Problème du castor affairé

Maxime Rochkoulets

10 avril 2023

Table des matières

1 Machines de Turing **2**
1.1 Généralités 2
1.2 Castors affairés 3

2 Fonctions Σ et S **3**
2.1 Définitions 3
2.2 Non-calculabilité 3
2.3 Preuve 4
2.4 Compléxité algorithmique 4

3 Implémentation **5**

1 Machines de Turing

1.1 Généralités

Ce problème implique une certaine classe de machines de Turing, nous allons donc définir cet objet mathématique.

Une machine de Turing (théorisée par le mathématicien anglais Alan Turing en 1936[2]) est un système de règles, d'états et de transitions.

On peut la voir comme une machine opérant sur un ruban infini divisé en cellules. Chacune pouvant représenter un symbole, parmi un ensemble fini non vide de symboles noté Σ (souvent, $\Sigma = \{0, 1, B\},\ B =$ Vide), qu'on appelle aussi son *alphabet*.

Une machine peut avoir plusieurs états différents, parmi un ensemble fini non vide d'états noté Q, avec un état initial noté $q0$.

On peut décrire le fonctionnement d'une machine de Turing comme une suite d'étapes :

1. `La machine lit la cellule sur laquelle elle se trouve.`
2. `Écrit un symbole dans la cellule.`
3. `Effectue une des trois actions suivantes :`
   - `Arrêter le calcul.`
   - `Se décaler d'une cellule vers la gauche.`
   - `Se décaler d'une cellule vers la droite.`
4. `Change (ou pas) d'état.`

Le comportement d'une machine peut être représenté sous forme d'un tableau ayant $|\Sigma \times Q|$ lignes, décrivant les actions à effectuer en fonction de tous les cas possibles.

État Actuel	Lecture	Écriture	Direction	État Prochain
$q1$	1	0	G	$q0$
$q0$	0	1	D	$q1$
$q1$	B	0	G	$q0$
$q0$	1	B	G	$q0$
$q1$	0	1	D	$q0$
$q0$	B	0	D	$q1$

TABLE 1 – Tableau d'une MT avec $\Sigma = \{0, 1, B\}$ et $Q = \{q0, q1\}$.

Malgrès (ou grâce à) son apparente simplicité, une machine de Turing peut réaliser n'importe quel calcul ou algorithme (calculable), c'est le niveau d'automates le plus puissant dans la hiérarchie de Chomsky.

Aussi, une machine peut ne pas se terminer (une qui n'écrirait que des "1" à l'infini en allant vers la gauche par exemple), on dit alors qu'elle **diverge**, ou, à l'opposé, qu'elle **termine**.

Il existe d'autres catégories de machines de Turing mais nous avons ici décrit la version la plus simple et générale, qui est aussi celle utilisée dans le problème du castor affairé.

1.2 Castors affairés

Un **n-ième castor affairé** est une machine de Turing terminante qui, pour un n donné, produit le plus de "1" sur le ruban parmi toutes les autres machines à n-états à la fin de son exécution. Ici $\Sigma = \{0, 1\}$, et le ruban est initialisé avec des "0" (*le problème est à la base joué avec cet alphabet, mais il peut être étendu*).

Le "**problème du castor affairé**", ou encore "**jeu du castor affairé**" consiste donc à trouver une des machines de Turing gagnantes (il y'a en a au moins 4) pour tout n.

Parfois, un castor affairé est aussi défini comme la machine à n états qui effectue un maximum de déplacements (ou d'étapes) avant de terminer, cette définition correspond à la fonction S donnée ci-dessous.

Ce concept fut introduit par Tibor Radó en 1962, dans son article "On Non-Computable functions"[1], le nom venant de l'expression "busy beaver" en anglais.

2 Fonctions Σ et S

2.1 Définitions

Tibor Radó définit $\Sigma : \mathbb{N} \to \mathbb{N}$ la fonction qui associe à un nombre n le nombre de "1" sur le ruban à la fin de l'exécution d'un n-ième castor affairé.

En addition à la fonction Σ, il définit $S : \mathbb{N} \to \mathbb{N}$ comme :
— $s(M)$: le nombre de déplacements de M pour tout $M \in E_n$, avec E_n l'ensemble des MTs de taille n.
— $S(n) : max\{s(M) : M \in E\}$

2.2 Non-calculabilité

Ces deux fonctions (que l'on va montrer non calculables) font partie des fonctions ayant les croissances les plus rapides en mathématiques, plus encore que la fonction non recursive primitive d'Ackermann par exemple ($A(n) <$

$\Sigma(n) < S(n)$ quand $n \to \infty$). En fait, il a été prouvé que ces fonctions grandissent plus rapidement que n'importe quelle fonction $f(n)$ calculable, pour n assez grand.

Leur non-calculabilité vient du fait qu'à partir d'un certain n, il est impossible de vérifier si une machine de Turing donnée est en train de boucler temporairement avant de continuer, ou si elle bouclera à l'infini (nos ordinateurs peuvent seulement le faire quand n est assez petit). Résoudre ce problème consiste à résoudre le problème de l'arrêt, qui fut montré indécidable par Alan Turing[2].

2.3 Preuve

On procède par contradiction. Supposons qu'un tel programme H existe, capable de dire si un programme (en l'occurence, une machine de Turing) P avec une entrée E bouclera indéfiniment ou non. $H(P, E)$ retourne *"vrai"* si $P(E)$ ne bouclera pas indéfiniment, *"faux"* sinon.

Maintenant, créons un programme T prenant en entrée un autre programme P et étant définit comme suit :

— `Si H(P, P) = "vrai", boucler indéfiniment.`
— `Sinon, arrêter l'exécution.`

Exécutons $T(T)$ et observons les 2 cas possibles :

1. Si $H(T, T)$ retourne "vrai", alors H a tord, car $T(T)$ boucle indéfiniment.
2. Sinon, H a encore tord, car $T(T)$ s'arrête.

On arrive à une contradiction dans les deux cas, prouvant qu'un tel programme H est impossible. Et ainsi, puisqu'on ne peut construire un programme capable de nous dire si l'exécution d'une certaine machine de Turing bouclera à l'infini ou non, on ne peut construire un programme résolvant le problème du castor affairé.

2.4 Compléxité algorithmique

L'algorithme, pour trouver $\Sigma(2)$ par exemple, consiste à tester toutes les MTs à 2 symboles et 2 états, en **essayant** de rejeter les machines divergentes.

Le nombre de machines de Turing à n-états, s symboles et d directions nous est donné par

$$((n+1) \times s \times d)^{n \times s}$$

Pour se rendre compte de la compléxité temporelle de l'algorithme calculant $\Sigma(n)$ (*en supposant que toutes les machines de Turing soient terminantes*), définissons $f(n) : \mathbb{N} \to \mathbb{N}$ comme étant la fonction qui donne le nombre de machines à

tester pour trouver le n-ième castor affairé. Dans ce problème, la formule donnée ci-dessus est simplifiable par

$$f(n) = (4n + 4)^{2n}$$

On a donc $f(1) = 64$ *machines à tester,* $f(2) = 20736$, $f(3) = 16^8$, $f(4) = 20^8$...

De plus, le temps requis pour tester chaque machine de Turing croît exponentiellement par rapport à n, on comprend aisément pourquoi si peu de valeurs de Σ et S ont été trouvées à ce jour .

3 Implémentation

On peut tester simplement le problème du castor affairé sur des petites valeurs de n en implémentant une machine de Turing à 2 états, on définira une constante à partir de laquelle on considérera que la machine est en train de boucler indéfiniment.

La fonction qui associe les valeurs d'entrées de la machine à son comportement peut être vue comme une table de hachage associant une paire `(symbole, état)` au triplet `(symbole, direction, état)`, et le ruban comme un tableau de taille constante.

Vous pouvez retrouver une machine de Turing à 2 états qui teste différents castors affairés écrite en Go sur mon Github. Une version en C du professeur D. Brailsford est trouvable ici, en complément à une vidéo qui parle de ce problème.

Références

[1] Tibor Rado. On non-computable functions. *Bell System Technical Journal*, 41 :877–884, 1962.

[2] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42) :230–265, 1936.