Note cours Machine de Turing

Valeran Maytié

Table des matières

1 Intro

Une machine de Turing est composée d'une bande et d'un automate avec deux principes en plus par transition :

— déplacement dans la bande (gauche, droite ou immobile)

— écriture sur la bande

Pendant l'exécution on a une configuration composée :

— de l'état actuel

— du pointeur dans la bande

— du contenu de la bande

Pour exécuter la machine de Turing on applique les transitions en partant de la configuration initiale (état de départ, début du mot, mot entré). L'exécution s'arrête quand un état acceptant est atteint. Pour appliquer les transitions, on reconnait la lettre pour trouver la/les bonnes transitions à appliquer (déterministe/non déterministe). Ensuite, on écrit sur la bande, puis on déplace le curseur. Enfin, on va à l'état désigné par la transition.

On va aussi définir un alphabet d'entrée (caractère utilisé dans la configuration initiale) et un alphabet de travail (caractère utilisé pour l'écriture).

Il existe beaucoup de variante des machines de Turing. Par exemple on peut avoir plusieurs bandes, plusieurs pointeurs, ...

Les machines de Turing peuvent être utilisé pour :

- décider : Reconnait ou non.
- calculer : transformer la bande
- énumérer : faire de l'affichage

Définition plus formelle des machines de Turing :

(Q, A, B, Q_0, Q_f, T)

avec : $Q_0 \subseteq Q, Q_f \subseteq Q, T = Q \times (B \times \{\leftarrow, I, \rightarrow\} \times B)^k \times Q$

— Q :États

— A :Alphabet d'entré

— B :Alphabet de travail

— Q_0 :États initia

— Q_f :États finals

— T : Transition (k nombre de bandes)

2 Première Machine de Turing

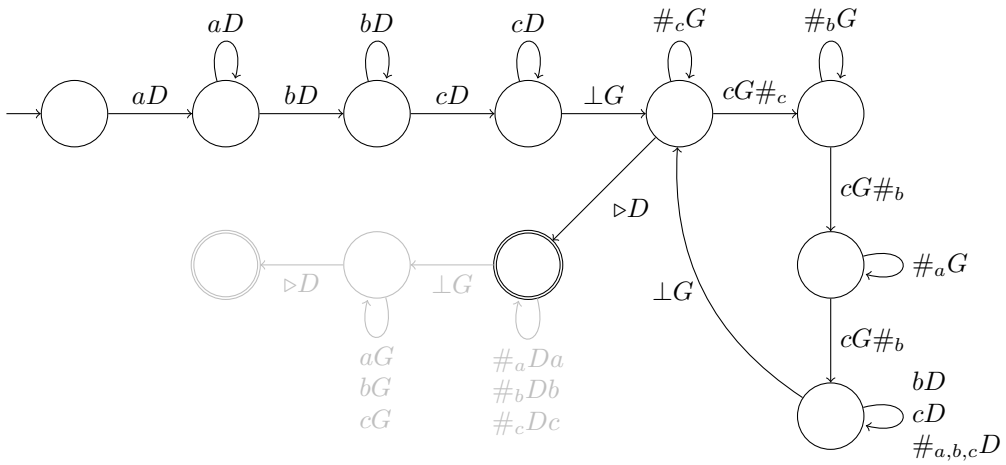


FIGURE 1 – Machine de Turing reconnaissant $\{a^n b^n c^n | n \geq 1\}$

On va exécuter la machine de Turing sur l'entrée

▷	a	a	a	b	b	b	c	c	c	...

On donne les lignes lorsqu'il y a une écriture

▷	a	a	a	b	b	b	c	c	c
▷	a	a	a	b	b	b	c	c	$\#_c$
▷	a	a	a	b	b	$\#_b$	c	c	$\#_c$
▷	a	a	$\#_a$	b	b	$\#_b$	c	c	$\#_c$
▷	a	a	$\#_a$	b	b	$\#_b$	c	$\#_c$	$\#_c$
▷	a	a	$\#_a$	b	$\#_b$	$\#_b$	c	$\#_c$	$\#_c$
▷	a	$\#_a$	$\#_a$	b	$\#_b$	$\#_b$	c	$\#_c$	$\#_c$
▷	a	$\#_a$	$\#_a$	b	$\#_b$	$\#_b$	$\#_c$	$\#_c$	$\#_c$
▷	a	$\#_a$	$\#_a$	$\#_b$	$\#_b$	$\#_b$	$\#_c$	$\#_c$	$\#_c$
▷	$\#_a$	$\#_a$	$\#_a$	$\#_b$	$\#_b$	$\#_b$	$\#_c$	$\#_c$	$\#_c$

La partie en gris n'est pas obligatoire (elle remet la mémoire dans son état d'origine si c'est accepté)

3 Problème de l'arrêt

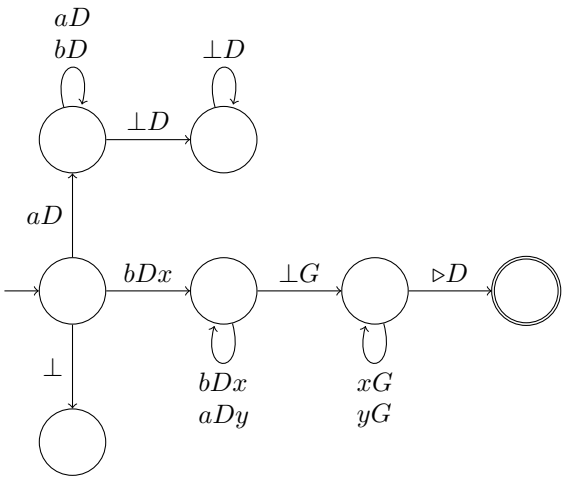


FIGURE 2 – Machine de Turing reconnaissant $b(a|b)^*$

On a 3 cas :

ε : arrêt non final $\rightarrow$ refusé

$b(a|b)^*$: arrêt final (après avoir remplacé les b par des x et les a par de y)

$a(a|b)^*$: exécution infinie (math : refusé) (info : comment le détecter ?)

Définition (Problème de l'arrêt) – Le problème de l'arrêt consiste à écrire une machine de Turing qui prend en entré une machine de Turing et son entée et détermine si elle s'arrête ou non.

Théorème (Indécidabilité du problème de l'arrêt) – Il n'existe pas de machine de Turing résolvant le problème de l'arrêt.

Preuve – Supposons qu'il existe une machine de Turing *Halt* qui prend en entré une machine de Turing et son entrée et dit décide le problème de l'arrêt.

— Si M(b) s'arrête alors Halt accepte M(b)

— Si M(b) ne s'arrête pas alors Halt n'accepte pas M(b)

On construit la machine paradoxe :

```
paradoxe(M)
  if Halt accepte M(M)
    boucler
  else
    accepter
```

Si on applique paradoxe à lui-même on a 2 cas :

— Si *paradoxe*(*paradoxe*()) se termine en un temps fini, alors Halt accepte *paradoxe*(*paradoxe*) donc il déclenche une boucle infinie. On a donc contradiction.

— Si *paradoxe*(*paradoxe*()) ne se termine pas en un temps fini, alors Halt n'accepte pas *paradoxe*(*paradoxe*) et par définition de paradoxe, il est censé s'arrêter.
On a donc une contradiction.

4 Non déterministe

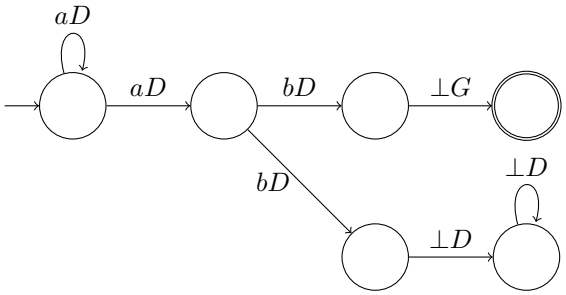
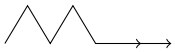


FIGURE 3 – Machine de Turing non déterministe reconnaissant a^*b

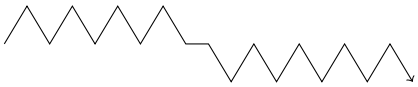
(a) Pas de boucle ab



(b) Sortie de boucle avec fin a^nb

(c) Sortie de boucle sans fin a^n

(d) Première boucle (termine sur $\perp$) a^n



(e) boucle infinie sur la fin a^nb

FIGURE 4 – Comportement possible

On définit alors : $\mathcal{L}(\mathcal{M}) = \{u|$ il existe une exécution acceptante $\}$

Donc la machine de Turing ($\mathcal{M}$) au-dessus reconnait le langage a^*b, car il y a toujours un chemin dans $\mathcal{M}$ si le mot ne contient que des a et fini par un b.

5 Complexité

On définit la complexité d'une machine de Turing de deux manières différente selon si elle est déterministe ou non.

5.1 Déterministe

$C_{temps} : n \mapsto \max_{|u| \leq n} \{$tems d'exécution de $\mathcal{M}$ sur u$\}$

5.2 Non Déterministe

$C_{temps} : n \mapsto \max_{|u| \leq n, u \in \mathcal{L}(\mathcal{M})} (\min\{|e|| \text{ e exécution acceptante de } \mathcal{M}suru\})$

5.3 P et NP

P : X est P ssi il existe $\mathcal{M}$ une machine de Turing déterministe de complexité polynomiale tq $\mathcal{M}(\mathcal{M}) = X$

NP : X est NP ssi il existe $\mathcal{M}$ une machine de Turing non déterministe de complexité polynomiale tq $\mathcal{M}(\mathcal{M}) = X$

6 Plusieurs bandes

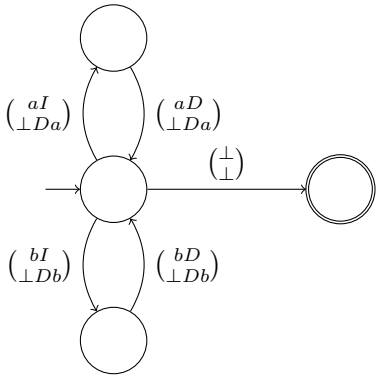


FIGURE 5 – Machine de Turing sur 2 bandes réalisant la fonction *begaye*

1. | $\triangleright$ | a | b | b | a | a | ... |
2. | $\triangleright$ | a | b | b | a | a | ... |

Après avoir exécuté la machine on a :

1. | $\triangleright$ | a | b | b | a | a | ... |
2. | $\triangleright$ | a | a | b | b | b | b | a | a | a | a | ... |

$C_{temps} = \mathcal{O}(n)$ avec n la taille de l'entrée
On remarque qu'il est aussi possible de faire la fonction bégaie sur une seule bande :

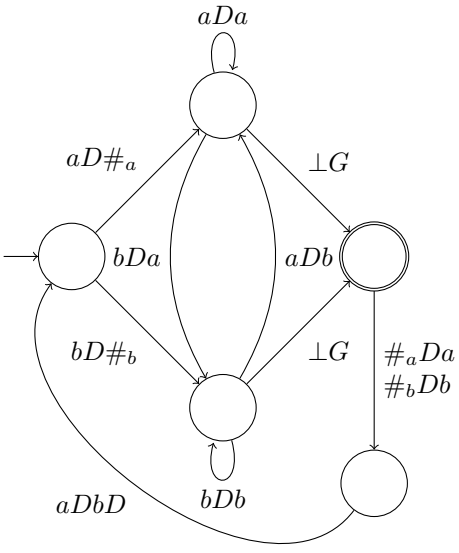


FIGURE 6 – Machine de Turing sur 1 bande réalisant la fonction *begaye*

Proposition 1 *Un calcul est faisable sur 1 bande ⇔ faisable sur k bandes.*

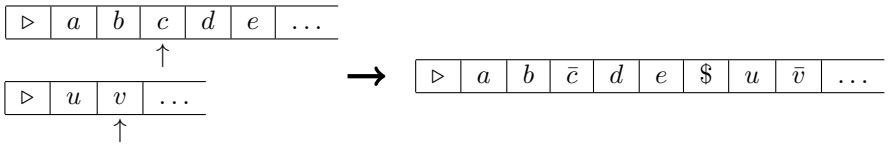
Preuve – Preuve de la proposition

⇒ : triviale (si le calcule est sur une bande on utilise juste $(k-1)$ bande vide)

⇐ : Il y a plusieurs solutions pour simuler k bandes en une bande. (détaillé dans les sections en dessous).

6.1 Bout à Bout

On peut mettre les bandes bout à bout séparer par un \$.
La lettre pointées est surligné pour la reconnaitre.
Par exemple :

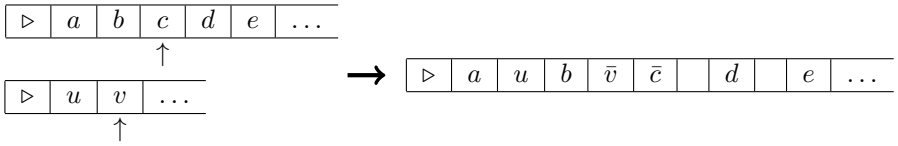


Algo :
— On parcoourt le mot et on retient les lettres lues.
— On en déduit la transition à simuler.
— On revient à gauche et on exécute la transition.
— Chaque fois que le mot déborde, il faut décaler la partie droite d'une case vers la droite. (Cela peut arriver k fois)
Coût :
— Pour 1 pas : $\mathcal{O}(k \times C_{espace,\Sigma})$
— Total :

$$\begin{aligned} &\mathcal{O}(kC_{esapce,\Sigma} \times C_{temps}) + \mathcal{O}(n) \\ &\leq \mathcal{O}(k^2 \times (C_{temps})^2) + \mathcal{O}(n) \qquad\qquad C_{espace,\Sigma} \leq kC_{temps} \end{aligned}$$

6.2 Interclassement

On peut allterner les lettres de chaques bandes.
La lettre pointées est surligné pour la reconnaitre.
Par exemple :

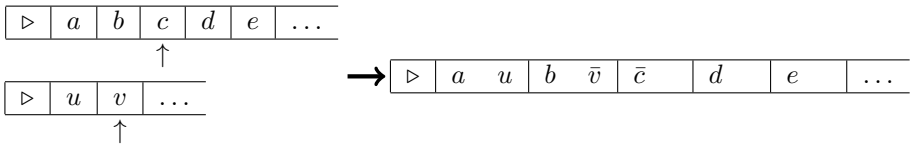


Algo :
— On parcoourt le mot et on retient les lettres lues.
— On en déduit la transition à simuler.
— On revient à gauche et on exécute la transition.
Coût :
— Pour 1 pas : $\mathcal{O}(k \times C_{espace,\Sigma})$
— Total :

$$\begin{aligned} &\mathcal{O}(kC_{esapce,Max} \times C_{temps}) + \mathcal{O}(n^2) \\ &\leq \mathcal{O}(k^2 \times (C_{temps})^2) + \mathcal{O}(n^2) \qquad\qquad C_{espace,Max} \leq kC_{temps} \end{aligned}$$

6.3 Superposition

On supperpose toutes les lettres pour quelles rentrent dans une case.
La lettre pointées est surligné pour la reconnaitre.
Par exemple :

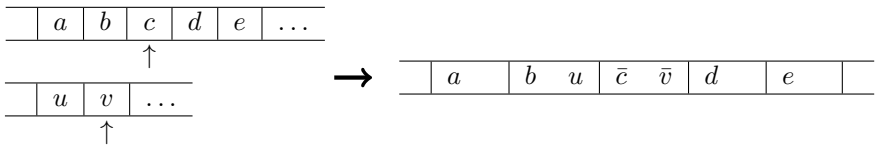


Algo :
— On parcoourt le mot et on retient les lettres lues.
— On en déduit la transition à simuler.
— On revient à gauche et on exécute la transition.
Coût :
— Pour 1 pas : $\mathcal{O}(k \times C_{espace,Max})$
— Total :

$$\begin{aligned} &\mathcal{O}(C_{esapce,Max} \times C_{temps}) \\ &\leq \mathcal{O}((C_{temps})^2) && C_{espace,Max} \leq kC_{temps} \end{aligned}$$

6.4 Superposition bi-infinie

On supperpose toutes les lettres pour quelles rentrent dans une case.
La lettre pointées est surligné pour la reconnaitre.
Par exemple :



Algo :
— On connaît immédiatement la transition à simuler
— On effectue les modifications de lettres
— Pour chaques bandes on effectue le déplacement (droit ou gauche).
Coût :
— Pour 1 pas : $\mathcal{O}(C_{espace,Max})$
— Total :

$$\begin{aligned} &\mathcal{O}(C_{esapce,Max} \times C_{temps}) \\ &\leq \mathcal{O}((C_{temps})^2) && C_{espace,Max} \leq kC_{temps} \end{aligned}$$

6.5 Palindrome