TP Prolog
Prise en main et exercices

1. Version de Prolog utilisée

Il s'agit de la version SWI-Prolog développée par Jan Wielemaker.

- Téléchargement : ftp://ftp.cict.fr/pub/lang/prolog/SWI/win95/w32pl331.exe
- Site Web pour la documentation (en anglais), d'autres téléchargements, etc. : http://www.swi-prolog.org

2. Démarrage

Il faut commencer par taper le programme dans un éditeur de textes (évitez Word !), puis il faut le sauvegarder (en `.pl`), et enfin le compiler sous SWI-Prolog.
NB : Les notions de programme principal, de début, fin… n'existent pas en Prolog. Pour lancer un programme, il faut soumettre une requête.

Chaque requête ou clause se termine toujours par un point.

2.1 Compilation

Pour compiler le programme `toto.pl`,
```
?- consult(toto).
```
ou bien
```
?- [toto].
```

Pour compiler le programme `toto.pl` si ça n'a pas encore été fait,
```
ensure_loaded(toto)
```

Pour recompiler tous les fichiers sources qui ont changé:
```
make.   %NB: pas d'argument donc pas de parenthèses
```

2.2 Requêtes

Soit un programme Prolog comportant les clauses suivantes :
```
homme(jean).
homme(pierre).
```

Exemples de requêtes :
```
?- homme(jean).
Yes

?- homme(paul).
No

?- homme(X).
X = jean    Enter
Yes

?- homme(X).
X = jean    ;
X = pierre  ;
No
```

Explications : Quand il y a plusieurs solutions, « Retour chariot » stoppe la recherche des solutions tandis que « ; » permet de demander la solution suivante. Quand Prolog ne trouve plus (ou pas) de solutions, il répond « No ».

2.3 Noms

Les majuscules ou minuscules ont une importance. Une constante commence par une minuscule, une variable par une majuscule ou un `underscore _`

Pour définir des noms avec des espaces et/ou des majuscules, il faut utiliser des apostrophes. *NB* : pere et 'pere' sont équivalents.

2.4 Règles

Les caractères « `:-` » représentent la déduction.

La règle
`conclusion :- premisse1, premisse2.`
signifie « `conclusion` est vraie dès que `premisse1` et `premisse2` sont vraies ».

`?- clause/2` permet de regarder le contenu d'une règle ; d'obtenir les clauses d'un prédicat (*to get clauses of a predicate*).

2.5 Aide

Aide sur un sujet (topic) ou sur un mot (word):
```
?- help(Topic).
?- apropos(Word).
```

2.6 Déboguage

Pour voir tous les appels lors d'une exécution :
```
?- trace.
```
Pour sortir de ce mode :
```
?- notrace.
```

2.7 Sortir

`halt.` permet de sortir de Prolog.

3. Exercices

3.1 Reprendre les exercices vus en cours et TD sur la généalogie de Louis XVI, les programmer, les tester.

3.2 Ecrire le prédicat frereousoeur/2.

3.3 Reprendre les exercices vus en cours sur les listes : concat, renverser, partition… Pensez à utiliser la clause trace. sur quelques appels de prédicats.

3.4 En particulier, reprendre le prédicat elt_impair. Que se passe t-il si le nombre d'éléments de la liste n'est pas pair ? Modifiez le prédicat, de sorte qu'il donne le bon résultat, quel que soit le nombre d'éléments dans la liste.

3.5 Reprendre les exercices vus en cours et TD sur les *cuts*.