Machine de Turing

Informatique Théorique 2
Licence 3 informatique

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

Université du Littoral Côte d'Opale
Laboratoire LISIC
Equipe OSMOSE

Objectifs de la séance 04

- 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

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 :

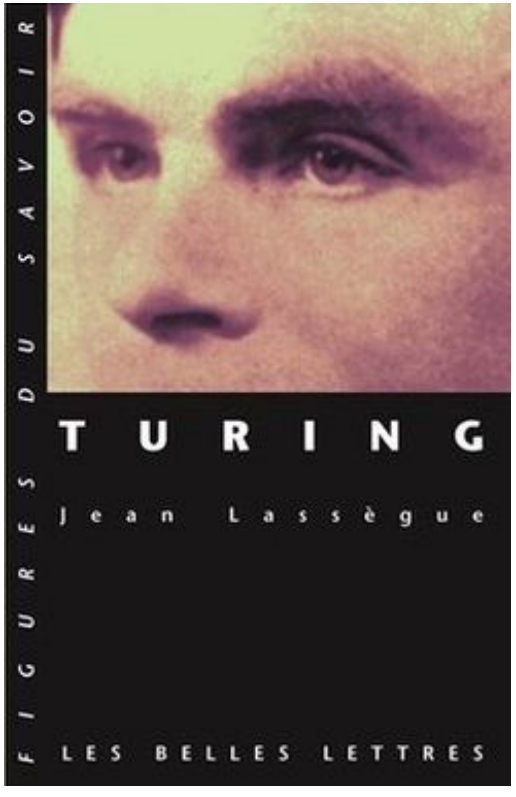
  modèle analogue avec une notion plus élaborée de calcul

  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

Bibliographie

Turing, Jean Lassègue, Les belles lettres, 2003.



Alan Turing

- 1912 : naissance à Londres, parents en Inde (1912-21)
- 1926 : entre à la Public School Sherborne
- 1928 : D. Hilbert renouvelle son Programme au Congrés de Bologne
- 1931 : entre au King's College de Cambridge pour suivre des études de mathématiques
- 1931 : K. Gödel : sur les propositions indécidables des Principia Mathematica
- 1933 : Hitler arrive au pouvoir, exil des intellectuels d'Allemagne et d'Europe centrale
- 1934 : Licence de mathématiques avec mention
- 1935 : devient Fellow de Kibg's College
- 1936 : Prouve le résultat négatif de la décidabilité proposé par Hilbert. Part travailler avec A. Church et J. Von Neumann
- 1939 : 4 septembre début de la guerre. Entre au service GCCS à Bletchley Park.
- 1943 : janv. - mars, laboratoire Bell sur des questions de cryptage de la parole ; renontre Shannon
- Commence à concevoir son projet de construire un cerveau

- 1939 : 4 septembre début de la guerre. Entre au service GCCS à Bletchley Park.
- 1943 : janv. - mars, laboratoire Bell sur des questions de cryptage de la parole ; renontre Shannon
- Commence à concevoir son projet de construire un cerveau. National Physical Laboratory pour construire un prototype d'ordinateur
- 1950 Publication de "computing machinery and intelligence"
- 1952 : Publication "La base chimique de la morphogénèse"
- 1954 : Suicide le 7 juin par ingestion d'une pomme au cyanure

Les intérêts scientifiques de M. Turing

- Mathématiques pures (calculs des probabilités et statistiques, théories des nombres, théories des groupes)
- Logique mathématique (décidabilité, calculabilité)
- Cryptologie
- Construction effective des premiers ordinateurs
- Morphogénèse

Articles majeurs

- 1936 (24 ans) : fonde la théorie de la calculabilité, Machine de Turing
- 1952 : Théorie générale sur les principes de la morphogénèse
- 1950 : Lien entre logique et biologie. Etude des processus cognitifs par simulation informatique

Articles majeurs

- 1936 (24 ans) : fonde la théorie de la calculabilité, Machine de Turing
- 1952 : Théorie générale sur les principes de la morphogénèse
- 1950 : Lien entre logique et biologie. Etude des processus cognitifs par simulation informatique

Turing se place du point de vue de **l'effectivité du calcul** : condition pratique de la réalisation de celui-ci

A. Turin, "On computable numbers, with an application to the Entscheidungsproblem", nov 1936.

Calculabilité

Intuition de calculable

"résultat d'une opération conduisant à la détermination exacte et achevée d'un nombre"

3 états de la calculabilité

1. Trouver en un nombre fini d'étapes un résultat exact et achevé
2. Trouver en un nombre fini d'étapes un résultat approché à n'importe quel degré d'approximation décidé à l'avance
3. "incalculable" : pas moyen de trouver une approximation en un nombre fini d'étape

Notion de fonction calculable

si sa valeur pour tout nombre calculable de l'ensemble de départ est un nombre calculable

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 $\lvert\alpha\rvert \leq \lvert\beta\rvert$	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

Automate fini

Fonction à valeur booléenne définie sur l'ensemble des mots

$$A : \Sigma^* \rightarrow \{0, 1\}$$

Répond à un problème de décision :

si $A(w) = 0$ alors $w \notin L_A$

si $A(w) = 1$ alors $w \in L_A$

Machine de Turing : point de vue langage, mémoire infinie

$$MT : \Sigma^* \rightarrow \{0, 1\}$$

Répond à un problème de décision :

si $MT(w) = 0$ alors $w \notin L_{MT}$

si $MT(w) = 1$ alors $w \in L_{MT}$

Mais :

en utilisant un support infini de mémorisation

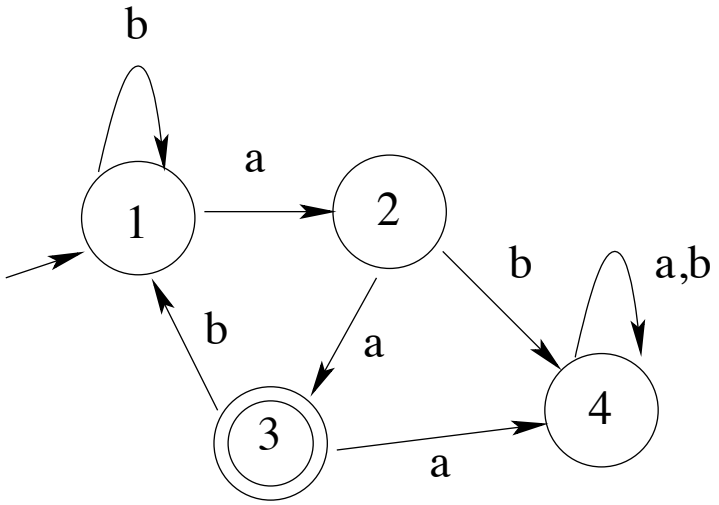
en plus d'un nombre fini d'état

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
3	4	1
4	4	4



a	a	b	a

$\wedge$

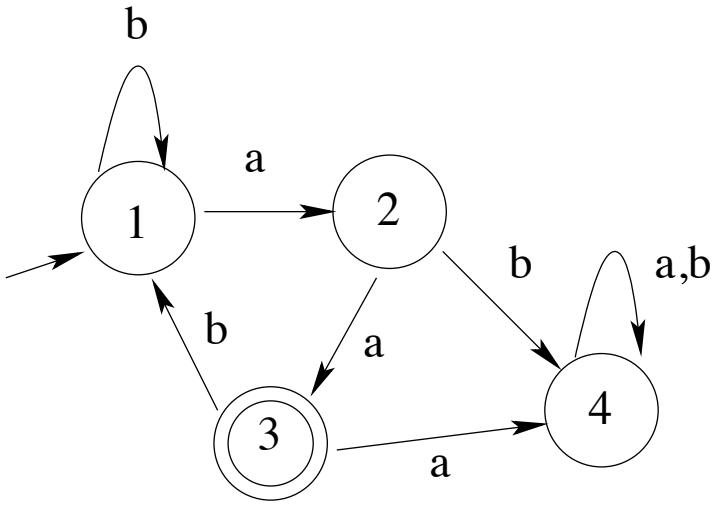
1

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
3	4	1
4	4	4



a	a	b	a

$\wedge$

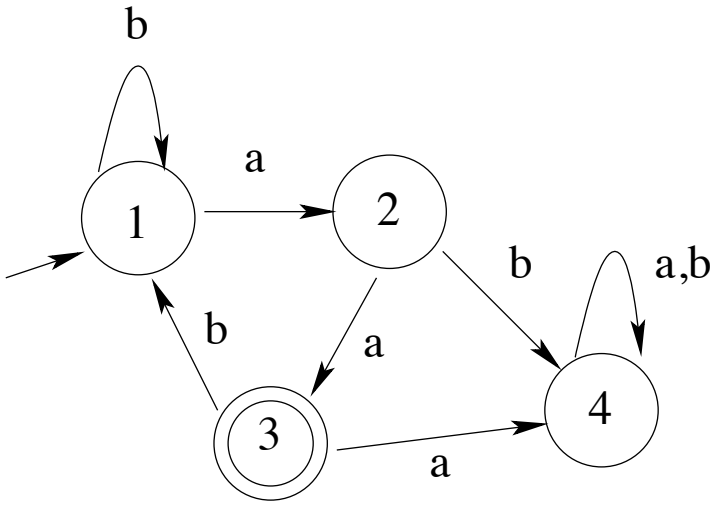
2

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
3	4	1
4	4	4



a	a	b	a

$\wedge$

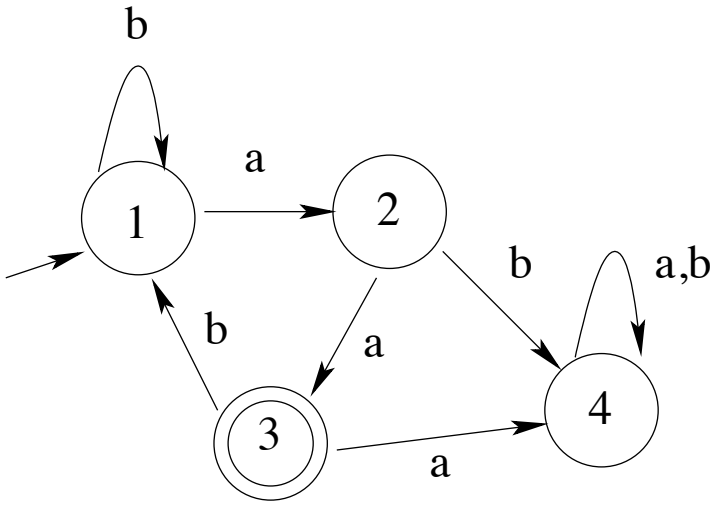
3

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
3	4	1
4	4	4



a	a	b	a

$\wedge$

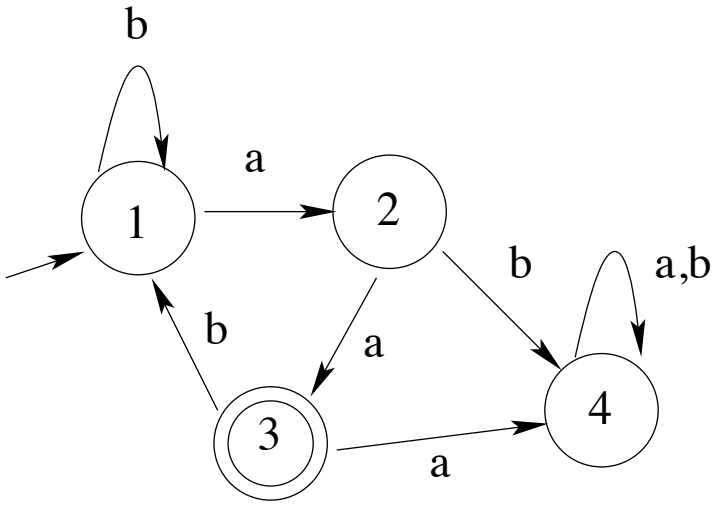
1

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
3	4	1
4	4	4



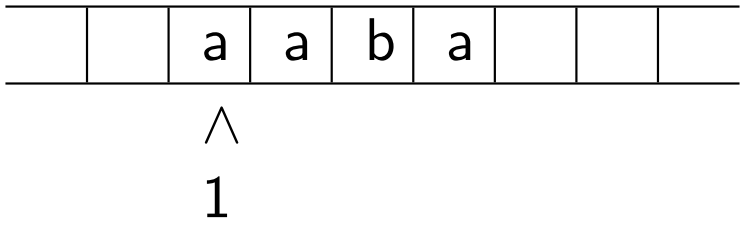
a	a	b	a

$\wedge$

2

Caractéristiques d'une machine de Turing

Support illimité de l'information : Ruban

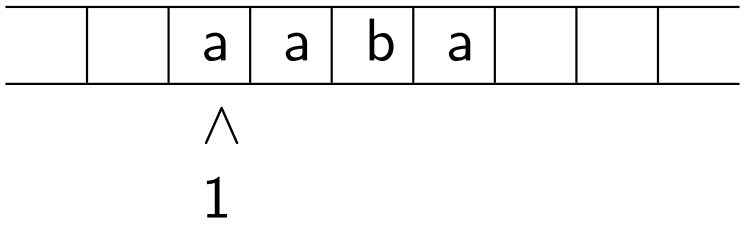


Machine de Turing

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

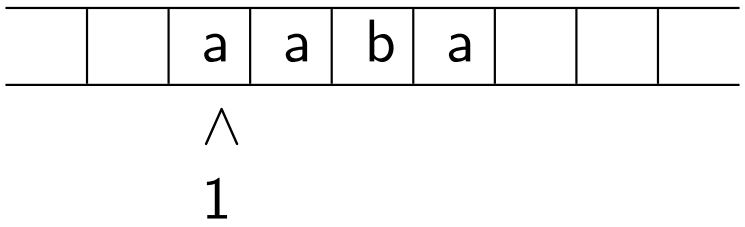
Transition

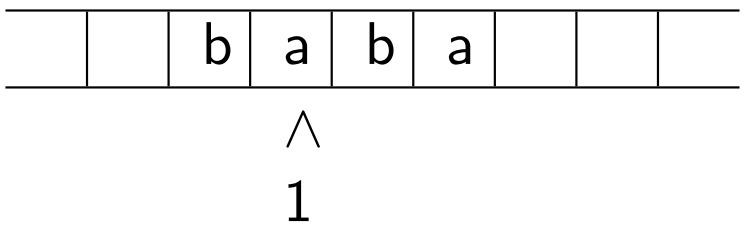
Ancien état	Symbole lu	Symbole écrit	Mouv.	Nouvel état
1	□	□	→	accepté
	a	b	→	1
	b	a	→	1



Transition

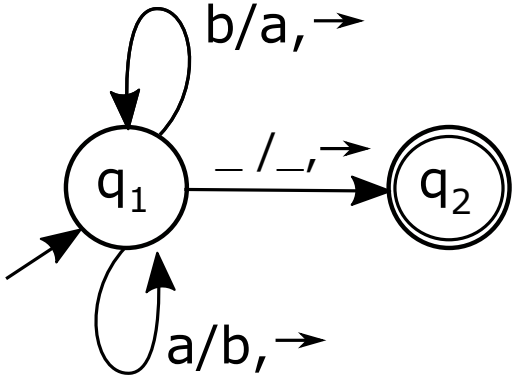
Ancien état	Symbole lu	Symbole écrit	Mouv.	Nouvel état
1	□	□	→	accepté
	a	b	→	1
	b	a	→	1

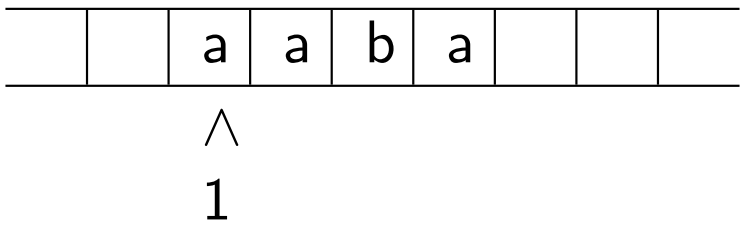




Transition : Tableau à double entrée

	a	b	□
$\rightarrow 1$	b , $\rightarrow$, 1	a , $\rightarrow$, 1	accepté





Exercice 1

Exécuter la machine de Turing ci-dessus.

Quel est le langage reconnu par la machine ?

Transition : Tableau à double entrée

Exercice 1

Le langage reconnu est $\Sigma^* = (a + b)^*$, l'ensemble de tous les mots.

Exemple : M2

	a	b	□
→0	□ , →, 1	□ , →, 2	accepté
1	a , →, 1	b , →, 1	□, ←, 3
2	a , →, 2	b , →, 2	□, ←, 4
3	accepté	refusé	accepté
4	refusé	accepté	accepté

	a	a	a	b	b	b		

	b	a	a	b	a	b		

Exercice 2

Exécuter la machine de Turing ci-dessus.
Quel est le langage reconnu L_{M2} par la machine M2 ?
L_{M2} aurait-il pu être reconnu à l'aide d'un automate fini ?

Exemple : M2

Exercice 2

Le langage reconnu est l'ensemble de tous les mots qui commencent et terminent par le même symbole. Il peut être reconnu par un automate fini.

Exemple : M3

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

	a	a	a	b	b	b		

	a	a	a	b	a	b		

Exercice 3

Exécuter la machine de Turing ci-dessus.
Quel est le langage reconnu L_{M3} par la machine M3 ?
L_{M3} aurait-il pu être reconnu à l'aide d'un automate fini ?

Exemple : M3

Exercice 3

Le langage reconnue est $L_{M3} = \{a^n b^n \mid n \geq 0\}$.
Il ne peut pas être reconnu à l'aide d'un automate fini.

Cet automate est à connaitre quasiment par coeur.

Définition formelle

Machine de Turing (MT)

Une machine de 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

$\}$

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.

Machine de Turing : point de vue calcul

Fonction de Σ^* à valeur dans Σ^* :

$$M : \Sigma^* \to \Sigma^*$$

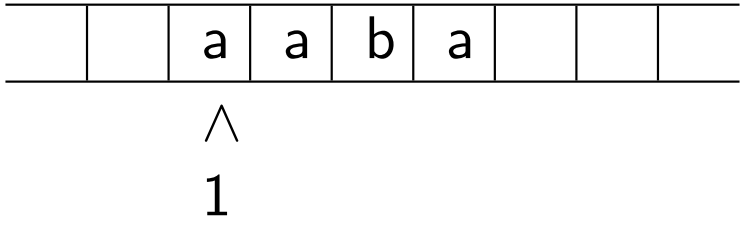
Définition : Fonction calculée

La **fonction calculée** par une MT M, notée f_M, est définie par :
Pour toute entrée $x \in \Sigma^*$ sur laquelle $M(x)$ s'arrête,
on associe la sortie $y \in \Sigma^*$.
Pour toute entrée $x \in \Sigma^*$ sur laquelle $M(x)$ ne s'arrête pas,
aucune image n'est associée à x.

Pour tout $x \in \Sigma^*$, $f_M(x) = y$ si et seulement si $M(x) = (\text{arrêt}, y)$

Exemple simple

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



Exercice 4

Exécuter la machine de Turing ci-dessus.
Décrire la fonction calculée.

Exemple simple

Exercice 4

L'automate associe à tout mot fini, le mot où les symbole a et b ont été permuté. Plus précisement,

$$f_{M4}(\sigma_1\sigma_2\ldots\sigma_p) = \sigma_1^{'}\sigma_2^{'}\ldots\sigma_p^{'} \text{ tel que } \sigma_i^{'} = \begin{cases} b & \text{si } \sigma_i = a \\ a & \text{si } \sigma_i = b \end{cases}$$

Exemple : définition de machine

Exercice 5

Définir une chaine de Turing sur l'alphabet $\{a, b, c\}$ qui enlève les symboles c.

Par exemple, $M(accbbca) = a\_\_bb\_a$

Exercice 6

Définir une chaine de Turing sur l'alphabet $\{a, b, c\}$ qui projette les mots sur l'alphabet $\{a, b\}$, autrement dit, qui "gomme" les symboles c.

Par exemple, $M(accbbca) = abba$

Exemple : correction

Exercice 5

1 seul état qui déplace la tête de lecture à droite jusqu'à la fin du mot repéré par la case vide. Lors de la lecture de c, la tête écrit une case vide.

	a	b	c	□
$\rightarrow 1$	a , $\rightarrow$, 1	b , $\rightarrow$, 1	□, $\rightarrow$, 1	arrêt

Exemple : correction

Exercice 6

L'état 1 a la même fonction, excepté lors de la lecture de c qui initie le sous-programme "décalage du mot d'une case à gauche". Les états 3, 4, et 5 mémorisent le symbole à recopier dans la case de gauche, l'état 2 permet la lecture du premier symbole, enfin l'état 6 permet un repositionnnement sur le symbole suivant et l'itération du sous-programme.

	a	b	c	$\square$
$\rightarrow$1	a , $\rightarrow$, 1	b , $\rightarrow$, 1	$\square$, $\rightarrow$, 2	arrêt
2	$\square$, $\leftarrow$, 3	$\square$, $\leftarrow$, 4	$\square$, $\leftarrow$, 5	$\square$, $\leftarrow$, 1
3				a, $\rightarrow$, 6
4				b, $\rightarrow$, 6
5				c, $\rightarrow$, 6
6				$\square$, $\rightarrow$, 2

Exemple : calcul en unaire

Il est possible de représenter les nombres entiers plus grand que 1 en écriture unaire.
Le nombre x est représenté par x "batons".

Par exemple, le nombre (en écriture décimal) 3 est représenté par :

		1	1	1				

Exercice 7

Définir une machine de Turing qui calcule l'addition de 2 nombres écrits en unaire qui sont séparés par un seul espace. Par exemple, $3 + 2$:

		1	1	1		1	1	

Exemple : calcul en unaire

Exercice 7

De nombreuses solutions sont possibles comme toujours. Ici, le principe consiste à boucher l'espace entre les nombres et effacer le dernier baton.

	1	□
$\rightarrow$1	1 , $\rightarrow$, 1	1, $\rightarrow$, 2
2	1 , $\rightarrow$, 2	□ , $\leftarrow$, 3
3	□ , $\rightarrow$, arrêt	

Réflexion

Questions, réflexions

Pourquoi avoir défini une machine de Turing comme cela ?

A-t-on défini une machine qui "peut" tout calculer ?

Est-ce qu'il existe des machines plus puissantes ou qui calculent autre chose ?

Relisons Turing in the text

Réflexion

Questions, réflexions

Pourquoi avoir défini une machine de Turing comme cela ?

A-t-on défini une machine qui "peut" tout calculer ?

Est-ce qu'il existe des machines plus puissantes ou qui calculent autre chose ?

Relisons Turing in the text

Définition : machines équivalentes

Deux machines de Turing sont **équivalentes**
si elles calculent la même fonction.

Imaginons

Pourquoi ne pas imaginer une machine avec 2 têtes de lectures pour pouvoir lire plus de choses ou aller plus vite ?

Exercice 8

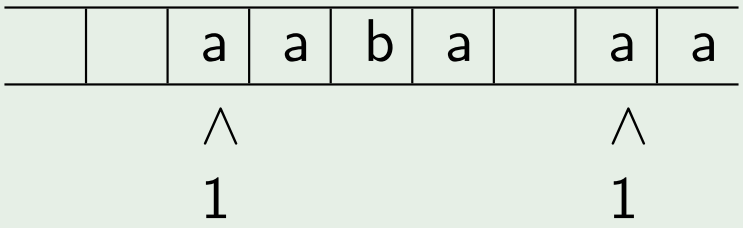
Définir une machine de Turing à 2 têtes de lecture.

Montrer que la machine à deux têtes de lecture est équivalente à une machine de Turing à 1 tête de lecture.

Imaginons : correction

Exercice 8

Une machine a deux têtes de lecture peut se définir comme ci-dessous. Un seul état commun pour la machine, mais 2 têtes qui peuvent lire, écrire et se déplacer indépendamment.



Une règle de transition doit prendre en compte la lecture des 2 symboles, écrire 2 symboles et donner 2 ordres de déplacement :

$$q, \sigma_1, \sigma_2 \rightarrow q', \sigma_2', \sigma_2', m_1, m_2$$

La table de règles de transitions peut alors s'écrire comme suit :

	a a	a b	b a	b b	□ a	□ b	a □	b □
→1	a b , →←, 1	a b , →→, 3	...	...	...	...	...	...
2	...							
...	...							

Imaginons : correction

Exercice 8

Une MT à 2 têtes est équivalente à une MT à 1 tête. Il est simple de voir qu'une machine à 1 tête peut simuler avec 2 têtes.
Pour l'inverse, il faut construire l'alphabet produit. L'alphabet de la MT à 1 tête est alors $\Sigma \times \Sigma$. Pour simuler le déplacement sur 2 cases à la fois, il faut aussi construire le ruban produit $\mathbb{Z} \times \mathbb{Z}$ qui est une grille à 2 dimensions. Les déplacements sont alors les 4 déplacements possibles en diagonal. Néanmoins pour être parfaitement en adéquation avec la définition d'un ruban à 1 dimension il faut utiliser la bijection de $\mathbb{Z} \times \mathbb{Z}$ dans $\mathbb{Z}$ avec les déplacements qui ne sont alors plus unitaire. Mais là encore il possible d'ajouter des états à la machine de Turing qui permettent ses déplacements longues distances sans modifier le ruban.

Plusieurs rubans

Pourquoi ne pas imaginer une machine avec 2 rubans ?

Exercice 9

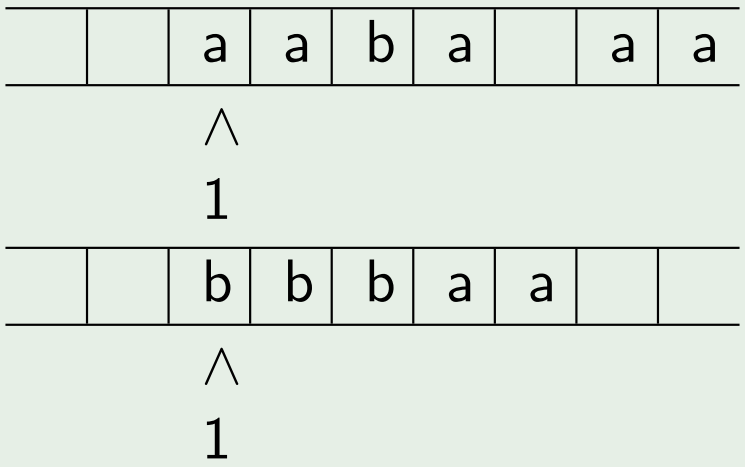
Définir une machine de Turing à 2 rubans.

Montrer que la machine à deux rubans est équivalente à une machine de Turing à 1 ruban.

Imaginons : correction

Exercice 9

Une machine a deux rubans se défini de la même manière qu'une machine à 2 têtes de lecture. Un seul état commun pour la machine, mais 2 têtes qui peuvent lire, écrire et se déplacer indépendamment sur 2 rubans indépendants.



Une règle de transition doit prendre en compte la lecture des 2 symboles, écrire 2 symboles et donner 2 ordres de déplacement :

$$q, \sigma_1, \sigma_2 \rightarrow q', \sigma_2', \sigma_2', m_1, m_2$$

La construction et la preuve d'équivalence sont similaires.

Machines de Turing équivalentes

On peut imaginer beaucoup de variantes de MT :

- Deux têtes de lectures
- 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

En utilisant 2 rubans :

(1) sur un ruban le programme de la machine de Turing originale

(2) sur l'autre ruban, l'entrée et le calcul de cette machine

La machine a exécuter est écrite sur un ruban sous forme de mot, le second ruban est l'entrée de la machine à exécuter, et la machine de Turing universelle lit et exécute la machine à exécuter. Cette propriété universalité montre en quelque sorte une capacité réflexive à la machine de Turing.

Machine de Turing universelle

Existence

(1) En effet, on peut coder à l'aide d'un mot la table de transition d'une machine de Turing. Chaque règle peut être codée par un mot :
Chaque état i peut être représenté en unaire par la succession de i symboles E terminés par un symbole &. Les symboles $\rightarrow$ et $\leftarrow$ peuvent être codés par R et L. Le symbole vide d'entrée peut être aussi codé par un symbole B, et l'arrêt par le symbole S par exemple.
Par exemple la table suivante

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

E&abRE&E&baRE&E&BBRS&

(2) pour écrire une MT universelle, on peut utiliser un troisième ruban qui mémorise l'état actuelle de la machine à simuler. Il s'agit alors de définir une machine qui recherche (compare) l'état actuelle dans le ruban programme, puis lit le symbole sur le ruban d'entrée et simule la règle en écrivant et bougeant la tête de lecture sur le ruban d'entrée.

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 ? Oui puisque toute table de transition peut être codée sous forme de mot, et que l'ensemble des mots est dénombrable.
  D'ailleurs c'est aussi le cas pour les langages java, c++, etc.
- Existe-il un ensemble de fonctions non-dénombrables ?
  Oui, par exemple l'ensemble des fonctions $f : \mathbb{N} \to \mathbb{N}$ est non-dénombrable.
- Existe-t-il des fonctions non calculables ?
  Oui car il y a (beaucoup) plus de fonctions à calculer que de machine de Turing...

On peut rêver à ces fonctions non-calculables qu'aucun ordinateur ne pourra jamais calculer
Cela serait sympa d'en voir un exemple concret...

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...

Phrase autoréférente

À 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 : fonction caractéristique

Soit $\{P(x) \mid x \in \Sigma^*\}$ une famille dénombrable de propriétés.
La fonction caractéristique, notée f_P, de cette famille P est définie par :

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

Décidabilité

Définition : fonction caractéristique

Soit $\{P(x) \mid x \in \Sigma^*\}$ une famille dénombrable de propriétés.
La fonction caractéristique, notée f_P, de cette famille P est définie par :

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

Définition : décidabilité

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

Problème de l'arrêt

Fonction arrêt des Machines de Turing

Fonction A qui associe l'arrêt à chaque machine de Turing :

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

Question

Peut-on calculer la fonction A avec une machine de Turing ?

Problème de l'arrêt : fonction non calculable

Problème de l'arrêt

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

Aucune machine de Turing, aucun ordinateur ne pourra calculer l'arrêt d'un programme !...

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
- ...

Esquisse de la preuve : argument diagonal de Cantor

L'ensemble des MT est dénombrable. L'alphabet des machines de Turing va être supposé inclus dans l'alphabet qui permet de décrire une MT (cf. MT universelle).
La fonction d'arrêt A peut être représentée sous forme d'un tableau comme ci-dessous. En ligne toutes les MT, et en colonne toutes les entrées possibles.

	e_1	e_2	e_3	e_4	e_5	...
M_1	1	0	1	1	0	...
M_2	0	0	1	1	1	...
M_3	1	0	1	1	1	...
M_4	1	1	1	0	0	...
M_5	0	1	1	1	0	...
...	...					

	e_1	e_2	e_3	e_4	e_5	...
M_1	$1/0$	0	1	1	0	...
M_2	0	$0/1$	1	1	1	...
M_3	1	0	$1/0$	1	1	...
M_4	1	1	1	$0/1$	0	...
M_5	0	1	1	1	$0/1$	...

- Supposons qu'il existe une MT M_A qui calcule la fonction A.
- Définissons la MT suivante :

$$M(e) = \begin{cases} \text{si } M_A(e,e) = 1 & \text{alors faire une boucle infinie} \\ \text{si } M_A(e,e) = 0 & \text{alors arrêt} \end{cases}$$

M_A étant une MT, M est une MT correctement définie sur les entrées qui sont potentiellement des MT elles-même. Donc M est sur une ligne du tableau. Supposons $M = M_i$ (valeurs de M sont indiquées sur la diagonale).

- Si $A(M_i, M_i) = 1$ alors $M_A(M_i, M_i) = 1$, donc par définition de $M = M_i$, $M_i(M_i)$ ne s'arrête pas, donc $M_A(M_i, M_i) = 0$, ce qui est contradictoire.
- Même raisonnement lorsque $A(M_i, M_i) = 0$. On obtient une contradiction. Il n'existe pas de MT qui calcule l'arrêt. Remarque : la construction de M repose sur l'inversion de la diagonale (cf. Cantor).

Conclusion

Merci à Alan Turing !

pour cette formidable machine
qui a ouvert la porte en grand à l'informatique