Traitement Automatique du Langage

Cours 4 - Modèle syntaxique : Lexical Functional Grammar

Antoine Rozenknop
antoine.rozenknop@lipn.univ-paris13.fr

18 novembre 2010

Plan

1 Grammaire lexicale fonctionnelle

Créée à la fin des années 70, en réaction aux grammaires Chomskyennes, la grammaire LFG (pour Lexical functional grammar) se voulait plus plausible et réaliste linguistiquement parlant que ces dernières. D'un point de vue formel, elle dispose d'un appareillage théorique faible et ne fait pas appel aux transformations ou à la notion d'événement vide, qui apparaissent dans la théorie $\bar{X}$ de Chomsky (nous verrons cette théorie plus tard dans le cours : du point de vue informatique, son utilisation est en effet trop complexe. Malgré ce défaut, les grammaires Chomskyennes font référence lorsqu'il s'agit de concevoir des grammaires, même exprimées avec d'autres formalismes : elles induisent certains principes, en particulier dans la formation des syntagmes, qui peuvent s'appliquer à peu près quel que soit le formalisme adopté).

1.1 c-structures et f-structures

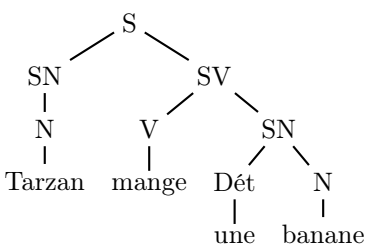
Une grammaire LFG fournit une **structure-c** (c'est-à-dire une structure en *constituants*) qui représente une *organisation* en syntagme d'une phrase ainsi que l'*ordre* de ses mots, et une **structure-f** (une structure *fonctionnelle*) qui apporte des informations relatives, par exemple, aux cadres de sous-catégorisation, à l'accord, au temps, au mode...

Exemple : Une phrase comme

Tarzan mange une banane

donnera lieu à l'analyse suivante :

Structure-C :



$$Structure - F \ : f_1 \left[\begin{array}{ll} \text{PRED} & \text{'manger'<SUBJ, OBJ>} \\ \text{SUBJ} & \left[\begin{array}{ll} \text{PRED} & \text{'Tarzan'} \\ \text{NUM} & \text{Sg} \\ \text{GEND} & \text{Masc} \\ \text{PERS} & 3 \end{array}\right] \\ \text{TENSE} & \text{Pres} \\ \text{MODE} & \text{Ind} \\ \text{NUM} & \text{sg} \\ \text{GEND} & \text{Masc} \\ \text{PERS} & 3 \\ \text{OBJ} & \left[\begin{array}{ll} \text{PRED} & \text{'banane'} \\ \text{NUM} & \text{Sg} \\ \text{GEND} & \text{Fem} \\ \text{Def} & - \end{array}\right] \end{array}\right]$$

En réalité, chacun des nœuds de la structure-c est associé à une structure-f. La structure-f de la phrase correspond à celle du nœud Sde la structure-c ci-dessus. Dans cet exemple, le premier nœud SN possède également sa structure-f associée :

$$\left[\begin{array}{ll} \text{PRED} & \text{'Tarzan'} \\ \text{NUM} & \text{Sg} \\ \text{GEND} & \text{Masc} \\ \text{PERS} & 3 \end{array}\right]$$

Définition 1. *Une structure-f est un ensemble de traits.*

Définition 2. *Un trait est un couple (attribut, valeur).*

Une valeur peut être :
- ***atomique**; Ex : Sg pour l'attribut NUM, Masc pour l'attribut GEND*
- *constituée d'une **structure de traits** enchâssée; Ex : la valeur de l'attribut SUBJ*
- *constituée d'une **liste de valeurs**, notée entre accolades.*

2 Composants de la grammaire

Pour parvenir à un tel résultat, on utilise une grammaire et un lexique.

2.1 Lexique

Voici le lexique correspondant à l'exemple analysé :

Tarzan, N : PRED 'Tarzan', NUM Sg, GEND Masc. banane, N : PRED 'banane', NUM Sg, GEND Fem. une, Det : Def -, NUM Sg, GEND Fem. mange, V : PRED 'manger <SUBJ, OBJ>', TENSE Pres, NUM Sg, PERS 3.

Chaque entrée du lexique compte **trois** éléments :
- la **graphie** (Tarzan, banane...) ;
- la **catégorie morphosyntaxique**, ou constituant (N, Det, V) : elle apparaît dans les structures-c des analyses ;
- une liste de **traits** fonctionnels (Def -, NUM Sg, GEND Fem..) : ils apparaissent dans les structures-f des analyses, et sont "propagés" d'un constituant à l'autre par l'intermédiaire des équations fonctionnelles.

2.2 Grammaire

Voici un exemple de grammaire, correspondant à l'exemple d'analyse donné plus haut :

S	→	SN	SV
		↑SUBJ=↓	↑= ↓
			↑MODE=ind
			↑SUBJ NUM=↓NUM
			↑SUBJ GEND=↓GEND
			↑SUBJ PERS=↓PERS

- ↑SUBJ=↓ : le SNest sujet.
- ↑MODE=ind : le mode du SV est l'indicatif.
- ↑SUBJ NUM=↓NUM : Accord du sujet nombre.
- ↑SUBJ GEND=↓GEND : Accord du sujet en genre.
- ↑SUBJ PERS=↓PERS : Accord du sujet en personne.
- ↑= ↓ : indique que la structure fonctionnelle de Set de SVsont les mêmes.

SN	→	N		
		↑= ↓		
SN	→	Det	N	
		↑= ↓	↑= ↓	
SV	→	V	SN	(Le SN est objet)
		↑= ↓	↑OBJ = ↓	

L'écriture des règles est un peu plus souple que pour une grammaire hors-contexte. On peut en particulier utiliser :

- **les parenthèses** pour marquer un symbole optionnel
- *** (étoile de Kleene)** pour marquer un symbole pouvant se répéter un nombre indéfini de fois (y compris zéro fois).

2.3 Equations fonctionnelles

Les règles de grammaire (réécriture) sont associées à des équations fonctionnelles. Ce sont ces équations qui permettent à la fois de construire les structures-f, et de vérifier que la phrase est bien formée.

Plus précisément, chaque équation fonctionnelle est associée à **UN non-terminal** apparaissant dans la partie droite d'une règle.

Une équation fonctionnelle fait le plus souvent appel aux symboles ↓ et ↑, qui désignent chacun une structure-f :

- ↓ renvoie à l'ensemble des traits du groupe ou de la catégorie désignée (le non-terminal associé à l'équation fonctionnelle) ;
- ↑ renvoie à l'ensemble des traits du groupe immédiatement dominant dans la structure-c (qui est aussi la tête de la règle).

Ces symboles peuvent être suivis d'un « chemin » vers une sous-structure.

Ainsi :

- ↑ = ↓ dans la première règle indique que les structures-f associées au nœud S(↑) et au nœud SV(↓) sont égales.
- (↑MODE) = ind : indique que le trait MODE de la structure-f de S(↑) a pour valeur « ind ».
- ↑ SUBJ NUM = ↓ NUM : indique que le trait NUM de la sous-structure SUBJ de la f-structure de S(↑) a la même valeur que le trait NUM de la f-structure de SV(↓).

3 Analyse syntaxique

L'analyse de la phrase se fait en deux temps (pas forcément disjoints dans la pratique) :

- **création de la structure-c**, à l'aide de la première partie des règles grammaticales ; peut se faire à l'aide des algorithmes des grammaires hors-contexte ;
- **recherche des f-structures** associées aux nœuds de la c-structure, de manière à ce que toutes les équations fonctionnelles soient vérifiées.

3.1 Unification

Les équations fonctionnelles sont vraiment des équations : elles indiquent les propriétés que doivent vérifier les f-structures d'une analyse pour que l'analyse soit correcte au sens de la grammaire LFG. Elles doivent être vraies une fois que l'analyse est complète.

Mais comment fait-on pour **construire** une analyse ? Ou plutôt, étant donné un arbre syntagmatique, pour construire les f-structures associées à chacun des nœuds ?

On utilise pour cela l'opération d'unification :

Définition 3. *L'unification de deux structures de traits A et B est à la fois une* ***extension de A et de B****. C'est-à-dire qu'elle doit contenir toutes les informations de A et de B en même temps. Ainsi si les structures A et B contiennent des informations qui ne sont pas cohérentes (i.e. qui ne sont pas compatibles, par exemple en genre), on dira que l'unification échoue.*

La structure résultant de l'unification est ***la plus petite*** *des extensions de A et de B : un trait qui n'apparaît ni dans A ni dans B ne peut pas se trouver dans l'unification de A et B.*

L'opération d'unification est **idempotente** ($A \cup A = A$), **commutative** ($A \cup B = B \cup A$)et **associative** ($(A \cup B) \cup C = A \cup (B \cup C)$)

3.2 Formation des structures-f

On construit les structures-f petit-à-petit, en partant des traits déjà associés aux feuilles de l'arbre, qui viennent des informations contenues dans le lexique.

En remontant dans l'arbre, on effectue les opérations d'unification pour chacune des équations fonctionnelles devant être vérifiées : lorsqu'on a une équation du type $chemin_1 = chemin_2$, on réalise l'unification des structures-f correspondant à $chemin_1$ et $chemin_2$, puis on **remplace** chacune de ces structures-f par le résultat de l'unification, si celle-ci a réussi. Si l'unification échoue, c'est que les règles utilisées pour construire l'arbre ne conduisent pas à une analyse syntaxique de la phrase, au sens de la grammaire LFG.

- en partant des feuilles (lexique)
- en remontant le long des branches
- en appliquant les opérations d'unification pour chaque règle

3.3 Schémas de sous-catégorisation

La plupart des traits des structures-f n'ont pas d'autre fonction que de vérifier les équations fonctionnelles. Lorsqu'on construit une grammaire LFG, on est libre de créer ou de choisir les traits dont on a besoin. Par exemple, en français, on pourra associer un trait GENRE aux adjectifs, et contraindre l'accord de l'adjectif épithète avec le nom en utilisant la règle grammaticale :

SN	→	Adj	N
		« ↑GENRE=↓GENRE »	« ↑GENRE=↓GENRE »

En anglais, on n'aura sans doute pas besoin de ce trait pour les adjectifs, qui ont la même forme quel que soit leur genre.

Il existe cependant un trait qui a une signification imposée ; le trait **PRED**. Il est constitué :

- du **prédicat** du groupe ;
- du **schéma de sous-catégorisation** du groupe : une liste de fonctions devant apparaître dans la structure-f. On dit que ces fonctions sont **gouvernées** par le prédicat du groupe.

Exemples :

PRED 'manger'<SUBJ, OBJ>

- le prédicat est 'manger'
- la structure-f doit contenir les traits SUBJ et OBJ (sujet et objet).

PRED 'dormir'<SUBJ>

- le prédicat est 'dormir'
- la structure-f doit contenir le trait SUBJ.

3.4 Principes de bonne formation des structures-f

Les fonctions apparaissant dans les schéma de sous-catégorisation sont dites **sous-catégorisables**. Il est important de spécifier à l'avance la liste des fonctions sous-catégorisables de la grammaire, car elles ont une fonction particulière, par l'intermédiaire des principes de bonne formation des structures-f.

Ces principes sont au nombre de trois :
- **Unicité** : un même attribut ne peut apparaître deux fois dans une même structure-f ;
- **Cohérence** : toutes les sous-structures doivent être localement cohérentes, ce qui signifie que les fonctions sous-catégorisables qui y apparaissent doivent être gouvernées par le prédicat local.
- **Complétude** : toutes les sous-structures doivent être localement complètes, ce qui veut dire que toutes les fonctions gouvernées par leur prédicat local doivent être présentes.

Les principes de cohérence et de complétude font appel aux schémas de sous-catégorisation : ils sont là pour permettre par exemple de vérifier qu'un verbe a **tous** ses compléments et **seulement** ceux qu'il peut admettre, donc pour interdire des phrases comme :

Tarzan dort la banane.	(principe de cohérence violé)
Tarzan mange.	(principe de complétude violé)

Le principe de complétude indique qu'à la fin d'une analyse, une structure-f contenant le trait [PRED 'mange<SUJ,OBJ>'] doit aussi contenir les attributs SUJ et OBJ.
Le principe de cohérence indique qu'à la fin de l'analyse, une structure-f contenant le trait [PRED 'dormir<SUJ>'] ne doit pas contenir de trait OBJ (par exemple).

Les principes de complétude et de cohérence doivent être **vérifiés à la fin** de la construction des structures-f, lorsque toutes les unifications ont eu lieu. Ils n'interviennent pas au cours de cette construction, qui ne fait appel qu'au principe de l'unification.

3.5 Equations fonctionnelles spéciales

Il existe des équations fonctionnelles qui n'interviennent pas dans l'unification, mais qui doivent être vérifiées **à la fin** de l'analyse : les équations-contraintes :

- **le signe** $=_c$: (à la place de =) indique une équation devant être vérifiée à la fin de l'analyse, mais qui ne doit pas être utilisée lors de l'unification.
  Par exemple : $\uparrow$ PERS $=_c$ 3 : vérifie que l'attribut PERS a la valeur 3 dans la structure-f du nœud supérieur, mais si l'attribut n'est pas présent dans la f-structure finale, le test échoue et la f-structure n'est pas valide (alors qu'avec '=', une unification aurait ajouté PERS 3 à la f-structure) ;
- **le signe** $\sim$: indique l'absence d'un trait.
  Par exemple : $\sim$ [$\uparrow$PERS 3] : vérifie que le trait PERS 3 n'est pas dans la f-structure du nœud supérieur.

Il existe enfin des équations fonctionnelles spéciales pour traiter les attributs dont la valeur est une **liste** de traits. Elles font appel à l'opérateur $\ni$:
« A $\ni$ B » indique que la structure de trait B est un élément de la liste A.

4 Conclusion

Il faut bien faire la distinction entre les structures-f et les structures-c. Leur formation repose sur deux parties bien distinctes des règles grammaticales. En principe, les symboles apparaissant dans les structures-c représentent la **nature** des groupes syntagmatiques (groupes nominaux, verbaux...), alors que les structures-f représentent les **fonctions** qui y apparaissent (prédicat, compléments). Cependant, les choses ne sont pas si distinctes en pratique : le plus souvent, les structures-f contiennent aussi des informations sur la nature du syntagme (genre, nombre), qui sont utilisées pour contrôler les interactions de ce syntagme avec son environnement (accords). En ce sens, une grammaire LFG n'est donc pas une grammaire hors-contexte.

Les LFG laissent une grande liberté dans l'écriture de grammaires, et l'on peut exprimer le même langage avec plusieurs LFG différentes. L'écriture de « bonnes » LFG relève de la pratique et de l'étude de LFG existantes.

Pour finir, voici une liste non exhaustive de différentes fonctions que l'on trouve couramment dans les grammaires LFG (sans qu'il y ait de « nomenclature officielle » des fonctions LFG).

La plupart de ces fonctions sont en général sous-catégorisables.

SUBJ :	sujet	Max dort
OBJ :	objet Max	mange une pomme
A_OBJ :	objet introduit par la préposition « à »	Max donne une pomme **à Marie**
DE_OBJ :	objet introduit par la préposition « de »	Max rêve **de Marie**
COMP :	complétive	Max sait **que Marie viendra demain**
VCOMP :	infinitive	Max veut **devenir professeur**
ACOMP :	attribut adjectival	Max trouve **magnifique cet ouvrage**
NCOMP :	attribut nominal	Le peuple à élu Max **président**
Vajout :	subordonnée circonstancielle	**Voulant partir tôt**, Max se réveille à 5h
Aajout :	adjectif apposé	Max est parti **heureux**