TP 6 - Arbres, quelques applications

20 janvier 2011

Pour toutes questions, suggestions, remarques ou autres, n'hésitez pas à m'envoyer un mail à edbonnet@hotmail.com en mettant en objet [TP Caml].

1 Introduction

Ce TP propose un assortiment d'exercices indépendants visant à se familiariser avec la structure d'arbre et à découvrir quelques applications.

2 Expression booléenne

On peut représenter une expression booléenne de la logique propositionnelle par un arbre où les nœuds internes sont étiquetés par "$\wedge$", "$\vee$" et "$\neg$" et les feuilles sont étiquetés par "0" et "1". Dans la suite, on pourra utiliser le type arbre suivant : `type 'a arbre = Noeud of 'a * 'a arbre list` et le type `string arbre` pour les expressions booléennes.

Question 1 *Écrire une fonction* `est_correct : string arbre -> bool` *qui dit si l'arbre passé en argument a la syntaxe correcte d'une expression booléenne.*

Question 2 *Écrire une fonction* `evalue : string arbre -> bool` *qui donne la valeur booléenne de l'expression représentée par l'arbre.*

On appelle ici formule booléenne une expression booléenne où les feuilles sont étiquetées par des variables "X_i" et non plus par des valeurs booléennes. La forme normale conjonctive d'une formule est une formule logiquement équivalente écrite comme la conjonction de disjonctions de littéraux (un littéral est soit une variable soit sa négation).

Question 3 *En utilisant les lois de De Morgan, écrire une fonction* `fnc : string arbre -> string arbre` *qui renvoie une forme normale conjonctive d'une formule booléenne.*

3 Minimax

Dans les jeux à somme nulle au tour par tour (comme les échecs ou les dames pour n'en citer que 2), il est possible de chercher un bon coup de la façon suivante : établir l'arbre exhaustif des possibilités sur une profondeur fixée, évaluer les positions dans les feuilles de l'arbre, puis déterminer le coup qui permet d'obtenir la meilleure position quelque soit les coups adverses.

Question 4 *Expliquer comment on peut obtenir ce dernier point et écrire une fonction* `minimaxi : 'a arbre -> ('a -> int) -> int` *qui renvoie le coup choisi et l'évaluation correspondante.*

Question 5 *Établir une situation dans laquelle, on peut se passer d'explorer (et donc de construire) toute une partie de l'arbre. Implémenter cette amélioration. C'est l'élagage alpha-bêta.*

Question 6 *Tester vos algorithmes sur un jeu où le nombre de coups possibles en moyenne est relativement réduit et où établir cette liste de coups est simple.*

4 L'Allumeur de réverbère

On considère le jeu suivant sur un arbre enraciné. Chaque nœud du graphe est un réverbère. Au départ, tous les réverbères sont éteints. On peut allumer un réverbère éteint dès lors que tous ses fils sont allumés, tandis qu'éteindre un réverbère allumé est toujours possible. Le but du jeu est d'allumer la racine de l'arbre de telle sorte qu'on minimise le nombre maximum de réverbères allumés simultanément au cours de la partie. Pour reprendre le petit prince : "Peut-être bien que ce [jeu] est absurde. Cependant il est moins absurde que le roi, que le vaniteux, que le businessman et que le buveur."

Question 7 *Écrire une fonction qui calcule ce nombre minimum de réverbères.*

Question 8 *Écrire une fonction qui donne la stratégie correspondante à adopter.*

On supposera que l'arbre sera donné avec un étiquetage injectif et donc une stratégie pourra être donnée comme une liste d'étiquettes.