***UNIVERSITÉ* D'ABOMEY-CALAVI**

****

ÉCOLE POLYTECHNIQUE D'ABOMEY-CALAVI

****

GÉNIE INFORMATIQUE ET TÉLÉCOMMUNICATION(GIT4)

****

DÉFINITION DE LANGAGE TEXTUEL POUR CODER UN GRAFCET

II- ANALYSE SYNTAXIQUE AVEC BISON

Étudiant

HOUDJI Godwin

Sous la supervision de :

Dr Médésu SOGBOHOSSOU

ANNÉE ACADÉMIQUE: 2019-2020

Lors du premier exercice, nous avons défini des expressions régulières pour notre langages. Dans cette deuxième partie, nous utiliserons bison pour l'analyse syntaxique afin de déclencher des erreurs aux probables erreurs dans l'écriture du code.

Certaines expressions dans la première partie ont été améliorées. Les voici énumérer :

TRANSITION

Une transition est traduite par le fait que le système passe d'une étape donnée à une autre lorsqu'une certaine condition est vérifiée. Elle est comme suit :

go(nom_étape_début, nom_étape_suivante, condition) ; ***ou***
go(nom_étape_début, nom_étape_suivante1,nom_étape_suivant2 … nom_étape_suivanten,condition) ;

ANALYSE SYNTAXIQUE AVEC BISON

En utilisant bison, l'architecture de chaque programme doit être sous la forme :

- ***program : START ETAPE bloc END***

avec le non terminal bloc qui est sous la forme :

- ***bloc : bloc inst***
  | inst
- ***inst : expr***
- ***expr : TRANSITION***
  | ETAPE
  | CONDITION
  | IDENT
  | COMMENT
  | SLEEP

APPLICATION

Pour notre application, nous écrirons un code correspondant à ce diagramme de système représenté avec JgrafChart.

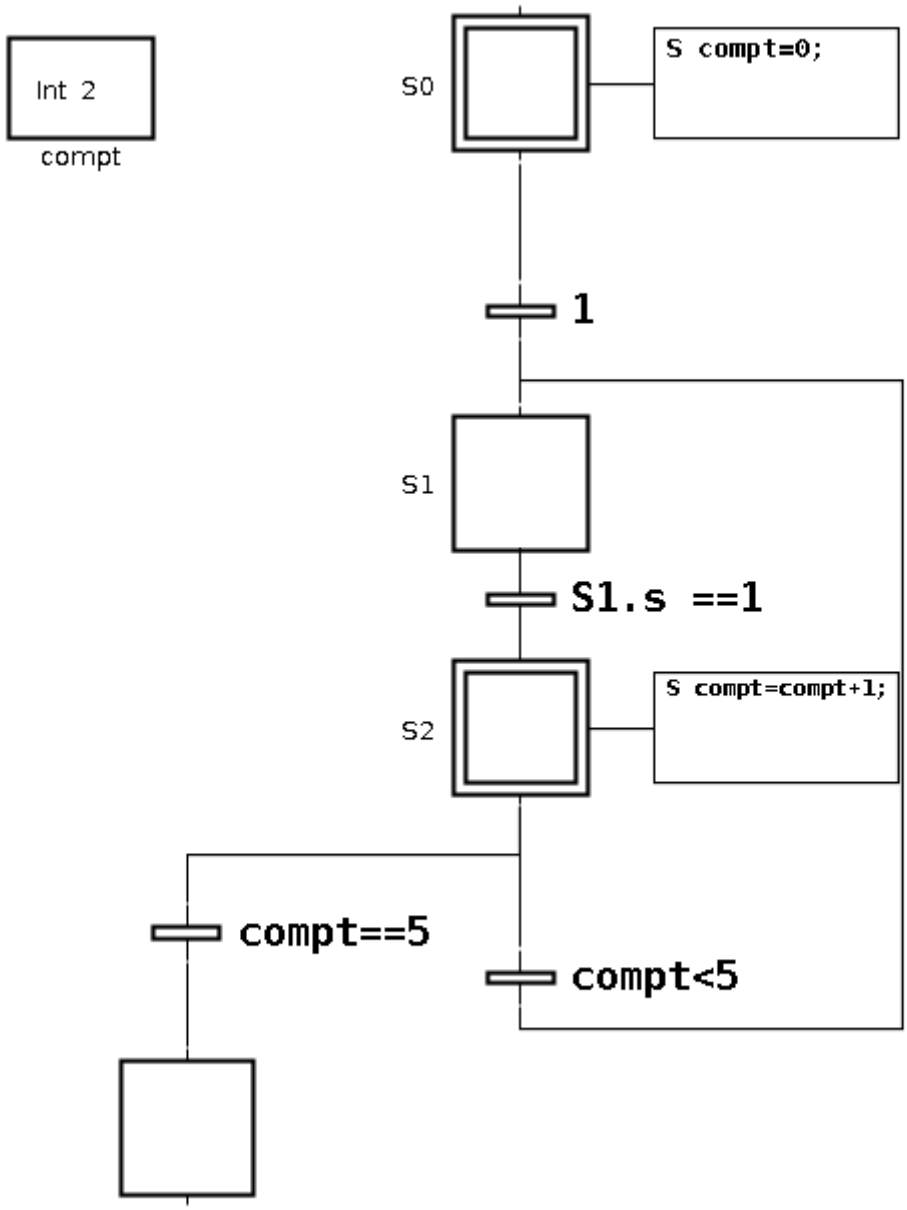


Figure1 : Un automate avec un compteur

NB A cet exemple, nous avons ajouté un autre état S4 pour illustrer l'amélioration sur les transitions. Nous aurons donc 5 étapes, 3 transitions et 3 conditions dans notre système.

Code :

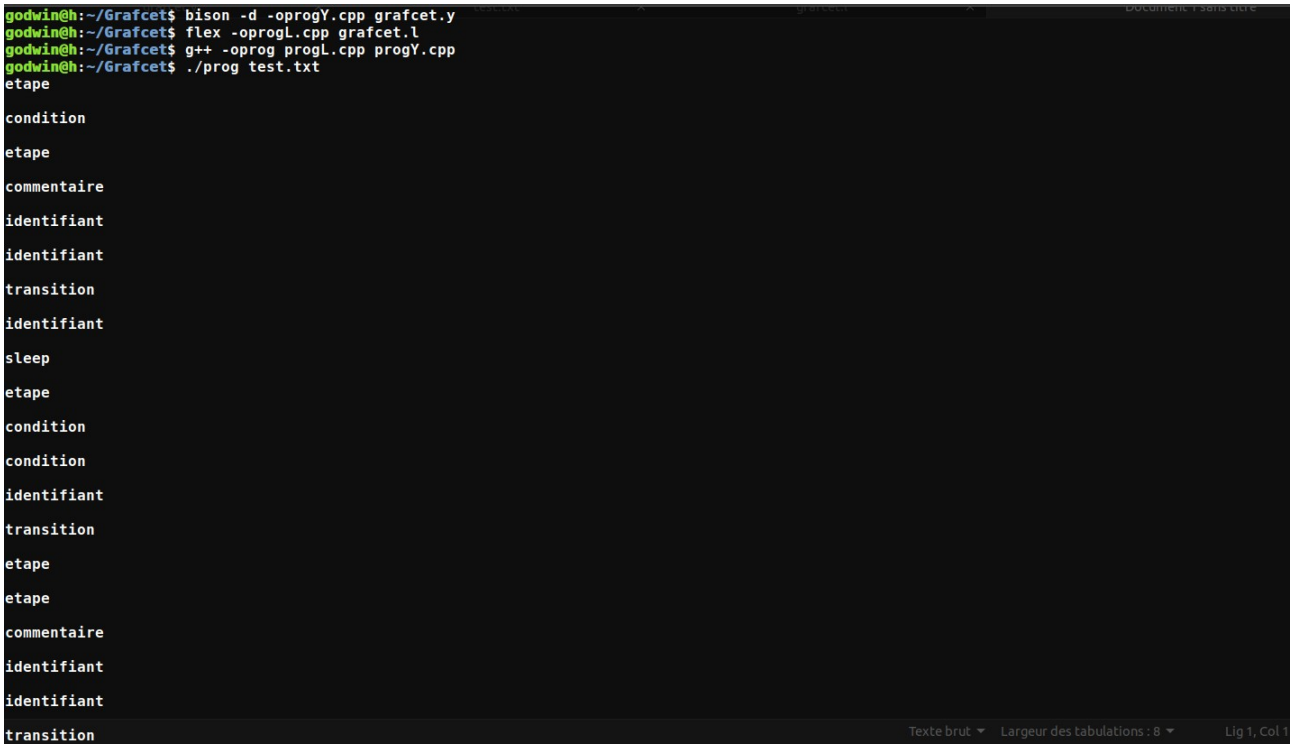
```
START
S S0 A compt=0;
C C1 1;
S S1;
# Aller de la premiere etape a la deuxieme
go(S0,S1,C1);
wait(1);
S S2 A compt=compt+1;
C C2 compt<5;
C C3 compt==5;
go(S2,S1,C2);
S S3;
S S4;
#Aller a deux etats a la condition C3
go(S2,S3,S4,C3);
END
```

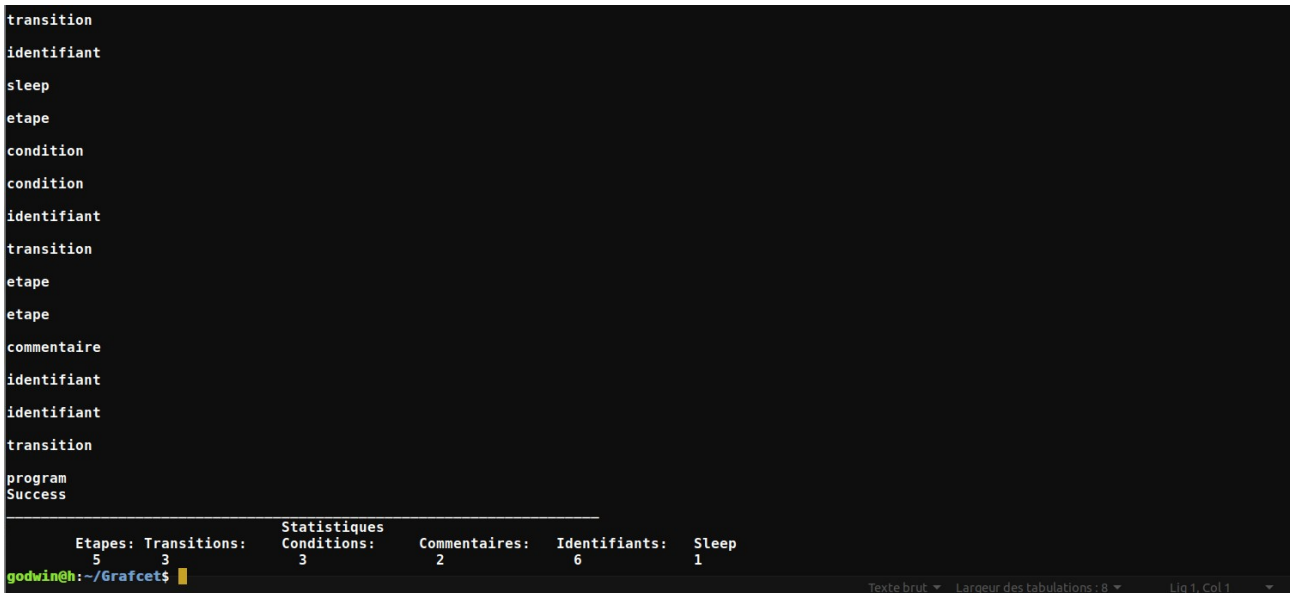
Pour exécuter le code, on tape dans le terminal :

- ***bison -d -oprogY.cpp grafcet.y***
- ***flex -oprogL.cpp grafcet.l***
- ***g++ -oprog progL.cpp progY.cpp***
- ***./prog test.txt***

Avec test.txt, le programme décrivant le système de notre application.

Après l’exécution du code dans notre interpréteur, nous obtenons à la sortie :





Prenons cet exemple :

START

S S0 A compt=0;

C C1 a=1; # erreur lexiale a==1 et non a=1

nous avons aussi omis le « END » de la fin.

Comme attendu, nous obtenons deux types d’erreurs : une erreur lexicale à la ligne 3 et une erreur syntaxique à la fin du programme.