Grammaires Catégorielles Combinatoires

Pascal Amsili,
très inspiré de (Kallmeyer *et al.* , 2007)

Langages Formels ME03LI 2015

CCG

- Combinatory Categorial Grammar (CCG) : formalisme grammatical équivalent à TAG ($\approx$ mildly context sensitive) :
  - formalisme lexicalisé
  - parsable en temps polynomial
  - capable de représenter des dépendances croisées (cross-serial dependancies)
- Comme HPSG ou TAG, utilisé pour l'« ingénierie grammaticale »
- Particulièrement approprié pour le parsing statistique

En résumé

- Inventaire de types : pour la spécification des listes de sous-catégorisation des constituants
- Lexique : association de catégories et d'une sémantique aux mots
- Règles de combinaison : pour la spécification du mode de combinaison des constituants
- Dérivation : historique des combinaisons
- Interface syntaxe-sémantique : les catégories et les règles de combinaison ont un pendant sémantique.

Catégories

Il y a deux types de catégories :

- les types primitifs (catégories simples) : S, NP
- les types fonctionnels (catégories complexes) :
  - Verbe intransitif : S\NP
  - Verbe transitif : (S\NP)/NP
  - Préposition : (NP\NP)/NP

Notations *à la Steedman :*

type rés. / type arg. (à droite)
type rés. \ type arg. (à gauche)

Notations *à la Lambek :*

type rés. / type arg. (à droite)
type arg. \ type rés. (à gauche)
« result on top »

Le lexique

- Le lexique attribue à chaque mot la ou les catégories qu'il peut avoir ;
- De plus, le lexique spécifie la contrepartie sémantique de la règle syntaxique, par exemple :
  aime (S\NP)/NP $\lambda x.\lambda y.((love')x)y$
- Les règles combinatoires déterminent ce qui se passe au niveau syntaxique et au niveau sémantique lors de la combinaison

Application fonctionnelle

- $X/Y : f \quad Y : a \quad \Rightarrow \quad X : (f)a \qquad >$
- $Y : a \quad X\backslash Y : f \quad \Rightarrow \quad X : (f)a \qquad <$

$$\dfrac{\dfrac{\text{Jean}}{NP : j} \quad \dfrac{\dfrac{\text{aime}}{(S\backslash NP)/NP : \lambda x \lambda y.Lxy} \quad \dfrac{\text{Marie}}{NP : m}}{S\backslash NP : \lambda x Lxm}>}{S : Ljm}<$$

Montée de type

- $X : a \quad \Rightarrow \quad T/(T\backslash X) : \lambda f.(f)a \qquad >_T$

- La montée de type transforme un argument en une fonction
- ex : NP $\Rightarrow$ (S / (S\NP)) (prédicat $\rightarrow$ qg)
- Utile pour les mouvements qu- (par exemple)

Composition de types

- $X/Y : f \quad Y/Z : g \quad \Rightarrow \quad X/Z : \lambda x.((f)g)x \qquad >_B$

- La composition fonctionnelle permet la composition de fonctions/types fonctionnels
- ex : (S\NP)/PP PP/NP $\Rightarrow$ (S\NP)/NP

Restriction

Ces deux opérations additionnelles donnent de la puissance au formalisme, et on ajoute la contrainte que leur emploi ne se fait qu'en cas de nécessité syntaxique (pour « débloquer » un calcul).

Exemple

Un exemple avec montée de type et composition :

$$\dfrac{\dfrac{\dots}{NP} \quad \dfrac{\dfrac{\text{que}}{(NP\backslash NP)/(S/NP)} \quad \dfrac{\dfrac{\dfrac{\text{Jean}}{NP}}{\begin{array}{c}NP\\ :j\end{array}}}{\dfrac{}{S/(S\backslash NP)}\,{>}\mathbf{T}}}{\dfrac{}{S/NP}\,{>}\mathbf{B}}}{\dfrac{}{NP\backslash NP}\,{<}}}{NP}\,{<}$$

$$\begin{array}{c}S/(S\backslash NP)\\ \lambda P.(P)j\end{array}$$

$$\begin{array}{c}S/NP\\ \lambda x.((A)x)j\end{array}$$

$$\dfrac{\dfrac{\text{aime}}{(S\backslash NP)/NP}}{\begin{array}{c}(S\backslash NP)/NP\\ :\lambda x\lambda y.((A)x)y\end{array}}$$

References

Kallmeyer, Laura, Lichte, Timm, & Maier, Wolfgang. 2007. *Grammar Formalisms : Combinatorial Categorial Grammar (CCG)*. Slides for a class on grammar formalisms, Universität Tübingen, `http://www.sfs.uni-tuebingen.de/emmy/gf/slides/200607-ccg.pdf`.

Steedman, Mark, *et al.* . 2012 (June). *Combinatory Categorial Grammars for Robust Natural Language Processing*. Slides for NASSLLI course `http://homepages.inf.ed.ac.uk/steedman/papers/ccg/nasslli12.pdf`.