TP4 Caml : Formules booléennes

1 Représentation des formules booléennes

On modélise les formules booléennes par des arbres grâce au type suivant :

```
type binop = Et | Ou | Oubien | Impl | Equiv;;

type formule =
  | Vrai
  | Faux
  | Var  of string
  | Non  of formule
  | Bin  of binop * formule * formule
;;
```

`Vrai` et `Faux` sont des formules constantes ; `Var(`x`)` représente une variable booléenne où x est la chaîne de caractères contenant le nom de cette variable ; `Non(`f`)` représente la formule $\overline{f}$; `Bin(Et,`f`,`g`)`, `Bin(Ou,`f`,`g`)`, `Bin(OuBien,`f`,`g`)`, `Bin(Impl,`f`,`g`)` et `Bin(Equiv,`f`,`g`)` représentent les formules $f \cdot g$, $f + g$, $f \oplus g$, $f \Rightarrow g$ et $f \Leftrightarrow g$. Par exemple, si p et q sont deux variables booléennes, les formules siuvantes

$$f \equiv (p \Rightarrow q) \Leftrightarrow (\overline{q} \Rightarrow \overline{p}) \quad \text{et} \quad g \equiv (p \Rightarrow q) \Rightarrow (q \Rightarrow p)$$

sont représentées par

```
Bin(Equiv, Bin(Impl, Var "p", Var "q"),
           Bin(Impl, Non(Var "q"), Non(Var "p")))
```

et

```
Bin(Impl, Bin(Impl, Var "p", Var "q"), Bin(Impl, Var "q", Var "p"))
```

Pour améliorer l'affichage des formules manipulées, on utilisera la fonction `print_formule : formule -> unit` du fichier `TP4.squ` qui affiche une formule selon la notation infixe en plaçant des parenthèses autour des expressions composées quand c'est nécessaire. Par exemple :

```
#print_formule Bin(Equiv, Bin(Impl, Var "p", Var "q"),
                          Bin(Impl, Non(Var "q"), Non(Var "p")));;
(p => q) => (q => p)
- : unit = ()
```

2 Vérificateur de tautologie

1. Ecrire une fonction `evalue : formule -> bool` qui retourne le booléen associé à une formule, en supposant qu'elle ne contient pas de variables booléennes.
Exemple :

```
#evalue (Bin(Et, Vrai, Bin(Ou, Faux, Vrai)));;
- : bool = true
```

2. Ecrire une fonction `subs : string -> formule -> formule -> formule` telle que `(subs` x f g`)` remplace dans la formule g toutes les occurrences de la variable x par la formule f. Par exemple :

```
#print_formule (subs "p" (Bin(Ou, Var "q", Var "r"))
                         (Bin(OuBien, Var "p", Non (Var "q"))));;
(q + r) ++ Non(q)
- : unit = ()
```

3. Une formule est une *tautologie* si elle est vraie, quelles que soient les valeurs de ces variables booléennes. Par exemple : $p \Rightarrow p$ et $\overline{p+q} \Leftrightarrow \overline{p} \cdot \overline{q}$.
Afin de tester si une formule f est une tautologie, on utilise les règles suivantes :
– si f n'a pas de variable booléenne, `(evalue` f`)` doit être vrai,
– si p est l'une des variables de f, les formules `(subs "p" Vrai)` et `(subs "p" Faux)` doivent être des tautologies.
Ecrire une fonction `tautologie : formule -> string list -> bool` qui teste si une formule est une tautologie, en prenant comme arguments la formule et la liste de ces variables.

4. La fonction précédente manipule des formules que l'on peut souvent simplifier, comme `Vrai` $+ (p \cdot (q \Rightarrow r))$ qui est égale à `Vrai`. Ecrire une fonction `simplifie : formule -> formule` qui réalise ce genre de petites simplifications sur une formule. Tester avec :

```
#simplifie (Bin(Et, Bin(Ou, Faux, Var "a"), Bin(Ou, Vrai, Var "b")));;
- : formule = Var "a"
#simplifie (Non (Bin(Ou, Faux, Non (Var "p"))));;
- : formule = Var "p"
```

5. En déduire une version plus efficace de `tautologie`.

6. Lorsqu'une formule n'est pas une tautologie, on aimerait avoir un exemple de valeurs pour lesquelles cette formule est fausse. Modifier la fonction `tautologie` pour qu'elle renvoie un couple de type `bool*string list` qui contient le booléen indiquant si la formule est une tautologie et une liste donnant un exemple de valeurs booléennes rendant la formule fausse, si cet exemple existe. On obtiendra par exemple :

```
#tautologie3 (Bin(Impl, Var "q", Bin(Et, Var "q", Var "r"))) ["q","r"];;
- : bool * string list = false, ["q=Vrai"; "r=Faux"]
```

7. Améliorer la fonction `tautologie` de façon à ne plus avoir à lui fournir la liste des variables d'une formule.

8. Ecrire une fonction qui exprime une formule sous forme d'une conjonction de plusieurs disjonctions de variables booléennes ou de négations de variables booléennes. En déduire une autre méthode pour vérifier les tautologies.