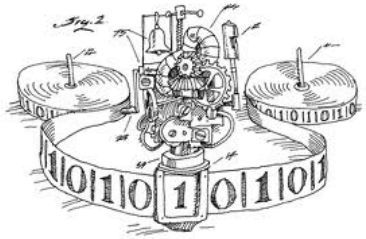
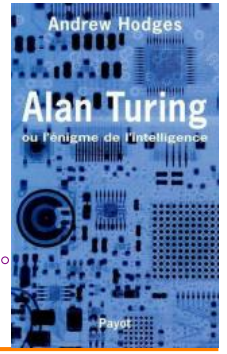
11 - Machines de Turing



Introduction

- Alan Turing* inventa cette machine abstraite en 1936
- il s'agissait alors de clarifier la notion de procédure effective
- toute tâche exécutée par une machine de Turing MT peut l'être par un ordinateur ... et vice-versa !
- une machine de Turing
  - se programme (un peu comme un ordinateur)
  - peut simuler n'importe quel automate fini ou automate à pile vus jusqu'ici
- il existe une MT universelle capable de simuler toutes les autres MT



Idée-cadeau !

* Alan Turing (1912-1954)
mathématicien britannique

Principe

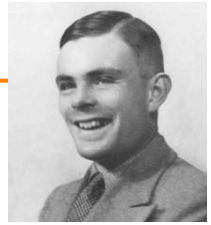
- une MT est composée d'une unité de contrôle, d'un ruban semi-infini divisé en cases, d'une tête de lecture/écriture (RWH)
- une MT change d'état en fonction de ce qu'elle lit sur la case courante et de son état courant
  A chaque instant, conformément à sa relation de transition δ, elle peut :
  - écrire sur sa case courante
  - déplacer sa RWH d'1 case (à droite, à gauche, ou rester stationnaire)
  - changer d'état
- le ruban procure une mémoire non-bornée accessible en tout point (et pas seulement LIFO)
- l'arrêt de la machine a lieu quand elle entre dans un état final ou bien si elle ne peut plus évoluer
- on appelle entrée la chaîne écrite initialement sur son ruban

Hiérarchie de Chomsky*

Type	Langage	Grammaire	Procédure effective
3	Rationnels (réguliers)	Régulières à droite régulières à gauche	Automates finis
2	Hors-contexte (algébriques)	Hors-contexte (algébriques)	Automates à pile
1	Contextuels	Contextuelles monotones	Machine de Turing à l'espace linéairement borné par la taille de l'entrée (LBA)
0	Récursivement énumérables	Contextuelles avec effacement	Machine de Turing

* Noam Chomsky, né en 1928, linguiste américain

Machine de Turing



Une machine de Turing à 1 ruban semi-infini est un septuplet $(Q,\Gamma,\Sigma,\delta,q_0,B,F)$ où :

- Q : ensemble fini d'états
- Γ : alphabet fini des symboles de ruban
- $\Sigma \subseteq \Gamma$: alphabet fini des symboles d'entrée
- $B \in \Gamma \setminus \Sigma$: symbole particulier dit « blanc »
- $q_0 \in Q$: état initial
- $F \subseteq Q$: ensemble des états d'acceptation
- δ : relation de transition

La MT est déterministe si, pour chaque configuration, elle a au plus 1 possibilité d'évolution.

Transition

- relation de transition
$$\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{G,D,S\}$$
- la transition $(q,\sigma \rightarrow q',\sigma',s)$ considère :
  - l'état courant q de la machine
  - le caractère lu σ sur le ruban
- une règle de transition indique :
  - le caractère σ' à écrire sur la case courante
  - le cas échéant, le sens s du déplacement à effectuer par la tête de lecture/écriture
  - le prochain état q' de la machine

Reconnaissance

- une machine de Turing M peut accepter des entrées en vue de la reconnaissance d'un langage
- le langage accepté par M est ainsi défini :

  $L(M) = \{ w \in \Sigma^* \text{ tels que :}$
  - q_0 est l'état initial de M
  - w est calé à gauche du ruban
  - la RWH pointe sur la première lettre de w
  - l'exécution de M lit l'intégralité du mot w
  - M atteint un état d'acceptation dans F $\}$

Exemple

La machine de Turing $M = (Q,\Gamma,\Sigma,\delta,q_0,\#,F)$

$\Gamma = \{a,b,X,Y,\#\}$
$\Sigma = \{a,b\}$
$Q = \{q_0,q_1,q_2,q_3,q_4\}$
q_0 initial

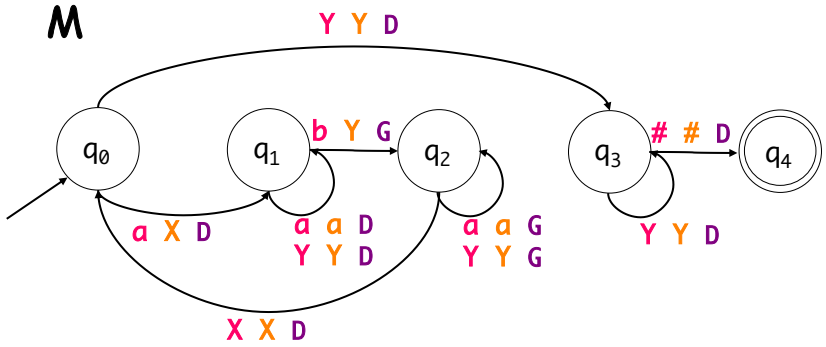
$\delta = \{ (q_0,a,q_1,X,D), (q_0,Y,q_3,Y,D),$
$(q_1,a,q_1,a,D), (q_1,b,q_2,Y,G),$
$(q_1,Y,q_1,Y,D), (q_2,a,q_2,a,G),$
$(q_2,X,q_0,X,D), (q_2,Y,q_2,Y,G),$
$(q_3,Y,q_3,Y,D), (q_3,\#,q_4,\#,D) \}$

$F = \{q_4\}$

M accepte le langage $L(M) = \{a^n b^n, n > 0\}$

Exemple

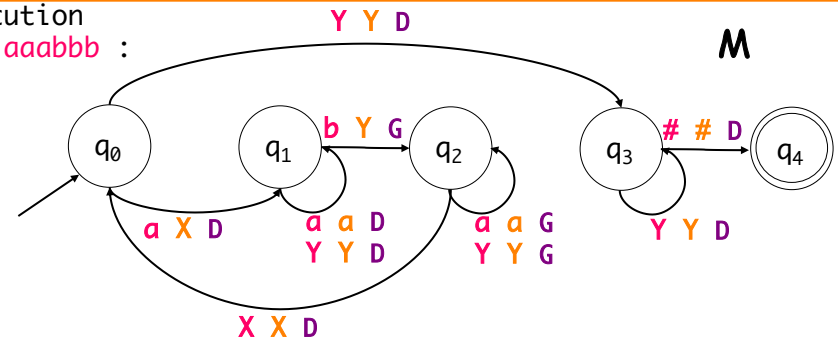
$\delta = \{(q_0,a,q_1,X,D),(q_0,Y,q_3,Y,D),(q_1,a,q_1,a,D),(q_1,b,q_2,Y,G),$
$(q_1,Y,q_1,Y,D),(q_2,a,q_2,a,G),(q_2,X,q_0,X,D),(q_2,Y,q_2,Y,G),$
$(q_3,Y,q_3,Y,D),(q_3,\#,q_4,\#,D)\}$



$L(M)=\{a^n b^n, n>0\}$ *(d'accord, un automate à pile aurait suffit ...)*

Exemple

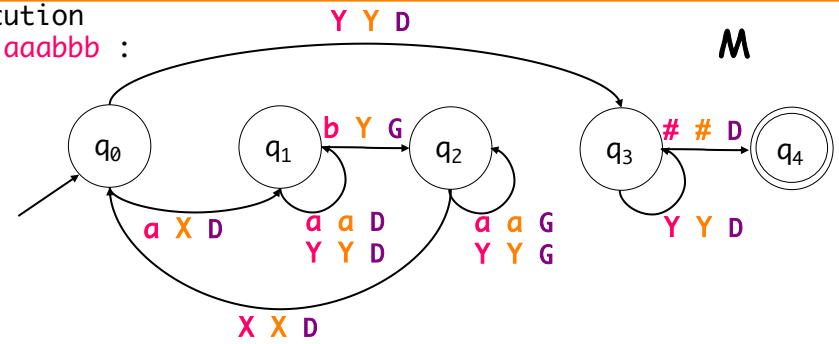
Exécution sur aaabbb :



q_0	↑a a a b b b # #	q_2	X a↑a Y b b # #
q_1	X↑a a b b b # #	q_2	X↑a a Y b b # #
q_1	X a↑a b b b # #	q_2	↑X a a Y b b # #
q_1	X a a↑b b b # #	q_0	X↑a a Y b b # #

Exemple

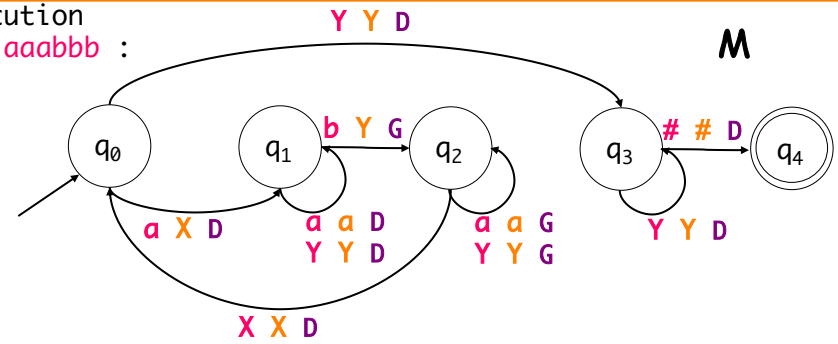
Exécution sur aaabbb :



q_1	X X↑a Y b b # #	q_2	X X↑a Y Y b # #
q_1	X X a↑Y b b # #	q_2	X↑X a Y Y b # #
q_1	X X a Y↑b b # #	q_0	X X↑a Y Y b # #
q_2	X X a↑Y Y b # #	q_1	X X X↑Y Y b # #

Exemple

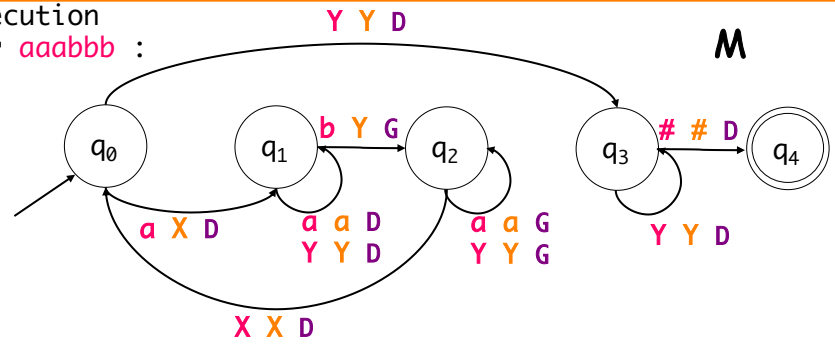
Exécution sur aaabbb :



q_1	X X X Y↑Y b # #	q_2	X X↑X Y Y Y # #
q_1	X X X Y Y↑b # #	q_0	X X X↑Y Y Y # #
q_2	X X X Y↑Y Y # #	q_3	X X X Y↑Y Y # #
q_2	X X X↑Y Y Y # #	q_3	X X X Y Y↑Y # #

Exemple

Exécution sur aaabbb :



q_3 X X X Y Y Y # #

q_4 X X X Y Y Y # #

➢ après avoir lu le mot entièrement, la machine entre dans un état final et *elle s'arrête* : le mot aaabbb appartient donc au langage L(M).

Définitions équivalentes

La définition des machines de Turing peut être modifiée sans changer leur pouvoir de reconnaissance :

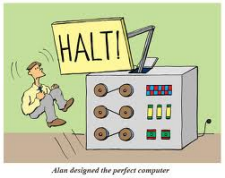
- il peut y avoir plusieurs rubans
- le(s) ruban(s) peu(ven)t être bi-infini(s)
- la tête de lecture/écriture peut ne pas pouvoir rester stationnaire
- on peut écrire ou non le symbole « blanc »
- ...

A vouloir améliorer les MT, on retombe **toujours** sur des machines reconnaissant la même classe de langages !

➢ elles semblent englober toute idée de « procédure effective »

On choisira alors d'utiliser la définition la plus adaptée au problème traité.

Procédure effective



Alan designed the perfect computer

- une machine de Turing (MT) « s'arrête » *(halts en anglais)* si :
  - elle atteint un état final

  ou
  - elle ne peut plus effectuer de transition
- c'est alors une procédure effective qui donne lieu à un algorithme ; mais une MT peut ... ne jamais s'arrêter !
  ➢ elle évolue indéfiniment sans atteindre d'états finals
- un langage reconnu par une MT qui « s'arrête » sur toutes les entrées s'appelle un langage récursif
  exemple : l'ensemble des nombres premiers est récursif
- un langage peut cependant être « énuméré » par une MT qui peut ne pas « s'arrêter » sur des entrées en dehors du langage :
  ➢ un tel langage est dit récursivement énumérable (r.é.)
  ➢ un langage r.é. est engendré par une grammaire de type 0.

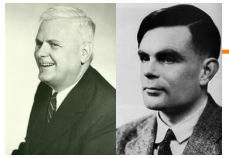
Calcul de fonction

La sortie *(output)* d'une machine de Turing M est la chaîne inscrite sur son ruban n° 1 lorsqu'elle s'arrête :

- à toute entrée x sur laquelle M s'arrête on associe la sortie y
- la correspondance entre les entrées et les sorties est la fonction calculée par M
- survient le problème du codage des entrées/sorties :
  Exemple :
  une fonction f sur un entier n pourra être calculée en unaire en plaçant n fois le symbole 1 en entrée, on obtient $1^{f(n)}$ en sortie si la machine s'arrête

Les machines de Turing servent d'étalon au calcul de la complexité des algorithmes sur lequel se fonde la théorie de la complexité (i.e. complexité des problèmes).

Thèse de Church-Turing



Thèse de Church*-Turing *(indépendamment mais les 2 en 1936)*
Les fonctions calculables par une **procédure effective** le sont par une machine de Turing

- adopter cette thèse, c'est choisir une modélisation du concept de procédure effective
- il n'y a pas de démonstration car ce n'est pas un théorème
- les **fonctions calculables** par les machines de Turing coïncident avec la classe des fonctions décrites par le **λ-calcul** de Church
- la théorie de la calculabilité se fonde sur cette thèse
- elle permet aussi de concevoir l'existence de fonctions non-calculables, par un simple argument de diagonalisation

* Alonzo Church (1903-1995) : mathématicien et logicien américain

Le λ-calcul en bref

- on se donne un ensemble *dénombrable* de variables Var
- 4 symboles :

  λ . ( )
- et 2 opérations : application et abstraction
- définition inductive des λ-termes :

  (B) tout v de l'ensemble Var est un λ-terme

  (I) si t et t' sont des λ-termes, si v ∈ Var, alors
  (t t') est un λ-terme
  (λv . t) est un λ-terme

(on peut omettre les parenthèses autour de l'application si ça ne crée pas d'ambiguïté et curryfier les abstractions imbriquées)

Exemples

- comment modéliser les entiers ?
  $\bar{0}$: λf x. x
  $\bar{1}$: λf x. f x
  $\bar{2}$: λf x. f(f x)
  ...
- la fonction successeur notée SUCC ?
  $\lambda \bar{n}\, f\, x\, .\, f((\bar{n}\, f)\, x)$
- l'addition notée ADD ?
  $\lambda \bar{n}\, \bar{m}\, f\, x\, .\, (\bar{n}\, f)((\bar{m}\, f)\, x)$
- la multiplication MULT ?
  $\lambda \bar{n}\, \bar{m}\, f\, x\, .\, (\bar{n}\, (\bar{m}\, f))\, x$

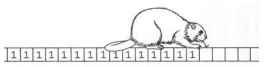
Toutes les fonctions calculables se traduisent en λ-calcul ...

Une fonction non-calculable

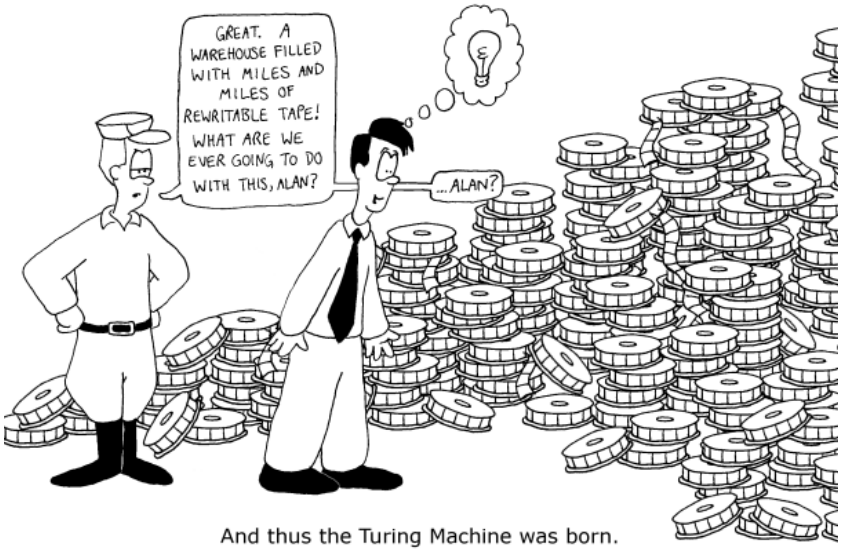
- en 1962, Tibor Radó* montre que la fonction dite « ***Le castor affairé*** **» n'est pas calculable
- Σ(n) est défini comme étant le plus grand nombre codé en unaire que peut calculer une MT à n états ***avant de s'arrêter***
- seules les premières valeurs pour n<5 sont connues avec précision :
  (S(n) est le nombre d'étapes de la machine)

Σ(0) = 0	S(0) = 0
Σ(1) = 1	S(1) = 1
Σ(2) = 4	S(2) = 6
Σ(3) = 6	S(3) = 21
Σ(4) = 13	S(4) = 107

- au-delà, on sait depuis 1989 que Σ(5) ≥ 4098 avec S(5) ≥ 47176870
  et aussi, depuis juin 2010, que $\Sigma(6) \geq 3{,}5.10^{18267}$



* Mathématicien hongrois (1895-1965)
** *Busy beaver* en anglais : les MT se nomment BB(n) pour cela.



And thus the Turing Machine was born.