Rapport du projet d'IPI Nicolas Desan

Traitement des automates LR(1)

Sommaire :

I) Introduction :

L'objectif est d'implémenter un programme qui exécute un automate LR(1) afin de reconnaître des langages. Le programme lira un fichier en paramètre de l'exécutable qui contient un description de l'automate LR(1). Ensuite, sur l'entrée standard le programme lira un ensemble de lignes et affichera « Accepted » ou « Rejected » en fonction de si l'ensemble des lignes est respectivement reconnu ou non par l'automate.

De plus, l'automate à besoin d'une pile qui contient initialement un unique élément qui sera l'état initial (cf pile.h).

II) Piles

Les piles sont nécessaires pour traiter les automates LR(1). En effet, pour réaliser réduit(s)=(n, A) par exemple, nous avons besoin de dépiler n fois la pile d'état afin de récupérer l'état voulu s'.

Alors, c'est pour cette raison que j'ai établi la partie automate_traitement.h comme dépendante de la partie pile.h (cf Makefile et automate_traitement.h).

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include "pile.h"
4   #include "automate_traitement.h"
```

Les piles en question contiennent des états sous forme d'entiers et la structure de la pile est constituée d'un état top et d'un tableau d'états d'une taille maximal SIZE = 256.

```
3   #define SIZE 256
```

```
9   struct pile{
10      etat top;
11      etat pile[SIZE];
12  };
13
14  typedef struct pile pile;
```

On a donc les fichiers pile.h et pile.c qui contiennent les fonctions :

- pile empty() qui renvoie une pile vide
- pile initial() qui renvoie une pile avec l'état initial 0
- pile empilement(pile *p, etat e) qui empile l'état e à la pile p
- pile pop(pile *p) qui renvoie et dépile le dernier état de la pile p

III) Automate traitement

Les fichiers automate_traitement.c et automate_traitement.h traitent les fonctions partielles de l'automate. Cependant, j'ai fait le choix d'établir les fonctions partielles avec un tableau et/ou une pile en entrée.

Ainsi, on a donc les fonctions partielles :

- int action(etat s, lettre e, int* act) qui renvoie la (s*128 + e)ème valeur du tableau act (avec e convertie en entier).
- etat decale(int* dec, pile *p, etat e, lettre l) qui renvoie la valeur du tableau dec correspondant à l'état e. La fonction empile par ailleurs l'état e à la pile p.
- etat reduit_entier(pile *p, etat e, int* red1) qui renvoie la première composante de la variable reduit(s) décrite dans le projet. Pour cela, on dépile un n fois la pile p et on renvoie la dernière valeur dépilé (n étant l'entier correspondant à red1[e]).
- int réduit_sym_non_term(etat e, int* red2) qui renvoie la deuxième composante de la variable reduit(s) décrite dans le projet. Pour cela, la fonction renvoie red2[e].
- etat branchement(int* branch, etat s, symbole sym) qui renvoie l'état s' = branchement(s,A) décrite dans le projet. Pour cela on cherche i tel que branch[i]=s et on renvoie branch[i+2] pour obtenir s'.

Par ailleurs, j'ai fait le choix d'établir les fonctions :

- int nombre_etats(char* aut) qui renverra dans le main.c le nombre maximal d'états de l'automate en renvoyant l'entier de la première ligne de l'automate LR(1).
- int comportement(pile *p, etat s, lettre c, int *a, int *red1, int *red2, int *branch) qui établit récursivement le comportement de l'automate sur une lettre en renvoyant l'entier correspondant à « Accepted » ou « Rejected ».
- void print(int a) qui renvoie « Accepté » si a=0 , Rejeté si a=1 et « Erreur » sinon.

IV) Main :

Le fichier main.c contient respectivement les fichiers pile.h et automate_traitement.h.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>
#include "pile.h"
#include "automate_traitement.h"
```

Le fichier main.c va être le fichier qui va directement traiter l'automate de l'entrée de l'exécutable pour lire le texte de l'entrée standard.

Le programme renvoie un message d'erreur s'il y a le mauvais nombre d'arguments passé en entrée.

Ensuite, si l'entrée contient effectivement un unique automate, alors int main() va récupérer l'ensemble des tableaux avec les valeurs de action, decale, réduit, branchement et le nombre d'états de l'automate passé en entrée.

La première ligne de l'automate (obtenue avec fgets) contient le nombre d'état de l'automate que l'on obtient avec nombre_etats de automate_traitement.h.

Les valeurs d'actions sont obtenues une à une grâce à fgetc et stockées dans le tableau action.

Les valeurs de la première composante de réduit obtenues grâce à fgetc de la même manière sont stockées dans le tableau red1.

Les valeurs de la deuxième composante de réduit obtenues aussi avec fgetc sont stockées dans le tableau red2.

Enfin, les valeurs de branchement et decale sont stockées respectivement dans branch et dec.

Cependant, les valeurs de branch et dec obtenues de cette manière semblent fausses.

V) Conclusion :

Le programme compile et reçoit des valeurs pour action, decale, réduit_entier, réduit_sym_non_term et branchement. Cependant, les valeurs de decale et branchement semblent fausses (valeur de -1 dans le tableau).

Finalement, le programme renvoie toujours « Accepté ». L'erreur provient probablement de l'obtention de branchement, decale ou de l'exécution de la fonction comportement du fichier automate_traitement.c.