Rapport de projet
`automaton`

Thomas Roiseux

2 janvier 2022

Résumé

Le programme `automaton` est un programme qui lit des automates LR1 et ensuite lit des mots afin de reconnaitre s'ils font partie ou non du langage reconnu par l'automate. Pour cela, il lit un fichier de description d'automates puis lit des mots sur l'entrée standard.

Ce rapport présente la création du projets et les choix qui y sont relatifs, ainsi que les problèmes techniques et les solutions trouvées. Enfin, il se termine par la présentation des limites du programme.

Chapitre 1

Choix, problèmes et solutions

1.1 Choix majeurs

Par souci de lisibilité, j'ai créé une énumération afin d'avoir un type booléen en plus. J'ai également codé un fichier qui contient des fonctions sur les chaînes de caractères.

1.2 Problèmes et solutions

1.2.1 Comparaison de chaînes de caractères

1.2.1.1 Problème

Lors de son fonctionnement, `automaton` a besoin de comparer efficacement des chaînes de caractères pou savoir si elles sont égales. Or la comparaison via l'opérateur `==` n'a pas fonctionné correctement.

1.2.1.2 Solution

Pour éviter ce problème, j'ai utilisé une fonction de la libraire standard, `strcmp`, qui compare des chaines de caractères efficacement. Cette fonction est dans l'en-tête `string.h`.

1.2.2 Génération du fichier DOT

1.2.2.1 Problème

Le fichier DOT doit contenir les différentes flèches liées aux branchements dans l'automate. Or ces flèches ont été générées plusieurs fois dans certains fichiers.

1.2.2.2 Solution

Pour éviter cela, j'ai intégré un système qui élimine les branchements qui partent de l'état en cours de traitement quand il faut dépiler des états. Il fonctionne en comparant l'état en cours avec les états qui contiennent des branchements.

1.2.3 Sortie du programme

1.2.3.1 Problème

Une fois que le programme fonctionne, il faut un moyen de le fermer pour qu'il libère la mémoire. Pour cela, il faut un sysgtème qui différencie les mots et reconnait un mot de sortie.

1.2.3.2 Solution

Le programme va distinguer les mots « exit » et « quit », qui seront des entrées qui peuvent faire quitter le programme. En effet, ces mots restent des mots qui pourraient être reconnus. Pour cela, il va demander, si l'un de ces mots est saisi, si le programme doit se fermer.

1.2.4 Vérification du fichier d'automate

1.2.4.1 Problème

`automaton` attend le fichier d'automate dans un format bien particulier. Si celui-ci ne convient pas, il ne pourra pas lires les informations nécessaires sur l'automate.

1.2.4.2 Solution

Pour éviter cela, lorsque le programme ouvre et lit le fichier, il vérifie si le fichier contient toutes les informations nécessaires et dans le bon format. Si le fichier convient, alors il lance la saisie reconnaissance des mots, dans le cas contraire, il s'arrête en produisant une erreur.

En outre, la lecture du fichier nécessite que celui-ci possède des séquences de trois ectes qui se terminent par une séquence bien définie. Pour les lire en intégralité, j'ai choisi de lire trois octets à la fois en testant si ces trois octets sont égaux à la séquence de terminaison. Enfin, on sait que, pour un automate de n états reconnaissant 128 caractères, il ne peut y avoir plus de $n \times 128$ groupes d'octets à lire, ce qui donne une condition pour éviter une boucle infinie.

1.3 Fonction de lecture du fichier d'automate

Avant de créer l'automate, le programme vérifie que le fichier est conforme. Cette vérification est effectuée par la fonction `readfile` du fichier `automaton.c`. Elle va vérifier la conformité de :

1. La première ligne du fichier.
2. La ligne contenant les différentes actions et leurs valeurs.
3. Les deux parties de la fonction de réduction et leurs valeurs.
4. La fonction de décalage.
5. La fonction de branchement.

Chapitre 2

Limites du programme

Le programme `automaton` présente plusieurs limites :

1. Il ne peut reconnaitre que les caractères de la table ASCII.
2. Il suppose que chaque automate ne peut pas avoir plus de 256 états.
3. Le tri pour le branchement dans le fichier DOT ne semble pas optimal.
4. Il suppose également qu'on ne peut pas dépiler plus de 256 états à chaque fois.

Table des matières