Université Montpellier II
Master d'Informatique
UEMINM 122 Génération de code — 2 heures

R. Ducournau – M. Lafourcade

janvier 2005

Documents autorisés : notes de cours, polys, pas de livre.

Compilation d'automates

Nous disposons d'une machine virtuelle (VM) semblable à celle du cours. L'objectif est de générer, un code de cette VM permettant l'interprétation d'automates, c'est-à-dire la reconnaissance d'un mot par un automate.. On suppose que la VM peut gérer des listes LISP, avec des opérations `CAR` et `CDR` pour récupérer le contenu d'une cellule, ainsi que `CONSP` et `NULL` pour vérifier que le contenu d'un registre est une cellule ou la liste vide (NIL).

Les conventions pour le code d'interprétation des automates sont les suivantes. La donnée (mot à reconnaître) est une liste de caractères (symboles) contenue dans le registre R0. A l'issue de l'exécution, R0 contient l'état final atteint lors de l'exécution de l'automate ou NIL suivant que le mot a été reconnu ou pas. L'état de la pile doit être inchangé.

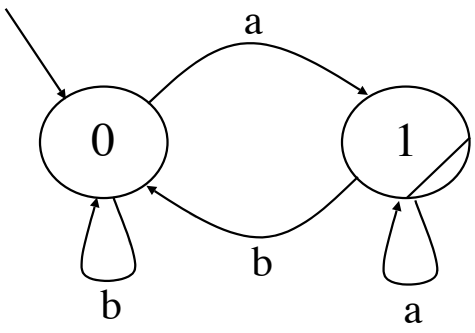
Question 1

Pour éviter les ambiguités, commencez par définir précisément et succinctement les instructions de la VM dont vous vous servez.

(NB vous pouvez traiter cette question à la fin, pour ne pas avoir à décrire toutes les instructions de la VM.)

Question 2

Soit l'automate déterministe suivant :



Commencez par indiquer comment il est possible de traduire les états de l'automate dans la VM. Une pile est-elle nécessaire ? Pourquoi ?

Ecrire le code de la VM correspondant à l'automate donné en exemple ci-dessus, en le commentant.

On dispose, en LISP, d'un type de données abstrait automate, muni de l'interface fonctionnelle suivante :

(auto-etat-liste auto) retourne la liste des états (entiers) de l'automate : pour l'automate de la figure, (0 1)

(auto-init auto) retourne l'état (entier) initial de l'automate ;

(auto-final-p auto etat) retourne vrai si l'état argument est final ;

(auto-tran-list auto etat) retourne la liste des transitions issues de l'état argument, sous la forme d'une liste d'associations dont les clés sont les caractères et les valeurs associées la liste des états que l'on peut atteindre par une transition d'origine l'état argument et de caractère la clé.

Pour l'état 0 et l'automate de la figure, on obtiendrait ((a 1) (b 0)).

Question 3

L'objectif est d'écrire la fonction LISP `auto2vm` qui retourne le code VM correspondant à un automate déterministe (c'est-à-dire un code voisin de celui que vous avez écrit pour la question précédente et l'automate de la figure).

1. Spécifier le principe de la génération : comment traduire les états, les transitions, les états finaux, l'état initial, etc.
2. Décomposer le problème en définissant des fonctions annexes pour traiter séparément, chaque transition, chaque état, etc.
3. que faut-il faire pour que la fonction `auto-vers-vm` retourne le code VM d'une fonction à un argument, le mot à reconnaître, et retournant l'état final ou NIL ?

Question 4

On souhaite maintenant traiter le cas des automates non-déterministes. Rappelez d'abord leur définition.

Indiquer les différences par rapport au cas déterministe. Une pile est-elle nécessaire ? Pourquoi ?

Question 5

On veut maintenant générer une fonction LISP avec une fonction `auto-vers-lisp`. Sans écrire complètement `auto-vers-lisp`, spécifier le principe de la génération : comment traduire les états, les transitions, les états finaux, l'état initial, etc. compte-tenu du langage cible, LISP.