5 Machines de Turing restreintes

Objectif : quelques «simplifications» des MT

5.1 MT sans notion d'état final

Rappels : acceptation $\Leftrightarrow$ arrêt dans un état final
on peut n'avoir qu'un seul état final f, sans transition

Sol. 1 : acceptation $\Leftrightarrow$ arrêt
$\rightarrow$ Ajouter $\delta(q, X) = (q, X, S) \quad \forall q \neq f$ quand $\delta(q, X)$ indéfini
Inconvénient : la notion de langage *récursif* (décidé) disparaît...

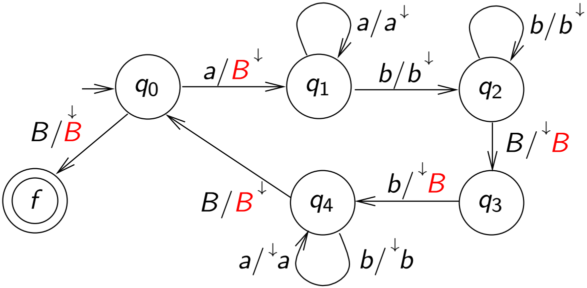
Sol. 2 : acceptation $\Leftrightarrow$ symbole sous la tête $= H$
(H : nouveau symbole)
$\rightarrow$ Ajouter $\delta(f, X) = (f, H, S) \quad \forall X \neq H$
ou
$\rightarrow$ Changer tous les $\delta(q, X) = (f, ..., ...)$ en $\delta(q, X) = (..., H, S)$

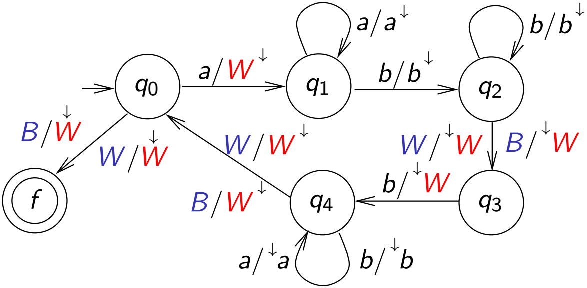
5.2 MT qui n'écrit jamais B

Idée : un nouveau symbole W qui remplace les B écrits

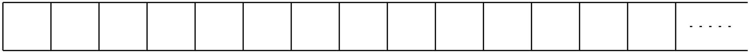
1. Changer $\delta(q, X) = (p, B, M)$ en $\delta(q, X) = (p, W, M) \quad \forall q, X$
2. Ajouter $\delta(q, W) = (p, X, M) \quad \forall \delta(q, B) = (p, X, M)$

Exemple : $\{a^n b^n\}$





5.3 MT à ruban semi-infini



Config : ruban $\alpha X \beta B^\infty$, α et $\beta \in \Gamma^*$, état q, tête $\rightsquigarrow X$: $\boxed{\alpha q X \beta}$

Si $\delta(q, X) = (p, Y, G)$ alors $qX\beta \not\vdash$

(on ne peut pas aller à gauche de la première cellule...)

Une MT à ruban infini peut «évidemment» simuler une MT à ruban semi-infini... $\rightarrow$ insérer à gauche du mot en entrée un nouveau symbole, qui dénote le «début du ruban...»... et pas de transition sur ce symbole...

Plus intéressant : une MT à ruban semi-infini peut *simuler* une MT à ruban infini

Simuler ? passer d'un formalisme (un «style») S à un autre S'

m' dans S' simule m dans $S \Leftrightarrow$

1. $\exists$ un *codage* $\mathcal{C}$ de Σ^* dans Σ'^*, injectif
2. on peut décider pour un $w' \in \Sigma'^*$ si $\exists w \in \Sigma^* : w' = \mathcal{C}(w)$
3. $w \in \Sigma^* \in L(m) \Leftrightarrow \mathcal{C}(w) \in L(m')$

Idée : «replier le ruban»

·····	B	B	B	B	X_1	X_2	···········	X_i	·····	X_n	B	B	·····

X_i	····· X_n	B	B	B	·········
$\star$	X_{i-1} ·····	X_2	X_1	B	·········

$\Gamma' = \Gamma \times (\Gamma \cup \{\star\})$ $\qquad B' = (B, B)$

$\Sigma' = \Sigma \times \{B, \star\}$

$\mathcal{C}(w_1 w_2 ... w_p) = (w_1, \star)(w_2, B)...(w_p, B)$ $\qquad \mathcal{C}(\varepsilon) = (B, \star)$

Config init : pour m : $q_0 w$, pour m' : $q'_0 \mathcal{C}(w)$

w_1	·····	w_p	B	B	B	·········
$\star$	B ··· B	B	B	B	B	·········

$Q' = Q \times \{h, b\}$: en q, et

on s'intéresse au haut du ruban (q^h)

ou au bas du ruban (q^b)

$q'_0 = q^h_0$

La suite n'est que technique...

·····	B	B	B	B	X_1	X_2	··········	X_i	·····	X_n	B	B	·····

X_i	····· X_n	B	B	B	·································
$\star$	X_{i-1} ·····	X_2	X_1	B	·································

Transitions : Convention : $T \in \Gamma \cup \{\star\}, X, Y, Z \in \Gamma$

$$\delta(q, X) = (p, Z, D) \rightsquigarrow \delta'(q^h, (X, T)) = (p^h, (Z, T), D)$$

$$\delta(q, X) = (p, Z, G) \rightsquigarrow \delta'(q^h, (X, Y)) = (p^h, (Z, Y), G)$$

$$\delta'(q^h, (X, \star)) = (p^b, (Z, \star), D)$$

$$\delta(q, Y) = (p, Z, G) \rightsquigarrow \delta'(q^b, (X, Y)) = (p^b, (X, Z), D)$$

$$\delta(q, Y) = (p, Z, D) \rightsquigarrow \delta'(q^b, (X, Y)) = (p^b, (X, Z), G) \qquad [1]$$

$$\forall q, X \; : \; \delta'(q^b, (X, \star)) = (q^h, (X, \star), S) \qquad [2]$$

Note : on ne peut arriver en [2] qu'en venant de [1]...

On peut toujours avoir un ruban «à plusieurs pistes» (ici 2)...

5.4 MT à alphabet réduit

Si $|\Sigma| = k \geq 2$, on peut toujours simuler par $\Sigma' = \{0, 1\}$ = ZOU
en codant chaque symbole de Σ par $\lceil \log_2 k \rceil$ symboles de ZOU
Ex : $\Sigma = \{a, b, c\}$: $a \rightarrow 00$, $b \rightarrow 01$, $c \rightarrow 10$

Si $|\Gamma| = k$, on peut aussi simuler par $\Gamma' = \{0, 1, B\}$ en codant
chaque symbole de $\Gamma - \{B\}$ par $p = \lceil \log_2 k \rceil$ symboles de ZOU
et en prenant $\mathcal{C}(B) = B^p$ (pour garder les «blancs vers l'infini»)
ou même $\mathcal{C}(B) = B$ si m (et donc m') n'écrit jamais B

Restriction plus drastique : $\Gamma' = \{1, B\}$
$\Rightarrow$ deux symboles pour TOUT coder mais autoriser B dans Σ'... :
$a \rightarrow B1$, $b \rightarrow 1B$, $c \rightarrow 11$, $B \rightarrow BB$

Conclusion : Une MT (plus proche du réel) qui...
- n'«efface» jamais tout à fait ce qu'elle voit (5.2)
- a une notion de «mémoire» plus proche de la réalité (5.3)
- travaille avec des 0 et des 1 (5.4)

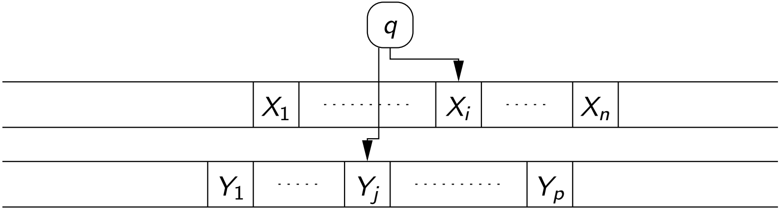
... peut «faire» autant qu'une MT générale...

Note : 5.1 (pas d'état final) pas «plus proche du réel»
mais intéressant d'un point de vue théorique...

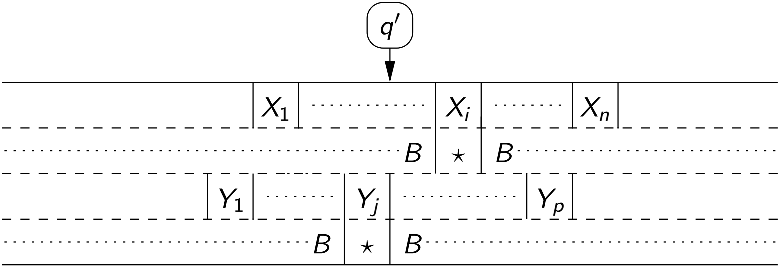
6 Machines de Turing étendues

Objectif : étendre pour voir si on peut gagner en expressivité...

6.1 Plusieurs rubans *Exo* : définir les MT à r rubans



Idée : un ruban, $2 \times r$ pistes... en q' on sait où on est /$\star$...



7 Un langage non récursivement énumérable

7.1 Codage des machines de Turing dans $\text{ZOU}^* = \{0,1\}^*$

Objectif : une MT $\leftrightarrow$ un mot de $\text{ZOU}^* \equiv$ un $n \in \mathbb{N}$

On ne s'intéresse qu'aux MT avec $\Sigma = \{0,1\}$...

- États : $q_1, \ldots q_n$ — q_1 initial, q_2 seul état final
- Symboles : $X_1, \ldots X_p$ — $X_1 : 0, X_2 : 1, X_3 : B$
- Mouvements : G, D, S — $M_1 : G, M_2 : D, M_3 : S$

Transition $\delta(q_i, X_j) = (q_k, X_\ell, M_r)$ codée par $0^i 1 0^j 1 0^k 1 0^\ell 1 0^r 1$

Code de m : séquence des codes des transitions (ordre quelconque)

Exemple : $\delta(q_1, 1) = (q_3, 1, D) \rightarrow$ 010010001001001

$\delta(q_3, 0) = (q_1, 1, G) \rightarrow$ 0001010100101

$\delta(q_3, B) = (q_2, 0, S) \rightarrow$ 0001000100101010001

Code de m : 0100100010010010001010100101000100010010101010001

Codes valides : $((0^+1)^5)^*$ (langage régulier...)

$w \in \text{ZOU}^*$ pas un code valide $\rightarrow m$ sans transition ($L(m) = \emptyset$)

On peut donc parler de «la n-ième MT» $\hat{n}$:

m_n, dont le code est w_n, le n-ième mot de ZOU^* (i.e. $\tilde{n}$)

Note : si on assimile une MT et son code, on a en fait $m_n = w_n$

7.2 Le langage «de diagonalisation» n'est pas RE

Définition :

$L_d = \{w_n \in \mathrm{ZOU}^* \mid w_n \notin L(m_n)\} \qquad (= \{\tilde{n} \in \mathrm{ZOU}^* \mid \tilde{n} \notin L(\hat{n})\})$

Théorème : $L_d \notin \mathrm{RE}$

Preuve par l'absurde : si $L_d \in \mathrm{RE}, \exists k : L(m_k) = L_d$

- $w_k \in L_d \Rightarrow w_k \in L(m_k) \Rightarrow w_k \notin L_d$
- $w_k \notin L_d \Rightarrow w_k \notin L(m_k) \Rightarrow w_k \in L_d$ ■

«Diagonalisation» :

i \ j	0	1	2	3	4	……
0	0	1	0	0	1	……
1	1	1	0	1	0	……
2	1	0	0	1	1	……
3	0	1	0	0	1	……
4	1	0	1	1	1	……
⋮	⋮	⋮	⋮	⋮	⋮	⋱

1 : m_i accepte w_j

0 : m_i n'accepte pas w_j

vect. caract. de $L(m_2)$: 10011...

Diagonale : 01001...

Complémentation : 10110...

$\rightarrow$ vect. caract. de L_d

diffère de chaque ligne i au point i

8 Un langage récursivement énumérable mais non récursif

Objectif :

- R $\neq$ RE
- $\exists L$: L accepté (arrêt dans un état final $\iff w \in L$)
  mais pas décidé (toujours arrêt)
- $\exists L$: on peut toujours savoir que $w \in L$,
  mais on peut ne jamais savoir que $w \notin L$
- $\exists f$, fonction partielle calculable pour laquelle aucun programme qui la réalise s'arrête toujours (e.g. en levant une exception hors de son domaine)

8.1 Préliminaires

Théorème : $L \in \mathrm{R} \Rightarrow \overline{L} \in \mathrm{R}$ ($\overline{L}$: complémentaire de L)

Preuve : $L = L(m)$ avec m qui s'arrête toujours
pour $\overline{m} \stackrel{\text{def}}{=} (m$ sauf $F \leftrightarrow Q - F)$ on a $L(\overline{m}) = \overline{L}$ ■

Théorème : $L \in \mathrm{RE}$ et $\overline{L} \in \mathrm{RE} \Rightarrow L \in \mathrm{R}$ (et $\overline{L} \in \mathrm{R}$ aussi, donc)

Preuve : de m ($L(m) = L$) et $\overline{m}$ ($L(\overline{m}) = \overline{L}$) construire m' :
États : $(q, \overline{q}')$, 2 rubans, transitions «en parallèle»
m ouex $\overline{m}$ accepte w... m accepte : OK ; $\overline{m}$ accepte : KO ■

8.2 Le langage «universel», la MT universelle [Turing 1936]

Définition : $L_u = \{(m, w) \in (\textsc{zou}^*)^2 \mid w \in L(m)\}$

Théorème : $L_u \in$ RE

Preuve : $\exists$ MTU («Machine de Turing Universelle») qui accepte L_u

m : codée selon 7.1 ; (m, w) codé par «$m1w$» (fin de m : 11)

MTU a plusieurs rubans :

- r_1 contient l'entrée $m1w$
- r_2 *encode* le ruban de m, codage de 7.1 : 0→0, 1→00...
  chacun suivi de 1... sauf «B de m»→«B de MTU»...
  init : w (e.g. $w = 1001$: $B^\infty \underline{00}1\underline{0}1\underline{0}1\underline{00}1B^\infty$)
  B remplacé par 0001 au besoin
- r_3 contient l'état courant de m (init : 0)...
- r_4...r_k pour les comptages, recopies, décalages...

MTU *simule* m sur w : MTU *accepte* $(m, w) \iff m$ *accepte* w

MTU : une machine de Turing *interprète* de machines de Turing

MTU $\Leftrightarrow$ «programme» du processeur d'un ordi. élémentaire !

Existence de MTU $\Rightarrow$ existence des ordinateurs !

Pour simuler un pas de m :

1. chercher sur r_1 une transition $0^i 1 0^j 1 0^k 1 0^\ell 1 0^r 1$
   sachant 0^i sur r_3 et $0^j 1$ à partir de la tête de r_2
2. si pas trouvée : s'arrêter (en état final $\Leftrightarrow i = 2$)
3. si trouvée :
   a. changer r_3 pour y mettre 0^k (nouvel état)
   b. sur r_2, remplacer 0^j par 0^ℓ (nouveau symbole)
      avec décalages si $j \neq \ell$
   c. déplacer la tête de r_2 pour se placer sur le bon symbole
      selon r («à gauche», «à droite», «stationnaire») ■

Note : m ne s'arrête pas sur $w \Leftrightarrow$ MTU ne s'arrête pas sur (m, w)

Théorème : $L_u \notin \mathrm{R}$

Preuve : si $L_u \in \mathrm{R}$ alors $\overline{L_u} \in \mathrm{R}$

$$\overline{L_u} = \{(m, w) \in (\text{ZOU}^*)^2 \mid w \notin L(m)\}$$

Supposons qu'on ait $m_{\overline{L_u}}$, une MT pour *décider* $\overline{L_u}$

Alors on pourrait construire une MT m_{L_d} pour *décider* L_d...

Rappel : $L_d = \{w \in \text{ZOU}^* \mid w \notin L(w)\}$... et $L_d \notin \mathrm{RE}$

m_{L_d} sur w : lancer $m_{\overline{L_u}}$ sur (w, w) ; accepter w ssi $m_{\overline{L_u}}$ accepte ! ■