Rapport Projet de stage : Limitedness

Brebant Alexandre
Xue Juedong

24 juin 2014

Table des matières

I Mise en place théorique de l'algorithme 4

1 De l'automate aux matrices 5

2 Matrices idempotentes 6

3 Matrices stables 7

II Exécution détaillée du programme 8

4 Réutilisation du programme d'Adrien Boussicault 9

5 Création et manipulation des matrice 10

6 Structure de stockage des matrices 11

7 Résolution du problème de limitedness 12

Introduction au problème

Durant ce stage, nous avons voulu approfondir nos connaissances dans une branche de l'informatique théorique que nous avons commencé à étudier lors de ce dernier semestre de licence. Ce sont donc les automates et les interrogations qui gravitent autour de cette structure qui nous ont intéressé.
Un automate est une machine abstraite qui va accepter ou rejeter des mots dans un alphabet fini. Plus formellement, un automate est définit, par 5 ensembles, de la façon suivante :

$$\mathcal{A} = \{A, Q, I, F, \delta\} \text{ avec,}$$

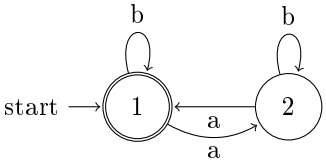
A : l'alphabet
Q : l'ensemble des états
I : l'ensemble des états initiaux
F : l'ensemble des états finaux
δ : l'ensemble des transitions

Pour représenter graphiquement un automate, on utilise un graphe orienté dans lequel les sommets du graphe sont les états de l'automate et où chaque arrête correspond à une transition de l'automate. Un mot est accepté par un automate s'il existe une chemin dans celui-ci, partant d'un état initial et allant jusqu'à un état final en lisant les lettres une par une. Un tel chemin est appelé *chemin acceptant* et l'ensemble des mots accepté par un automate est appelé *Langage*, on le note $\mathcal{L}(\mathcal{A})$.

Ainsi les automates permettent de caractériser certains langage, par exemple, les automate finis définissent ce qu'on appelle les langages rationnels ou encore les automates à pile qui permettent de définir les langages algébriques (voir [1]) et qui sont grandement utiliser en analyse syntaxique.

Voici l'exemple d'un automate très basique qui accepte tous les mots sur l'alphabet $\{a, b\}$ qui comportent un nombre pair de a.
Cet automate permet d'introduire une notion élémentaire de la théorie des automates : le déterminisme. Un automate est déterministe si et seulement si il possède un unique état initial et que pour état, il existe au plus, une transition par lettre de l'alphabet. L'automate ci-dessous est donc déterministe.

$$\mathcal{A} = \left(\quad A = \{a, b\},\ Q = \{1, 2\},\ I = \{1\},\ F = \{1\},\ \delta = \left\{ \begin{array}{lcl} (1, a) & \to & 2 \\ (2, a) & \to & 1 \\ (1, b) & \to & 1 \\ (2, b) & \to & 2 \end{array} \right. \right)$$



La théorie des automates et des langages est un domaine très actif depuis le milieu du 20e siècle et il existe aujourd'hui plusieurs modèles d'automate nés pour répondre à certaines problématiques. C'est un de ces modèle particulier qui nous a intéressé lors de ce stage : les automates à distance (distance automata). Ce sont des automates finis, non déterministes, dans lesquels les transitions sont pondérées. On associe donc un coût pour chaque mot accepté par l'automate. Le coût d'un mot correspond au minimum des coûts de tous ses chemins acceptants dans l'automate .
Ce type d'automate a été présenté par Hashiguchi dans [3] pour la résolution d'une célèbre problématique, la question de hauteur d'étoile (star height problem) [2]. Mais les automates à distance posent également un autre problème, celui limitedness.
Le problème de limitedness est très simple à comprendre, il est simplement question de savoir si il existe une majoration des coûts des mots acceptés par un automate à distance.

Le travail d'Hashiguchi étant considéré comme difficile à comprendre, d'autre chercheurs, comme Simon [5] ou Leung [4], ont étudié le problème et apporté d'autres solutions sur le sujet.

Notre travail a donc consisté, dans un premier temps, à lire et comprendre les différents travaux de ces chercheurs afin d'être capable d'implémenter un algorithme permettant la détection de limite dans un automate à distance.

Première partie

Mise en place théorique de l'algorithme

Chapitre 1

De l'automate aux matrices

Le papier dont nous nous somme le plus servit afin de comprendre et résoudre le problème de limitedness, est [5] mais il nous a d'abord fallu comprendre la complexité du problème.

Chapitre 2

Matrices idempotentes

Chapitre 3

Matrices stables

Deuxième partie

Exécution détaillée du programme

Chapitre 4

Réutilisation du programme d'Adrien Boussicault

(présentation des différents modules et fonctions disponible + détails des ajouts pour la gestion des coût, de l'affichage et autres fonctions additionnelles que nous avons ajouté)

Chapitre 5

Création et manipulation des matrice

(creer_matrice_transition, multiplication, etc...)

Chapitre 6

Structure de stockage des matrices

(mautomate, descrption des fonctions de création)

Chapitre 7

Résolution du problème de limitedness

Bibliographie

[1] J. Berstel, D. Beauquier, and P. Chrétienne. *Elements d'algorithmique*. 2005.

[2] L. Eggan. Transition graphs and the star-height of regular events. *Michigan Mathematical Journal*, 10 :385–397, 1963.

[3] K. Hashiguchi. Limitedness theorem on finite automata with distance functions. *J. Comput. Syst. Sci.*, 5201 :233–244, 1982.

[4] H. Leung and V. Podolskiy. The limitedness problem on distance automata : Hashiguchi's method revisited. *Theoret. Comput. Sci.*, 310(1-3) :147–158, 2004.

[5] I. Simon. On semigroups of matrices over the tropical semiring. *RAIRO Inform. Théor. Appl.*, 28(3-4) :277–294, 1994.