Magistère STIC, première année | 2003–2004
Module *Logique* | Feuille de Projet

Construction des circuits

Les projets sont a priori individuels (ceux qui souhaitent le faire à deux doivent contacter en avance Ralf Treinen ou Hubert Comon). Les programmes seront examinés devant les machines avec les élèves individuellement jeudi, 29 janvier. Envoyez par courrier électronique les sources des programmes en GNU Prolog à Ralf Treinen (`treinen@lsv.ens-cachan.fr`) au plus tard mercredi, 28 janvier.

Il est essentiel que votre programme soit compréhensible. Quelque consignes pour rendre votre programme compréhensible :
- Évitez des noms farfelus pour les prédicats (P351, P352, ...) et les variables (X, X', X", X1", ...). Choisissez plutôt des noms parlants même s'ils sont un peu plus long à taper.
- Structurez votre programme en sous-sections.
- Commentez votre programme. Expliquez l'idée générale de votre programme, et expliquer les prédicats que vous définissez.

Exercice 1

Une *DNF positive* est une formule propositionnelle en forme disjonctive normale qui ne contient aucune négation. Une telle DNF positive peut être représentée en Prolog comme une liste de listes de noms de variables.

Par exemple, la formule $(v \wedge x \wedge y) \vee (y \wedge z)$ peut être représentée par la liste

```
[ [v, x, y], [y, z] ]
```

On appelle une DNF positive *indépendante* si elle ne contient pas deux clauses différentes dont une subsume l'autre.

1. Écrire un prédicat `DNFpos(Vars,Formule)` qui est vrai si `Formule` est une DNF positive où toutes les variables propositionnelles sont des éléments de la liste `Vars`. Vous pouvez supposer que `Vars` est une liste d'atomes différentes.
2. Écrire un prédicat `DNFpos-indep(Vars,Formule)` qui est comme dans la question précédente mais qui de plus vérifie que `Formule` est une DNF *indépendante*.

Exercice 2

On s'intéresse maintenant au problème suivant :

> On souhaite construire un circuit (constitué de portes binaires « et » et « ou », et de portes unaires « non ») avec trois entrées appelées x, y, et z et trois sorties x', y' et z', telle que pour toutes les valeurs des trois entrées, la valeur de x' est la négation de la valeur de x, la valeur de y' est la négation de la valeur de y, et la valeur z' est la négation de la valeur de z.
>
> Il y a une restriction cruciale (sans laquelle le problème est trivial) : vous avez le droit à autant portes « et » et « ou » que vous voulez mais vous n'avez que *deux* portes de négation !

Écrire un programme qui calcule tous les circuits qui répondent à cette spécification. Vous pouvez construire ce circuit comme une combinaison des DNF positives et des portes de négation au lieu de construire le circuit avec des portes binaires.

Remarques : Dans la représentation proposée (avec des DNF positives) il y a une seule solution. La solution n'est pas complètement évidente à trouver à la main, mais une fois trouvée la solution on voit rapidement pourquoi elle est correcte.

Le programme que j'ai écrit trouve la solution, et montre qu'il n'y aucune d'autre, en 2 :30 minutes (programme en Prolog compilé et exécuté sur un PC à 2.4 GHz). Il est pourtant facile d'écrire un programme qui prend beaucoup plus de temps si on fait des mauvais choix.