La Sphère sémantique 2

IEML pour ingénieurs

Pierre Lévy

27 janvier 2016

version 2

Table des matières

Chapitre 1

Degrés de complexité

1.1 Vue d'ensemble

1.1.1 Les objets

En IEML, les objets se rangent en trois degrés de complexité, chaque degré de complexité étant lui-même structuré par un emboîtement récursif d'étages. Chaque degré de complexité s'organise autour d'un type d'objet fondamental:

- *Les termes* du dictionnaire pour le premier degré de complexité,
- *Les propositions* pour le second degré de complexité
- *Les USLs* (ensembles de propositions) pour le troisième et dernier degré de complexité. L'USL correspond intuitivement aux notions de message ou d'énoncé en IEML. On peut penser aux USLs comme à des *textes* ou *notes* catégorisant des données.

1.1.2 Les opérations

L'algèbre IEML utilise trois types d'opérations.

- Une **multiplication** ternaire non-commutative entre objets du même étage dans le même degré de complexité construit des *objets*.
  - La multiplication est récursive mais le nombre de récursions est limité.
  - Les *trois rôles* de la multiplication IEML sont, dans l'ordre : la substance, l'attribut, le mode.
  - Les variables qui remplissent ces rôles sont appelées des *sèmes*.
- Une **addition** commutative entre objets du même étage dans le même degré de complexité construit des *ensembles d'objets*.
- Une opération **d'enchâssement** d'un USL dans une proposition (qui n'existe donc qu'au au troisième degré de complexité) construit les strates successives des USLs. L'enchâssement est récursif mais le nombre de récursions est limité.

1.1.3 Le script

L'écriture script assure que chaque objet et chaque ensemble d'objets soit représenté par une notation et une seule et que deux notations différentes ne puissent pas représenter le même objet. L'écriture script comporte notamment des abréviations obligatoires et un ordre obligatoire des variables additives.

1.2 Termes étagés en couches

Les **termes** du dictionnaire sont construits récursivement à partir de **primitives** {E, U, A, S, B, T} auxquelles s'appliquent deux opérations:

— *La multiplication* de l'algèbre IEML *concatène les primitives en séquences ternaires.* Chaque récursion de la multiplication élève le résultat à une *couche* supérieure.

— *L'addition* construit des ensembles de séquences de même couche.

1.3 Propositions étagées en niveaux

Les **propositions** sont construites récursivement à partir des **termes** du dictionnaire. Les trois principaux étages de ce degré de complexité sémantique sont : les mots, les phrases et les super-phrases.

— *La multiplication* construit les mots (relation entre deux morphèmes), les clauses (relation entre deux mots) et les super-clauses (relation entre deux phrases).

— *L'addition* construit les morphèmes à partir des termes, les phrases à partir des clauses et les super-phrases à partir des super-clauses.

1.4 USLs étagés en strates

Les **USLs** (ou textes, ou notes) sont construits récursivement à partir des **propositions**.

— *L'addition* ajoute des propositions (quelque soit leur niveau) à un USL.

— *L'enchâssement* insère un USL dans une proposition incluse dans un USL. L'enchâssement correspond intuitivement à une note ou référence. L'USL enchâssé peut à son tour enchâsser des USLs dans ses propositions, et ainsi de suite récursivement. On dit que l'USL enchâssé se situe dans la *strate* hypertextuelle immédiatement inférieure à celle de l'USL enchâssant. Le nombre de strate est *limité*, de telle sorte que la plus basse strate ne contienne aucun USL enchâssé et puisse être décodée uniquement en fonction des termes du dictionnaire.

Chapitre 2

Les termes

2.1 L'objet terme

2.1.1 L'algèbre IEML au premier degré de complexité

1. L'algèbre IEML manipule les éléments d'un alphabet de primitives.
2. **La multiplication** de l'algèbre IEML *concatène récursivement les primitives en séquences ternaires.*
3. Une séquence de primitives produite en utilisant uniquement la multiplication est appelée une *séquence singulière.*
4. **L'addition** de l'algèbre IEML réunit des *ensembles de séquences singulières de même couche* (c'est-à-dire de même longueur) *ou catégories IEML.*

2.1.2 Scripts de termes

— Un script de terme note une fonction de l'algèbre IEML de premier degré.
— Un script de terme note, en même temps qu'une fonction, son résultat, c'est-à-dire un ensemble de séquences singulières de même couche (une catégorie IEML).
— Un script de terme ne comporte que les caractères suivants :
  — + note l'addition
  — **: . - ' , _ ;** notent les multiplications respectivement aux couches 0, 1, 2, 3, 4, 5, 6
  — **E U A S B T** notent les primitives
  — **O M F I** notent des additions remarquables de primitives
  — **wo wa wu we y o e u a i j g h c p x** notent des multiplications remarquables de couche 1
— Le script de termes ne comporte ni parenthèse ni crochet.
— Pour plus de détails on se reportera à la section 2.2 de la grammaire d'IEML.

2.1.3 Termes

— SI un script est couplé à un descripteur unique dans la langue naturelle LN.
— SI toutes les séquences singulières que génère le script sont couplés à un descripteur d unique dans la langue naturelle LN
— ALORS le script est un terme A

1. Aussi bien le terme A que le descripteur d_{LN} sont des séquences de caractères, mais A est codé en IEML alors que d est codé en langue naturelle.
2. Du point de vue de la langue naturelle, A est le signifiant et d le signifié. Du point de vue d'IEML, d est le signifiant et A le signifié.
3. Un dictionnaire IEML comprend un ensemble de termes dont chacun est couplé à n descripteurs, n étant étant le nombre de langues naturelles supporté par le dictionnaire. Pour la même langue naturelle LN, chaque descripteur est unique.
4. Un *terme* renvoie donc à un ensemble de couples (A, d_{LN1}), (A, d_{LN2}), (A, d_{LNn}) appartenant au même dictionnaire.

2.1.4 Classe d'un terme

— Tous les termes ont une classe : auxiliaire, verbe ou nom.

1. Dans **un terme multiplicatif**, l'initiale (substance de la substance, etc.) E détermine un auxiliaire, l'initiale O un verbe et l'initiale M un nom. Si la substance de la substance (etc.) est une addition, on applique la règle suivante:
   E+O⇒O
   E+M⇒M
   O+M⇒M.
   Par exemple, E:+U:+B: est considéré comme un terme nominal.
2. Les mêmes règles sont appliquées pour déterminer la classe d'un **terme additif**, par exemple lorsqu'il résulte de l'addition d'une multiplication dont l'initiale est O et d'un script dont l'initiale est M.

2.2 Relation d'inclusion entre termes

2.2.1 Généralités

1. Les termes de la langue IEML étant des scripts et les scripts étant des notations de l'algèbre IEML, les relations entre les termes d'un dictionnaire IEML peuvent être calculées automatiquement.
2. Les relations entre les descripteurs de même langue naturelle peuvent être calculées automatiquement à partir des relations entre les termes IEML qu'ils décrivent.

3. Toutes les relations entre termes concernent des *relations entre termes d'un même dictionnaire.*

2.2.2 La relation d'inclusion

— SI l'ensemble de séquences singulières de même couche généré par le terme A est un *sous-ensemble* de l'ensemble de séquences singulières généré par le terme C.
— ALORS A est inclus dans C

Deux termes ne peuvent être en relation d'inclusion que s'ils sont à la même couche.

— Exemple 1: *U:M:.A:M:.-** est inclus dans *O:M:.O:M:.-**
— Exemple 2: *O:.** n'est pas inclus dans *U:.+A:F:.**
  *U:.+A:F:.** génère l'ensemble de séquences singulières qui suit:
  — U:.
  — A:U:.
  — A:A:.
  — A:S:.
  — A:B:.
  — A:T:.
— Exemple 3: *O:.** est inclus dans *U:.+A:I:.**
  *U:.+A:I:.** génère l'ensemble de séquences singulières qui suit:
  — U:.
  — A:.
  — A:U:.
  — A:A:.
  — A:S:.
  — A:B:.
  — A:T:.

U:.+A:. = O:. Donc O:. est incluse dans U:.+A:I:.

2.2.3 Paradigmes

Un terme paradigmatique - ou paradigme - est un terme choisi par convention par les auteurs d'un dictionnaire IEML.

— **Un paradigme génère un ensemble de séquences singulières n'ayant aucune intersection avec les ensembles de séquences singulières générés par les autres paradigmes du même dictionnaire.**
  Autrement dit, bien que choisis « arbitrairement », les paradigmes doivent satisfaire à la condition de générer des ensembles mutuellement exclusifs de séquences singulières de même couche.

1. Un terme d'un dictionnaire est inclus dans un paradigme et un seul.
2. Il n'existe pas de terme d'un dictionnaire qui ne soit inclus dans aucun paradigme.

3. Puisque les paradigmes ont des intersections vides et qu'un terme est inclus dans un paradigme et un seul, alors il n'existe de relation d'inclusion entre un terme A et un terme C que si A et C sont inclus dans le même paradigme.

2.2.4 Tables

— **Une table est un terme de couche l dont le script note *une seule multiplication* de couche l, multiplication qui produit *plusieurs* séquences singulières.**

1. Dans le dictionnaire, une table se présente comme une matrice à 1, 2 ou 3 axes, ces axes correspondant à des variables multiplicatives de même rôle.
2. Une table ne peut être incluse que dans un seul paradigme.
3. Un paradigme peut inclure une ou plusieurs tables.
4. Une table peut inclure une ou plusieurs tables.

2.2.5 Têtes de rangées, de colonnes et de tabulations

Les tables incluent (respectivement et selon leur nombre d'axes) des têtes de rangées, des têtes de colonnes et des têtes de tabulations.

— **Le script d'un *terme d'en tête* code la somme des séquences singulières de sa rangée, de sa colonne ou de sa tabulation.**

2.2.6 Cellules

— **Une cellule est un terme dont le script génère une seule séquence singulière.**

1. Dans les tables, les cellules se trouvent à l'intersection d'une rangée, d'une colonne et d'une tabulation (matrice 3D), ou bien à l'intersection d'une rangée et une colonne (matrice 2D) ou bien sur une rangée (matrice 1D), selon le nombre des axes de la matrice multiplicative.
2. Une cellule peut être incluse dans plusieurs tables incluses dans le même paradigme.

2.3 Relation de descendance entre termes

2.3.1 Relation de descendance

L'inclusion d'un ensemble de séquences singulières de couche l dans un ensemble de séquences singulières de couche l est « ensembliste ». En revanche, la relation de descendance est « méréologique »: elle concerne *l'insertion* d'une sous-séquence dans une séquence.

— SI le script de C est inséré dans le script de A

— ALORS le terme A est un descendant du terme C

1. Si A est un descendant de C, alors C est un ascendant de A
   Si C est un ascendant de A, alors A est un descendant de C
2. Le terme ascendant est de couche inférieure au terme descendant.
3. Les termes en relation de descendance appartiennent à des paradigmes distincts.
4. La relation de descendance passe par un des trois sèmes (substance, attribut, mode).

Exemple 1

M:M:.a.-n.a.-f.o.-' est un descendant de:

— En substance:
  — M:M:.a.-
  — M:M:.
  — a.
  — M:
— En attribut:
  — n.a.-
  — n.
  — a.
— En mode:
  — f.o.-
  — f.
  — o.

2.3.2 Descendance et cardinalité des ensembles de séquences

1. Si le terme descendant génère *une seule séquence singulière*, alors le terme ascendant génère nécessairement *une seule séquence singulière*.
2. Si le terme ascendant génère *un ensemble de séquences singulières* (cardinal supérieur à zéro ou un), alors le terme descendant génère nécessairement *un ensemble de séquences singulières* (cardinal supérieur à zéro ou un).
3. Il est possible que le terme ascendant génère un ensemble de séquences singulières dont le cardinal est *plus petit* que l'ensemble de séquences singulières généré par le terme descendant, notamment si certains sèmes génèrent des ensembles de séquences plus petits que leur produit ou si les sèmes et sous-sèmes du terme descendant sont composés au moyen d'additions.

Exemple 2

*e.n.o.-+a.n.o.-b.M:M:.-'** (27 séquences de couche 4) est un descendant de *e.n.o.-** (une séquence de couche 3) et de *a.n.o.-** (une séquence de couche 3) en substance.

2.4 Relation de germains entre termes

La « germanité » correspond intuitivement à une relation de fraternité, sororité ou cousinage. Les germains sont de même génération et ils ont une ascendance commune. Germain se traduit par « *sibling* » en anglais.

2.4.1 Germains ordinaires

— SI A et C sont inclus dans le même paradigme
— SI A et C génèrent des ensembles de séquences singulières de même cardinalité
— SI les ensembles de séquences singulières générés par A et C ont une intersection vide.
— ALORS A et C sont germains

1. Si A est germain de C alors C est germain de A (symétrie)
2. Les relations de germains ordinaires restent tacites dans le dictionnaire, qui ne marque que les relations entre germains remarquables.

2.4.2 Germains remarquables

Il existe plusieurs types de *germains remarquables*, dont la liste qui suit est ouverte. En plus d'être germains, les termes en relation de germains remarquables doivent respecter des conditions supplémentaires.

2.4.2.1 Germains opposés

— SI deux termes A et C sont germains
— SI la substance et l'attribut de A sont distincts
— SI la substance de A est égale à l'attribut de C
— SI l'attribut de A est égale à la substance de C
— ALORS A et B sont en relation de *germains opposé*s.

2.4.2.2 Germains associés

— SI deux termes A et C sont germains
— SI la substance de A et la substance de C sont identiques
— SI l'attribut de A et l'attribut de C sont identiques
— SI le mode de A et le mode de C sont distincts
— ALORS A et C sont en relation de *germains associés*.

2.4.2.3 Germains jumeaux

— SI deux termes A et C sont germains
— SI la substance de A et l'attribut de A sont identiques
— SI la substance de C et l'attribut de C sont identiques
— ALORS A et B sont en relation de *germains jumeaux*.

2.4.2.4 Germains croisés

— SI deux termes A et C sont germains
— SI la substance de A est le germain opposé de la substance de C
— SI l'attribut de A est le germain opposé de l'attribut de C
— ALORS A et C sont en relation de *germains croisés*

2.5 Inhibition des relations

Les relations de descendance et de germanité peuvent être inhibées au niveau des paradigmes. Les relations d'inclusion ne peuvent pas être inhibées.

2.5.1 Inhibition des relations de descendance

L'éditeur du dictionnaire permet d'inhiber les relations de descendance pour un, deux ou trois des rôles :

1. substance
2. attribut
3. mode

2.5.2 Inhibition des relations « germain remarquable »

L'éditeur du dictionnaire permet d'inhiber les relations de germains remarquables au niveau d'un paradigme.

1. G. associés
2. G. opposés
3. G. jumeaux
4. G. croisés
5. etc.

2.6 Définitions de termes

Tous les termes d'un dictionnaire sont définis. La définition comprend, dans l'ordre : un en-tête, une liste de tables incluses, une liste de tables incluantes, une liste de germains remarquables, une liste d'ascendants, une liste de descendants.

2.6.1 En-tête de la définition

1. Le terme défini lui-même
2. La classe du terme défini (auxiliaire, verbe ou nom, voir la section: 2.1.4)
3. La couche du terme défini
4. Le paradigme dans lequel le terme défini est inclus
5. La cardinalité du terme défini (la quantité de séquences singulières incluses)

2.6.2 Tables incluses

La liste des tables qui sont incluses dans le terme défini, par ordre de cardinalité décroissante et par ordre du script pour les tables de même cardinalité.

2.6.3 Tables incluantes

La liste des tables qui incluent le terme défini, par ordre de cardinalité croissante et par ordre du script pour les tables de même cardinalité.

2.6.4 Germains remarquables

Si les relations ne sont pas inhibées:

1. Germains associés
2. Germains opposés
3. Germains jumeaux
4. Germains croisés
5. etc.

2.6.5 Ascendants

Si les relations ne sont pas inhibées...

2.6.5.1 Ascendants en substance

Par ordre de couche décroissante et par ordre du script pour les ascendants de même couche.

2.6.5.2 Ascendants en attribut

Par ordre de couche décroissante et par ordre du script pour les ascendants de même couche.

2.6.5.3 Ascendants en mode

Par ordre de couche décroissante et par ordre du script pour les ascendants de même couche.

2.6.6 Descendants

2.6.6.1 Descendants en substance

Par ordre de couche croissante et par ordre du script pour les descendants de même couche.

2.6.6.2 Descendants en attribut

Par ordre de couche croissante et par ordre du script pour les descendants de même couche.

2.6.6.3 Descendants en mode

Par ordre de couche croissante et par ordre du script pour les descendants de même couche.

2.7 Récapitulation

2.7.1 Liste des objets

— Sème 2.1.1 1, 2, 3, 4
— Séquence singulière 2.1.1 5
— Ensemble de séquences singulières de même couche 2.1.1 6
— Script 2.1.2
— Descripteur 2.1.3
— Terme 2.1.3
— Dictionnaire 2.1.3 1, 2, 3, 4
— Paradigme 2.2.3
— Table 2.2.4
— En-tête 2.6.1
— Cellule 2.2.6

2.7.2 Résumé des principales relations entre objets

— Les sèmes sont les variables multiplicatives de l'algèbre IEML.
— La multiplication IEML construit des séquences singulières.
— L'addition IEML construit des ensembles de séquences singulières de même couche.
— Les scripts codent des ensembles de séquences singulières de même couche.
— Les termes sont des couples (script, descripteur).
— Les relations entre termes sont déterminées par les relations entre les ensembles de séquences de même couche que codent leurs scripts.
— Les paradigmes, tables, en-têtes et cellules sont des termes.
— Le dictionnaire inclut des paradigmes.
— Les paradigmes incluent des tables.
— Les tables incluent des en-têtes.
— Les en-têtes incluent des cellules.

2.7.3 Relations entre termes

— Inclusion (inclus / incluant) *ordre 2.2.2*
— Descendance (ascendant / descendant) *ordre 2.3.1*

- Germain (germain / germain) *symétrie 2.4.1*
  - Germain associé 2.4.2.2
  - Germain opposé 2.4.2.1
  - Germain jumeau 2.4.2.3
  - Germain croisé 2.4.2.4

Chapitre 3

Les propositions

3.1 L'objet mot

3.1.1 Le morphème

3.1.1.1 Définition du morphème

— SI un ou plusieurs termes sont additionnés
— SI les termes additionnés sont distincts
— SI les termes additionnés n'ont aucune intersection deux-à-deux
— SI les termes additionnés sont dans l'ordre du script
— ALORS l'addition est un morphème

1. L'addition peut ne contenir qu'un seul terme
2. Le nombre de termes est limité (12 termes, par exemple)

3.1.1.2 Morphème vide et morphème plein

— SI le seul terme additionné est E:
ALORS le morphème est **vide**
— SI le morphème n'est pas vide
ALORS il est **plein**

3.1.1.3 Détermination de la classe du morphème plein

— Un morphème plein doit être **auxiliaire** ou (exclusif) **verbal** ou (exclusif) **nominal**.
— Une addition de termes nominaux donne un morphème nominal
— Une addition de termes verbaux donne un morphème verbal
— Une addition de termes auxiliaires donne un morphème auxiliaire
— Lorsque l'addition a comme variables des termes appartenant à plus d'une classe, pour déterminer la classe du morphème, on applique les règles expliquées à la section 2.1.4.

3.1.1.4 Notation du morphème en script

— Les *premiers crochets* contiennent un terme et un seul. $[terme]$
— Un morphème se note $([terme_i] \oplus [terme_j] \oplus [terme_k]...)$
— Les *premières parenthèses* contiennent une addition de termes, c'est-à-dire un morphème.
— Un morphème commence par ([et se termine par])
— $([morphème])$

3.1.1.5 Interface pour la composition du morphème

L'utilisateur de l'éditeur choisit les termes à additionner pour composer un morphème. L'éditeur les ordonne, lui signale les intersections et détermine la classe grammaticale.

3.1.2 Le mot

3.1.2.1 Définition du mot

— SI trois morphèmes sont multipliés
— SI la substance est un morphème plein
— SI l'attribut est un morphème vide
— ALORS la multiplication est un mot

3.1.2.2 Signification des rôles multiplicatifs du mot

— Le morphème en rôle de substance représente la racine du mot.
— Le morphème en rôle de mode représente la flexion du mot.

3.1.2.3 Règles pour la détermination des propriétés des mots

— SI le morphème de mode est vide
ALORS c'est un **mot-morphème**.
— SI le morphème de mode est plein,
ALORS c'est un **mot fléchi**.
— Un mot est **auxiliaire**, **verbe** ou **nom** selon la classe de son morphème de substance.

3.1.2.4 Impossibilités syntaxiques concernant les mots

1. Un mot ne peut pas multiplier moins d'un morphème plein
2. Un mot ne peut pas multiplier plus de deux morphèmes pleins.
3. Il n'existe pas de mots vides.
4. *Il est impossible d'additionner des mots.*
On notera que les mots sont déjà composés par une (mots morphèmes) ou deux (mots fléchis) additions de termes.

3.1.2.5 Notation du mot en script

- Les *seconds crochets* contiennent un seul morphème *ou bien* une multiplication de deux morphèmes, c'est-à-dire un mot. La substance de la multiplication représente la racine du mot et le mode sa flexion.
- Un mot-morphème se note
  - $[([terme_i] \oplus [terme_j] \oplus [terme_k]...)]$
  - Il code un morphème plein en substance, un morphème vide en attribut et un morphème vide en mode.
- Un mot fléchi se note
  - $[([terme_i] \oplus [terme_j] \oplus [terme_k]...) \otimes ([terme_l] \oplus [terme_m] \oplus [terme_n]...)]$
  - Il code un morphème plein en substance, un morphème vide en attribut et un morphème plein en mode.
- Un mot commence par [([et se termine par])]
- $[([mot])]$

3.2 L'objet phrase

3.2.1 Les arbres syntagmatiques

3.2.1.1 Les arbres en général

« En théorie des graphes, un arbre enraciné - ou arborescence - est un graphe acyclique orienté possédant une unique racine, tel que tous les nœuds sauf la racine ont un unique parent. » (Wikipedia français : arbre enraciné). La même référence contient des définitions que je résume ici :

- les **feuilles** (ou nœuds externes), sont des éléments ne possèdent pas d'enfants dans l'arbre (« leaves or external nodes »).
- les **nœuds internes** sont des éléments possédant des **enfants** (« nodes »).
- La **racine** de l'arbre est l'unique nœud ne possédant pas de **parent** (« root »).
- Les nœuds sont reliés entre eux par une **arête** (« edge »).
- Selon le contexte, un **nœud** (« node ») peut désigner un nœud interne ou externe (feuille) de l'arbre.
- La **profondeur** d'un nœud est la distance, i.e. le nombre d'arêtes, de la racine au nœud.
- La **hauteur** d'un arbre est la plus grande profondeur d'une feuille de l'arbre.
- La **taille** d'un arbre est son nombre de nœuds (en comptant les feuilles ou non).
- La **longueur de cheminement** est la somme des profondeurs de chacune des feuilles.
- Un « niveau » d'un arbre est un ensemble de nœuds internes ou de feuilles situés à *la même profondeur* — on parle aussi de nœud ou de feuille *de même hauteur* dans l'arbre considéré. Pour éviter les confusions avec le « niveau » des propositions, on parlera ici d'une **génération** pour

désigner *l'ensemble de noeuds ou de feuilles de même profondeur* (ou de même hauteur).
— Un ensemble d'arbres est appelé une **forêt**.

3.2.1.2 Arbres de mots et de phrases

1. Une phrase est un arbre orienté de mots.
2. Une super-phrase est un arbre orienté de phrases.
3. Les phrases et super-phrases sont des arbres syntagmatiques.
4. Les arbres syntagmatiques sont codés en ajoutant des multiplications ternaires de même niveau (un graphe est un ensemble de triplets).
5. La multiplication code le lien parent-enfant dans un arbre.
   (a) La *substance* code le noeud *parent*.
   (b) *L'attribut* code le noeud *enfant*.
   (c) Le *mode* code *l'arête* entre le noeud parent et le noeud enfant. Le mode décrit la relation entre le noeud parent et le noeud enfant.
6. IEML adopte l'approche du **parcours en largeur** pour la description et l'exploration des arbres syntagmatiques (voir https://fr.wikipedia.org/wiki/Algorithme_de_parcour

3.2.2 La clause

3.2.2.1 Définition de la clause

— SI trois mots sont multipliés
— SI le mot substance et le mot attribut sont *distincts*
— ALORS la multiplication est une clause.

3.2.2.2 Signification des rôles multiplicatifs de la clause

— Conformément à la section 3.2.1.2 point 5, une clause représente une arête entre deux noeuds dans un arbre syntagmatique de niveau phrase.
  — Le mot en rôle de substance représente un noeud-parent.
  — Le mot en rôle d'attribut représente un noeud-enfant.
  — Le mot en rôle de mode représente une arête et code la relation entre le mot parent et le mot enfant.

3.2.2.3 Propriétés des clauses

— Une clause est **auxiliaire**, **verbale** ou **nominale** selon la classe de son mot en substance.

3.2.2.4 Impossibilité syntaxique

— Il n'existe pas de clause vide.
— La clause ne peut contenir moins ou plus de trois variables multiplicatives.

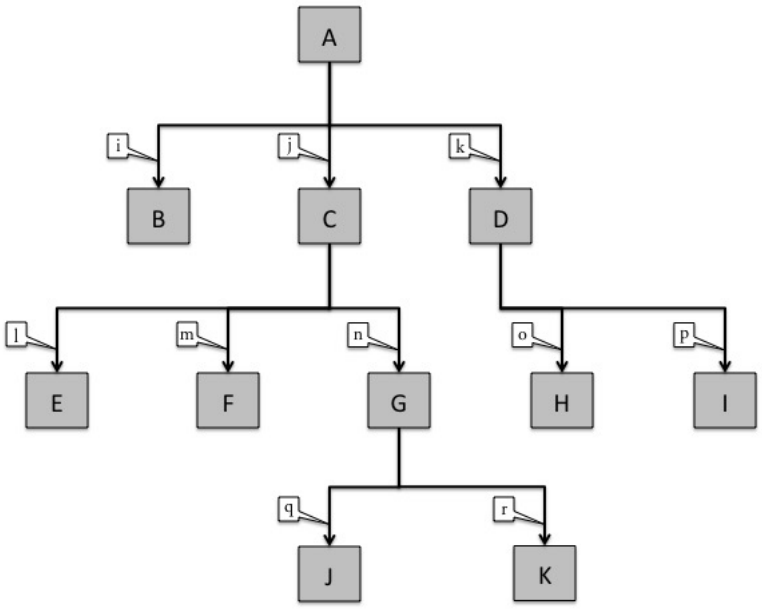

FIGURE 3.1 – Arbre syntagmatique

3.2.2.5 Notation de la clause en script

— Les *deuxièmes parenthèses* contiennent une multiplication de trois mots, c'est-à-dire une clause.
— La clause se note en script $([mot_i] \otimes [mot_j] \otimes [mot_k])$
— Une clause commence par ([[et se termine par)])
— ([([*clause*])])

3.2.3 La phrase

3.2.3.1 La phrase comme arbre de mots

La figure 3.1 représente l'arbre syntagmatique d'une phrase. Les carrés gris marqués de lettres capitales représentent les noeuds de l'arbre. Les carrés blancs marqués de lettres minuscules représentent les arêtes entre les noeuds. Aussi bien les arêtes que les noeuds sont des mots. A est la racine de l'arbre. B, C et D représentent la première génération de noeuds. E, F, G, H et I représentent la deuxième génération. J et K, représentent la troisième génération.

3.2.3.2 Ordre d'addition des clauses dans une phrase

Pour produire une phrase, on additionne des clauses en se conformant à l'ordre qui suit.

1. Allure générale de l'ordre
   (a) On suit d'abord l'ordre des *générations*, de haut en bas (c'est-à-dire de la racine vers les feuilles) ;
   (b) ensuite, à l'intérieur de la même génération, on suit l'ordre des *mots-parents*, de gauche à droite ;
   (c) finalement, les *mots-enfants* du même mot-parent sont rangés, de gauche à droite, dans l'ordre du script (grammaire IEML section 2.2.7), en fonction du premier terme de leur racine.
2. Identité et variation des substances et des attributs
   (a) Dans les clauses qui décrivent la première génération, la substance est identique et les attribuent varient. Les clauses sont rangées par l'ordre des premiers termes de leurs attributs.
   (b) La première variation de la substance marque le passage à la description de la deuxième génération. La première substance des clauses qui décrivent la deuxième génération correspond au premier attribut dans les clauses de première génération, et ainsi de suite, dans l'ordre des attributs de la première génération.
   (c) A l'intérieur des clauses de même substance de la deuxième génération, les attributs varient. Ces clauses de même substance sont rangées par l'ordre des premiers termes de leurs attributs...
   (d) Les mêmes règles s'appliquent récursivement de génération en génération.

Exemple La phrase de la figure 3.1 se note ainsi:
[(A⊗B⊗i)⊕(A⊗C⊗j)⊕(A⊗D⊗k)⊕(C⊗E⊗l)⊕(C⊗F⊗m)⊕
(C⊗G⊗n)⊕(D⊗H⊗o)⊕(D⊗I⊗p)⊕(G⊗J⊗q)⊕(G⊗K⊗r)]

3.2.3.3 Définition d'une phrase

— SI des clauses sont additionnées selon l'ordre décrit à la section 3.2.3.2 de sorte que l'addition décrive un arbre
— SI tous les mots-noeuds de cet arbre sont distincts
— ALORS l'addition de clauses est une phrase.

1. Les mots-arêtes de la même phrase ne sont pas nécessairement distincts.
2. La taille de l'arbre – le nombre de noeuds – est limitée (25 noeuds, par exemple).

3.2.3.4 Notation d'une phrase en script

— Les *troisièmes crochets* contiennent une addition de clauses, c'est à dire une phrase.
— Une phrase se note en script : $[(clause_i) \oplus (clause_j) \oplus (clause_k) \oplus ...]$
— Une phrase commence par [([[et se termine par)])]]
— [([([*phrase*])])]

3.2.4 Interface pour les phrases

L'utilisateur de l'éditeur construit un arbre de mots de manière « sensori-motrice » (glisser, déposer, cliquer, etc.). L'éditeur génère automatiquement la phrase en script qui correspond à l'arbre construit.

3.3 L'objet super-phrase

3.3.1 Super-clauses

Les super-clauses sont aux phrases ce que les clauses sont aux mots. Les règles pour les super-clauses sont les même que pour les clauses, avec cette seule différence que les super-clauses multiplient des phrases au lieu de multiplier des mots.

— Les *troisièmes parenthèses* contiennent une multiplication de phrases, c'est-à-dire une super-clause.
— Une super-clause se note : $([phrase_i] \otimes [phrase_j] \otimes [phrase_k])$
— Une super-clause commence par ([([[et se termine par)])])])
— ([([([*superclause*])])])

3.3.2 Super-phrases

On suit les même règles de construction (unicité des noeuds, ordre, etc.) pour les arbres de phrases que pour les arbres de mots.

Les règles pour les super phrases sont les même que pour les phrases, avec cette seule différence que les super-phrases additionnent des super clauses au lieu d'additionner des clauses.

— Les *quatrièmes crochets* contiennent une addition de clauses, c'est-à-dire une super-phrase.
— Une super-phrase se note : $[(superclause_i) \oplus (superclause_j) \oplus (superclause_k)...]$
— Une super-phrase commence par [([([[et se termine par)])])])]
— [([([([*superphrase*])])])]

L'éditeur génère automatiquement la super-phrase en script à partir de la construction « sensori-motrice » d'un arbre de phrases par l'utilisateur.

3.4 Propositions ouvertes et fermées

3.4.1 Synthèse des niveaux propositionnels

0 [Terme] _ entre crochets.

1 (Morphème) _ *addition* de termes.

2 [Mot] _ *multiplication* de morphèmes avec attribut vide.

3 (Clause) _ *multiplication* de mots.

4 [Phrase] _ *addition* de clauses.

5 (Super-clause) _ *multiplication* de phrases.

6 [Super-phrase] _ *addition* de super-clauses.

3.4.2 Synthèse des étages propositionnels

I [Mot] _ noeud de termes.

II [Phrase] _ arbre de mots.

III [Super-phrase] _ arbre de phrases.

3.4.3 Fermeture d'une proposition

Une proposition peut être *ouverte*, c'est-à-dire insérée dans une proposition de niveau supérieur, ou bien *fermée*, c'est-à-dire terminée et se tenant en quelque sorte debout toute seule. Les morphèmes, clauses et super-clauses appartiennent à des niveaux qui composent (respectivement) des mots, phrases et super-phrases. Seules les propositions appartenant aux trois étages des mots, des phrases et des super-phrases peuvent être fermées. Pour marquer la fermeture d'une proposition en script on place cette proposition entre deux slashes /.

- **Un morphème est toujours ouvert**.
  - $/([terme_i] \oplus [terme_j] \oplus [terme_k]...)/$est impossible.
- **Un mot est ouvert *ou* fermé**.
  - $/[(morphème_r) \otimes (morphème_f)]/$ et $/[(morphème)]/$ sont possibles.
- **Une clause est *toujours* ouverte**.
  - $/([mot_i] \otimes [mot_j] \otimes [mot_k])/$ est impossible.
- **Une phrase est ouverte *ou* fermée**.
  - $/[(clause_i) \oplus (clause_j) \oplus (clause_k)...]/$est possible.
  - Si une phrase ne contient qu'une seule clause, on écrit $/[(clause)]/$
- **Une super-clause est *toujours* ouverte**.
  - $/([phrase_i] \otimes [phrase_j] \otimes [phrase_k])/$est impossible.
- **Une super-phrase est *toujours* fermée**.
  - $/[(superclause_i) \oplus (superclause_j) \oplus (superclause_k)...]/$ est nécessaire.

On remarquera que seuls les mots, phrases et super-phrases, c'est-à-dire les trois « étages » *notés entre crochets* peuvent être fermés.

3.5 Le script pour les propositions (recapitulation)

— Les variables sont des termes IEML (termes du dictionnaire)
— Les opérateurs, ou signes de ponctuation sont :

⊕ addition de n (limité) variables dans l'ordre du script.

⊗ multiplication de trois variables

[initie les termes, les mots, les phrases, les super-phrases

] termine les termes, les mots, les phrases, les super-phrases

(initie les morphèmes, clauses et super-clauses

) termine les morphèmes, clauses et super-clauses

/ initie et termine les propositions fermées (mots, phrases, super-phrases)

Chapitre 4

Les USLs

4.1 Définition

Un USL est un *ensemble ordonné de propositions fermées*, quelque soit le niveau des propositions.

En script, l'USL commence par une étoile et se termine par deux étoiles.

$*/proposition_i//proposition_j//proposition_k//.../**$

Puisque les propositions sont des séquences ponctuées de termes et que les USLs sont des séquences de propositions, alors *les USLs sont des séquences ponctuées de termes.*

4.2 Ordre des propositions dans un USL

L'ordre des propositions fermées dans un USL est le suivant :

1. D'abord les mots, puis les phrases, enfin les super-phrases.
2. Les propositions fermées de même niveau sont rangées dans l'ordre croissant du cardinal de l'ensemble de termes qui les composent.
3. Pour les propositions qui contiennent le même nombre de termes, on les range par leur premier terme dans la séquence selon la grammaire IEML section 2.2.7. Au cas où les deux premiers termes sont identiques, on compare les seconds termes, et ainsi de suite.

4.3 Enchâssement récursif des USLs

4.3.1 L'enchâssement

— Un USL s'enchâsse dans un autre USL juste avant le *crochet final* d'un mot, d'une phrase ou d'une super-phrase.

— *...**]

— On peut enchâsser un USL dans un mot ou dans une phrase, même si ces propositions ne sont pas fermées.
— On ne peut pas enchâsser un USL dans un morphème, dans une clause ou dans une super-clause.
— L'USL enchâssée commente, annote ou précise la proposition qui se ferme par le crochet.

Exemple d'un USL qui contient un mot-morphème et qui enchâsse un USL:

$*/[(morphème) * /proposition_i//proposition_j//proposition_k//... * *]/ * *$

Dans la pratique, les USLs seront enchâssé par l'intermédiaire du tag en langue naturelle qui les intitule.

$*/[(morphème) * tag * *]/ * *$

4.3.2 Les strates

— Un USL de strate 0 est un USL qui ne contient aucun USL enchâssé.
— Un USL de strate 1 enchâsse exclusivement des USLs de strate 0
— Un USL de strate 2 enchâsse des USLs de strate 0 ou 1
— Un USL de strate n enchâsse des USLs dont la strate ne dépasse pas n–1

Le nombre de strate est *limité*, de telle sorte que la plus basse strate (0) ne contienne aucun USL enchâssé et puisse être décodée uniquement en fonction des termes du dictionnaire.

4.4 Insertion d'une séquence de caractères non-IEML

— Pour insérer une séquence de caractères non-IEML dans un USL on utilise les *chevrons* $< >$.
— La séquence de caractères non-IEML entre chevrons peut contenir un nombre, un hyperlien, un nom propre, un terme scientifique, etc. Cette séquence de caractères n'est pas prise en compte dans les calculs sémantiques et sa longueur est limitée (cent caractères, par exemple).
— La séquence de caractères insérée se situe immédiatement *après* le crochet final d'un mot, d'une phrase ou d'une super-phrase.
— $]<...>$
— La séquence entre chevrons commente, annote ou précise la proposition qui se ferme par le crochet.

Exemple: $*/[mot]//[phrase_i]//[phrase_j] < 2011 > / * *$

4.5 Les symboles du script d'USL

$<$ initie une séquence de caractères non-IEML.
$>$ termine une séquence de caractères non-IEML
***** initie un USL
****** termine un USL.

Chapitre 5

Distances sémantiques

Chapitre 6

Patterns sémantiques