REPUBLIQUE DU CAMEROUN
Paix-Travail-Patrie
****
Université de Yaoundé I
****
Faculté des Sciences
****
Département d'informatique

REPUBLIC OF CAMEROON
Peace-Work-Fatherland
****
University of Yaounde I
****
Faculty of Sciences
****
Department of Computer Sciences

RAPPORT DE TP D'INF342

THEME :

Création d'un automate et vérification des chaînes de caractères acceptées en Java

Rédiger par :

- **KONGA MARC YVAN** ***20U2938***
- **TANGA NOUNEGNE DUREL** ***21T2617***
- **FOKOU NZOKOU FRANCK ADAM** ***19M2451***
- **DJIHANE MA'AI ZERI** ***20V2201***
- **NZALI KENGMEGNIE JOHAN ALAIN** ***20U2709***
- **FOUADJI FOSSO HERMANN EDMOND** ***21T2822***
- **BOUMKWO EMMANUEL DIEUDONNE** ***18T2526***
- **TEGUIA YVES** ***19M2389***
- **WOUOTCHIESSI WAFEU JOËL** ***20U2967***
- **TANKOUA TCHATCHOU STEPHANE** ***20U2847***
- **TCHIO NOUBOSSI ULRICH WILFRIED** ***21T2544***
- **MICHAEL LOUIS JACON** ***19M2553***

Superviser par :
Dr ETIENNE KOUOKAM

Année Académique :

2023-2024

Création d'un automate et vérification des chaînes de caractères acceptées en Java

REPARTITION DES TACHES

Chef de groupe : **KONGA MARC YVAN *20U2938***

- Création de l'automate :
  - **KONGA MARC YVAN *20U2938***
  - **FOKOU NZOKOU FRANCK ADAM *19M2451***
  - **DJIHANE MA'AI ZERI *20V2201***
  - **NZALI KENGMEGNIE JOHAN ALAIN *20U2709***

- Vérification des chaînes de caractères acceptées
  - **FOUADJI FOSSO Hermann Edmond *21T2822***
  - **BOUMKWO EMMANUEL DIEUDONNE *18T2526***
  - **TEGUIA YVES *19M2389***
  - **WOUOTCHIESSI WAFEU Joel *20U2967***

- Vérification de la nature de l'automate
  - **TANGA NOUNEGNE DUREL *21T2617***
  - **TANKOUA TCHATCHOU STEPHANE *20U2847***
  - **TCHIO NOUBOSSI Ulrich Wilfried *21T2544***
  - **MICHAEL LOUIS JACON *19M2553***

Création d'un automate et vérification des chaînes de caractères acceptées en Java

SOMMAIRE

Création d'un automate et vérification des chaînes de caractères acceptées en Java

Introduction

Ce rapport présente les différentes étapes de la création d'un automate fini déterministe en Java, ainsi que la vérification des chaînes de caractères acceptées par ce dernier. Le travail a été divisé en trois tâches distinctes : la création de l'automate, la vérification des chaînes de caractères acceptées, et la détermination de la nature de l'automate (déterministe fini ou non).

Création d'un automate et vérification des chaînes de caractères acceptées en Java

I. Choix du langage de programmation

Le langage ***Java*** a été choisi pour ce projet pour plusieurs raisons :

- ***Java*** est un langage orienté objet, ce qui facilite la modélisation et l'implémentation de l'automate sous forme de classes et d'objets.
- ***Java*** dispose d'une vaste bibliothèque standard qui offre de nombreuses fonctionnalités utiles pour la manipulation de chaînes de caractères et la gestion des états de l'automate.
- ***Java*** est un langage largement utilisé dans l'industrie et bénéficie d'une forte communauté de développeurs, facilitant ainsi la recherche de ressources et d'informations.
- ***Java*** est un langage portable, permettant l'exécution du programme sur différentes plateformes sans nécessiter de modifications majeures.

II. Tâche 1 : Création de l'automate

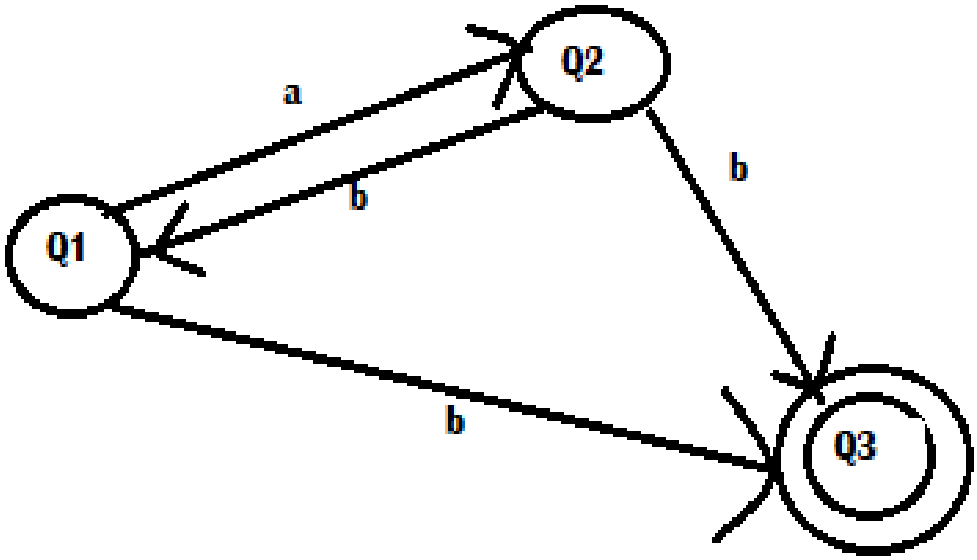
La première tâche consiste à créer un automate fini déterministe (AFD) en Java. Cet automate sera représenté par une classe AutomateCreator qui encapsulera les éléments suivants :

- Un ensemble d'états
- Un alphabet d'entrée
- Une fonction de transition
- Un état initial
- Un ensemble d'états finaux

La classe ***AutomateCreator*** fournira des méthodes pour ajouter des états, définir l'alphabet d'entrée, spécifier la fonction de transition, et identifier l'état initial et les états finaux. Elle comportera également une méthode accept () qui prendra une chaîne de caractères en entrée et déterminera si elle est acceptée par l'automate.

Création d'un automate et vérification des chaînes de caractères acceptées en Java

Illustration :




```
C:\Users\GREAT GEEK KURAMA\Desktop\TP\kuokam\test f>java AutomateCreator
Nombre d'états : 3
Nom de l'état 1 : Q1
Cet état est-il initial ? (true/false) : true
Cet état est-il final ? (true/false) : false
Nom de l'état 2 : Q2
Cet état est-il initial ? (true/false) : false
Cet état est-il final ? (true/false) : false
Nom de l'état 3 : Q3
Cet état est-il initial ? (true/false) : false
Cet état est-il final ? (true/false) : true
Nombre de transitions : 4
État de départ : Q1
État d'arrivée : Q2
Caractère d'entrée : a
État de départ : Q2
État d'arrivée : Q1
Caractère d'entrée : b
État de départ : Q1
État d'arrivée : Q3
Caractère d'entrée : b
État de départ : Q2
État d'arrivée : Q3
Caractère d'entrée : b
```

III. Tâche 2 : Vérification des chaînes de caractères acceptées

La deuxième tâche consiste à vérifier si une chaîne de caractères donnée est acceptée par l'automate créé précédemment. Pour ce faire, la classe AutomateCreator implémentera la méthode accept () qui, à partir de l'état initial et de la chaîne d'entrée, effectuera une simulation du comportement de l'automate. Si l'automate atteint un état final après avoir parcouru toute la chaîne, alors celle-ci est acceptée. Sinon, elle est rejetée.

Algorithme de vérification d'acceptation

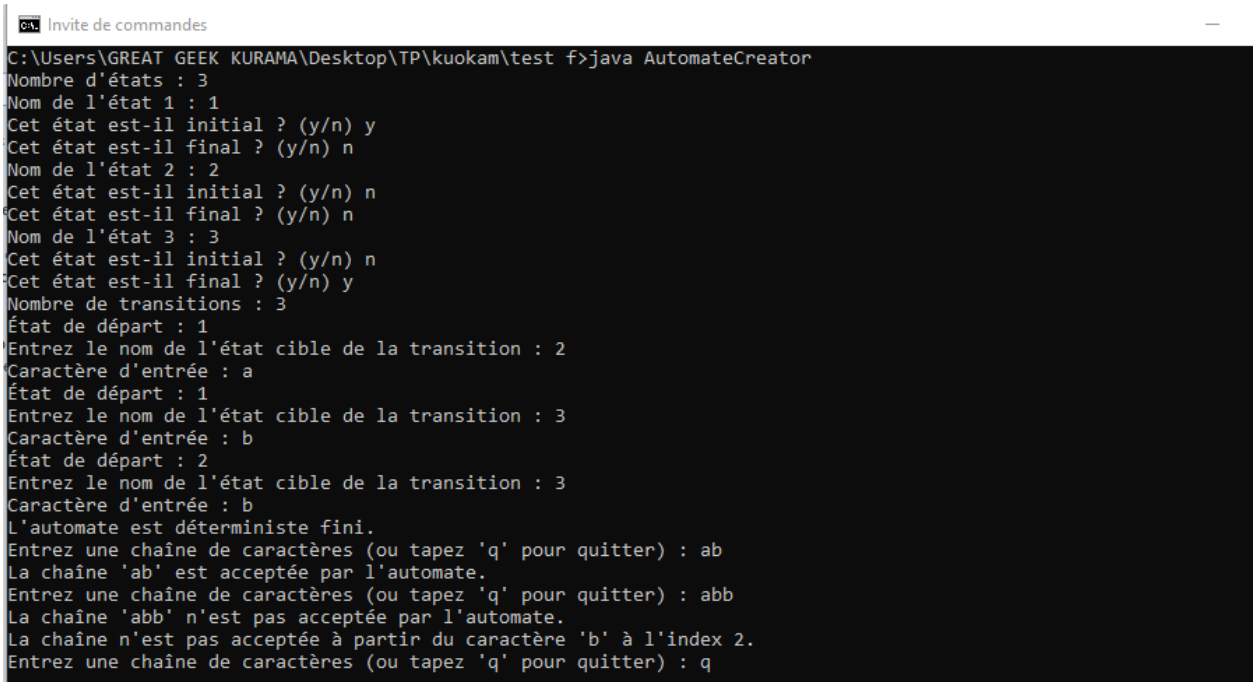
L'algorithme de vérification d'acceptation d'une chaîne de caractères par l'automate se déroule comme suit :

1. Initialiser l'état courant à l'état initial de l'automate.
2. Parcourir la chaîne de caractères entrée, caractère par caractère :
   a. Rechercher la transition partant de l'état courant et étiquetée par le caractère courant de la chaîne.
   b. Si une telle transition existe, mettre à jour l'état courant en fonction de la transition.
   c. Si aucune transition n'est trouvée, la chaîne est rejetée.
3. Une fois le parcours de la chaîne terminé, vérifier si l'état courant est un état final de l'automate :
   a. Si oui, la chaîne est acceptée.
   b. Sinon, la chaîne est rejetée.

procédure accepte
entrée : un AFD (K, T, M, S_0, F)
un tableau de caractères u indicé de 1 à n
sortie : retourne vrai si $u[1..n]$ appartient au langage, faux sinon
debut
$etatCrt \leftarrow S_0$
$i \leftarrow 1$
tant que $i \leq n$ faire
$etatCrt \leftarrow M(etatCrt, u[i])$
$i \leftarrow i + 1$
fin tant que
si $etatCrt \in F$ alors retourne vrai sinon retourne faux
fin

Cet algorithme simule le comportement de l'automate en partant de l'état initial et en suivant les transitions correspondant aux caractères de la chaîne d'entrée. S'il atteint un état final après avoir parcouru toute la chaîne, celle-ci est acceptée, sinon elle est rejetée.

Illustration :



IV. Tâche 3 : Vérification de la nature de l'automate

La troisième tâche consiste à déterminer si l'automate créé est déterministe fini ou non. Pour ce faire, la classe ***FiniteAutomaton*** comportera une méthode ***isDeterministic ()*** qui vérifiera les propriétés suivantes :

- L'automate possède un seul état initial.
- Pour chaque état et chaque symbole d'entrée, il existe au plus une transition possible.

Si ces conditions sont remplies, l'automate est considéré comme déterministe fini. Sinon, il est non-déterministe.

Algorithme de vérification de la nature de l'automate
L'algorithme de vérification de la nature de l'automate, déterministe fini ou non, se déroule comme suit :

1. Vérifier que l'automate possède un seul état initial.
2. Pour chaque état de l'automate et chaque symbole de l'alphabet d'entrée, vérifier qu'il existe au plus une transition possible :

    a. Parcourir toutes les transitions partant de l'état courant.

    b. Compter le nombre de transitions étiquetées par le symbole courant de l'alphabet.

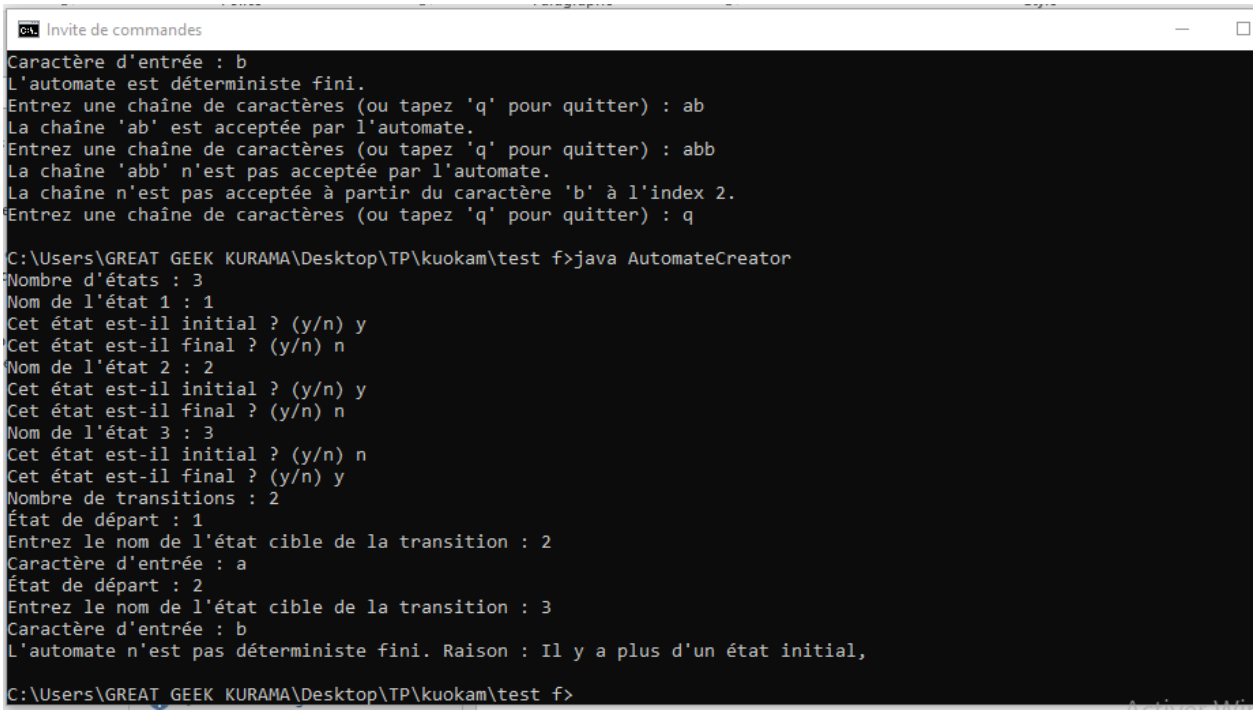
    c. Si le nombre de transitions est strictement supérieur à 1, alors l'automate est non-déterministe.

3. Si les deux conditions précédentes sont remplies, alors l'automate est déterministe fini.

Cet algorithme vérifie les propriétés caractéristiques d'un automate fini déterministe : l'unicité de l'état initial et l'existence d'au plus une transition pour chaque état et chaque symbole d'entrée. Si ces conditions sont respectées, alors l'automate est considéré comme déterministe fini.

Ces deux algorithmes sont implémentés dans les méthodes ***accept ()*** et ***isDeterministic ()*** de la ***classe FiniteAutomaton***, permettant ainsi de vérifier l'acceptation des chaînes de caractères et la nature de l'automate.

Illustration :



V. Défis et limitations

Lors de la réalisation de ce projet, certains défis ont été rencontrés :

- Une représentation visuelle de l'automate après sa création
- Une représentation visuelle du parcours de l'automate pour la validation de la chaine de caractère
- Gestion de la représentation des états et des transitions : la modélisation de l'automate sous forme de classes et d'objets a nécessité une réflexion approfondie sur la meilleure façon de représenter les états et les transitions.
- Implémentation de l'algorithme de vérification d'acceptation : la conception de l'algorithme permettant de déterminer si une chaîne de caractères est acceptée par l'automate a représenté un défi majeur.
- Vérification de la nature de l'automate : la mise en place de la méthode permettant de déterminer si l'automate est déterministe fini ou non a nécessité une compréhension approfondie des propriétés des automates finis.

Conclusion

Ce rapport a présenté les différentes étapes de la création d'un automate fini déterministe en Java, ainsi que la vérification des chaînes de caractères acceptées par ce dernier. Le choix du langage Java a été justifié, et les trois tâches principales ont été détaillées : la création de l'automate, la vérification des chaînes de caractères acceptées, et la détermination de la nature de l'automate. Bien que certains défis aient été rencontrés, ce projet a permis d'acquérir une meilleure compréhension des concepts liés aux automates finis et de leur implémentation en programmation orientée objet.