Colle 16

IF-expressions

MPII : Ocaml

Dans ce T.P. nous allons utiliser un nouveau connecteur logique qui, comme le connecteur de Sheffer, permet de définir toutes les formules logiques avec un seul connecteur. On propose ensuite un algorithme de décision qui reconnaît les tautologies. Cet algorithme fournit, de plus, une réfutation lorsque la proposition n'est pas valide, c'est-à-dire une valuation qui donne la valeur **Faux** lorsqu'on l'applique à la proposition.
La satisfiabilité d'une formule F se teste en appliquant le test à $\neg F$, une réfutation de la négation fournissant une valuation qui vérifie F.

I IF-expressions

Formules classiques

On considère les formules logiques vues en cours.
On ajoute 3 nouveaux symboles.

- Les propositions Vrai et Faux. Sémantiquement ce sont des constantes qui prennent la valeur **Vrai** ou **Faux** respectivement pour toute valuation.
  Pour toute formule P, Vrai (resp. Faux) est équivalente à $P \vee \neg P$ (resp. à $P \wedge \neg P$).
  Elles seront considérées comme des formules élémentaires, au même titre que les variables.
- Le connecteur $\rightarrow$. $F \rightarrow G$ est équivalent à $G \vee \neg F$.
  Son but est de permettre des écritures simplifiées.

On représentera les propositions par le type Caml suivant :

```
type proposition = |Atome of string
                   |Vrai
                   |Faux
                   |Et of proposition * proposition
                   |Ou of proposition * proposition
                   |Imp of proposition * proposition
                   |Neg of proposition;;
```

IF-expressions

On propose une autre écriture des propositions logiques sous forme de **IF-expression** :

Définition 1 - IF-expression

Une IF-expression est soit une variable (une proposition atomique), soit Vrai ou Faux, soit une proposition de la forme ite(M, P, Q) où M, P et Q sont des IF-expressions.

La sémantique d'une IF-expression ite(M, P, Q) est déterminée récursivement comme prenant la valeur de P si M est évaluée à **Vrai** et la valeur de Q sinon.
Autrement dit ite(M, P, Q) se traduit par "*si M alors P sinon Q*".

ITE est un acronyme pour If Then Else.
On représentera les IF-expressions par le type Caml suivant :

```
type ifExpr = |Atom of string |True |False
              |Ite of ifExpr * ifExpr * ifExpr;;
```

I.1 Traduction

Exercice 1
Écrire une fonction `prop2if p : proposition -> ifExpr` qui transforme une proposition en une IF-expression équivalente.

Exercice 2
Donner les IF-expressions correspondant aux trois propositions suivantes :
$P_1 : (A \wedge B) \to A$, $P_2 : A \to (A \wedge B)$, $P_3 : \big(A \to (A \to B)\big) \to B$.

I.2 Forme normale

Définition 2 - If-expression normale

On dit qu'une IF-expression est normale si elle est

- une variable,
- Vrai ou Faux
- ou $\text{ITE}(s, P, Q)$ où s est une variable et P et Q sont deux IF-expressions normales.

Exercice 3
Écrire une fonction `estNormale : ifExpr -> bool`
qui teste si une IF-expression est normale.

Exercice 4
Prouver que toute IF-expression est équivalente à une IF-expression normale.

On pourra considérere la fonction ϕ définie récursivement par

- $\phi(\text{Vrai}) = \phi(\text{Faux}) = 1$ et $\phi(s) = 1$ pour toute variable s
- $\phi\big(\text{ITE}(M, P, Q)\big) = \phi(M)\big(1 + \phi(P) + \phi(Q)\big)$

et prouver que les transformations $\text{ITE}(\text{Vrai}, P, Q) \rightsquigarrow P$, $\text{ITE}(\text{Faux}, P, Q) \rightsquigarrow Q$ et $\text{ITE}\big((\text{ITE}(M, P, Q), R, S\big) \rightsquigarrow \text{ITE}\big(M, \text{ITE}(P, R, S), \text{ITE}(Q, R, S)\big)$ diminuent la valeur de ϕ.

Exercice 5
Écrire une fonction `normalise : ifExpr -> ifExpr`
qui transforme toute IF-expression en une IF-expression normale équivalente.

Exercice 6
Donner les IF-expressions normales correspondant à P_1, P_2 et P_3.

II Décision sur les IF-expressions

II.1 Valuation partielle

Définition 3 - Valuation partielle

On appelle valuation partielle une fonction définie sur une partie finie des variables et à valeurs dans dans {VRAI, FAUX}.
Si α est une valuation partielle et P est une proposition, on dit que α est compatible avec P si toutes les variables de P sont dans le domaine de définition de α.
Une expression P est une α-tautologie si elle vérifiée pour toutes les valuations partielles qui prolongent α et qui sont compatibles avec P.

Ainsi P est une tautologie si et seulement si elle est une ω-tautologie où ω est la valuation partielle de domaine vide.
Comme son domaine est fini, on peut représenter une valuation partielle par une liste d'association :

```
type valuation = (string * bool) list;;
```

II.2 Algorithme

On décrit un algorithme qui s'applique à une IF-expression normale selon les règles suivantes.

1. Si P est `True` ou `False` le résultat est immédiat.
2. Si P est une variable `s` alors trois cas se présentent :
   - si $\alpha(s)$ est défini et vaut **V** alors P est une α-tautologie,
   - si $\alpha(s)$ est défini et vaut **F** alors P n'est pas une α-tautologie, réfutée par α,
   - si $\alpha(s)$ n'est pas défini alors P n'est pas une α-tautologie
     et on obtient une réfutation en étendant α par $\alpha(s) = \mathbf{F}$.
3. Si P est une IF-expression `ite(s, Q, R)` alors
   - si $\alpha(s)$ est défini et vaut VRAI
     alors P est une α-tautologie si et seulement si Q est une α-tautologie,
   - si $\alpha(s)$ est défini et vaut FAUX
     alors P est une α-tautologie si et seulement si R est une α-tautologie,
   - si $\alpha(s)$ n'est pas défini alors on définit deux prolongements de α, α_v et α_f, par $\alpha_v(s) = \mathbf{V}$ et $\alpha_f(s) = \mathbf{F}$. P est une α -tautologie si et seulement si Q est une α_v-tautologie et R est une α_f-tautologie.

Exercice 7
Si `P = ite(s, Q, R)` n'est pas une α-tautologie donner une réfutation dans chacun des 3 cas précédents.

On définit donc le type suivant pour gérer le résultat de l'algorithme.

```
type resultat = Tautologie | Refutation of valuation;;
```

Exercice 8
Écrire une fonction `decisionP : valuation -> ifExpr -> resultat` qui décide si une IF-expression normale est une tautologie par rapport à une valuation (partielle) et donne une réfutation si elle ne l'est pas.

Exercice 9
En déduire une fonction `decisionIF : ifExpr -> resultat` qui décide si une IF-expression normale est une tautologie et donne une réfutation si elle ne l'est pas.

Exercice 10
Écrire enfin une fonction `decision : proposition -> resultat` qui décide si une proposition est une tautologie et donne une réfutation si elle ne l'est pas.

II.3 Applications

Exercice 11
Tester les propositions P_1, P_2 et P_3.

II.3.a Les dahuts

Le dahut est une espèce très rare de bouquetin qui a la particularité d'avoir les deux pattes d'un coté plus courtes que les autres. Il vit donc dans la montagne en ayant toujours le sommet du coté de ses pattes courtes. Un dahut est appelé de dextrogyre ou lévogyre selon les cas.
Ils ont d'autres particularités :

- Tout dahut non lévogyre a des rayures noires.
- Tout dahut qui a des oreilles blanches est lévogyre et vit dans les forêts.
- Tout dahut a des oreilles blanches ou n'a pas de rayures noires.
- Les dahuts qui vivent dans les forêts ne mangent pas de mulots.
- Un dahut mange des mulots si et seulement s'il est lévogyre.
- Tout dahut lévogyre a des oreilles blanches.

Exercice 12
Prouver que les dahuts n'existent pas.

II.3.b La princesse

Un chevalier doit partir délivrer des princesses. Arrivé à une intersection, il a le choix entre trois chemins, chacun précédé d'un panneau. Le gardien des lieux lui déclare :
"*Parmi ces trois chemins, l'un mène à une princesse, et son panneau dit la vérité.*
Quant aux deux autres, ils aboutissent à une mort certaine.
Au moins l'un des panneaux ment."
Voici ce qui est écrit à l'entrée de chaque chemin :

- Le deuxième chemin mène à une mort certaine.
- Ce chemin mène à une mort certaine.
- Le premier chemin mène à une mort certaine.

Exercice 13
Comment délivrer la princesse ?