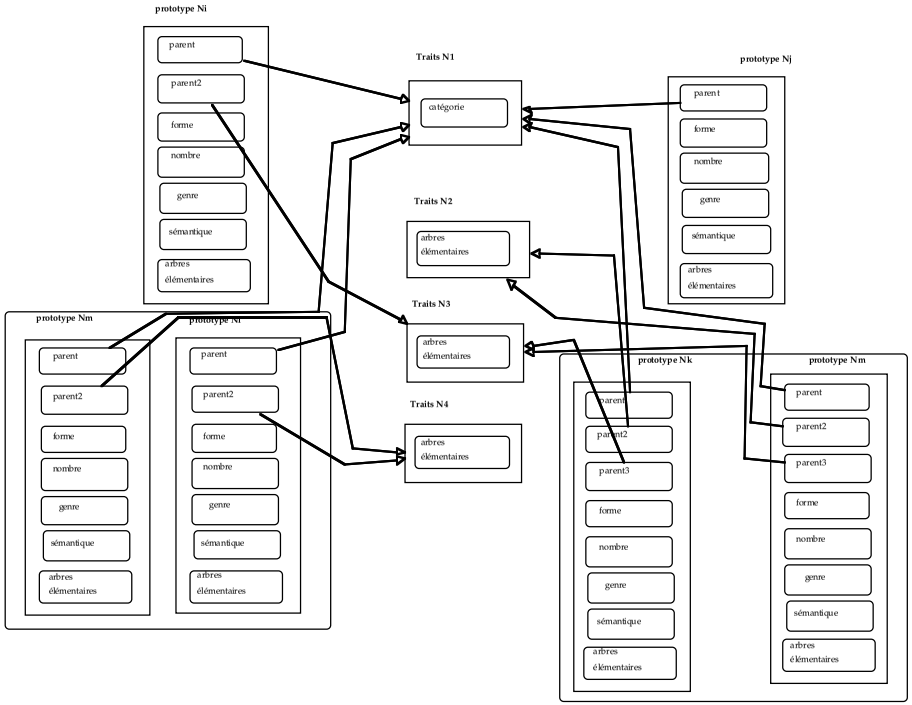


Figure 7.21 : Un réseau de comportements partagés.

## Difficultés techniques

La mise en place d'un tel réseau de familles de prototypes qui partagent ensemble des arbres élémentaires ne pose pas de problèmes techniques insurmontables dans la réalisation de ce réseau (-> annexes partie 2 chapitres 7 et 8 : traces complètes de la génération des prototypes et de la construction d'un réseau de pôles de comportements partagés). Il est facile de repérer sur un ensemble de prototypes les comportements communs à certains d'entre eux puis de définir les objets et les liens qui vont unir les objets et leurs pôles de comportements partagés. **L'automatisation du classement et la pertinence de cette automatisation posent davantage des problèmes linguistiques et théoriques (classification automatique) que techniques**. Repérer des similitudes formelles entre les représentations créées est une tâche à la portée de la programmation à prototypes. Le classement obtenu est une amorce de classement sur lequel un travail d'interprétation reste à faire: **Evaluer ces similitudes formelles et interpréter les regroupements restent des tâches auxquelles doit être confronté le linguiste dans la mesure où il est le seul à pouvoir y apporter une réponse précise**.

## Difficultés linguistiques





La difficulté est donc principalement d'ordre linguistique. Le classement obtenu va mettre un peu d'ordre là où, au départ, il n'y en avait pas. On va organiser les différentes représentations d'unités lexicales sur la base des comportements syntaxiques communs à certaines d'entre elles. Ces pôles de savoirs partagés restent malgré tout à affiner. Le degré de précision du classement initial des unités lexicales sera directement dépendant de la somme des savoirs qui sera utilisée pour opérer ce classement initial. Si on ne prend appui que sur des informations syntaxiques, on retrouvera des problèmes déjà soulignés (les similitudes formelles masquent souvent des interprétations de différentes natures).

### 7.6.3.4. Intervenir manuellement pour interpréter les regroupements

Enfin, si on peut automatiser le classement des prototypes lexicaux sur la base des comportements qui leurs sont associés, les résultats obtenus restent à qualifier, à nommer.

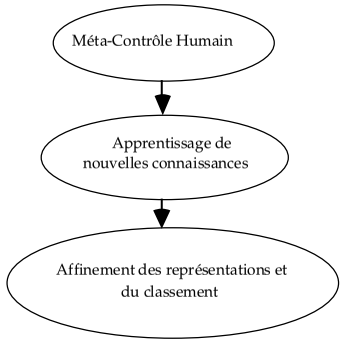


Figure 7.22 : Un méta-contrôle humain pour nommer les choses

Si le classement rapproche des mots comme `"absence"` ou `"ensemble"` sur des comportements identiques, cette proximité comportementale n'est pas identifiée comme étant propre à celle des noms prédicatifs. C'est en examinant à la main ce type de rapprochement que l'on pourra leur donner un nom c'est à dire nommer les choses.

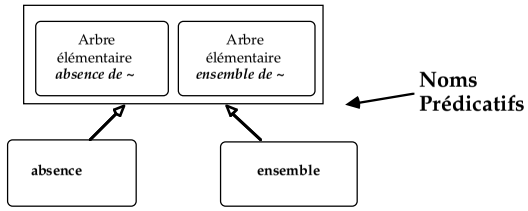


Figure 7.23 : absence-ensemble

D'une part il est nécessaire de définir les informations pertinentes pour amorcer le classement, d'autre part il convient de ne pas perdre de vue ces critères initiaux qui permettent d'obtenir ce

premier classement afin de ne pas sur-estimer ce classement. Il s'agit au contraire de considérer que le résultat obtenu reste en deça de ce que l'on souhaite mettre en oeuvre, l'écart résultant des carences inévitables que le traitement automatique met en avant entre la complexité des informations à représenter et notre incapacité à saisir cette complexité dans sa globalité.





# 7.7. Vers une classification dynamique : bilans et perspectives

## 7.7.1. La PàP : une nouvelle approche pour le TALN?

La PàP remet en question la notion de représentation vue comme un processus qui ne peut représenter qu'un monde prédéfini. Ce processus de représentation construit progressivement les entités informatiques suivant les connaissances dont on dispose sur un domaine visé : ce processus de représentation peut donc être défini de manière continue (dans le temps) en tenant compte des informations disponibles pour affiner les structures déjà construites sans avoir à reconstruire entièrement de nouvelles structures.

- PàP et Sous-Langages

*"J.McNaught (1993) présente une critique sévère de la tendance à vouloir développer des systèmes généralistes, fondés sur des grammaires et des lexiques généraux, que l'on entend spécialiser par ajouts d'éléments lexicaux et de règles"* (Péry-Wooodley 1995). A l'image de ce que propose McNaught, notre démarche de représentation est ascendante. Elle vise à construire des niveaux de généralisation à partir de descriptions qui s'appuient sur l'analyse de corpus de textes spécialisés.

- Classement dynamique et généralisation : Mettre à jour les savoirs communs puis construire des généralisations.

Le processus de représentation peut se dérouler en créant "à la volée" (manuellement ou automatiquement) les objets qui vont porter les savoirs disponibles. Il reste possible ensuite de réorganiser les objets construits si des savoirs communs existent.

L'automatisation de la phase de généralisation ou de classement reste malgré tout délicate. Il faut disposer d'un méta-regard sur les entités créées pour y détecter des analogies de structures ou de comportements. Cette méta-analyse d'une domaine de connaissances représentées avec la PàP est potentiellement disponible avec Self qui dispose d'outils qui agissent à un méta-niveau et qui permettent par exemple de selectionner tous les objets construits avec certains types d'attributs.

- Héritage dynamique

La délégation est un mécanisme qui permet de définir des liens d'héritage entre les objets construits : ces liens peuvent être définis dynamiquement; il est possible à tout moment de définir ou de reconfigurer des liens de délégation entre objets (ajout, retrait, modification).

## 7.7.2. Un dispositif expérimental guidé par le linguiste

- Dans l'état actuel du classement effectué, une intervention manuelle est semble-t-il indispensable pour ce travail d'interprétation en amont et en aval de la phase de représentation des structures représentées : le système construit s'inscrit dans une approche qui vise à établir un dialogue entre le dispositif informatique construit et les problèmes posés par les faits linguistiques étudiés. L'intervention manuelle du linguiste s'inscrit parfaitement dans une telle approche. Dans la mesure où les dispositifs informatiques (et

en particulier le nôtre) ne sont pas encore capables de résoudre tous les problèmes posés par le TALN, un dispositif informatique qui permet une interaction dynamique entre l'utilisateur privilégié de ce type d'outils et les résultats produits ou les données traitées correspond au préalable définitoire que nous avions posé pour la mise au point d'un tel dispositif.

- Le classement à mettre en place est fortement dépendant des informations reçues en amont. Il reste donc indispensable de s'interroger sur la nature des informations linguistiques qui seront manipulées pour ce classement. Comme on l'a dit précédemment, les informations recueillies actuellement en amont doivent être renforcées et affinées. Notre travail doit donc préciser quelle est la nature des informations à traiter et suivant les informations manipulées, quel type de classement on obtient, et quel type de classement on veut ou on peut obtenir sur la base de ces informations. Il s'agit donc de mettre en avant les limites du classement automatique obtenu suivant les critères de classement retenus. En précisant les limites de notre travail, d'une part on ne surestime pas les résultats obtenus, d'autre part, on garde en point de mire la trace de ce qu'il reste à faire.

- Les limites de l'automatisation marquent le champ de travail qu'il reste à effectuer manuellement pour mener à bien le classement des unités lexicales. Il semble clair que ce travail d'affinement des représentations obtenues va éclairer et guider la mise au point des éléments de savoirs pertinents que l'on souhaite mettre à jour pour aller vers une automatisation de plus en plus fine du classement escompté. Cette démarche qui vise à définir des outils de classement et les modèles de représentation par touches successives s'accommode assez bien avec le modèle de représentation des connaissances retenu. La PàP encourage une approche de représentation faite de petits sauts successifs qui améliore la qualité de la représentation produite. D'autre part, cette démarche s'accorde aussi avec la nécessaire approche artisanale que constitue le travail du linguiste dans sa volonté de décrire les comportements des faits de langue.





# Incise Chapitre 7

## Incise 1. Self et l'héritage

### Self permet l'héritage multiple et l'héritage dynamique.

On revient ici sur la présentation de l'héritage avec Self et notamment sur la notion de délégation. On construit des hiérarchies locales de contraintes syntaxiques partagées par un ensemble de prototypes lexicaux. Il est donc nécessaire de pouvoir retrouver et utiliser les liens de délégation définis pour adresser les messages pertinents aux objets adéquats. En effet, les processus de génération et de classement mettent automatiquement en place des liens de délégation entre les objets. Ces liens de délégation ne sont pas figés de manière définitive : le processus de classement ajoute par exemple un lien de délégation entre une sous famille de mots et un pôle de comportements partagés. Il est donc important de pouvoir sélectionner le destinataire d'un message si un objet donné délègue des comportements à différents parents : un mot peut déléguer des comportements (des arbres) à plusieurs parents, il convient donc de pouvoir retrouver les différents chemins possibles pour récupérer le comportement adéquat.

### I.1.1. Héritage non-ordonné

Self utilise une notion d'héritage multiple désordonné et reconfigurable.

- Il n'est pas possible depuis la version 3.0 de Self d'affecter aux parents d'un objet donné un rang de priorité. Avant cette version, il était en effet possible d'ordonner les liens de délégation. Ce changement de perspective oblige à une attention particulière dans la définition des liens de délégation si l'on ne veut pas produire de l'ambiguïté.

- En effet deux parents directs d'un objet donné ont un même rang de priorité; ils ne sont donc pas ordonnés hiérarchiquement, et sont donc accessibles l'un et l'autre sans préférence. On peut donc facilement générer des messages ambigüs si un ou plusieurs attributs de ces parents portent les mêmes noms. Il peut donc sembler nécessaire de savoir précisement à qui on souhaite adresser un message si l'on veut être sûr de ne pas produire ce type d'erreur.

- On verra cependant infra que les méta-processus disponibles avec Self fournissent des outils pour une description précise des prototypes construits. On peut ainsi sélectionner les objets à qui on adresse un message de manière non ambigue.

### I.1.2. Héritage multiple

Self ne restreint pas le nombre de lien de délégation entre les objets. Dans la mesure où il possible d'ajuster dynamiquement les prototypes (ajout ou retrait d'attribut), on peut à tout moment construire des liens de délégation.

## I.1.3. Règles de précédence dans l'héritage

L'héritage multiple à la Self s'appuie sur les règles suivantes :

- Si un objet et l'un de ces ancêtres définissent un attribut avec le même nom, c'est l'attribut de l'objet qui sera pris en compte.
- Si deux parents de ce même objet définissent un attribut avec le même nom, Self génère un message d'erreur d'accès à l'attribut (`messageAmbiguous`).
- "Sender Path Tiebreaker Rule" : Les règles d'héritage de Self spécifient que si deux attributs avec le même nom sont définis dans des parents de même rang de priorité, mais un seul de ces parents possède un lien d'héritage avec l'objet qui contient la méthode qui a émis le message, alors c'est l'attribut de ce parent qui est pris en compte.





## I.1.4. Renvoi (dirigés) de messages

### I.1.4.1. Receveur implicite

On dispose aussi en Self d'un mécanisme de renvoi de message qui permet de diriger un message à un et seul parent d'un objet donné au lieu de l'adresser à l'ensemble. Ce renvoi dirigé peut être utilisé pour résoudre des problèmes d'ambigüité entre les différents parents possibles d'un objet donné. Syntactiquement, ce message de renvoi est spécifié en préfixant le nom du message avec le nom de l'attribut du parent auquel ce renvoi est dirigé. Ce mécanisme est analogue au mécanisme général de renvoi utilisé par Self. Ce dernier permet à une méthode d'invoquer l'appel à un attribut couvert (du fait de la précédence des attributs) par l'attribut auquel est adressé le message : le renvoi de message est écrit avec un receveur implicite (ce mécanisme est similaire au `call-next-method` de CLOS (Habert 1996)).

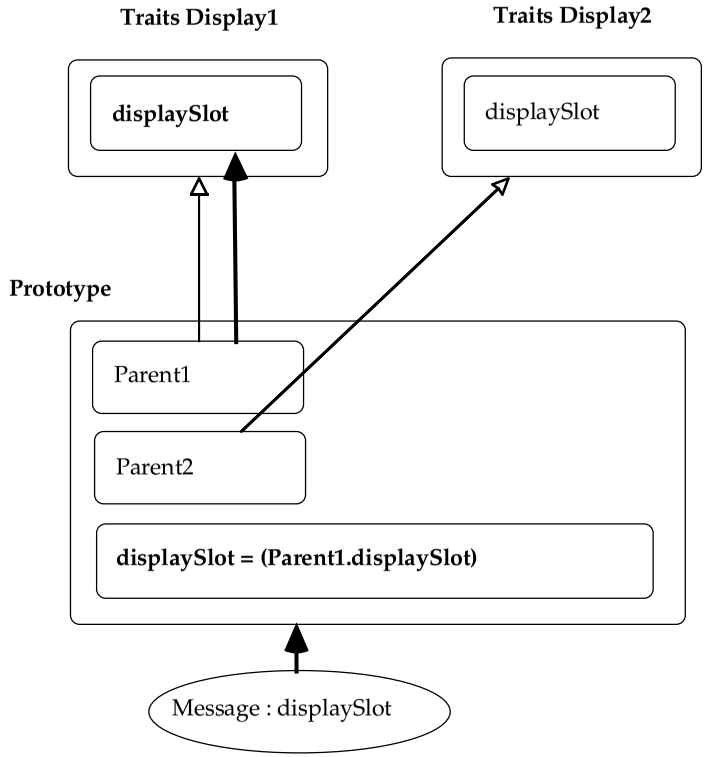


Figure 7.24 : Receveur implicite.

### I.1.4.2. Receveur explicite

On dispose aussi avec Self de primitives qui permettent de "forcer" l'envoi d'un message à un parent donné en utilisant par exemple la syntaxe suivante :

```
'message' sendTo: Objet DelegatingTo: parentChoisi
```

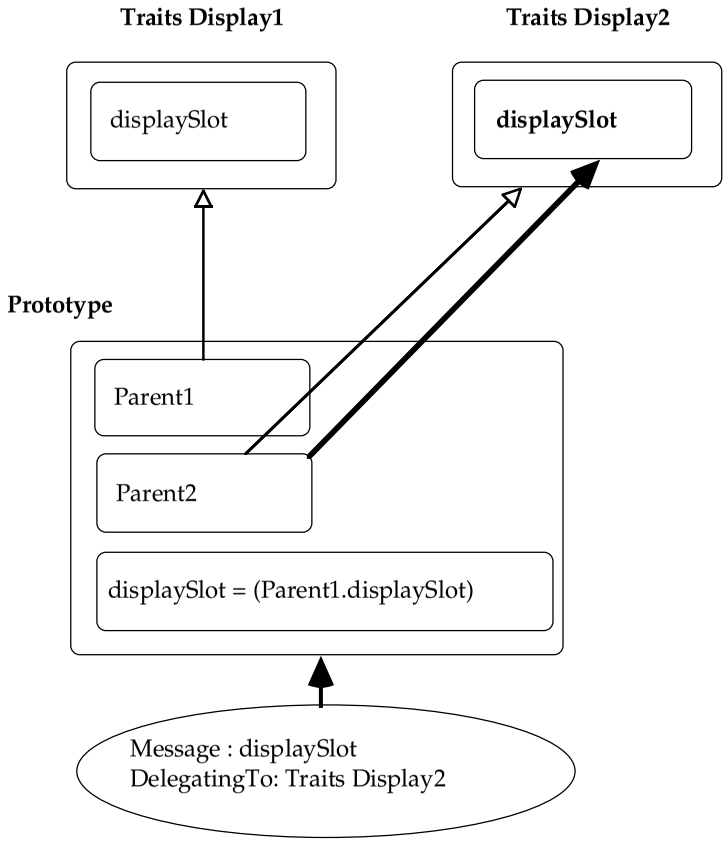


Figure 7.25 : Receveur explicite.





# Incise 2. Positionnement par rapport à l'approche probabiliste

> *"Le pessimisme qu'il pratiquait avait toujours été de cette variété vigoureuse que l'on trouve en Italie : comme Machiavel, il plaçait l'equilibre entre virtus et fortuna aux alentours de fifty/fifty; or voilà que les équations introduisaient un élément aléatoire qui donnait un taux de probabilité si indéterminé, si indicible, qu'il craignait même de le calculer."*
> Thomas Pynchon, "L'homme qui apprenait lentement (Entropie)", Ed. du Seuil.

## I.2.1. Probabilisation des modèles syntaxiques (Rajman 1995)

### I.2.1.1. Approche probabiliste pour l'analyse syntaxique

La production de modèles syntaxiques généraux pour les langues naturelles se révèle souvent être une tâche délicate voire difficile à réaliser de manière totalement satisfaisante. La production de ces modèles se heurte, par exemple, à des problèmes de couverture linguistique (permissivité trop grande ou trop restrictive) ou d'ambiguïté d'analyse (ce qui peut conduire à une extension considérable de structures syntaxiques particulières...). Pour pallier ces difficultés, la probabilisation des modèles syntaxiques vise à remplacer lors de la définition de la mise en correspondance entre séquences de mots et structures syntaxiques, la relation d'association binaire définie par tout modèle syntaxique sur l'ensemble des couples (séquences de mots, structure syntaxique) par une probabilité p définie sur ce même ensemble.
L'association entre une séquence W et une structure s n'est plus alors caractérisée en termes booléens mais mesurée par la probabilité conditionnelle suivante :

$$P(s/W) = p(W,s) \;/\; \sum_u p(W,u)$$

Cette approche permet de dissocier l'étape de détermination des structures syntaxiques potentiellement associables avec une séquence donnée et l'étape d'évaluation, pour chacune des structures candidates, de la qualité de l'association effective. Cette approche vise aussi à soulager la phase de construction de modèle syntaxique en laissant cette modélisation relativement simple en ce qui concerne la détermination des structures candidates; elle permet aussi réaliser une adéquation fine de la mise en correspondance par une adaptation bien choisie des probabilités. De plus, si l'on travaille sur des corpus, la détermination des probabilité à utiliser peut s'effectuer à partir de décomptes fréquentiels d'occurrences réalisées dans ces corpus.

### I.2.1.2. Schéma Global de probabilisation d'un modèle syntaxique

- Phase 1 : détermination des séquences incorrectes.

$$\forall W,\ W \text{ syntaxiquement incorrecte ssi } \forall s,\ P(s/W) = 0$$

i.e la probabilité d'une mise en correspondance avec une structure syntaxique quelconque est strictement nulle.

- Phase 2 : probabilités des séquences acceptables.

Pour obtenir une forme utilisable pour l'expression des probabilités conditionnelles associées aux séquences syntaxiquement correctes, il est nécessaire de postuler des prpriétés supplémentaires pour le modèle syntaxique. Le plus souvent, on utilise des modèles paramétriques, où l'on fait l'hypothèse de l'existence, pour la probabilité P(s,W) d'une expression fonctionnelle de la forme :

$$P_{\theta}(s,W) = F(\theta(s,W),...,\theta_n(s,W))$$

où : $\theta = (\theta(s,W),..,\theta_n(s,W))$ représente les paramètres du modèle.

La mise en oeuvre du modèle probabiliste passe par la résolution de deux problèmes principaux :

- Apprentissage des paramètres du modèle : Ce qui revient à déterminer, à partir des données et des connaissances disponibles, les valeurs des paramètres du modèle "optimal".

- Evaluation des structures syntaxiques associées à une séquence donnée : Il faut en effet déterminer, pour toute séquence de mots W syntaxiquement correcte, l'ensemble des structures associées qui maximisent la probabilité conditionnelle P(s/W).

## I.2.1.3. Prototypes et Comportements vs. Approche Probabiliste

Les techniques probabilistes appliquées à l'analyse syntaxique visent à concentrer l'attention sur la fréquence des mots ou des structures syntaxiques en réduisant l'impact des mots ou des structures à faible fréquence (assimilables à du bruit). Cette approche s'éloigne, sur ce dernier point en tout cas, de ce que nous cherchons à mettre en lumière. Notre travail s'attache à prendre en compte les savoirs de manière inductive et à affiner les différentes étapes de représentation pour tenir compte des évolutions potentielles des choses représentées ou de pouvoir prendre en compte de nouveaux éléments à représenter. Il semble donc délicat voir incongru de vouloir pré-affirmer le poids de savoirs non encore complètement connus et/ou représentés. Nous devons au contraire traiter et enregister les savoirs comme des entités de même "niveau de validité"; on se place ici d'ailleurs dans une approche voisine de celle de (Bensch & Savitch 1995).