Calculabilité - Indécidabilité

1 Introduction

Alan Turing est un mathématicien et informaticien anglais du 20 ième siècle (1912-1954). Il est considéré comme l'inventeur de l'informatique moderne en créant la machine qui porte son nom. La machine de Turing est une machine virtuelle qui effectue des calculs d'une manière mécanique sans intervention de l'homme à part pour l'entrée des données et la lecture du résultat.

2 Description

Une machine de Turing comprend:

- un ruban **infini** divisé en plusieurs cases;
- une tête de lecture et d'écriture pour lire et écrire sur le ruban;
- une table de transition qui contient les instructions du programme à réaliser.

La table de transition est divisée en plusieurs états. À chaque état, des informations de lecture et d'écriture sont données selon la valeur lue sur le ruban. Pour chaque information, un nouvel état est indiqué.

2.1 Représentation d'une machine de Turing:

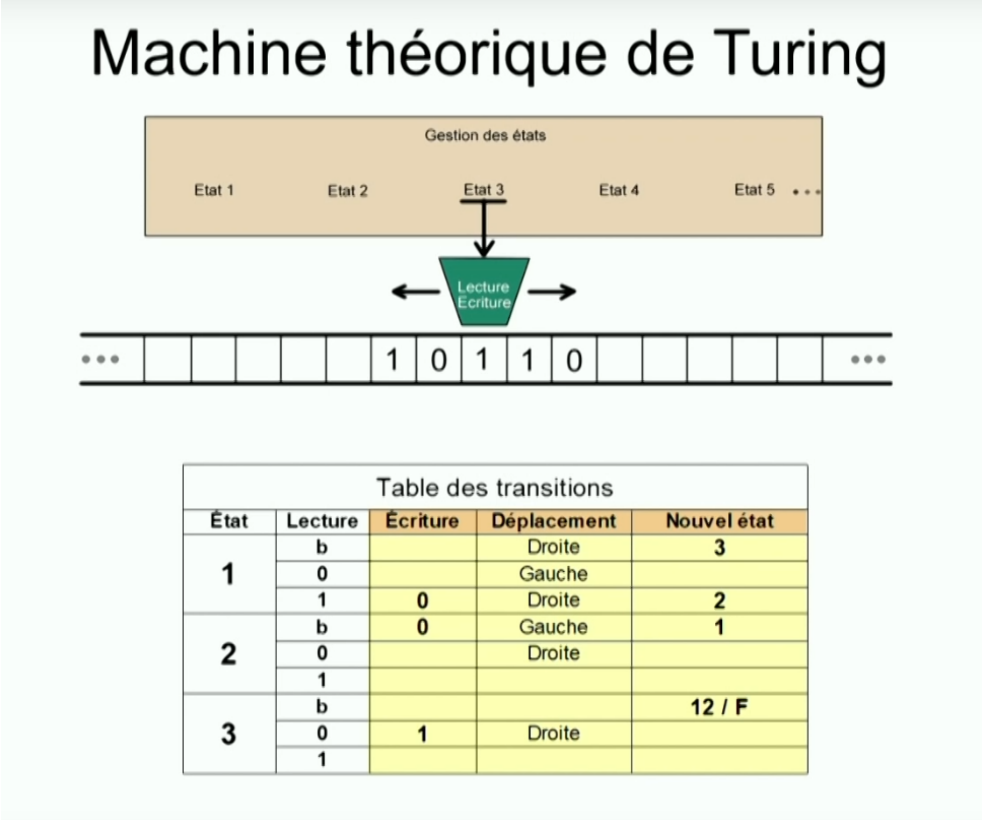


Table des transitions

État	Lecture	Écriture	Déplacement	Nouvel état
1	b		Droite	3
	0		Gauche	
	1	0	Droite	2
2	b	0	Gauche	1
	0		Droite	
	1			
3	b			12 / F
	0	1	Droite	
	1			

CC - BY - $

2.2 Principe

Les cases du ruban sont blanches. L'écriture ajoute des symboles issus d'un alphabet défini à l'avance. Pour le binaire, l'alphabet est donc constitué des symboles 0, 1 et blanc.

La tête de lecture et écriture peut se déplacer à droite ou à gauche. Elle peut lire ou écrire les symboles de l'alphabet sur le ruban.

Les différents états de la table de transition indiquent si la tête de lecture / écriture doit lire ou écrire un symbole et se déplacer à droite ou à gauche.

A l'issue du traitement, un nouvel état (ou la répétition du même état) est indiqué.

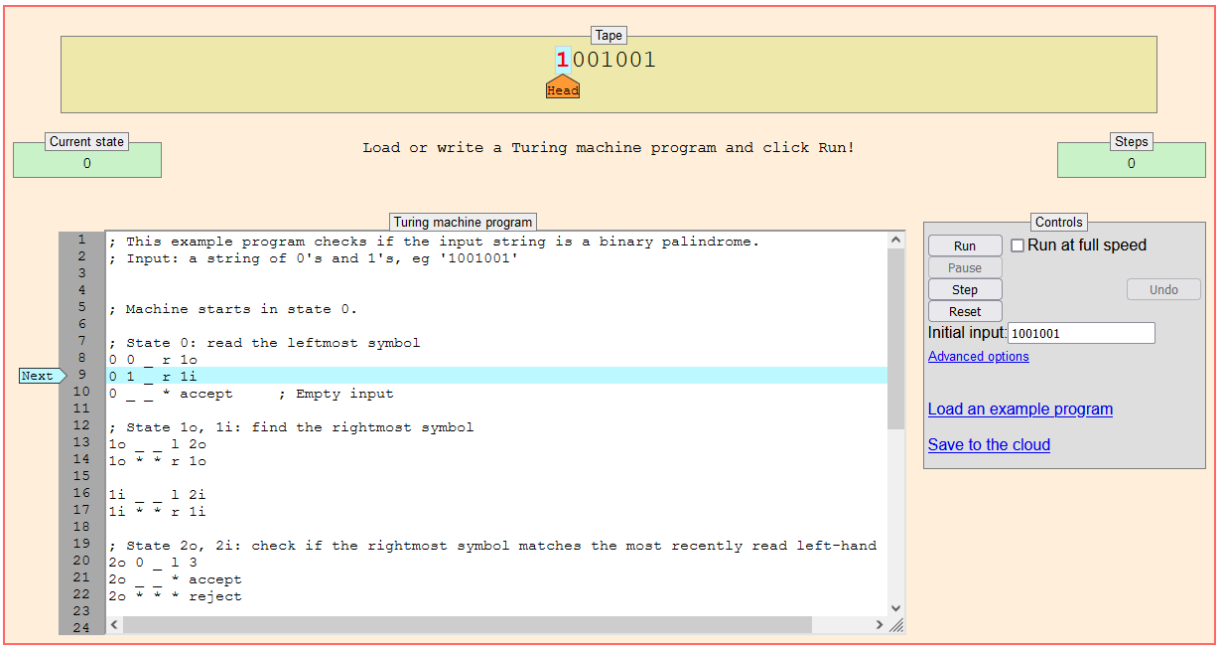
Un état final met fin à l'exécution du programme.

Exercice

1. L'exemple de la machine de Turing ci-dessus est appliqué au nombre binaire 111 avec la tête de lecture située sous le chiffre 1 situé le plus à droite. Quel est le résultat écrit sur le ruban après exécution de la machine de Turing ?
2. Écrire la table de transition d'une machine de Turing qui ajoute 1 à un nombre binaire inscrit sur le ruban.

Simuler une machine de Turing

Il existe des concepteurs de machines de Turing. Il existe également des programmes qui permettent d'appliquer une machine de Turing à un ruban virtuel. Ci-dessous, un programme **javascript**, qui simule une machine de Turing.



Vous pouvez compléter la table de transition pour vérifier votre code. **Attention**, la tête de lecture est située sous le chiffre de gauche et peut nécessiter un état supplémentaire.

Exercice

1. Écrire la table de transition d'une machine de Turing qui renvoie le complément à 1 d'un nombre binaire.

CC - BY - $

2. Appliquer votre table sur l'émulateur.

Exercice

Déduire des deux exercices précédents la valeur binaire d'un nombre entier signé.

3 Machine universelle

Turing a montré qu'il existe une machine **universelle** qui peut simuler n'importe quelle autre machine de Turing et donc exécuter n'importe quel programme. Le programme d'une machine de Turing est une donnée de la machine universelle.

Un programme peut donc être une donnée d'un autre programme. Nous avons différents exemples de situations ou des programmes sont des données d'un autre programme:

1. Un système d'exploitation est un ensemble de programmes capbles d'exécuter différents programmes. Cela signifie qu'il sont les données des programmes du système d'exploitation.
2. Les logiciels sont pour la plupart des programmes compilés, par souci d'optimisation. Cela signifie qu'ils sont transformés en fichiers binaires directement lisibles par le processeur et donc rapide à l'exécution. La compilation est un programme qui prend en donnée un programme écrit dans un certain langage et transformé en fichier binaire exécutable. Par exemple, sur le système Linux, on a le compilateur **gcc**
3. Python est un langage **interprété**. L'interpréteur de python prend en donnée un programme et le rend exécutable par l'ordinateur.

L'idée sous-jacente à la machine universelle est l'inscription du programme à réaliser sur le ruban, c'est à dire en tant que donnée de la machine de Turing. Aujourd'hui, de nombreux logiciels utilisent des programmes en tant que donnée et les exécutent.

- L'interpréteur Python prend le programme Python en tant que donnée et le transforme en programme exécutable par la machine. Cet interpréteur peut être écrit dans un autre langage de programmation comme le langage C.
- La compilation est un procédé qui consiste à transformer les programmes écrits avec un langage en binaire directement lisible par le processeur. Cela est un gain de temps par rapport à un iterpreteur qui réalise cette tache à chaque lancement du programme. Sur linux, le compilateur **gcc** transforme les programmes **sources** (écrit en langage C) en binaire.
- Le langage Python possède deux instructions, **eval** et **exec**, qui prennent en entrée des chaines de caractères et cherchent à les exécuter.

Exercice

Ouvrir l'éditeur Python Thonny et créer un fichier python nommé *donnée-programme.py*.

Les questions sont liées par 2 ! 1. Écrire l'instruction python qui affiche le message : machine de Turing. 2. Transformer cette instruction en chaine de caractère puis procéder à son exécution avec l'instruction eval. 3. Écrire un programme qui vérifie la parité d'un nombre entier saisi par l'utilisateur sous forme d'une chaîne de caractères. 4. Procéder à son exécution dans l'interpréteur python. 5. Écrire la création d'une liste de nombres entiers positifs multiples de 3 compris entre 1 et 20 sous forme d'une chaine de caractères. 6. Procéder à son exécution et contrôler le résultat.

CC - BY - $

4 Calculabilité - Décidabilité

Définition

Un nombre, une fonction ou un problème sont **calculables** s'il existe un algorithme et donc une machine de Turing capable de renvoyer un résultat numérique.

Un problème est **décidable** si et seulement si, il existe un algorithme qui résout le problème. Alan Turing a montré en 1936, qu'il y a des problèmes indécidables. Pour le montrer, il a utilisé le **problème de l'arrêt**.

Preuve de l'indécidabilité

On considère que les algorithmes sont de 2 natures. Ceux qui s'arrêtent et ceux qui ne s'arrêtent pas. La preuve repose sur un raisonnement par l'absurde qui est le suivant:

- On suppose qu'il existe un algorithme **P** qui résout le problème de l'arrêt.
  Pour tout algorithme **A**, l'algorithme **P** renvoie vrai si l'algorithme **A** s'arrête et faux dans le cas contraire. On peut déjà affirmer que notre algorithme **P** s'arrête.
- On construit un algorithme **Q** tel que:
  - si un algorithme **A** s'arrête donc **P(A)** est vrai, alors **Q(A)** ne s'arrête pas;
  - si un algorithme **A** ne s'arrête pas donc **P(A)** est faux, alors **Q(A)** s'arrête.
    L'existence de l'algorithme **Q** dépend de l'existence de l'algorithme **P**.
- Appliquons notre algorithme **Q** à lui même soit évaluer **Q(Q)** :
  - si **Q** s'arrête, alors **Q(Q)** ne s'arrête pas et donc **Q** est un algorithme qui ne s'arrête pas !
  - si **Q** ne s'arrête pas, alors **Q(Q)** s'arrête et donc **Q** est un algorithme qui s'arrête !
    On a une contradiction, donc l'algorithme **Q** n'existe pas et alors **P** n'existe pas non plus.

Le problème de l'arrêt ne peux pas se résoudre par un algorithme donc c'est un problème **indécidable**. Il existe d'autres problèmes indécidables et fonctions non calculables. Cela prouve qu'un ordinateur ne peut pas tout calculer ou tout décider.

5 Sitographie

Les liens des sites qui ont documenté cette page et pour approfonfir la notion. - Un prototype programmable pour concrétiser la machine de Turing - Comment fonctionne une machine de Turing ? - Du carreau de Truchet au carreau de Wang : atteindre l'atome de l'apériodique et du calculable

CC - BY - $