CAPES MI

Turing Machines and decidable problems

Laure Gonnord
http://laure.gonnord.org/pro/teaching/
Laure.Gonnord@univ-lyon1.fr

Université Claude Bernard Lyon1

2017

1 Motivation

2 Turing Machines

3 Decidability - Complexity

Un peu en vrac

- Informatique fondamentale
- Formalisation de ce qu'est un calcul (mais avant les ordinateurs)
- Notion d'impossibilité, de problème difficile

Notion de Modèle de calcul

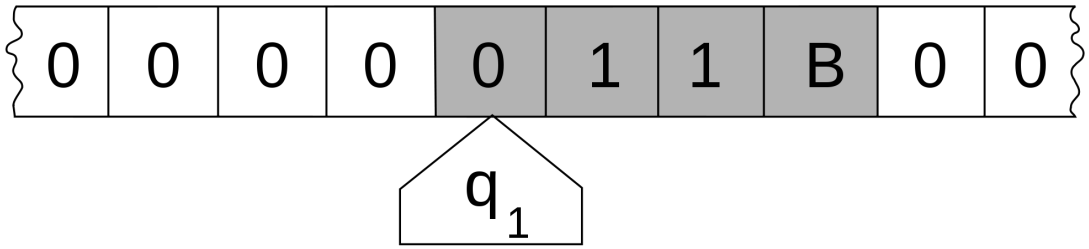
- Les classes UML - les BD
- Les Graphcet
- Les automates
- Les machines de Turing

1 Motivation

2 Turing Machines

3 Decidability - Complexity

Turing Machine, def - 1



Turing Machine, Turing, 1935

- A **tape** (infinite in both sides) : the memory. (it has a working alphabet which contains a **blank** symbol).
- A **head** : read/write symbols on the tape.
- A finite transition table (or function)

image credits : Wikipedia.org

Turing Machine, def - 2

Turing Machine elements

- States (Q), initial state (q_0), final (accepting) states (F).
- One alphabet Γ for the tape.
- $B \in \Gamma$ the blank letter.
- Transitions are finitely many : read the letter under the head, and then :
  - Erase a symbol or write one under the head
  - Move the head (Left, Right, or Stay)
  - The “state” can be modified.

- The transition function is :

$$Q \times \Gamma \times \rightarrow Q \times \Gamma \times \{L, R, S\}$$

An Example

TM that decides if x is odd (impair) : (final State : q_4)

State/char	0	1	B
q_1	(q_1,0,R)	(q_1,1,R)	(q_2,B,L)
q_2	(q_3,0,L)	(q_4,1,L)	—

Play on $BBBBBBB11BBBBB$ and $BBBBBBB10BBBBB$

- A **configuration** is a tuple (word on the tape, position of the head, state).

Adapted from : `http://www.madchat.fr/coding/algo/algo_epfl.pdf` slide 4

Another Example

Demo : `https://turingmachinesimulator.com/`

General results 1

There exists Turing machines for the following languages :

- palindromes
- $a^n b^n c^n$ (non algebraic language)
- a^i with i prime (non algebraic)
- a^{n^2}, with $n \geqslant 0$

▶ Turing machines are more powerful than all other models (we have seen yet)

Decidable Languages

A language that is recognised by a Turing Machine is said to be decidable.

Accepting or computing

A TM can also compute functions

TM that writes 1 if x is odd, 0 else (q_6 is final state) :

State/char	0	1	B
q_1	(q_1,0,R)	(q_1,1,R)	(q_2,B,L)
q_2	(q_3,B,L)	(q_4,B,L)	—
q_3	(q_3,B,L)	(q_3,B,L)	(q_5,1,R)
q_4	(q_4,B,L)	(q_4,B,L)	(q_5,0,R)
q_5	—	—	(q_6,B,R)

Play on $BBBBBBB11BBBBB$ and $BBBBBBB10BBBBB$

Another Example

Demo of a TM computing the addition.

General results 2 - extensions

- infinite tape in both directions
- more than one tape
- many heads per tape
- bidimensional tape
- non determinism
- random access

All these extensions can be simulated by a standard machine.

Example : Infinite tape

(copyright S.Brandel)

- Pour chaque type d'extension, nous montrons que l'opération de la machine étendue peut être simulée par une machine de Turing normale.
- La démonstration consiste dans chaque cas :
  (1) à montrer comment construire une machine normale à partir de la machine étendue considérée,
  (2) à prouver que la machine normale construite simule correctement le comportement de la machine de départ.

Example : Infinite tape

- Soit une machine de Turing M = (K, $\sum$, Γ, δ, q_0, F) dont le ruban n'a pas de borne à gauche :
  - la chaîne d'entrée peut se trouver n'importe où sur le ruban,
  - la tête pointe sur le premier blanc à gauche de la chaîne.
- Dans cette machine, une configuration est de la forme : (q, w$\underline{a}$u) avec q $\in$ K, w, u $\in \Gamma^*$, a $\in \Gamma$, où :
  - w ne commence pas par un blanc
  - u ne finit pas par un blanc.
- On étend la relation entre configurations pour prendre en compte les déplacements à gauche :
  - si δ(q, a) = (p, b, G) alors (q, $\underline{a}$u) $\vdash_M$ (p, $\underline{\#}$bu).

Example : Infinite tape

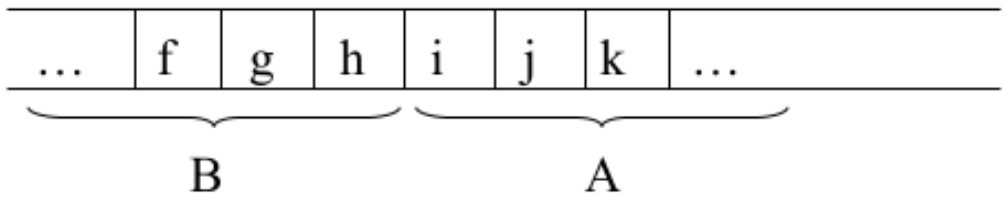
- Montrons qu'une machine M avec ruban infini dans les deux sens n'est pas plus puissante qu'une machine normale (dans le sens qu'elle ne permet pas de reconnaître plus de langages, ou calculer plus de fonctions).
- Pour cela montrons comment construire une machine M' = (K', $\sum$, Γ', δ', q_0', F'), à partir de M et qui simule M :
  - si M s'arrête sur un mot w, alors M' s'arrête sur ce même mot w,
  - si M ne s'arrête pas sur un mot w, alors M' ne s'arrête pas non plus sur ce même mot w.

Example : Infinite tape

- Pour simuler le ruban doublement infini de M dans celui de M', on définit pour M' un ruban à 2 pistes :
- Ce ruban est obtenu en coupant en 2 celui de M de façon arbitraire.

- Exemple

Ruban de M :



Ruban de M' :

$	i	j	k	…
	h	g	f	…

General results 3

There exists Turing machines for the computation of :

- $x \mapsto x + 1$
- $(x, y) \mapsto x + y$
- $x \mapsto x^2$
- all the functions you are able to write on computers

Computable functions

A function (defined for all its input) that is computable by a Turing Machine is said to be TM computable

▶ A Model of what can be computed with machines. (Church Thesis - see later)

Important result 1 - Universal MT

- Existe-t-il une machine de Turing qui peut simuler n'importe quelle machine de Turing ?
- Le but est de construire une machine de Turing M à laquelle on fournit :
  - la description d'une machine de Turing quelconque M'
  - un mot w

  et qui simule l'exécution de M sur w.
- En clair construire une machine de Turing qui serait un interpréteur de machines de Turing…

- Ces machines de Turing existent et sont appelées machines de Turing universelles.

Church

Thèse de Church – Turing

- Formulation Lewis – Papadimitriou

  Un algorithme est une machine de Turing qui s'arrête pour toutes ses entrées.

  We therefore propose to adapt the Turing machine that halts on all inputs as the precise formal notion corresponding to the intuitive notion of an "algorithm".

- Formulation Wolper

  Les langages reconnus par une procédure effective sont ceux décidés par une machine de Turing.

- Encore une autre formulation

  Every computational process that is intuitively considered to be an algorithm can be converted to a Turing machine

1 Motivation

2 Turing Machines

3 Decidability - Complexity

Decidable - Semi-decidable languages

If L is a language, L is **decidable** if there exists a Turing Machine (or an algorithm) that outputs for all w :

- 1 if $w \in L$
- else 0.

Semi-decidable :

- 1 if $w \in L$
- else does not terminate

- equivalent definition for problems.

Complexity

Link with computational complexity :

- The number of steps in the execution of a TM gives the **complexity in time**,
- The number of seen squares in the execution of a TM gives the **complexity in space**.

Examples of undecidable problems

- The halting problem for TM or counter automata (for more that 2 counters)
- Given a program, does it loops ?
- Is a given algebraic expression (with log, *, exp, sin, abs) equal to 0 ? (Richardson, 1968)
- The 10^{th} Hilbert Problem (Diophantine equations) - 1900 (23 non solved pbs 2017).
- Garbage Collecting
- Absence of bugs

Summary

Turing Machines :

- A model closed to **programming languages**.
- Non deterministic.
- Acceptors for decidable languages. But some languages are not decidable !
- (equivalently) Computes solutions to problems. But ...

Important fact

Some common problems are undecidable !