Projet de Théorie des Langages

Corentin BEAULIEU - 21 90 69 66
Victor SCHWEISTHAL - 21 80 21 03

Sommaire

1 Introduction **1**
1.1 Composition de l'archive . . . 1
1.2 Usage du programme . . . 1
1.3 Description du langage du code des machines . . . 1

2 Simulation de l'exécution d'une machine de Turing **2**
2.1 Question 1 . . . 2
2.1.1 Structures de données . . . 2
2.1.2 Fonctions d'initialisation . . . 2
2.2 Question 2 . . . 2
2.3 Question 3 . . . 2
2.4 Question 4 . . . 3
2.5 Question 5 . . . 3
2.5.1 Machine de Turing 1 . . . 3
2.5.2 Machine de Turing 2 . . . 3
2.5.3 Machine de Turing 3 . . . 3

3 Réductions entre différents modèles de machines de Turing **4**
3.1 Question 6 . . . 4
3.2 Question 7 . . . 5
3.3 Question 8 . . . 5
3.3.1 Machine de Turing Q6 . . . 6
3.3.2 Machine de Turing Q7 . . . 6

1 Introduction

1.1 Composition de l'archive

L'archive remise contient les fichiers suivants :

`compte_rendu.pdf` le compte rendu que vous êtes en train de lire.

`src/main.c` contenant la fonction *main* du programme.

`src/partie1.c` contenant les implémentations des fonctions de base d'initialisation et de manipulation de la machine de Turing nécessaire pour répondre aux questions de la partie 1 directement et des autres parties de façon indirecte.

`src/partie1.h` contenant les structures de données definissant les machines de Turing ainsi que les signatures des fonctions de base d'initialisation et de manipulation des machines.

`src/partie2.c` contenant l'implémentation des fonctions demandées dans la partie 2.

`src/partie2.h` contenant les signatures des fonctions demandées dans la partie 2.

`mt/mt1` contenant le code dans notre langage d'une machine de Turing pour tester la simulation.

`mt/mt2` contenant le code dans notre langage d'une machine de Turing pour tester la simulation.

`mt/mt3` contenant le code dans notre langage d'une machine de Turing pour tester la simulation.

`mt/mtQ6` contenant le code dans notre langage d'une machine de Turing. Elle permet d'illustrer la question 6 comme demandé dans la question 8.

`mt/mtQ7` contenant le code dans notre langage d'une machine de Turing. Elle permet d'illustrer la question 7 comme demandé dans la question 8.

`src/Makefile` disposant de diverses cibles :
- `make` permet de lancer la cible *run* qui compile et lance le programme sur le fichier `mt/mt3` avec le mot `100`.
- `make test` permet de lancer la cible *test* qui compile et lance le programme sur les cinq fichiers contenus dans le dossier `mt`.
- `make mt` qui compile le programme et nomme le binaire `mt`.
- `make clean` qui supprime les divers produits de la compilation et de l'exécution des tests.

1.2 Usage du programme

L'usage du programme compilé est le suivant :

./mt [option] fichier mot_entrée

avec `fichier` le chemin vers le fichier contenant le code de la machine de Turing que l'on souhaite simuler et `mot_entrée` le mot écrit sur la bande au début de la simulation. Les options sont les suivantes
- `b` : permet de passer d'une simulation sur bande semi-infinie à droite à une simulation sur une bande biinfinie.
- `6` : lance le programme en mode transformation de bande bi-infinie à bande infinie à droite comme demandé dans la question 6.
- `7` : lance le programme en mode transformation de machine avec le codage demandé dans la question 7.

A savoir que les options `6` et `7` sont incompatibles. L'option `b` est sans effet si une autre option est activée.
Si une des deux dernières options est activée, l'usage change pour le suivant

./mt [option] fichier [destination]

où *destination* est le chemin vers le fichier qui contiendra le code de la machine après transformation. Si aucun chemin n'est précisé, le code sera écrit dans le fichier `sortieMT`.

1.3 Description du langage du code des machines

Le langage du code des machines de Turing est basé sur celui de https ://turingmachinesimulator.com/.

Tout d'abord le code doit être écrit dans une fichier texte. Il commence par l'état final sur la première ligne. Puis, il doit être composé d'une transition uniquement par ligne. Les espaces et les sauts de lignes autre que pour changer de transition sont interdits.

Les transitions s'écrivent sous forme d'un quintuplet chaque élément étant séparé du suivant par une virgule. Il ne doit y avoir aucun espace.

Les quintuples sont de la forme :

état_départ,lettre_lue,état_arrivée,lettre_écrite,déplacement

avec

1. `état_départ` une chaine caractères de sept caractères maximum représentant le nom de l'état de départ de la transition.
2. `lettre_lue` un caractère ASCII représentant la lettre lue sur la bande au moment de la transition.
3. `état_arrivée` une chaine de caractères de sept caractères maximum représentant le nom de l'état d'arrivée de la transition.
4. `lettre_écrite` un caractères ASCII représentant la lettre écrite sous la tête de lecture pendant la transition.
5. `déplacement` un caractère représentant le déplacement de la tête de lecture parmi les suivants
   — `<` pour un déplacement d'une case sur la gauche.
   — `>` pour un déplacement d'une case sur la droite.
   — `-` pour un déplacement nul. La tête de lecture reste sur place.

Les états sont des chaines des caractères pour pouvoir répondre aux questions de la deuxième partie. Les caractères ne sont pas restreints à l'alphabet de travail donné dans le sujet pour pouvoir l'agrandir au besoin.

Une case vide est représenté par le caractère '_'.

Le but de ce projet est de concevoir un programme simulant des machines de Turing.

Ce document décrit une partie de la démarche menant à la réalisation de cette objectif.

2 Simulation de l'exécution d'une machine de Turing

2.1 Question 1

2.1.1 Structures de données

Nous avons défini trois structures de données pour représenter une machine de Turing.

Tout d'abord, nous avons défini une structure représentant la bande de la machine. On a choisit de faire une liste doublement chainée. La tête est positionnée sur l'élément dont l'on garde l'adresse dans notre structure de machine. Les cases vides ne sont pas allouées sauf si la tête passe dessus. Comme la bande est semi-infinie sur la droite, un déplacement sur la gauche lorsque la tête est positionnée sur la première ne pourra être fait et la tête restera sur place. Cela peut créer de mauvais résultats si ça n'a pas été pris en compte dans la conception de la machine de Turing.

Ensuite, on a défini une structure pour représenter les deltas transitions. C'est une quintuplet contenant : l'état de départ sous forme de chaine de caractères, un `char` pour la lettre lu, une chaine de caractères pour l'état d'arrivée, un `char` pour la lettre écrite et une énumération des différents déplacements possibles.

Enfin, on a une structure représentant une machine de Turing ainsi que sa configuration courante. Elle est composée d'une chaine de caractères pour l'état courant, d'un pointeur vers l'élément sous la tête de la bande, le nombre de deltas transitions, un tableau contenant toutes les deltas transitions de la machine et une chaine de caractères pour l'état final.

2.1.2 Fonctions d'initialisation

L'initialisation de la machine se divise en plusieurs étapes. Tout d'abord, l'état courant est initialisé à l'état initial qui est A.

Ensuite, la bande est initialisé à partir du mot d'entrée et la tête positionnée sur la première case.

Enfin, à partir du fichier donné en argument, l'état final est initialisé et le tableau contenant les deltas transitions est créé.

2.2 Question 2

Pour simuler un pas de calcul de la machine, on commence à chercher l'état courant comme état de départ d'une delta transition et la lettre sous la tête comme lettre lue. Si on ne trouve pas, une erreur est générée.

Alors, la lettre sous la tête devient la lettre à écrire et l'état courant devient l'état d'arrivée de la transition.

Enfin, selon le déplacement, on déplace la tête. Comme dit précédent, un déplacement sur la gauche alors que la tête est positionnée sur la première case est transformé par un déplacement sur place.

2.3 Question 3

Étant donné que l'état final est unique et dénote la fin du calcul, la simulation totale de la machine de Turing se déroule dans une boucle `while`. A chaque tour de boucle, on simule un pas en modifiant la configuration de la machine jusqu'à ce que l'état courant et l'état final soient le même.

2.4 Question 4

L'affichage que nous avons implémenté contient le numéro de l'étape de calcul à laquelle la machine est, l'état courant et l'état de la bande.

La bande est représentée par les caractères qu'elle contient les uns après les autres et séparés par un espace.

La position de la tête est représenté par des chevrons pointant sur une case particulière qui est celle sous la tête de lecture.

2.5 Question 5

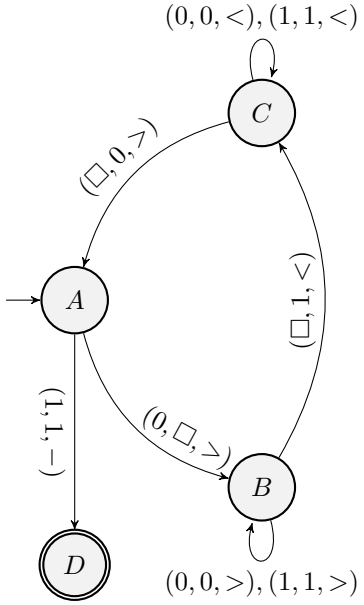
Vous trouverez ci-dessous une description sommaire des machines de Turing que l'on a écrit dans notre langage pour tester le fonctionnement de notre programme de simulation.

2.5.1 Machine de Turing 1

La machine dont le code est contenu dans le fichier `mt1` calcule la fonction suivante :

$$0^n \to 0^n 1^n$$

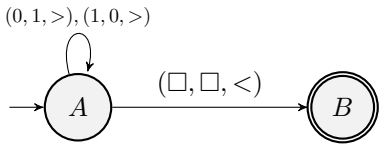
Elle se représente de la manière suivante



2.5.2 Machine de Turing 2

La machine dont le code est contenu dans le fichier `mt2` inverse les 0 et les 1 du mot d'entrée.

On la représente ainsi

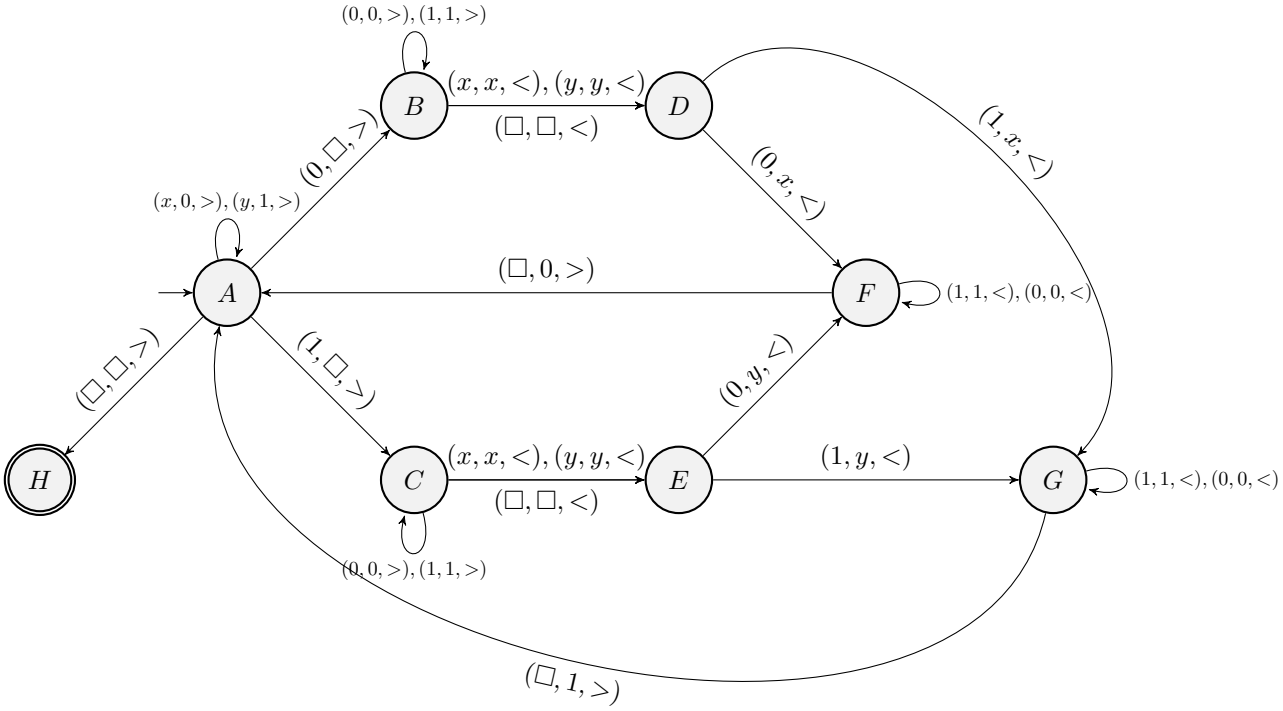


2.5.3 Machine de Turing 3

La machine dont le code est contenu dans le fichier `mt3` calcule la fonction

$$w \to ww$$

Pour pouvoir faire cette machine l'alphabet de travail a été agrandi et est donc $\Gamma = \{0, 1, x, y, \square\}$.



3 Réductions entre différents modèles de machines de Turing

3.1 Question 6

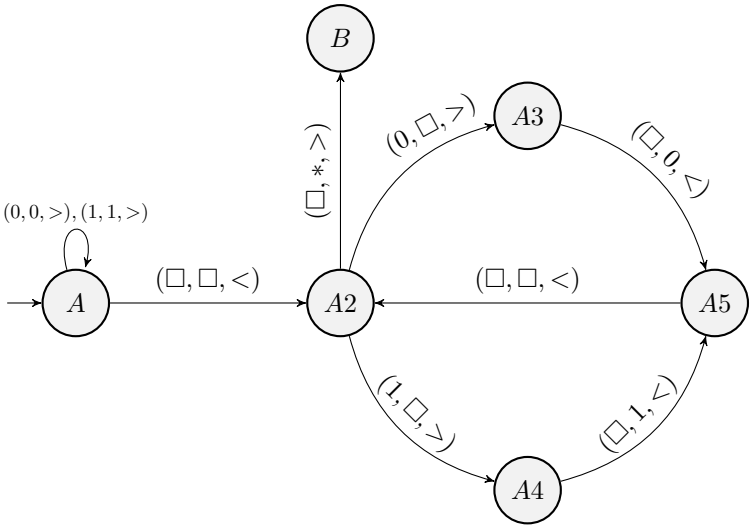
Pour passer d'un calcul sur une bande biinfinie à une bande semi infinie à droite, on a fait le choix de décaler toute la bande sur la droite si une transition nous emmène à gauche de la première case.

Pour cela, il a fallu rajouter la lettre '*' à l'alphabet de travail. Cette lettre permet de savoir où la bande s'arrête à gauche.

Puis, on décale tous les états d'une lettre plus loin dans l'alphabet dans toutes les transitions existantes (A devient B, B devient C...).

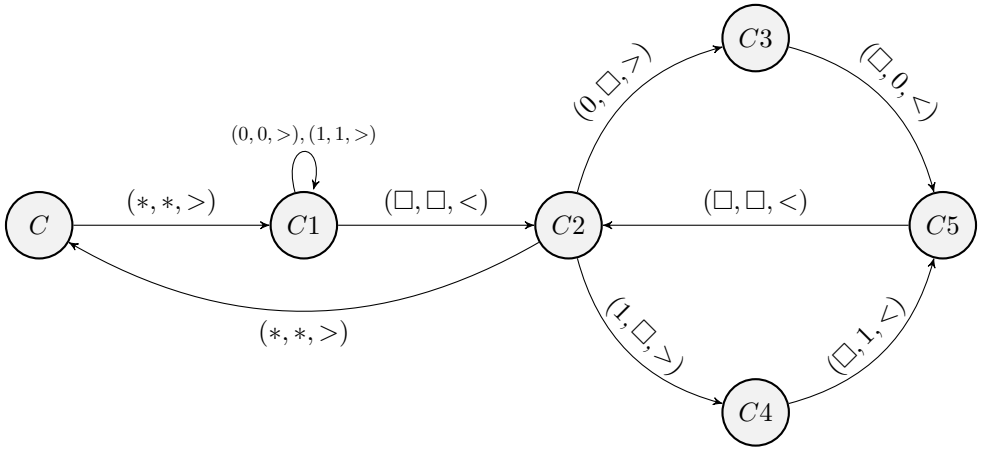
Ensuite, il faut rajouter des états et des transitions à l'automate de base.

Tout d'abord, on ajoute un nouvel état initial qui permet d'ajouter le caractère de début de bande après avoir décaler le mot d'entrée sur la droite. On peut voir ces nouveaux états et transitions ainsi



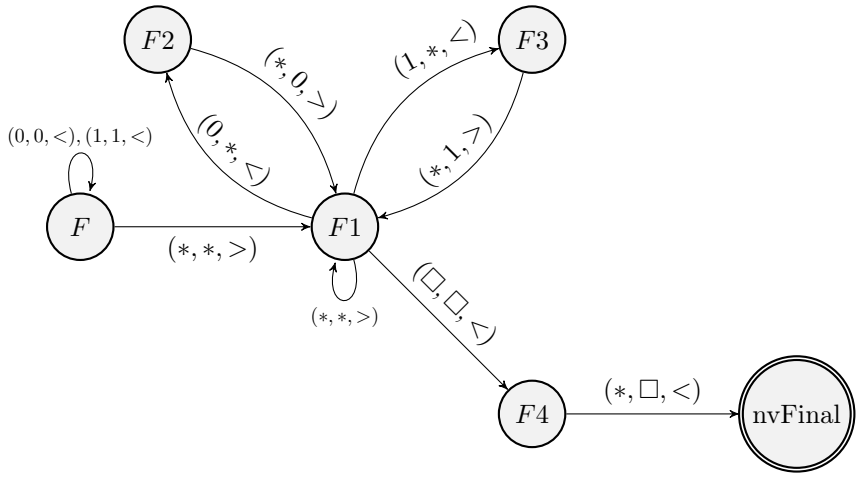
On ajoute ici quatre états et neufs transitions.

Pour chaque état non final de la machine de départ on rajoute les transitions suivantes qui permettent de décaler le mot sur la droite et ainsi écrire à sa gauche. On peut voir les transitions pour un état C quelconque ainsi



Pour chaque état non final, on ajoute donc cinq nouveaux états et dix transitions.

Enfin, on ajoute un nouvel état final que l'on appelle "nvFinal" pour que ne pas le confondre. On ajoute des transitions au départ de l'état anciennement final, que l'on nommera ici F, pour supprimer le caractère de début de bande et décaler le mot sur la gauche et ainsi obtenir la même sortie que la machine originel. On dessine ces nouvelles transitions et états ainsi



On ajoute donc en plus cinq états et dix transitions.
Au total, cette solution ajoute $5 \times (nbEtats + 2) - 1$ états et $10 \times (nbEtats + 2) - 1$ transitions.

3.2 Question 7

Pour la question 7 on construit une fonction traduisant une machine de Turing sur l'alphabet a,b,c,d en notre structure MT sur l'alphabet 0,1.

Pour cela on a besoin d'états supplémentaires :

— Les états de lecture puisque celle-ci se fait sur 2 bits, (A,a,B,b,>) engendrera par exemple (A,0,A0,0,>) et (A0,0,A00,0,<).

— On se trouve après la lecture de a à l'état A en A00.

Ensuite on traduit la transition originale sur l'alphabet a,b,c,d en instructions adaptées :

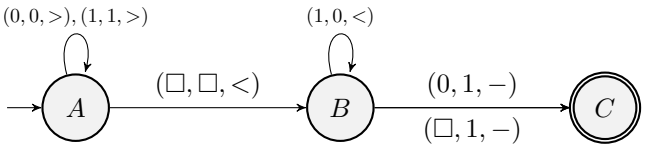
On veut maintenant écrire 01, se déplacer vers la droite et passer à l'état B. Cette instruction peut s'écrire "01»B" et on effectuera les instructions intermédiaires à la suite afin de se retrouver avec la bande modifiée et dans l'état B. On obtient donc les états supplémentaires (01»B,*,1»B,0,>) , (1»B,*,»B,1,-) , (»B,*,>B,*,>) , (>B,*,B,*,-).

3.3 Question 8

Vous trouverez ci-dessous une description sommaire des machines de Turing que l'on a écrit dans notre langage pour tester le fonctionnement de nos fonctions de transformation.

3.3.1 Machine de Turing Q6

La machine dont le code est contenu dans le fichier `mtQ6` incrémente d'un un nombre écrit en binaire. Elle se représente de la manière suivante



Pour pouvoir tester la transformation, on donnera un nombre composé uniquement de 1 comme mot d'entrée.

3.3.2 Machine de Turing Q7

La machine dont le code est contenu dans le fichier `mtQ7` change les a en d, les b en c, les c en b et les d en a. On la représente ainsi