TME 5, 6 et 7 : Machines de Turing déterministes

(définitions, exemples, programmation en Python)

– version 2.1 (Janvier 2023) –

Mathieu.Jaume@lip6.fr

Le code présenté dans cette section se trouve dans le fichier `Turing.py`. Dans tous les TME sur les machines de Turing, *contrairement au cours et aux TD*, on considère que la bande d'une machine de Turing est infinie à gauche et à droite. Dans les implantations de machines de Turing, le symbole $\sqcup$ est représenté par le caractère `'Z'`.

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 :

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

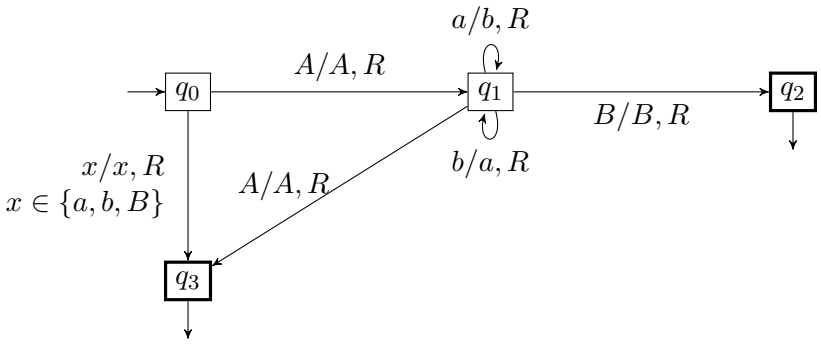


FIGURE 1 – Représentation graphique de la machine de Turing M_0 (exemple 1)

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) :

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='')
```

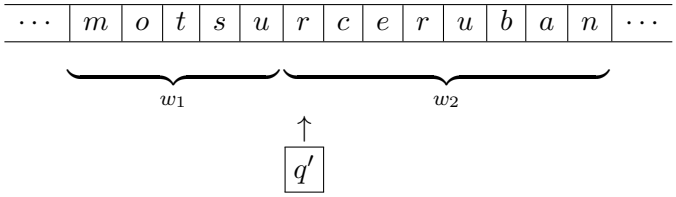
```
    else:
        print(q,end='')
    print("|",end='')
    for s in L[t:]:
        print(s,end='')
    print(" ")
```

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

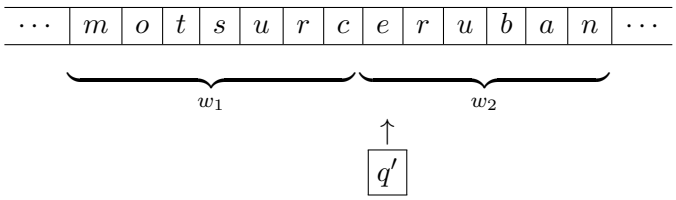
```
>>> print_config_1([1,2,3,4,5,6,7,8,9],0,"q","q2","q3")
|q|123456789
>>> print_config_1([1,2,3,4,5,6,7,8,9],4,"q2","q2","q3")
1234|ok|56789
>>> print_config_1([1,2,3,4,5,6,7,8,9],9,"q","q2","q3")
123456789|q|
```

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 $C' = motsu|q'|rceruban$:



— $C' = uc|q'|v$ 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 $C' = motsurc|q'|eruban$:



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'])
```

2 Composition de machines de Turing

Pour construire une machine de Turing M effectuant un certain calcul, il peut être utile de décomposer ce calcul en sous-calculs, de définir les machines de Turing effectuant chacun de ces sous-calculs et de les assembler afin d'obtenir la machine M. Nous présentons dans cette section les compositions qui correspondent à trois constructions de base des langages de programmation.

2.1 Séquence

Soit $M_1 = \left(Q_1, \Sigma, \Gamma, \delta_1, q_0^1, q_{ok}^1, q_{ko}^1\right)$ et $M_2 = \left(Q_2, \Sigma, \Gamma, \delta_2, q_0^2, q_{ok}^2, q_{ko}^2\right)$ deux machines de Turing déterministes à une bande. On souhaite exécuter la machine M_1 à partir d'une bande L sur laquelle la tête de lecture est à la position i_1, puis exécuter la machine M_2 à partir de la bande issue du calcul de M_1 (avec la tête de lecture positionnée par la machine M_1 en fin de calcul). Si le calcul de M_1 échoue alors M_2 n'est pas exécutée et le calcul de la séquence échoue également.

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.

Puisque l'on dispose d'un simulateur d'exécution d'une machine de Turing qui renvoie la bande finale et la position finale de la tête de lecture sur la bande, il est facile de définir une fonction simulant l'exécution en séquence de deux machines de Turing :

simulation de l'exécution de 2 machines en séquence

```
def exec_seq_MT_1(M1,M2,L,i1):
    (b,i2,L2)=exec_MT_1(M1,L,i1)
    if b:
        return exec_MT_1(M2,L2,i2)
    else:
        return (b,i2,L2)
```

Il est aussi possible de construire à partir des machines M_1 et M_2 une nouvelle machine de Turing M exécutant le calcul correspondant au calcul en séquence de M_1 et M_2. Le principe de cette construction est illustré sur la figure 2 et la machine M est formellement définie par :

$$M = \left(Q_1 \uplus Q_2, \Sigma, \Gamma, \delta, q_0^1, q_{ok}^2, q_{ko}^2\right)$$

avec :

$$\begin{aligned}\delta = \delta_2 \quad &\cup \quad \left\{ q_1 \xrightarrow{a_1/a_2,d}_M q_2 \mid q_1 \xrightarrow{a_1/a_2,d}_{M_1} q_2 \text{ et } q_2 \neq q_{ok}^1 \text{ et } q_2 \neq q_{ko}^1 \right\} \\ &\cup \quad \left\{ q_1 \xrightarrow{a_1/a_2,d}_M q_0^2 \mid q_1 \xrightarrow{a_1/a_2,d}_{M_1} q_{ok}^1 \right\} \\ &\cup \quad \left\{ q_1 \xrightarrow{a_1/a_2,d}_M q_{ko}^2 \mid q_1 \xrightarrow{a_1/a_2,d}_{M_1} q_{ko}^1 \right\}\end{aligned}$$

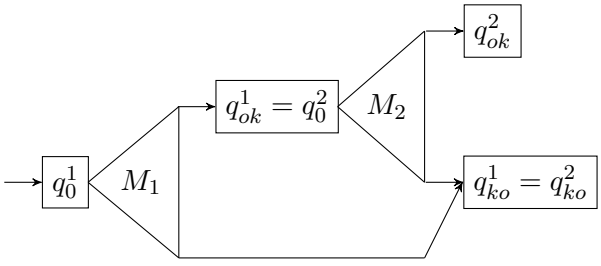


FIGURE 2 – Composition de machine de Turing : séquence

Implantation Les états de la machine construite devront être représentés par des paires de la forme $(1, q_i)$ pour les états $q_i \in Q_1$ et $(2, q_i)$ pour les états $q_i \in Q_2$.

à compléter : composition en séquence de 2 machines

```
def make_seq_MT(M1,M2):
    # M1,M2 : machines de Turing deterministes a 1 bande
```

Exemple 4 (Opposé d'un entier représenté en complément à deux)
Soit w le mot binaire représentant le codage en complément à 2 d'un entier relatif k. Pour calculer la représentation binaire de $-k$ (en complément à 2) il suffit de calculer le complément $\overline{w}$ de w puis d'effectuer l'addition binaire $\overline{w} + 1$ [2]. Il s'agit donc ici d'exécuter en séquence les machines

2. Rappelons qu'un mot binaire ne contenant que des symboles 1 représente en complément à 2 l'entier relatif -1. En remarquant que pour tout mot binaire w la somme booléenne $w + \overline{w}$ produit toujours un mot binaire ne contenant que des symboles 1, en notant k_w (resp. $k_{\overline{w}}$) l'entier relatif représentant w (resp. $\overline{w}$) on a donc $k_w + k_{\overline{w}} = -1$ et on obtient finalement $-k_w = k_{\overline{w}} + 1$.

de Turing M_{comp} et M_{succ} définies en annexe. On note M_{opp} la machine obtenue. Considérons par exemple l'entier relatif 5 dont le codage en complément à 2 est le mot binaire $w = 0101$. Le complément de w est $\overline{w} = 1010$ et le successeur de $\overline{w}$ est alors 1011 qui correspond au codage en complément à 2 de l'entier relatif -5. En faisant figurer les bits de poids faibles à gauche sur les bandes des machines de Turing utilisées, on peut retrouver ce résultat en exécutant le simulateur implanté par la fonction `exec_seq_MT_1` :

exemples : calcul de -5 et de -(-5)

```
>>> exec_seq_MT_1(M_compl_bin,M_succ_bin,["1","0","1","0"],0)
(True, 1, ['Z', '1', '1', '0', '1', 'Z'])
>>> exec_MT_1(M_opp_int_bin,["1","1","0","1"],0)
(True, 1, ['Z', '1', '0', '1', '0', 'Z'])
```

On peut également obtenir ce résultat en construisant la machine M_{opp} et en l'exécutant.

à compléter : construction de la machine M_{opp}

```
M_opp_int_bin = ...
```

exemples d'exécution de la machine M_{opp}

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

2.2 Conditionnelle

On souhaite construire une machine de Turing M dont l'exécution correspond à l'exécution de la conditionnelle «if C then P_1 else P_2» à partir des trois machines de Turing M_C, M_1 et M_2 telles que :

- $M_C = \left(Q_C, \Sigma, \Gamma, \delta_C, q_0^C, q_{ok}^C, q_{ko}^C\right)$ est une machine de Turing telle que $M_C(w) \downarrow^{ok}$ lorsque w vérifie la propriété C et $M_C(w) \downarrow^{ko}$ lorsque w ne vérifie pas la propriété C
- $M_i = \left(Q_i, \Sigma, \Gamma, \delta_i, q_0^i, q_{ok}^i, q_{ko}^i\right)$ (pour $i \in \{1, 2\}$) est une machine effectuant le calcul P_i

Puisque l'on dispose d'un simulateur d'exécution d'une machine de Turing qui renvoie la bande finale et la position finale de la tête de lecture sur la bande, il est facile de définir une fonction simulant l'exécution de la conditionnelle «if C then P_1 else P_2» à partir des exécutions des trois machines de Turing M_C, M_1 et M_2. Les machines M_1 et M_2 s'exécutent à partir de la bande et de la position de la tête de lecture sur cette bande issues du calcul effectué par la machine M_C.

simulation de l'exécution d'une conditionnelle à partir de 3 machines

```
def exec_cond_MT_1(MC,M1,M2,L,i0):
    (bc,ic,Lc)=exec_MT_1(MC,L,i0)
    if bc:
        return exec_MT_1(M1,Lc,ic)
    else:
        return exec_MT_1(M2,Lc,ic)
```

Il est aussi possible de construire à partir des machines M_C, M_1 et M_2 une nouvelle machine de Turing M exécutant la conditionnelle «if C then P_1 else P_2». Le principe de cette construction est illustré sur la figure 3 et la machine M est formellement définie par :

$$M = \left(Q_C \uplus Q_1 \uplus Q_2, \Sigma, \Gamma, \delta, q_0^C, q_{ok}^2, q_{ko}^2\right)$$

avec :

$$\begin{aligned}
\delta = \delta_2 \quad & \cup \left\{ q_1 \xrightarrow{a_1/a_2,d}_M q_2 \mid q_1 \xrightarrow{a_1/a_2,d}_{M_C} q_2 \text{ et } q_2 \neq q^C_{ok} \text{ et } q_2 \neq q^C_{ko} \right\} \\
& \cup \left\{ q_1 \xrightarrow{a_1/a_2,d}_M q^1_0 \mid q_1 \xrightarrow{a_1/a_2,d}_{M_C} q^C_{ok} \right\} \\
& \cup \left\{ q_1 \xrightarrow{a_1/a_2,d}_M q^2_0 \mid q_1 \xrightarrow{a_1/a_2,d}_{M_C} q^C_{ko} \right\} \\
& \cup \left\{ q_1 \xrightarrow{a_1/a_2,d}_M q_2 \mid q_1 \xrightarrow{a_1/a_2,d}_{M_1} q_2 \text{ et } q_2 \neq q^1_{ok} \text{ et } q_2 \neq q^1_{ko} \right\} \\
& \cup \left\{ q_1 \xrightarrow{a_1/a_2,d}_M q^2_{ok} \mid q_1 \xrightarrow{a_1/a_2,d}_{M_1} q^1_{ok} \right\} \\
& \cup \left\{ q_1 \xrightarrow{a_1/a_2,d}_M q^2_{ko} \mid q_1 \xrightarrow{a_1/a_2,d}_{M_1} q^1_{ko} \right\}
\end{aligned}$$

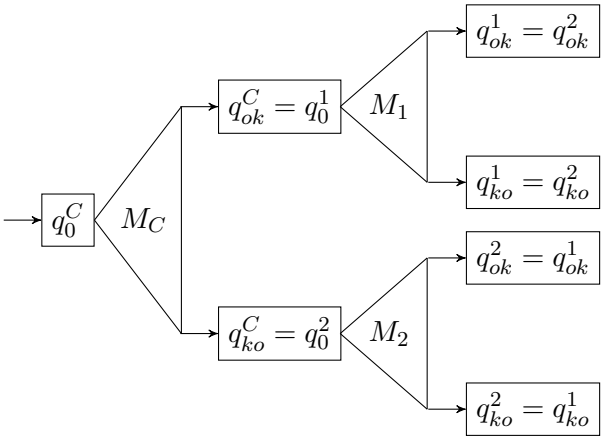


FIGURE 3 – Composition de machine de Turing : conditionnelle

Implantation Les états de la machine construite devront être représentés par des paires de la forme $(0, q_i)$ pour les états $q_i \in Q_C$, $(1, q_i)$ pour les états $q_i \in Q_1$ et $(2, q_i)$ pour les états $q_i \in Q_2$.

à compléter : conditionnelle par composition de machines

```
def make_cond_MT(MC,M1,M2):
    # MC, M1, M2 : machines de Turing deterministes a 1 bande
```

Exemple 5 (Valeur absolue d'un entier représenté en complément à deux)
La machine de Turing M_{abs} permettant de calculer la valeur absolue d'un entier relatif à partir de sa représentation binaire en complément à 2 avec les bits de poids faibles à gauche peut être construite à partir de la machine M_{isneg} de l'exemple 3, de la machine M_{opp} et de la machine M_{id} définie en annexe. Par exemple, en utilisant le simulateur `exec_cond_MT_1` on peut calculer la valeur absolue des entiers relatifs 5 et -5 comme suit.

exemples : calcul de $|5|$ et $|-5|$

```
>>> exec_cond_MT_1(M_isneg,M_opp_int_bin,M_id,["1","0","1","0"],0)
(True, 1, ['Z', '1', '0', '1', '0', 'Z'])
>>> exec_cond_MT_1(M_isneg,M_opp_int_bin,M_id,["1","1","0","1"],0)
(True, 1, ['Z', '1', '0', '1', '0', 'Z'])
```

On peut également obtenir ces résultats en construisant la machine M_{abs} et en l'exécutant.

à compléter : construction de la machine M_{abs}

```
M_abs = ...
```

exemples d'exécution de la machine M_{abs}

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

2.3 Boucle

On souhaite construire une machine de Turing M dont l'exécution correspond à l'exécution de la boucle «while C do P» à partir des deux machines de Turing M_C, M_P telles que :

— $M_C = \left(Q_C, \Sigma, \Gamma, \delta_C, q_0^C, q_{ok}^C, q_{ko}^C\right)$ est une machine de Turing telle que $M_C(w) \downarrow^{ok}$ lorsque w vérifie la propriété C et $M_C(w) \downarrow^{ko}$ lorsque w ne vérifie pas la propriété C

— $M_P = \left(Q_P, \Sigma, \Gamma, \delta_P, q_0^P, q_{ok}^P, q_{ko}^P\right)$est une machine effectuant le calcul P

Puisque l'on dispose d'un simulateur d'exécution d'une machine de Turing qui renvoie la bande finale et la position finale de la tête de lecture sur la bande, il est facile de définir une fonction simulant l'exécution de la boucle «while C do P» à partir des exécutions des deux machines de Turing M_C et M_P. La machine M_P s'exécute à partir de la bande et de la position de la tête de lecture sur cette bande issues du calcul effectué par la machine M_C.

simulation de l'exécution d'une boucle à partir de 2 machines

```
def exec_loop_MT_1(MC,M,L,i0):
    (bc,ic,Lc)=exec_MT_1(MC,L,i0)
    if bc:
        (bM,iM,LM) = exec_MT_1(M,Lc,ic)
        if bM:
            return exec_loop_MT_1(MC,M,LM,iM)
        else:
            return (False,iM,LM)
    else:
        return (True,ic,Lc)
```

Il est aussi possible de construire à partir des machines M_C et M_P une nouvelle machine de Turing M exécutant la boucle «while C do P». Le principe de cette construction est illustré sur la figure 4 et la machine M est formellement définie par :

$$M = \left(Q_C \uplus Q_1 \uplus Q_2, \Sigma, \Gamma, \delta, q_0^C, q_{ok}^2, q_{ko}^2\right)$$

avec :

$$\begin{aligned}
\delta = & \left\{ q_1 \xrightarrow{a_1/a_2,d}_M q_2 \mid q_1 \xrightarrow{a_1/a_2,d}_{M_C} q_2 \text{ et } q_2 \neq q_{ok}^C \right\} \\
\cup & \left\{ q_1 \xrightarrow{a_1/a_2,d}_M q_0^P \mid q_1 \xrightarrow{a_1/a_2,d}_{M_C} q_{ok}^C \right\} \\
\cup & \left\{ q_1 \xrightarrow{a_1/a_2,d}_M q_2 \mid q_1 \xrightarrow{a_1/a_2,d}_{M_P} q_2 \text{ et } q_2 \neq q_{ok}^P \right\} \\
\cup & \left\{ q_1 \xrightarrow{a_1/a_2,d}_M q_0^C \mid q_1 \xrightarrow{a_1/a_2,d}_{M_P} q_{ok}^P \right\}
\end{aligned}$$

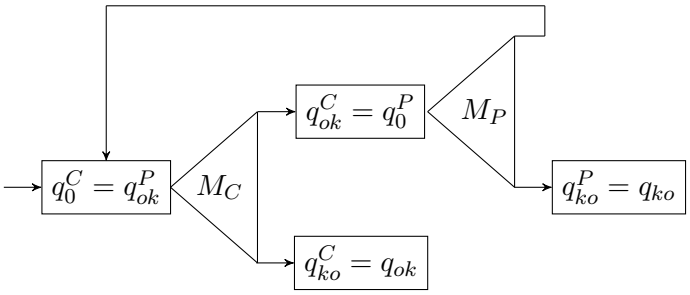


FIGURE 4 – Composition de machine de Turing : boucle

Implantation Les états de la machine construite devront être représentés par des paires de la forme $(0, q_i)$ pour les états $q_i \in Q_C$ et $(1, q_i)$ pour les états $q_i \in Q_P$.

à compléter : boucle par composition de machines

```
def make_loop_MT(MC,M):
    # MC,M : machines de Turing deterministes a 1 bande
```

Exemple 6 (Positionnement de la tête de lecture sur le premier symbole 1)
On souhaite définir une machine de Turing M_{foo1} qui étant donnée une bande sur l'alphabet $\Gamma = \{0, 1, \sqcup\}$ positionne la tête de lecture sur le premier symbole 1 du mot en entrée s'il existe et sur le symbole $\sqcup$ de fin du mot sinon. Le comportement de cette machine peut être spécifié par la construction :

«while caractère lu = 0 do déplacer la tête de lecture à droite»

La machine $M_{=0}$ permettant de tester si le caractère lu est le symbole 0 et la machine M_{right} permettant de déplacer la tête de lecture d'une position vers la droite sont définies en annexe. La machine M_{foo1} peut alors être directement construite comme suit.

à compléter : construction de la machine M_{foo1}

```
M_foo1 = ...
```

exemples d'exécution de la machine M_{foo1}

```
>>> exec_MT_1(M_foo1,["1","0","1","0"],0)
(True, 0, ['1', '0', '1', '0'])
>>> exec_MT_1(M_foo1,["0","0","1","0","1"],0)
(True, 2, ['0', '0', '1', '0', '1'])
>>> exec_MT_1(M_foo1,["0","0","1"],0)
(True, 2, ['0', '0', '1', 'Z'])
>>> exec_MT_1(M_foo1,["0","0","0"],0)
(True, 3, ['0', '0', '0', 'Z', 'Z'])
```

Exemple 7 (Opposé d'un entier représenté en complément à deux)
La machine M_{opp} définie dans l'exemple 4 procède en deux temps : le parcours du mot w pour calculer $\overline{w}$ puis l'addition binaire de $\overline{w}$ avec 1. Il est possible d'effectuer ces deux opérations de manière différente. En effet, en utilisant les machines de Turing définies dans les exemples précédents et en annexe, on peut vérifier que ce calcul peut s'effectuer comme suit :

$$
\left.\begin{array}{l}
\left.\begin{array}{l}
\left.\begin{array}{l}
\text{if } M_{isneg}(w) \downarrow^{ok} \\
\quad \text{then } M_{propag}(w) \\
\quad \text{else } M_{id}(w)
\end{array}\right\} M_{foo0} \\
; \quad \text{while } M_{=0}(w) \downarrow^{ok} \\
\quad \text{do } M_{right}(w)
\end{array}\right\} M_{foo1} \\
\text{if } M_{=\sqcup}(w) \downarrow^{ok} \\
\quad \text{then } M_{id}(w) \\
; \quad \text{else } \underbrace{M_{right}(w); M_{comp}(w)}_{M_{foo3}}
\end{array}\right\} M_{foo4}
\left.\begin{array}{l}
\\ \\ \\ \\ \\ \\ \\
\end{array}\right\} M_{foo2}
\left.\begin{array}{l}
\\ \\ \\ \\ \\ \\ \\
\end{array}\right\} M_{foo}
$$

La machine M_{foo} permettant d'effectuer ce procédé de calcul peut être obtenue en construisant les machines suivantes et en les composant.

à compléter : construction des machines M_{foo0}, M_{foo2}, M_{foo3}, $M_{=\sqcup}$, M_{foo4}, M_{foo}

```
M_foo0 = ...
M_foo2 = ...
M_foo3 = ...
M_eq_Z = ...
M_foo4 = ...
M_foo = ...
```

exemples d'exécution de la machine M_{foo}

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

3 Machines de Turing déterministes multi-bandes

Pour faciliter la conception de machines de Turing, il est possible de considérer des machines disposant de plusieurs bandes. Toutefois, cette possibilité n'augmente pas le pouvoir d'expression des machines de Turing à une bande [3]. Une *machine de Turing déterministe à k bandes* M est une machine munie de k bandes sur lesquelles se déplacent k têtes de lecture/écriture. La fonction de transition porte donc sur un k-uplet de symboles de Γ :

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$$

où S (pour *Stay*) indique que la tête de lecture/écriture de la bande concernée ne se déplace pas. Ainsi $\delta(q, (a_1, \cdots, a_k)) = (q', (a'_1, \cdots, a'_k), (d_1, \cdots, d_k))$ exprime que dans l'état q, si chaque tête de lecture i pointe sur le symbole a_i, alors ce symbole est remplacé par le symbole a'_i (sur la i-ième bande), la tête de lecture i effectue le déplacement d_i et la machine passe dans l'état q'. On note :

$$q \xrightarrow{(a_1,\cdots,a_n)/(a'_1,\cdots,a'_n),(d_1,\cdots,d_n)}_M q'$$

cette transition.

3. Toute machine de Turing multi-bandes M peut être simulée par une machine de Turing M' à une seule bande telle que si M s'arrête sur toutes ses entrées, alors M' s'arrête aussi sur toutes ses entrées.

Implantation Le principe de l'implantation d'une machine de Turing déterministe à k bandes est le même que celui d'une machine à une bande. Il s'agit d'un tuple `M = (d,q0,qok,qko)` où `d` est une liste représentant la fonction de transition. Dans cette liste, chaque transition est exprimée par un élement de la forme `((q,(a1,...,an)),(q',(a'1,...,a'n),(d1,...,dn)))`. Une configuration d'une machine à k bandes est la donnée de l'état dans lequel se trouve la machine, du contenu des k bandes et des positions des k têtes de lecture sur ces bandes. Afficher une telle configuration revient en fait à afficher k configurations d'une machine à une bande. La fonction suivante permet cet affichage (`L` est la liste des bandes, c'est donc une liste de listes et `T` est la liste des positions des têtes de lecture : `T[i]` est la position de la tête de lecture sur la i-ième bande `L[i]`, le caractère lu sur cette bande est donc `L[i][T[i]]`).

affichage de la configuration d'une machine à k bandes

```
def print_config_k(L,T,q,qok,qko,k):
    for i in range(k):
        print_config_1(L[i],T[i],q,qok,qko)
```

La fonction qui simule l'exécution d'une machine à k bandes est similaire à celle qui simule l'exécution d'une machine à une bande : à chaque étape il faut effectuer les actions sur chacune des k bandes et il est désormais possible pour une tête de lecture de rester à la même position durant une étape.

à compléter : exécution d'une machine déterministe à k bandes

```
def exec_MT_k(M,k,L,T):
    # M : machine de Turing deterministe a k bandes
    # k : nombre de bandes
    # L : liste des representations des bandes initiales
    # T : positions initiales des k tetes de lecture
```

Exemple 8 ($L_1 = \{w \in \{a,b\}^\star \mid |w|_a = |w|_b\}$)
Considérons à nouveau le langage L_1 de l'exemple 2. On souhaite construire une machine M_1^2 à deux bandes permettant d'accepter ce langage.
Le principe de la machine de Turing M_1^2 à deux bandes qui accepte ce langage est simple :

- la première bande est la bande sur laquelle figure le mot en entrée, cette bande n'est jamais modifiée, elle est seulement parcourue durant le calcul
- la deuxième bande sert à compter la différence qui existe entre le nombre de symboles a déjà rencontrés et le nombre de symboles b déjà rencontrés ; dans l'état initial on positionne un symbole spécial X sur cette bande

Lorsque la machine a rencontré plus de symboles a que de symboles b, elle se trouve dans l'état q_1 et dans cet état, à chaque nouveau symbole a rencontré la tête de lecture de la deuxième bande est déplacée vers la droite. Dans cet état q_1, si le caractère lu est le symbole b alors la tête de lecture est déplacée vers la gauche si elle ne pointe pas le symbole X, sinon elle est déplacée vers la droite et la machine passe dans l'état q_2 qui correspond à l'état de la machine lorsque plus de symboles b ont été parcourus que de symboles a. Le comportement de la machine dans l'état q_2 est symétrique à celui de q_1. A la fin du mot, la tête de lecture de la deuxième bande pointe sur le symbole X si et seulement si le mot en entrée (sur la première bande) contient autant de symboles a que de symboles b. Cette machine est représentée sur la figure 5.

machine M_1^2 à 2 bandes telle que $\mathcal{L}(M_1^2) = \{w \in \{a,b\}^\star \mid |w|_a = |w|_b\}$ ($q_{ok} = 3$ et $q_{ko} = 4$)

```
d2_ex1 = [((0,("a","Z")),(1,("a","X"),("R","R"))),
          ((0,("b","Z")),(2,("b","X"),("R","R"))),
          ((0,("Z","Z")),(3,("Z","Z"),("S","S"))),
```

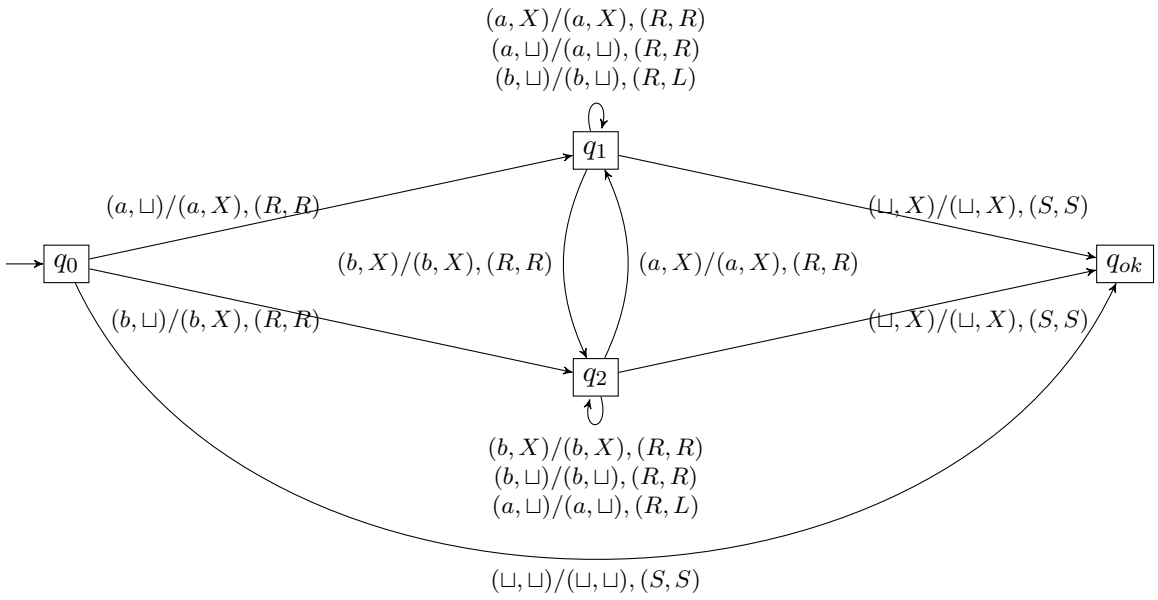


FIGURE 5 – Machine de Turing M_1^2 de l'exemple 8

```
          ((1,("a","X")),(1,("a","X"),("R","R"))),
          ((1,("a","Z")),(1,("a","Z"),("R","R"))),
          ((1,("b","Z")),(1,("b","Z"),("R","L"))),
          ((1,("b","X")),(2,("b","X"),("R","R"))),
          ((1,("Z","X")),(3,("Z","X"),("S","S"))),
          ((2,("a","X")),(1,("a","X"),("R","R"))),
          ((2,("a","Z")),(2,("a","Z"),("R","L"))),
          ((2,("b","Z")),(2,("b","Z"),("R","R"))),
          ((2,("b","X")),(2,("b","X"),("R","R"))),
          ((2,("Z","X")),(3,("Z","X"),("S","S")))]
M2_ex1 = (d2_ex1,0,3,4)
```

exemples d'exécution de la machine M_1^2 telle que $\mathcal{L}(M_1^2) = \{w \in \{a,b\}^\star \mid |w|_a = |w|_b\}$

```
>>> exec_MT_k(M2_ex1,2,[["Z"],["Z"]],[0,0])
(True, [0, 0], [['Z'], ['Z']])
>>> exec_MT_k(M2_ex1,2,[["a","b","b","a","a","a","b","b","Z"],["Z"]],[0,0])
(True, [8, 0], [['a', 'b', 'b', 'a', 'a', 'a', 'b', 'b', 'Z'], ['X', 'Z', 'Z']])
>>> exec_MT_k(M2_ex1,2,[["a","b","b","a","a","a","b","a","Z"],["Z"]],[0,0])
(False, [8, 2], [['a', 'b', 'b', 'a', 'a', 'a', 'b', 'a', 'Z'], ['X', 'Z', 'Z']])
```

Exemple 9 (Langage des palindromes)
Soit $L_{palin} = \{w \in \{0,1\}^\star \mid w = \tilde{w}\}$. On souhaite construire une machine de Turing M_{palin}^2 à deux bandes qui accepte ce langage.

à compléter : machine M_{palin}^2 à 2 bandes

```
d2_palin_bin = ...
M2_palin_bin =(d2_palin_bin,...)
```

exemples d'exécution de la machine M^2_{palin}

```
>>> exec_MT_k(M2_palin_bin,2,[["Z"],["Z"]],[0,0])
(True, [0, 0], [['Z', 'Z'], ['Z', 'Z']])
>>> exec_MT_k(M2_palin_bin,2,[["0","Z"],["Z"]],[0,0])
(True, [2, 0], [['Z', 'X', 'Z'], ['Z', 'X', 'Z']])
>>> exec_MT_k(M2_palin_bin,2,[["1","0","1","Z"],["Z"]],[0,0])
(True, [4, 0], [['Z', 'X', 'X', 'X', 'Z'], ['Z', 'X', 'X', 'X', 'Z']])
>>> exec_MT_k(M2_palin_bin,2,[["1","0","0","1","Z"],["Z"]],[0,0])
(True, [5, 0], [['Z', 'X', 'X', 'X', 'X', 'Z'], ['Z', 'X', 'X', 'X', 'X', 'Z']])
>>> exec_MT_k(M2_palin_bin,2,[["1","1","1","0","1","Z"],["Z"]],[0,0])
(False, [2, 3], [['Z', 'X', '1', '1', '0', '1', 'Z'], ['1', '1', '1', '0', 'X', 'Z']])
>>> exec_MT_k(M2_palin_bin,2,[["1","0","1","0","1","Z"],["Z"]],[0,0])
(True, [6, 0], [['Z', 'X', 'X', 'X', 'X', 'X', 'Z'], ['Z', 'X', 'X', 'X', 'X', 'X', 'Z']])
```

Exemple 10 (Conversion unaire vers binaire)
On souhaite construire une machine à Turing $M_{1\to 2}$ à 2 bandes permettant de convertir un entier naturel en représentation unaire en un entier naturel en représentation binaire. Initialement, la première bande contient le mot unaire (composé de symboles I) en entrée précédé du symbole \$ et la deuxième bande est vide. A l'issue de l'exécution de la machine, la première bande est inchangée et la deuxième bande contient le mot binaire (composé de symboles 0 et 1) précédé du symbole \$.

à compléter : machine $M_{1\to 2}$ à 2 bandes

```
d2_un_to_bin = ...
M2_un_to_bin =(d2_un_to_bin,...)
```

exemples d'exécution de la machine $M_{1\to 2}$

```
>>> exec_MT_k(M2_un_to_bin,2,[["$"],["Z"]],[0,0])
(True, [1, 1], [['$', 'Z'], ['$', '0']])
>>> exec_MT_k(M2_un_to_bin,2,[["$","I"],["Z"]],[0,0])
(True, [2, 1], [['$', 'I', 'Z'], ['$', '1']])
>>> exec_MT_k(M2_un_to_bin,2,[["$","I","I","I"],["Z"]],[0,0])
(True, [4, 1], [['$', 'I', 'I', 'I', 'Z'], ['$', '1', '1', 'Z']])
>>> exec_MT_k(M2_un_to_bin,2,[["$","I","I","I","I"],["Z"]],[0,0])
(True, [5, 1], [['$', 'I', 'I', 'I', 'I', 'Z'], ['$', '1', '0', '0', 'Z']])
>>> exec_MT_k(M2_un_to_bin,2,[["$","I","I","I","I","I"],["Z"]],[0,0])
(True, [6, 1], [['$', 'I', 'I', 'I', 'I', 'I', 'Z'], ['$', '1', '0', '1', 'Z']])
>>> exec_MT_k(M2_un_to_bin,2,[["$","I","I","I","I","I","I"],["Z"]],[0,0])
(True, [7, 1], [['$', 'I', 'I', 'I', 'I', 'I', 'I', 'Z'], ['$', '1', '1', '0', 'Z']])
```

A Machines de Turing utilisées dans le TME

Exemple 11 (Complément d'un mot binaire)
Soit f_{comp} la fonction qui calcule le complément d'un mot binaire. On définit la machine M_{comp} qui implante cette fonction et repositionne la tête de lecture sur le bit le plus à gauche à l'issue du calcul. Par exemple :

$$\begin{array}{lcl} 1001011\sqcup & & 0110100\sqcup \\ \uparrow & \rightsquigarrow^\star & \uparrow \\ q_0 & & q_{ok} \end{array}$$

La machine M_{comp}, représentée sur la figure 6, permet de parcourir le mot binaire en remplaçant chaque symbole 1 par le symbole 0 et chaque symbole 0 par le symbole 1. A l'issue de ce parcours on se place dans l'état q_1 qui permet de replacer la tête de lecture au début du mot.

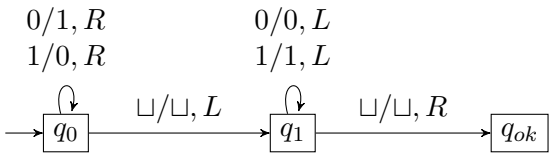


FIGURE 6 – Machine de Turing M_{comp} (exemple 11)

machine M_{comp} ($q_{ok} = 2$ et $q_{ko} = 3$)

```
d_compl_bin = [((0,"0"),(0,"1","R")), ((0,"1"),(0,"0","R")),
               ((0,"Z"),(1,"Z","L")), ((1,"0"),(1,"0","L")),
               ((1,"1"),(1,"1","L")), ((1,"Z"),(2,"Z","R"))]
M_compl_bin = (d_compl_bin,0,2,3)
```

exemple d'exécution de la machine M_{comp}

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

Exemple 12 (Successeur de la représentation binaire d'un entier)
Soit f_{succ} la fonction qui étant donnée un mot binaire w avec les bits de poids forts à droite, effectue l'addition binaire $w + 1$. On définit la machine M_{succ} qui implante cette fonction et repositionne la tête de lecture sur le bit le plus à gauche (à la fin du calcul, l'extrémité droite de la bande correspond soit au bit de poids fort, soit au symbole ⊔). Par exemple :

⊔		1(⊔)	100⊔		110(⊔)	111⊔		0001(⊔)
↑	$\rightsquigarrow^\star$	↑	↑	$\rightsquigarrow^\star$	↑	↑	$\rightsquigarrow^\star$	↑
q_0		q_{ok}	q_0		q_{ok}	q_0		q_{ok}

La machine M_{succ}, représentée sur la figure 7, permet de parcourir le mot binaire en partant du bit de poids faible : dès qu'un symbole 0 est rencontré il est remplacé par le symbole 1 et il ne reste alors qu'à repositionner la tête de lecture sur le bit de poids faible, sinon tant que le symbole lu est le caractère 1, il est remplacé par le symbole 0 et la propagation de la retenue s'effectue sur le bit suivant. On note q_0 l'état dans lequel on propage la retenue (c'est aussi l'état initial), q_1 l'état atteint lorsque le calcul se termine avant la fin de la bande et q_2 l'état atteint lorsque le calcul se termine à la fin de la bande, c-à-d lorsqu'un bit supplémentaire est ajouté à la fin de la bande.

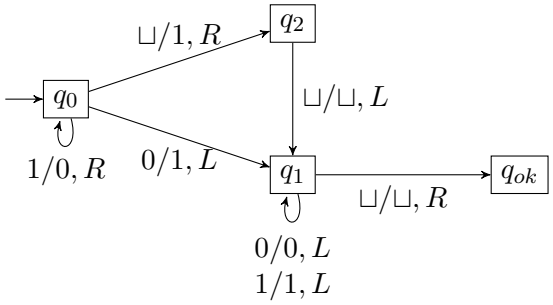


FIGURE 7 – Machine de Turing M_{succ} (exemple 12)

machine M_{succ} ($q_{ok} = 3$ et $q_{ko} = 4$)

```
d_succ_bin = [((0,"0"),(1,"1","L")), ((0,"1"),(0,"0","R")),
              ((0,"Z"),(2,"1","R")), ((1,"0"),(1,"0","L")),
              ((1,"1"),(1,"1","L")), ((1,"Z"),(3,"Z","R")),
              ((2,"Z"),(1,"Z","L"))]
M_succ_bin = (d_succ_bin,0,3,4)
```

exemples d'exécution de la machine M_{succ}

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

Exemple 13 (Fonction identité)
La fonction identité peut être implantée par une machine de Turing M_{id} dont l'état initial est l'état q_{ok}.

machine calculant la fonction identité

```
M_id = ([],0,0,1)
```

Exemple 14 ($L_c = \{c\}$)
Le langage L_c contient un unique mot constitué d'un unique symbole c. On définit une machine de Turing $M_{=c}$ qui détermine si le caractère lu par la tête de lecture est le symbole c et dans ce cas passe dans l'état acceptant q_{ok} (dans le cas contraire la machine passe dans l'état rejetant q_{ko}). Dans les deux cas la tête de lecture est repositionnée sur le caractère testé. Pour définir une telle machine, il faut disposer de l'alphabet Γ contenant tous les caractères possibles sur la bande. Par exemple, la figure 8 représente la machine de Turing $M_{=0}$ lorsque $\Gamma = \{0, 1, \sqcup\}$. Les transitions de la machine $M_{=c}$ dépendent donc de l'alphabet Γ et pour ne pas avoir à

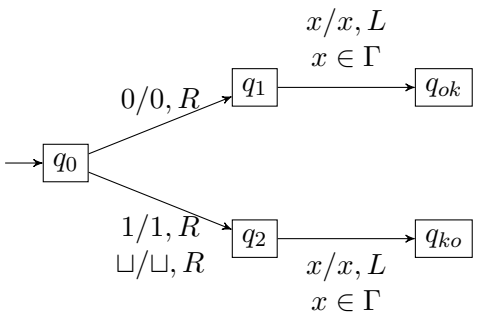


FIGURE 8 – Machine de Turing $M_{=0}$ (exemple 14)

reconstruire «manuellement» cette machine pour des alphabets différents, nous définissons la fonction `make_test_eq` qui automatise cette construction à partir du symbole c et de l'alphabet.

construction de la machine $M_{=c}$ à partir de l'alphabet Γ

```
def make_test_eq(c,alphabet):
    d = []
    for x in alphabet:
        if c==x:
            d = d + [((0,c),(1,c,"R"))]
        else:
            d = d + [((0,x),(2,x,"R"))]
        d = d + [((1,x),(3,x,"L")), ((2,x),(4,x,"L"))]
    M = (d,0,3,4)
    return M
```

exemple : construction de la machine $M_{=0}$ à partir de l'alphabet $\{0, 1, \sqcup\}$

```
M_eq_0 = make_test_eq("0",["0","1","Z"])
```

La machine $M_{\neq c}$ permettant de tester si le caractère lu est différent du symbole c s'obtient simplement en inversant les états acceptant et rejetant de la machine $M_{=c}$.

construction de la machine $M_{\neq c}$ à partir de l'alphabet Γ

```
def make_test_neq(c,alphabet):
    (d,q0,qok,qko) = make_test_eq(c,alphabet)
    return (d,q0,qko,qok)
```

Exemple 15 (Déplacement la tête de lecture d'une position vers la droite)
La machine M_{right} permettant de déplacer la tête de lecture d'une position vers la droite être obtenue avec la fonction suivante paramétrée par l'alphabet Γ.

construction de la machine M_{right} à partir de l'alphabet Γ

```
def make_MTright(alphabet):
    d = []
    for a in alphabet:
        d = d + [((0,a),(1,a,"R"))]
    M = (d,0,1,2)
    return M
```

machine M_{right} à partir de l'alphabet $\Gamma = \{0, 1, \sqcup\}$

```
M_Right_bin = make_MTright(["0","1","Z"])
```

Exemple 16 (Propagation du bit de signe)
La fonction qui duplique le bit situé à l'extrémité droite d'un mot binaire w correspond à la propagation du bit de signe [4] si w est à la représentation binaire en complément à 2 d'un entier relatif, avec les bits de poids faibles à gauche. On définit la machine de Turing M_{propag} qui implante cette fonction et qui repositionne la tête de lecture sur le premier bit du mot à l'issue du calcul.
Le principe de la machine M_{propag}, représentée sur la figure 9, est similaire à celui de la machine M_{isneg} du TME 5. Lorsque que la machine détecte que le dernier bit du mot est le symbole 0 (resp. 1), un symbole 0 (resp. 1) est ajouté à la fin du mot et la tête de lecture est ensuite repositionnée sur le premier bit du mot.

4. Etant donné un mot binaire $a_{n-1} \cdots a_0$ de n bits représentant un entier relatif k en complément à 2 avec les bits de poids forts à gauche, il est possible de représenter ce même entier k par un mot binaire $a_n a_{n-1} \cdots a_0$ de $n+1$ bits en dupliquant le bit de poids fort c-à-d tel que $a_n = a_{n-1}$.

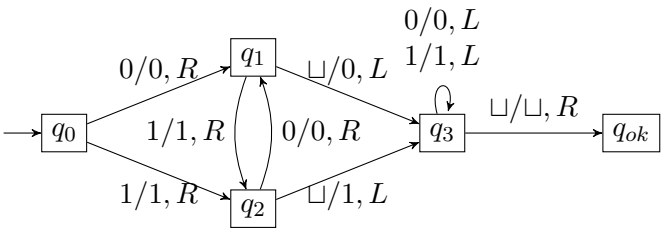


FIGURE 9 – Machine de Turing M_{propag} (exemple 16)

machine M_{propag} ($q_{ok} = 4$ et $q_{ko} = 5$)

```
d_prop1 = [((0,"0"),(1,"0","R")), ((0,"1"),(2,"1","R")),
           ((1,"0"),(1,"0","R")), ((1,"1"),(2,"1","R")), ((1,"Z"),(3,"0","L")),
           ((2,"0"),(1,"0","R")), ((2,"1"),(2,"1","R")), ((2,"Z"),(3,"1","L")),
           ((3,"0"),(3,"0","L")), ((3,"1"),(3,"1","L")), ((3,"Z"),(4,"Z","R"))]
M_prop1 =(d_prop1,0,4,5)
```

exemples d'exécution de la machine M_{propag}

```
>>> exec_MT_1(M_prop1,["0","0","1","0","Z"],0)
(True, 1, ['Z', '0', '0', '1', '0', '0'])
>>> exec_MT_1(M_prop1,["1","0","1","0","1","Z"],0)
(True, 1, ['Z', '1', '0', '1', '0', '1', '1'])
>>> exec_MT_1(M_prop1,["1","Z"],0)
(True, 1, ['Z', '1', '1'])
```