Introduction à Prolog

Pour installer ECLiPSe CLP sur votre machine, suivez les instructions sur la page Web d'ECLiPSe (`eclipseclp.org`).

Sur les machines de l'UFR il faut utiliser la commande `eclipse-clp` pour lancer ECLiPSe CLP. Dans un terminal, on peut lancer `rlwrap eclipse-clp`[1].

Pour compiler le programme `toto.pl`, après l'avoir écrit et sauvegardé avec un éditeur de texte (comme `emacs`), ou pour le recompiler après modification, taper `compile(toto).` (attention: le point final est nécessaire. Chaque requête doit se terminer par un point.)

Pour quitter l'application: taper `halt.` ou `Ctrl d`.

Exercice 1
Écrire le programme prolog suivant dans un fichier `fact.pl` :

```
fact(0,1).
fact(N,R) :- N>0, M is N-1, fact(M,T), R is N*T.
```

Que calcule le prédicat binaire `fact` ?

Dans le même répertoire, lancer ECLiPSe, et compiler `fact.pl` comme expliquer ci-dessus.

Interroger le programme, en évaluant les requêtes de la liste suivante (un ';' tapé après une réponse de l'interpréteur lance la recherche d'éventuelles nouvelles réponses, un retour chariot arrête l'exécution et redonne la main à l'utilisateur). Essayer d'interpréter les résultats obtenus.

```
fact(5,120).
fact(3,7).
fact(6,X).
fact(10,X).
fact(X,1).
fact(X,6).
fact(X,Y).
```

Exercice 2
Donnez des phrases en français pour les faits et règles suivants:

- `homme(socrate).`
- `mortel(X) :- homme(X).`
- `animal(X) :- chien(X).`
- `ornithorynque(X) :- mammifere(X), ovipare(X), amphibie(X).`

Exercice 3
Adam aime les pommes. Clara aime les carottes. Olivier aime les oranges. Les pommes sont des fruits. Les oranges sont des fruits. Les carottes sont des légumes. Ceux qui aiment les fruits sont en bonne santé.

[1] `rlwrap` permet d'utiliser l'historique via les flèches du clavier, ce qui est pratique. En cas d'erreur, il faut effacer le fichier de l'historique.

1. Formalisez ces faits et règles en PROLOG.
2. Quelle est la requête pour "Qui aime les pommes?" ?
3. Quelle est la requête pour savoir qui est en bonne santé ?
4. Comment savoir les fruits que connaît le programme ?
5. Donnez l'arbre de dérivation pour la requête "Qui est en bonne santé?".

Exercice 4
Considérons l'ensemble de faits suivant
(à copier de www.irif.fr/∼haberm/cours/clp/bio.pl

```
bio(louis13, h, 1601, 1643, henri4, marie_medicis).
bio(elisabeth_france, f, 1603, 1644, henri4, marie_medicis).
bio(marie_therese_autriche,f, 1638, 1683, philippe4, elisabeth_france).
bio(louis14, h, 1638, 1715, louis13, anne_autriche).
bio(grand_dauphin, h, 1661, 1711, louis14, marie_therese_autriche).
bio(louis_bourbon, h, 1682, 1712, grand_dauphin, marie_anne_baviere).
bio(philippe5, h, 1683, 1746, grand_dauphin, marie_anne_baviere).
bio(louis15, h, 1710, 1774, louis_bourbon, marie_adelaide_savoie).
bio(louis_dauphin, h, 1729, 1765, louis15, marie_leczcynska).
bio(louis16, h, 1754, 1793, louis_dauphin, marie_josephe_saxe).
bio(louis18, h, 1755, 1824, louis_dauphin, marie_josephe_saxe).
bio(charles10, h, 1757, 1836, louis_dauphin, marie_josephe_saxe).
bio(clotilde, f, 1759, 1802, louis_dauphin, marie_josephe_saxe).
bio(louis17, h, 1785, 1795, louis16, marie_antoinette).
bio(philippe1, h, 1640, 1701, louis13, anne_autriche).
bio(philippe2, h, 1674, 1723, philippe1, charlotte_baviere).
bio(louis_orleans, h, 1703, 1752, philippe, francoise_marie_bourbon).
bio(louis_philippe, h,1725, 1785, louis_orleans, augusta_marie_bade).
bio(philippe_egalite, h, 1747, 1793, louis_philippe, louise_henriette_bourbon_conti).
bio(louis_philippe1, h, 1773, 1850, philippe_egalite,louise_marie_adelaide_bourbon).
```

Définir des requêtes pour:

1. Chercher les femmes mentionnées dans la base de données.
2. Chercher les femmes mentionnées comme mères dans la base de données.
3. Chercher les personnages de la base nés entre 1750 et 1800.
4. Chercher les enfants d'Henry 4 mentionnées comme mères dans la base de données.

Rajouter a `bio.pl` des règles pour les prédicats:
`enfant(enfant,parent)`
`ptenfant(petit-enfant,grand-parent)`
`descendant(descendant,ancetre)`
Définir des requêtes pour:

1. Chercher les petites-filles d'Henry 4 mentionnées dans la base de données.
2. Chercher les descendants de Louis 14 mentionnés dans la base de données.
3. Chercher les ascendants de Louis 17 mentionnés dans la base de données.

Exercice 5

On peut représenter les entiers naturels en *notation unaire* par les termes suivants:
`z, s(z), s(s(z)), s(s(s(z))),...`

- Écrire un prédicat `transform(X,Y)` qui étant donné un entier en notation unaire `X` donne dans `Y` sa valeur. Par exemple

  `?- transform(s(s(s(z))),Y).`

  donne

  `Y=3.`

  Est-ce qu'on peut utiliser ce prédicat pour une requête de la forme `?- transform(X,3)` (qui devrait donner `X = s(s(s(z)))`) ? Sinon définissez un prédicat pour cela.

- Sans utiliser `transform`, écrire un prédicat `somme(X,Y,Z)` qui prend deux entiers en notation unaire et qui calcule leur somme. Par exemple

  `?- somme(s(s(z)),s(z),Z).`

  donne

  `Z = s(s(s(z)))`

  Est-ce que ça marche aussi pour `?- somme(X,Y,s(s(s(z)))).` ?

- Utilisant le prédicat `somme`, définir un prédicat `produit(X,Y,Z)` qui prend deux entiers en notation unaire et qui calcule leur produit. Par exemple

  `?- produit(s(s(z)),s(s(s(z))),Z).`

  donne

  `Z = s(s(s(s(s(z))))))`

- Utilisant le prédicat `produit`, définir un prédicat `puissance(X,Y,Z)` qui prend deux entiers unaires et qui calcule X^Y. Par exemple

  `?- puissance(s(s(z)),s(s(s(z))),Z).`

  donne

  `Z = s(s(s(s(s(s(s(z)))))))`