Devoir de Programmation Fonctionnelle

I) Sujet du devoir

On désire implémenter en `Ocaml` un solveur booléen, c'est-à-dire un algorithme permettant de calculer l'ensemble des solutions d'un système d'équations booléennes.

Un exemple

On considère le système de trois équations booléennes suivant :

{ V_1 OR V_2 = TRUE ; V_1 XOR V_3 = V_2 ; NOT (V_1 AND (V_2 AND V_3)) = TRUE }

Le solveur doit trouver les deux solutions suivantes :

V_1 = TRUE ; V_2 = TRUE ; V_3 = FALSE

V_1 = FALSE ; V_2 = TRUE ; V_3 = TRUE

Type utilisé

Il se nomme `eb` (pour « expression booléenne ») et contient une infinité de variables « V_i », deux constantes « TRUE » et « FALSE », ainsi que les quatre connecteurs « AND », « OR », « XOR » et « NOT ». Voici la définition de ce type en `Ocaml` :

```
# type eb = V of int | TRUE | FALSE | AND of eb * eb | OR of eb * eb | XOR of eb * eb | NOT of eb ;;
```

Principe de l'algorithme

Les principales fonctionnalités de l'algorithme sont les suivantes :
1) Détermination de l'ensemble des variables du système d'équations.
2) Génération de tous les environnements possibles. (Un environnement est une liste de couples (variable, valeur), en l'occurrence de couples (variable booléenne, valeur de vérité)).
3) Evaluation de la satisfaction d'une équation donnée dans un environnement donné. (On évaluera chacun des deux membres, gauche et droit, et on vérifiera l'égalité de ces deux valeurs.)

Illustrons ces fonctionnalités dans le cas de l'exemple précédent.
La première aboutira à la liste [V_1 ; V_2 ; V_3].
La seconde produira la liste (ici abrégée) des huit environnements possibles suivante :
[((V_1, TRUE) ; (V_2, TRUE) ; (V_3, TRUE)) ; … ((V_1, FALSE) ; (V_2, FALSE) ; (V_3, FALSE))]
La troisième permettra par exemple de savoir que la seconde équation évaluée dans l'environnement ((V_1, TRUE) ; (V_2, TRUE) ; (V_3, TRUE)) n'est pas satisfaite car son membre gauche vaut FALSE alors que son membre droit vaut TRUE.

II) Document attendu

Le devoir sera réalisé **en binôme** et donnera lieu à un rapport **papier** à rendre le mardi 14 mai à 14h. Ce document comprendra :

- Une présentation du problème et de sa solution algorithmique en pseudo-code.
- Un listing commenté des types et fonctions `Ocaml` utilisés.
- Des jeux d'essais nombreux et pertinents.

La notation prendra en compte **à parts égales** le contenu (*i.e.* le code) et la forme (*i.e.* le rapport), ce devoir étant autant l'occasion de rédiger un document que celle de programmer en `Ocaml`.