TME5 : Machines de Turing déterministes à une bande

(définitions, exemples, programmation en Python)

– version 2.0 (Janvier 2022) –

Mathieu.Jaume@lip6.fr

1 Machines de Turing déterministes à une bande

1.1 Représentation

En Python, on représente une machine de Turing déterministe à une bande par un quadruplet `M = (d,q0,qok,qko)` où `q0`, `qok` et `qko` représentent respectivement l'état initial, l'état acceptant et l'état rejetant et où `d` est la liste représentant la fonction de transition contenant pour chaque transition $q_1 \xrightarrow{a_1/a_2,x}_M q_2$ un élément de la forme `((q1,a1),(q2,a2,x))`.

Puisque la fonction de transition est définie par une liste, on définit une fonction `assoc_f` qui étant donnés une liste représentant une fonction f et un élément x renvoie $f(x)$ si x appartient au domaine de f et renvoie `None` sinon (on suppose ici que les éléments du domaine de f sont comparables avec l'égalité atomique, ce qui reviendra à supposer que les états d'une machine de Turing sont aussi comparables avec l'égalité atomique).

assoc_f

```
def assoc_f(lf,x):
    """ list[alpha*beta] * alpha -> beta """
    for (xf,yf) in lf:
        if xf == x:
            return yf
    return None
```

Exemple 1 On construit la machine :

$$M_0 = \left(\underbrace{\{q_0, q_1, q_2, q_3\}}_{Q}, \underbrace{\{A, B, a, b\}}_{\Sigma}, \underbrace{\{A, B, a, b, \sqcup\}}_{\Gamma}, \delta, q_0, q_2, q_3 \right)$$

où δ est définie par :

$$\begin{array}{lll} \delta(q_0, A) = (q_1, A, R) & \delta(q_0, a) = (q_3, a, R) & \delta(q_0, b) = (q_3, b, R) \\ \delta(q_0, B) = (q_3, B, R) & \delta(q_1, A) = (q_3, A, R) & \delta(q_1, B) = (q_2, B, R) \\ \delta(q_1, a) = (q_1, b, R) & \delta(q_1, b) = (q_1, a, R) & \end{array}$$

La représentation graphique de M_0 est donnée sur la figure 1.

La liste représentant la fonction de transition de la machine M_0 de l'exemple 1 (représentée sur la figure 1) s'écrit (i désigne l'état q_i) :

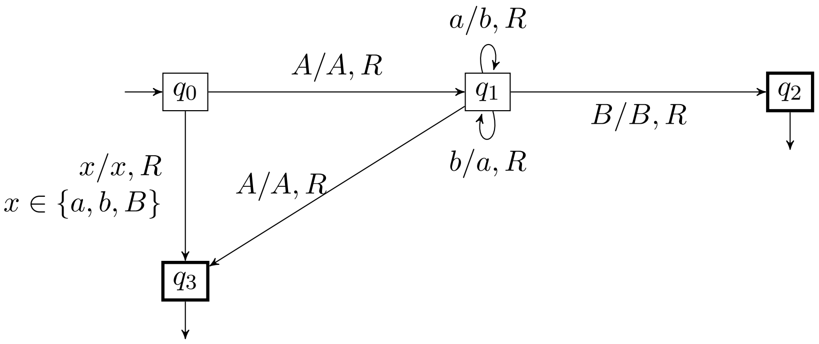

FIGURE 1 – Représentation graphique de la machine de Turing M_0 (exemple 1)

exemple : fonction de transition de la machine M_0

```
l_M_ex1 = [((0,"A"),(1,"A","R")), ((0,"a"),(3,"a","R")), ((0,"b"),(3,"b","R")),
           ((0,"B"),(3,"B","R")), ((1,"A"),(3,"A","R")), ((1,"B"),(2,"B","R")),
           ((1,"a"),(1,"b","R")), ((1,"b"),(1,"a","R"))]
```

machine M_0 de l'exemple 1 (représentée sur la figure 1)

```
M_ex1 =(l_M_ex1,0,2,3)
```

1.2 Exécutions d'une machine de Turing déterministe à une bande

Configurations Une *configuration* d'une machine de Turing M est la donnée de l'état courant de M, de la bande (c-à-d de son contenu) et de la position de la tête de lecture sur la bande. On note $w_1|q|w_2$ la configuration de M lorsque l'état courant de M est $q \in Q$ et que le mot $w_1w_2 \in \Gamma^\star$ est inscrit sur la bande et que la tête de lecture est positionnée sur la première lettre du mot w_2. Etant donnée une bande sur laquelle le mot $w \in \Sigma^\star$ est inscrit, la *configuration initiale* d'une machine de Turing M est $|q_0|w$. Une *configuration acceptante* (resp. *rejetante*) est une configuration de la forme $w_1|q_{ok}|w_2$ (resp. $w_1|q_{ko}|w_2$).

Implantation On définit une fonction `print_config_1` qui permet d'afficher la configuration d'une machine de Turing à partir d'une liste `L` représentant la bande, d'un entier `t` désignant la position de la tête de lecture sur la bande (`t` $\leq$ `len(L)`), de l'état `q` $\in Q$ de la machine et des états q_{ok} et q_{ko} de la machine. Lorsque l'état q_{ok} (resp. q_{ko}) est atteint, la configuration affichée contient `ok` (resp. `ko`) à la place de l'état.

affichage de la configuration d'une machine à une bande

```
def print_config_1(L,t,q,qok,qko):
    for s in L[:t]:
        print(s,end='')
    print("|",end='')
    if q == qok:
        print("ok",end='')
    elif q == qko:
        print("ko",end='')
    else:
        print(q,end='')
    print("|",end='')
    for s in L[t:]:
```

```
        print(s,end='')
    print(" ")
```

exemple : affichage de la configuration d'une machine à une bande

```
>>> print_config_1([1,2,3,4,5,6,7,8,9],0,"q","q2","q3")
|q|123456789
>>> print_config_1([1,2,3,4,5,6,7,8,9],4,"q2","q2","q3")
1234|ok|56789
>>> print_config_1([1,2,3,4,5,6,7,8,9],9,"q","q2","q3")
123456789|q|
```

Exécutions Un étape d'exécution d'une machine de Turing M dans une configuration C produit la configuration C', ce que l'on note $C \rightsquigarrow_M C'$, définie par :

— $C' = u|q'|acv$ si $C = ua|q|bv$ et $\delta(q, b) = (q', c, L)$
Exemple si $\delta(q, l) = (q', c, L)$ alors à partir de la configuration $C = motsur|q|leruban$, on obtient la configuration $C' = motsu|q'|rceruban$:

$$\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c} \hline \cdots & m & o & t & s & u & r & c & e & r & u & b & a & n & \cdots \\ \hline \end{array}$$

$$\underbrace{m\ o\ t\ s\ u}_{w_1}\ \underbrace{r\ c\ e\ r\ u\ b\ a\ n}_{w_2}$$

$$\uparrow$$

$$\boxed{q'}$$

— $C' = uc|q'|v$ si $C = u|q|bv$ et $\delta(q, b) = (q', c, R)$
Exemple si $\delta(q, l) = (q', c, R)$ alors à partir de la configuration $C = motsur|q|leruban$, on obtient la configuration $C' = motsurc|q'|eruban$:

$$\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c} \hline \cdots & m & o & t & s & u & r & c & e & r & u & b & a & n & \cdots \\ \hline \end{array}$$

$$\underbrace{m\ o\ t\ s\ u\ r\ c}_{w_1}\ \underbrace{e\ r\ u\ b\ a\ n}_{w_2}$$

$$\uparrow$$

$$\boxed{q'}$$

Une machine M s'*arrête* sur un mot $w \in \Sigma^\star$, ce que l'on note $M(w) \downarrow$, s'il existe une séquence finie $C_0, C_1, \cdots, C_n$ de configurations, notée $C_0 \rightsquigarrow_M^\star C_n$, telle que :

— $C_0 = |q_0|w$ est la configuration initiale
— pour tout $0 \leq i < n$, $C_i \rightsquigarrow_M C_{i+1}$
— C_n est une configuration acceptante ou rejetante $M(w) \downarrow^{ko}$

L'exécution d'une machine de Turing M sur un mot $w \in \Sigma^\star$ peut conduire à quatre situations distinctes :

— la machine s'arrête dans une configuration acceptante, ce que l'on note $M(w) \downarrow^{ok}$
— la machine s'arrête dans une configuration rejetante, ce que l'on note $M(w) \downarrow^{ko}$
— la machine atteint une configuration à partir de laquelle aucune configuration n'est accessible
— la machine « boucle à l'infini », ce que l'on note $M(w) \uparrow$

Dans toute le suite, pour alléger l'écriture de δ, et pour simplifier la description des exécutions possibles d'une machine de Turing, lorsque $\delta(q, b)$ n'est pas défini, on transforme C en une configuration rejetante. Ainsi, dans les représentations des machines de Turing qui suivent l'absence d'une transition à partir d'un état $q \in Q$ et d'un symbole $x \in \Gamma$ signifie que la machine de Turing considérée dispose par défaut de la transition $q \xrightarrow{a/a,R} q_{ko}$. Avec cette convention, étant donné un mot $w \in \Sigma^\star$, trois situations sont donc possibles : $M(w) \downarrow^{ok}$, $M(w) \downarrow^{ko}$ et $M(w) \uparrow$.

Implantation On définit une fonction `exec_MT_1` qui permet d'exécuter une machine de Turing M (déterministe à une bande) en affichant les configurations successives de M durant cette exécution. Les arguments de cette fonction sont une machine de Turing `M`, une liste `L` représentant la bande initiale et un entier `i0` désignant la position initiale de la tête de lecture sur la bande (qui correspond à un indice de la liste `L`). Cette fonction retourne un triplet (b, i_F, L) où b est un booléen indiquant si le calcul a réussi (c-à-d si le calcul se termine sur l'état acceptant q_{ok}), i_F est la position de la tête de lecture à la fin du calcul et L est la liste représentant la bande à l'issue du calcul. Enfin, puisque la bande est supposée de longueur infinie mais que la liste `L` est finie, cette fonction ajoutera les éléments dans `L` (à gauche ou à droite) qui sont nécessaires au calcul (chaque élément ajouté sera initialisé avec le symbole $\sqcup$, codé par `'Z'` dans l'implantation). Bien sûr si le calcul effectué par la machine de Turing M ne termine pas avec le bande fournie, l'exécution de la fonction `exec_MT_1` ne termine pas non plus.

à compléter : exécution d'une machine déterministe à une bande

```
def exec_MT_1(M,L,i0):
    # M : machine de Turing deterministe a 1 bande
    # L : liste representant la bande initiale
    # i0 : position initiale de la tete de lecture
```

exemple : exécution de la machine M_0 l'exemple 1

```
>>> exec_MT_1(M_ex1,["A","a","b","a","a","B"],0)
|0|AabaaB
A|1|abaaB
Ab|1|baaB
Aba|1|aaB
Abab|1|aB
Ababb|1|B
AbabbB|ok|Z
(True, 6, ['A', 'b', 'a', 'b', 'b', 'B', 'Z'])

>>> exec_MT_1(M_ex1,["B","a","b","a","a","B"],0)
|0|BabaaB
B|ko|abaaB
(False, 1, ['B', 'a', 'b', 'a', 'a', 'B'])
```

Dans la suite, seul le résultat de l'exécution de la fonction `exec_MT_1` sera donné dans les exemples (sans faire figurer les affichages produits).

Exemple 2 $(L_1 = \{w \in \{a,b\}^\star \mid |w|_a = |w|_b\} \in \mathbf{R})$
Le langage L_1 est récursif et il existe donc une machine de Turing M_1 telle que $\mathcal{L}(M_1) = L_1$.

à compléter : machine M_1 telle que $\mathcal{L}(M_1) = \{w \in \{a,b\}^\star \mid |w|_a = |w|_b\}$

```
l_ex2 = ...
M_ex2 =(l_ex2,...)
```

Plusieurs machines de Turing peuvent accepter le langage L_1. Le contenu de la bande et la position de la tête de lecture à l'issue du calcul peuvent donc varier selon l'approche adoptée, mais toutes les versions possibles de M_1 doivent produire le même booléen (indiquant si le calcul a réussi).

exemples d'exécution d'une machine M_1 telle que $\mathcal{L}(M_1) = \{w \in \{a,b\}^\star \mid |w|_a = |w|_b\}$

```
>>> exec_MT_1(M_ex2,["Z"],0)
(True, 1, ['Z', 'Z'])
>>> exec_MT_1(M_ex2,["a","b","a","b","a","a","Z"],0)
(False, 6, ['X', 'Y', 'X', 'Y', 'X', 'a', 'Z'])
>>> exec_MT_1(M_ex2,["a","b","b","a","b","a","Z"],0)
(True, 7, ['X', 'Y', 'X', 'Y', 'X', 'Y', 'Z', 'Z'])
>>> exec_MT_1(M_ex2,["b","b","a","b","a","a","Z"],0)
(True, 7, ['X', 'X', 'Y', 'X', 'Y', 'Y', 'Z', 'Z'])
>>> exec_MT_1(M_ex2,["b","b","a","Z"],0)
(False, 3, ['X', 'X', 'Y', 'Z'])
```

Exemple 3 ($L_{isneg} = \{w \in \{0,1\}^\star \mid w$ se termine par un $1\} \in \mathbf{R}$)
Le langage L_{isneg} est récursif et contient les représentations binaires en complément à 2 des entiers relatifs négatifs, lorsque l'on fait figurer les bits de poids faibles à gauche [1]. On définit une machine M_{isneg} telle que $M_{isneg}(w) \downarrow^{ok}$ si $w \in L_{isneg}$ et $M_{isneg}(w) \downarrow^{ko}$ si $w \notin L_{isneg}$. On souhaite que cette machine ne modifie pas le contenu de la bande et replace la tête de lecture sur le premier symbole de w à l'issue du calcul.

à compléter : machine M_{isneg}

```
d_isneg = ...
M_isneg =(d_isneg,...)
```

exemples d'exécution de la machine M_{isneg}

```
>>> exec_MT_1(M_isneg,["1","0","1","0","0","Z"],0)
(False, 1, ['Z', '1', '0', '1', '0', '0', 'Z'])
>>> exec_MT_1(M_isneg,["0","0","1","0","1","Z"],0)
(True, 1, ['Z', '0', '0', '1', '0', '1', 'Z'])
```

1. Rappelons qu'un mot binaire de n bits (les bits de poids forts figurant à gauche) $a_{n-1}\cdots a_0$ représente l'entier relatif $-a_{n-1}2^{n-1} + \sum_{i=0}^{i=n-2} a_i 2^i$ en adoptant le codage en complément à 2. Un mot binaire représente donc un entier relatif négatif lorsque son bit de poids fort est 1.