

Machines universelles*

Cours d'Alain Colmerauer†

mai 1999

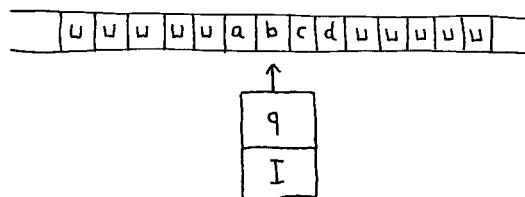
Table des matières

1	Machine de Turing	1
1.1	La machine physique	1
1.2	Monoïde libre engendré par un alphabet	2
1.3	La machine mathématique	2
1.4	Exemple : somme d'entiers codés par des bâtons	2
1.5	Exemple : duplication d'un mot	3
1.6	Machine universelle	3
2	Calculabilité et décidabilité	3
2.1	Calculabilité	3
2.2	Décidabilité et indécidabilité	4
2.3	Indécidabilité de l'arrêt d'une machine de Turing	4
2.4	Autres langages et problèmes indécidables	5
2.5	Semi-décidabilité	5
3	Machine à s'attraper (tag machine)	5
3.1	La machine physique	5
3.2	La machine mathématique	6
3.3	Exemple : somme d'entiers codés par des bâtons	6
3.4	Exemple : machine euclidienne droite	7
3.5	Exemple : machine euclidienne gauche	7
3.6	Equivalence des machines de Turing et des machines à s'attraper	8
4	Machine arithmétique minimale	8
4.1	Machine physique	8
4.2	Machine mathématique	8
4.3	Machine arithmétique restreinte à deux cases	9
4.4	Equivalence avec les machines de Turing	9
5	Machines farfelues	10
5.1	Jeux de la vie	10
5.2	Chemin de fer	11
5.3	Tas de sables	12
5.4	Machine biologique par épissages	12

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 chaque 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

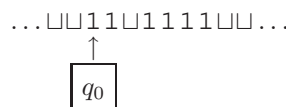
1. écrit le symbole a' à la place du symbole a ,
2. se déplace d'une case à gauche ou à droite suivant que $d = \triangleleft$ ou $d = \triangleright$,
3. passe de l'état q à l'état q' .

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.

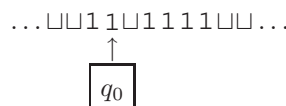
Par exemple si l'ensemble enregistré d'instructions est

$$\{(q_0, 1, 1, q_0, \triangleright), (q_0, \sqcup, 1, q_1, \triangleleft), (q_1, 1, 1, q_1, \triangleleft), (q_1, \sqcup, \sqcup, q_2, \triangleright), (q_2, 1, \sqcup, q_3, \triangleright), (q_2, \sqcup, \sqcup, q_3, \triangleright)\}$$

en partant de la configuration

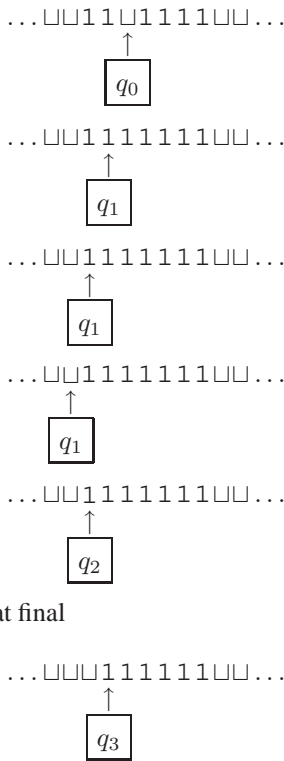


on obtiendra successivement

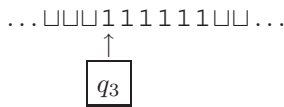


*Cours de 2^e année de DEUG (Diplôme d'études universitaires générales), option MIAS (Mathématiques, Informatique et Applications aux Sciences) et option MASS (Mathématiques Appliquées et Sciences Sociales)

†Laboratoire d'Informatique Fondamentale de Marseille, CNRS et Universités de Provence et de la Méditerranée



et donc pour résultat final



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

$$\underbrace{11\dots 1}_m + \underbrace{11\dots 1}_n = \underbrace{11\dots 1}_{m+n}$$

1.2 Monoïde libre engendré par un alphabet

Soit Σ un ensemble appelé *alphabet*. Un *mot* construit sur Σ , est une suite finie $a = a_1 a_2 \dots a_n$ 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 Σ^* .

Si $a = a_1 \dots a_m$ et $b = b_1 \dots b_n$ sont des éléments de Σ^* , alors $a \cdot b$ où ab est le mot $a_1 \dots a_m b_1 \dots b_n$. L'application $(a, b) \mapsto a \cdot b$ de $\Sigma^* \times \Sigma^*$ dans Σ^* 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 entier positif ou nul et x un élément de Σ^* , on désigne par x^n l'élément $\underbrace{x \cdot \dots \cdot x}_n$ de Σ^* . Dans le cas particulier où $n = 0$,

on considère que $x^0 = \varepsilon$.

Le couple (Σ^*, \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 $D(x)$ le mot obtenu en enlevant du début de x tous les \sqcup et $F(x)$ 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 $(Q - Q') \times \Sigma \rightarrow Q \times \Sigma \times \{\triangleleft, \triangleright\}$.

L'ensemble I 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 $q \in Q$, $x \in \Sigma^*$ et $y \in \Sigma^*$. Dans l'ensemble des configurations de M on introduit la relation binaire \xrightarrow{M} définie par

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

ssi

$$(u, v) = \begin{cases} (\varepsilon, ab'), & \text{si } (q, b, b', q', \triangleleft) \in I, \\ (ab', \varepsilon), & \text{si } (q, b, b', q', \triangleright) \in I. \end{cases}$$

pour tous $a, b, b' \in \Sigma$, $x, y \in \Sigma^*$, et $q, q' \in Q$.

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

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

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

On dit que la machine M est *fiable* si pour aucun $x \in \Sigma^*$ 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 défini par l'ensemble d'instructions

$$\{(q_0, 1, 1, q_0, \triangleright), \quad (q_0, \sqcup, 1, q_1, \triangleleft), \\ (q_1, 1, 1, q_1, \triangleleft), \quad (q_1, \sqcup, \sqcup, q_2, \triangleright), \\ (q_2, 1, \sqcup, q_3, \triangleright), \quad (q_2, \sqcup, \sqcup, q_3, \triangleright)\}$$

On a par exemple

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

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 $\Sigma = \{a_0, \dots, a_{n-1}\}$ un alphabet de n symboles avec $a_0 = \sqcup$. Voici maintenant une machine $M = (\Sigma, Q, q_0, Q', \tau)$ qui duplique les mots sur $\{a_1, \dots, a_{n-1}\}$, c'est-à-dire telle que

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

pour tout $x \in (\Sigma - \{\sqcup\})^*$. L'ensemble Q de ses états est formé des $m = 5(n-1) + 6$ éléments, de l'une des formes $q_0, q_1, q_{2a_i}, q_{3a_i}, q_{4a_i}, q_{5a_i}, q_6, q_7, q_{8a_i}, q_9, q_{10}$, avec $i \in \{1, \dots, n-1\}$. L'ensemble Q' de ses états finaux est $\{q_{10}\}$ et sa fonction de transition τ est définie par les $n(m-1)$ instructions, de l'une des 20 formes,

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

avec $i, j \in \{1, \dots, n-1\}$. Prenons $n = 3$ 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

$$\begin{array}{l} (q_0, \varepsilon, \varepsilon) \\ \rightarrow (q_7, \varepsilon, \varepsilon) \\ \rightarrow (q_{10}, \varepsilon, \varepsilon) \end{array}$$

On a

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

car

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

Dans un cas plus général, on a

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

car

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

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 $\mu : T(\Upsilon) \mapsto \Upsilon^*$ telle que,

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

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

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

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 $\mu : T(\Sigma) \mapsto \Upsilon^*$ et $\delta : \Sigma^* \mapsto \Upsilon^*$, avec δ injectives, telles que

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

pour tout $M \in T(\Sigma)$ et $x \in \Sigma^*$. 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 $n \geq 0$,

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

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 $\Sigma^* \rightarrow \Sigma^*$ 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 $x \in \Sigma^*$.

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

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

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

L'ensemble G des fonctions calculables est dénombrable, c'est-à-dire qu'il existe une application bijective

$$i \mapsto g_i \quad (3)$$

de type $\mathbf{N} \rightarrow G$, où \mathbf{N} désigne l'ensemble des entiers naturels, (positifs ou nuls). En effet, l'ensemble G_n des éléments de G qui sont représentables par une machine de n états est fini. On peut donc poser $G_n = \{f_{n1}, f_{n2}, \dots, f_{n|G_n|}\}$ et prendre pour suite infinie g_0, g_1, g_2, \dots la suite

$$f_{11}, \dots, f_{1|G_1|}, f_{21}, \dots, f_{2|G_2|}, f_{31}, \dots, f_{3|G_3|}, \dots$$

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

$$i \mapsto x_i \quad (4)$$

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

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

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

$$d : x_i \mapsto g_i(x_i) \quad (5)$$

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

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

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 $d'(x_j) = d(x_j)$. 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 $Q = \{q_0, \dots, q_{n-1}\}$. 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

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

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 $\Sigma^* \rightarrow \Sigma^*$. Par supposition et construction cette fonction d' serait calculable et telle que

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

pour tout $y \in \Sigma^*$. Du fait que d' satisfait 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 Σ^* notés *vrai* et *faux*. On s'intéresse maintenant au cas particulier des fonctions de type

$$\Sigma^* \rightarrow \{\text{vrai}, \text{faux}\}. \quad (7)$$

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

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

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

Il existe des langages dont la reconnaissance 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 $d : x_i \mapsto g_i(x_i)$ et la fonction d' qui ne peut être que la *négation* de d , définie par

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

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

Si l'on dispose 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 $\Sigma^ - L$ 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 $T(\Sigma) \rightarrow \Sigma^*$ et soit le langage

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

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*.

Montrer que la reconnaissance de L_μ est indécidable revient à montrer qu'il n'existe pas de machine de Turing $A \in T(\Sigma)$ telle que $\bar{A}(x) \in \{\text{vrai}, \text{faux}\}$ et

$$\bar{A}(\mu(M) \cdot x) = \begin{cases} \text{vrai}, & \text{si } \bar{M}(x) \neq \omega, \\ \text{faux}, & \text{si } \bar{M}(x) = \omega, \end{cases}$$

pour tout $x \in \Sigma^*$ et $M \in T(\Sigma)$.

Supposons qu'il existe une telle machine A et montrons que l'on aboutit à une contradiction. En ajoutant quelques instructions à la machine A , on peut la modifier de façon à ce qu'elle boucle au lieu de sortir *vrai*, c'est-à-dire de façon à ce que

$$\bar{A}(\mu(M) \cdot x) = \begin{cases} \omega, & \text{si } \bar{M}(x) \neq \omega, \\ \text{faux}, & \text{si } \bar{M}(x) = \omega, \end{cases}$$

Il s'ensuit que

$$\bar{A}(\mu(M) \cdot x) = \omega \text{ ssi } \bar{M}(x) \neq \omega, \quad (9)$$

pour tout $x \in \Sigma^*$ et $M \in T(\Sigma)$. Soit B la machine duplicatrice, définie à la section 1.5 et qui est telle que $\bar{B}(x) = x \cdot x$. En enchaînant les exécutions de B et de cette nouvelle machine A , on peut alors construire une machine C telle que $\bar{C}(x) = \bar{A}(\bar{B}(x))$, donc telle que, pour tout $x \in \Sigma^*$,

$$\bar{C}(x) = \bar{A}(x \cdot x). \quad (10)$$

En donnant à x la valeur $\mu(C)$ et à M la valeur C dans (9) et (10), on aboutit à la contradiction :

$$\begin{aligned} \bar{C}(\mu(C)) &= \bar{A}(\mu(C) \cdot \mu(C)), \\ \bar{A}(\mu(C) \cdot \mu(C)) &= \omega \text{ ssi } \bar{C}(\mu(C)) \neq \omega. \end{aligned}$$

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

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

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

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

Soit la fonction de codage

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

qui est de type $C \rightarrow \Sigma^*$ et soit le langage

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

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

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

dont le membre gauche est un polynôme à coefficients entiers, faisant intervenir les variables x_1, \dots, x_n et ayant au moins une solution dans \mathbf{Z} . Soit π une fonction de codage de type $P \rightarrow \Sigma^*$ et soit le langage

$$L_P = \{\pi(p) \in \Sigma^* \mid p \in P\}$$

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

Soit L un langage sur l'alphabet Σ et soit *vrai* un mot particulier sur Σ . On dit que L , ou que sa reconnaissance, est *semi-décidable*, s'il existe une machine de Turing M , d'alphabet $\Sigma \cup \{\sqcup\}$, telle que, pour tout $x \in \Sigma^*$,

$$x \in L \text{ ssi } \bar{M}(x) = \text{vrai}.$$

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 :

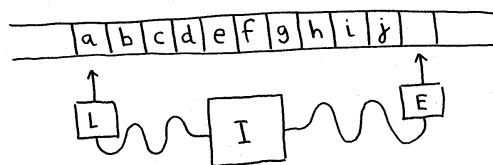
Il existe des langages indécidables qui sont semi-décidables (récursivement énumérables).

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.

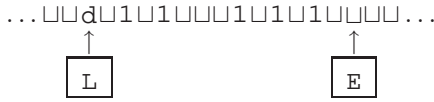
Chaque instruction est un doublet de la forme $a_0 \rightarrow a_1 \dots a_n$ et signifie : si le symbole lu est a_0 alors

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 $a_1 \dots a_n$ et se positionne n cases plus loin à droite.

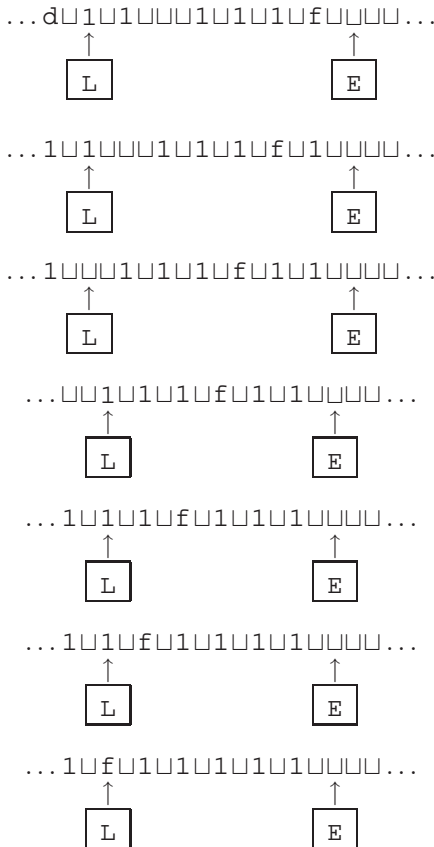
Le déroulement d’un calcul avec une telle machine consiste à partir d’un ruban initial, à exécuter tant que possible les instructions de la machine. Par exemple si $d = 2$ et que l’ensemble enregistré d’instructions est constitué de

$$\begin{aligned} d &\rightarrow f\sqcup \\ 1 &\rightarrow 1\sqcup \\ \sqcup &\rightarrow \varepsilon \end{aligned}$$

(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

$$d \underbrace{\sqcup 1 \sqcup 1 \dots 1}_{m \text{ fois } 1} \sqcup \underbrace{\sqcup 1 \sqcup 1 \dots 1}_{n \text{ fois } 1}$$

calcule

$$f \underbrace{\sqcup 1 \sqcup 1 \dots 1}_{m+n \text{ fois } 1}$$

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 $(\Sigma - \Sigma') \rightarrow \Sigma^*$.

L’ensemble des instructions de M est l’ensemble des doublets de la forme

$$a \rightarrow \tau(a),$$

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

$$a_1 \dots a_d x \xrightarrow{M} x \tau(a_1)$$

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

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

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

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

On dit que la machine M est fiable si pour aucun $x \in \Sigma^*$ 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 $M = (d, \Sigma, \Sigma', \tau)$ avec $d = 2, \Sigma = \{\sqcup, 1, d, f\}, \Sigma' = \{d, f\}$ et τ est défini par l’ensemble d’instruction

$$\{d \rightarrow f\sqcup, 1 \rightarrow 1\sqcup, \sqcup \rightarrow \varepsilon\}$$

On a par exemple

$$\begin{array}{l} d\sqcup 1\sqcup 1\sqcup \dots 1\sqcup 1\sqcup 1\sqcup \\ \xrightarrow{M} 1\sqcup 1\sqcup \dots 1\sqcup 1\sqcup f\sqcup \\ \xrightarrow{M} 1\sqcup \dots 1\sqcup 1\sqcup f\sqcup 1\sqcup \\ \xrightarrow{M} \sqcup \sqcup 1\sqcup 1\sqcup 1\sqcup f\sqcup 1\sqcup 1\sqcup \\ \xrightarrow{M} 1\sqcup 1\sqcup 1\sqcup f\sqcup 1\sqcup 1\sqcup \\ \xrightarrow{M} 1\sqcup 1\sqcup f\sqcup 1\sqcup 1\sqcup 1\sqcup \\ \xrightarrow{M} 1\sqcup f\sqcup 1\sqcup 1\sqcup 1\sqcup 1\sqcup \\ \xrightarrow{M} f\sqcup 1\sqcup 1\sqcup 1\sqcup 1\sqcup 1\sqcup \end{array}$$

Pour tous les entiers naturels m, n on a alors

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

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

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

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

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

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

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

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

$$\overline{M}(A\lambda(A'\lambda)^p B\lambda(B'\lambda)^q) = C_r \lambda(C'_r \lambda)^{p'} D_r \lambda(D'_r \lambda)^{q'}, \quad (11)$$

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

$p' = pd + k,$
$q = q'd + r,$
$r < d.$

Montrons que l'on a bien (11). Pour alléger les notations, posons $e = d-1$. On a successivement

$$\begin{aligned} & A\lambda(A'\lambda)^p B\lambda(B'\lambda)^q \\ & \xrightarrow{M} \\ & B\lambda(B'\lambda)^q E\lambda(E'\lambda)^{p'} \\ & \xrightarrow{M} \\ & E\lambda(E'\lambda)^{p'} F(F')^q \\ & \xrightarrow{M} \\ & F(F')^{q'+r} G_e \dots G_0 (G'_e \dots G'_0)^{p'} \end{aligned}$$

$$\begin{aligned} & = \\ & F \underbrace{(F' \dots F')^q}_e \underbrace{F'}_e \underbrace{F' \dots F'}_r \underbrace{G_e \dots G_{r+1}}_{e-r} \\ & \quad G_r \dots G_0 (G'_e \dots G'_0)^{p'} \\ & \xrightarrow{M} \\ & G_r \dots G_0 (G'_e \dots G'_0)^{p'} H_e \dots H_0 (H'_e \dots H'_0)^{q'} \\ & = \\ & G_r \underbrace{G_{r+1} \dots G_0}_r \underbrace{(G'_e \dots G'_{r+1})^{p'}}_{e-r} \underbrace{G'_r}_{r} \underbrace{G'_{r+1} \dots G'_0}_{r} \underbrace{H_e \dots H_{r+1}}_{e-r} \\ & \quad H_r \dots H_0 (H'_e \dots H'_0)^{q'} \\ & \xrightarrow{M} \\ & H_r \dots H_0 (H'_e \dots H'_0)^{q'} \#^{e-r} C_r \lambda(C'_r \lambda)^{p'} \\ & = \\ & H_r \underbrace{H_{r-1} \dots H_0}_r \underbrace{(H'_e \dots H'_{r+1})^{q'}}_{e-r} \underbrace{H'_r}_{r} \underbrace{H'_{r-1} \dots H'_0}_{r} \underbrace{\# \dots \#}_{e-r} \\ & \quad C_r \lambda(C'_r \lambda)^{p'} \\ & \xrightarrow{M} \\ & C_r \lambda(C'_r \lambda)^{p'} D_r \lambda(D'_r \lambda)^{q'} \end{aligned}$$

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

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

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

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

et où τ est défini par les instructions

$A \rightarrow \underline{A}\lambda,$
$A' \rightarrow \underline{A}'\lambda,$
$\underline{D}_i \rightarrow D_i \lambda,$
$\underline{D}'_i \rightarrow D'_i \lambda,$

auxquelles on ajoute les instructions d'une machine appartenant à

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

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

$$\overline{M}(A\lambda(A'\lambda)^p B\lambda(B'\lambda)^q) = C_r \lambda(C'_r \lambda)^{p'} D_r \lambda(D'_r \lambda)^{q'}, \quad (12)$$

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

$q' = qd + k,$
$p = p'd + r,$
$r < d.$

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 $\Sigma = \{a_0, a_1, \dots, a_{d-1}\}$ avec $a_0 = \sqcup$. Une configuration quelconque de T

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

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

$$(Q, a_j, a_k, R, \triangleright) \quad (13)$$

ou

$$(Q, a_j, a_k, R, \triangleleft), \quad (14)$$

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 $\bar{\Sigma}$, 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 $\bar{\Sigma}$:

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

l'instruction (13) par les instructions d'une machine de type

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

et l'instruction (14) par les instructions d'une machine de type

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

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

$$c \xrightarrow{T^*} c' \quad \text{ssi} \quad \nu(c) \xrightarrow{A^*} \nu(c').$$

4 Machine arithmétique minimale

4.1 Machine physique

Physiquement, une machine arithmétique restreinte est composée

- d'une infinité de cases, numérotées $0, 1, 2, \dots$, 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 diminuer 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

$$q_0 \quad \begin{array}{ccc} 0 & 1 & 2 \\ \boxed{4} & \boxed{3} & \boxed{0} \end{array} \dots$$

on obtiendra successivement

$$q_1 \quad \begin{array}{ccc} 0 & 1 & 2 \\ \boxed{4} & \boxed{2} & \boxed{0} \end{array} \dots$$

$$q_0 \quad \begin{array}{ccc} 0 & 1 & 2 \\ \boxed{5} & \boxed{2} & \boxed{0} \end{array} \dots$$

$$q_1 \quad \begin{array}{ccc} 0 & 1 & 2 \\ \boxed{5} & \boxed{1} & \boxed{0} \end{array} \dots$$

$$q_0 \quad \begin{array}{ccc} 0 & 1 & 2 \\ \boxed{6} & \boxed{1} & \boxed{0} \end{array} \dots$$

$$q_1 \quad \begin{array}{ccc} 0 & 1 & 2 \\ \boxed{6} & \boxed{0} & \boxed{0} \end{array} \dots$$

$$q_0 \quad \begin{array}{ccc} 0 & 1 & 2 \\ \boxed{7} & \boxed{0} & \boxed{0} \end{array} \dots$$

et finalement

$$q_2 \quad \begin{array}{ccc} 0 & 1 & 2 \\ \boxed{7} & \boxed{0} & \boxed{0} \end{array} \dots$$

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 $(Q - Q') \rightarrow (\mathbf{N} \times Q) \cup (\mathbf{N} \times Q \times Q)$.

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}^* , on note $F(x)$ le mot obtenu en enlevant tous les 0 qui terminent x . On note $F(\mathbf{N}^*)$ l'ensemble des $x \in \mathbf{N}^*$ tels que $x = F(x)$.

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

$$(q, F(x)),$$

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

$$(q, F(xky)) \xrightarrow{M} (q', F(xly)),$$

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

- 1 $\tau(q) = (i, q')$, $|x| = i$, $l = k + 1$,
- 2 $\tau(q) = (i, q', q'')$, $|x| = i$, $l = k - 1$, $k > 0$,
- 3 $\tau(q) = (i, q'', q')$, $|x| = i$, $l = k$, $k = 0$.

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

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

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

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

On dit que la machine M est *fiable* si pour aucun $x \in \mathbf{N}^*$ 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 $F(\mathbf{N}^*) \rightarrow \mathbf{N}$

$$u_1 \cdots u_n \mapsto \prod_{i=1}^n \pi_i^{u_i},$$

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,

$$\overline{M}(x) = \mu^{-1}(\overline{M'}(\mu(x))), \quad (15)$$

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

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

$$(q, i, q') \text{ par } \begin{pmatrix} (q, 0, q_{11}, q_2), \\ (q_{11}, 1, q_{12}), \\ (q_{12}, 1, q_{13}), \\ \vdots \\ (q_{1\pi_i}, 1, q), \\ (q_2, 1, q_3, q'), \\ (q_3, 0, q_2) \end{pmatrix} \quad (16)$$

et en remplaçant chaque instruction de I de la forme

$$(q, i, q', q'') \text{ par } \begin{pmatrix} (q, 0, q_{12}, q_5), \\ (q_{11}, 0, q_{12}, q_3), \\ (q_{12}, 0, q_{13}, q_5), \\ (q_{13}, 0, q_{14}, q_5), \\ \vdots \\ (q_{1\pi_i}, 0, q_2, q_5), \\ (q_2, 1, q_{11}), \\ (q_3, 1, q_4, q'), \\ (q_4, 0, q_3), \\ (q_5, 1, q_{61}, q''), \\ (q_{61}, 0, q_{62}), \\ (q_{62}, 0, q_{63}), \\ \vdots \\ (q_{6\pi_i}, 0, q_5). \end{pmatrix} \quad (17)$$

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 (16) 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 (17) 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 $u, v \in F(\mathbf{N}^*)$ et $q \in Q$,

$$(q_0, u) \xrightarrow{M^*} (q, v) \text{ ssi } (q, \mu(u)) \xrightarrow{M'^*} (q_0, \mu(v)),$$

d'où l'égalité (15).

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 d'alphabet $\{\sqcup, 0, 1\}$. Pour ceci à chaque configuration

$$c = (Q, F(pq))$$

de M' , avec p et q pris dans \mathbf{N} , on fait correspondre la configuration

$$\mu(c) = (Q, \varepsilon, \underbrace{1\dots 1}_p 0 \underbrace{1\dots 1}_q)$$

de T . Il est alors facile de définir T de façon à ce que

$$c \xrightarrow{M'^*} c' \quad \text{ssi} \quad \mu(c) \xrightarrow{T^*} \mu(c')$$

et de simuler ainsi le fonctionnement de M^* .

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

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

peut se coder par le doublet

$$\mu(c) = (Q_j, F(pq)),$$

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

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

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

$$(Q, a_j, a_k, R, \triangleright) \tag{18}$$

ou

$$(Q, a_j, a_k, R, \triangleleft), \tag{19}$$

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

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

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

$$c \xrightarrow{T^*} c' \quad \text{ssi} \quad \mu(c) \xrightarrow{M^*} \mu(c')$$

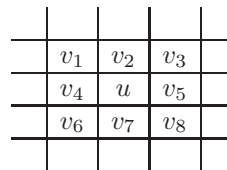
et qui simulera donc le fonctionnement de T .

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 une « 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 :

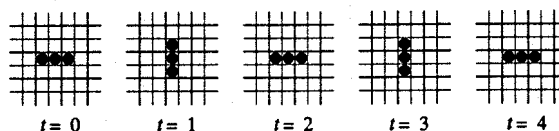
Soit une case quelconque u de la grille et soit n le nombre de pions qui se trouve dans les 8 cases voisines $v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8$:



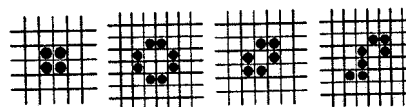
- Naissance. Si la case u est vide et si $n = 3$ alors à la génération suivante il y aura un pion dans la case u , sinon la case u restera vide.
- Survie. Si la case u n'est pas vide et si $n = 2$ ou $n = 3$, alors à la génération suivante il y aura toujours un pion dans la case u .
- Mort. Si la case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

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

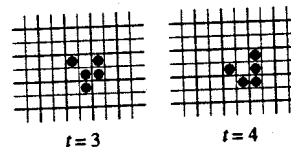
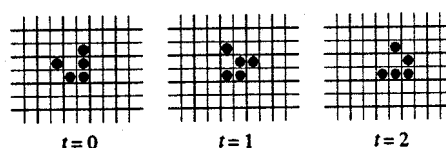
Par exemple si on part à l'instant $t = 0$ avec trois pions voisins alignés horizontalement on engendrera périodiquement les configurations suivantes :



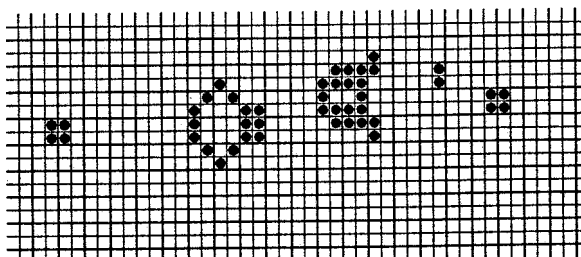
Par contre si l'on part de l'une des configurations stables suivantes :

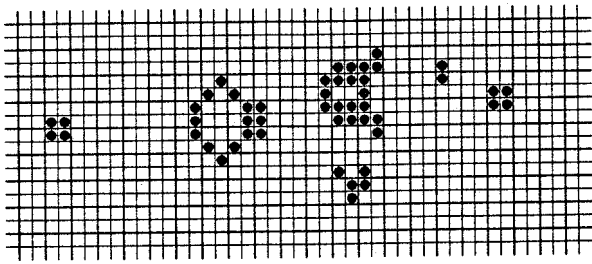


toutes les générations resteront identiques. Un cas intéressant est le « planeur » qui se propage dans l'espace sur un axe à 45 degrés (ici vers le bas et la droite) :



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 :

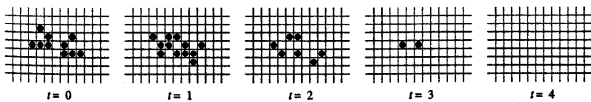




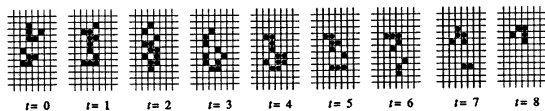
L'existence de ce canon montre qu'il existe des configurations dont la population croit infiniment.

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

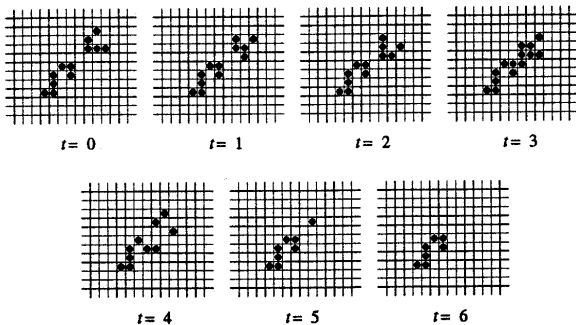
– soit, par une collision qui détruit deux planeurs,



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

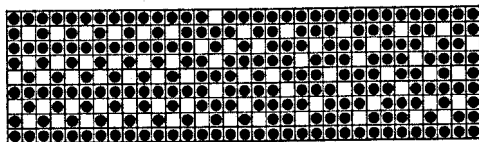


– soit, par la présence d'un obstacle qui détruit un planeur,



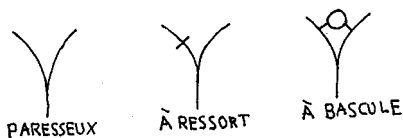
A l'aide de ces différentes configurations il est alors possible de simuler une machine de Turing, ou une machine arithmétique restreinte à deux cases.

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 :



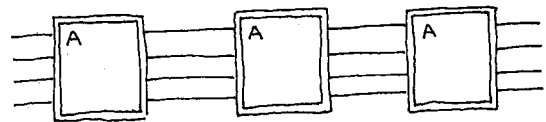
5.2 Chemin de fer

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 :



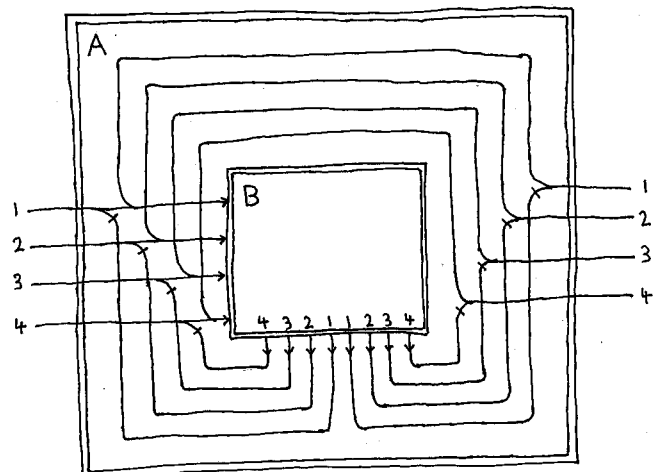
1. l'aiguillage *paresseux*, qui mémorise la façon dont il a été pris dans le sens confluent et renvoie dans la même direction dans le sens bifurquant,
2. l'aiguillage *à ressort*, qui peut être pris dans le sens confluent, mais qui envoie toujours dans la même direction dans le sens bifurquant,
3. l'aiguillage *à bascule*, 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, sera matérialisé par n voies parallèles et chaque case sera matérialisée par une gare A .

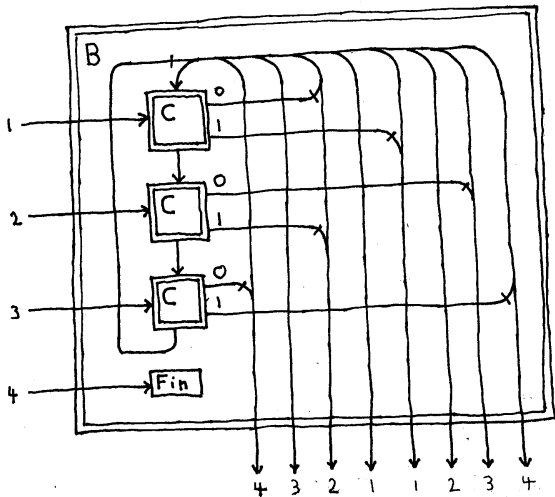


- Le fait que la machine se trouve dans l'état q_1, \dots, q_n se traduit par le fait que la locomotive se trouve sur la voie $1, \dots, n$.
- Le fait que la tête d'écriture-lecture est positionnée sur une certaine case se traduit par le fait que la locomotive se trouve dans la gare A correspondante.
- Le fait que la machine s'arrête sur une certaine case se traduit par le fait que la locomotive rentre dans le hangar terminus, marqué Fin, de la gare A 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 aiguillages à bascule au cœur de la gare correspondante sont orientés dans un sens plutôt que dans un autre.

Voici maintenant l'architecture d'une gare A ,



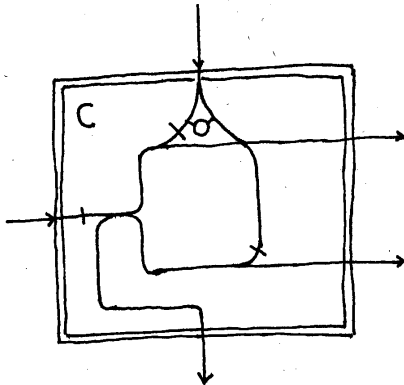
et son cœur B ,



où est programmée la machine de Turing,

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

avec q_4 , état final. Enfin voici les détails des parties C de B , qui jouent le rôle de tête d'écriture-lecture.



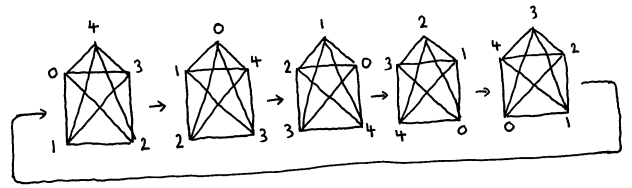
5.3 Tas de sables

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

$$t = 0 \quad \dots 11021111 \dots \\ t = 1 \quad \dots 11102111 \dots \\ t = 2 \quad \dots 11110211 \dots$$

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

$$t = 0 \quad 1102112111 \\ \quad \quad \quad 1 \\ \quad \quad \quad 1 \\ t = 1 \quad 1110212111 \\ \quad \quad \quad 1 \\ \quad \quad \quad 1 \\ \quad \quad \quad 1 \\ t = 2 \quad 1111022111 \\ \quad \quad \quad 1 \\ \quad \quad \quad 1 \\ \quad \quad \quad 1 \\ t = 3 \quad 1111103111 \\ \quad \quad \quad 1 \\ \quad \quad \quad 1 \\ \quad \quad \quad 1 \\ t = 4 \quad 1111110211 \\ \quad \quad \quad 2 \\ \quad \quad \quad 1 \\ \quad \quad \quad 1 \\ t = 5 \quad 1111112021 \\ \quad \quad \quad 0 \\ \quad \quad \quad 2 \\ \quad \quad \quad 1 \\ t = 6 \quad 1111112102 \\ \quad \quad \quad 1 \\ \quad \quad \quad 0 \\ \quad \quad \quad 2$$

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 $r = (u_1, u_2, u'_1, u'_2)$ de mots sur Σ qu'on note aussi :

$$\frac{u_1 \mid u_2}{u'_1 \mid u'_2}$$

Etant donnés deux sous-ensemble V, W de Σ^* 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 $(u_1, u_2, u'_1, u'_2) \in R$ et $v_1, v_2, v'_1, v'_2 \in \Sigma^*$, 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, \dots, R_{n-1}) un n -uplet d'ensemble R_i de règles d'épissage. Etant donné deux sous-ensemble V, W de Σ^* On écrit

$$V \xrightarrow{(R_0, \dots, R_{n-1})} W \quad (20)$$

pour signifier qu'il existe une suite V_0, \dots, V_k de sous-ensemble V_i de Σ^* , tels que

$$\begin{aligned} V &= V_0 \\ V_i &\xrightarrow{R_{f(i)}} V_{i+1} \\ V_n &= W \end{aligned}$$

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

Une machine à épissages cycliques de degré n est un quintuplet $S = (\Sigma, \Sigma_t, A, (R_0, \dots, R_{n-1}))$, 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 Σ^* ,
- 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 $\Sigma_t^* \cup \{\omega\} \rightarrow \Sigma_t^* \cup \{\omega\}$, définie par

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

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