Projet IPI 2021-22

Programme qui éxécute des automates LR(1)

Arthur BABIN

Janvier 2022

1 Présentation du sujet

Premièrement un automate LR1 est donné par les éléments suivants :

- un ensemble fini d'états ;
- un état initial ;
- un alphabet d'entrée ;
- un autre alphabet de symboles dit non-terminaux ;
- une fonction action qui à chaque état et chaque lettre associe une action : cette action peut être soit Rejette, soit Accepte, soit Décale, soit Réduit ;
- une fonction (partielle) décale qui à un état et une lettre associe un état ; cette fonction n'a besoin d'être définie que quand l'action associée à l'état et la lettre est Décale ;
- une fonction (partielle) réduit qui à un état associe un entier et un symbole non-terminal ; cette fonction n'a besoin d'être définie que quand l'action associée à l'état et la lettre est Réduit ;
- une fonction (partielle) branchement qui à un état et un symbole non-terminal associe un état.

En second lieu un fichier contenant une description d'un automate LR(1) respectera le format suivant :

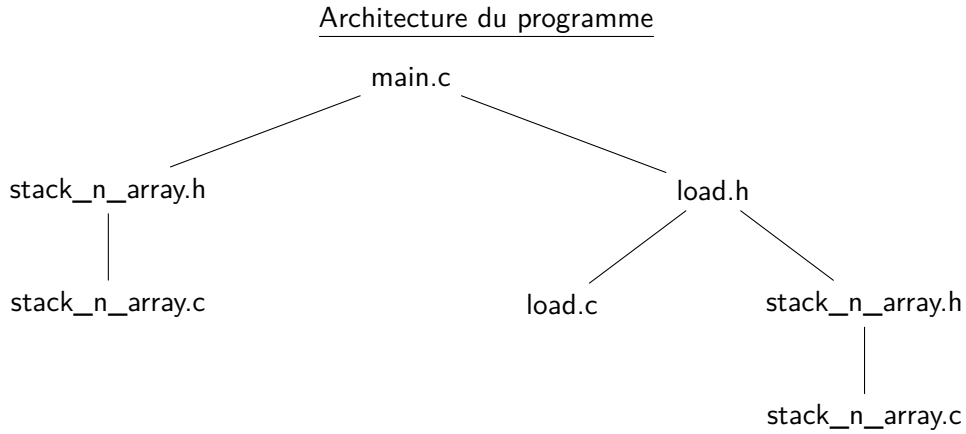
- une première ligne contenant a n où n est le nombre d'états de l'automate ;
- n $\times$ 128 octets représentant les valeurs de action(s,c) pour tout état s et toute lettre c (dans l'ordre action(0,0) action(0, 1) action(0, 2) ... action(0, 127) action(1, 0) action(1, 1) ... action(n,127)), le tout suivi d'un retour à la ligne ;
- n octets représentant la première composante de réduit(s) pour tout état s, le tout suivi d'un retour à la ligne ;
- n octets représentant la deuxième composante de réduit(s) pour tout état s, le tout suivi d'un retour à la ligne ;
- une séquence de groupement de trois octets représentant la fonction partielle décale ; un groupement de trois octets s c s' indique que décale(s,c) = s' ; cette séquence se terminera par le groupement de trois octets '55' '55' '55' ;
- une séquence de groupement de trois octets représentant la fonction partielle branchement ; un groupement de trois octets s A s' indique que branchement(s,A) = s' ; cette séquence se terminera par le groupement de trois octets '55' '55' '55'.

Ainsi il s'agit de manipuler un fichier .aut en utilisant des structures comme les piles et les tableaux pour accéder aux informations de l'automate afin d'ensuite pouvoir appliquer l'algorithme de reconnaissance des automates LR(1) suivant:

- Un tel automate fonctionne à l'aide d'une pile d'état initialisé avec l'état 0.

- Le comportement d'un tel automate est le suivant : si l'état courant (sommet de la pile) est s et la lettre d'entrée est c, on agit en fonction de action(s,c) :
    - Si c'est Rejette : on s'arrête, le mot ne fait pas partie du langage.
    - Si c'est Accepte : on s'arrête, le mot fait partie du langage.
    - Si c'est Décale : on empile décale(s,c), on passe à la lettre suivante de l'entrée.
    - Si c'est Réduit : si réduit(s) vaut (n, A), on commence par dépiler n états ; l'état courant devient alors s' ; on empile alors branchement(s',A) ; on ne passe pas à la lettre suivante de l'entrée.

2 Implémentation



- stack_n_array :
    - implémentation des piles (création vide, est vide ?, affichage, empilement, dépilement)
    - implémentation des tableaux 1D (création d'un tableau vide d'une taille en paramètre, affichage)
- load :
    - un struct pour l'automate avec comme paramètres :
        - nb_states le nombre d'états de l'automate
        - length_shift la longueur du tableau shift
        - length_connect la longueur du tableau connect
        - le tableau action de $longueur = nb\_states * 128$
        - le tableau reduce de $longueur = nb\_states$
        - le tableau shift de $longueur = length\_shift$
        - le tableau connect de $longueur = length\_connect$;
    - une fonction pour charger les paramètres dans un automate vide en parcourant un tableau de caractères correspondant à un fichier .aut
    - une fonction pour l'algorithme de reconnaissance des automates LR(1) qui affiche si l'entrée a été acceptée et si non à quel endroit de l'entrée celle-ci a été refusée
- main.c :
    1. on initialise une pile vide et un automate vide
    2. on stocke l'entrée de l'utilisateur dans la pile et on charge l'automate
    3. boucle infinie de reconnaissance LR(1)
    4. nb: on utilise l'automate tout au long de l'éxécution du programme. Le dump mémoire à la fin de son éxécution permet

3 Limites du programme

- Le programme est très vulnérable face aux mauvais fichiers. Si l'utilisateur n'entre pas le chemin d'accès vers un fichier .aut correctement écrit alors il y aura des erreurs dans le programme.
- Forcer le Ctrl + C pour sortir du programme à cause de la boucle infinie
- J'ai préféré utiliser des tableaux 1D pour implémenter les matrices des fonctions de l'automate ce qui peut rendre la lecture du programme compliquée à certains endroits
- Si l'automate est très grand, comme ses paramètres sont stockés dans des tableaux ont peut se retrouver avec un manque de mémoire et il serait dans ce cas judicieux d'opter pour une version dynamique qui parcourerait le fichier .aut pendant l'application de l'algorithme LR(1)
- Un gestionnaire d'erreur plus complet serait aussi une bonne façon d'améliorer le programme.