2021-2022

Projet – Simulation de Machines de Turing

Version du 7 d´ecembre 2022

Le but de ce projet est d'exécuter pas à pas des machines de Turing, ainsi que de simuler des machines de Turing complexes par des machines simples. On appliquera également des règles de simplification sur les machines de Turing. Deux machines sont équivalentes si sur des entrées identiques, elles calculent le même résultat.

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

Le code du projet est à faire en C.

On utilisera le langage de description de machine du site https://turingmachinesimulator.com/. Vous pouvez restreindre ce langage pour vous simplifier le travail, par exemple interdire des espaces ou des sauts de ligne. Dans ce cas, il faut spécifier ces restritions avec un commentaire dans votre code.

Pour simplifier, on considérera des machines qui effectuent un calcul, avec un seul état final qui dénote la fin du calcul. Les états seront notés par des lettres majuscules et A sera l'état initial. L'alphabet d'entrée est $\{0, 1\}$, l'alphabet de travail $\{0, 1, \square\}$ et la bande est semi-infinie vers la droite. Dans la configuration initiale, la tête de lecture est positionnée à gauche de la bande.

Question 1 : Proposer une structure de données **MT** qui permet de représenter une machine de Turing ainsi que sa configuration (état courant, état de la bande, position de la tête de lecture). Écrire une fonction qui initialise une structure **MT** à partir d'un mot d'entrée et d'un fichier contenant une description d'une machine de Turing.

Question 2 : Écrire une fonction qui étant donné une machine de Turing, lui fait exécuter un pas de calcul.

Question 3 : Écrire une fonction qui prend comme argument un mot et une machine de Turing et qui simule le calcul de la machine sur le mot jusqu'à atteindre l'état final. Bravo, vous avez réalisé une machine universelle.

Question 4 : Modifier la fonction précédente pour que, à chaque pas de simulation, la configuration de la machine s'affiche.

Question 5 : Vous écrirez le code de trois machines de Turing (non triviales) de votre choix dans trois fichiers, et elles devront être simulées quand l'utilisateur tape *make test*.

2 Réduction entre différents modèles de machine de Turing

Chaque membre du binôme doit traiter une des deux transformations. Dans cette partie, les états seront des chaînes de caractères. Par exemple à partir de l'état C de la machine M, on peut se retrouver à créer les états $C1$, $C2$ et $C3$ dans la machine qui simule M.

Question 6 : Écrire une fonction qui lit dans un fichier le code d'une machine de Turing avec ruban bi-infini et qui écrit dans un autre fichier le code d'une machine équivalente avec un ruban infini seulement vers la droite.

Question 7 : Écrire une fonction qui lit dans un fichier le code d'une machine de Turing qui travaille sur l'alphabet d'entrée $\{a, b, c, d\}$ et qui écrit dans un autre fichier le code d'une machine équivalente sur l'alphabet $\{0, 1\}$ avec le codage $code(a) = 00$, $code(b) = 01$, $code(c) = 10$, $code(d) = 11$.

Question 8 : Proposer deux machines de Turing qui font usage pour l'une de la bande bi-infinie et pour l'autre de l'alphabet $\{a, b, c, d\}$. Quand l'utilisateur fait *make test*, ces machines sont converties en machines de Turing simples par les fonctions précédentes et simulées sur une entrée.

3 Optimisation de machine de Turing

On veut optimiser le code des machines de Turing afin d'obtenir des machines plus petites et plus rapides . On propose deux types d'optimisation, mais vous pouvez aussi proposer les votres.

Question 9 : Écrire une fonction qui prend en entrée une machine de Turing et renvoie une machine équivalente dans laquelle des suites de transitions ont été remplacées par une seule transition. Par exemple, si on a les transitions $A, 0 \to B, 0, -$ et $B, 0 \to C, 1, >$, on peut remplacer $A, 0 \to B, 0, -$ par $A, 0 \to C, 1, >$.

Question 10 : (Élimination du code mort) Écrire une fonction qui prend en entrée une machine de Turing et renvoie une machine équivalente dans lequel des transitions qui ne sont jamais utilisées lors de l'exécution ont été éliminées.