TP2 : Un puzzle logique et des prédicats récursifs en DATALOG

1 Princesses et tigres (2/4)

Pour résoudre cet exercise, nous vous invitons a utiliser la correction des princesses (1/4) disponible ici.

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

Question 2 Traduisez les deux affiches en formules logiques.

Question 3 Écrivez une formule logique qui inclut aussi bien le contenu des affiches, que les règles du jeux pour ce jour.

Question 4 Implémentez un programme qui résout le puzzle en Datalog, en suivant la solution du premier puzzle.

2 Arbre généalogique

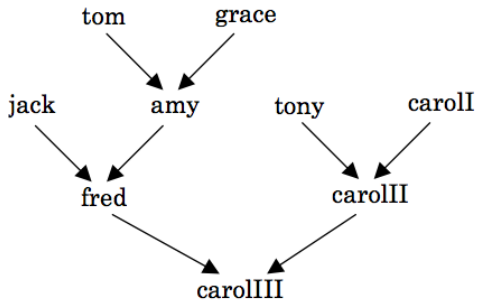
Dans cet exercice, vous travaillez la récursivité, à l'exemple d'un arbre généalogique. Le programme P définit quatre prédicats, $pere/2$, $mere/2$, $parent/2$, et $ancetre/2$. Dans la base, on trouve comme fait uniquement des informations sur les directes relations entre parents enfants. Par exemple, pour le fait que Tom est le père d'Amy, on retrouve $pere(tom, amy)$ dans la EDB. Attention à l'ordre de lecture, 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 5 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. Pour vous faciliter le travail, vous pouvez remplacer les chiffres romains par des chiffres arabes. Saisissez egalement le prédicat *ancetre*/2. Puis, testez ce dernier prédicat, pour vous assurer de le comprendre.

Question 6 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 7 Cette question est a rendre sur papier, au debut de votre prochaine séance de TD. Elle porte sur les arbres de preuve, vus en amphi dans le cours 3. Construisez l'arbre de preuve pour le fait que *Grace* est l'ancêtre de *CarolIII*.

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 8 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 9 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.
```

3 Le bar

Question 10

Implémentez tous les prédicats de l'exercice avec le bar et les buveurs.