Problème du Castor Affairé

Maxime Rochkoulets

5 avril 2023

Résumé

Petit article où j'explique ce que sont la machine de Turing et les fonctions non calculables de Tibo Radó.

Table des matières

1 Machines de Turing

1.1 Généralités

Le jeu est conçu pour être joué par une certaine classe de machines de Turing, nous allons donc définir cet objet mathématique.

Une machine de Turing (inventée par le mathématicien anglais Alan Turing en 1936) 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 aussi 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 suit :

1. `La machine lit la cellule sur laquelle elle se trouve.`
2. `En fonction de sa position, de son état et du résultat de sa lecture, 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. `Enfin, la machine change (ou pas) d'état.`

Les instructions peuvent être représentées sous forme d'un tableau ayant $\#(\Sigma \times Q)$ lignes, décrivant les actions à effectuer en fonction de tous les cas possibles.

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

TABLE 1 – Exemple de tableau des instructions d'une MT à 3 symboles et 2 états.

Malgrès son apparente simplicité, il a été prouvé qu'une machine de Turing (universelle) pouvait réaliser n'importe quel calcul ou algorithme, c'est le niveau

d'automates le plus puissant dans la hiérarchie de Chomsky.

Aussi, une machine peut ne pas se terminer (une machine 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

Le **n-ième castor affairé** est une machine de Turing terminante qui, pour un n donné, produit le maximum de "1" sur le ruban parmi toutes les autres machines à n-états à la fin de son éxécution.

Ici, $\Sigma = \{0, 1\}$, et le ruban est initialisé avec des "0". Le "problème du castor affairé", ou encore "jeu du castor affairé" consiste donc à trouver la machine de Turing gagnante pour tout n. *Dans certains articles, on dit que la machine de Turing gagnante est celle par la fonction S décrite ci-dessous.*

Ce concept fut introduit par Tibor Radó en 1962, dans son article *On Non-Computable functions*, 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'éxécution du 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 MT de taille n.
- $S(n)$: $max\{s(M) : M \in E\}$, le nombre maximal de déplacements (ou étapes) effectuées parmi toutes les MT de taille n avant de terminer.

2.2 Non-calculabilité

Ces deux fonctions sont non-calculables.

Cela vient du fait qu'à partir d'un n assez grand, on ne sait pas si une machine de Turing donnée est en train de boucler temporairement avant de continuer, ou si elle bouclera indéfiniment. Résoudre ce problème consiste à résoudre le problème de l'arrêt, qui fut montré indécidable par Alan Turing.

Elles font partie des fonctions ayant les croissances les plus rapides en mathématiques, plus rapides que la fonction non recursive primitive d'Ackermann ($A(m,n) < \Sigma(m,n) < S(m,n)$). En fait, il a été prouvé ces fonctions grandissent plus rapidement que n'importe quelle fonction $f(n)$ calculable, pour n assez grand.

2.3 Compléxité algorithmique

L'algorithme, pour trouver $\Sigma(2)$ par exemple, consiste à tester toutes les MTs à 2 symboles et 2 états, en définissant une valeur arbitraire après laquelle stopper leur éxécution si elles bouclent trop longtemps.

Le nombre de machines de Turing de taille n à 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 MTs soient terminantes*), définissons $f(n) : \mathbb{N} \to \mathbb{N}$ comme étant la fonction qui donne le nombre de MTs à tester pour trouver le n-ième castor affairé. La formule donnée ci-dessus est simplifiable par $(4n+4)^{2n}$. On a donc, $f(1) = 64$ machines à tester, $f(2) = 20736$, $f(3) = 16^8$...

En supposant que toutes les MTs soient terminantes, on a une compléxité de l'ordre de $O(n^n)$.

3 Implémentation

On peut tester le jeu du Castor Affairé sur des petites valeurs de n en implémentant une machine de Turing, si n est assez petit, on définira une valeur constante à partir de laquelle on considérera que la machine est en train de boucler indéfiniment.

La fonction qui associe au tuple $(etat, symbole)$ les actions que la machine doit effectuer peut être vue comme une table de hachage.

J'ai codé une simple machine de Turing à 2 états pour tester différents castors affairés, le code est sur mon Github.

4 Valeurs connues