TP Option Info MP/MP* : Satisfiabilité

Formules logiques

Pour représenter les formules logiques, nous utiliserons le typage suivant:

```
(*    Les connecteurs    *)
type connecteur1 = NON ;;      (* connecteur  à un argument*)
type connecteur2 = ET | OU | ALORS ;; (*connecteurs à 2 arguments *)

(* Les expressions booleennes  *)
type expression =   Vrai | Faux
   |Variable of int   (* On numérote les variables *)
   |Noeud1 of connecteur1*expression
   |Noeud2 of connecteur2*expression*expression
;;

(*NB cette implémentation permet d'ajouter facilement d'autres connecteurs *)
```

Représenter les formules suivantes avec les typages ci-dessus, les variables étant de la forme `Vi`, `i` étant le numéro de la variable:

```
Si V0 Alors V1
Si (V0 Et V1) Alors V2
Si V0 Alors Non (V1)
V0 Et Non V0
```

Instanciations

Étant donnée une formule logique `P` dont les variables sont `V0..Vn`, une instanciation pour `P` sera représentée par un tableau de booléen `inst` tel que `inst.(i)` contient la valeur attribuée à `Vi`.
Écrire une fonction `eval : expression -> bool array -> bool` qui prend en argument une formule logique et une instanciation et qui applique l'instanciation à la formule logique.

Ne pas hésiter à utiliser des fonctions auxiliaires pour la lisibilité du code.

Satisfiabilité

Pour tester la satisfiabilité d'une formule logique, on lui applique toutes les instanciations possibles.
L'écriture binaire permet de réaliser une bijection entre l'ensemble des instanciations de n variables et l'intervalle d'entiers $[\![0, 2^n - 1]\!]$.

1. Écrire une fonction `variable_max : expression -> int` qui détermine le numéro de variable maximal pour une expression.
2. Écrire une fonction `instanciation : int -> int -> bool array` telle que `instanciation i m` rend l'instanciation correspondant à l'entier `i` dans un tableau de longueur `m`. Par exemple `instanciation 4 3` renverra le tableau de taille 3 `[|true ; false ; false|]` associé à l'écriture binaire `100` de 4.

3. Écrire une fonction `satisfiable : expression -> bool * bool array` qui détermine si une formule logique est satisfiable et si oui rend un exemple d'instanciation satisfaisant la formule.

   Remarque: *Pour calculer* 2^n*, il est possible d'utiliser l'opérateur de décalage à gauche en faisant* `1 lsl n`*.*

Forme disjonctive

Une variante pour déterminer si une formule est satisfiable est de la mettre sous forme disjonctive (comme dans les sujets CCP).

Cette mise sous forme disjonctive se fait en plusieurs étapes.

Étape 1: On exprime la formule uniquement avec les opérateurs de base `Non`, `Et` et `Ou`.

Étape 2: En utilisant les loi de De Morgan, on transforme la formule de façon à n'avoir que des `Et`, des `Ou` et des littéraux. (Rappel : un littéral est une variable ou la négation d'une variable.)

Étape 3 On développe en distribuant les `Et` sur les `Ou`.

Pour cette étape, on représente chaque «facteur» par un couple de liste d'entiers , la liste des variables «sans négation» et la liste des variables «avec négation», la formule étant rendue sous la forme d'une liste de «facteurs».

Étape 4: À ce stade, il reste des facteurs «contradictoires» et des variables répétées dans les facteurs, on simplifie alors les expressions.

Écrire un ensemble de fonctions qui réalise le travail décrit ci-dessus.