Travaux Pratiques d'Informatique n°6

L2.2 programmation avancée

Expressions logiques

Dans cette séance nous allons apprendre à manipuler les expressions logiques. L'un des buts de ce T.D. est d'écrire une fonction qui trouve les sous formules d'une expression logique, sans faire l'analyse de l'expression.

De même que pour les expressions arithmétiques, il y a plusieurs manières de noter une expression logique. Dans ce T.D., nous utiliserons la notation polonaise, encore appelée notation préfixe. Une expression logique sera donc stockée comme une chaîne de caractères. Pour simplifier, les chaînes de caractères utilisées seront de taille fixe `MAXCHAINE`. Les variables logiques seront les lettres alphabétiques, minuscules ou majuscules. Le tableau suivant donne les notations des différents connecteurs.

Ou	`|`	Non	`!`	Implique	`>`	Vrai	`1`
Et	`&`	Ou-Exclusif	`%`	Équivalent	`=`	Faux	`0`

FIG. 1 – Notation des connecteurs logiques

▸ **Exercice 1.** Écrire les notations des expressions suivantes (axiomes de Hilbert) :

1. $a \Longrightarrow (b \Longrightarrow a)$;
2. $(a \Longrightarrow (b \Longrightarrow c)) \Longrightarrow ((a \Longrightarrow b) \Longrightarrow (a \Longrightarrow c))$;
3. $(\text{Non}(\text{Non}A)) \Longrightarrow A$.

▸ **Exercice 2.** On rappelle qu'une formule est un arbre syntaxique, et que la chaîne qui lui correspond est le parcours préfixe de cet arbre. Pour savoir si une formule est syntaxiquement correcte, il suffit donc de compter le nombre de branches libres (non encore explorées) dans le parcours de l'arbre. Au départ de la formule, une seule branche est libre, celle qui va à la racine. La formule est terminée quand il n'y a plus de branches libres. Par exemple la formule `|!a>bc` correspond à l'arbre de gauche sur la figure suivante : Le compte du nombre

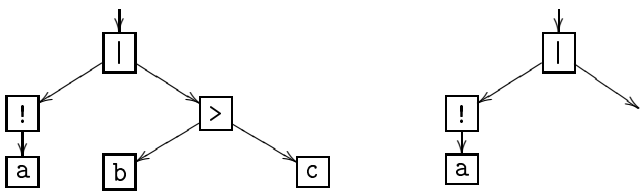

FIG. 2 – Arbres syntaxiques

de branches libres après chaque caractère donne $_1\texttt{|}_2\texttt{!}_2\texttt{a}_1\texttt{>}_2\texttt{b}_1\texttt{c}_0$, ainsi la formule est correcte. Après le `a` il reste bien une seule branche libre comme dessiné à droite.

1. Quelle est la règle qui donne le nombre de branches libres après un caractère en fonction du nombre de branches libres avant ce caractère ?
2. Écrire une fonction qui prend en paramètre une formule et qui répond si cette formule est syntaxiquement correcte.

En fait, on aurait pû partir de n'importe où au milieu de la formule, le calcul précédent répond où s'arrête la sous-formule dont la racine est le caractère dont on est parti.

3. Reprendre la fonction précédente pour en faire une fonction `finformule` qui prend en paramètre une formule et qui retourne le numéro du caractère qui suit celui qui termine la formule.

▸ **Exercice 3.** L'exercice précédent permet donc d'extraire les sous-formules d'une formule donnée. Il est alors facile de tester si une formule est vrai ou fausse en fonction des valeurs données à ses variables.

1. Écrire une fonction `EstVrai` qui prend en paramètre une formule et deux chaînes de caractères contenant les variables vraies d'une part et les variables fausses d'autre part et qui calcule la valeur de la formule.