**Rapport:Compilateur d'automate**

**Part 1:Introduction**

Dans ce projet,I'objectif est claire:Écrire un **compilateur d'automate** qui peutatteindre les fonctions suivantes:

·Tout d'abord,il lit un fichier contenant la description de lautomate.

·Ensuit,il produit un code capable de reconnatre les mots qui lui sera fourni en machine virtuelle.

·Enfin,il exécute les machines virtuelles écrites.

Afin deréaliser ces fonctions et bien tracer le comportement delautomate (mode debug),j'ai effectué des analyses dans les cinq parties suivantes.

**1.Analyse lexicale**

**2.Analyse syntaxique**

**3.Analysesémantique**

**4.Compilation**

**5.Exécution.**

**Partie 2:Analyse lexicale**

**Méthode d'extraction des lexèmes**

Aprèsavoir obtenu un fichier txt d automate,jedois séparer tous les éléments en lexèmes.je dois doncd'abord définir des différents types de lexèmes.Ici,j'aidivisé les six types suivants:

//utilser enum pour numeroser les different types

typedef enum lexeme\_t

{

**Symbole,** //[=()[]{}]+Tous les symboles sauf les guillemets simples,les guillemets doubles et les symboles de commentaire.

**Entier,** //[Entiers pas entre guillemets

**Chaine,** //les caractères simples entre guillemets simples

**caractere,** //les caractères entre guillemets

**Fleche** //"-"

}lexeme\_t;

**Mot\_clef,**//Il comtient 5 mots:Automate etats initial final transitions

En parallèle,j'aiconstruit les structures suivantes pour représenter les différentes caractéristiques du lexème:

·Structure de type et expression de chaque lexème:

//structure de type et expression de chaque lexeme

typedef struct lexeme

{

**lexeme\_t type;**

char \* **expression;**

**}1exeme;**

·Structure d'adresse et contenu(type et expression)de chaque membre:

//structure d'adress et contenu(type et expression)de chaque member

typedef struct indice

**]exeme\*content:**

int\* address

//int\*tail;

}indice;

· Structure d'une listedes tous les lexèmes:

//structure d'une liste des tous les expressions

typedef struct mot\_liste

{

int length

char \*liste;

}mot\_liste;

Avant de faire correspondre le lexème,je dois supprimer les informations "inutiles"dans le

texte,c'est-à-dire le commentaire correspondant.Grâce à l'observation,j'ai constaté qu'il existe deux types de commentaires,qui sont marqués par'//sur une ligne et marques par'/\*\*/'sur

plusieurs lignes.je dois donc supprimer ces deux types de commentaires séparément.Ici,j'ai

utilisé la méthode du double pointeur pour supprimer tous les commentaires.L'exemple de code est le suivant:

while ((char\_cur = fgetc(fp)) && char\_cur!=- 1){

//Tout d'abord nous négligeons les commentaires sur une ligne(par '//')

if (char\_tmp =='/' ){

if (char\_cur =='/' ){

while (Cchar\_tmp = fgetc(fp)) && char\_tmp!=- 1){

if (char\_tmp =='\n' ) break ;

}

//A la fin du fichier

if(char\_tmp ==EOF){

1iste\_sans\_commentaire- >1iste[count ++] ='\n' ;

break;

}

}

//Ensuite nous négligeons les commentaires sur plusieurs lignes(par

'/\*...\*/')

else if (char\_cur =='\*' ){

while ((char\_cur = fgetc(fp)) && char\_cur != -1){

if(char\_cur =='/'&& char\_tmp =='\*' ){

char\_tmp ='\n' ;

break;

}

//mis à jour char\_tmp

else {

char\_tmp =char\_cur;

}

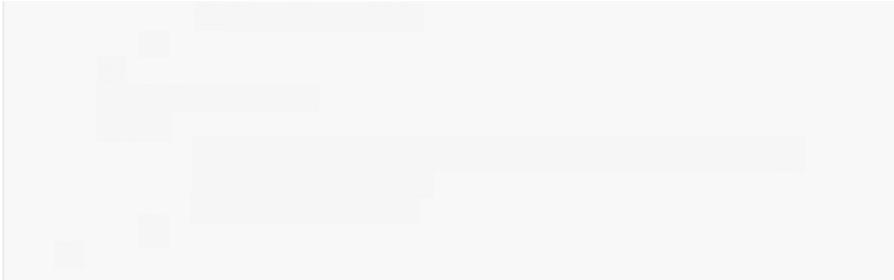
}

}

else {

1iste\_sans\_commentaire->1iste[count++]=(char)char\_tmp;

//mis à jour char\_tmp

char\_tmp =char\_cur;

}

}

//Pas commentaires

else{

liste\_sans\_commentaire->1iste[count++]=(char)char\_tmp;

//mis à jour char\_tmp

char\_tmp =char\_cur;

}

}

En même temps,dans le processus de suppression des commentaires,je lis chaque lexème du fichier.txt dans liste sans commentaire

1iste\_sans\_commentaire->1iste[count++]=(char)char\_tmp; liste\_sans\_commentaire->1ength =count;



Après avoir supprimé les commentaires et lu tous les lexemes,j'ai commencé à réfléchir à la manière de faire correspondre les lexemes avec leurs types correspondants.Mon idée initiale

était de trouver le type correspondant de chaque lexeme grâce à une traverséede boucle.Après avoir essayé,il s'estavéré être un peu encombrant.Après avoir revu les leçonsenseignées par le professeur,jai soudain réalisé que les expressions régulières peuvent être utilisées pour

résoudre ce problème:

regex\_t myregex\_mot\_clef;

regex\_t myregex\_symbole;

**regex\_t** myregex\_entier;

regex\_t myregex\_chaine;

regex\_t myregex\_caractere;

regex\_t myregex\_fleche;

**int** num;//Le nombre de lexemes dans le texte

//les expressions régulières

**char\*** mode\_mot\_clef ="(Automateletatslinitiallfinalltransition)"; **char\*** mode\_entier ="[0-9]+";

**char\*** mode\_chaine ="\"[^\".]+\"" ;

**char\*** mode\_caractere ="`[ʌ `.]+`" ;

**char\*** mode\_fleche ="[→]" ;

Ensuite,j'ai utilisé regular\_matching cette fonction pour obtenir une correspondance

d'expression régulière.Au cours du processus,j'ai découvert qu'il y avait un probleme,c'est-à-dire que les trois types de parenthèses,lessignes égaux et les virgules ne correspondaient pas.Afin de résoudre ce probleme,je utilise estsymbole cette fonction pour faire correspondre de tels

symboles d'une manière parcourant une boucle.La mise en œuvre defonctions spécifiques peut se référer au code.

Ensuite,j'ai complété l'appariement de tous les lexemes:

//Faire correspondre le lexème par type

regcomp(&myregex\_fleche,mode\_fleche,REG\_EXTENDED);

regular\_matching(liste\_sans\_commentaire,myregex\_fleche,Fleche,member);

regcomp(&myregex\_mot\_clef,mode\_mot\_clef,REG\_EXTENDED);



… … … … … … … … … … … … … … … … …

,expression:

1s

"expresston:[

,l'expresston:trans

regular\_matching(liste\_sans\_commentaire,myregex\_mot\_clef,Mot\_clef,

member);

regcomp(8myregex\_entier,mode\_entier,REG\_EXTENDED);

regular\_matching(liste\_sans\_commentaire,myregex\_entier,Entier,member);

regcomp(&myregex\_chaine,mode\_chaine,REG\_EXTENDED);

regular\_matching(liste\_sans\_commentaire,myregex\_chaine,Chaine,member);

regcomp(&myregex\_caractere,mode\_caractere,REG\_EXTENDED);

regular\_matching(liste\_sans\_commentaire,myregex\_caractere,Caractere, member);

estsymbole(liste\_sans\_commentaire,member);

Après avoir terminé la correspondance et le stockage de tous les lexèmes,j'envisagede sortir les résultats.Je n'ai pas suivil'ordre de chaque lexeme dans le fichier txt dans le processus de

correspondance et de stockage,mais je connais Iadresse dechaque lexème.Je complètedonc la sortiedechaque lexème dans l'ordre en triant les adresses dela plus petite à la plus grande.Ici, jechoisis d'utiliser le tri par sélection.

Apres avoir écrit le fichier Analyse\_lexicale.c,jai ecrit un fichier de test test\_AL.c pour

tester les résultats de sortie des trois fichiers txt.En fin de compte,jai obtenue la résultat correcte.

**Résultat du test**

Les résultats du test decode sont lessuivants:

**Pour Zpile.txt:**

etlannoghetlanmo-VLrtuateox:-/plt/projets gcc test\_AL.c -o test\_AL

hettannoghetianmo-VtrtualBox:-/plt/projets ./test\_AL

zpile.txt

Ficher Non:

ht n extst pas des errors!

1:type:Not\_clef,L'expresslon:Autonate

2:type:synbole,,L'expresston:(

tyy**p**4::SE:yn,**l**xl'ni、:)

y**p**ee:S:vynolele,l,l'expresston:e[xpresston:

type7:: Not\_clef,l'expresston:etats

typeB:: Symbole,l'expresslon:=

g;type! Symbole, l expresston:[

量

**1****:type:** Chatne,

12: type:

13: pe:

14: type:

15: type:

16: type:

ype: mbole

18:

19: 5ol

流 受

;

26: ;

离斋离离言离亲高

照

,tepression

照

34:

t'expression:

l'expression:o

37: ,soression:

38: type:

type:

39: 48: 41:

tur

e

texpression:)

l'expression:



**Figure 1:résultats des test de Zpile.txt (partiel)**

**Pour Upile.txt:**

hettannoghetianmo-VLrtualBox:~/plt/projets ·/test\_AL

6: **typ**mt olel,e'exx**p**rreesssst::e[tats

Ficher Non

uptle.txt

n n'exist pas des errors!

1:type:Mot\_clef,l'expression:Automate

VPe PT

3:type:tntier,Uexpression:1、

4:type:Synbole,l'expression:)

5: type: Synbole, l'expresston:=

87: :type:Synbole,l'expression:=

e Synbole,l'expresston:[

**1****typ** ,l'ee'esssitoio

13:type: Symbole,,l'expression:

14:type: l'expresston: *"三"*

15:type: symbole,l'expression:]

16:type:N ef, 1'expression:initial

**1**::**typ**ee:: Entler, Ul'**e**xx**p**rrlin:

tyyppee:: Nof'l

21:type: Synbole,l'expresston:[

22:type:Entier, l'expression:1

24:ttype:Mot cle,l'expression:transition

23:type: mbole 1'expression:]

2t: Synbole,ersosn:;

27:type: Synbole,,l'expression:(

28:type: Entler,l'expresston:o

29:ttype: ech,t'expresston:-

30: type: Entler, l'expresston:o

cotleer,'xetno:;n:a

33:type: 5vmbole. l'exnressdnn.

e,ol'lx,lp'teon:→ *n:*

36:type: xntoie,texpreston:

38:type: vmbole. l"exoressinn:

37: tyr Pe:caractere;expresston;a

39:type: *le,*expresston:)

48:t type:Synbole,l expression:

41:type:Symbole,t'expresston:(

42:type:Entter,l'expresston:o

peesh,p

Figure 2:résultats des test de Upile.txt (partiel)

**Pour Dpile.txt:**

**hetianooghetianmo-VirtuoiBox:~/plt/projets ./test\_AL**

Ficher Non :

Dpile.txt

i n'exist pas des errors!

1:type:Mot\_clef,l'expression:Automate

2:type:Synbole,l'expression:(

3yyep:、lon:)

type: symbolelex pre

8:ty:leession:=

type: synbole,l'expres

10:type:Caractere,l'expresston:0'

:`1'

13:type:Synbole,l'expression:,

14: type: xpr \*2`

16:type:Mot\_clef,l'expression:initial

15:type:Symbole,l'expression:1.

17:type:Synbole,l'expresston:-

18:type:Entler,l'expresston:0

19:type:Mot\_clef,l'expression:final

20: type: Symboe*,*xpression:;

21: type: Symbole, ss1on:

22:type:Entier, l'expresston:2

**2**:ypmobtol\_ex'epn:transltion

25:type:Synbole,l'expression:=

type: Synbole,l'expresston:[

27:type:Synbole,l'expresslon:(

28:type:Entier,l'expression:e

29:type:Flech,l'expression:→

30:type:Entler,l'expression:o

31:type:Symbole,l'expression:,

32:type:Caractere,l'expresston:'a

33:type:Synbole,l'expression:

34:type:Synbole,l'expresston:(

35:t type: Flech, 'ex1pression:→

36: type:Synboie,l'expresston:,

37:type:Caractere,l'expresston:a'

type: Symbole1, express

39:type: synbole,L'expression:)

40:type:Synbole,.expresston:

41:type:Synbole,l'expression:(

42:type:Entler,l'expression:o

**Méthode de réalisation**

Dans cette partie,jedois vérifier que lerésultatobtenu après analyse lexicale est

syntaxiquement correct.Par exemple,jedois juger sila position des crochets et des signes égal

est correcte.En même temps jedois aussivérifier que les opérateurs ont le bonnombres d'arguments.Dans ce processus,j'enregistre les informationsclés del'automate:

·le nombre de pile

·le nombre d'états,inclus initial et final

· les états

·Iinformation de chaque transition (par exemple linfo pour empiler et dépiler)

Mon idéegénérale est de parcourirles résultats dela premiere partie dulexique,et de

déterminer quelle opération doit être effectuée pour chaque lexème qui est bouclé.De cette façon,la collected'informations sur tous les mots-clés peut êtreréalisée.

Tout d'abord,j'ai créé deux structures pour representer toutes les informations dont nous

avons besoin pour créer un automate.La deuxième structure represente les informations privées de chaque transition:

typedef struct trans\_info

{

int source; //la source de lexeme dans une trasition

int destination; //la destination de lexeme dans une transition

char emission; //1'emission du lexeme

int em ou de[10]; //empiler:1 depiler:-1

**char** pile[10]; //la pile contient des lexemes qui empilent

}trans\_info;

typedef struct syntax\_info

{

char etats[20][20];

int etat\_nombre; //1'automate avoir plusieurs etats

int initial; //il existe seulement un etat initial

int final[15]; //1'automate peut avoir plusieurs etats final

int final\_nombre;

int trans\_info\_nombre; //le nomber de transitions

int pile\_nombre; //nombre de pile

trans\_info\*trans[50];

**}syntax\_info;**

Ensuite,j'ai crée une pileafindepouvoir jugersi les parentheses sont syntaxiquement correctes.Dans lemême temps,jedois égalementsimuler le processus d'empilement et

d'éclatement du lexème au cours de chaque transition.Ici,jutilise directement les fonctions de création de pile et deempiler et dépilerde la piledontle professeur a parlé en classe

typedef struct pile

{

char (\*pop)( void );

void (\*push)( char);

int (\*vide)( void );

}pile;

pile Mapile;

**char pop\_aux(void)**

{

if **(hauteur** == 0)

{ printf( "pile vide!\n");

**exit(1);**

}

**else**

**return (data[--hauteur]);**

};

**void push\_aux (char c)**

{

**if (hauteur == 50)**

**printf("pile pleine !\n");**

**exit(2);**

}

**else**

**data[hauteur++]=c;**

};

**int vide\_aux(void)**

{

**return (hauteur ==0)**;

};

{

Ensuite,je définis lexeme\_address ce parametre et le mets a jour constamment pour obtenir une lecture continue du lexème en res.Chaque fois que je lis un mot-clé,j'analyse les

informations de chaque mot-cle.Ici,j'ai construit 5 fonctions pour analyser les cinq mots-clés séparément.Par exemple la fonction ci-dessous:

//analyser mot clef "initial"

void analyse\_motclef\_initial(lexeme \* res)

//examine '='

1exeme\_address +=1;

is\_egale(1,res);

//noter l'info d'initial

ex\_\_asre-in =is\_entier(res);

1exeme\_address +=1;

return;

}

Une fois qu'il y a une erreur de syntaxe,j'utilise une fonction is error pour signaler l'erreur. De plus,jai également construit des fonctions pour juger des symboles,et j'ai appelé ces

fonctions dans la fonction d'analyse.

Par exemple

//Si c'est un caractere

char is\_caractere (lexeme\* res)

{

if(res[lexeme\_address].type !=Caractere)

is\_error(res);

return res[lexeme\_address].expression[1];

}

Le jugement de transition est le plus compliqué et le plus critique.Parce que chaque transition peutavoir plusieurs informationsempiler et dépiler,ou elle peut ne pas utiliserla pile.Jai donc construit une fonction search\_trans\_info pour compléter la collectedechaque information de transition.

Enfin,aprèsavoir collecté et stocké les informations de chaque mot-clé,je sors les informations et j'obtienslerésultat.

**Problèmes rencontrés**

Dan ce processus j'airencontré principalement trois problèmes:

1.Tout d'abord,comment dois-je stocker toutes les informations sur les mots clés?Au début, j'ai essayé d'utiliser une liste chainée ou un tas pour stocker des informations,mais j'ai

rencontré beaucoup d'erreurslors dela compilation et la mise en œuvre était lourde.Enfin, aprèsavoir discuté avec mes camarades de classe,jaidécidé de construire deux structures pour représenter clairementle stockage de chaque information.

2.Initialement,j'aicomplété laréférence du parametre lexeme\_address en passant le

paramètre dans la fonction,mais dans le processus,la valeurdu parametre n'est pas mise à jour,ce qui peut êtredü au passage de lavaleur.Pour résoudre ce probleme,jai choiside modifier la variable globale directement dans chaque fonction pour mettre à jour lavaleur du paramètre

3.A lorigine,jen'ai pas passé le parametre res en tant que parametre formeldanschaque

fonction,mais une erreur de **segmentation fault** s'est produitelors dela compilation finale. Cest ce qui me dérange le plus.Enfin,apresavoir essayé plusieurs fois,Jaidécouvert que je devais passer des parametres dans chaque fonction.Je pense que la raison possible est que s'iln'est pas transmis,Iaccès au tableau sera hors limites.

**Résultat du test**

**Exécution de Iinstruction:**



**hetiannochetianno-VirtualBox:~/plt/projets gcc test\_ASy.c -o test\_ASy hetiannochetlanno-VirtualBox:~/plt/projets ./test\_Asy**

Figure 4:Exécution del'instruction

**Pour Zpile.txt:**

pile…\_=aResultat … …

final etats=o 12

Pile info:-1# F

etat\_nombre=4

etats="1" "2""3""Init"

inittal =3

final\_nonbre =3

Transition 1:

destination=a

enlssion

**Transition 2**:

source =3

destination =1

**emtssien a1**

desttnation =2

Transitien 4

destination =1

Transitien s

source =0

destination =2

emission =2

Transttlon 6:

source =1

desttnation =0

emission =o

Transitton 7:

ian=2

Figure 5:résultats des test de Zpile.txt (partiel)

**Pour Upile.txt:**

…Syntax Resultat--

pile\_nombre =

etats-"一

"三"

final\_nombre=1

int

final etats =1

Transttian t:

source =0

destination =θ

entsston =a

Pile info:1:a

Transition 2:

Source a

 =1

P1le Lnfe:-1:

Transitlon 3:

destinatn =1

entsston =b

**Transitien 4;**

an s

enission =c

**Transition 5:**

**destinatlon =2**

enission =c

**Transttion 6;**

atn =1

enission Pile

=b

Figure 6:résultats des test de Upile.txt (partiel)

info:-1:a

**Pour Dpile.txt:**

;…syntax Resultat … …

pile nombre

Pile tnfo: -1:b



final nombre

tat\_nombre

etats

nitlal

inal etats -

**Transition 1:**

einati=oθn=@

enisston =a

**Ptle info:1:**

source = .0

Plle info:-1:a Plle info:1:b

**destinati ion=1**

entsston = b

**ransition 3:**

source =

**destinatlon =1**

niilsioi -1:a Ptle tnfo:1:b

t**ransttion4**

atn =2

source =1

Pile info: -1:

**Transition 5:**

source

estination 2

mission =

Figure 6:résultats des test de Dpile.txt (partiel)

**Partie 4:Analyse Sémantique**

**Méthode de réalisation**

Pour l'analysesémantique,jefais quelquesvérifications sur lerésultat del'analyse syntaxique:

·vérification que l'automate est bien déterministe;

·vérification que les transitions ont des syntaxes corrects avec lenombrede piles; ·vérification que les nombres des états dans les transitions sont corrects;

·vérification que les nombres des piles sont corrects

ll faut noter ici que jen'ai pas vérifié sile type de chaque lexème est correct, **car le type de**

**lexème a été vérifié à l'avance lors de l'analyse syntaxique** !Nous n'avons donc pas besoin de porter devérification sur le type lors del'analysesémantique.

Par exemple,pour vérifier les nombres des piles sont corrects,j'aiconstruit une fonction examine\_pile\_nombre():

//examiner le nombre des

void examine\_pile\_nombre()

{

for(intj=0;j<syntax\_res->trans\_info\_nombre;j++)

{

int count =-1;

for (int k=0;k<10;k++)

{

//noter les actions sur la

if(syntax\_res->trans[j]->em\_ou\_de[k]

{ //printf("\n%d",syntax\_res->trans[j]->em\_ou\_de[k]);

count++;

}

/printf("\n%d",count);

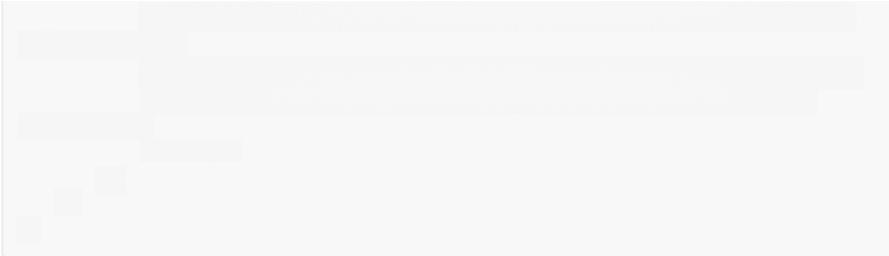
if(count>syntax\_res->pile\_nombre)

{

pile

piles

!=0)

printf("\nSémantique erreur:le nombre de la pile de transition : **%d\n”** , **count);**

printf( "lenombrede pile totals:%d\n",syntax\_res->pile\_nombre);

printf( "lenombredela pile de transition> lenombrede pile totals.\n" );**exit(1);**

}

}

}

D'autres fonctions devérification peuvent se référer au code complet.Parce que jaifait assez de travail dans lanalyse syntaxique,je n'ai pas de problemes insolubles dans Tanalyse

sémantique.

**Résultat du test**

Ici,nous utilisons Upile.txt comme exemple pour tester.

**Exécution de l'instruction:**



**hetlannochettanmo-VirtualBox:~/plt/projets hetlanmoghetiango-VirtualBox:~/plt/projets**

**gcc test\_ASe.c -o test\_ASe**

**./test\_ASe**

Le résultat montre que lefichier n'a pas d'erreurs sémantiques:



**D'apres analyse syntaxique,il n'existe pas des erreurs** senantiques

**hetiannochetianno-virtualBox:~/plt/projetS**

Ensuite,nous modifions Upile.txt pour vérifier que notre fonction peut être réalisée.

·vérification que lautomate est bien déterministe

J'ai changé le texte:

transitions=[(0→0,`a`,(→ ,`a^)),

(0→ 1,`,(`a`,→)),

Le résultat:



n**eitangoene tanpo vituatsox:/PL/Projes gcc tesi A**se.c o test\_Ase h**etianmoghettanmo-VirtualBox:~/plt/projets./test\_ASe**

**Ficher Non :**

**Upile.txt**

**Semantique erreur :Non deterministe,voir les deux transitions:**

**transition 1:o->0,par a**

**transition 2:o->1,par a**

·vérification que les transitions ont des syntaxes corrects avec le nombre de piles

J'ai changé le texte:

/\*Cet automate a une pile,il permet de programmer toutes les grammaires BNF et

de simuler la reconnaissance des langages algébriques\*/

Automate O={

/\*Cet automate a une pile

Le résultat:



**hetlanwoghetianno-VirtualBox:~/plt/projets**

**hetiannochetianno-virtualBox:~/plt/projets**

**Ficher Non:**

**uptle.txt**

**Senantique erreur :le value d'etat 4 est plus grand que le nombre d'etats 3**

**gcc test\_ASe.c -o test\_ASe**

**./test\_ASe**

·vérification que les nombres des états dans les transitions sont corrects

J'ai changé le texte:

etats =["-","二","三"]// Les noms peuvent etre differents de numéros

//Chaque état est repéré par son numéro dans la liste etats

//final 2,e s'iln'y a qu'un état final

Le résultat:

**ettenodnettanno vurtuatkox:~/PLL/Projes gcc test Ase.c -o test ASe**

**hettannoghettanno-virtualBox:-/plt/projets ./test\_ASe**

**Ficher Nom:**

**Upile.txt**

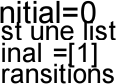
**Semantique erreur:le nombre d'etat final 4 est plus grand que le nombre d'etats 3**

·vérification que les nombres des piles sont corrects:

J'ai changé le texte:

a^n.c^p.b^n,où n>0 \*/

etats =["-","二","三"]// Les nons peuvent etre differents de numéros //Chaque état est repéré par son numéro dans la liste etats

//final `i(`'un état final

Le résultat:



**hetlanmoghettanmo-Vtrtualeox:-/plt/projets gcc test\_ASe.c -o test\_ASe**

**Upile.txt**

**Semantique erreur :le nombre de la pile de transition:1**

**le nombre de pile totals :0**

**le nonbre de la pile de transition>le nonbre de pile totals.**

**hettannochetianno-virtualBox:~/plt/projets ./test\_ASe Ficher Nom :**

Les résultats des tests montrent que nous pouvons détecter toutes nos erreurs sémantiques spécifiées.Nous avons donc terminé le travail d'analyse sémantique.

**Partie 5:Compilation**

**Méthode de réalisation**

Dans la partie compilation,jesouhaite envisagerde stocker les résultats obtenus dans les trois premières parties dans deux fichiers,lefichier TS.txt et le fichier VM.Les regles de stockage des

informations de chaque état dans la VM ont été expliquées en détail dans le rapport,et I'enseignant a également donné desexemples clairs en classe.

Je dois donc décider comment stocker l'adresse et les autres informations correspondant à chaque état du tableau vm.J'ai d'abord déterminé la longueur requise du tableau vm et le

nombre de transitions de chaque état:

intvm[100];

**int** etat\_vm\_len[20];

**int** etat\_trans\_nom[20];

**int** etat\_adresse[20];

**int** vm\_length =0;

void vm\_trans(int cur\_trans)

{

int source =syntax\_res->trans[cur\_trans]->source;

//Mettre à jour la longueur de chaque état(2 est longueur de source et destination)

etat\_vm\_len[source]+=2 \*syntax\_res->pile\_nombre +2;

//Mettre à jour le nombrede transitions pour chaque état

etat\_trans\_nom[source]+=1;

}

Ensuite,je le mappe a chaque état en itérant sur la source de chaque transition.Apres que chaque transition correspond à état,jeremplissele tableau vm avec les informations de

transition de chaque état à tour derôle.

if (etat == temp->source)

vm[index]=temp->emission;

index += 1;

vm[index]=etat\_adresse[temp->destination];

index += 1;

for (int i=0;i<syntax\_res->pile\_nombre;i++)

{

vm[index]=temp->pile[i];

index += 1;

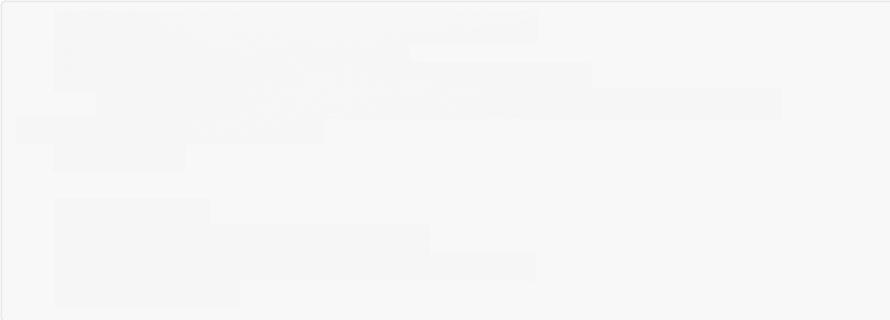
vm[index]=temp->emLou\_de[i];

index += 1:

}

}

Enfin,j'imprime les résultats et les stockedans deux fichiers,TS.txt et VM,respectivement.

//stocker la table des symboles dans TS.txt

FILE\* ts = fopen( "TS.txt" , "w");

for fpri (ts, "l+d\n",syntax\_res-

>etats[i],etat\_adresse[i]);

fclose(ts);

//Stocker vm

FILE\* vm\_file = fopen( "VM", "wb");

fwrite(vm, sizeof (int),vm\_length,vm\_file);

fclose(vm\_file);

**Problèmes rencontrés**

declencher c,l'adresse de desttnatton =22

Le principal problème que j'airencontré dans cette partie était que la transition n'était pas

correctement mappée sur chaque état au début,mais était directement stockéedemanière

séquentielle.Cela entrane une incompatibilité entre le contenu et I'adresse stockés par la

machine virtuelle.J'ai résolule probleme apres avoir demandé conseil à mes camarades de classe et obtenu lebonrésultat.

**Résultat du test**

**Exécution de I'instruction:**



**hetianwoghettanno-VirtualBox:~/plt/projets hetiannoghetianno-VirtualBox:~/plt/projets**

**gcc test\_compile.c -o test\_compile**

**./test\_compile**

**Pour Zpile.txt:**

Tableau VM:

213611162491150162486501624864911348649115016

aitip 1

etat e;

le non d'etat="1",l'adresse de source=6

er d,Itr 2 destination =11

declencher 2,l'adresse de destination =16

leeta d'etat="2",l'adresse de source =11

le nombre de transttlon -2,

declencher o,l'adresse de destination=6

declencher 2,l'adresse de destlhatton =16

etat 2:

le non d etat="3",l'adresse de source =16

le nombre de transition-2

leetatno: d'etat ="Init",l'adresse de source =21

e nombredde transition =3

declencher 0,l'adresse de destination =6 declencher 1,l'adresse de destination =11

declencher 0,l'adresse de destination =6

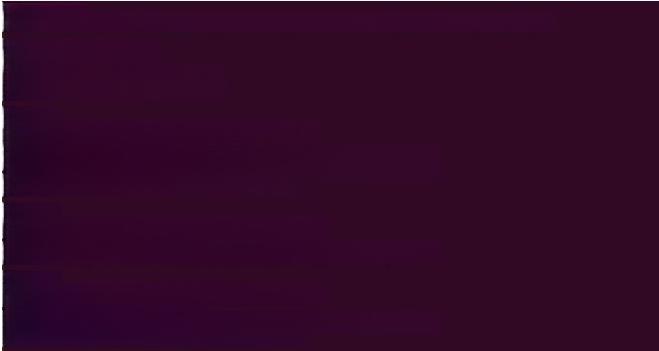
declencher 1,l'adresse de destination =11

declencher 2,l'adresse de destination =16

le nombre des etats finals =3

l'adresse des etats finals =61116

**Pour Upile.txt:**

**Tableau VM:**

141173974971981797 -19922 θ θ1981797 -12981797 -1992200

 ititll 

le nonbre des etats finals =1

l'adresse des etats finals =17

le non d'etat-"—",l'adresse de source=4

etat o:

ec d,aon d3 destination =4,Plle 1:enpiler a

declencher b,l'adresse de destination=17 Pile 1:dépiler a

le non d'etat-"=,l'adresse de source=17

etat 1:

ecl adtio1 destlnatton =17 Ptle

eletat n d'etat-"",l'adresse de source =22

ec l t destinatlon =17 Ptle

declencher c,l'adresse de destination =22

**Pour Dpile.txt:**

1:deptler a

1:deptler a

**Tableau VM:**

24130297497100981797 -19812981797 -198199300098 -1199300098 -1

 finals =1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| état o:  le nombre de transition =2  état 1:  dleecnleonncbr d,l'n  declencher c,l'adresse de | de source =4  destination =4 Pile destination =17 Pile  de source =17  destination=17 Plle  destination=30 Plle | 1:enpiler a  1:depiler a  1:depiler a  2:depiler b | Ptle  Pile | 2:enpller  2:enptler | b  b |

etat 2:

le non d'etat =2',l'adresse de source =30

ecr d,l'n d1 destination =30 Plle 2:depiler b

l'adresse des etats finals =30

le non d'etat =0,l'adresse

le non d'etat =1,l'adresse

declencher a,l"adresse de

declencher b,l'adresse de

Après vérification,j'ai trouvé que tous les résultats des tests étaient corrects.Jusqu'à présent, j'ai terminé le travail de compilation d'automate.

**Partie 6:Exécution**

**Méthode de réalisation**

La dernière partie de l'exécution consiste à entrer une chaine de chanes dans le compilateur que nous avons écrit.Ce processus nécessite I'utilisation de deux fichiers que jai générés dans la quatrième partie,à savoir le fichier TS.txt et le fichier VM.

1.Tout d'abord,je dois juger la commande d'entrée.J'ai appris de projet qu'il existe deux

modes d'exécution,./runtime VM et **./runtime debug VM**,ou. **/runtime debug VM** montre

I'ensemble du processus de debug.J'ai défini deux modes dans main fonction pour l'exécution.

if (argc == 2)

{

//non debug mode

debug =0;

if(strcmp(argv[1], "vm" ) != 0){

printf( "Erreur dans le parametre d'entrée!\n");

**exit(2);**

}

}

else if (argc == 3)

{

//debug mode

debug =1;

if(strcmp(argv[1],"-debug")!=0 ll strcmp(argv[2],"VM")!=0){

printf( "Erreur dans le parametre d'entrée!\n");

exit(2);

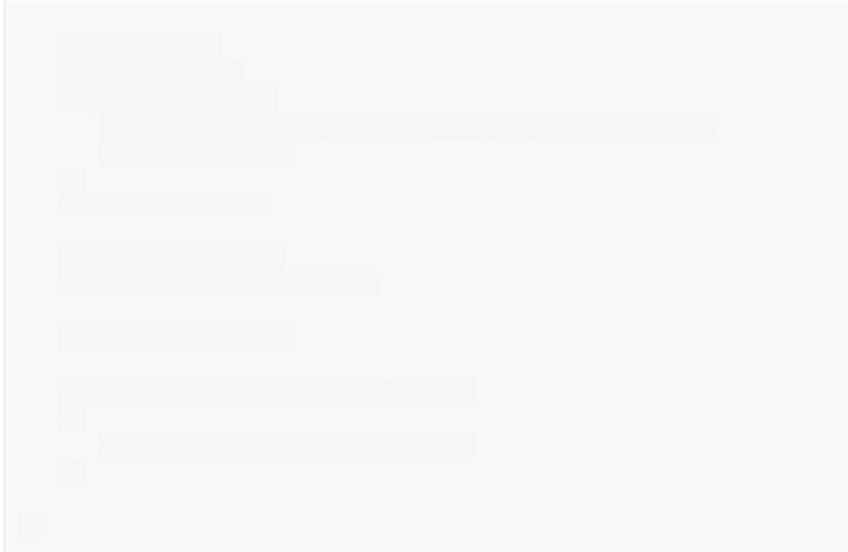
}

}

2.Ensuite,nous utilisons la fonction fscanf pour extraire les informations sur I'automatisation qui ont été obtenues à partir des fichiers VM et TS.txt.

void analyse\_vm()

FILE\* vm\_fichier = fopen("VM" , "r" );

vm\_length =0;

size\_t len =10;

while (len == 10){

len =fread(vm+vm\_length,sizeof(int),10,vm\_fichier); vm\_length += len;

fclose(vm\_fichier);

}

initial\_etat\_adresse =vm[1];

pile\_nombre =vm[0];

final\_nombre =vm[2];

for(int i=0;i<final\_nombre;i++)

{

final\_etat\_adresse[i]=vm[i+3];

}

}

3.Je stocke ces informations dans une nouvelle structure définie.Dans le meme temps,nous perfectionnerons la pile créée lors de Ianalyse syntaxique,pour indiquer clairement la

position de chaque élément dans la pile.

|  |  |
| --- | --- |
| //Définir la structure état  fichiers vM et TS.txt  {  **int** etat\_adresse[10];  char **etat\_nom[**20][20];  //Definir la nouvelle pile adresses des elements des  typedef struct pile  { | pour stocker les informations extraites des  realiser les operations sur different |

**void** ( \*push)( int\* , char\* , char );

**char** ( \*pop)( int\* , char\* );

**int** (\*vide)( int );

int **hauteur;**

char **data**[100];

**pile;**

typedef struct etat\_info

**}etat\_info;**

pour piles

}

4.Ensuite,nous commençons à faire le cycle de traverser chaque caractèrede la chane

d'entrée.Pour chaque caractère,nous déterminons s'il y a une transition appropriée pour la correspondre et que la commandedanslapile est correcte au cours de ce processus.

Lorsqu'un caractere est jugé,nous mettons a jour ladresse de la machine virtuelle pour

effectuer le caractère suivant.Une fois que tous les caracteres sont completement traversés, nous vérifions que la pile est vide.Sil est vide,cette chaine est refusé,si la pile est vide,

indiquant que cette chaine est acceptée.Un processus de mise en euvre détaillé peut faire référence au code et aux commentaires:

hetlanmoghettanmo-VirtualBox:~/plt/projets ./runtime -debug VM

//si i **( \_**le**vi**etralement,le mot est accepté!

printf( "Le mot %s est accepté!\n",s);

else {

}

printf( "Le mot %s est refusé!",s);

printf("Piles non vides \n");

**Problèmes rencontrés**

1.Lorsque vous rencontrez la premiere question est lorsque je lisle fichier DPILE.TXT.Les guillemets simples dans lefichierdonné ne peuvent pas correspondre,nous changeons doncles guillemets simples en guillemets doubles pour être cohérents avec les autres

fichiers:

4/\*Cet automate a deux piles

**5//Il correspond a l'automate reconnaissant le langage**

5//a^n.b^n.c^n où n>0(cours 10,page 18)\*/

7 etats =["A","B","C"]

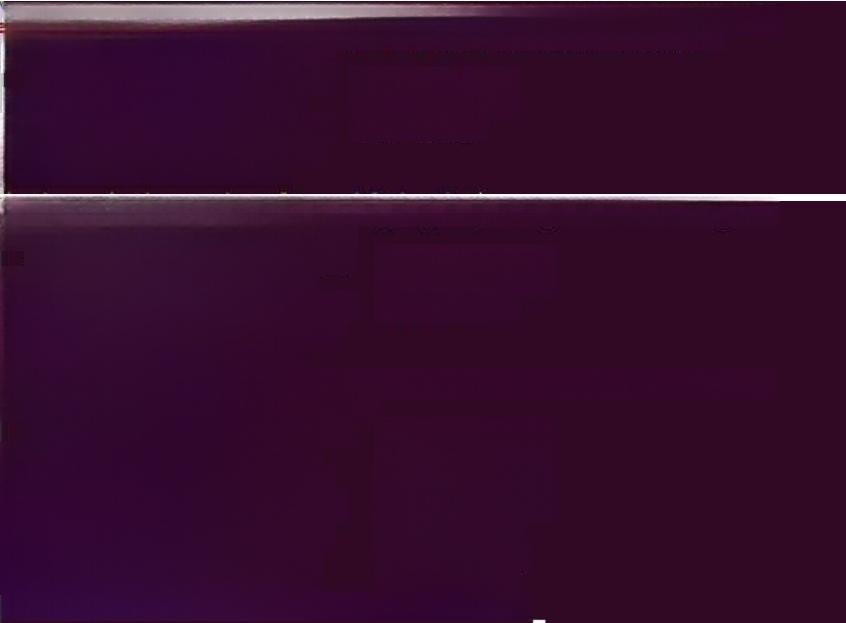
3 //On pourrait mettre aussi [`a',`b`,c`]

//Chaque iti péré par son numéro dans la liste etats

2.Initialement,j'ai utilisé la pile crééeprécédemment,mais je ne peux pas représenter Iemplacement des éléments dela pile,ce quiapporte beaucoup de problemes,alors je compte encore et construire une pile.

**Résultat du test**

Voici utiliser le Dpile.txt comme exemplede test:

**hetiannoghetianmo-virtualBox:~/plt/projets gcc runtine.c -o runtime hetiannoghetianmo-virtualEox:** ~/plt/projets ./runtime -debug VM

Doat le :Amot Pintri Pile 2:Vide

->>EEt 1de de

E ab e! !

**hetianmoghetlanmo-VirtualBox:~** /plt/projets./runtine -debug VM

oat-A moied'entre iill i

**b**-->E>reur :B:Pile Pil vide Pile 2:b

Le mot abbc est refusé !

on>Etaraa:ttAleat t:ntr il i

b>Etat :B Pile 1:aa Pile 2:b

b->État :B Pile 1:a Pile 2:bb

c>Etat:C Pile 1:a Pile 2:b

Le mot aaabbc est refusé !Piles non vides

Les résultats des tests prouvent que notre compilateur réussit!

**Partie 7:Conclusion**

Résumez notre travail,nous avons effectué une analyse lexicale,une analysedela syntaxe,une analysesémantique,une compilation du compilateur et une mise en œuvre finale,nous avons

reçu lesrésultats corrects,réalisez pleinement les objectifs du projet.