PLAN DU TRAVAIL

INTRODUCTION

Les automates finis déterministes (AFD) et les automates finis déterministes minimales (AFDM) sont des concepts fondamentaux en informatique théorique et en théorie des langages formels. Ils jouent un rôle crucial dans la modélisation et la reconnaissance de langage régulier, ainsi que dans la conception de systèmes logiciels tels que les compilateurs et les analyseurs syntaxiques. Les AFD sont utilisés pour reconnaitre les langages réguliers, c'est-à-dire des ensembles de mots sur l'alphabet $\sum$ qui respectent certaines règles de formation. Ils sont simples à implémenter et à analyser, mais peuvent parfois être plus grands que nécessaires en nombre d'états. Les AFDM sont des AFD qui ont été réduits au minimum de nombre d'états nécessaire pour reconnaitre le même langage. La minimisation d'un AFD consiste à fusionner les états équivalents pour obtenir un automate plus compact tout en conservant les mêmes capacités de reconnaissance.

I- AUTOMATE FINI DETERMINISTE

A- DEFINITION FORMELLE D'UN (AFD)

Un automate fini déterministe est défini formellement comme un quintuplé (Q, Σ, δ, q0, F) où :

- **Q** est un ensemble fini d'états.
- **Σ** est l'alphabet.
- **δ** est l'ensemble des transitions, défini par une fonction δ : Q × Σ → Q.
- **q0** est l'état initial.
- **F** est l'ensemble des états finaux.

L'automate fini déterministe peut être représenté graphiquement par des arcs étiquetés par des symboles. Un automate fini déterministe accepte un mot si, en partant de l'état initial et en suivant les transitions correspondant aux symboles du mot, on arrive à un état final à la fin du mot.

B- FONCTION DE LA TRANSITION D'UN AFD

La fonction de transition d'un automate fini déterministe est une fonction qui associe à chaque état de l'automate et à chaque symbole de l'alphabet une transition vers un autre état. Formellement, cette fonction peut être définie comme suit :

- Un AFD est un quintuplé ((Q, \Sigma, \delta, q0, F)) où :
  - (Q) est un ensemble fini d'états.
  - (\Sigma) est un alphabet fini (ensemble de symboles).
  - (\delta : Q \times \Sigma \to Q) est une fonction de transition qui associe à chaque état et à chaque symbole d'entrée un état de sortie.
  - (q_0) est l'état initial.
  - (F) est l'ensemble des états finaux.
  - L'automate est dit **déterministe** car la fonction de transition (\delta) est définie de manière unique pour chaque

état et chaque symbole d'entrée. Il n'y a pas d'ambiguïté dans les transitions.

La fonction de transition permet de déterminer l'état dans lequel l'automate doit passer en fonction de l'état actuel et du symbole lu sur l'entrée. Elle est déterministe car pour chaque combinaison d'état et de symbole, il n'y a qu'une seule transition.

Exemple d'un AFD

Comment savoir si un automate à nombre d'états fini est déterministe ou non déterministe ?

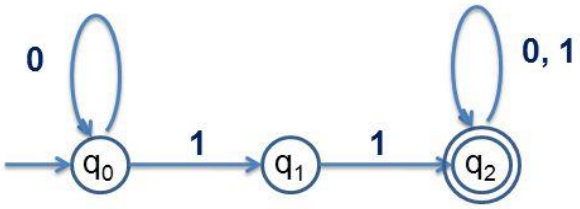
Un automate est déterministe si et seulement si les deux conditions suivantes sont vérifiées :

L'automate possède un et un seul état initial ;

Pour chaque état q et pour chaque lettre a, il existe au plus une transition issue de q d'étiquète a.

Automate Fini Déterministe

- $Q = \{q_0, q_1, q_2\}$; $\Sigma = \{0,1\}$; q_0 est l'état initial; $F=\{q_2\}$ est l'ensemble des états terminaux



- Langage reconnu par l'automate:

$0^*11\ \Sigma^*$

APPLICATION DANS LA VIE REELLE

1. Systèmes de contrôle d'accès : les automates finis déterministes peuvent être utiliser pour modéliser et contrôler l'accès à un système, par exemple pour autoriser ou refuser l'accès à une zone sécurisée en fonction d'un code d'identification.

2. Jeux vidéos: les automates finis déterministes peuvent être utilisés pour modéliser le comportement des personnages non joueurs (PNJ) dans les jeux vidéo, par exemple pour définir des stratégies de mouvement ou de prise de décision en fonction des actions du joueur.

II- AUTOMATES FINIS DETERMINISTES MINIMAL

A. DEFINITION FORMELLE D'UN AFDM

Un automate fini déterministe minimisé (AFDM) est un automate fini déterministe (AFD) qui a été réduit au minimum de nombre d'états nécessaires pour reconnaître le même langage que l'AFD d'origine. Formellement, un AFDM est défini par un quintuplé (Q',Σ,δ',q0',F'), où:

-Q' est un ensemble fini d'états du AFDM, qui est généralement plus petit que l'ensemble d'états de l'AFD d'origine ;

-Σ est un alphabet fini d'entrées ;

-δ' est une fonction de transition du AFDM qui associe à chaque état et symbole d'entrée, une transition vers un nouvel état dans Q' ;

-q0 'est l'état initial du AFDM

-F' est un ensemble d'états finaux ou acceptants du AFDM La différence principale entre un AFD et un AFDM réside dans le fait que l'AFDM a été optimisé en fusionnant les états équivalents pour réduire le nombre total d'états tout en conservant la même capacité de reconnaissance du langage régulier spécifié. La minimisation d'un AFD pour obtenir un AFDM implique généralement des algorithmes basés sur l'équivalence d'états, tels que l'algorithme de minimisation de Moore ou l'algorithme de minimisation de Hopcroft.

B. MINIMISATION D'UN AUTOMATE FINI DETERMINISTE

LES ALGORITHMES DE MINIMISATION

Algorithme de Moore

Algorithme de Moore de minimisation d'un automate fini.

L'algorithme de Moore est un algorithme utilisé pour minimiser un automate fini déterministe (AFD). Il se base sur l'équivalence des états en termes de langage. Le principe est le suivant :
1. Commencer par diviser l'ensemble des états de l'automate en deux ensembles : les états finaux et les états non finaux.
2. Ensuite, pour chaque paire d'états (q, p) dans les deux ensembles, vérifier s'ils sont distinguables ou non. Deux états sont dits distinguables s'ils ont des comportements différents sur au moins un symbole d'entrée.
3. Regrouper les états indistinguables ensemble pour former des classes d'équivalence. Ces classes regroupent des états qui ont le même comportement sur tous les symboles d'entrée.
4. Répéter l'étape précédente jusqu'à ce qu'il n'y ait plus de modifications à apporter aux classes d'équivalence.
5. Une fois que la division est stable, chaque classe d'équivalence forme un nouvel état de l'automate minimal.
6. Enfin, reconstruire l'automate minimal en remplaçant les classes d'équivalence par des états uniques et en mettant à jour les transitions en conséquence.

En suivant ces étapes, l'algorithme de Moore permet de minimiser un automate fini déterministe tout en préservant le langage de l'automate initial

Algorithme de Hopcroft

Algorithme de Hopcroft est utilisé également pour la minimisation d'un automate fini. . Contrairement à l'algorithme de Moore qui se base sur l'équivalence des états en termes de langage, l'algorithme Hopcroft se base sur la partition des états en fonction de leurs transitions. L'algorithme, appelé ainsi d'après son inventeur, est dû à John Hopcroft et il a été présenté en 1971; il opère par raffinement successif d'une partition initialement grossière de l'ensemble des états de l'automate déterministe fini à minimiser.

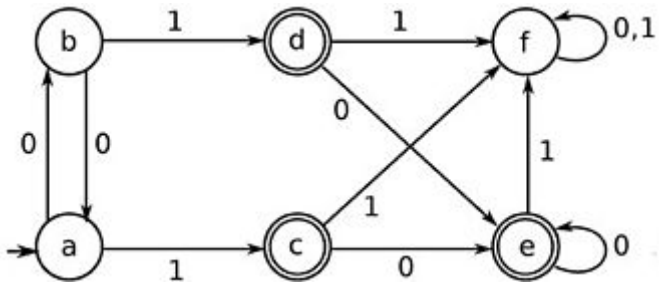
Principe

L'algorithme de Hopcroft est un algorithme utilisé pour minimiser un automate fini déterministe (AFD). L'objectif de la minimisation d'un AFD est de réduire le nombre d'états de l'automate tout en conservant son langage.

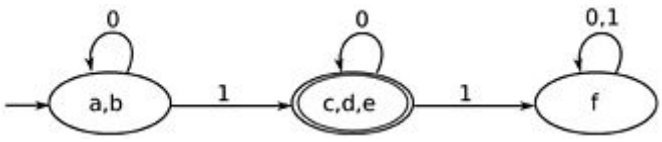
Le principe de l'algorithme de Hopcroft est le suivant :
1. Commencer par diviser l'ensemble des états de l'automate en deux ensembles : les états finaux et les états non finaux.
2. Ensuite, diviser chaque ensemble en sous-ensembles en fonction des transitions des états vers les différents états de l'automate. Cela permet de regrouper les états qui ont des comportements similaires.
3. Répéter l'étape précédente jusqu'à ce qu'il n'y ait plus de modifications à apporter aux regroupements d'états.
4. Une fois que la division est stable, chaque groupe d'états forme un nouvel état de l'automate minimisé.
5. Enfin, reconstruire l'automate minimal en remplaçant les groupes d'états par des états uniques et en mettant à jour les transitions en conséquence.

En suivant ces étapes, l'algorithme de Hopcroft permet de minimiser un automate fini déterministe de manière efficace tout en préservant le langage de l'automate initial.

Exemple de Minimisation en utilisant l'algorithme de moore



Dans cet automate, tous les états sont accessibles, les états *c, d* et *e* sont indistinguables, ainsi que les états *a* et *b*.



Automate minimal équivalent. Les états indistinguables sont regroupés en un seul état.

APPLICATION DANS LA VIE REELLE

1. Traitement du langage naturel: Les automates finis déterministes minimaux sont utilisés pour reconnaître et analyser les structures grammaticales dans le langage naturel, ce qui est essentiel dans les applications de traitement automatique du langage naturel comme la reconnaissance de la parole et la traduction automatique.

2. Reconnaissance de motifs: Les automates finis déterministes minimaux peuvent être utilisés pour la reconnaissance de motifs dans des séquences de caractères, ce qui est utile dans des domaines tels que la bioinformatique pour l'analyse de séquences d'ADN ou de protéines.

CONCLUSION

La minimisation des automates finis déterministes (AFD) est un processus essentiel en informatique théorique, offrant une méthode efficace pour simplifier la représentation des systèmes à états finis. À travers cet exposé, nous avons exploré en profondeur les concepts fondamentaux des AFD, leur fonctionnement, ainsi que les différentes techniques de minimisation, notamment l'algorithme de Hopcroft et l'algorithme de Moore. Nous avons également souligné l'importance pratique de la

minimisation dans la réduction de la complexité des machines à états, l'amélioration des performances des algorithmes et la facilitation de la compréhension des systèmes. En suivant les étapes du partitionnement ou de l'équivalence, nous avons illustré comment regrouper les états équivalents ensemble, formant ainsi un automate minimal qui conserve le même langage que l'automate original. Ces méthodes de minimisation, bien que différentes dans leur approche, visent toutes à atteindre un objectif commun : réduire le nombre d'états tout en préservant la fonctionnalité de l'automate.