REPUBLIQUE DU CAMEROUN
Paix – Travail - Patrie
******
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
******
UNIVERSITE DE YAOUNDE I
******
FACULTE DES SCIENCES

REPUBLIC OF CAMEROON
Peace- work–Fatherland
******
MINISTRY OF HIGHER EDUCATION
******
UNIVERSITY OF YAOUNDE I
******
FACULTY OF SCIENCE

: RAPPORT THEME :

TP Examen Langage Formel et Compilation

Rédigé par :

- AMBEME AMBASSA LEONEL BRICE 19M2514
- MOGHAM YOUONYONO ABDEL AZIZ 19M2218
- KOUABITCHOU NJIECHEU RAPHAËL ANGE 18T2389
- TSANGO EDOU BENJAMIN 19M2400

Encadré par : Dr KOUOKAM ETIENNE

Année académique 2022 - 2023

Table des matières

I. INTRODUCTION

Le langage formel est un langage qui utilise les symboles et les règles précises pour représenter des concepts mathématiques, logique ou informatique.
La compilation est le processus de traduction d'un programme écrit dans un langage de programmation en un code exécutable par une machine. Le compilateur analyse le code source vérifie, vérifie sa syntaxe et sa sémantique puis génère du code binaire et peut être exécuter sur une machine.

Les automates sont des outils fondamentaux en informatique théorique. Ils sont utilisés pour modéliser des systèmes qui ont un comportement répétitif Et qui peuvent être décrits par des états et des transitions. Les automates peuvent être utilisés pour résoudre des problèmes de reconnaissance motifs, de vérification de la syntaxe, etc.

En Python, il existe plusieurs bibliothèques qui permettent de créer et de manipuler des automates. Dans ce guide, nous allons nous concentrer sur deux bibliothèques : `**pydot**` et `**graphviz**`, qui permettent de créer des graphes d'automates et de les visualiser.

II. OBJECTIF GENERALE

Fournir un outil de manipulation d'automates et s'en servir pour la reconnaissance de tokens dans un texte source.

III. OBJECTIFS

- Permettre de communiquer avec les ordinateurs de manier plus efficace et précise.
- Permet au programmeur de communiquer avec les ordinateurs de manière plus clair et sans ambiguïté
- Améliore l'efficacité et la précision de la communication entre les programmeurs et les ordinateurs, ce qui facilite le développement des logicielles plus complexe

IV. TECHNOLOGIE

- Langage de programmation python.
- Système linux

V. INSTALLATION

Pour utiliser les bibliothèques pydot et `**graphviz**`, vous devez d'abord les installer sur votre ordinateur. Vous pouvez le faire en utilisant `**pip**`, le gestionnaire de paquets Python :

- **Bash**
- **pip install pydot**
- **pip install graphviz**

Assurez-vous également que Graphviz est installé sur votre système. Vous pouvez le télécharger depuis le site officiel : https://graphviz.org/download/

VI. VERS L'ANALYSE LEXICAL

A. Création d'un automate

Pour créer un automate en Python, vous pouvez utiliser un la **class Automaton** qui représente l'ensemble des états et des transitions.

```
python
etat_initial = 0
states = {0,1,2}
etats_finaux = {2}
alphabet = {'a','b'}
transition_function =     {
    0: {'a': 1},
    1: {'a': 1, 'b': 2},
    2: {}
}
automate = Automaton(alphabet,states,etat_initial,etats_finaux,transition_function)
```

Par exemple, voici un automate qui reconnaît le langage `a*b` :

Dans cet exemple, l'état initial est `0` et l'état final est `2`. Par exemple, la transition de l'état `0` à l'état `1` se fait avec la lettre `a`.

B. Visualisation de l'automate

Pour visualiser l'automate, vous pouvez utiliser la bibliothèque `**pydot**` et `**graphviz`.** Voici un exemple de code qui crée un graphe d'automate et l'affiche dans une fenêtre :

```
python

import pydot

from IPython.display import Image, display


draw2_automaton():pour les automates utilisant les tuple() comme cle pour la fonction de transition

draw_automaton():pour les automates utilisant les set() pour  definir les
```

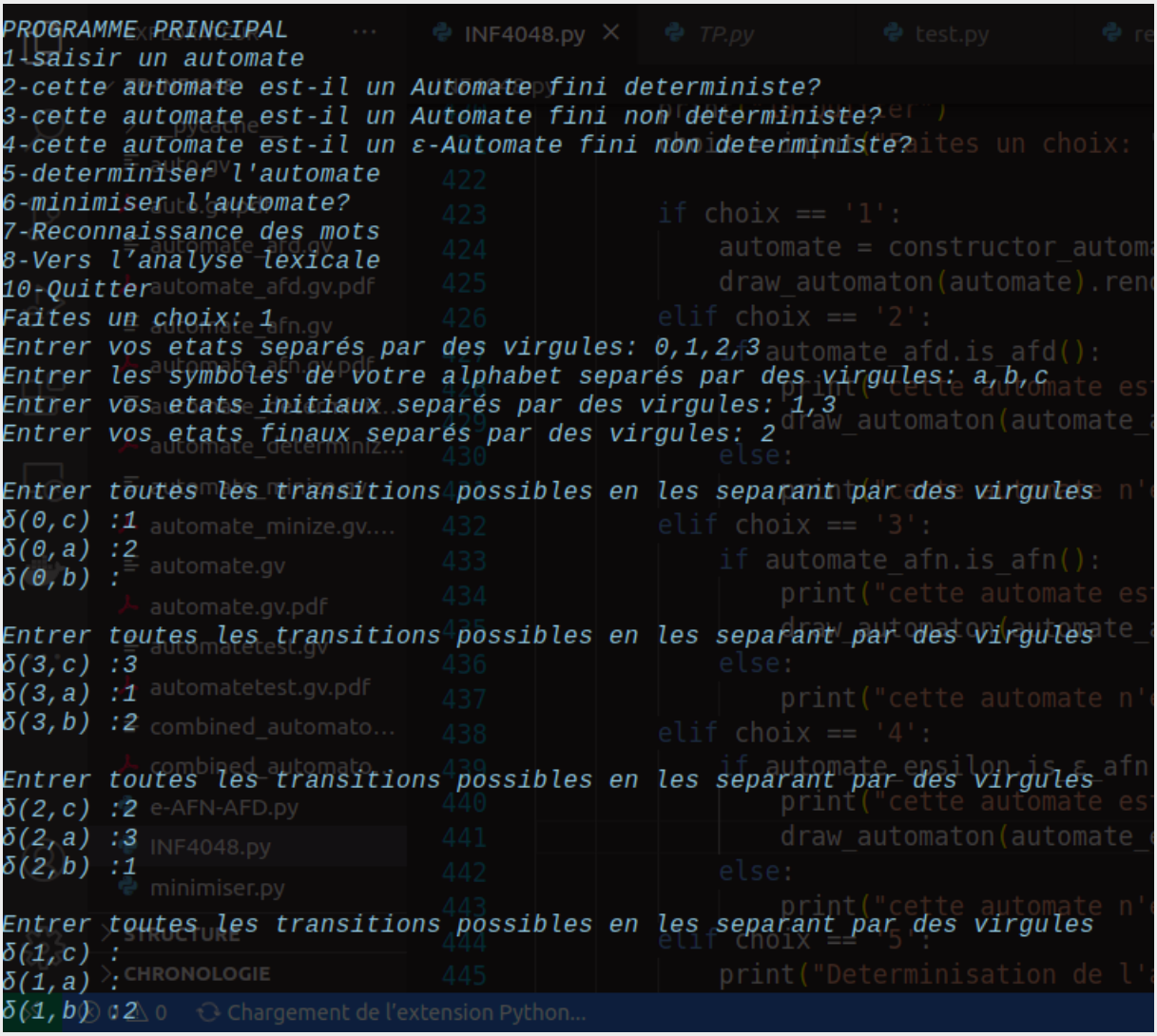
Ce code utilise la fonction `**afficher_automate**` pour créer un graphe d'automate à partir du dictionnaire `automate`, de l'état initial et des états finaux. La fonction crée un objet `**pydot.Dot`,** ajoute des nœuds pour chaque état et des arêtes pour chaque transition, puis affiche le graphe dans une fenêtre.

C. Vérification d'un mot

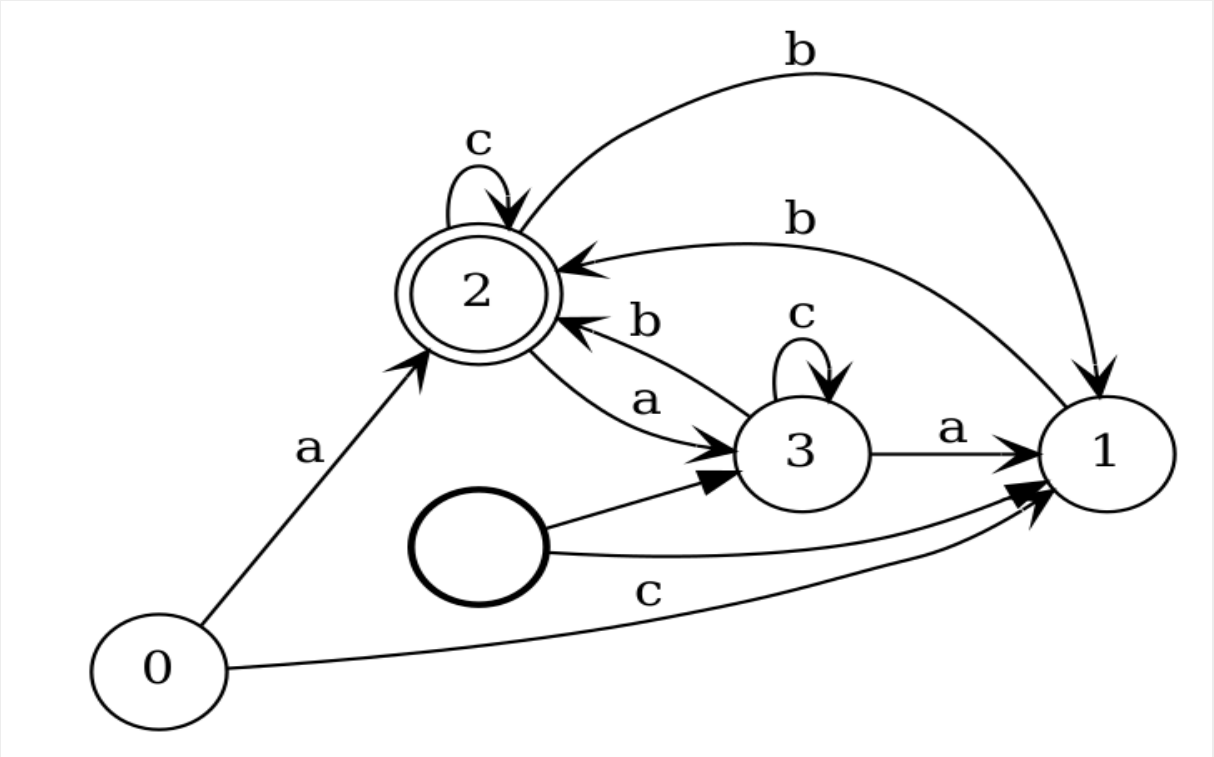
Pour vérifier si un mot est accepté par l'automate, vous pouvez utiliser une fonction qui parcourt le mot en suivant les transitions de l'automate. Voici un exemple de code qui vérifie si un mot est accepté par l'automate :

recognize_word(texte,automate)

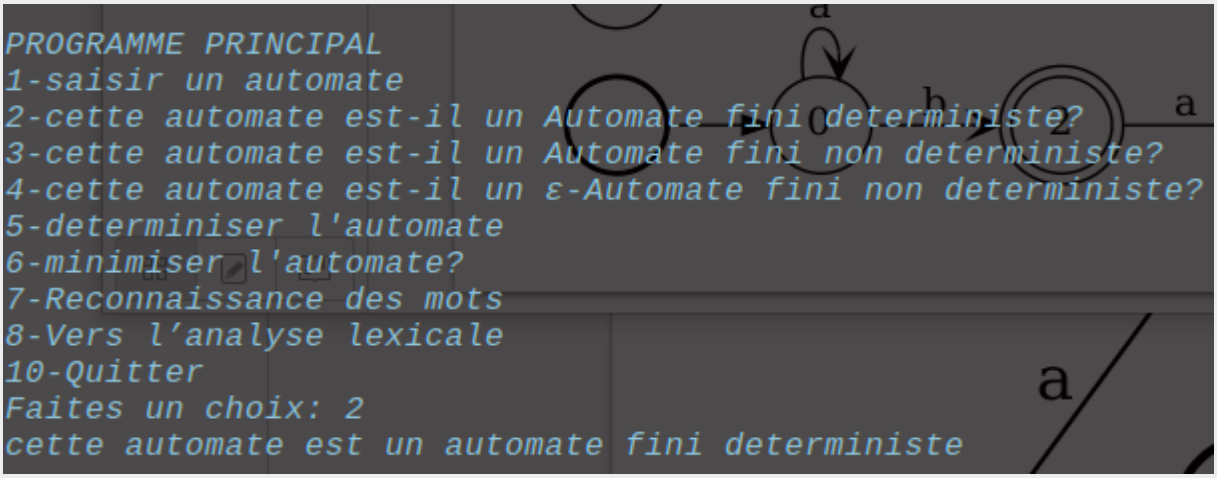
1. Saisi de la matrice



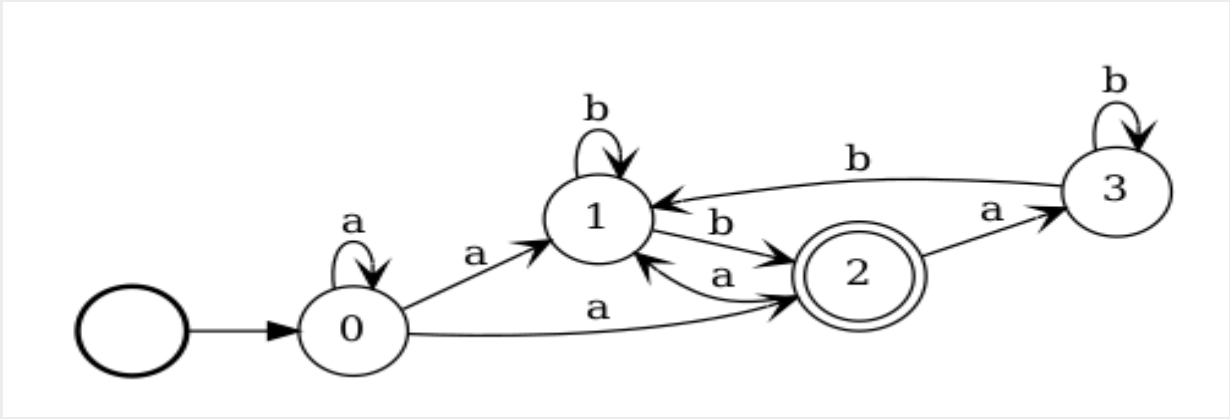
2. Automate



3. Question AFN OU AFD



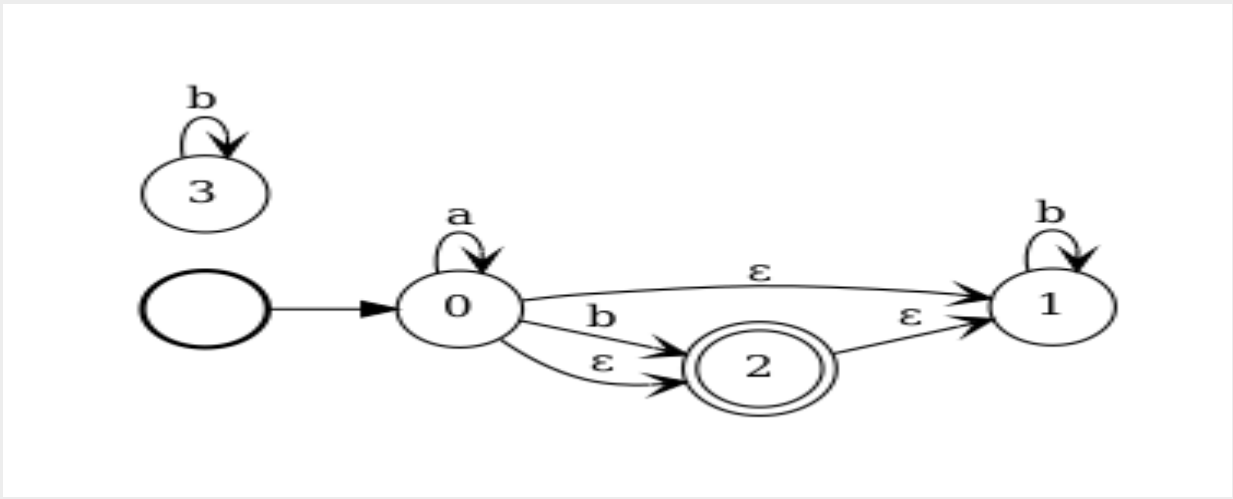
4. AFN



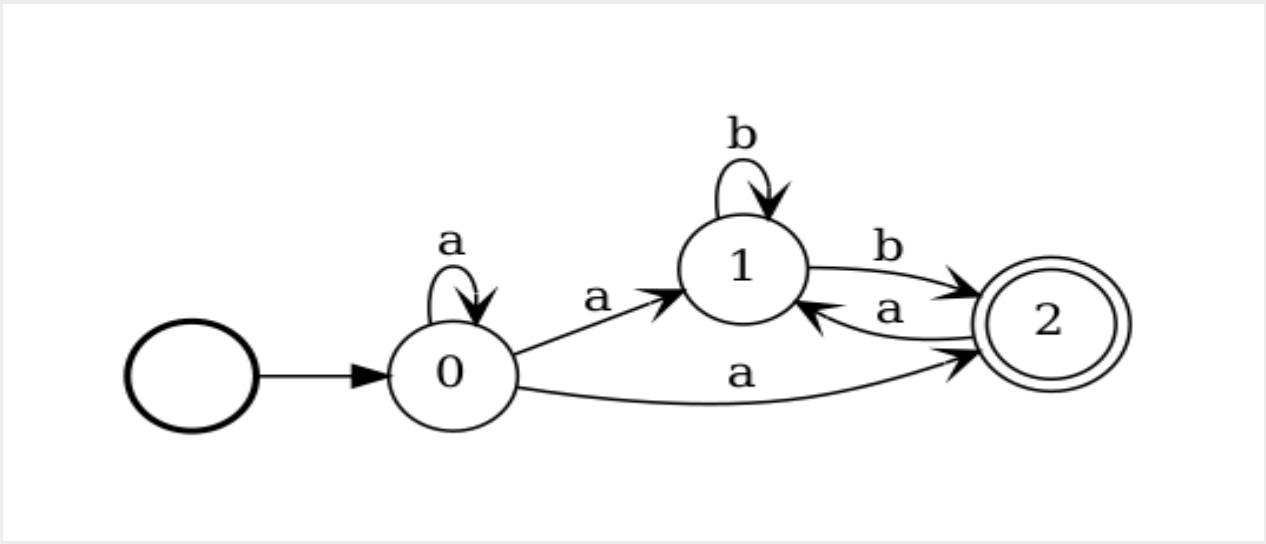
5. QUESTION D'EXCILON AUTOMATE ε-AFN

```
PROGRAMME PRINCIPAL
1-saisir un automate
2-cette automate est-il un Automate fini deterministe?
3-cette automate est-il un Automate fini non deterministe?
4-cette automate est-il un ε-Automate fini non deterministe?
5-deterministe l'automate
6-minimiser l'automate?
7-Reconnaissance des mots
8-Vers l'analyse lexicale
10-Quitter
Faites un choix: 4
cette automate est un ε-Automate fini non deterministe
```

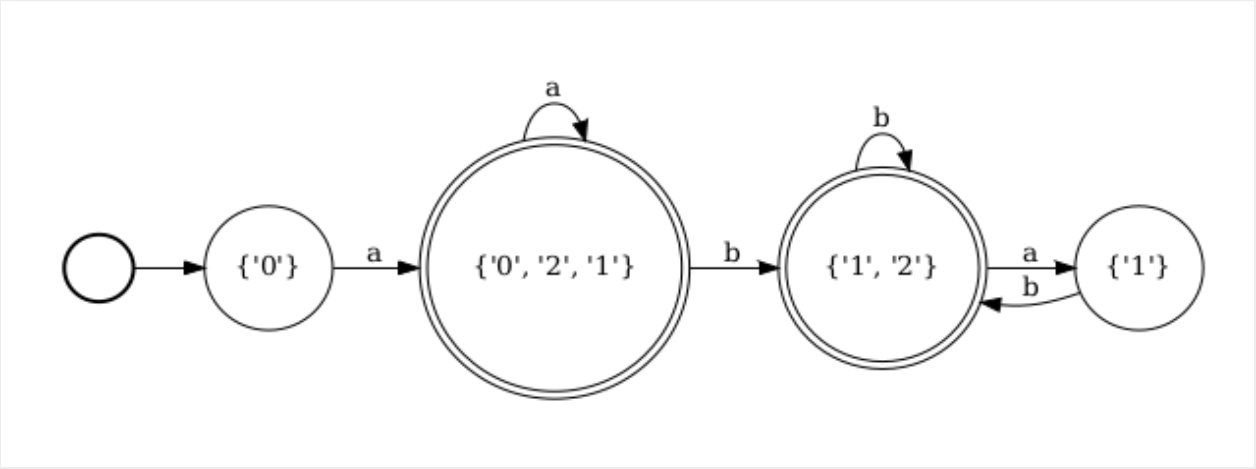
6. Automate



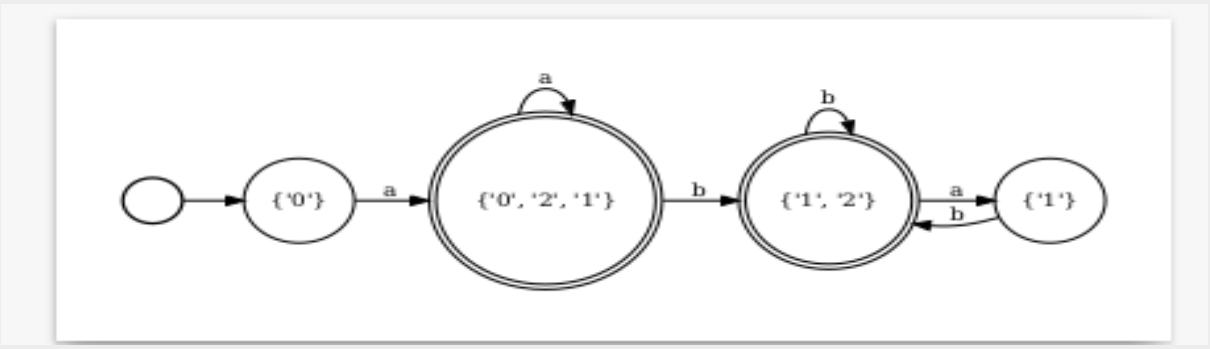
7. Automate AFN de base



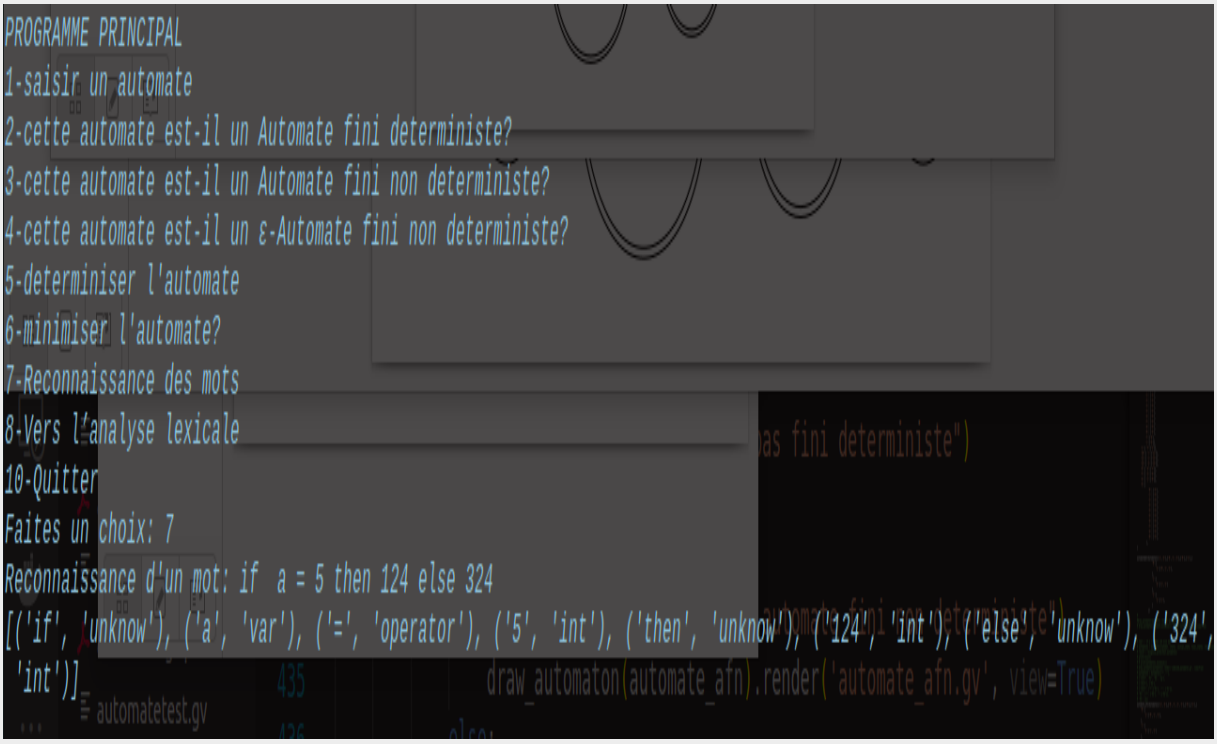
8. Automate déterministe



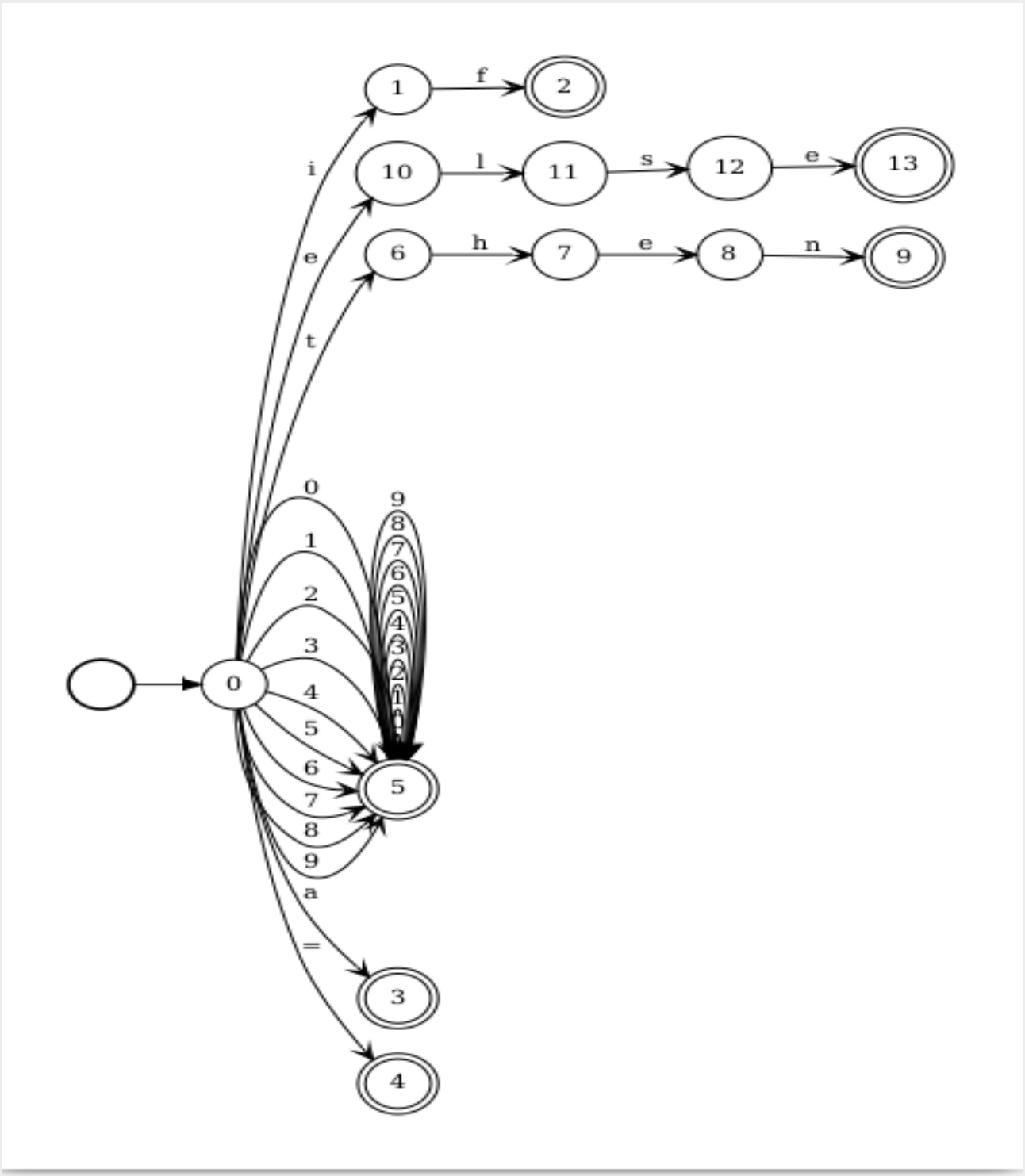
9. Minimiser



10. Reconnaissance d'un mot



11. Automate de reconnaissance



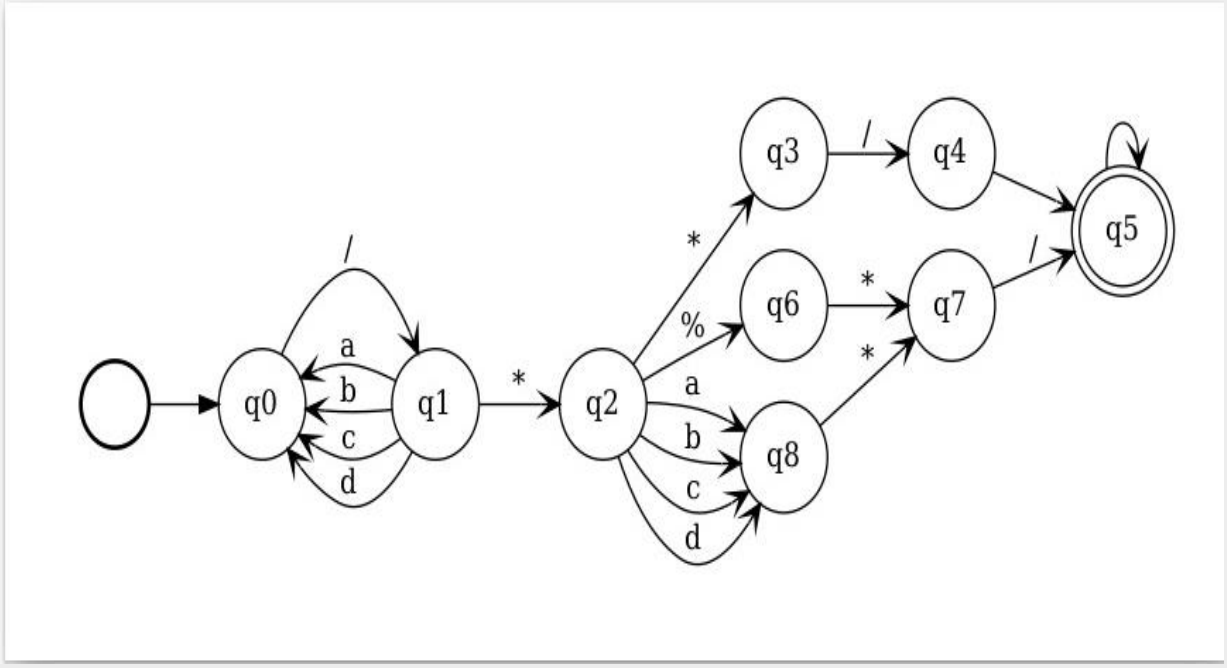
VII. LANGAGE DES COMMENTAIRES

A. Problème

Produire un outil qui permet la reconnaissance des commentaires dans texte donné. Les commentaires ont la forme : / ∗ w ∗ /, où le commentaire proprement dit w ne peut pas contenir le facteur ∗/, sauf s'il est immédiatement précédé du caractère d'échappement %. Vous définirez vous même votre alphabet.

1. Langage de commentaire

2. Automate langage des commentaires



VIII. CONCLUSION

Les automates sont des outils puissants pour modéliser des systèmes répétitifs et pour résoudre des problèmes de reconnaissance de motifs. En Python, les bibliothèques `pydot` et `graphviz` permettent de créer et de visualiser des automates facilement. N'hésitez pas à explorer ces bibliothèques pour découvrir d'autres fonctionnalités intéressantes !