Introduction aux bases de données relationnelles

TP6 : Premiers contacts avec DATALOG

Le but de cette séance de TP est de vous familiariser avec les environnements DATALOG et PROLOG. Nous utiliserons DES, un freeware de DATALOG du professeur Saenz-Perez de l'université de Madrid. Lui-même, DES repose sur SWI-PROLOG, freeware PROLOG de l'université d'Amsterdam. Vous pouvez récupérez des distributions (linux, windows ou Max OS X) de DES librement sur le site http://des.sourceforge.net et de SWI-PROLOG sur le site http://www.swi-prolog.org. Si vous voulez installer DES sur votre machine personnelle, prenez en compte que nous utilisons la version sans GUI, pour Linux 64 bits avec SWI-Prolog.

DES

- Le binaire DES (version 4.2) est disponible au M5 dans */home/enseign/DES/des/des*. Vous devez saisir le chemin manuellement. En combinant DES avec *rlwrap* (déjà vu en TP1 avec RA) vous aurez des avantages (correction sur la ligne courante et historique). La création d'un alias est conseillée (voir TP1 pour la syntaxe).
- Le signal d'invité "DES>" attend que vous saisissiez un *but* (requete) que le moteur DATALOG tentera de satisfaire.
- Dans l'interprète, vous pouvez obtenir de l' aide en tapant /help.
- Puisque vous n'avez pas encore chargé de base, vous ne pouvez pas encore faire grand chose. Une possibilité est de tester une comparaison 42=42. Observez le résultat. Comparez avec le résultat de 42=2015. L'opérateur d'inégalité vous servira, il est \=
- Dans l'interprète, vous pouvez changer de répertoire avec /cd , faire afficher le répertoire courant avec /pwd, et charger une base (nommée *mabase.dl*) avec /consult mabase.dl. Les commandes suivantes sont equivalentes :

```
DES> /c exemple
DES> /consult exemple
DES> /c exemple.dl
DES> /consult exemple.dl
```

- Lorsqu'un fichier est chargé (par /consult) la base chargée précédemment est "oubliée". Donc un "\consult" réinitialise la base de donnée extensionnelle dans le fichier "consulté".
- Si vous voulez accumuler plusieurs fichiers, utilisez /reconsult a partir du second.
- Avec la commande /listing vous pouvez afficher le contenu de la base de connaissance courante. Ce prédicat peut également prendre en argument le prédicat dont on souhaite connaître la définition courante. Par exemple dans l'exercice suivant :

```
DES> /listing femme
```

- Pour quitter DES, tapez /terminate. Vous vous retrouverez ensuite au niveau de SWI-PROLOG, donc l'interprète vous affiche "?-". Pour quitter SWI-PROLOG, tapez halt. (avec le "." !).

Rappel : en DATALOG*, tout identificateur commençant par une majuscule est une variable.*

1 Histoires de jalousie

A rendre : un fichier *jalousie.dl*.
On considère la base de connaissance suivante :

- *mia* est une femme ;
- *jody* est une femme ;
- *yolande* est une femme ;
- *vincent* aime *mia* ;
- *vincent* aime *pierre* ;
- *marcellus* aime *mia* ;
- *mon_chou* aime *lapin* ;
- *lapin* aime *mon_chou* ;
- X est jaloux de Y s'ils aiment tous les deux une même personne.

Question 1 Saisissez les assertions de cette base dans un fichier [1] sous forme de *faits* DATALOGqui définissent les prédicats suivants : $femme/1, aime/2, est\_jaloux\_de/2$

Question 2 Chargez votre base. Au chargement, votre fichier est évalué syntaxiquement par DES, toute erreur de syntaxe éventuelle est alors annoncée.

Sachez que Datalog mémorise les résultats déjà calculés. Vous pouvez les afficher à tout moment avec la commande /list_et .

Question 3 Vérifiez (en définissant un *but* DATALOG) que *mia* et *yolande* sont des femmes mais que lapin n'est pas une femme.

Question 4 Interrogez DATALOG pour connaître tous les noms de femmes.

Question 5 Interrogez DATALOG pour connaître toutes les femmes que *vincent* aime.

Question 6 Interrogez DATALOG pour connaître tous les hommes que *vincent* aime. Quel problème observez-vous ? Pourquoi ? Comment y remédier ?

Question 7 Interrogez DATALOG pour connaître tous ceux dont *vincent* est jaloux.

Lorsqu'on doit utiliser une variable mais qu'on ne désire pas connaître son instanciation on utilise une *variable anonyme*. Celle-ci est représentée par le symbole _ (tiret bas).

Question 8 En utilisant un variable anonyme, interrogez DATALOG pour savoir si *vincent* est jaloux. En déduire la définition du prédicat correspondant.

2 La boutique

Le script datalog avec la EDB de la boutique est disponible sur moodle. Chargez-le, testez des exemples du cours, egalement sur moodle.

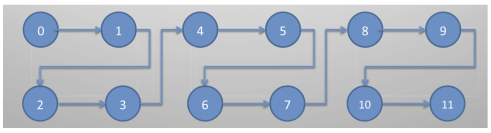
Dans un fichier *requetes_boutique.dl* que vous rendrez, formulez des predicats pour :

Question 9 Les couleurs rares, pour lesquelles il n'y a qu'un seul article.

Question 10 Définissez un prédicat *deuxrouges* tel que la requête $deuxrouges(X)$ rende les noms de fournisseurs d'au moins *deux* articles rouges.

Question 11 Définissez un prédicat *bonmarche* tel que la requête $bonmarche(X)$ rende les noms de fournisseurs d'articles à moins de 10 euros.

3 Graphes



Question 12 Saisissez le graphe de l'image dans un nouveau fichier *graph.dl* (que vous rendrez). Utilisez pour cela le $e/2$ définit en cours (e pour edge en anglais).

Question 13 Utilisez le prédicat $p/2$ pour afficher toutes les paires de sommets connectés, quelque soit la longueur du chemin (p pour path en anglais). Rendez votre requête, et son résultat.

Question 14 Ajoutez un ou plusieurs cycles au graphe. Puis, retestez $p/2$.

Question 15 Pouvez-vous determiner combien de fois un cycle est traversé ?

Question 16 Déclarez un prédicat $impair(X, Y)$ qui est vrai s'il existe un chemin de longueur impaire entre les sommets X et Y. A l'aide d'une requête, affichez les paires de sommets, entre lesquels existe un chemin de longueur impaire.

Question 17 Déclarez un prédicat $injoignable(X, Y)$ qui est vrai lorsqu'il n'existe aucun chemin entre les commets X et Y.

4 Arbre généalogique

A rendre : un fichier *famille.dl*.

Dans cet exercice, vous travaillez la récursivité, à l'exemple d'un arbre généalogique. Le programme P définit quatre prédicats, $pere/2$, $mere/2$, $parent/2$, et $ancetre/2$. Dans la base, on trouve comme fait uniquement des informations sur les directes relations entre parents enfants. Par exemple, pour le fait que Tom est le père d'Amy, on retrouve $pere(tom, amy)$ dans la EDB. Attention à l'ordre de lecture,

1. Utilisez l'éditeur de textes que vous souhaitez, si possible, un qui pemet d'avoir dans une même fenêtre l'éditeur et un shell dans lequel vous pouvez exécuter ce que vous voulez.

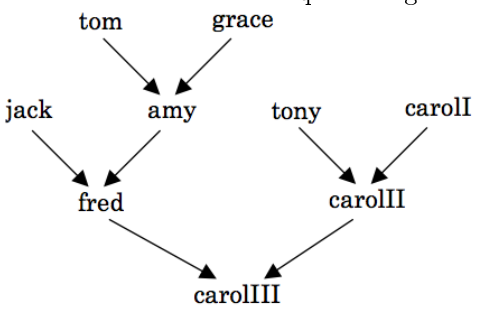
pour laquelle nous convenons que le premier arguement d'un prédicat est toujours le sujet, pour tous les predicats. On pourrait donc lire $pere(X, Y)$ comme X *est le père de* Y, etc.

Les autre relations sont définies par des règles récursives. Prolog peut déduire du programme que la IDB Pour le contient par exemple $ancetre(grace, fred)$, correspondant au fait que Grace est l'ancêtre de Fred. Cette information implicite est obtenue avec une règle récursive, et une règle de base, non récursive. Comparez à la définition de lien direct dans un graphe, et atteignabilité par un chemin.

```
% la EDB
pere(tom,amy).
pere(jack,fred).
mere(grace,amy).
mere(amy,fred).
% le programme (les regles)
parent(X,Y) :- mere(X,Y).
parent(X,Y) :- pere(X,Y).

ancetre(X,Y) :- parent(X,Y).
ancetre(X,Y) :- parent(X,Z), ancetre(Z,Y).
```

Le code ci-haut ne donne qu'un fragment de la EDB. Voici l'arbre complet :



Question 18 Dans un fichier *famille.dl*, saisissez la EDB complète, donc toutes les personnes qui apparaissent dans l'image, pour les prédicats *pere*/2 et *mere*/2. Vous pouvez remplacer les chiffres romains par des chiffres arabes.

Question 19 Saisissez egalement le prédicat *ancetre*/2, et testez-le. Expliquez son fonctionnement, avec 1-2 phrases que vous rendez dans votre compte-rendu.

Question 20 Calculez toutes les conséquences possibles du programme, à l'aide de DATALOG. Rendez les requêtes qui vous permettent de faire cela, ainsi que leur résultats, dans votre compte-rendu.

Question 21 Programmer un prédicat *cousin/2*, qui est vrai pour deux personnes deux cousin(e)s.

Le but du reste de cet exercice est de programmer un nouveau prédicat récursif *mg/2*, à lire, *même génération* . Cette tache se décompose en deux questions :

Question 22 L'idée du *cas de base* est simple : chaque personne est de la même génération qu'elle-même. Pourtant, puisqu'il n'est pas définit ce qu'est une personne, vous devrez penser a ce point. Lorsque votre définition du cas de base sera correcte, elle devrait produire :

```
DES> mg(X,Y)

{
  mg(amy,amy),
  mg(carol1,carol1),
  mg(carol2,carol2),
  mg(carol3,carol3),
  mg(fred,fred),
  mg(grace,grace),
  mg(jack,jack),
  mg(tom,tom),
  mg(tony,tony)
}
Info: 9 tuples computed.

DES>
```

Question 23 Pour le cas récursif, sachez qu'il faut remonter de la racine de l'arbre aux feuilles. Avec la bonne définition, vous obtiendrez l'affichage suivant pour la requête $mg(X, Y)$:

```
DES> mg(Y,M)
{
  mg(amy,amy),
  mg(amy,carol1),
  mg(amy,jack),
  mg(amy,tony),
  mg(carol1,amy),
  mg(carol1,carol1),
  mg(carol1,jack),
  mg(carol1,tony),
  mg(carol2,carol2),
  mg(carol2,fred),
  mg(carol3,carol3),
  mg(fred,carol2),
  mg(fred,fred),
  mg(grace,grace),
  mg(grace,tom),
  mg(jack,amy),
  mg(jack,carol1),
  mg(jack,jack),
  mg(jack,tony),
  mg(tom,grace),
  mg(tom,tom),
  mg(tony,amy),
  mg(tony,carol1),
  mg(tony,jack),
  mg(tony,tony)
}
Info: 25 tuples computed.
```

5 Résolution en SWI Prolog

A rendre : un fichier *crisefinanciere_reponse.dl*

Enregistrez le fichier *crisefinanciere.pl* disponible sur moodle, et lisez-le attentivement. 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 verrez 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 24 Posez une requête permettant de vous faire afficher les dettes.

Question 25 Posez une requête pour afficher quelle fille évite quelle autre. Assurez-vous de faire afficher *toutes* les réponses. Qu'observez-vous ?

Question 26 Comparez le résultat des mêmes requêtes en DES, et expliquez d'où viennent les différences entre les deux systèmes.

6 Analyse d'un programme non stratifiable

Nous revenos sur le paradoxe du barbier présenté en cours. Dans un village, deux catégories d'hommes existent : ceux qui se rasent eux-mêmes, et ceux qui ne se rasent pas eux-mêmes. Le barbier rase tous ceux qui ne se rasent pas eux-mêmes. En Datalog, cela se traduit en :

```
homme(barbier).
homme(maire).
rase(barbier,H) :- homme(H), not(rase(H,H)).
```

— analyser l'information memorisee pour $not(rase(maire, maire)$ et $not(rase(barbier, barbier)$.
— expliquer la negation dans la call table
— reprendre des choses du manuel de DES
— comprendre le probleme des paradoxes logiques dans les programmes non stratifiables

Question 27 Après avoir saisi le code dans un fichier *barbier.dl*, activez en DES le mode *verbose* avec la commande */verboseon* charchez-le et observez le retour de DES. Quel message important obtenez-vous ?

Question 28 Lancez la requête $rase(X, Y)$. Vous obtenez deux réponses, de types differenst : l'une a une valeur fiable (elle a été prouvé vraie), et l'autre une valeur logique est $undefined$. Formulez en francais : quelle est l'infomation prouvée vraie, et quelle est problématique ?

Question 29 Quels sont les faits qui ont été prouvés au cours de l'évaluation de votre requête précédente ? Comment les obtenir de DES ? Quelle est l'information négative prouvée par DES ? Quelle est l'information pour laquelle DES ne donne pas de valeur de vérité ?

Question 30 Ajoutez un nouveau prédicat $est\_rase(X) :- rase(barbier, X)$. Qu'est DES capable de renseigner concernant l'état du barbier, pourquoi et comment ?

7 Les princesses et les tigres (2)

Pour résoudre cet exercise, nous vous invitons à utiliser la correction des princesses et tigres (partie 1) disponible sur moodle.

Le premier prisonnier à réussi à s'en sortir. Le roi, mécontent (il était un peu sadique), modifie les affiches puis en fait venir un deuxième. Voici ce que le nouveau prisonnier pouvait lire :

– 1 –	– 2 –
Il y a une princesse dans cette cellule et un tigre dans l'autre	Il y a une princesse dans une cellule et il y a un tigre dans une cellule

Seulement le roi a aussi modifié les règles du jeu : maintenant, *une des deux affiches ment, et l'autre dit la vérité* ! Répondez aux questions suivantes :

Question 31 Discutez avec vos voisins, essayez de trouver une solution sans formaliser. Ne passez pas plus de 5 minutes là-dessus.

Question 32 Traduisez les deux affiches en formules logiques.

Question 33 Écrivez une formule logique qui inclut aussi bien le contenu des affiches, que les règles du jeux pour ce jour.

Question 34 Implémentez un programme qui résout le puzzle en Datalog, en suivant la solution du premier puzzle.

8 Les princesses et les tigres (3)

Le deuxième prisonnier à réussi à s'en sortir lui aussi ! Le roi, cette fois franchement furieux, appelle un troisième prisonnier :

– 1 –	– 2 –
Les deux cellules contiennent des princesses	Les deux cellules contiennent des princesses

Cette fois-ci, le roi était sûr de son coup : *l'affiche sur la porte de la première cellule dit la vérité si la première cellule contient une princesse, et elle ment sinon.* Pour la deuxième porte, c'est exactement le contraire ! *La deuxième affiche ment si la deuxième cellule contient une princesse et dit la vérité si elle contient un tigre...*

Question 35 Codez ce problème en Datalog. Pour s'en sortir, il faut définir deux prédicats porte_1(X,Y) et porte_2(X,Y) qui expriment la validité de ce qui est écrit sur chacune des deux portes. Ainsi, porte_1(X,Y) coincide avec affiche_1(X,Y) tant que X = princesse, mais dit le contraire de affiche_1(X,Y) si X = tigre.