Machines universelles

Notes de cours d'Alain Colmerauer*

mai 1999

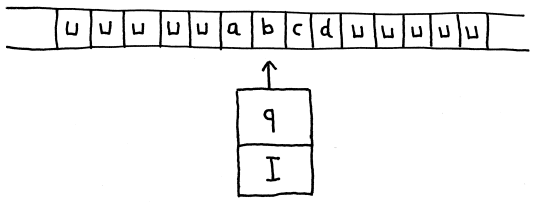
Table des matières

*Laboratoire d'Informatique de Marseille, ESA 6077, CNRS et Universités de Provence et de la Méditerranée

1 Machine de Turing

1.1 La machine physique

Physiquement, une machine de Turing ressemble à ceci :



Elle est composée

- d'un ruban doublement infini à gauche et à droite, découpé en cases avec un symbole dans chaqe case,
- d'une tête de lecture et d'écriture positionnée à tout instant en face d'une case,
- d'une unité centrale, dans laquelle est enregistré un ensemble fixe I d'instructions, et pouvant se trouver dans un ensemble fini d'états q.

Chaque instruction est un quintuplet de la forme (q, a, a', q', d) et signifie : *si l'état de la machine est q et si a est le symbole lu par la tête de lecture, alors la machine*

1. *écrit le symbole a' à la place du symbole a,*

2. *se déplace d'une case à gauche ou à droite suivant que $d = \triangleleft$ ou $d = \triangleright$,*
3. *passe de l'état q à l'état q'.*

Le déroulement d'un calcul avec une telle machine consiste à partir d'une configuration initiale, d'exécuter tant que possible les instructions de la machine et de fournir comme résultat le contenu final du ruban.

Par exemple si l'ensemble enregistré d'instructions est

$$\begin{array}{ll} \{(q_0, 1, 1, q_0, \triangleright), & (q_0, \sqcup, 1, q_1, \triangleleft), \\ (q_1, 1, 1, q_1, \triangleleft), & (q_1, \sqcup, \sqcup, q_2, \triangleright), \\ (q_2, 1, \sqcup, q_3, \triangleright), & (q_2, \sqcup, \sqcup, q_3, \triangleright)\} \end{array}$$

en partant de la configuration

$$\ldots\sqcup\sqcup\underset{\underset{\boxed{q_0}}{\uparrow}}{1}1\sqcup1111\sqcup\sqcup\ldots$$

on obtiendra successivement

$$\ldots\sqcup\sqcup1\underset{\underset{\boxed{q_0}}{\uparrow}}{1}\sqcup1111\sqcup\sqcup\ldots$$

$$\ldots\sqcup\sqcup11\underset{\underset{\boxed{q_0}}{\uparrow}}{\sqcup}1111\sqcup\sqcup\ldots$$

$$\ldots\sqcup\sqcup1\underset{\underset{\boxed{q_1}}{\uparrow}}{1}11111\sqcup\sqcup\ldots$$

$$\ldots\sqcup\sqcup\underset{\underset{\boxed{q_1}}{\uparrow}}{1}111111\sqcup\sqcup\ldots$$

$$\ldots\sqcup\underset{\underset{\boxed{q_1}}{\uparrow}}{\sqcup}1111111\sqcup\sqcup\ldots$$

$$\ldots\sqcup\sqcup\underset{\underset{\boxed{q_2}}{\uparrow}}{1}111111\sqcup\sqcup\ldots$$

et donc pour résultat final

$$\ldots\sqcup\sqcup\sqcup\underset{\underset{\boxed{q_3}}{\uparrow}}{1}11111\sqcup\sqcup\ldots$$

On dispose donc d'une machine permettant de réaliser des additions du genre

$$\overbrace{11\ldots1}^{m} + \overbrace{11\ldots1}^{n} = \overbrace{11\ldots1}^{m+n}$$

1.2 Monoïde libre engendré par un alphabet

Soit Σ un ensemble appelé *alphabet*. Un *mot* construit sur Σ, est une suite finie $a = a_1 a_2 \ldots a_n$ d'éléments a_i de Σ. L'entier n, qui peut être nul, est la *longueur* du mot a. L'unique mot de longueur nul est noté ε et l'ensemble des mots construits sur Σ est noté $\Sigma^\star$.

Si $a = a_1 \ldots a_m$ et $b = b_1 \ldots b_n$ sont des éléments de $\Sigma^\star$, alors $a \cdot b$ où ab est le mot $a_1 \ldots a_m b_1 \ldots b_n$. L'application $(a, b) \mapsto a \cdot b$ de $\Sigma^\star \times \Sigma^\star$ dans $\Sigma^\star$ est appelée *opération de concaténation*. C'est une opération associative, c'est à dire que $(x \cdot y) \cdot z = x \cdot (y \cdot z)$, et elle admet ε pour élément neutre, c'est-à-dire que $x \cdot \varepsilon = \varepsilon \cdot x = x$. Si n est est entier positif ou nul et x un élément de $\Sigma^\star$, on désigne par x^n l'élément $\underbrace{x \cdots x}_{n}$ de $\Sigma^\star$. Dans le cas particulier où $n = 0$, on considère que $x^n = \varepsilon$.

Le couple $(\Sigma^\star, \cdot)$ est appelé *monoïde libre* engendré par l'alphabet Σ.

1.3 La machine mathématique

Donnons nous une bonne fois pour toutes un élément $\sqcup$ appelé *symbole blanc* et, si x est un mot quelconque sur un alphabet Σ, notons $\mathrm{D}(x)$ le mot obtenu en enlevant du début de x tous les $\sqcup$ et $\mathrm{F}(x)$ le mot obtenu en enlevant de la fin de x tous les $\sqcup$.

Une machine de Turing est un quintuplet $M = (\Sigma, Q, q_0, Q', \tau)$ où

– Σ, l'*alphabet de M*, est un ensemble fini ayant $\sqcup$ pour élément,

– Q, l'*ensemble des états de M*, est un ensemble fini,

– q_0, l'*état initial de M*, est un élément privilégié de Q,

– Q', l'*ensemble des états finaux de M*, est un sous-ensemble de Q,

– τ, la *fonction de transition de M*, est une application de type $(Q - Q') \times \Sigma \to Q \times \Sigma \times \{\triangleleft, \triangleright\}$.

L'ensemble des *instructions de M* est l'ensemble des quintuplets de la forme (q, a, a', q', d), avec

$$(q, a) \in (Q - Q') \times \Sigma \text{ et } (q', a', d) = \tau(q, a).$$

Une *configuration de M* est un triplet de la forme

$$(q, D(x), F(y)),$$

avec $q \in Q$, $x \in \Sigma^\star$ et $y \in \Sigma^\star$. Dans l'ensemble des configurations de M on introduit la relation binaire $\stackrel{M}{\to}$ définie par

$$(q, \mathrm{D}(xa), \mathrm{F}(by)) \stackrel{M}{\to} (q', \mathrm{D}(xu), \mathrm{F}(vy))$$

ssi

$$(u,v) = \begin{cases} (\varepsilon, ab'), & \text{si } d = \lhd, \\ (ab', \varepsilon), & \text{si } d = \rhd. \end{cases}$$

pour tous $a, b, b' \in \Sigma$, $x, y \in \Sigma^s tar$, $q, q' \in Q$, et $d \in \{\lhd, \rhd\}$ tels que $\tau(q,a) = (a', q', d)$.

On note $\overset{M^\star}{\to}$ la fermeture transitive réflexive de $\overset{M}{\to}$ c'est-à-dire que $c \overset{M^\star}{\to} c'$ si et seulement si il existe $c_0, c_1, \ldots, c_n$, avec $n \geq 0$, et $c = c_0$, $c_0 \overset{M}{\to} c_1$, $c_1 \overset{M}{\to} c_2$, $\ldots$, $c_{n-1} \overset{M}{\to} c_n$, $c_n = c'$.

Soit ω une valeur que l'on lit « indéfini » et qui n'appartient pas à $\Sigma^\star$. A la machine de Turing M on associe la fonction $\overline{M}$, de type $\Sigma^\star \cup \{\omega\} \to \Sigma^\star \cup \{\omega\}$, définie par

$$\overline{M}(x) = \begin{cases} y, & \text{si } x \in \Sigma^\star \text{ et il existe } (q_i, y', y) \text{ avec} \\ & (q_0, \varepsilon, \mathrm{D}(x)) \overset{M^\star}{\to} (q_i, y', y) \text{ et } q_i \in Q', \\ \omega, & \text{sinon.} \end{cases}$$

On dit que la machine M est *fiable* si pour aucun $x \in \Sigma^\star$ on a $\overline{M}(x) = \omega$.

1.4 Exemple : somme d'entiers codés par des bâtons

Reprenons l'exemple de machine donné à la section 1.1. C'est donc la machine $M = (\Sigma, Q, q_0, Q', \tau)$ avec $\Sigma = \{\sqcup, 1\}$, $Q = \{q_0, q_1, q_2, q_3\}$, $Q' = \{q_3\}$ et τ est défini par l'ensemble d'instructions

$$\begin{array}{ll} \{(q_0, 1, 1, q_0, \rhd), & (q_0, \sqcup, 1, q_1, \lhd), \\ (q_1, 1, 1, q_1, \lhd), & (q_1, \sqcup, \sqcup, q_2, \rhd), \\ (q_2, 1, \sqcup, q_3, \rhd), & (q_2, \sqcup, \sqcup, q_3, \rhd)\} \end{array}$$

On a par exemple

$$\begin{array}{ll} & (q_0, \varepsilon, \sqcup 111 \sqcup 1111) \\ \overset{M}{\to} & (q_0, \sqcup, 111 \sqcup 1111) \\ \overset{M}{\to} & (q_0, 1, 11 \sqcup 1111) \\ \overset{M}{\to} & (q_0, 11, 1 \sqcup 1111) \\ \overset{M}{\to} & (q_0, 111, \sqcup 1111) \\ \overset{M}{\to} & (q_1, 11, 111111) \\ \overset{M}{\to} & (q_1, 1, 1111111) \\ \overset{M}{\to} & (q_1, \varepsilon, 11111111) \\ \overset{M}{\to} & (q_1, \varepsilon, \sqcup 11111111) \\ \overset{M}{\to} & (q_2, \varepsilon, 11111111) \\ \overset{M}{\to} & (q_3, \varepsilon, 1111111) \end{array}$$

Pour tous les entiers naturels m, n on a alors

$$\overline{M}(1^m \sqcup 1^n) \;=\; 1^{m+n}$$

1.5 Exemple : duplication d'un mot

Soit un alphabet de la forme $\Sigma = \{\sqcup, a_1, \ldots, a_n\}$. Voici maintenant une machine $M = (\Sigma, Q, q_0, Q', \tau)$ qui duplique les mots sur $\{a_1, \ldots, a_n\}$, c'est-à-dire telle que

$$\overline{M}(x) = x \cdot x$$

pour tout $x \in \Sigma^\star - \{\sqcup\}$. L'ensemble Q de ses états est formé des $5(n+1)+1$ éléments, $q_0, q_1, q_{2a_i}, q_{3a_i}, q_{4a_i}, q_{5a_i}, q_6, q_7, q_{8a_i}, q_9, q_{10}$, avec $i \in \{1, \ldots, n\}$. L'ensemble Q' de ses états finaux est $\{q_{10}\}$ et sa fonction de transition τ est définie par les $5(n+1)^2$ instructions,

$$\begin{array}{ll} (q_0, \sqcup, \sqcup, q_7, \rhd), & (q_0, a_i, a_i, q_1, \rhd), \\ (q_1, \sqcup, \sqcup, q_6, \lhd), & (q_1, a_i, \sqcup, q_{2a_i}, \rhd), \\ (q_{2a_i}, \sqcup, \sqcup, q_{3a_i}, \rhd), & (q_{2a_i}, a_j, a_j, q_{2a_i}, \rhd), \\ (q_{3a_i}, \sqcup, a_i, q_{4a_i}, \lhd), & (q_{3a_i}, a_j, a_j, q_{3a_i}, \rhd), \\ (q_{4a_i}, \sqcup, \sqcup, q_{5a_i}, \lhd), & (q_{4a_i}, a_j, a_j, q_{4a_i}, \lhd), \\ (q_{5a_i}, \sqcup, a_i, q_1, \rhd), & (q_{5a_i}, a_j, a_j, q_{5a_i}, \lhd), \\ (q_6, \sqcup, \sqcup, q_7, \rhd), & (q_6, a_i, a_i, q_6, \lhd), \\ (q_7, \sqcup, \sqcup, q_{10}, \lhd), & (q_7, a_i, a_i, q_{8a_i}, \rhd), \\ (q_{8a_i}, \sqcup, a_i, q_9, \lhd), & (q_{8a_i}, a_j, a_j, q_{8a_i}, \rhd), \\ (q_9, \sqcup, \sqcup, q_{10}, \rhd), & (q_9, a_i, a_i, q_9, \lhd), \end{array}$$

avec $i, j \in \{1, \ldots, n\}$. Prenons $n = 2$ et $0, 1$ pour les symboles a_1, a_2. La machine fonctionne bien dans le cas particulier où x est de longueur 0 ou 1. On a

$$\overline{M}(\varepsilon) = \varepsilon$$

car

$$\begin{array}{ll} & (q_0, \varepsilon, \varepsilon) \\ \to & (q_7, \varepsilon, \varepsilon) \\ \to & (q_{10}, \varepsilon, \varepsilon) \end{array}$$

On a

$$\overline{M}(1) = 11$$

car

$$\begin{array}{ll} & (q_0, \varepsilon, 1) \\ \to & (q_1, 1, \varepsilon) \\ \to & (q_6, \varepsilon, 1) \\ \to & (q_6, \varepsilon, \sqcup 1) \\ \to & (q_7, \varepsilon, 1) \\ \to & (q_{81}, 1, \varepsilon) \\ \to & (q_9, \varepsilon, 11) \\ \to & (q_9, \varepsilon, \sqcup 11) \\ \to & (q_{10}, \varepsilon, 11) \end{array}$$

Dans un cas plus général, on a

$$\overline{M}(101) = 101101$$

car

	$(q_0, \varepsilon, 101)$		$(q_{51}, 10, \sqcup\sqcup 01)$
$\to$	$(q_1, 1, 01)$	$\to$	$(q_1, 101, \sqcup 01)$
$\to$	$(q_{20}, 1\sqcup, 1)$	$\to$	$(q_6, 10, 1\sqcup 01)$
$\to$	$(q_{20}, 1\sqcup 1, \varepsilon)$	$\to$	$(q_6, 1, 01\sqcup 01)$
$\to$	$(q_{30}, 1\sqcup 1\sqcup, \varepsilon)$	$\to$	$(q_6, \varepsilon, 101\sqcup 01)$
$\to$	$(q_{40}, 1\sqcup 1, \sqcup 0)$	$\to$	$(q_6, \varepsilon, \sqcup 101\sqcup 01)$
$\to$	$(q_{50}, 1\sqcup, 1\sqcup 0)$	$\to$	$(q_7, \varepsilon, 101\sqcup 01)$
$\to$	$(q_{50}, 1, \sqcup 1\sqcup 0)$	$\to$	$(q_{81}, 1, 01\sqcup 01)$
$\to$	$(q_1, 10, 1\sqcup 0)$	$\to$	$(q_{81}, 10, 1\sqcup 01)$
$\to$	$(q_{21}, 10\sqcup, \sqcup 0)$	$\to$	$(q_{81}, 101, \sqcup 01)$
$\to$	$(q_{31}, 10\sqcup\sqcup, 0)$	$\to$	$(q_9, 10, 1101)$
$\to$	$(q_{31}, 10\sqcup\sqcup 0, \varepsilon)$	$\to$	$(q_9, 1, 01101)$
$\to$	$(q_{41}, 10\sqcup\sqcup, 01)$	$\to$	$(q_9, \varepsilon, 101101)$
$\to$	$(q_{41}, 10\sqcup, \sqcup 01)$	$\to$	$(q_9, \varepsilon, \sqcup 101101)$
$\to$	$(q_{51}, 10, \sqcup\sqcup 01)$	$\to$	$(q_{10}, \varepsilon, 101101)$

1.6 Machine universelle

Désignons par $T(\Sigma)$ l'ensemble des machines de Turing d'alphabet Σ. Une machine de Turing U d'alphabet Υ est dite *universelle pour son alphabet*, si il existe une fonction de *codage* $\mu : T(\Upsilon) \mapsto \Upsilon^\star$ telle que,

$$\overline{U}(\mu(M) \cdot x) = \overline{M}(x),$$

pour tout $M \in T(\Upsilon)$ et $x \in \Upsilon^\star$. On remarque que $\overline{U}(\mu(U) \cdot \mu(M) \cdot x) = \overline{M}(x)$ et, d'une façon plus générale que, pour tout $n \geq 0$,

$$\overline{U}(\mu(U)^n \cdot \mu(M) \cdot x) = \overline{M}(x). \tag{1}$$

Il est possible de construire un telle machine U avec un codage μ simple et l'objet des travaux pratiques de ce cours sera d'en construire une sur un alphabet de 4 symboles.

Une machine de Turing U sur l'alphabet Υ est dite simplement *universelle* si, pour tout alphabet fini Σ, il existe deux fonctions de *codage* $\mu : T(\Sigma) \mapsto \Upsilon^\star$ et $\delta : \Sigma^\star \mapsto \Upsilon^\star$, avec δ injectives, telles que

$$\overline{U}(\mu(M) \cdot \delta(x)) = \delta(\overline{M}(x)),$$

pour tout $M \in T(\Sigma)$ et $x \in \Sigma^\star$. Ici aussi on remarque que $\overline{U}(\mu(U) \cdot \delta(\mu(M) \cdot \delta(x))) = \delta(\overline{M}(x))$ et, d'une façon plus générale que, pour tout $n \geq 0$,

$$\overline{U}(f^n(\mu(M) \cdot \delta(x))) = \delta^n(\overline{M}(x)), \tag{2}$$

où f est la fonction

$$f : y \mapsto \mu(U) \cdot \delta(y).$$

On montre qu'il est possible de construire une machine universelle avec des codages μ, δ simples.

2 Calculabilité et décidabilité

2.1 Calculabilité

Soit Σ un alphabet fini avec $\sqcup \notin \Sigma$. Une fonction f de type $\Sigma^\star \to \Sigma^\star$ est dite *calculable* s'il existe une machine de Turing M sur l'alphabet $\Sigma \cup \{\sqcup\}$ telle que $f(x) = \overline{M}(x)$, pour tout $x \in \Sigma^\star$.

Nous allons montrer par une technique dite de *diagonalisation* que :

> *Il existe des fonctions de type $\Sigma^\star \to \Sigma^\star$ qui ne sont pas calculables.*

Autrement dit, les machines de Turing ne peuvent pas tout faire !

Soit $\mathbf{N}$ l'ensemble des entiers naturels, c'est-à-dire positifs (ou nul). L'ensemble G des fonctions calculables est dénombrable, c'est-à-dire qu'il existe une application bijective

$$i \mapsto g_i \tag{3}$$

de type $\mathbf{N} \to G$. En effet l'ensemble G_n des éléments de G qui sont représentables par une machine d'au moins n états est fini. On peut donc poser $G_n = \{g_{n1}, g_{n2}, \ldots, _{n|G_n|}\}$ et prendre pour suite infinie $g_0, g_1, g_2, \ldots$ la suite

$$g_{11}, \ldots, g_{1|G_1|}, g_{21}, \ldots, g_{2|G_2|}, g_{31}, \ldots, g_{3|G_3|}, \ldots$$

De la même façon, en remarquant que pour un n donné l'ensemble des mot de longueur n sur Σ est fini, on conclut qu'il existe une application bijective

$$i \mapsto x_i \tag{4}$$

de type $\mathbf{N} \to \Sigma^\star$. Les applications (3) et (4) peuvent être visualisées par le tableau

	x_0	x_1	x_2	$\cdots$	x_i	$\cdots$
g_0	$g_0(x_0)$					
g_1		$g_1(x_1)$				
g_2			$g_2(x_2)$			
$\vdots$				$\ddots$		
g_i					$g_i(x_i)$	
$\vdots$						$\ddots$

dont les lignes sont indicées par les g_i, les colonnes par les x_j et dont les cases de coordonnées (g_i, x_j) ont pour valeur $g_i(x_j)$. On considère la fonction

$$d : x_i \mapsto g_i(x_i) \tag{5}$$

de type $\Sigma^\star \to \Sigma^\star$, qui est représentée par la diagonale du tableau. Soit d' une fonction de type $\Sigma^\star \to \Sigma^\star$ telle que

$$d'(y) \neq d(y), \quad \text{pour tout } y \in \Sigma^\star. \tag{6}$$

La fonction d' n'est pas calculable. En effet si elle l'était, il existerait un indice j tel que $d' = g_j$ et donc telle que $d'(x_j) = g_j(x_j)$, c'est-à-dire, d'après (5), telle que $d'(x_i) = d(x_i)$. Il y aurait alors contradiction avec (6).

Curieusement on montre aussi que la fonction « diagonale » d, définie en (5), n'est pas calculable. Supposons que d soit calculable par une machine de Turing

$$D = (\Sigma \cup \{\sqcup\}, Q, q_0, Q', \tau),$$

avec $Q = \{q_0, \ldots, q_{n-1}\}$. Soit la machine de Turing

$$D' = (\Sigma \cup \{\sqcup\}, Q \cup \{q_n, q_{n+1}, q_{n+2}\}, q_0, \{q_{n+2}\}, \tau'),$$

obtenue en ajoutant trois nouveaux état q_n, q_{n+1}, q_{n+2}, en considérant que q_{n+2} est le seul état final et en ajoutant aux instructions de τ l'ensemble des instructions de la forme

$$(q_i, a, a, q_n, \triangleleft),$$
$$(q_n, a, b, q_{n+1}, \triangleleft),$$
$$(q_{n+1}, a, a, q_{n+2}, \triangleright),$$

avec $q_i \in Q'$, $a \in \Sigma \cup \{\sqcup\}$ et b toujours le même élément de Σ. Soit la fonction $d' : y \mapsto \overline{D'}(y)$ de type $\Sigma^\star \to \Sigma^\star$. Par supposition et construction cette fonction d' serait calculable et telle que

$$d'(y) = b \cdot d(y),$$

pour tout $y \in \Sigma^\star$. Du fait que d' satisferait à (6), d'après ce que nous venons de montrer, d' ne serait pas calculable, ce qui contredirait notre supposition. Il s'ensuit que d n'est pas calculable.

2.2 Décidabilité et indécidabilité

Soit toujours Σ un alphabet fini avec $\sqcup \notin \Sigma$ et soient deux éléments privilégiés de $\Sigma^\star$ notés *vrai* et *faux*. On s'intéresse maintenant au cas particulier des fonctions de type

$$\Sigma^\star \to \{vrai, faux\}. \tag{7}$$

On appelle *langage* tout sous-ensemble L de $\Sigma^\star$. On dit que la reconnaissance de L est *décidable*, ou tout simplement que L est décidable, s'il existe une fonction caculable f de type (7) telle que, pour tout $x \in \Sigma^\star$,

$$x \in L \quad \text{si et seulement si} \quad f(x) = vrai.$$

On utilise le mot *indécidable* dans le sens de non décidable. On montre que :

> *Il existe des langages dont la reconnaissnce est indécidable.*

Pour ceci il suffit de montrer qu'il existe des fonctions de type (7) qui ne sont pas calculables. On procède alors comme à la section précédente en restreignant G à l'ensemble des fonctions calculables de type (7). On introduit la fonction diagonale $d : x_i \mapsto g(x_i)$ et la fonction d' qui ne peut être que la *négation* de d, définie par

$$x_i \mapsto \begin{cases} vrai, & \text{si } d(x) = faux, \\ faux, & \text{si } d(x) = vrai. \end{cases} \tag{8}$$

On conclut alors que d' n'est pas calculable.

Si l'on dipose d'une machine de Turing pour calculer une fonction f de type (7) il est facile de transformer cette machine en une machine qui calcule la négation de f. Comme à la section précédente on conclut alors que la fonction diagonale d n'est pas calculable mais aussi d'une façon générale que :

> *Si la reconnaissance d'un langage L est décidable alors la reconnaissance de son complément $\Sigma^\star - L$ l'est aussi.*

2.3 Indécidabilité de l'arrêt d'une machine de Turing

Soit $T(\Sigma)$ l'ensemble des machines construites sur l'alphabet $\Sigma \cup \{\sqcup\}$ avec $\sqcup \notin \Sigma$. Soit μ une application de type $T(\Sigma) \to \Sigma^\star$ et soit le langage

$$L_\mu = \{\mu(M) \cdot x \mid M \in T(\Sigma),\ x \in \Sigma^\star \text{ et } M(x) \neq \omega\}$$

Rappelons que, par définition, $\overline{M}(x) = \omega$ signifie que l'exécution de la machine M sur la donnée x ne s'arrête pas. Nous allons montrer que :

> *La reconnaissance du langage L_μ est indécidable.*

La reconnaissance de L_μ est connue sous le nom de *problème de l'arrêt d'une machine de Turing.*

Supposons que la reconnaissance de L_μ soit décidable et montrons que l'on aboutit à une contradiction. Il existe donc une machine de Turing A telle que

$$\overline{A}(\mu(M) \cdot x) = faux \text{ ssi } \overline{M}(x) = \omega,$$

pour tout $M \in T(\Sigma)$ et $x \in \Sigma^\star$. Soit B la machine duplicatrice définie à la section 1.5, qui est telle que $\overline{B}(y) = y \cdot y$. On peut alors construire une machine C telle que $\overline{C}(y) = \overline{A}(\overline{B}(y))$, donc telle que, pour tout $M \in T(\Sigma)$,

$$\overline{C}(\mu(M)) = faux \text{ ssi } \overline{A}(\mu(M) \cdot \mu(M)) = faux,$$

c'est-à-dire

$$\overline{C}(\mu(M)) = faux \text{ ssi } \overline{M}(\mu(M)) = \omega.$$

En prenant $M = C$ on aboutit à la contradiction

$$\overline{C}(\mu(C)) = faux \text{ ssi } \overline{C}(\mu(C)) = \omega.$$

2.4 Autres langages et problèmes indécidables

Voici deux autres célèbres langages indécidables

Problème de correspondance de Post

Soit un alphabet fini de la forme $\Sigma = \Sigma' \cup \{\#\}$ avec $\# \notin \Sigma'$. Soit C l'ensemble des ensembles de la forme

$$\{(x_1, y_1), \ldots, (x_n, y_n)\}$$

où les x_i et y_i sont des mots sur Σ tels qu'il existe une suite $i_1, \ldots, i_k$ d'indices avec

$$x_{i_1} \cdots x_{i_k} = y_{i_1} \cdots y_{i_k}$$

Soit la fonction de codage

$$\pi : \{(x_1, y_1), \ldots, (x_n, y_n)\} \mapsto x_1 \# y_1 \# \cdots x_n \# y_n \#$$

qui est de type $C \to \Sigma^\star$ et soit le langage

$$L_C = \{\pi(c) \in \Sigma^\star \mid c \in C\}.$$

On montre, mais nous ne le ferons pas ici, que

> *La reconnaissance du langage L_C n'est pas décidable*

La reconnaissance de L_C est connue sous le nom de *problème de correspondance de Post.*

Dixième problème de Hilbert

Soit $\mathbf{Z}$ l'ensemble des entiers (relatif) et soit P l'ensemble des équations de la forme

$$p(x_1, \ldots, x_n) = 0$$

faisant intervenir les variables $x_1, \ldots, x_n$, ayant des coefficients entiers et ayant au moins une solution dans $\mathbf{Z}$. Soit π une fonction de codage immédiate de type $P \to \Sigma^\star$ et soit le langage

$$L_P = \{\mu(p) \in \Sigma^\star \mid p \in P\}$$

On montre, mais nous ne le ferons pas ici, que

> *La reconnaissance du langage L_P est indécidable*

La reconnaissance de L_P est connue sous le nom de *dixième problème de Hilbert.*

2.5 Semi-décidabilité

On dit que le langage L ou que sa reconnaissance est *semi-décidable*, s'il existe une application f de type

$$\Sigma^\star \to \{vrai, faux, \omega\}$$

telle que, pour tout $x \in \Sigma^\star$,

$$x \in L \text{ ssi } f(x) = vrai$$

On dit aussi que L est *récursivement énumérable.* On peut montrer, mais nous ne le ferons pas ici, que langage L_μ défini à la section 2.3 est semi-décidable. Donc :

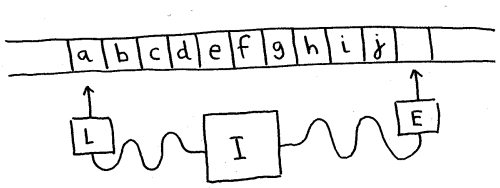
> *Il existe des langages indécidables qui sont semi-décidables (récursivement énumérables).*

3 Machine à s'attraper (tag machine)

Cette section est consacrée à une machine encore plus simple qu'une machine de Turing, mais ayant la même puissance : la machine à s'attraper, en anglais, tag machine.

3.1 La machine physique

Donnons un entier d strictement plus grand que 1. Physiquement, une machine à s'attraper de pas d ressemble à ceci :



Elle est composée

- d'un ruban doublement infini à gauche et à droite, découpé en cases avec un symbole dans chaque case,
- d'une tête de lecture positionnée à tout instant en face d'une case,
- d'une tête d'écriture positionnée à tout instant en face d'une case se trouvant à droite de la case sur laquelle est positionnée la tête de lecture,
- d'une unité centrale, dans laquelle est enregistré un ensemble fixe I d'instructions.

Chaque instruction est un doublet de la forme $a_0 \to a_1 \ldots a_n$ et signifie : *si le symbole lu est a_0 alors*

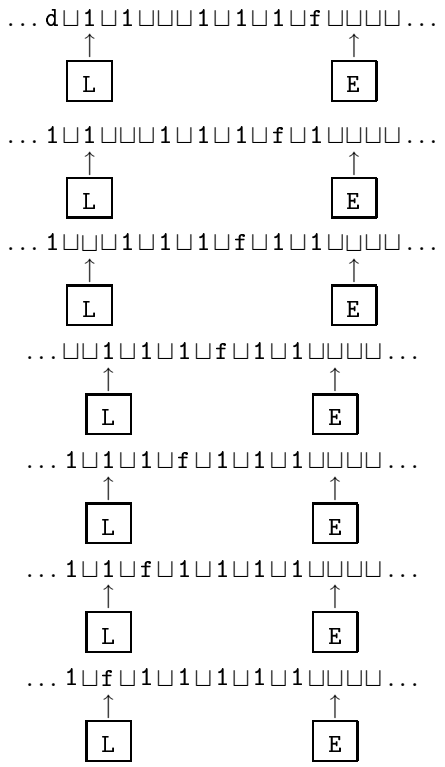
1. *la tête de lecture se positionne d cases plus loin à droite,*
2. *la tête d'écriture écrit de gauche à droite la suite de symboles $a_1 \ldots a_n$ et se positionne n cases plus loin à droite.*

Le déroulement d'un calcul avec une telle machine consiste à partir d'un ruban initial, à exécuter tant que possible les instructions de la machine. Par exemple si $d = 2$ et que l'ensemble enregistré d'instructions est constitué de

$$\begin{array}{lcl} \texttt{d} & \to & \texttt{f}\sqcup \\ \texttt{1} & \to & \texttt{1}\sqcup \\ \sqcup & \to & \varepsilon \end{array}$$

(on rappelle que ε est le mot vide), en partant de la configuration

on obtiendra successivement



On a donc construit un additionneur qui à partir de

$$\mathtt{d}\sqcup\overbrace{\mathtt{1}\sqcup\mathtt{1}\sqcup\ldots\mathtt{1}\sqcup}^{m\text{ fois }1}\sqcup\sqcup\overbrace{\mathtt{1}\sqcup\mathtt{1}\sqcup\ldots\mathtt{1}\sqcup}^{n\text{ fois }1}$$

calcule

$$\mathtt{f}\sqcup\overbrace{\mathtt{1}\sqcup\mathtt{1}\sqcup\ldots\mathtt{1}\sqcup}^{m+n\text{ fois }1}$$

3.2 La machine mathématique

Une machine à s'attraper se formalise comme un quadruplet $M = (d, \Sigma, \Sigma', \tau)$ où

- d, le *pas de* M, est un entier strictement plus grand que 1,
- Σ, l'*alphabet de* M, est un ensemble fini de symboles,
- Σ', l'*ensemble des symboles finaux de* M, est un sous-ensemble de Σ,
- τ, la *fonction de transition de* M, est une application de type $(\Sigma - \Sigma') \to \Sigma^\star$.

L'ensemble des *instructions de* M est l'ensemble des doublets de la forme

$$a \to \tau(a),$$

avec $a \in (\Sigma - \Sigma')$. Une *configuration de* M est un élément de $\Sigma^\star$. Dans l'ensemble des configurations de M on introduit la relation binaire $\overset{M}{\to}$ définie par

$$a_1 \cdots a_d x \ \overset{M}{\to}\ x{\cdot}\tau(a_1)$$

pour tout $a_i \in \Sigma$, avec $a_1 \in (\Sigma - \Sigma')$, et $x \in \Sigma^\star$.

On note $\overset{M^\star}{\to}$ la fermeture transitive réflexive de $\overset{M}{\to}$ c'est-à-dire que $x \overset{M^\star}{\to} y$ si et seulement si il existe $u_0, u_1, \ldots, u_n$, avec $n \geq 0$, et $x = u_0$, $u_0 \overset{M}{\to} u_1$, $u_1 \overset{M}{\to} u_2$, $\ldots$, $u_{n-1} \overset{M}{\to} u_n$, $u_n = y$.

Soit ω une valeur que l'on lit « indéfini » et qui n'appartient pas à $\Sigma^\star$. A la machine à s'attraper M on associe la fonction $\overline{M}$, de type $\Sigma^\star \cup \{\omega\} \to \Sigma^\star \cup \{\omega\}$, définie par

$$\overline{M}(x) = \begin{cases} y, & \text{si } x \in \Sigma^\star \text{ et } x \overset{M^\star}{\to} y \text{ et, soit } y = \varepsilon, \\ & \text{soit } y \text{ commence par un élément de } \Sigma', \\ \omega, & \text{sinon.} \end{cases}$$

On dit que la machine M est *fiable* si pour aucun $x \in \Sigma^\star$ on a $\overline{M}(x) = \omega$.

3.3 Exemple : somme d'entiers codés par des bâtons

Reprenons l'exemple de machine donné à la section 3.1. C'est donc la machine de $M = (d, \Sigma, \Sigma', \tau)$ avec $d = 2$ $\Sigma = \{\sqcup, \mathtt{1}, \mathtt{d}, \mathtt{f}\}$, $\Sigma' = \{\mathtt{d}, \mathtt{f}\}$ et τ est défini par l'ensemble d'instruction

$$\{\mathtt{d} \to \mathtt{f}\sqcup,\ \mathtt{1} \to \mathtt{1}\sqcup,\ \sqcup \to \varepsilon\}$$

On a par exemple

$$\begin{array}{ll} & \mathtt{d}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup\sqcup\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup \\ \overset{M}{\to} & \mathtt{1}\sqcup\mathtt{1}\sqcup\sqcup\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup\mathtt{f}\sqcup \\ \overset{M}{\to} & \mathtt{1}\sqcup\sqcup\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup\mathtt{f}\sqcup\mathtt{1}\sqcup \\ \overset{M}{\to} & \sqcup\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup\mathtt{f}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup \\ \overset{M}{\to} & \mathtt{1}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup\mathtt{f}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup \\ \overset{M}{\to} & \mathtt{1}\sqcup\mathtt{1}\sqcup\mathtt{f}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup \\ \overset{M}{\to} & \mathtt{1}\sqcup\mathtt{f}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup \\ \overset{M}{\to} & \mathtt{f}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup\mathtt{1}\sqcup \end{array}$$

Pour tous les entiers naturels m, n on a alors

$$\overline{M}(\mathtt{d}\sqcup(\mathtt{1}\sqcup)^m\sqcup\sqcup\mathtt{d}\sqcup(\mathtt{1}\sqcup)^n) \ = \ \mathtt{f}\sqcup(\mathtt{1}\sqcup)^{m+n}$$

3.4 Exemple : machine euclidienne droite

Voici maintenant une machine un peu compliquée, dont nous aurons besoin par la suite. Cette machine, entre autres, calcule la division euclidienne par d d'un entier q codé dans la partie droite d'un mot.

Etant donné un alphabet Σ, on désigne par

$$\mathit{droit}\,(d, k, A, A', B, B', C, C', D, D')$$

l'ensemble des machines $M = (d, \Sigma, \Sigma', \tau)$, où pour chaque $i \in 0..(d-1)$

$$A, A', B, B', C_i, C'_i, D_i, D'_i \in \Sigma,$$

où

$$\Sigma' = \{C_0, C'_0, D_0, D'_0, \ldots, C_{d-1}, C'_{d-1}, D_{d-1}, D'_{d-1}\}$$

et où τ est défini par les $10+2d$ instructions :

$$\boxed{\begin{array}{l} A \to E\lambda(E'\lambda)^k, \\ A' \to (E'\lambda)^d, \\ \\ B \to F, \\ B' \to F', \\ \\ E \to G_{d-1}\cdots G_1 G_0,, \\ E' \to G'_{d-1}\cdots G'_1 G'_0 \\ \\ F \to H_{d-1}\cdots H_1 H_0, \\ F' \to H'_{d-1}\cdots H'_1 H'_0, \\ \\ G_i \to \#^{d-1-i} C_i \lambda, \\ G'_i \to C'_i \lambda, \\ \\ H_i \to D_i \lambda, \\ H'_i \to D'_i \lambda, \end{array}}$$

avec $\lambda = \#^{d-1}$, $i \in 0..(d-1)$ et

$$\#, E, E', F, F', G_i, G'_i, H_i, H'_i \in \Sigma.$$

Si p, q, k sont des entiers naturels avec $k < d$ alors

$$\boxed{\overline{M}(\ A\lambda(A'\lambda)^p B\lambda(B'\lambda)^q\ ) \ = \ C_r\lambda(C'_r\lambda)^{p'} D_r\lambda(D'_r\lambda)^{q'},} \quad (9)$$

où p', q', r sont les entiers naturels tels que

$$\boxed{\begin{array}{lcl} p' & = & pd+k, \\ q & = & q'd+r, \\ r & < & d. \end{array}}$$

Montrons que l'on a bien (9). Pour alléger les notations, posons $e = d-1$. On a successivement

$$A\lambda(A'\lambda)^p B\lambda(B'\lambda)^q$$
$$\xrightarrow{M}$$
$$B\lambda(B'\lambda)^q E\lambda(E'\lambda)^{p'}$$
$$\xrightarrow{M}$$
$$E\lambda(E'\lambda)^{p'} F(F')^q$$
$$\xrightarrow{M}$$
$$F(F')^{q'd+r} G_e\cdots G_0 (G'_e\cdots G'_0)^{p'}$$
$$=$$
$$F\,(\underbrace{F'\cdots F'}_{e}\ F')^{q'}\ \overbrace{\underbrace{F'\cdots F'}_{e}}^{r}\ \overbrace{G_e\cdots G_{r+1}}^{e-r}$$
$$G_r\cdots G_0\,(G'_e\cdots G'_0)^{p'}$$
$$\xrightarrow{M}$$
$$G_r\cdots G_0\,(G'_e\cdots G'_0)^{p'}\ H_e\cdots H_0\,(H'_e\cdots H'_0)^{q'}$$
$$=$$
$$G_r\ \underbrace{G_{r+1}\cdots G_0}_{r}\ \underbrace{(G'_e\cdots G'_{r+1}}_{e-r}\ G'_r\ \underbrace{G'_{r+1}\cdots G'_0)^{p'}}_{r}\ \underbrace{H_e\cdots H_{r+1}}_{e-r}$$
$$H_r\cdots H_0\,(H'_e\cdots H'_0)^{q'}$$
$$\xrightarrow{M}$$
$$H_r\cdots H_0\,(H'_e\cdots H'_0)^{q'}\ \#^{e-r}\ C_r\,\lambda\,(C'_r\,\lambda)^{p'}$$
$$=$$
$$H_r\ \underbrace{H_{r-1}\cdots H_0}_{r}\ \underbrace{(H'_e\cdots H'_{r+1}}_{e-r}\ H'_r\ \underbrace{H'_{r-1}\cdots H'_0)^{q'}}_{r}\ \underbrace{\#\cdots\#}_{e-r}$$
$$C_r\,\lambda\,(C'_r\,\lambda)^{p'}$$
$$\xrightarrow{M}$$
$$C_r\,\lambda\,(C'_r\,\lambda)^{p'}\ D_r\,\lambda\,(D'_r\,\lambda)^{q'}$$

3.5 Exemple : machine euclidienne gauche

A partir d'une machine euclidienne droite on peut construire une machine euclidienne gauche qui fait la même chose mais en inversant les rôles de p et q.

Etant donné un alphabet Σ, on désigne par

$$\mathit{gauche}\,(d, k, A, A', B, B', C, C', D, D')$$

l'ensemble des machines $M = (d, k, \Sigma, \Sigma', \tau)$, où, pour chaque $i \in 0..(d-1)$,

$$A, A', B, B', C_i, C'_i, D_i, D'_i \in \Sigma,$$

où

$$\Sigma' = \{C_0, C'_0, D_0, D'_0, \ldots, C_{d-1}, C'_{d-1}, D_{d-1}, D'_{d-1}\}$$

et où τ est défini par les instructions

$$\boxed{\begin{array}{l} A \to \underline{A}\lambda, \\ A' \to \underline{A}'\lambda, \\ \\ \underline{D_i} \to D_i\lambda, \\ \underline{D'_i} \to D'_i\lambda, \end{array}}$$

auxquelles on ajoute les instructions d'une machine appartenant à

$$\mathit{droit}\,(d, k, B, B', \underline{A}, \underline{A}', \underline{D}, \underline{D}', C, C'),$$

avec $i \in 0..(d-1)$, $\lambda = \#^{d-1}$ et $\# \in \Sigma$.

Si p, q, k sont des entiers naturels avec $k < d$, on voit immédiatement que

$$\boxed{\overline{M}(\ A\lambda(A'\lambda)^p B\lambda(B'\lambda)^q\ ) \ = \ C_r\lambda(C'_r\lambda)^{p'} D_r\lambda(D'_r\lambda)^{q'},} \quad (10)$$

où p', q', r sont les entiers naturels tels que

$$\boxed{\begin{array}{lcl} q' & = & qd+k, \\ p & = & p'd+r, \\ r & < & d. \end{array}}$$

3.6 Equivalence des machines de Turing et des machines à s'attraper

On se convainc facilement que l'on peut simuler le fonctionnement d'une machine à s'attraper A par une machine de Turing T. Nous allons montrer que l'inverse est aussi possible, ce qui nous permettra de conclure par :

Les machines à s'attraper ont la même puissance de calcul que les machines de Turing.

Soit une machine de Turing T construit sur l'alphabet de d éléments $\Sigma = \{a_0, a_1, \ldots, a_{d-1}\}$ avec $a_0 = \sqcup$. Une configuration quelconque de T

$$c \;=\; (Q,\, a_{p_m} \cdots a_{p_1} a_{p_0},\, a_j\, a_{q_0} a_{q_1} \cdots a_{q_n})$$

peut se coder par le triplet

$$\mu(c) = (Q_j, p, q)$$

avec

$$Q_j = (Q, j) \quad \text{et} \quad p = \sum_{i=0}^{m} p_i \times d^i \quad \text{et} \quad q = \sum_{i=0}^{n} q_i \times d^i$$

Si $c \xrightarrow{T} c'$ alors, suivant que l'instruction exécutée est de la forme

$$(Q, a_j, a_k, R, \triangleright) \tag{11}$$

ou

$$(Q, a_j, a_k, R, \triangleleft), \tag{12}$$

on introduit les entiers naturels p', q', r définis par les relations

$$p' = pd + k \quad \text{et} \quad q = q'd + r \quad \text{et} \quad r < d$$

ou

$$q' = qd + k \quad \text{et} \quad p = p'd + r \quad \text{et} \quad r < d.$$

On a alors

$$\mu(c') = (R_r, p', q').$$

Introduisons un alphabet $\overline{\Sigma}$, comportant notamment le symbole $\#$, tous les couples $Q_k = (Q, k)$, où Q est un état quelconque de T et $k \in 0..(d-1)$ et des variantes Q'_k, $\underline{Q}_k$, $\underline{Q'_k}$ de ces couples. En posant $\lambda = \#^{d-1}$, codons le triplet

$$\mu(c) = (Q_k, p, q)$$

par le mot sur $\overline{\Sigma}$:

$$\nu(c) \;=\; Q_k\, \lambda\, (Q'_k\, \lambda)^p\, \underline{Q}_k\, \lambda\, (\underline{Q'_k}\, \lambda)^q,$$

l'instruction (11) par les instructions d'une machine de type

$$\mathit{droit}\,(d, k, Q_j, Q'_j, \underline{Q}_j, \underline{Q'_j}, R, R', \underline{R}, \underline{R'})$$

et l'instruction (12) par les instructions d'une machine de type

$$\mathit{gauche}\,(d, k, Q_j, Q'_j, \underline{Q}_j, \underline{Q'_j}, R, R', \underline{R}, \underline{R'})$$

En prenant un alphabet $\overline{\Sigma}$ ayant suffisamment d'éléments, on construira ainsi une machine à s'attraper A telle que

$$c \overset{T}{\rightarrow}{}^{\star}\; c' \quad \text{ssi} \quad \nu(c) \overset{A}{\rightarrow}{}^{\star}\; \nu(c').$$

4 Machine arithmétique minimale

4.1 Machine physique

Physiquement, une machine arithmétique restreinte est composée

- d'une infinité de cases, numérotées $0, 1, 2, \ldots$, chacune pouvant contenir un entier naturel aussi grand que l'on veut,
- d'une unité centrale dans laquelle est enregistré un ensemble fixe I d'instructions, pouvant se trouver dans un ensemble fini d'états q et communiquant avec toutes les cases.

Chaque instruction est

1. soit un triplet (q, i, q') qui signifie : *si l'état de la machine est q alors augmenter de 1 le contenu de la case numéro i,*
2. soit un quadruplet (q, i, q', q'') qui signifie : *si l'état de la machine est q alors, s'il est possible de dimunuer de 1 le contenu de la case numéro i, le faire et passer à l'état q', sinon ne rien faire mais passer à l'état q''.*

Par exemple si l'ensemble d'instructions I est

$$\{(q_0, 1, q_1, q_2),\, (q_1, 0, q_0)\}$$

en partant de la configuration

$$q_0 \quad \overset{0}{\boxed{4}}\,\overset{1}{\boxed{3}}\,\overset{2}{\boxed{0}}\ \ldots$$

on obtiendra successivement

$$q_1 \quad \overset{0}{\boxed{4}}\,\overset{1}{\boxed{2}}\,\overset{2}{\boxed{0}}\ \ldots$$

$$q_0 \quad \overset{0}{\boxed{5}}\,\overset{1}{\boxed{2}}\,\overset{2}{\boxed{0}}\ \ldots$$

$$q_1 \quad \overset{0}{\boxed{5}}\,\overset{1}{\boxed{1}}\,\overset{2}{\boxed{0}}\ \ldots$$

$$q_0 \quad \overset{0}{\boxed{6}}\,\overset{1}{\boxed{1}}\,\overset{2}{\boxed{0}}\ \ldots$$

$$q_1 \quad \overset{0}{\boxed{6}}\,\overset{1}{\boxed{0}}\,\overset{2}{\boxed{0}}\ \ldots$$

$$q_0 \quad \overset{0}{\boxed{7}}\overset{1}{\boxed{0}}\overset{2}{\boxed{0}} \ \cdots$$

et finalement

$$q_2 \quad \overset{0}{\boxed{7}}\overset{1}{\boxed{0}}\overset{2}{\boxed{0}} \ \cdots$$

On a donc calculé $4 + 3 = 7$.

4.2 Machine mathématique

Une machine arithmétique restreinte est un quadruplet $M = (Q, q_0, Q', \tau)$ où

- Q, l'*ensemble des états de* M, est un ensemble fini,
- q_0, l'*état initial de* M, est un élément choisi de Q,
- Q', l'*ensemble des états finaux de* M, est un sous-ensemble de Q,
- τ, la *fonction de transition de* M, est une application de type $(Q - Q') \to ((1..n) \times Q \cup \mathbf{N} \times Q^2)$.

L'ensemble des *instructions de* M est l'ensemble des triplets et quadruplets qui sont de la forme (q, i, q') ou (q, i, q', q'') avec $q \in (Q - Q'$ et soit $(i, q') = \tau(q)$ ou $(i, q', q'') = \tau(q)$, suivant le cas.

Etant donné un mot x sur $\mathbf{N}$, c'est-à-dire un élément de $\mathbf{N}^\star$, on note $\mathrm{F}(x)$ le mot obtenu en enlevant tous les 0 qui terminent x. On note $\mathrm{F}(\mathbf{N}^\star)$ l'ensemble des $x \in \mathbf{N}^\star$ tels que $x = \mathrm{F}(x)$.

Une *configuration de* M est un doublet de la forme

$$(q, \mathrm{F}(x)),$$

avec $q \in Q$ et $x \in \mathbf{N}^\star$. Dans l'ensemble des configuration de M on introduit la relation binaire $\overset{M}{\to}$ formée de l'ensemble des couples qui sont de la forme

$$(q, \mathrm{F}(xky))) \ \overset{M}{\to} \ (q', \mathrm{F}(xly),)$$

avec $x, y \in \mathbf{N}^\star$ et $i, j \in \mathbf{N}$, et qui satisfont à l'une des trois conditions :

$$\begin{array}{llll} 1 & \tau(q) = (i, q'), & |x| = i, & l = k + 1, \\ 2 & \tau(q) = (i, q', q''), & |x| = i, & l = k - 1, \quad k > 0, \\ 3 & \tau(q) = (i, q'', q'), & |x| = i, & l = k, \quad k = 0. \end{array}$$

Ici $|x|$ désigne la longueur du mot x.

On note toujours $\overset{M^\star}{\to}$ la fermeture transitive réflexive de $\overset{M}{\to}$ et on introduit l'élément ω que l'on lit « indéfini ».

A la machine M on associe la fonction $\overline{M}$, de type $\mathbf{N}^\star \cup \{\omega\} \to \mathbf{N}^\star \cup \{\omega\}$, définie par

$$\overline{M}(x) = \begin{cases} y, & \text{si il existe } q_i \in Q' \text{ et } y \in \mathbf{N}^\star, \\ & \text{tels que } (q_0, \mathrm{F}(x)) \overset{M^\star}{\to} (q_i, y) \\ \omega, & \text{sinon.} \end{cases}$$

On dit que la machine M est *fiable* si pour aucun $x \in \mathbf{N}^\star$ on a $\overline{M}(x) = \omega$.

4.3 Machine arithmétique restreinte à deux cases

Nous allons montrer que

> *Les machines arithmétiques restreintes qui n'utilisent que deux cases ont la même puissance de calcul que les autres machines arithmétiques restreintes.*

Soit μ l'application bijective de type $\mathrm{F}(\mathbf{N}^\star) \to \mathbf{N} - \{0\}$

$$u_1 \cdots u_n \mapsto \sum_{i=1}^{n} \pi^{u_i},$$

où π_i désigne le i^{e}nombre premier : $\pi_1 = 2$, $\pi_2 = 3$, $\pi_3 = 5$, etc. Soit M une machine arithmétique restreinte quelconque. Nous allons construire une machine arithmétique restreinte M', dont les instructions (q, i, q'), (q, i, q', q'') respectent la condition $i \in \{0, 1\}$, et qui est telle que,

$$\overline{M}(x) = \mu^{-1}(\overline{M'}(\mu(x))), \tag{13}$$

pour tout $x \in \mathrm{F}(\mathbf{N}^\star)$.

Si la première machine est de la forme

$$M = (Q, q_0, Q', \tau),$$

τ étant défini par l'ensemble I d'instructions, la seconde machine sera de la forme

$$M' = (Q \cup R, q_0, Q', \tau')$$

où $Q \cup R$ et l'ensemble d'instructions I' qui définissent τ' sera obtenu en remplaçant chaque instruction de I de la forme

$$(q, i, q') \quad \text{par} \quad \begin{pmatrix} (q, 0, q_{11}, q_2), \\ (q_{11}, 1, q_{12}), \\ (q_{12}, 1, q_{13}), \\ \vdots \\ (q_{1\pi_i}, 1, q), \\ (q_2, 1, q_3, q'), \\ (q_3, 0, q_2) \end{pmatrix} \tag{14}$$

et en remplaçant chaque instruction de I de la forme

$$(q, i, q', q'') \quad \text{par} \quad \begin{pmatrix} (q, 0, q_{12}, q_5), \\ (q_{11}, 0, q_{12}, q_3), \\ (q_{12}, 0, q_{13}, q_5), \\ (q_{13}, 0, q_{14}, q_5), \\ \vdots \\ (q_{1\pi_i}, 0, q_2, q_5), \\ (q_2, 1, q_{11}), \\ (q_3, 1, q_4, q'), \\ (q_4, 0, q_3), \\ (q_5, 1, q_{61}, q''), \\ (q_{61}, 0, q_{62}), \\ (q_{62}, 0, q_{63}), \\ \vdots \\ (q_{6\pi_i}, 0, q_5). \end{pmatrix} \tag{15}$$

Bien entendu, les q_i et q_{ij} sont des nouveaux états tous distincts.

Supposons que la case numéro 1 contienne l'entier 0. La série d'instructions en (14) exprime que dans l'état q on multiplie le contenu de la case numéro 0 par π_i et qu'on passe à l'état q', avec le contenu de la case 1 égal à 0. La série d'instructions en (14) exprime que dans l'état q, suivant qu'il est possible ou qu'il n'est pas possible de diviser le contenu de la case numéro 0 par π_i, on le fait et on passe à l'état Q' avec le contenu de la case 1 égal à 0, ou on se contente de passer à l'état q'' avec le contenu de la case 1 toujours égal à 0. Il s'ensuit que, pour tout $u, v \in \mathrm{F}(\mathbf{N}^\star)$ et $q \in Q$,

$$(q_0,\, u) \overset{M^\star}{\to} (q,\, v) \quad \text{ssi} \quad (q,\, \mu(u)) \overset{M'^\star}{\to} (q_0,\, \mu(v)),$$

d'où l'égalité (13).

4.4 Equivalence avec les machines de Turing

Nous allons aussi montrer que

> *Les machines arithmétiques restreintes ont la même puissance de calcul que les machines de Turing.*

Soit M une machine arithmétique restreinte quelconque. Nous avons vu qu'il est possible de simuler le fonctionnement de M par une machine M' de même nature, mais n'utilisant que deux cases. Il est alors facile de simuler M' par une machine de Turing T.

Soit maintenant T une machine de Turing construite sur l'alphabet de d éléments $\Sigma = \{a_0, a_1, \ldots, a_{d-1}\}$ avec $a_0 = \sqcup$. Une configuration quelconque de T

$$c \;=\; (Q,\, a_{p_m} \cdots a_{p_1} a_{p_0},\, a_j\, a_{q_0} a_{q_1} \cdots a_{q_n})$$

peut se coder par le doublet

$$\mu(c) = (Q_j,\, \mathrm{F}(pq)),$$

(attention pq désigne ici un mot de longeur 2) avec

$$Q_j = (Q, j) \quad \text{et} \quad p = \sum_{i=0}^{m} p_i \times d^i \quad \text{et} \quad q = \sum_{i=0}^{n} q_i \times d^i$$

Si $c \overset{T}{\to} c'$ alors, suivant que l'instruction exécutée est de la forme

$$(Q, a_j, a_k, R, \triangleright) \tag{16}$$

ou

$$(Q, a_j, a_k, R, \triangleleft), \tag{17}$$

on introduit les entiers naturels p', q', r définis par les relations

$$p' = pd + k \quad \text{et} \quad q = q'd + r \quad \text{et} \quad r < d$$

ou

$$q' = qd + k \quad \text{et} \quad p = p'd + r \quad \text{et} \quad r < d.$$

On a alors

$$\mu(c') = (R_r,\, \mathrm{F}(p'q')).$$

Il est alors facile de construire une machine arithmétique restreinte M telle que

$$c \overset{T^\star}{\to} c' \quad \text{ssi} \quad \mu(c) \overset{M^\star}{\to} \mu(c')$$

et qui simulera donc le fonctionnement de T.

5 Machines farfelues

5.1 Jeux de la vie

En 1970 J.H. Conway a introduit le jeu suivant. On se donne une grille infinie et un « population » représentée par un ensemble de pions repartis dans les différentes cases de la grille. Cette population évolue de génération en génération selon les règles suivantes :

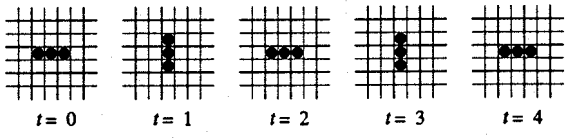
Soit une case quelconque u de la grille et soit n le nombre de pions qui se trouve dans les 8 cases voisines $v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8$:

v_1	v_2	v_3	
v_4	u	v_5	
v_6	v_7	v_8	

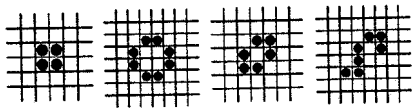
- Naissance. Si la case u est vide et si $n = 3$ alors à la génération suivante il y aura un pion dans la case u, sinon la case u restera vide.
- Survie. Si la case u n'est pas vide et si $n = 2$ ou $n = 3$, alors à la génération suivante il y aura toujours un pion das la case u.
- Mort. Si la case u n'est pas vide et si $n \neq 2$ et $n \neq 3$, alors à la génération suivante il n'y aura plus de pion das la case u.

Le passage d'une génération à l'autre s'effectue en appliquant ces règles sur toutes les cases en parallèle.

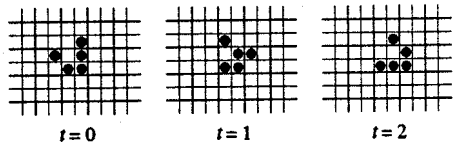
Par exemple si on part à l'instant $t = 0$ avec trois pions voisins alignés horizontalement on engendrera périodiquement les configurations suivantes :

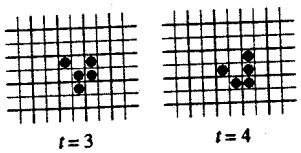


Par contre si l'on part de l'une des configuration stables suivantes :

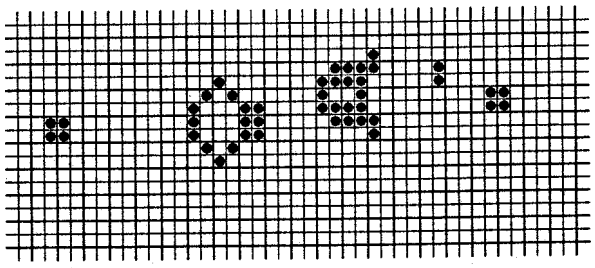


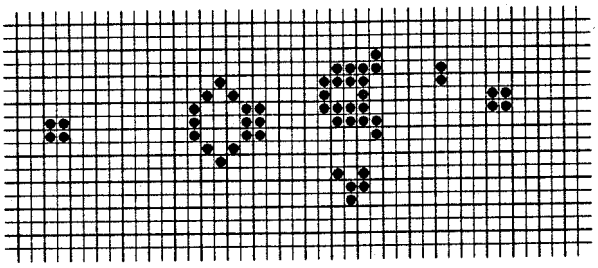
toutes les générations resterons identiques. Un cas intéressant est le « planeur » qui se propage dans l'espace sur un axe à 45 degrès (ici vers le bas et la droite) :





Un cas encore plus intéressant est le « canon à planeurs » qui toutes les 30 générations émet un planeur qui se dirige vers le bas et la droite de la grille :

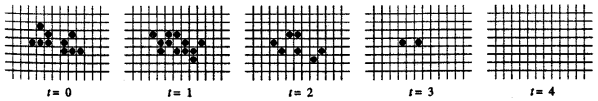




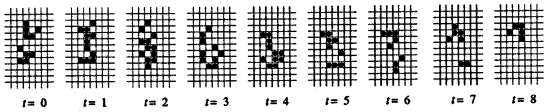
L'existence de ce canon montre qu'il existe des configurations dont la population croit infiniment.

Les canons à planeurs permettent de fabriquer des planeurs qui transportent de l'information. Cette information se manipule

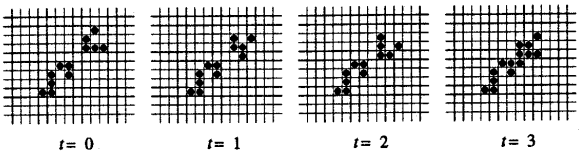
- soit, par une collision qui détruit deux planeurs,

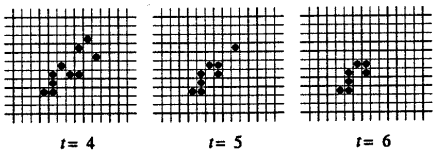


- soit, par une collision qui détruit un seul planeur,



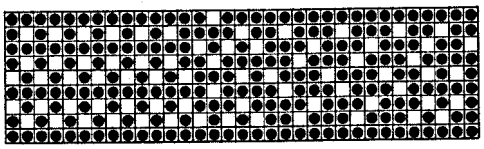
- soit, par la présence d'un obstacle qui détruit un planeur,





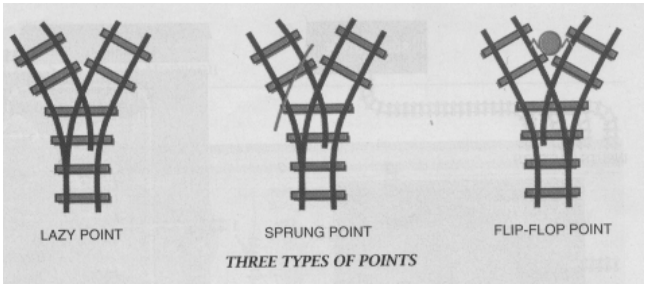
A l'aide de ces différentes configurations il est alors possible de simuler une machine de Turing, ou une machine arithmétique restreinte à deus cases.

Terminons cette section par une autre configuration difficile à construire : « un jardin d'Eden », c'est-à-dire une configuration qui ne peut être produite par aucune autre configuration :



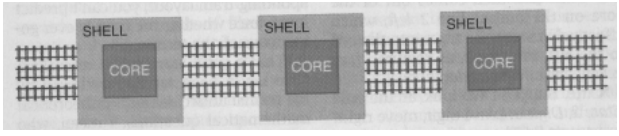
5.2 Chemin de fer

Nous allons simuler fidèlement une machine de Turing à deux symboles 0 et 1, par une locomotive parcourant un réseau de chemin de fer, faisant intervenir trois types d'aiguillages :



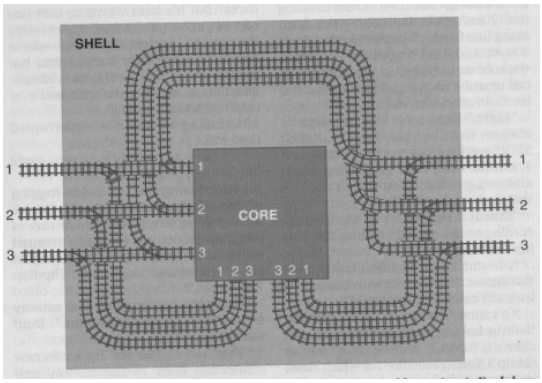
1. l'aiguillage LAZY qui mémorise la façon dont il a été pris dans le sens confluent et renvoie dans la même direction dans le sens bifurquant,
2. l'aiguillage SPRUNG qui peut être pris dans le sens confluent, mais qui envoie toujours dans la même direction dans le sens bifurquant,
3. l'aiguillage à bascule FLIP-FLOP, qui dans le sens bifurquant envoie une fois dans une direction une fois dans l'autre. Cet aiguillage n'est pas censé être pris dans le sens confluent.

Le ruban de la machine de Turing à n états non finaux sera matérialisé par n voies parallèles et chaque case sera matérialisée par une gare (shell).

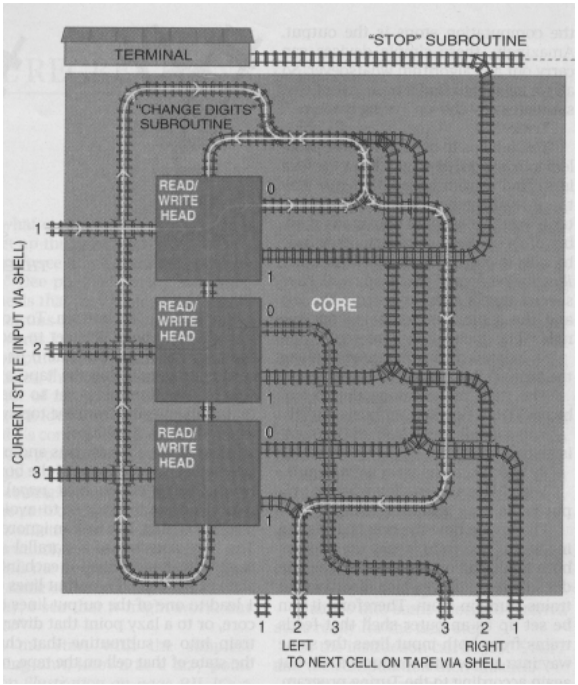


- Le fait que la machine se trouve dans l'état $q_1, \ldots, q_n$ se traduira par le fait que la locomotive se trouve sur la voie $1, \ldots, n$.
- Le fait que la tête d'écriture-lecture est positionnée sur une certaine case se traduira par le fait que la locomotive se trouve dans la gare correspondante.
- Le fait que la machine s'arrête sur une certaine case se traduira par le fait que la locomotive rentre dans le hangar terminus de la gare correspondante.
- Le fait que le symbole 0 ou 1 est écrit sur une certaine case du ruban se traduira par le fait que les n aigillages à bascule de la gare correspondante sont orientés dans un sens plutôt que dans un autre.

Voici maintenant l'architecture d'une gare (shell)



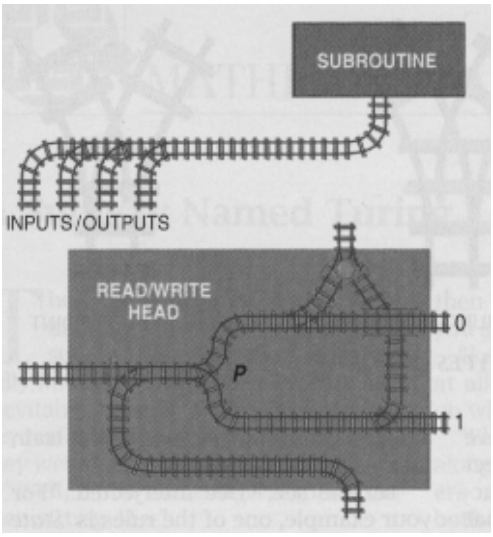
et la partie centrale de la gare (core) :



où est programmée la machine de Turing :

$$\{(q_1, 0, 1, q_2, \triangleleft), \quad (q_1, 1, 1, q_4, \triangleleft),$$
$$(q_2, 0, 0, q_3, \triangleright), \quad (q_2, 1, 0, q_2, \triangleright),$$
$$(q_3, 0, 1, q_1, \triangleright), \quad (q_3, 1, 1, q_2, \triangleright)\},$$

avec q_4, état final. Enfin voici les détails de la partie écriture-lecture (read write head) précédée du mécanisme général d'appel de sous-routines.

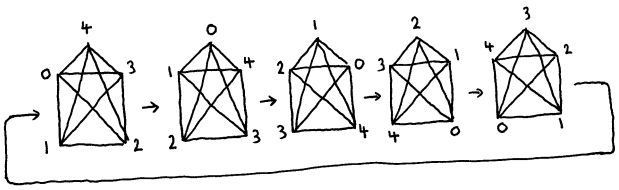


5.3 Tas de sables

On considère un graphe non orienté dont l'ensemble S de sommets est un sous-ensemble de $\mathbf{N}$. Pour chaque sommet i on désigne par V_i l'ensemble de ses sommets voisins. A l'étape $t = 0$ on associe à chaque sommet i un entier $x_i(0) \geq 0$ qui physiquement représente le nombre de grains d'un tas de sable. Ces tas de sable évoluent comme suit : si à l'étape t le nombre de grain du tas de sable i est plus grand ou égal au nombre d_i de ses voisins, alors le tas de sable i donne un grain à chacun des tas de sables voisins. On a donc, pour tout $i \in \mathbf{N}$,

$$x_i(t+1) =$$
$$x_i(t) - d_i \times (x_i(t) \geq d_i) + \sum_{j \in V_i} (x_j(t) \geq d_j)$$

où $d_i = |V_i|$ et où l'expression $(u \geq v)$ vaut 1 ou 0 suivant qu'on a ou qu'on n'a pas $u \geq v$. Voici par exemple une évolution cyclique de 5 tas de sables :



Lorsque qu'on considère un graphe linéaire, où chaque sommet a exactement 2 voisins, on obtient un phénomène de propagation gauche-droite de 02 à l'intérieur d'une suite de 1

$$\begin{array}{ll} t=0 & \ldots 11021111 \ldots \\ t=1 & \ldots 11102111 \ldots \\ t=2 & \ldots 11110211 \ldots \end{array}$$

On peut aussi propager 02 et le dupliquer dans une bifurcation marquée par un 2 :

$$\begin{array}{ll} t=0 & 1102112111 \\ & \quad\;\; 1 \\ & \quad\;\; 1 \\ \\ t=1 & 1110212111 \\ & \quad\;\; 1 \\ & \quad\;\; 1 \\ & \quad\;\; 1 \\ t=2 & 1111022111 \\ & \quad\;\; 1 \\ & \quad\;\; 1 \\ & \quad\;\; 1 \\ t=3 & 1111103111 \\ & \quad\;\; 1 \\ & \quad\;\; 1 \\ & \quad\;\; 1 \\ t=4 & 1111110211 \\ & \quad\;\; 2 \\ & \quad\;\; 1 \\ & \quad\;\; 1 \\ t=5 & 1111112021 \\ & \quad\;\; 0 \\ & \quad\;\; 2 \\ & \quad\;\; 1 \\ t=6 & 1111112102 \\ & \quad\;\; 1 \\ & \quad\;\; 0 \\ & \quad\;\; 2 \end{array}$$

A l'aide de ces configurations on peut construire un graphe dont les évolutions simulent celui d'une machine arithmétique restreinte. On a donc la puissance de calcul d'une machine de Turing.

5.4 Machine biologique par épissages

On se donne un alphabet Σ. Une règle d'épissage est un quadruplet $r = (u_1, u_2, u'_1, u'_1)$ de mots sur Σ qu'on note aussi :

$$\begin{array}{c|c} u_1 & u_2 \\ \hline u'_1 & u'_2 \end{array}$$

Etant donnés deux sous-ensemble V, W de $\Sigma^\star$ et un ensemble R de règles d'épissage, on écrit

$$V \xrightarrow{R} W$$

pour signifier que M est formé de l'ensemble des mots w pour lesquels il existe $(u_1, u_2, u'_1, u'_1) \in R$ et $v_1, v_2, v'_1, v'_2 \in \Sigma^\star$, tels que

$$v_1 u_1 u_2 v_2 \in V,\ v'_1 u'_1 u'_2 v'_2 \in V,\ v_1 u_1 u'_2 v'_2 = w.$$

Soit maintenant $(R_0, \ldots, R_{n-1})$ un n-uplet d'ensemble R_i de règles d'épissage. Etant donnés deux sous-ensemble V, W de $\Sigma^\star$ On écrit

$$V \xrightarrow{(R_0,\ldots,R_{n-1})} W \qquad (18)$$

pour signifier qu'il existe une suite $V_0, \ldots, V_k$ de sous-ensemble V_i de $\Sigma^\star$, tels que

$$V = V_0$$

$$V_i \overset{R_{f(i)}}{\to} V_{i+1}$$

$$V_n = W$$

avec $i \in 0..(k-1)$ et $f(i) = i \bmod n$.

Une machine à épissages cycliques de degré n est un quintuplet $S = (\Sigma, \Sigma_t, A, (R_0, \ldots, R_{n-1}))$, où

- Σ, l'*alphabet de* M, est un ensemble fini ayant X et Y pour éléments,
- Σ_t, l'*alphabet terminal de* M, est un sous-ensemble de Σ,
- A, *l'ensemble d'axiomes de* M, est un sous-ensemble de $\Sigma^\star$,
- chaque R_i, *un composant de* M, est un ensemble fini de règles d'épissages construites sur Σ.

A la machine à épissages cycliques M on associe la fonction $\overline{M}$, de type $\Sigma_t^\star \cup \{\omega\} \to \Sigma_t^\star \cup \{\omega\}$, définie par

$$\overline{M}(x) = \begin{cases} y, & \text{si } x \in \Sigma_t^\star \text{ et il existe } y \in \Sigma_t^\star \text{ avec} \\ & \{XxY\} \cup A \overset{(R_0,\ldots,R_{n-1})}{\longrightarrow} \{y\}, \\ \omega, & \text{sinon.} \end{cases}$$

On montre que les machines à épissage cyclique ont la même puissance de calcul que les machines de Turing.

6 TP et TD

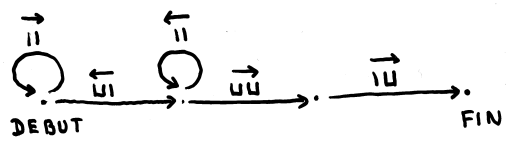
6.1 Travaux dirigés numéro 1

1 Schéma d'une machine de Turing

Une machine de Turing se schématise bien en représentant chaque état par un point et chaque instruction (q, a, a', q') par une flèche allant du point q au point q' et étiquetée par $\overleftarrow{aa'}$ ou $\overrightarrow{aa'}$ suivant que $d = \triangleleft$ ou $d = \triangleright$. Pour la machine additionneuse M qui est telle que $\overline{M}(1^m \sqcup 1^n) = 1^{m+n}$ et dont l'ensemble d'instructions est

$$\begin{array}{ll} \{(q_0, 1, 1, q_0, \triangleright), & (q_0, \sqcup, 1, q_1, \triangleleft), \\ (q_1, 1, 1, q_1, \triangleleft), & (q_1, \sqcup, \sqcup, q_2, \triangleright), \\ (q_2, 1, \sqcup, q_3, \triangleright), & (q_2, \sqcup, \sqcup, q_3, \triangleright)\} \end{array}$$

en omettant la dernière instruction, qui est inutile, on obtient le schéma :



Dessiner le schéma de la machine duplicatrice M qui est telle que $\overline{M}(x) = x \cdot x$, pour tout $x \in \{a_1, \ldots, a_n\}$, et dont les instructions sont

$$\begin{array}{ll} (q_0, \sqcup, \sqcup, q_7, \triangleright), & (q_0, a_i, a_i, q_1, \triangleright), \\ (q_1, \sqcup, \sqcup, q_6, \triangleleft), & (q_1, a_i, \sqcup, q_{2a_i}, \triangleright), \\ (q_{2a_i}, \sqcup, \sqcup, q_{3a_i}, \triangleright), & (q_{2a_i}, a_i, a_i, q_{2a_i}, \triangleright), \\ (q_{3a_i}, \sqcup, a_i, q_{4a_i}, \triangleleft), & (q_{3a_i}, a_i, a_i, q_{3a_i}, \triangleright), \\ (q_{4a_i}, \sqcup, \sqcup, q_{5a_i}, \triangleleft), & (q_{4a_i}, a_i, a_i, q_{4a_i}, \triangleleft), \\ (q_{5a_i}, \sqcup, a_i, q_1, \triangleright), & (q_{5a_i}, a_i, a_i, q_{5a_i}, \triangleleft), \\ (q_6, \sqcup, \sqcup, q_7, \triangleright), & (q_6, a_i, a_i, q_6, \triangleleft), \\ (q_7, \sqcup, \sqcup, q_{10}, \triangleleft), & (q_7, a_i, a_i, q_{8a_i}, \triangleright), \\ (q_{8a_i}, \sqcup, a_i, q_9, \triangleleft), & (q_{8a_i}, a_i, a_i, q_{8a_i}, \triangleright), \\ (q_9, \sqcup, \sqcup, q_{10}, \triangleright), & (q_9, a_i, a_i, q_9, \triangleleft), \end{array}$$

avec $i \in \{1, \ldots, n\}$. On se limitera au cas où $n = 2$ et $\{a_1, a_2\} = \{0, 1\}$. Essayer de comprendre le fonctionnement de cette machine à partir de son schéma.

2 Décalage circulaire

Dessiner une machine de Turing M d'alphabet $\{\sqcup, 0, 1\}$ telle que

$$\overline{M}(a_1 \ldots a_{n-1} a_n) = a_n a_1 \ldots a_{n-1}$$

pour tout $n \geq 1$ et $a_i \in \{0, 1\}$.

3 Mémoire circulante

Dessiner une machine de Turing M d'alphabet $\{\sqcup, 0, 1\}$ qui passe par la suite infinie de configurations :

$$(q_{a_1}, \varepsilon, a_2 a_3 \ldots a_n) \overset{M^\star}{\to}$$

$$(q_{a_n}, \varepsilon, a_1 a_2 \ldots a_{n-1}) \overset{M^\star}{\to}$$

$$\ldots$$

$$(q_{a_i}, \varepsilon, a_{i+1} \ldots a_n a_1 \ldots a_{i-1}) \overset{M^\star}{\to}$$

$$\ldots$$

avec $a_j \in \{0, 1\}$.

4 Addition binaire

Construire une machine de Turing d'aphabet $\{\sqcup, 0, 1, \#\}$ qui additionne des nombres binaires, c'est-à-dire telle que

$$\overline{M}(a_0 \ldots a_p \# b_0 \ldots b_q) = c_0 \ldots c_r$$

avec

$$\sum_{i=0}^{p} a_i 2^i + \sum_{i=0}^{q} b_i 2^i = \sum_{i=0}^{r} c_i 2^i$$

et bien entendu $a_i, b_i, c_i \in \{0, 1\}$.

6.2 Travaux pratiques numéro 1

1 Outils *Maple* pour machine de Turing

Lancer *Maple* et, par un copier-coller, insérer le contenu du fichier `Turing.txt`. Tester le fonctionnement des machines `somme` et `copie` (duplicatrice) qui y sont définies. Attention les symboles $\sqcup, 0, 1$ sont codés u, o, i. Le contenu du fichier `Turing.txt` est donné en appendice.

2 Mise au point de différentes machines

Tester la machine de Turing : (1) qui effectue un décalage circulaire, (2) qui simule une mémoire circulante, (3) qui additionne deux nombres binaires. Les caractères $\sqcup, 0, 1, \#$ seront codés u, o, i, z.

6.3 Travaux dirigés numéro 2

1 Reprise d'une démonstration intéressante

Reprendre la démonstration de la section 5 sur l'impossibilté de construire une machine de Turing qui décide de l'arrêt d'une autre machine de Turing.

2 MT Mouvement gauche

Construire une machine de Turing qui passe d'une configuration de la forme

$$(I1, a_1 \dots a_{i-1}\, z b_1 \dots b_{j-1}, b_j \dots b_n z\, a_i \dots a_m),$$

avec $a_k, \in \{o, i, u, z\}$ et $b_k \in \{o, i\}$, à la configuration

$$(H4, a_1 \dots a_{i-2}\, z b_1 \dots b_{n-1}, b_n z\, a_{i-1} \dots a_m).$$

On donnera à la fois le schéma de la machine et l'ensemble de ses instructions. A part l'état $H4$ les noms des états commenceront tous par I. Ceci facilitera l'utilisation de cette machine comme composant d'une machine universelle. Pour simplifier on supposera que $b_n = o$.

3 MT Mouvement droit

Construire une machine de Turing qui passe d'une configuration de la forme

$$(J1, a_1 \dots a_{i-1}\, z b_1 \dots b_{j-1}, b_j \dots b_n z\, a_i \dots a_m),$$

avec $a_k \in \{o, i, u, z\}$ et $b_k \in \{o, i\}$, à la configuration

$$(H4, a_1 \dots a_i\, z b_1 \dots b_{n-1}, b_n z\, a_{i+1} \dots a_m).$$

On donnera à la fois le schéma de la machine et l'ensemble de ses instructions. A part l'état $H4$ les noms des états commenceront tous par J.

4 MT Addition binaire

Poursuivre la construction d'une machine de Turing d'aphabet $\{\sqcup, 0, 1, \#\}$ qui additionne des nombres binaires, c'est-à-dire telle que

$$\overline{M}(a_0 \dots a_p \# b_0 \dots b_q) = c_0 \dots c_r$$

avec

$$\sum_{i=0}^{p} a_i 2^i + \sum_{i=0}^{q} b_i 2^i = \sum_{i=0}^{r} c_i 2^i$$

et bien entendu $a_i, b_i, c_i \in \{0, 1\}$.

6.4 Travaux pratiques numéro 2

1 MT Mouvement gauche

Faire tourner la MT Mouvement gauche sur les exemples :

```
> surtest(mvtgauche,[I1,oizuoziioii,iioiozoizu],[50,50]);
                    0, [I1, oizuoziioii, iioiozoiz]
                   31, [H4, oizuziioiiiioi, ozooiz]
> surtest(mvtgauche,[I1,oizuziiioii,iioiozoizu],[50,50]);
                    0, [I1, oizuziioii, iioiozoiz]
                   31, [H4, oizuziioiiiioi, ozioiz]
> surtest(mvtgauche,[I1,oizuzziioii,iioiozoizu],[50,50]);
                    0, [I1, oizuzziioii, iioiizoiz]
                   31, [H4, oizuziioiiiioi, izzoiz]
> surtest(mvtgauche,[I1,oizuuziioii,iioiozoizu],[50,50]);
                    0, [I1, oizuuziioii, iioiizoiz]
                   31, [H4, oizuziioiiiioi, izuoiz]
```

2 MT Mouvement droit

Faire tourner la MT Mouvement droit sur les exemples

```
> surtest(mvtdroit,[J1,oizuoziioii,iioiizooizu],[50,50]);
                    0, [J1, oizuoziioii, iioiizooiz]
                   31, [H4, oizuoziioiiiioi, izoiz]
> surtest(mvtdroit,[J1,oizuoziioii,iioiizioizu],[50,50]);
                    0, [J1, oizuoziioii, iioiizioiz]
                   31, [H4, oizuoziioiiiioi, izoiz]
> surtest(mvtdroit,[J1,oizuoziioii,iioiizzoizu],[50,50]);
                    0, [J1, oizuoziioii, iioiizzoiz]
                   31, [H4, oizuozziioiiiioi, izoiz]
> surtest(mvtdroit,[J1,oizuoziioii,iioiizuoizu],[50,50]);
                    0, [J1, oizuoziioii, iioiizuoiz]
                   31, [H4, oizuouziioiiiioi, izoiz]
```

3 MT Addition binaire

Faire tourner l'additionneur binaire.

```
> sur(sommebinaire,oiooiiziiiii);
                          iooioioi
```

6.5 Travaux dirigés numéro 3

1 Architecture de la machine universelle sur $\{u, o, i, z\}$

Prendre connaissance de l'architecture de la machine universelle U décrite ci-dessous.

Soit M une machine de Turing quelconque sur l'alphabet $\Sigma = \{u, o, i, z\}$ et soit un de ses couples : *configuration initiale, configuration finale*,

$$\boxed{q}$$
$$\dots \sqcup \underset{\downarrow}{x_1}\ x_2 \dots x_m \sqcup \dots$$

$$\boxed{q'}$$
$$\dots\ \underset{\downarrow}{x'_1}\ x'_2 \dots x'_{m'} \sqcup \dots$$

où q, q' sont des états de M et les x_i, x'_i des éléments de Σ. Sur le même alphabet Σ, on se propose de construire une machine universelle U qui admettra un couple, *configuration initiale, configuration finale*, de la forme :

$$\begin{array}{c} \boxed{r} \\ \downarrow \\ \ldots\ y_1\ y_2 \ldots y_n x_1 x_2 \ldots x_n \sqcup \ldots \end{array}$$

$$\begin{array}{c} \boxed{r'} \\ \downarrow \\ \ldots \sqcup y'_1 y'_2 \ldots y'_{n'}\ x'_1\ x'_2 \ldots x'_{m'} \sqcup \ldots \end{array}$$

où r, r' sont des états de U, les y_i, y'_i des éléments de Σ et $y_1 y_2 \ldots y_n$ est un codage de M par un mot de Σ.

A toute configuration courante de M de la forme

$$\begin{array}{c} \boxed{s} \\ \downarrow \\ \ldots \sqcup u_1 \ldots u_{i-1}\ u_i\ \ldots u_k \sqcup \ldots \end{array}$$

correspondront plusieurs configurations de U de la forme

$$\begin{array}{c} \boxed{t} \\ \downarrow \\ \ldots \sqcup u_1 \ldots u_{i-1} v_1 \ldots\ v_j\ \ldots v_l u_i \ldots u_k \sqcup \ldots \end{array} \qquad (19)$$

où s est un état de M, où t est un état de U et où les u_i, v_i sont des éléments de Σ. Le mot $v_1 \ldots v_l$ code le couple (M, s). Il est de la forme

$$z\, I_1^1\, o\, I_1^2\, o\, I_1^3\, o\, \ldots\, o\, I_{4n}^1\, o\, I_{4n}^2\, o\, I_{4n}^3\, z\, \underbrace{o \ldots o\, \overbrace{i \ldots i}^{k}}_{12n-1}\, z$$

où

- chaque instruction de M est codée par trois micro-instructions I_i^1, I_i^2, I_i^3,
- chaque micro-instruction I_i^j est un entier positif, éventuellement nul, représenté par une suite de caractères i, de longueur I_i^j,
- la suite de o et i de longueur $12n-1$ représente un compteur de micro-instruction valant k, un entier dépendant de l'état s de la machine M.

Le fonctionnement de la machine universelle U consiste, étant donné une initialisation à 0 de la valeur k du compteur de micro-instructions,

1. suivant que x_j est le caractère o, i, z ou u, à ajouter 3×0, 3×1, 3×2 ou 3×3 à k, et à remplacer k par $k-12n$, si $k \geq 12n$.
2. tant que $k \neq 0$, à décaler circulairement (de droite à gauche) les micro-instructions codées entre les deux premiers z et, à chaque passage d'une micro instruction, à diminuer k de 1,
3. lorsque $k = 0$, à continuer à décaler les trois prochaines micro-instructions I_i^1, I_i^2, I_i^3 tout en les exécutant,
4. à s'arrêter, ou à reprendre le processus en 1, avec une valeur k réinitialisée par la dernière micro-instruction exécutée I_i^3.

Plus précisément

- l'exécution de I_i^1 consiste, suivant que I_i^i vaut 0, 1, 2 ou 3, à remplacer dans la configuration (19) le caractère x_j par o, i, z ou u.
- l'exécution de I_i^2 consiste, suivant que I_i^i vaut 0 ou 1 à effectuer un mouvement gauche ou droit, c'est-à-dire à passer de la configuration (19) à l'une des configurations :

$$\begin{array}{c} \boxed{t} \\ \downarrow \\ \ldots \sqcup u_1 \ldots u_{i-2} v_1 \ldots\ v_j\ \ldots v_l u_{i-1} \ldots u_k \sqcup \ldots \end{array}$$

$$\begin{array}{c} \boxed{t} \\ \downarrow \\ \ldots \sqcup u_1 \ldots u_i v_1 \ldots\ v_j\ \ldots v_l u_{i+1} \ldots u_k \sqcup \ldots \end{array}$$

- l'éxécution de la micro-instruction I_3^1 consiste, en posant $h = 3\times I_3^1$, à s'arrêter, si $h = 12n$, et sinon, à continuer avec $k = h$.

Considérons par exemple la machine M

```
minimachine := [
[q0,o,u,q1,+1],
[q0,i,z,q0,+1],
[q0,z,i,q1,-1],
[q0,u,o,q0,+1]];
```

dont l'état initial est `q0`, l'état final `q1` et dont le nombre d'états non-finaux est $n = 1$. Compte tenu du fonctionnement de U, la suite

$$I_1^1, I_1^2, I_1^3,\ I_2^1, I_2^2, I_2^3,\ I_3^1, I_3^2, I_3^3,\ I_4^1, I_4^2, I_4^3$$

de micro-instructions intervenant dans le codage du couple $(M, \texttt{q0})$ sera

$$3,1,4,\ 2,1,2,\ 1,0,4,\ 0,1,0 \qquad (20)$$

et le codage du couple $(M, \texttt{q0})$, que l'on obtient en exécutant `code(minimachine)`, sera

$$ziiioioiiiioiioioiioioiiiiooiozooooooooooz$$

La suite (20) étant égale à la suite

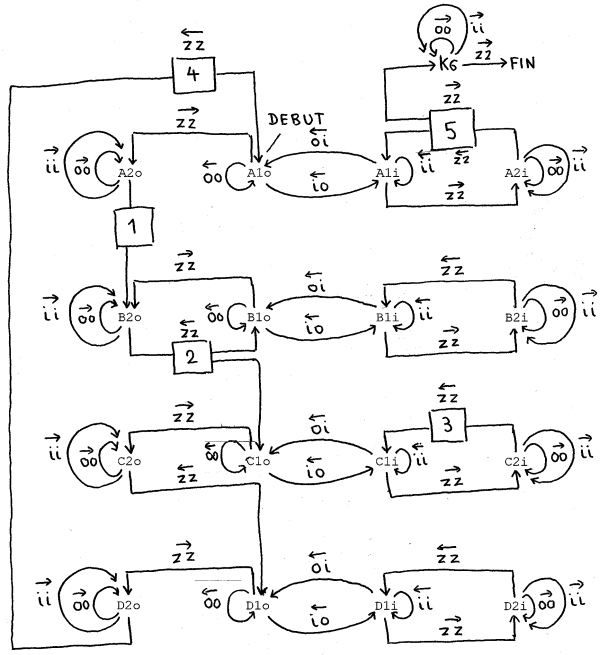
$$4,\ 0,1,0,\ 3,1,4,\ 2,1,2,\ 1,0,$$

décalée circulairement de 4 (de droite à gauche), le couple $(M, \texttt{q0})$ pourrait aussi se coder

$$ziiiiooiooiiioioiiiioiioioiioiozooooooooooz$$

L'archictecture de la machine U se présentera sous la forme de quatre boucles d'instructions A, B, C, D faisant

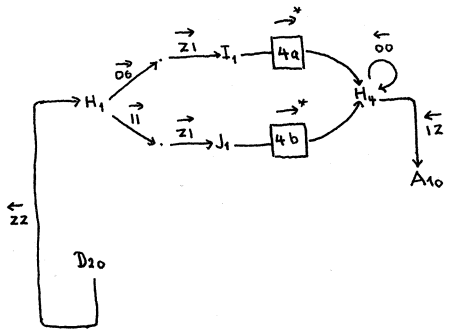
circuler les micro-instructions comme suit :



Dans la boucle B on se positionne sur la première micro-instruction I_i^1 à exécuter, dans la boucle C on décode la micro-instruction I_1^i, dans la boucle C on décode la micro-instruction I_2^i et dans la boucle A on décode la micro-instruction I_3^i. De plus on effectue les cinq actions suivantes :

1. lecture du symbole et incrémentation du compteur de micro-instructions,
2. test à zéro et décrémentation de un du compteur de micro-instructions,
3. écriture du nouveau symbole,
4. mouvement gauche ou droit,
5. arrêt ou ajout de trois au compteur de micro-instructions.

L'action 4 fait appel à l'action *mouvement gauche* 4a et l'action *mouvement droit* 4b, selon le schéma :



2 Conception du module 5 d'arrêt et d'incrémentation du compteur de micro-instructions

Dessiner le schéma de l'ensemble d'instructions constituant l'action 5 : *arrêt ou ajout de trois au compteur de micro-instructions.* Il s'agit donc de passer de la configuration

$$(\texttt{A2i}, \ldots a, \underbrace{\texttt{z}\overbrace{\texttt{o}\ldots\texttt{o}}^{3k}\underbrace{\texttt{i}\ldots\texttt{i}}_{12n-1}\texttt{z}}\ldots)$$

à l'une des configurations

$$(\texttt{A1i}, \ldots, a\texttt{z}\underbrace{\texttt{o}\ldots\texttt{o}\overbrace{\texttt{i}\ldots\texttt{i}}^{3k+3}}_{12n-1}\texttt{z}\ldots),$$

$$(\texttt{K6}, \ldots a\texttt{z}, \underbrace{\texttt{i}\ldots\texttt{i}}_{12n-1}\texttt{z}\ldots),$$

3 Conception du module 1 de lecture et d'incréméntation du compteur de micro-instructions

Dessiner le schéma de l'ensemble d'instructions constituant l'action 1 : *lecture du symbole et incrémentation du compteur de micro-instructions.*

Du fait que par la suite on remplacera le symbole lu x_j par un autre symbole, on peut le modifier. On peut donc écrire une boucle qui, tant que x_j est distinct du caractère o, ajoute 3 au compteur de micro-instructions et, suivant que x_j est i, z, ou u, remplace x_j par o, i ou z. Il ne faudra pas oublier le cas où le compteur déborde. Ceci revient donc à passer de la configuration

$$(\texttt{A2o}, \ldots a, \underbrace{\texttt{z}\overbrace{\texttt{o}\ldots\texttt{o}}^{3k}\underbrace{\texttt{i}\ldots\texttt{i}}_{12n-1}\texttt{z}}b\ldots)$$

à la configuration

$$(\texttt{B2o}, \ldots, a\texttt{z}\underbrace{\texttt{o}\ldots\texttt{o}\overbrace{\texttt{i}\ldots\texttt{i}}^{3k+3j \bmod 12n}}_{12n-1}\texttt{z}\sqcup\ldots)$$

avec $j=0$, $j=1$, $j=2$, $j=3$ suivant que $b=\texttt{o}$, $b=\texttt{i}$, $b=\texttt{z}$, $b=\sqcup$.

6.6 Travaux pratiques numéro 3

1 Architecture de la machine universelle sur $\{u, o, i, z\}$

Se familiariser avec la base de la machine universelle :

```
# ETAT INITIAL A1o, ETAT FINAL FIN
baseuniverselle := [
# BOUCLE DE DECODAGE DE L'ENTIER A TRANSFERER DANS LE COMPTEUR DE MICRO-INSTRUCTIONS
[A1o,o,o,A1o,-1], [A1o,i,o,A1i,-1], [A1o,z,z,A2o,+1],
[A1i,o,i,A1o,-1], [A1i,i,i,A1i,-1], [A1i,z,z,A2i,+1],
[A2o,o,o,A2o,+1], [A2o,i,i,A2o,+1], [A2o,z,z,E1,+1],
[A2i,o,o,A2i,+1], [A2i,i,i,A2i,+1], [A2i,z,z,K1,+1],

# BOUCLE DE POSITIONNEMENT SUR LA PREMIERE MICRO-INSTRUCTION A APPLIQUER
[B1o,o,o,B1o,-1], [B1o,i,o,B1i,-1], [B1o,z,z,B2o,+1],
[B1i,o,i,B1o,-1], [B1i,i,i,B1i,-1], [B1i,z,z,B2i,+1],
```

```
[B2o,o,o,B2o,+1], [B2o,i,i,B2o,+1], [B2o,z,z,F1,+1],
[B2i,o,o,B2i,+1], [B2i,i,i,B2i,+1], [B2i,z,z,B1i,-1],

# BOUCLE DE DECODAGE ET D'ECRITURE DU NOUVEAU SYMBOLE
[C1o,o,o,C1o,-1], [C1o,i,o,C1i,-1], [C1o,z,z,C2o,+1],
[C1i,o,i,C1o,-1], [C1i,i,i,C1i,-1], [C1i,z,z,C2i,+1],
[C2o,o,o,C2o,+1], [C2o,i,i,C2o,+1], [C2o,z,z,D1o,-1],
[C2i,o,o,C2i,+1], [C2i,i,i,C2i,+1], [C2i,z,z,G1,+1],

# BOUCLE DE DECODAGE DU MOUVEMENT A EFFECTUER
[D1o,o,o,D1o,-1], [D1o,i,o,D1i,-1], [D1o,z,z,D2o,+1],
[D1i,o,i,D1o,-1], [D1i,i,i,D1i,-1], [D1i,z,z,D2i,+1],
[D2o,o,o,D2o,+1], [D2o,i,i,D2o,+1], [D2o,z,z,H1,-1],
[D2i,o,o,D2i,+1], [D2i,i,i,D2i,+1], [D2i,z,z,D1i,-1],

# (1) LECTURE DU SYMBOLE ET INCREMENTATION DU COMPTEUR DE MICRO-INSTRUCTIONS
# A FAIRE
[E1,o,o,B2o,-1],  [E1,i,i,B2o,-1],  [E1,z,z,B2o,-1],  [E1,u,u,B2o,-1],


# (2) TEST A ZERO ET DECREMENTATION DE UN DU COMPTEUR DE MICRO-INSTRUCTIONS
# A FAIRE
[F1,o,o,F2,-1],   [F1,i,i,F2,-1],   [F1,z,z,F2,-1],  [F1,u,u,F2,-1],
                                    [F2,z,z,C1o,-1],

# (3) ECRITURE DU NOUVEAU SYMBOLE
# A FAIRE
[G1,o,o,G2,-1],   [G1,i,i,G2,-1],   [G1,z,z,G2,-1],  [G1,u,u,G2,-1],
                                    [G2,z,z,C1i,-1],

# (4) MOUVEMENTS, EN TRANSFORMANT LE 2EME z EN i PUIS A LA FIN EN RETABLISSANT z
[H1,o,o,H2g,+1],  [H1,i,i,H2d,+1],
[H2g,z,i,I1,+1],
[H2d,z,i,J1,+1],
[H3,o,o,H3,-1],   [H3,i,z,A1o,-1],

# (4a) MOUVEMENT GAUCHE, DEPART ENTRE LES z ET RETOUR JUSTE AVANT LE DERNIER z
# A FAIRE
[I1,o,o,I1,+1],   [I1,i,i,I1,+1],   [I1,z,z,H3,-1],

# (4b) MOUVEMENT DROIT, DEPART ENTRE LES z ET RETOUR JUSTE AVANT LE DERNIER z
# A FAIRE
[J1,o,o,J1,+1],   [J1,i,i,J1,+1],   [J1,z,z,H3,-1],

# (5) ARRET OU AJOUT DE TROIS AU COMPTEUR DE MICRO-INSTRUCTIONS
# A FAIRE
[K1,o,o,K2,-1],   [K1,i,i,K2,-1],   [K1,z,z,K2,-1], [K1,u,u,K2,-1],
                                    [K2,z,z,A1i,-1]
];
```

Observer ce qui ce passe lorqu'on lance des exécutions comme

```
code(minimachine);
```

et comme

```
surtest(baseuniverselle,[A1o,u,``.(code(minimachine)).oizo],
[1,2000],{[A2o,z],[B2o,z],[C2o,z],[D2o,z]});
```

2 Conception du module 5 d'arrêt et d'incrémentation du compteur de micro-instructions

Commencer à construire la machine `universelle` à partir de `baseuniverselle` en écrivant le module 5 : arrêt ou ajout de trois instructions au compteur de micro-instructions. Faire le test

```
> surtest(universelle,[A1o,u,``.(code(minimachine)).oizo],
> [1,2000],{[A2o,z],[B2o,z],[C2o,z],[D2o,z]});
    0, [A1o, u, ziiioioiiiioioioioioiiiioozooooooooooozoizo]
   31, [A2o, ziiioioiiiioioioioioiiiioo, zooooooooooozoizo]
   33, [B2o, ziiioioiiiioioioioioiiiioo, zooooooooooozoizo]
  289, [C2o, zioiiiioioioioioiiiioo1ooiii, zooooooooooozoizo]
  413, [D2o, ziiiioioioioioiiiioo1ooiii1oi, zooooooooooozoizo]
  760, [Fin, zoiioioiioioo1iiioo1ooiiioioiiizi1111111111iz, oizo]
```

3 Conception du module 1 de lecture et d'incrémentation du compteur de micro-instructions

Poursuivre la construction de la machine `universelle` en écrivant le module 1 : lecture du symbole et incrémentation du compteur de micro-instructions. Faire le test

```
> surtest(universelle,[A1o,u,``.(code(minimachine)).oizo],
> [1,2000],{[A2o,z],[B2o,z],[C2o,z],[D2o,z]});
    0, [A1o, u, ziiioioiiiioioioioioiiiioozooooooooooozoizo]
   31, [A2o, ziiioioiiiioioioioioiiiioo, zooooooooooozoizo]
   59, [B2o, ziiioioiiiioioioioioiiiioo, zooooooooooozoizo]
  315, [C2o, zioiiiioioioioioiiiioo1ooiii, zooooooooooozoizo]
  439, [D2o, ziiiioioioioioiiiioo1ooiii1oi, zooooooooooozoizo]
  786, [Fin, zoiioioiioioo1iiioo1ooiiioioiiizi1111111111iz, oizo]
```

6.7 Travaux dirigés numéro 4

1 Conception du module 2 de test à zéro et décrémentation du compteur de micro-instructions

Dessiner le schéma de l'ensemble d'instructions constituant l'action 2 : *test à zéro et décrémentation de 1 du compteur de micro-instructions.* Il s'agit donc de passer de la configuration

$$(\texttt{B2o}, \ldots a, \texttt{z}\underbrace{\texttt{o}\ldots\texttt{o}\overbrace{\texttt{i}\ldots\texttt{i}}^{k}}_{12n-1}\texttt{z}\ldots)$$

à l'une des configurations

$$(\texttt{B1o}, \ldots, a\texttt{z}\underbrace{\texttt{o}\ldots\texttt{o}\overbrace{\texttt{i}\ldots\texttt{i}}^{k-1}}_{12n-1}\texttt{z}\ldots),$$

$$(\texttt{C1o}, \ldots, a\texttt{z}\underbrace{\texttt{o}\ldots\texttt{o}}_{12n-1}\texttt{z}\ldots),$$

suivant que $k > 0$ ou $k = 0$.

2 Conception du module 3 d'écriture du nouveau symbole

Dessiner le schéma de l'ensemble d'instructions constituant l'action 3 : *écriture du nouveau symbole.* Il s'agit donc de passer de la configuration

$$(\texttt{C2i}, \ldots a, \texttt{z}\underbrace{\texttt{o}\ldots\texttt{oi}\ldots\texttt{i}}_{12n-1}\texttt{z}b\ldots)$$

à la configuration

$$(\texttt{C1i}, \ldots, a\texttt{z}\underbrace{\texttt{o}\ldots\texttt{oi}\ldots\texttt{i}}_{12n-1}\texttt{z}b'\ldots)$$

avec $b' = \texttt{i}$, $b' = \texttt{z}$, $b' = \texttt{u}$, suivant que $b = \texttt{o}$, $b = \texttt{i}$, $b = \texttt{z}$.

6.8 Travaux pratiques numéro 4 et 5

1 Conception du module 2 de test à zéro et décrémentation du compteur de micro-instructions

Poursuivre la construction de la machine `universelle` en écrivant le module (2) : test à zéro et décrémentation de un du compteur de micro-instructions. Faire les tests :

```
> surtest(universelle,[A1o,u,``.(code(minimachine)).zoizo],
> [1,2000],{[A2o,z],[B2o,z],[C2o,z],[D2o,z]});
    0, [A1o, u, ziiioioiiiioioioioioiiiioozooooooooooozzoizo]
   31, [A2o, ziiioioiiiioioioioioiiiioo, zooooooooooozzoizo]
   81, [B2o, ziiioioiiiioioioioioiiiioo, zoooooiiiiiizoizo]
  341, [B2o, zioiiiioioioioioiiiioo1ooiii, zoooooiiiiizoizo]
  479, [B2o, ziiiioioioioioioiiiioo1ooiii1, zoooooooiiiizoizo]
  805, [B2o, ziioioiioioiiiioo1ooiiioioiii, zooooooooiiizoizo]
 1009, [B2o, zioiiioioiiiioo1ooiiioioiiii1, zooooooooiizoizo]
 1153, [B2o, ziiioioiiiioo1ooiiioioiiii1oi1, zoooooooooizoizo]
 1361, [B2o, zioiiiioo1ooiiioioiiii1oioii, zoooooooooozoizo]
 1511, [C2o, ziiiioioo1iioioiiii1oioii1oi1, zoooooooooozoizo]
 1573, [D2o, ziiiioo1ooiiioioiiii1oioii1oi, zooooooooooozoizo]
 1920, [Fin, zoo1ooiiioioiiii1oioioii1ooiiizi1111111111iz, ooizo]
```

2 Conception du module 3 d'écriture du nouveau symbole

Poursuivre la construction de la machine `universelle` en écrivant le module (2) : écriture du nouveau symbole. Faire les tests :

```
> surtest(universelle,[A1o,u,``.(code(minimachine)).oizo],
> [1,2000],{[A2o,z],[B2o,z],[C2o,z],[D2o,z]});
    0, [A1o, u, ziiioioiiiioiioioiioioiiiioozooooooooooozoizo]
   31, [A2o, ziiioioiiiioiioioiioioiiiioo, zooooooooooozoizo]
   59, [B2o, ziiioioiiiioiioioiioioiiiioo, zooooooooooozoizo]
  409, [C2o, zioiiiioiioioiioioiiiiooioiii, zooooooooooozuizo]
  533, [D2o, ziiiioiioioiioioiiiiooioiiioi, zooooooooooozuizo]
  880, [Fin, zoiioioiioioiiiioooioiiioioiiizjiiiiiiiiiiz, uizo]
> surtest(universelle,[A1o,u,``.(code(minimachine)).zoizo],
> [1,2000],{[A2o,z],[B2o,z],[C2o,z],[D2o,z]});
    0, [A1o, u, ziiioioiiiioiioioiioioiiiioozooooooooooozoizo]
   31, [A2o, ziiioioiiiioiioioiioioiiiioo, zooooooooooozzoizo]
   81, [B2o, ziiioioiiiioiioioiioioiiiioo, zooooiiiiiiizooizo]
  341, [B2o, zioiiiioioioiioioiiiioooioiii, zooooooiiiiizooizo]
  479, [B2o, ziiiioiioioiioioiiiioooioiioi, zooooooiiiizooizo]
  805, [B2o, ziioioiioioiiiioooioiioioiiii, zooooooooiizooizo]
 1009, [B2o, zioiioioiiiioooioiioioiiiioii, zooooooooiizooizo]
 1153, [B2o, ziioioiiiioooioiioioiiiioiioi, zooooooooizooizo]
 1361, [B2o, zioooiiiooioiiioioiiiioiioioi, zooooooooozooizo]
 1535, [C2o, zoiiiiooioiiioioiiiioiioioi, zooooooooozoizoizo]
 1597, [D2o, ziiiiooiooiiioioiiiioioioioo, zooooooooozoizoizo]
 1944, [Fin, zooiooiiioioiiiioioioioioiiizjiiiiiiiiiiz, ioizo]
```

3 Achèvement de la machine universelle

Terminer la construction de la machine universelle en intégrant les modules (4a) et (4b) de mouvement gauche et mouvement droit. Faire les tests :

```
> surtest(universelle,[A1o,u,``.(code(minimachine)).oizo],
> [1,2000],{[A2o,z],[B2o,z],[C2o,z],[D2o,z]});
    0, [A1o, u, ziiioioiiiioiioioiioioiiiioozooooooooooozoizo]
   31, [A2o, ziiioioiiiioiioioiioioiiiioo, zooooooooooozoizo]
   59, [B2o, ziiioioiiiioiioioiioioiiiioo, zooooooooooozoizo]
  409, [C2o, zioiiiioiioioiioioiiiioooioiii, zooooooooooozuizo]
  533, [D2o, ziiiioiioioiioioiiiioooioiioi, zooooooooooozuizo]
  969, [Fin, zoiioioiioioiiiioooioiiioioiiizjiiiiiiiiiz, izo]
> surtest(universelle,[A1o,u,``.(code(minimachine)).zoizo],
> [1,2000],{[A2o,z],[B2o,z],[C2o,z],[D2o,z]});
    0, [A1o, u, ziiioioiiiioiioioiioioiiiioozooooooooooozzoizo]
   31, [A2o, ziiioioiiiioiioioiioioiiiioo, zooooooooooozzoizo]
   81, [B2o, ziiioioiiiioiioioiioioiiiioo, zooooiiiiiiizooizo]
  341, [B2o, zioiiiioioioiioioiiiioooioiii, zooooooiiiiizooizo]
  479, [B2o, ziiiioiioioiioioiiiioooioiioi, zooooooiiiizooizo]
  805, [B2o, ziioioiioioiiiioooioiioioiiii, zooooooooiizooizo]
 1009, [B2o, zioiiioioiiiioooioiioioiiiioii, zooooooooiizooizo]
 1153, [B2o, ziioioiiiioooioiioioiiiioiioi, zooooooooizooizo]
 1361, [B2o, zioooiiiooioiioioiiiioiioioi, zooooooooozooizo]
 1535, [C2o, zoiiiiooioiioioiiiioiioioi, zooooooooozoizoizo]
 1597, [D2o, ziiiiooiooiiioioiiiioioioioo, zooooooooozoizoizo]
 2000, [A21, zooiooiiioioiiio, iioioioiooiizooiiiiiiiiizuioizo]
```

Puis faire les deux tests :

```
> sur(universelle,``.(code(minimachine)).iuiuiuo);
                                    u
> sur(universelle,``.(code(minimachine)).iuiuiuz);
                                   oi
```

Enfin faire les tests finaux qui prennent de plus en plus de temps :

```
> sur(universelle,``.(code(somme)).u);
                                    u
> nbinstructions;
                                15566
> sur(universelle,``.(code(somme)).iuii);
                                  iii
> nbinstructions;
                                17460
> sur(universelle,``.(code(somme)).iiuii);
                                 iiii
> nbinstructions;
                                47666
> sur(universelle,``.(code(somme)).iiiuii);
                                iiiii
> nbinstructions;
                                77872
> sur(universelle,``.(code(somme)).iiiiuii);
                               iiiiii
> nbinstructions;
                               108078
> sur(universelle,``.(code(copie)).u);
                                    u
> nbinstructions;
                              5892265
> sur(universelle,``.(code(copie)).i);
                                   ii
> nbinstructions;
                              7773723
> sur(universelle,``.(code(copie)).io);
                                 ioio
> nbinstructions;
                             15550593
> sur(universelle,``.(code(copie)).ioo);
                                iooioo
> nbinstructions;
                             46398359
```

Bien entendu le nombre d'instructions exécutées par la machine universelle variera suivant son implantation.

6.9 Travaux dirigés numéro 5

Un *palindrome* sur un alphabet Σ est un mot sur Σ qui est vide ou de la forme

$$a_n a_{n-1} \dots a_2 a_1 a_1 a_2 \dots a_{n-1} a_n,$$

avec $n > 0$ et les a_i pris dans Σ.

Dessiner une machine de Turing M d'alphabet $\{u, o, i\}$ qui reconnaît des palindromes construits sur l'alphabet $\{o, i\}$, c'est-à-dire telle que, pour tout $x \in \{o, i\}^\star$,

$$\overline{M}(x) = \begin{cases} i, & \text{si } x \text{ est un palidrome} \\ 0, & \text{si } x \text{ n'est pas un palindrome} \end{cases}$$

et telle que le ruban final de la machine ne comporte qu'une seule case de contenu distinct de blanc (le symbole u).

Plus précisément, soit q_0 l'état initial et q_f l'état fianl de la machine : en partant de la configuration (q_0, ε, x), on atteindra la configuration (q_f, ε, i) ou (q_f, ε, o), suivant que x est ou n'est pas un palindrome.

Il est conseillé de traiter d'abords le cas où x est le palindrome vide, puis de se ramener à ce cas par une succession d'effacements du premier et du dernier symbole de x. Il faudra faire attention à ce que la tête de lecture-écriture soit bien positionnée à l'arrêt de la machine.

6.10 Examen

Tous les documents sont autorisés. Les questions sont indépendantes et peuvent donc être traitées dans n'importe quel ordre.

En ce qui concerne les machines de Turing, on rappelle qu'elles se schématisent en représentant chaque état par un point et chaque instruction (q, a, a', q') par une flèche

allant du point q au point q' et étiquetée par $\overleftarrow{aa'}$ ou $\overrightarrow{aa'}$ suivant que $d = \triangleleft$ ou $d = \triangleright$. On rapelle aussi, qu'au début du calcul, la tête de lecture-écriture de la machine est positionnée sur le premier symbole du mot constituant la donnée et, qu'à la fin du calcul, cette même tête doit être positionnée sur le premier symbole du mot qui constitue le résultat.

Question 1, notée sur 2 points

Dessiner le schéma d'une machine de Turing M d'alphabet $\{\sqcup, i\}$ qui effectue des multiplication par 2, c'est-à-dire, telle que

$$\overline{M}(\underbrace{i\dots i}_{n}) = \underbrace{i\dots i}_{2n},$$

avec $n \geq 0$.

Pour ceci on particularisera la machine duplicatrice D d'alphabet $\{\sqcup, a_1, \dots, a_n\}$ au cas où $n = 1$ et $a_n = i$. On rapelle que cette machine D est telle que

$$\overline{D}(x) = x \cdot x$$

pour tout $x \in \{a_1, \dots, a_n\}^\star$, et que ses instructions sont de la forme

$$\begin{array}{ll}
(q_0, \sqcup, \sqcup, q_7, \triangleright), & (q_0, a_i, a_i, q_1, \triangleright), \\
(q_1, \sqcup, \sqcup, q_6, \triangleleft), & (q_1, a_i, \sqcup, q_{2a_i}, \triangleright), \\
(q_{2a_i}, \sqcup, \sqcup, q_{3a_i}, \triangleright), & (q_{2a_i}, a_j, a_j, q_{2a_i}, \triangleright), \\
(q_{3a_i}, \sqcup, a_i, q_{4a_i}, \triangleleft), & (q_{3a_i}, a_j, a_j, q_{3a_i}, \triangleright), \\
(q_{4a_i}, \sqcup, \sqcup, q_{5a_i}, \triangleleft), & (q_{4a_i}, a_j, a_j, q_{4a_i}, \triangleleft), \\
(q_{5a_i}, \sqcup, a_i, q_1, \triangleright), & (q_{5a_i}, a_j, a_j, q_{5a_i}, \triangleleft), \\
(q_6, \sqcup, \sqcup, q_7, \triangleright), & (q_6, a_i, a_i, q_6, \triangleleft), \\
(q_7, \sqcup, \sqcup, q_{10}, \triangleleft), & (q_7, a_i, a_i, q_{8a_i}, \triangleright), \\
(q_{8a_i}, \sqcup, a_i, q_9, \triangleleft), & (q_{8a_i}, a_j, a_j, q_{8a_i}, \triangleright), \\
(q_9, \sqcup, \sqcup, q_{10}, \triangleright), & (q_9, a_i, a_i, q_9, \triangleleft),
\end{array}$$

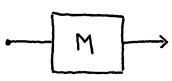
avec $i, j \in \{1, \dots, n\}$, avec q_0 état initial et q_{10} état final.

Question 2, notée sur 3 points

Supposons que la machine M qui est telle que

$$\overline{M}(\underbrace{i\dots i}_{n}) = \underbrace{i\dots i}_{2n},$$

avec $n \geq 0$, soit schématisée par



(21)

où la flèche va de l'état initial de M à son état final. En représentant la machine M par le schéma (21), dessiner le schéma d'une machine de Turing P qui calcule des puissances de 2, c'est-à-dire telle que

$$\overline{P}(\underbrace{i\dots i}_{n}) = \underbrace{i\dots i}_{2^n},$$

avec $n \geq 0$.

Pour ceci on passera du mot

$$\underbrace{i\dots i}_{n}$$

au mot

$$\underbrace{i\dots i}_{n} \sqcup i$$

puis, en se servant du fait que $2^{i+1} = 2 \times 2^i$, dans une boucle, on passera successivement aux mots

$$\underbrace{i\dots i}_{n-1} \sqcup \underbrace{i\dots i}_{2^1}$$

$$\underbrace{i\dots i}_{n-2} \sqcup \underbrace{i\dots i}_{2^2}$$

$$\underbrace{i\dots i}_{n-3} \sqcup \underbrace{i\dots i}_{2^3}$$

$$\dots$$

$$\underbrace{i\dots i}_{n-n} \sqcup \underbrace{i\dots i}_{2^n}$$

et en positionnant correctement la tête de lecture-écriture on obtiendra le mot

$$\underbrace{i\dots i}_{2^n}$$

Question 3, notée sur 3 points

Dessiner le schéma d'une machine de Turing S qui effectue des soustractions, c'est-à-dire telle que

$$\overline{S}(\underbrace{i\dots i}_{m} \sqcup \underbrace{i\dots i}_{n}) = \underbrace{i\dots i}_{|m-n|},$$

avec $m \geq 0$ et $n \geq 0$

Pour ceci on passera successivement dans une boucle aux mots

$$\underbrace{i\dots i}_{m} \sqcup \underbrace{i\dots i}_{n}$$

$$\underbrace{i\dots i}_{m-1} \sqcup \underbrace{i\dots i}_{n-1}$$

$$\underbrace{i\dots i}_{m-2} \sqcup \underbrace{i\dots i}_{n-2}$$

$$\dots$$

pour aboutir à

$$\underbrace{i\dots i}_{m-m} \sqcup \underbrace{i\dots i}_{n-m}$$

où

$$\underbrace{i\dots i}_{m-n} \sqcup \underbrace{i\dots i}_{n-n}$$

En positionnant alors correctement la tête de lecture-écriture, on obtiendra

$$\underbrace{i\dots i}_{|m-n|}$$

7 Correction des TP et TD

Confidentiel pour le moment.

8 Appendice : utilitaires *Maple*

Voici le contenu du fichier `Turing.txt` :

```
# SOUS-PROGRAMMES MAPLE POUR
# TESTER DES MACHINES DE TURING

# ALAIN COLMERAUER,
# fevrier 1999

# ------------------------------


# CREATIONS DE MACHINES

# Le symbole blanc est represente par le caractere "u".

# Creation de la machine de Turing "somme"  qui fait la
# somme de deux entiers codes par des batons.
# L'alphabet est u,o,i

somme := [
[q0,i,i,q0,+1], [q0,u,i,q1,-1],
[q1,i,i,q1,-1], [q1,u,u,q2,+1],
[q2,i,u,q3,+1]
];

# Creation de la machine "copie" qui duplique un mot
# sur l'alphabet u,o,i

copie := [
[q0,u,u,q7,+1],   [q0,o,o,q1,+1],   [q0,i,i,q1,+1],
[q1,u,u,q6,-1],   [q1,o,u,q2o,+1],  [q1,i,u,q2i,+1],
[q2o,u,u,q3o,+1], [q2o,o,o,q2o,+1], [q2o,i,i,q2o,+1],
[q2i,u,u,q3i,+1], [q2i,o,o,q2i,+1], [q2i,i,i,q2i,+1],
[q3o,u,o,q4o,-1], [q3o,o,o,q3o,+1], [q3o,i,i,q3o,+1],
[q3i,u,i,q4i,-1], [q3i,o,o,q3i,+1], [q3i,i,i,q3i,+1],
[q4o,u,u,q5o,-1], [q4o,o,o,q4o,-1], [q4o,i,i,q4o,-1],
[q4i,u,u,q5i,-1], [q4i,o,o,q4i,-1], [q4i,i,i,q4i,-1],
[q5o,u,o,q1,+1],  [q5o,o,o,q5o,-1], [q5o,i,i,q5o,-1],
[q5i,u,i,q1,+1],  [q5i,o,o,q5i,-1], [q5i,i,i,q5i,-1],
[q6,u,u,q7,+1],   [q6,o,o,q6,-1],   [q6,i,i,q6,-1],
[q7,u,u,q10,-1],  [q7,o,o,q8o,+1],  [q7,i,i,q8i,+1],
[q8o,u,o,q9,-1],  [q8o,o,o,q8o,+1], [q8o,i,i,q8o,+1],
[q8i,u,i,q9,-1],  [q8i,o,o,q8i,+1], [q8i,i,i,q8i,+1],
[q9,u,u,q10,+1],  [q9,o,o,q9,-1],   [q9,i,i,q9,-1]
];

# EXEMPLES D'EXECUTIONS

# sur(somme,iiiuiiii);
#     - calcule le mot iiiiiii

# sur(copie,ioiioi);
#     - calcule le mot  ioiioiioiioi


# surtest(somme,[q0,i,iiiuii]);
#     - execute "somme" a partir de la configuration
#       [q0,i,iiiuii],
#     - numerote toutes les configurations et les imprime.

# surtest(somme,[q0,i,iiiuii],[3,12]);
#     - execute "somme" a partir de la configuration
#       [q0,i,iiiuii],
#     - numerote toutes les configurations et imprime
#       celles qui ont
#       - des numeros égaux compris entre 3 et 12,
#     - imprime toujours la configuration initiale
#       et la derniere.

# surtest(somme,[q0,i,iiiuii],[3,12],{[q1,i],[q2,u]});
#     - execute "somme" a partir de la configuration
#       [q0,i,iiiuii],
#     - numerote toutes les configurations et imprime
#       celles qui ont
#       - des numeros compris entre 3 et 12,
#       - des couples (etat, symbole lu) dans {(q1,i),(q2,u)},
#     - imprime toujours la configuration intiale
#       et la derniere.

# ---------------------------------------------------------------------




# ENSEMBLE DE PROCEDURES A INSERER

# Initialisation

vide := ``;
taillesvidesruban := 10;
nbinstructions := 0;
o:=evaln(o);
i:=evaln(i);
z:=evaln(z);
u:=evaln(u);

fusion :=
proc(s)
   local j;
   convert([seq(convert(s[j],bytes)[1],j=1..nops(s))],bytes)
end;

configuration :=
proc(c)
   local q,r,x,y,j, debutruban,finruban;
   q := c[1];
   x := c[2];
   y := c[3];
   debutruban := 1-length(x)-taillesvidesruban;
   finruban := length(y)+taillesvidesruban;
   r := array(debutruban..finruban);
   for j from debutruban to finruban do r[j]:=u od;
   for j from 1 to length(x) do
      r[j-length(x)] := substring(x,j);
   od;
   for j from 1 to length(y) do
      r[j] := substring(y,j);
   od;
   [q,1,eval(r)];
end;

deconfiguration :=
proc(c)
   local q,k,r,x,y,l,j, debutruban,finruban;
   q := c[1];
   k := c[2];
   r := c[3];
   debutruban := op(1,op(2,eval(r)));
   finruban := op(2,op(2,eval(r)));
   for l from debutruban to k-2 do
      if r[l] <> u then break fi;
   od;
   x := [seq(r[j],j=l..k-1)];
   for l from finruban by -1 to k+1 do
      if r[l] <> u then break fi;
   od;
   y := [seq(r[j],j=k..l)];
   [q,fusion(x),fusion(y)];
end;

fonctiontransition :=
proc(Ii)
   local f,i;
   for i from 1 to nops(Ii) do
      f[Ii[i][1],Ii[i][2]] := (Ii[i][3],Ii[i][4],Ii[i][5])
   od;
   eval(f)
end;

# La fonction  sur(M,x)  a pour valeur l'execution de  M  sur le
# mot x.

sur :=
proc(M,x)
   local f, q, k, r, i, v, c;
   global nbinstructions;
   f := fonctiontransition(M);
   c := configuration([M[1][1],``,x]);
   q := c[1];
   k := c[2];
   r := c[3];
   nbinstructions := 0;
   while true do
      i := (q,r[k]);
      if not(assigned(f[i])) then break fi;
      v := f[i];
      r[k] := v[1];
      q := v[2];
      k := k+v[3];
      nbinstructions := nbinstructions+1
   od;
   deconfiguration([q,k,r])[3];
end;

# La procedure  surtest(M,c,[a,b]) detaille l'execution de M,
# en partant de la configuration c, en numérotant les configurations et en
# imprimant celles dont le numéro est compris entre a et b.
# La premiere et derniere configuration est cependant toujours imprimee meme
# si leurs numeros ne sont pas compris entre a et b.


surtest :=
proc()
   local M,c,I,S, a,b,f,q,k,r,duo,j,v,cp,n;
   M := args[1];
   f := fonctiontransition(M);
   c := args[2];
   if type(c,string) then c := [M[1][1],``,c] fi;
   cp := configuration(c);
   q := cp[1];
   k := cp[2];
   r := cp[3];
   I := [0,infini];
   S := {};
   for j from 3 to nargs do
      if type(args[j],list) then I := args[j] fi;
      if type(args[j],set) then S := args[j] fi
   od;
   a := I[1];
   b := I[2];
   n := 0;
   do
      duo := (q,r[k]);
      if n=0 or n=b or not(assigned(f[duo])) or
      (n >= a and (S={} or member([duo],S))) then
         print(n,deconfiguration([q,k,r]))
    #     print(n,[q,k,r[k],eval(r)])
      fi;
      if n=b or not(assigned(f[duo])) then break fi;
      n := n+1;
```

```
      v := f[duo];
      r[k] := v[1];
      q := v[2];
      k := k+v[3];
   od;
   ``
end;

code1 :=
proc(M)
   local q,dico,R,j,jp,k,m,n,c;
   S[o] := 0; S[i] :=1; S[z] := 2; S[u] := 3;
   dico[q] := 0;
   n := 0;
   for j from 1 to nops(M) do
      q := M[j][1];
      if(not assigned(dico[q]))then dico[q] := n; n:=n+1 fi;
   od;
   m := 4*n;
   R := array(1..m,1..3);
   for j from 1 to m do
      for k from 1 to 3 do R[j,k]:=0 od
   od;
   for j from 1 to nops(M) do
      jp:= 1+4*dico[M[j,1]]+S[M[j,2]];
      R[jp,1] := S[M[j,3]];
      R[jp,2]:= (M[j,5]+1)/2;
      if assigned(dico[M[j,4]]) then
         R[jp,3] := 4*(dico[M[j,4]]-dico[M[j,1]])-(S[M[j,2]]+1) mod m
      else
         R[jp,3] := m
      fi;
         od;
   eval(R);
end;

decode1 :=
proc(R)
   local S,T,j,k,m,v;
   S[0]:= o; S[1]:= i; S[2]:= z; S[3]:= u;
   m := op(2,op(2,eval(R))[1]);
   T := array(1..m,1..5);
   for j from 1 to m do
      T[j,1] := q.(floor((j-1)/4));
      T[j,2] := S[(j-1) mod 4];
      T[j,3] := S[R[j,1]];
      if R[j,3] = m then
         T[j,4] := q.(m/4)
      else
         v := (j+R[j,3] mod m)/4;
         if type(v,integer) then
            T[j,4] := q.v
         else
            T[j,4] := `?`
         fi
      fi;
      T[j,5] := 2*R[j,2]-1;
   od;
   eval(T)
end;

code2 :=
proc(R)
   local k,j,l,x,m;
   m := op(2,op(2,eval(R))[1]);
   x := z;
   for j from 1 to m do
      for k from 1 to 3 do
         if (j,k) <> (m,3) then x := x,seq(i,l=1..R[j,k]),o fi
      od
   od;
   x := x,seq(i,l=1..R[m,3]),z;
   x:= x,seq(o,l=1..3*m-1),z;
   fusion([x]);
end;

decode2 :=
proc(x)
   local R,j,k,l,n,m,v;
   m := 0;
   l := length(x)-1;
   while substring(x,l)=o do l:=l-1; m:=m+1 od;
   m := (m+1)/3;
   R := array(1..m,1..3);
   l := 2;
   for j  from 1 to m do
   for k  from 1 to 3 do
      v := 0;
      while substring(x,l) = i do l:=l+1; v:=v+1 od;
      l := l+1;
      R[j,k] := v;
   od od;
   R[m,3] := R[m,3]-1;
   eval(R)
end;

code :=
proc(M)
   code2(code1(M))
end;

decode :=
proc(x)
   decode1(decode2(x))
end;
```