Devoir surveillé
Tautologies propositionnelles

1er avril 1997[1] – 2 heures

Logique propositionnelle. On se donne un ensemble de symboles de propositions atomiques. On définit alors l'ensemble des propositions de la manière suivante :

- Vrai et Faux sont des propositions ;
- si P est un symbole de proposition atomique alors P est une proposition ;
- si P est une proposition alors $\neg P$ est une proposition ;
- si P et Q sont des propositions, alors $P \to Q$, $P \wedge Q$ et $P \vee Q$ sont des propositions.

On représentera les propositions par le type Caml Light suivant :

```
type proposition_atomique == string;;

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

On sait que la logique propositionnelle est décidable ; on peut par exemple examiner les tables de vérité des propositions (Post, 1921). On se propose ici d'implanter un autre algorithme de décision, qui nous donne de plus une réfutation lorsque la proposition n'est pas valide.

1 IF-expressions

Pour cela, on introduit la notion de IF-expression : une IF-expression est soit une proposition atomique, soit Vrai ou Faux, soit une proposition de la forme (`if` M `then` P `else` Q) où M, P et Q sont des IF-expressions. Comme on s'en doute, la valeur d'une IF-expression (`if` M `then` P `else` Q) est la valeur de P si M est Vrai et la valeur de Q sinon. On représentera les IF-expressions par le type Caml Light suivant :

```
type IFexpr =
    Var of proposition_atomique
  | Vr
  | Fx
  | If of IFexpr * IFexpr * IFexpr
```

1. Ce n'est pas une blague !

Toute proposition est équivalente à une IF-expression. En effet, on a les équivalences suivantes :

$$\begin{array}{lcl}
P \vee Q & \Longleftrightarrow & (\texttt{if}\ P\ \texttt{then}\ \textrm{V}\textsc{rai}\ \texttt{else}\ Q) \\
P \wedge Q & \Longleftrightarrow & (\texttt{if}\ P\ \texttt{then}\ Q\ \texttt{else}\ \textrm{F}\textsc{aux}) \\
P \rightarrow Q & \Longleftrightarrow & (\texttt{if}\ P\ \texttt{then}\ Q\ \texttt{else}\ \textrm{V}\textsc{rai}) \\
\neg P & \Longleftrightarrow & (\texttt{if}\ P\ \texttt{then}\ \textrm{F}\textsc{aux}\ \texttt{else}\ \textrm{V}\textsc{rai})
\end{array}$$

□ Ecrire une fonction `prop_if : proposition -> IFexpr` qui transforme une proposition en une IF-expression équivalente.

□ **Applications :** Donner les IF-expressions correspondant aux trois propositions suivantes : $P_1 = (A \wedge B) \rightarrow A$, $P_2 = A \rightarrow (A \wedge B)$ et $P_3 = A \rightarrow (A \rightarrow B) \rightarrow B$.

On dit qu'une IF-expression est *normale* si elle est soit atomique (Vrai, Faux ou une proposition atomique), soit de la forme (`if` (`Var` s) `then` P `else` Q) où P et Q sont deux IF-expressions normales.

□ Ecrire une fonction `est_normale : IFexpr -> bool` qui teste si une IF-expression est normale.

Toute IF-expression est équivalente à une IF-expression normale. Pour le voir, on peut constater que :

$$\begin{array}{lcl}
(\texttt{if}\ \textrm{V}\textsc{rai}\ \texttt{then}\ P\ \texttt{else}\ Q) & \Longleftrightarrow & P \\
(\texttt{if}\ \textrm{F}\textsc{aux}\ \texttt{then}\ P\ \texttt{else}\ Q) & \Longleftrightarrow & Q \\
(\texttt{if}\ (\texttt{if}\ M\ \texttt{then}\ P\ \texttt{else}\ Q)\ \texttt{then}\ R\ \texttt{else}\ S) & \Longleftrightarrow & (\texttt{if}\ M\ \texttt{then}\ (\texttt{if}\ P\ \texttt{then}\ R\ \texttt{else}\ S) \\
 & & \quad \texttt{else}\ (\texttt{if}\ Q\ \texttt{then}\ R\ \texttt{else}\ S))
\end{array}$$

□ Ecrire une fonction `normalise : IFexpr -> IFexpr` qui transforme toute IF-expression en une IF-expression normale équivalente.

□ **Applications :** Donner les IF-expressions normales correspondant à P_1, P_2 et P_3.

2 Décision sur les IF-expressions

On appelle *assignation partielle* une fonction des propositions atomiques dans les valeurs de vérité dont le domaine est fini. On représentera une assignation partielle par une liste d'association :

```
type assignation == (string * bool) list
```

On cherche à décider si une IF-expression est une *tautologie*, c'est-à-dire si elle est vraie pour toute assignation des propositions atomiques, ou si au contraire elle est *réfutable*, c'est-à-dire s'il existe une assignation particulière des propositions atomiques pour laquelle elle est fausse. On définit donc le type `resultat` suivant :

```
type resultat =
    Tautologie
  | Refutation of assignation
```

Pour décider si une IF-expression est ou n'est pas une tautologie, on définit plus généralement la notion *d'être une tautologie par rapport à une assignation partielle* α, de la manière suivante : une IF-expression est une tautologie par rapport à α si elle est vraie pour toutes les assignations qui coïncident avec α sur son domaine. Alors une IF-expression est une tautologie si elle est une tautologie par rapport à l'assignation partielle de domaine vide (la liste vide).

Pour décider si une IF-expression M est une tautologie par rapport à α, on procède de la manière suivante :

- Si M est Vrai ou Faux, le résultat est immédiat.
- Si M est une proposition atomique X, alors deux cas se présentent : soit $\alpha(X)$ est défini et alors M est une tautologie par rapport à α si et seulement si $\alpha(X)$ est Vrai ; soit $\alpha(X)$ n'est pas défini et on a alors une réfutation en étendant α en α' par $\alpha'(X) =$ Faux.
- Enfin, si M est une IF-expression normale (`if (Var` X`) then` P `else` Q), alors deux cas se présentent : soit $\alpha(X)$ est défini et vaut Vrai (*resp.* Faux) et alors M est une tautologie par rapport à α si et seulement si P (*resp.* Q) est une tautologie par rapport à α ; soit $\alpha(X)$ n'est pas défini et on définit α_T (*resp.* α_F) comme l'extension de α avec Vrai (*resp.* Faux) assigné à X. Alors M est une tautologie par rapport à α si et seulement si P est une tautologie par rapport à α_T et Q une tautologie par rapport à α_F.

□ Ecrire une fonction `decision_partielle : assignation -> IFexpr -> resultat` qui décide si une IF-expression est une tautologie par rapport à une assignation ou si au contraire elle est réfutable (et dans ce cas donnant une réfutation).

□ En déduire une fonction `decision : IFexpr -> resultat` qui décide si une IF-expression est une tautologie ou si au contraire elle est réfutable.

3 Décision propositionnelle

□ Déduire de tout ce qui précède une procédure de décision propositionnelle, c'est-à-dire une fonction prenant en argument une proposition, la transformant en une IF-expression équivalente, réduisant celle-ci en une IF-expression normale, puis déterminant s'il s'agit d'une tautologie ou au contraire d'une proposition réfutable (et dans ce deuxième cas, le résultat est une réfutation).

□ **Applications :** Appliquer cette procédure de décision aux trois propositions P_1, P_2 et P_3.

Application : l'Aquarium. Le poisson d'avril est une espèce très rare de poisson qui vérifie les critères suivants :

- Tout poisson d'avril qui ne nage pas en mer chaude a des rayures rouges ;
- Tout poisson d'avril a des nageoires bleues ou n'a pas de rayures rouges ;
- Les poissons d'avril qui vivent dans le corail ne mangent pas de crevettes ;
- Un poisson d'avril mange des crevettes si et seulement s'il nage en mer chaude ;
- Tout poisson d'avril qui a des nageoires bleues nage en mer chaude et vit dans le corail ;
- Tout poisson d'avril qui nage en mer chaude a des nageoires bleues.

On souhaite prouver que les poissons d'avril n'existent pas.

□ Donner une proposition P telle que P est vraie si et seulement si les poissons d'avril n'existent pas.

□ Appliquer la fonction `decision` ci-dessus à cette proposition pour montrer que les poissons d'avril n'existent pas.

4 Si vous avez le temps...

- ☐ Justifier la terminaison de la fonction `normalise`. Indication : pour cela, on pourra exhiber une mesure m sur le type `IFexpr`, c'est-à-dire une fonction de `IFexpr` dans $\mathbb{N}$, qui décroit strictement à chaque appel récursif de `normalise`.