TD numéro 2
Modules, Makefile

Programmation avancée, ENSIIE

Semestre 2, 2011–12

On reprend l'exemple des arbres binaires stricts vus dans les TD et TP précédents. On souhaite utiliser une structure d'ensembles pour stocker les valeurs contenues dans les feuilles. On aura donc trois modules : le module `Abs` des arbres binaires stricts comme vu en TP ; un module `Ensemble` définissant les ensembles d'entiers ; et un module `Main` où on écrira une fonction `ensemble_feuilles` qui retourne l'ensemble de valeurs contenues dans les feuilles, ainsi qu'un fonction principale de test qui crée un arbre, calcule l'ensemble de ses feuilles et l'affiche.

1. Quelles sont les relations de dépendance entre les modules ?
2. Écrire le `Makefile` correspondant à ce projet, pour C et pour OCaml.
3. Écrire l'interface du module `Ensemble` (`ensemble.h` et `ensemble.mli`) qui déclarera le type abstrait des ensembles d'entiers et des fonctions :
   - `empty` pour créer un ensemble vide ;
   - `mem` pour tester la présence d'un élément dans un ensemble ;
   - `add` pour ajouter un élément dans un ensemble ;
   - `uni` pour faire l'union de deux ensembles ;
   - `iter` pour appliquer une procédure à tous les éléments d'un ensemble.
4. Écrire le module `Main` (`main.c` ou `main.ml`).
5. Donner une première implémentation du module `Ensemble` (`ensemble.c` ou `ensemble.ml`) à l'aide de listes sans doublons.
6. Quelle est la complexité de la fonction `uni` ?
7. En déduire celle de la fonction `ensemble_feuille` en fonction du nombre de feuilles dans l'arbre.
8. Donner une deuxième implémentation à l'aide d'arbres AVL ; on ne détaillera que la définition du type et la fonction `uni`.
9. En supposant que la complexité de cette fonction `uni` est en $O(n + m)$, en déduire celle de la fonction `ensemble_feuille`.