Calculabilité
R. Rioboo

Travaux Pratiques URM

Vous devrez rendre par courrier électroique à `Renaud.Rioboo@ensiie.fr` un programme accompagné d'un rapport présentant votre implémentation.

Nous allons implanter quelques outils pour manipuler des URM. On reprend les représentations du TP précédent. Nous aurons besoin des utilitaires écrits pendant la séance précédente, notament :

— la fonction `rho` qui calcule le plus petit indice « libre ». C'est à dire non accédé ni modifié par un programme.

— la fonction `normalize` qui rempplace tous les sauts provoquant l'arret d'un programme par un saut à l'instruction qui suit la dernière.

Exercice 1

Pour tester nos fonctions nous utiliserons quelques programmes de base. Par exemple le programme `succ` rretourne le successeur du premier registre, le programme `somme` retourne la somme de ses deux arguments, le programme `constant` prend en argument un entier et retourne cet entier.

Nous pourrons aussi implémenter des fonctions à valeurs booléennes comme le programme `bigger` qui retourne 1 si le premier registre est supérieur au deuxième et 0 sinon.

Implémenter ces programmes.

Exercice 2

Pour nous assurer de la corrections des prgrammes que nous produisons nous devront parfois les lire. Écrire la fonction `string_of_prog` qui produit une chaine de caractères lisible.

Adapter la fonction `run_program` du TP précédent pour écrire une fonction `debug_program` qui affiche le contenu des registres 1 à $\rho(P)$ avant et après l'exécution d'un programme P.

Exercice 3

Nous voulons d'abord implémenter la compisition de programmes. Étant donnés deux programmes P_1 et P_2 nous souhaitons exécuter P_1 puis P_2 en séquance. La fonction `compose` prendra donc deux programmes et retournera un nouveau programme exécutant les deux en séquence.

Nous devons remplacer les instructions $J(r_1, r_2, i)$ de saut de P_2 par des instructions $J(r_1, r_2, n_1 + i)$ où n_1 est la longueur de P_1

Exercice 4

Nous souhaitons maimtenant implémenter le translaté $P\,[a_1, \ldots a_n, \rightarrow r]$ d'un programme P. La fonction `translate` prendra en paramètres un programmes un vecteur d'entiers désignant les registres $a_1, \ldots a_n$ où se trouvent les paramètres d'entrée et un entier désignant le registre r où on souhaite placer le résultat. Nous devons pour celà composer les opérations suivantes :

— transférer le contenu des registres $a_1, \ldots a_n$ dans les registres 1 à n,

— initialiser à 0 les registres de $n + 1$ à $\rho(P)$,

— exécuter P (on n'oublira pas de compter que les étapes précédentes prennent $\rho(P)$ instructions),

— transférer le contenu du premier registre dans le r-ième registre.

Exercice 5

Nous allons maintenant implanter la composition généralisée. Étant donné un programme F implantant une fonction à n arguments et n programmes G_i implémentant des fonctions g_i à k arguments nous voulons implémenter le programme implémentant la fonction à k arguements $x_1, \ldots x_k$ donnée par

$$f(g_1(x_1, \ldots x_k), \ldots g_n(x_1, \ldots x_k))$$

.

Notre fonction `compose` prendra en paramètres le programme F, un vesteur de programmes pour représenter les G_i ainsi que l'arité k des fonctions g_i (l'arité n de f est bien sur a taille du tableau des G_i). Elle retournera un programme.

Nous devons composer les opérations suivantes :

— calculer un registre N « libre » à partir de n, k, $\rho(F)$, $\max(\rho(G_i))$.

— sauvegarder la donnée qui se trouve dans les registres 1 à k dans les registres $N+1, \ldots N+k$.
— Exécuter les programmes translatés des G_i qui prennent leurs arguments dans les registres $N+1, \ldots N+k$ et stockent le résultat dans le registre $N+k+i$.
— Exécuter le translaté de F qui prend ses arguements dans les registres $N+k+1, \ldots N+k+n$ et place le résultat dans le premier registre.

Exercice 6
Nous prendrons un petit langage très simple d'expressions formé de constantes entières (positives) et de sommes d'expressions. Nous voulons traduire ces expressions en des programmes qui les calculent. Pour traduire les constantes nous utiliserons le programme `constant`. Pour traduire une somme nous devrons composer la somme avec chacuns des résultats des traductions des programmes arguements. Écrire la fonction `prog_of_expr`

Exercice 7
Étant donnés trois programmes P_t, P_v et P_f nous voulons écrire un programme P qui exécute P_v ou P_f suivant que l'exécution de P_t donne un résultat non nul.

Implémenter la fonction `if_then_else` qui prend trois programmes en arguments et retourne un programme qui s'exécute conditionellement.

Comment enrichir le langage d'expressions et la fonction `prog_of_expr` pour autoriser des cnditionelles dans les expressions ?