M2 P7 Prolog-TAL – Travaux Dirigés 3

Analyser le français, en toute modestie !

Eric de la Clergerie et André Bittar

27 Novembre 2008

Introduction

Le but de ce TD est de reprendre et compléter la grammaire jouet DCG du TD 2 en utilisant au mieux l'opérateur d'unification immédiate, les domaines finis et les structures de traits (sans héritage).

Plusieurs alternatives sont possibles niveau des non-terminaux :

1. soit ils sont représentés directement par des structures de traits (`gn{}`, `s{}`) et vont devoir inclure les champs `sem` et `tree` pour la sémantique et l'arbre d'analyse.
2. soit ils sont représentés par des termes Prolog standards mais incluent des arguments qui sont des structures de traits, par exemple pour gérer les accords : `gn(agr{},Tree,Sem)`. Ceci permet plus facilement de partager des structures entre non-terminaux.
3. on peut également combiner les approches 1 et 2 avec `gn{ agr => agr{} }`

Par rapport au TD 2, on peut en profiter pour ajouter quelques extensions, telles :

- un contrôle plus fin des adjectifs antéposés (comme *petit* ou *joli*) et postposés (comme *vert*).
- la possibilité de reconnaître des phrases infinitives ou impératives, en bloquant ces phrases en tant que relatives et interrogatives (pour l'impératif).
- la possibilité de gérer des sous-catégorisations plus complexes, par exemple pour *donner*, et éventuellement *proposer* :
  - Paul donne une pomme à Marie (verbe ditransitif datif)
  - Paul donne à Marie une pomme
  - Paul propose une pomme à Marie
  - Paul propose à Marie de manger une pomme (verbe ditransitif datif avec objet infinitif et contrôle objet)

En final, on doit pouvoir analyser une phrase comme « *Paul mange le gateau que Claude demande à Marie de couper* » et obtenir :

```
./french8 -- Paul mange le gateau que Claude demande à Marie de couper
Answer:
    L = [Paul,mange,le,gateau,que,Claude,demande,à,Marie,de,couper]
    Tree = s(gn(lexical{lemma=> Paul, form=> Paul}), gv(lexical{lemma=> manger
    , form=> mange}, gn(gn(lexical{lemma=> le, form=> le}, lexical{lemma=>
    gateau, form=> gateau}), srel(lexical{lemma=> que, form=> que}, s(gn(
    lexical{lemma=> Claude, form=> Claude}), gv(lexical{lemma=> demander, form
    => demande}, s(gv(lexical{lemma=> couper, form=> couper}, trace)), gp(
    lexical{lemma=> à, form=> à}, gn(lexical{lemma=> Marie, form=> Marie})))))
    )))
    Sem = ((_A : manger(Paul, _B) & (def(_B) & gateau(_B)) & (_C : demander(
    Claude, _D, Marie) & (_D : couper(Marie, _B) & true) & true) & true) &
    true)
    XSem = (_A : manger(Paul, _B) & def(_B) & gateau(_B) & _C : demander(
    Claude, _D, Marie) & _D : couper(Marie, _B))
```

Noter au niveau sémantique le fait que `proposer` à un deuxième argument qui est un évènement réifié et que `Marie` est à la fois un argument de `proposer` et de `manger`. Noter également que le pronom `que` est extrait d'une phrase enchassée comme argument (syntaxique) de *propose*.

Le matériel pour ce TD peut être récupéré sous forme d'archive `.tgz` à `http://alpage.inria.fr/~clerger/PrologTAL08/td3.tgz`. Il comprend :

- une ébauche de grammaire DCG pour le français `french1.dcg`
- une mini-segmenteur `lex2db` permettant de construire un treilli de mots à partir d'une chaîne.
- un jeu de quelques phrases `sentences.txt`
- un fichier `Makefile`

L'ensemble du jeu de phrases peut être testé en boucle avec la commande suivante (pour une version récente de DyALog) :

```
cat sentences.txt | ./lex2db | ./french1 -loop -utime
```

Ne pas oublier de reprendre également le matériel du TD 2.