LFG : mémo du formalisme

On présente ici les aspects formels de LFG, en laissant de côté les aspects théoriques.
On reviendra ensuite sur les concepts théoriques suivants :

- Fonctions grammaticales
- Distinction argument / ajout et sous-catégorisation
- Distinction argument sémantique / argument syntaxique, appariement
- Le trait PRED, forme sémantique

Représentation associée à une phrase

En LFG, la représentation associée à une phrase est composée de

- la **structure-c** (structure de constituants) : un arbre syntagmatique
- la **structure-f** (structure fonctionnelle) : une structure de traits qui explicite les fonctions grammaticales, et des informations comme les marques d'accord, certaines co-références.
- des liens de correspondance entre une structure de traits apparaissant dans la structure-f, et 0, 1 ou plusieurs nœuds de la structure-c

Remarque : dans ses récentes avancées LFG intègre également pour représenter une phrase

- une structure informationnelle distincte de la structure fonctionnelle (pour représenter la distinction entre *Marie, Paul l'a vue* et *Paul a vu Marie.*)
- une structure morpho-syntaxique pour distinguer les marques morphologiques grammaticales le temps, le mode de leur valeur sémantique (l'aspect par exemple)
- une structure sémantique : pour le calcul de la sémantique d'une phrase.

NB : Dans le cours, on se concentre sur la partie syntaxique de LFG, et un état de la théorie où la structure-f intègre une partie morpho-syntaxique (traits de personne, mode …) et une partie informationnelle (avec les fonctions TOPIC et FOCUS, qui seront abordées plus tard)

Une grammaire LFG

LFG constitue une grammaire au sens « théorie de la description linguistique » :
LFG définit comment organiser formellement la connaissance linguistique pour une langue donnée, et comment pour un énoncé donné, prédire si cet énoncé est grammatical ou pas.
En ce sens, LFG est une grammaire **générative**.

Une grammaire LFG est un objet formel destiné à **représenter** une langue donnée / un fragment de langue donnée. Elle est constituée de règles de réécriture « hors-contexte » (= de type 2, selon la hiérarchie de Chomsky), où chaque symbole peut être associé à 0 une ou plusieurs **équations fonctionnelles**.
Notation : Dans chaque règle, dans une équation fonctionnelle sous un symbole X, ↑ désigne la structure-f associée au nœud de la partie droite de la règle, et ↓ désigne la structure-f associée au symbole X.

Exemple de grammaire simplifiée pour la suite :

PetiteGrammaire :

(1) P →	SN ↑SUJ = ↓	SV ↑=↓	
(2) SV →	V ↑=↓	(SN) ↑OBJ = ↓ ou ↑NCOMP = ↓)	(SP)* ↑(↓PCAS) = ↑ ou ↓ ∈ ↑AJOUT
(3) SN →	Det ↑=↓	N ↑=↓	
(4) Det →	les ↑NB=pl		
(5) Det →	la ↑NB=sing ↑GENRE=fem		
(6) N →	lionnes ↑PRED='lionne' ↑NB=pl ↑GENRE=fem		
(7) N →	lionne ↑PRED='lionne' ↑NB=sing ↑GENRE=fem		
(8) V →	dorment ↑PRED='dormir<SUJ>' ↑SUJ NB = pl		
(9) V →	rencontrent ↑PRED='rencontrer<SUJ,OBJ>' ↑SUJ NB = pl		

Les principes de bonne formation

Une grammaire LFG pour un fragment de langue donné utilise des principes généraux, propres à la théorie LFG, pour construire la représentation associée à un énoncé, et prédire si cet énoncé est grammatical ou pas : des **principes de bonne formation** de la structure-f.

Principe de base	la structure-f est la solution minimale qui satisfasse les équations fonctionnelles associées à la structure-c, et cette structure minimale ne doit pas valoir ⊥ : les unifications imposées par l'instanciation des équations fonctionnelles ne doivent pas échouer.
Permet d'interdire :	(si la grammaire gère l'accord) **Marie regarderont Pauline.*

Des principes supplémentaires sont ajoutés, pour gérer les contraintes de sous-catégorisation

Principe d'unicité	un même attribut fonctionnel ne peut apparaître deux fois au même niveau dans une f-structure (pris en charge par le principe plus général d'unification)
Remarque : déjà pris en charge par le principe de base supra	
Permet d'interdire :	**Marie regarde Pauline le navire.*

Principe de cohérence	Les fonctions sous-catégorisables doivent être gouvernées par un prédicat local
Reformulation :	Dans une structure-f, si un attribut fonctionnel F apparaît à un niveau, et que F correspond à une fonction sous-catégorisable (SUJET, OBJET, …), alors la structure-f doit avoir un trait PRED au même niveau, et ce trait PRED doit contenir F dans la liste de sous-catégorisation
Permet d'interdire :	**Jean dort Marie* ** Le problème concerne Jean à Marie*

Principe de complétude	Toute fonction sous-catégorisée doit être remplie, et si elle correspond à un argument sémantique, alors sa valeur doit être sémantiquement pleine.

Reformulation :	Dans une structure-f, toute fonction apparaissant dans la valeur d'un trait PRED doit apparaître comme attribut au même niveau que ce trait PRED. Et si la fonction correspond à un argument sémantique (notée à l'intérieur des chevrons), alors la valeur de l'attribut fonctionnel doit avoir un trait PRED.
Permet d'interdire :	*Jean rencontre *Jean met le tableau

Construction de la représentation associée à une phrase.

Pour une phrase donnée, on peut essayer de fournir sa représentation d'après une grammaire LFG *magrammaire*, en suivant une méthode précise.
Si l'on réussit à fournir une représentation qui satisfasse les **principes de bonne formation,** alors on peut dire que *magrammaire* **représente / engendre / accepte** cette phrase.
Sinon elle ne l'engendre pas / elle la rejette.
Le but étant que *magrammaire* engendre des phrases grammaticales et seulement celles-là, et rejette des phrases agrammaticales et seulement celles-là.

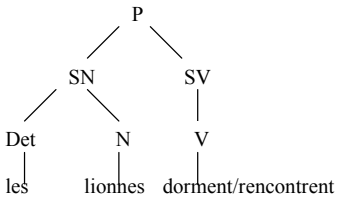
Remarque : l'acceptation / le rejet d'une phrase par une grammaire LFG est **binaire**, alors que la grammaticalité est gradable.

Pour construire la représentation associée à une phrase :

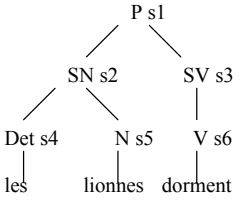
1. Appliquer les règles de réécriture (d'abord en ignorant les équations fonctionnelles) pour arriver à une ou plusieurs structures-c.
   → Si aucune structure-c n'est possible, la phrase est rejetée.

Exemple avec la PetiteGrammaire donnée supra :
Pour *lionnes les dorment*, on ne peut pas associer de structure-c → la phrase est rejetée.

Pour *les lionnes dorment* ou pour **les lionnes rencontrent*, on associe la structure-c :



2. Puis pour chaque structure-c trouvée (il peut y en avoir plusieurs) construire la structure-f associée à chaque nœud, d'après les équations fonctionnelles. La structure-f complète, si elle existe, est la structure-f associée au nœud racine (ici P).
   Pour cela, utiliser des variables pour la structure-f de chaque nœud syntagmatique :



Pour tout nœud non feuille, on connaît la règle de réécriture qui a permis de produire le morceau d'arbre (le nœud + tous ses fils).
Par exemple, le nœud SN et ses fils Det et N viennent de la règle (3)
« Instancier les équations fonctionnelles » de la règle 3), étant données les variables choisies pour les structures-f de chaque nœud, donne
s2=s4 et **s2=s5**.
Puis la règle 1) donne **s1=s3**, et **s1 ⊃ [SUJ = s2]**
la règle 2) donne **s3 = s6** , donc **s1 =s3=s6**
la règle 4) donne **s4 ⊃ [NB = pl]**
la règle 6) donne **s5 ⊃ [NB = pl, GENRE=fem, PRED ='lionnes']**
la règle 8) donne **s6 ⊃ [SUJ [NB = pl], PRED ='dormir<SUJ>']**

La structure-f pour cette phrase est s1, et doit valoir la plus petite structure-f qui satisfasse les équations fonctionnelles. Cette plus petite structure est :

$$s1=\begin{bmatrix} PRED\ \_\_\_ \text{'}dormir < SUJ > \text{'} \\ SUJ\ \_\_\_\_ \begin{bmatrix} PRED\ \_\_\_ \text{'}lionnes\text{'} \\ NB\ \_\_\_\_\_\ pl \\ GENRE\ \_\_\ fem \end{bmatrix} \end{bmatrix}$$

Le principe de base est vérifié : Les unifications imposées par les équations fonctionnelles instanciées n'ont pas échoué, i.e. il existe bien une structure de traits qui les satisfasse (qui ne vaut pas ⊥).
En outre les principes de cohérence / complétude / unicité sont vérifiés.
Donc cette grammaire accepte *Les lionnes dorment.*

Pour la phrase *les lionnes rencontrent*, on doit utiliser la règle 9) au lieu de 8), ce qui donne

$$\begin{bmatrix} PRED\ \_\_\_ \text{'}rencontrer < SUJ, OBJ > \text{'} \\ SUJ\ \_\_\_\_ \begin{bmatrix} PRED\ \_\_\_ \text{'}lionnes\text{'} \\ NB\ \_\_\_\_\_\ pl \\ GENRE\ \_\_\ fem \end{bmatrix} \end{bmatrix}$$

Le principe de base est vérifié, mais pas le principe de complétude : il manque un trait OBJ au niveau du trait PRED rencontrer.
Donc la grammaire rejette bien **Les lionnes rencontrent.*

Equations fonctionnelles

On utilise en LFG plusieurs types d'équations fonctionnelles, qui fonctionnent comme des contraintes (on note **sf** pour structure-f)

identité

notée =, entre deux sf ou bien entre une sf et une valeur atomique
Exemples

↑=↓	→ la sf du nœud supérieur et celle du nœud courant sont la même sf Pour résoudre cette contrainte, c'est trouver la sf minimale la satisfaisant. En pratique on unifie les traits provenant de ↓ et ceux provenant de ↑ dans d'autres équations
↑ SUJ = ↓	→ la sf associée au nœud supérieur a un trait SUJ, dont la valeur est la sf associée au nœud courant.
↑ NB = pl	→ la sf associée au nœud supérieur doit contenir (et donc être compatible avec) le trait NB = pl

↑ SUJ NB = pl → la sf associée au nœud supérieur a un attribut SUJ, qui a un attribut NB, qui vaut pl (la valeur de SUJ est une extention de NB = pl)

équation contrainte

notée =c, entre une sf et une valeur
Exemple
↑SUJ FORM =c il → la sf associée au nœud supérieur doit non seulement être compatible avec SUJ FORM=il, mais ce trait doit 'provenir' (être imposé) par d'autres équations fonctionnelles

équation d'appartenance

avec le symbole ∈, pour le cas d'attribut à valeur ensembliste
Exemple
↓ ∈ ↑ AJOUT → la sf associée au nœud supérieur a un trait AJOUT à valeur ensembliste, dont la valeur contient la sf associée au nœud courant.

contrainte existentielle

notée entre parenthèses : indique qu'un attribut doit être présent, sans contrainte sur sa valeur
Exemple
(↑DET) → la sf associée au nœud supérieur doit porter un attribut DET

combinaison d'équations fonctionnelles

Les équations fonctionnelles sous un symbole de règle peuvent être combinées, à l'aide d'opérateurs logiques :
- conjonction (notation sans opérateur)
- disjonction, notée |, ou 'ou' (↑OBJ=↓) | (↓ ∈ ↑ AJOUT)
- négation, notée ~ ~(DET) ~(MODE=inf)

chemin fonctionnel dans les équations fonctionnelles

Les opérandes des opérateurs =, =c ou ∈ peuvent être désignés par un **chemin** : une cascade de 0, 1 ou plus d'attributs
↑=↓ → équation avec chemins fonctionnels vides à gauche et à droite
↑ XCOMP SUJ = ↑OBJ → la sf du nœud supérieur a un attribut XCOMP, qui a un attribut SUJ qui vaut x et elle a aussi un attribut OBJ, qui pointe sur la même sf x

Ces exemples correspondent à suivre un chemin « descendant » dans les attributs contenus par une sf. On peut aussi commencer par un chemin « ascendant » : les attributs dans lesquels la sf est contenue.
(AJOUT ↑) NB = pl → la sf associée au nœud supérieur (↑) apparaît comme la valeur du trait AJOUT d'une sf englobante, qui a un trait NB=pl

chemin fonctionnel servant à désigner un attribut

On peut pour désigner un attribut donner le symbole (cf. tous les exemples supra), ou bien utiliser un chemin fonctionnel qui a in fine une valeur atomique
↑(↓PCAS) = ↓ la sf du nœud courant a un attribut PCAS, à valeur atomique x, et la sf du noeud supérieur a un attribut x qui vaut la sf du nœud courant (!)

équation optionnelle

notée (aussi) entre parenthèses (PRED ='pro') : équivalent à avoir 2 règles alternatives, l'une avec l'équation, l'autre sans.