Expression Logique et Fonctionnelle . . . Évidemment

TP4

1 Résolution en SWI Prolog

Enregistrez le code disponible ici. Veillez à conserver le nom crisefinanciere.pl avec l'extension **.pl** pour prolog. Dans le répertoire où vous avez enregistré le fichier, lancez SWI prolog à partir du terminal (commande : swipl). Vous devez voir un affichage similaire au suivant :

```
Welcome to SWI-Prolog (Multi-threaded, 64 bits,
Version 5.10.4)
Copyright (c) 1990-2011 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free
software, and you are welcome to redistribute it under
certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?-
```

puis, chargez le fichier :

```
?- [crisefinanciere].
% crisefinanciere compiled 0.00 sec, 1,736 bytes
true.
?-
```

Sachez que SWI prolog

- attend un point en fin de requête (ce point est optionnel en DES).
- affiche les réponses une par une. Vous obtenez la réponse suivante en frappant le point-virgule. Quand vous frappez (ENTER) après l'affichage d'un résultat, SWI Prolog abandonne la requête courante, sans calculer d'autres réponses.

Question 1 Posez une requête permettant de vous faire afficher les dettes.

Question 2 Posez une requête pour afficher quelle fille évite quelle autre. Assurez-vous de faire afficher *toutes* les réponses. Qu'observez-vous ?

Question 3 Comparez le résultat des mêmes requêtes en DES, et expliquez d'où viennent les différences entre les deux systèmes.

2 Parsing de langage naturel

Le but du dernier excercice de programmation logique en ELFE est d'écrire, en Datalog, un parser pour un anglais simplifié. La méthode repose sur l'algorithme CYK. Il n'est pas nécessaire d'en connaitre les détails pour faire le travail pratique, mais vous êtes encouragés de rattraper cette lecture à l'occasion.

Le point de départ est un dictionnaire avec plusieurs catégories de mots :

Catégorie	mots
N	man, song, unicorn, telescope
Det	the, a
V	sees, sings
Prep	with

où N signifie noun, V verb (verbe), *Det* determiner. Un déterminant est, par exemple, un article défini (the) ou indéfini (a).

A partir du dictionnaire, la grammaire permet de construire des sous-phrases nominales NP (noun phrase) et verbales VP (verb phrase). Des exemples de NP sont *the man* et *a song*. Un exemple de VP est *sings a song*. Les règles correspondantes sont :

$$NP -> Det\ N$$
$$VP -> V\ NP$$

La règle pour construire une phrase S (sentence) est :

$$S -> NP\ \ VP$$

Question 4 Dessinez l'arbre syntaxique pour la phrase 1 : *The man sings a song*. Rendez une version ASCI en suivant l'exemple

$$((the\ DET)(man\ N)\ NP)$$

dans un fichier *arbresphrase*1.*txt*.

Question 5 Dans un fichier nommé *phrase*1.*dl*, codez la phrase *The man sings a song* sous forme de faits datalog, suivant le schéma :

```
the(0,1).
man(1,2).
...
```

Sachez que vous pouvez charger plusieurs fichiers avec DES de la manière suivante :

```
DES> /[file1,file2,file3]
```

Question 6 Codez le dictionnaire, dans un fichier *dictionnaire*1.*dl* . Il faut, pour chaque catégorie de mots apparaissant dans la grammaire, définir un prédicat IDB, et saisir chaque mot à l'aide d'une règle pour le prédicat correspondant.

Votre codage doit permettre d'obtenir les réponses suivantes : Pour la catégorie déterminant (*det*) :

```
DES> det(X,Y)

{
  det(0,1),
  det(3,4)
}
Info: 2 tuples computed.
```

Et pour la catégorie des substantifs (*noun*)

```
DES> n(X,Y)

{
  n(1,2),
  n(4,5)
}
Info: 2 tuples computed.
```

Notez que votre dictionnaire ne doit *pas* refléter l'ordre des mots dans la phrase 1.

Question 7 Codez les trois règles de la grammaire anglaise, données en début de sujet, dans un fichier *grammaire*1.*dl*. Si votre codage fonctionne, vous devez parvenir à parser la phrase entière, ou la phrase nominale *the man* qui commence à la position 0. Test :

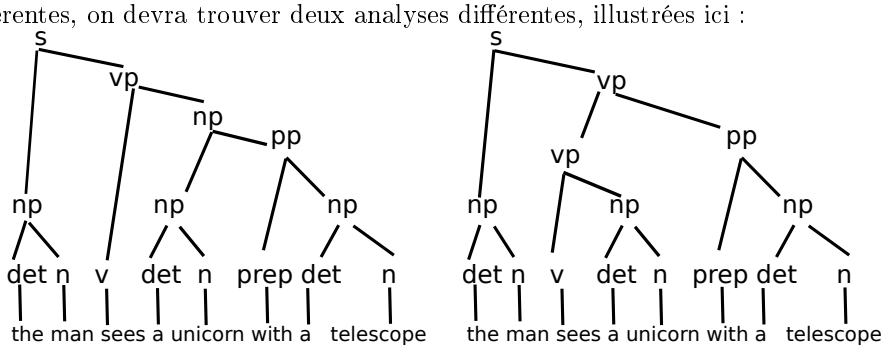
```
DES> np(0,X)

{
  np(0,2)
}
Info: 1 tuple computed.
```

Question 8 Comment tester si la phrase *the man sings a song* peut être parsée ?

Le prochain but est de pouvoir parser de manière non-deterministe, ce qui permet d'obtenir différentes analyses syntaxiques pour une phrase, s'il en existe plusieurs.

Exemple : Pour la phrase 2 *The man sees a unicorn with a telescope*, ayant deux interprétations différentes, on devra trouver deux analyses différentes, illustrées ici :



Question 9 Donnez, en français, une traduction de la phrase 2, qui préserve les ambiguïtés. Puis, expliquez les deux interprétations différentes de la phrase. Quelle est la différence sémantique ?

Question 10 Coder la phrase *The man sees a unicorn with a telescope* en Datalog. Enregistrez-la dans un fichier *phrase*2.*dl*.

Le but concret est d'enrichir le dictionnaire et la grammaire pour pouvoir traiter des phrases prépositionnelles (PP), avec des prépositions telles que *with*. La première étape est de trouver les nouvelles règles, permettant les analyses alternatives de la phrase *The man sees a unicorn with a telescope*, illustrées dans les figures.

Question 11 Quelles sont les règles de la grammaire pour traiter les PPs ?

Question 12 Ajoutez le nécessaire au dictionnaire, et enregistrez le dictionnaire complet dans *dictionnaire*2.*dl*

Question 13 Codez en Datalog les règles de la grammaire concernant les PPs. Enregistrez l'ensemble des règles de la grammaire dans un fichier *grammaire*2.*dl*.

Question 14 Proposez une requête qui montre que votre parseur trouve deux analyses différentes pour la phrase 2. Alternativement, trouvez une trace des deux analyse différentes dans l'information mémorisée.

Question 15 Comment pourrait-on, en utilisant une idée du TD5, obtenir des informations sur les differentes analyses syntaxiques ?

Question 16 **Bonus.** Proposez une manière de traiter les adjectifs, par exemple, dans la phrase *The tall man sees a white unicorn with a long telescope*, mais egalement dans la phrase *The tall and lucky man sees a white unicorn with a telescope.*