Machine de Turing

Systèmes Formels
Master 1

Sébastien Verel
verel@lisic.univ-littoral.fr
http://www-lisic.univ-littoral.fr/~verel

Université du Littoral Côte d'Opale
Laboratoire LISIC
Equipe OSMOSE

Objectifs de la séance 06

- Connaitre la définition d'une machine de Turing
- Savoir exécuter une machine de Turing
- Savoir définir le langage reconnu par une machine de Turing
- Savoir définir la fonction calculée par une machine de Turing
- Savoir définir une machine de Turing pour reconnaître un langage ou calculer une fonction
- Connaitre la thèse de Church-Turing
- Savoir que le problème de l'arrêt est non calculable

Questions principales du jour :

Qu'est-ce qu'une procédure effective ?

Plan

1. Introduction
2. Machine de Turing
3. Fonction calculée, Indécidabilité

Introduction

D'après P. Dehornoy, université de Caen

- Automate :

  modèle abstrayant la notion de calcul sans écriture

  L est décidable par automate si pour tout mot w de L, on peut répondre à la question "w appartient-il à L ?" en **lisant** le mot et en utilisant la **mémoire finie**.

- Machine de Turing :

  modèle analogue avec une notion plus élaborée de calcul

  L est décidable par automate si pour tout mot w de L, on peut répondre à la question "w appartient-il à L ?" en lisant le mot et en utilisant la mémoire finie mais aussi en **écrivant** des informations sur un **support illimité**

Version formalisée du calcul mental

Version formalisée du calcul au sens général

La machine d'A. Turing

Motivation

- Machine abstraite crée par Alan Turing, 1936
  Alan M. Turing, "On computable numbers with an application to the Entsheidungsproblem", Proc. London Math. Society, 2, 42, pp. 230-265, 1936.
- Date du premier ordinateur ?
- Précurseur théorique des machines
- Participe à la construction de système de décryptage pendant la seconde guerre mondiale reposant sur ces principes
- Modèle de calcul parmi les plus commodes
- Tous les ordinateurs sont équivalents à cette "petite" machine

La machine d'A. Turing

Motivation

- Machine abstraite crée par Alan Turing, 1936
  Alan M. Turing, "On computable numbers with an application to the Entsheidungsproblem", Proc. London Math. Society, 2, 42, pp. 230-265, 1936.
- Date du premier ordinateur ? 1941 le Z3, 1944 le ASCC/Mark1
- Précurseur théorique des machines
- Participe à la construction de système de décryptage pendant la seconde guerre mondiale reposant sur ces principes
- Modèle de calcul parmi les plus commodes
- Tous les ordinateurs sont équivalents à cette "petite" machine

Introduction par les langages

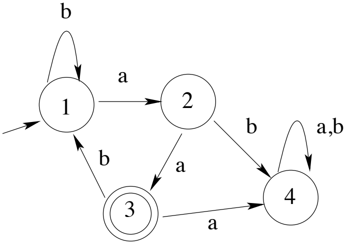
	Langages	Grammaires	Procédure effective
3	Rationnels ou réguliers	régulières à droite $A \to a$, $A \to aB$, $A \to \epsilon$ $A, B \in N$ $a \in T$ (régulières à gauche)	Automates finis
2	algébriques ou non-contextuels	algébriques, non-contextuelles $A \to \alpha$ $A \in N$ $\alpha \in (N \cup T)^*$	Automates à pile
1	contextuels	contextuelles, monotones $\alpha \to \beta$ ou $A \to \epsilon$ $\alpha, \beta \in (N \cup T)^*$, A axiome $\|\alpha\| \leq \|\beta\|$	Machine de Turing à l'espace linéairement borné
0	récursivement énumérables	contextuelles avec effacement $\alpha \to \beta$ $\alpha \in (N \cup T)^+$ $\beta \in (N \cup T)^*$ aucune contrainte	Machine de Turing

Caractéristiques d'un automate fini

Automate fini

- Etats : mémoire finie,
- Lecture des symboles,
- Programme : fonction de transition d'états

états	a	b
$\to$ 1	2	1
2	3	4
$\boxed{3}$	4	1
4	4	4



a	a	b	a

$\wedge$

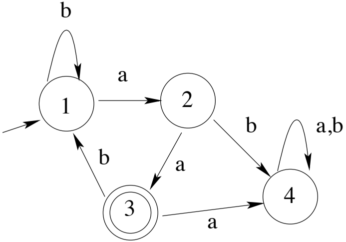
1

Caractéristiques d'un automate fini

Automate fini

- Etats : mémoire finie,
- Lecture des symboles,
- Programme : fonction de transition d'états

états	a	b
$\rightarrow$ 1	2	1
2	3	4
$\boxed{3}$	4	1
4	4	4



a	a	b	a

$\wedge$

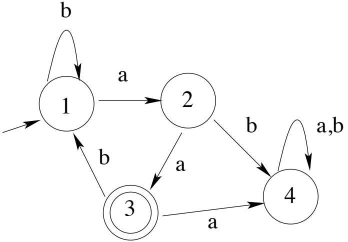
2

Caractéristiques d'un automate fini

Automate fini

- Etats : mémoire finie,
- Lecture des symboles,
- Programme : fonction de transition d'états

états	a	b
$\rightarrow$ 1	2	1
2	3	4
$\boxed{3}$	4	1
4	4	4



a	a	b	a

$\wedge$

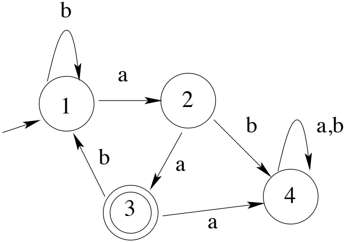
3

Caractéristiques d'un automate fini

Automate fini

- Etats : mémoire finie,
- Lecture des symboles,
- Programme : fonction de transition d'états

états	a	b
$\rightarrow$ 1	2	1
2	3	4
$\boxed{3}$	4	1
4	4	4



a	a	b	a

$\wedge$

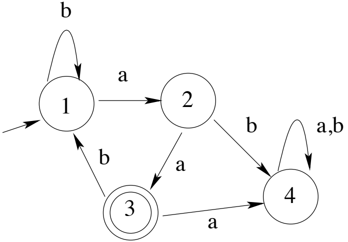
1

Caractéristiques d'un automate fini

Automate fini

- Etats : mémoire finie,
- Lecture des symboles,
- Programme : fonction de transition d'états

états	a	b
$\rightarrow$ 1	2	1
2	3	4
$\boxed{3}$	4	1
4	4	4



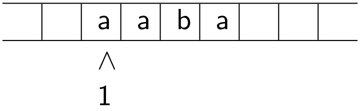
a	a	b	a

$\wedge$

2

Caractéristiques d'une machine de Turing

Support illimité de l'information : Ruban



Machine de Turing

- Etats : mémoire finie,
- Lecture des symboles du ruban,
- **Ecriture** sur le ruban
- Programme :
  fonction de transition d'états et de déplacement et d'écriture

Premier exemple
Fonction de transition

Ancien état	Symbole lu	Symbole écrit	Mouv.	Nouvel état
	$\square$	$\square$	$\rightarrow$	arrêt
1	a	b	$\rightarrow$	1
	b	a	$\rightarrow$	1

		a	a	b	a			
		$\wedge$						
		1						

Premier exemple

Fonction de transition

Ancien état	Symbole lu	Symbole écrit	Mouv.	Nouvel état
1	$\square$	$\square$	$\rightarrow$	arrêt
	a	b	$\rightarrow$	1
	b	a	$\rightarrow$	1

		a	a	b	a			
		$\wedge$						
		1						

		b	a	b	a			
			$\wedge$					
			1					

Transition : Tableau à double entrée

	a	b	$\square$
$\rightarrow$1	b , $\rightarrow$, 1	a , $\rightarrow$, 1	$\square$, $\rightarrow$, arrêt

		a	a	b	a			

$\wedge$

1

Exo

Exécutez la machine de Turing ci-dessus et décrivez sa fonction de calcul.

Exemple plus sophistiqué

	a	b	$\square$
$\to 0$	a , $\leftarrow$, 0	b , $\leftarrow$, 0	$\square$, $\to$, 1
1	b , $\to$, 1	a , $\to$, 1	$\square$, $\to$, arrêt

		a	a	b	a	b	a	

Langage décidé par une machine de Turing

	a	b	□
→0	□ , →, 1	refusé	accepté
1	a , →, 1	b , →, 1	□, ←, 2
2	refusé	□ , ←, 3	
3	a , ←, 3	b , ←, 3	□, →, 0

	a	a	a	b	b	b		

	a	a	a	b	a	b		

Exo

Exécutez la machine de Turing sur les mots ci-dessus et décrivez le langage reconnu.

Définition formelle

Machine de Turing (MT)

Une machined e Turing à un ruban infini est septuplet $(Q, \Gamma, \Sigma, \delta, q_0, B, F)$ où :

- Q ensemble fini d'état,
- Γ alphabet fini des symboles du ruban,
- $\Sigma \subset \Gamma$ alphabet fini des symboles d'entrée,
- $B \in \Gamma \setminus \Sigma$ symbole particulier dit "blanc"
- q_0 état initial
- F ensemble des états acceptants
- δ relation de transition

La MT est déterministe si pour chaque chaque configuration, elle a au plus une possibilité d'évolution.

Relation de transition

Relation de transition

$$\delta \subset Q \times \Gamma \times Q \times \Gamma \times \{\leftarrow, \rightarrow\}$$

Notation d'une règle :

$$q, \sigma \rightarrow q', \sigma', m$$

Prédécesseur :

- q : état courant de la machine
- σ symbole lu sur le ruban

Successeur :

- q' : nouvel état de la machine
- σ' symbole à écrire sur le ruban
- m déplacement de la tête de lecture

Relation de transition : sous forme de table ou de diagramme

Notion de configuration

La configuration d'une MT décrit l'"état général" de la machine : état du ruban, état courant de la machine et position de la tête de lecture.

$$(f, q, p)$$

- $f : \mathbb{N} \to \Gamma$ le ruban
- $q \in Q$ l'état de la machine
- $p \in \mathbb{N}$ la position sur le ruban

La relation de transition permet alors de calculer chaque élément de la nouvelle configuration.

Langage reconnu

Le langage accepté par $M = (Q, \Gamma, \Sigma, \delta, q_0, B, F)$ est défini par :
$L(M) = \{w \in \Sigma^*$ tels que :

- l'état initial de M est q_0
- le mot w est écrit sur le ruban
- la tête de lecture est positionnée sur la première lettre de w
- M atteint un état acceptant de F en un nombre fini d'étape

$\}$

cf. exercice 1 fiche 06

Classe de langages

Une MT s'arrête lorsque

- elle atteint un état final
- elle ne peut plus effectuer de transition

Langage récursif

Un langage reconnu par une MT qui s'arrête sur tous les mots en entrée est dit **langage récursif**

Langage récursivement énumérable

Un langage reconnu par une MT qui s'arrête sur tous les mots du langage (et peut ne pas s'arrêter sur les autres) est dit **langage récursivement énumérable** – engendré par une grammaire de type 0.

Fonction calculée par une machine de Turing

- Entrée d'une MT :
  mot inscrit sur le ruban initialement.
- Sortie d'une MT :
  mot inscrit sur le ruban lorsque la MT s'arrête.

Fonction calculée

La **fonction calculée** f par une MT M est définie par :

A toute entrée x sur laquelle M s'arrête, on associe la sortie y :

$$f(x) = y$$

Aucune image n'est associée au mot x sur lequel M ne s'arrête pas.

Machines de Turing équivalentes

On peut imaginer beaucoup de variantes de MT :

- sur un "demi" ruban
- sur deux ou plusieurs rubans
- la tête de lecture peut être stationnaire
- non-déterminisme
- écrire ou non de symbole blanc
- ...

Et pourtant, elles sont toutes équivalentes (reconnaissance du même langage ou fonction calculée identique)

La machine de Turing semble bien représenter une notion de "calcul" par une "procédure effective".

Machine de Turing universelle

Machine de Turing universelle

Une machine de Turing universelle est capable de simuler le comportement de n'importe quelle autre machine de Turing.

Existence

Par exemple, en utilisant 2 rubans :

- sur un ruban le programme de la machine de Turing originale
- sur l'autre ruban le calcul de cette machine

Fonctions calculables

Thèse de Church-Turing

Les fonctions calculables par une procédure effective le sont par une machine de Turing.

- Modélisation de la notion de calcul et procédure effective
- Ce n'est pas un résultat que l'on peut démontrer
- Fonctions calculables par MT = fonctions définies par λ-calcul de Chruch
- Base de la théorie de la calculabilité
- Alonzo Church (1903 -1995), mathématicien, logicien américain.

$\Rightarrow$ **Tous** les ordinateurs sont équivalents à une machine de Turing...

Fonctions non-calculables

Exercice

- L'ensemble des machines de Turing est-il dénombrable ?
- Existe-il un ensemble de fonctions non-dénombrables ?
- Existe-t-il des fonctions non calculables ?

Fonctions non-calculables

Exercice

- L'ensemble des machines de Turing est-il dénombrable ?
- Existe-il un ensemble de fonctions non-dénombrables ?
- Existe-t-il des fonctions non calculables ?

Le tout est de savoir lesquelles...

Propriété décidable et indécidable

Décidabilité /Indécidabilité

Intuitivement,

- propriété décidable :
  on peut savoir (démontrer)
  si pour tout x, $P(x)$ est vrai ou faux.
- propriété indécidable :
  on ne peut pas savoir (démontrer)
  si pour tout x, $P(x)$ est vrai ou faux.

Une phrase indécidable

Les phrases célèbres d'Alain

Alain dit : "Je mens."

Une phrase indécidable

Les phrases célèbres d'Alain

Alain dit : "Je mens."

De 2 choses l'une :

- Soit Alain dit vrai,
  et donc il ment et la phrase est donc fausse....
- Soit Alain dit faux,
  et donc il ne ment pas, et la phrase est vraie...

Une phrase indécidable

Les phrases célèbres d'Alain

Alain dit : "Je mens."

De 2 choses l'une :

- Soit Alain dit vrai,
  et donc il ment et la phrase est donc fausse....
- Soit Alain dit faux,
  et donc il ne ment pas, et la phrase est vraie...

A faire retourner tous les logiciens grecs dans leur tombe !

Exemple de phrase indécidable :
on ne peut pas démontrer que cette phrase est vraie ou fausse !

Décidabilité

Définition

Une famille dénombrable de propriétés $P(x)$ est **décidable** si sa fonction caractéristiques f_P est **calculable**.

$$f_P(x) = \begin{cases} 1 & \text{si } P(x) \text{ est vrai,} \\ 0 & \text{si } P(x) \text{ est faux.} \end{cases}$$

Problème de l'arrêt : fonction non calculable

Fonction qui associe l'arrêt d'une machine de Turing :

$$A(M,e) = \begin{cases} 1 & \text{si la machine de Turing } M \text{ s'arrête sur l'entrée } e \\ 0 & \text{sinon.} \end{cases}$$

Problème de l'arrêt

La fonction A, qui associe l'arrêt d'une machine de Turing, est non calculable.

Preuve : argument diagonal

- Supposons qu'il existe une MT MA qui calcule la fonction A
- soit la machine :

$$MD(e) = \begin{cases} \text{si } MA(e,e) = 1 & \text{alors boucle infinie} \\ \text{si } MA(e,e) = 0 & \text{alors terminer} \end{cases}$$

- Contradiction en analysant $MA(MD, MD)$

Conséquences

Impact pratique

- Pas de débugger parfait qui prédit l'arrêt ou non d'un programme !
- Ramasse-miette : libérer une zone mémoire lorsqu'elle n'est plus utilisée
- Détection de virus
- ...