Expression Logique et Fonctionnelle . . . Évidemment

TP5

1 Satisfaisabilité (SAT)

Relisez le codage vu en cours pour des formules propositionnelles, disponible ici.

Question 1 Appliquez ce codage pour tester la satisfaisabilité de la formule suivante :

$$(A \textit{ or } B) \textit{ and } (\textit{not } B \textit{ or } C \textit{ or not } D) \textit{ and } (D \textit{ or not } E)$$

Résultat attendu :

```
DES> sat(A,B,C,D,E)
{
  sat(0,1,0,0,0),
  sat(0,1,1,0,0),
  sat(0,1,1,1,0),
  sat(0,1,1,1,1),
  sat(1,0,0,0,0),
  sat(1,0,0,1,0),
  sat(1,0,0,1,1),
  sat(1,0,1,0,0),
  sat(1,0,1,1,0),
  sat(1,0,1,1,1),
  sat(1,1,0,0,0),
  sat(1,1,1,0,0),
  sat(1,1,1,1,0),
  sat(1,1,1,1,1)
}
Info: 14 tuples computed.

DES>
```

Question 2 Quel est le pourcentage des solutions pour la formule précédente, parmis toutes les combinaisons de valeurs booléenes possibles?

Question 3 Appliquez le même codage à la formule

$$A \textit{ and}(B \textit{ or } (D \textit{ and } E))$$

Pour vous simplifier le codage, il est conseillé de réécrire la formule en forme normale conjonctive (CNF sur Wikipedia), dans une première étape.

Résultat attendu :

```
DES> sat(A,B,D,E).
{
  sat(1,0,1,1),
  sat(1,1,0,0),
  sat(1,1,0,1),
  sat(1,1,1,0),
  sat(1,1,1,1)
}
Info: 5 tuples computed.
```

Si le sujet de la satisfaisabilité vous intéresse, lisez l'introduction sur Wikipedia.

2 Analyse d'un programme non stratifiable

Nous revenos sur le paradoxe du barbier présenté en Cours 2. 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 $not(shaves(mayor, mayor)$
- expliquer dans l'enoncé 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 4 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 5 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 6 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 7 Ajoutez un nouveau prédicat $est\_rase(X) : -rase(barbier, X)$. Qu'est DES capable de renseigner concernant l'état du barbier, pourquoi et comment ?

Question 8 Est-il possible de trouver un sous-programme stratifiable d'un programme non stratifiable, et comment ?

3 Jeu d'accessibilité sur les graphes

On considère le graphe disponible ici (image) et en là (en code) :

Sur ce graphe, on joue à un jeu à deux joueurs, avec un seul jeton. A tout moment, le jeton est positionné sur un sommet. Lorsque c'est son tour, chaque joueur doit pousser le jeton le long d'une arête. Le premier joueur qui ne peut pas jouer a perdu.

Question 9 Ecrivez un prédicat DataLog qui indique à partir de quels sommets le joueur dont c'est le tour peut gagner en 1 coup.

Question 10 Ecrivez un prédicat DataLog qui indique à partir de quels sommets le joueur dont c'est le tour peut ne pas perdre après 2 coup (il joue, puis son adversaire joue, et il ne doit pas avoir perdu à ce moment-là).

Question 11 Ecrivez un prédicat DataLog qui indique à partir de quels sommets le joueur dont c'est le tour peut gagner en 3 coups.

Question 12 Donnez le graphe de dépendance de votre prédicat et stratifiez-le.

Question 13 Déterminez un modèle minimal de votre programme. Procedez avec l'algorithme de calcul du point fixe, strate par strate. Rendez un fichier avec le résultat sur PROF.

Question 14 Décrivez une famille $P_1, P_3, \ldots, P_{2k+1}$ de prédicats DataLog qui permettent de déterminer l'ensemble des sommets d'où on peut gagner en $1, 3, \ldots, 2k + 1$ coups. Combien de strates possèdent les programmes en questions ?

Question 15 Est-il possible d'ecrire un predicat qui calcule l'union des P_k pour tous les k ? C'est a dire un predicat qui determine les points de departs gagnants pour le premier joueur, en un nombre arbitraire de coups.