Chapitre 14

Machines de Turing

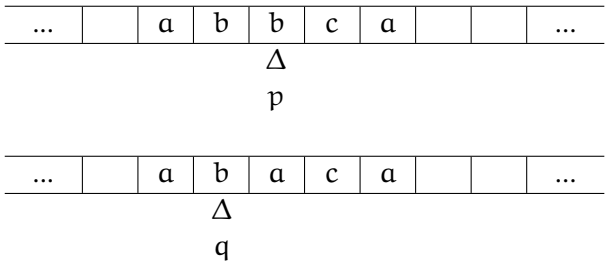
Dans ce chapitre on présente un modèle de calcul introduit dans les années 30 par Turing, les *machines de Turing*. Ces machines formalisent la notion de calculabilité. La thèse de Church affirme que tout calcul par un algorithme effectif peut-être effectué par une machine de Turing.

14.1. Définition et fonctionnement

Les automates représentent un modèle de calcul utilisant une mémoire finie correspondant à l'ensemble des états de la machine. Les machines de Turing sont également des machines "finies" (correspondant à des algorithmes effectifs), mais elles utilisent une mémoire plus élaborée. Elles utilisent un (ou des) ruban "infini" de cases mémoires qu'elles lisent et réécrivent.

Le fonctionnement d'une machine de Turing peut se représenter de manière informelle par une tête de lecture se trouvant sur un ruban qui à la lecture de la case mémoire correspondante change d'état, modifie la case mémoire et se déplace sur le ruban à droite ou à gauche

Exemple. Machine qui lisant un b dans l'état p, passe à l'état q, écrit un a à la place du b et déplace sa tête de lecture à gauche.

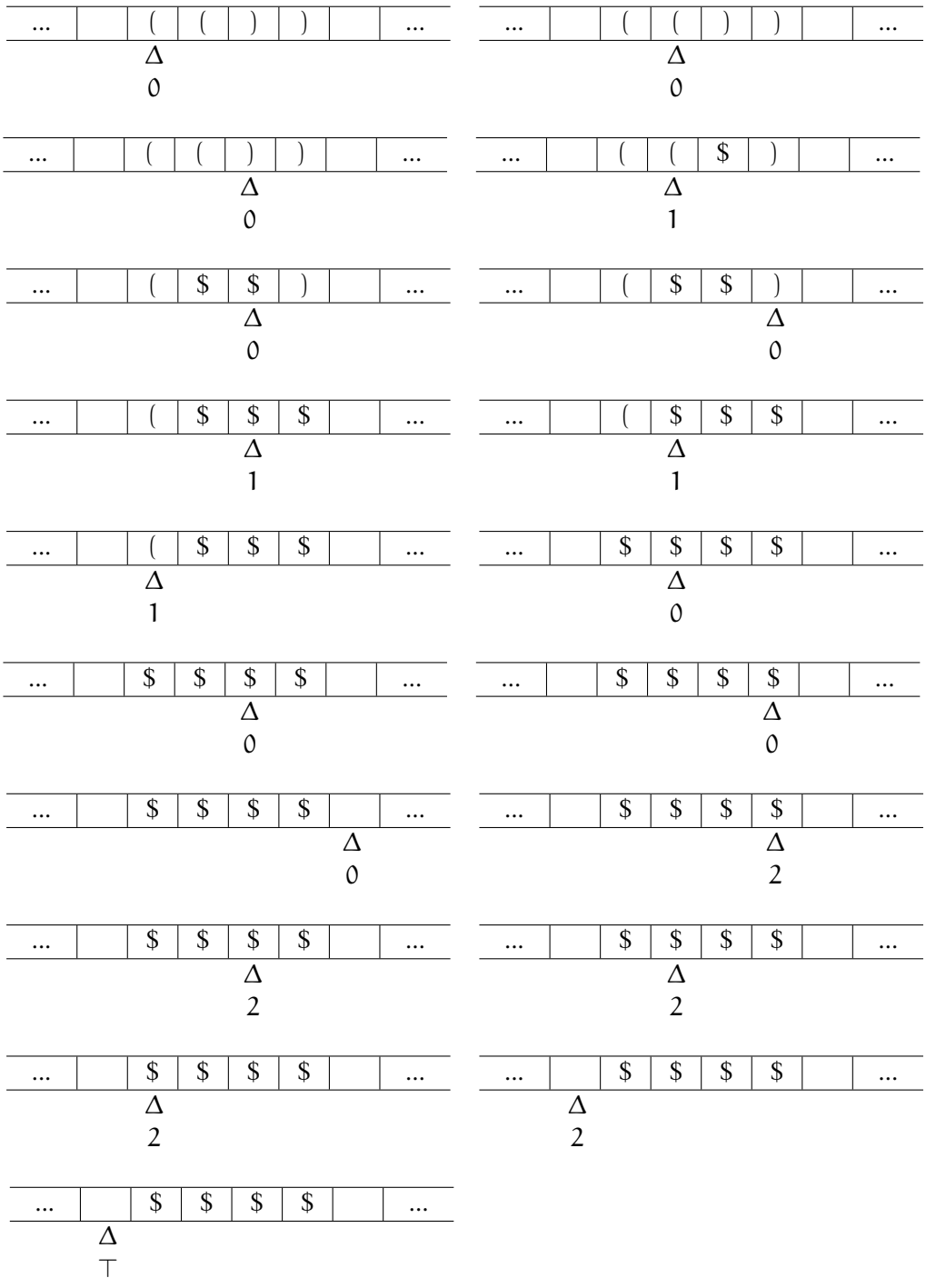


14.1.1. Définition. Une *machine de Turing* (à un ruban) est la donnée
- d'un alphabet fini Σ auquel l'on ajoute un symbole $\square$ représentant les cases vides, formant ainsi un alphabet Σ_+ ;
- d'un ensemble fini d'états Q auquel l'on ajoute deux états finaux, un état acceptant $\top$ et un état refusant $\bot$, formant ainsi un ensemble d'états Q_+ ;
- d'une fonction de changement d'état $\delta : Q \times \Sigma_+ \to Q_+$;
- d'une fonction d'écriture $\sigma : Q \times \Sigma_+ \to \Sigma_+$;
- d'une fonction de déplacement $\Delta : Q \times \Sigma_+ \to \{G, D\}$;
- d'un état initial $q_0 \in Q$.

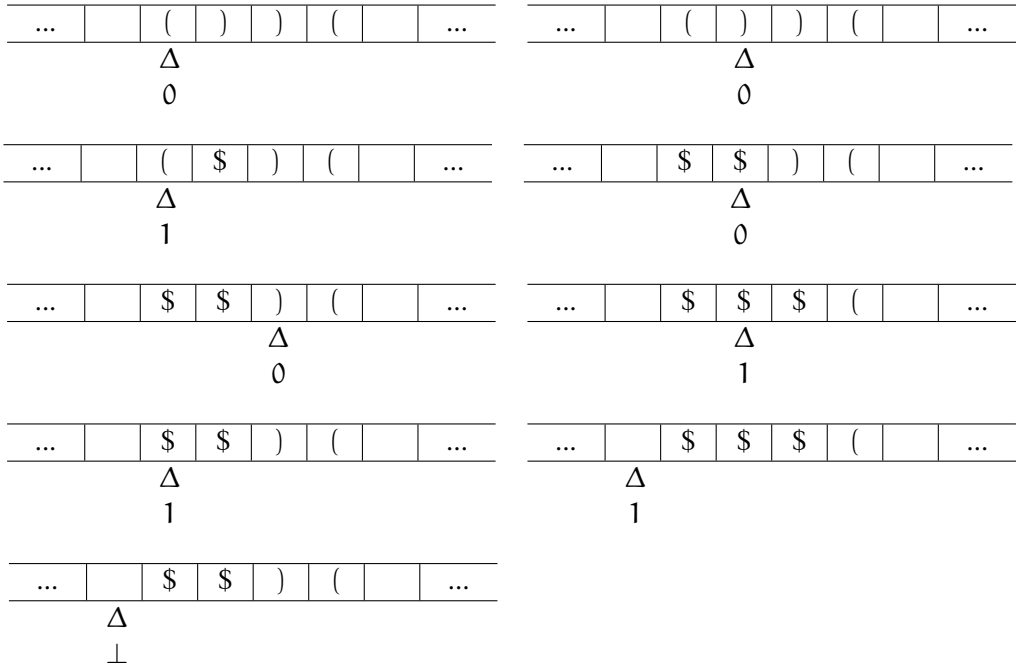
14.1.2. Exemple. Le tableau ci-dessous représente les transitions d'une machine de Turing d'alphabet $\{(,), \$\}$ et d'états $\{0, 1, 2\}$.

		(	)	□	$
→	0	0, (, D	1, $, G	2, □, G	0, $, D
	1	0, $, D	⊥	⊥	1, $, G
	2	⊥	⊥	⊤	2, $, G

Cette machine de Turing permet en fait de reconnaître le langage bien parenthésé L. Voici son fonctionnement sur le mot $(())$:



Son fonctionnement sur le mot $())($:



14.2. UTILISATION : DÉCIDABILITÉ ET CALCULABILITÉ

Afin de formaliser le fonctionnement d'une machine de Turing, on utilise la notation $u \uparrow v$ pour représenter la configuration du ruban sur lequel est écrit le mot uv avec sa tête de lecture placée au début du mot v et où toutes las autres cases du ruban sont vides. On note

$$(p, u \uparrow v) \vdash (q, u' \uparrow v')$$

la transition de l'état p avec la configuration de ruban $u \uparrow v$ à l'état q avec la configuration de ruban $(q, u' \uparrow v')$.

14.2.1. Mots acceptés, mots refusés. Soient M une machine de Turing d'alphabet Σ et u un mot sur Σ. On dit M accepte (respectivement refuse) u si le fonctionnement de M à partir de la configuration $(p_0, \uparrow u)$ mène en un nombre fini de transitions à l'état acceptant $\top$ (respectivement l'état refusant $\bot$).

Remarque. Une machine de Turing peut ni accepter, ni refuser un mot donné, mais ne pas s'arrêter de fonctionner à partir de la configuration associée à ce mot. Voici un exemple de machine qui accepte les mots formés d'un nombre impair de a mais ne s'arrête pas sur les mots formés d'un nombre pair de a.

		a	$\square$
$\rightarrow$	0	$1, a, D$	$2, a, G$
	1	$0, a, D$	$\top$
	2	$2, a, G$	$1, a, D$

14.2.2. Langages (ou ensembles) décidables et semi-décidables. Soit L un langage sur Σ. On dit que L est *décidable* s'il existe une machine de Turing d'alphabet Σ qui accepte tous les mots de L et refuse tous les mots qui ne sont pas dans L. On dit que L est *semi-décidable* s'il existe une machine de Turing qui n'accepte que les mots de L.

Exemples. Les langages rationnels sont tous décidables (un automate fini est une machine de Turing particulière). Nous verrons plus loin que les langages algébriques sont également décidables. Le complémentaire de tout langage décidable est décidable. La langage non-algébrique $\{a^n b^n c^n : n \in \mathbb{N}\}$ est décidable.

14.2.3. Fonctions calculables. Une fonction partielle $f : \Sigma \to \Sigma$ est *calculable* s'il existe une machine de Turing d'alphabet Σ qui n'accepte que les mots du domaine de définition de f et qui à partir de la configuration $(p_0, \uparrow w)$ où w est un mot du domaine de définition termine son calcul avec le mot $f(w)$ sur son ruban.

On étend cette définition aux fonctions de plusieurs variables en représentant les éléments $(u_1, .., u_k) \in \Sigma^k$ par le mot $u_1 \square u_2 \square ... \square u_k \in \Sigma_+$.

Exemples. Toutes les fonctions calculées par des machines séquentielles sont calculables. La fonction de concaténation qui à $(u_1, u_2, ..., u_k)$ associe $u_1 u_2 .. u_k$ est calculable. La somme, le produit de nombres représentés dans une base donnée sont calculables. Pour la somme en base unaire, il suffit d'utiliser la concaténation. De manière générale, pour calculer ces fonctions il est plus pratique d'utiliser plusieurs rubans mémoires. On se convaincra ensuite que l'on peut simuler de telles machines avec une machine à un seul ruban.

14.3. GÉNÉRALISATIONS

Une généralisation naturelle des machines ci-dessus est l'utilisation de plusieurs rubans. Chaque ruban possède alors une tête de lecture et à chaque pas de calcul la transition dépend de la lecture des caractères sur chaque ruban.

14.3.1. Machines de Turing à plusieurs rubans. Une *machine de Turing* à k rubans est la donnée
- d'un alphabet fini Σ auquel l'on ajoute un symbole $\square$ représentant les cases vides, formant ainsi un alphabet Σ_+ ;
- d'un ensemble fini d'états Q auquel l'on ajoute deux états finaux, un état acceptant $\top$ et un état refusant $\bot$, formant ainsi un ensemble d'états Q_+ ;
- d'une fonction de changement d'état $\delta : Q \times \Sigma_+^k \to Q_+$;
- d'une fonction d'écriture $\sigma : Q \times \Sigma_+^k \to \Sigma_+^k$;
- d'une fonction de déplacement $\Delta : Q \times \Sigma_+^k \to \{G, S, D\}^k$ (S correspondant à l'absence de déplacement la tête de lecture) ;
- d'un état initial $q_0 \in Q$.

Notons $\sigma_1, ..., \sigma_k$ et $\Delta_1, ..., \Delta_k$ les fonctions coordonnées de σ et Δ. La transition à partir d'une configuration $(p, u_1 \uparrow a_1 v_1, u_2 \uparrow a_2 v_2, ..., u_k \uparrow a_k v_k)$ consiste donc à :
- passer à l'état $\delta(p, a_1, .., a_k)$,
- remplacer chaque a_i par $\sigma_i(p, a_1, .., a_k)$,
- déplacer la tête de lecture du i-ème ruban suivant la valeur de $\Delta_i(p, a_1, .., a_k)$.

14.3.2. Exemples.

1. **Automates à pile** : Les automates à piles déterministes peuvent être représentées par des machines à deux rubans, le second ruban utilisé par la pile.
2. **Addition en représentation binaire** : on considère une machine de Turing à deux rubans. Cette machine commence par placer ses têtes de lectures à la fin des représentations binaires de deux nombres placés sur ces rubans. Ensuite elle fonctionne comme l'algorithme classique de l'addition

posée. Elle lit les chifres sur chaque ruban, les additionnent avec la retenue et écrit le résultat correspondant sur le premier ruban et utilise son état pour conserver la retenue qui vaut 0 ou 1. (Rappelons que ceci peut également s'effectuer par une machine séquentielle).

3. **Multiplication en représentation binaire** : on utilise une machine à trois rubans. Le premier sert au résultat, les deux autres aux donnés. On simule alors l'algorithme clssique de la multiplication posée, en utilisant l'addition à chaque étape.
4. **Passage d'une base à une autre pour la réprésentation d'un nombre** : il suffit pour cela de passer par la base 1. Le passage de la base 1 à la base n consiste à itérer la division euclidienne par n. On compte le nombre de bloc de 1 de longueur n sur un ruban auxiliaire. Le reste se lit alors directement sur le premier ruban que l'on écrit sur un ruban résultat. On itère ensuite sur le quotient que l'on a recopié sur le premier ruban. Le passage de la base n à 1 consiste à représenter le premier chiffre en base n par un bloc de 1 correspondant sur le ruban résultat, puis à dupliquer n fois le nombre de 1 sur le ruban résultat avant de passer au chiffre suivant.

14.3.3. Simulation d'une Machine de Turing à 2 ubans par une machine de Turing à un seul ruban. Nous allons voir rapidement que le calcul d'une machine de Turing à plusieurs rubans peut-être simulé par une machine de Turing à un seul ruban. Faisons le cas de deux rubans. Le principe est de coderles configurations sur deux rubans par une configuration sur un seul ruban. Pour cela on peut par exemple utiliser alternativement deux cases mémoires du ruban de la nouvelle machine pour chacune des cases des rubans de la machine de départ, de la manières suivante.

Configuration d'une machine à 2 rubans :

...	a	b	c	c	b	...
		Δ				

...	c	a	a	a	a	...
				Δ		
				p		

Codage de cette configuration sur un ruban :

...		a		c	Δ	b		a		c		a		c	Δ	a		b		a	...
					Δ																
					p																

Dans la nouvelle machine, le ruban est décomposé par bloc de 4 cases, la première case codant le fait que la tête de lecture du premier ruban originel se trouve ici ou non, la seconde la case mémoire correspondant à celle du premier ruban originel et de même les deux cases suivantes pour le second ruban originel. Le fonctionnement de la nouvelle machine à ruban consistera à détecter les cases à lire en utilisant les marqueurs des têtes de lecture puis à simuler les transitions sur les deux rubans de la machine originelle.

Remarques. La simulation ci-dessus se généralise facilement à k rubans. Par ailleurs si l'on considère un calcul sur une machine à k-rubans partant de la configuration $(\uparrow u_1, \uparrow u_2, ..., \uparrow u_k)$ il est facile de voir via la simulation ci-dessus que l'on peut faire le même calcul avec une machine à un ruban partant de la configuration $\uparrow u_1 \square u_2 \square ... \square u_k$. Donc tous les exemples de fonctions calculées par des machines à plusieurs rubans sont calculables au senn de la définition donnée précedemment.

14.3.4. Machines de Turing non-déterministes. Une autre généralisation naturelle des machines de Turing est d'avoir un nombre fini de transitions possibles à chaque pas de calcul. Il est assez facile de montrer que de telles machines peuvent être simulées par des machines déterministes. Pour cela on

énumère au fur et à mesure l'ensemble des configurations possibles sur un ruban auxiliaire. A l'aide de telle machine, on remarque par exemple facilement que tout automate à pile (non nécessairement déterministe) peut-être représenté par une machine de Turing. En particulier les langages algébriques sont bien décidables (au sens des machines de Turing).

14.4. EXEMPLES ET CONTRE-EXEMPLES

14.4.1. Réduction de l'alphabet. En pratique la mémoire physique dans une machine correspond à des bits. Il est assez simple de voir que toute machine de Turing peut être simulée par une machine de Turing utilisant un ruban constitué de bits, c'est-à-dire d'alphabet $\Sigma_+ = \{\square, 1\}$ que l'on notera plus en général $\Sigma_+ = \{0, 1\}$.

Pour cela il suffit de remarquer que tout ensemble fini se code par un nombre fini d'entiers ayant des représentations binaire d'une longueur fixée. Prenons par exemple une machine de Turing M d'alphabet $\Sigma = \{a, b, c\}$ que l'on codera par des représentations binaires de longueur 2 :

$$\begin{array}{rcl} \square & : & 00 \\ a & : & 01 \\ b & : & 10 \\ c & : & 11 \end{array}$$

Ainsi à chaque configuration de ruban de M on associe une configuration d'une machine simulante M' en précisant que l'on place la tête de lecture sur la première des deux cases représentant l'élément de l'alphabet de M. Puis M' simule la transition définie par M en lisant la case suivante afin de déterminer le caractère lu par M. Voici un exemple :

Configuration de M (état p et lecture de a) :

...		b		a	c	a		...
				Δ				
				p				

Représentation sur le ruban de M' :

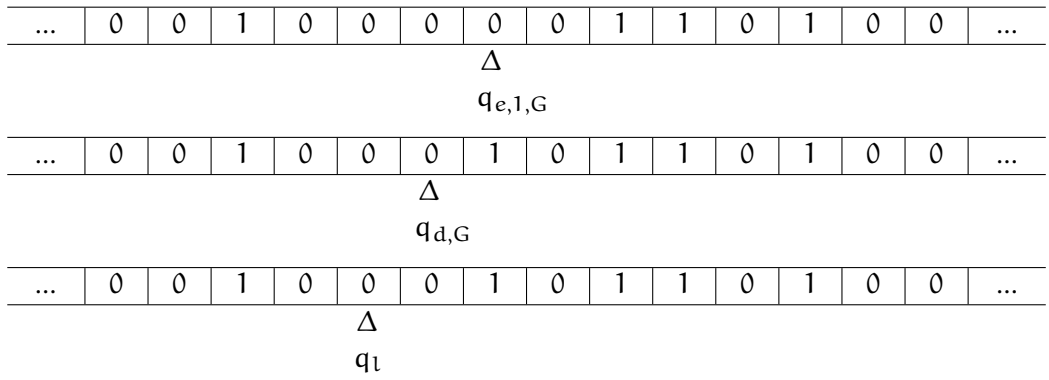
...	0	0	1	0	0	0	0	1	1	1	0	1	0	0	...
							Δ								
							p_1								

Transition de M : passage à l'état q, écriture de b et déplacement à gauche.

...		b		b	c	a		...
			Δ					
			q					

Simulation par M' en utilisant des états auxiliaires :

...	0	0	1	0	0	0	0	1	1	1	0	1	0	0	...
								Δ							
								$p_{1,0}$							



On remarque ainsi qu'à des codages simples près (codage calculables par des machines séquentielles) toute fonction calculable peut être calculée par une machine de Turing à un seul ruban d'alphabet $\Sigma_+ = \{0, 1\}$ où 0 est utilisé pour représenter les cases mémoires vides.

14.4.2. Machine de Turing universelle. Les exemples de machine de Turing que nous avons vu précédemment ne sont pas programmables ; elles effectuent un calcul prédéterminé. Nous allons maintenant décrire une machine de Turing "programmable" qui prendra en entrée sur un ruban le code d'une machine de Turing M d'alphabet $\{0, 1\}$ et simulera sur un autre ruban le fonctionnement de cette machine.

Notre machine universelle U sera constitué de trois rubans.

Premier ruban : sur ce ruban on simulera le calcul de la machine M sur un mot sur l'alphabet $\{0, 1\}$.

Second ruban : sur ce ruban on mettra sous forme codé l'ensemble des transitions de M ce qui correspondra en quelque sorte au "programme M".

Troisième ruban : On mettra le code correspondant à l'état courant de M.

Pour cela il est nécessaire de coder le tableau des transitions de M (ce code correspondra au "programme" qui sera écrit sur le premier ruban d'une machine de Turing, *machine universelle*. Nous supposerons que les états sont représentés par des entiers que l'état acceptant correspond à 1 et l'état refusant à 0. On peut alors représenter une transition $(p, l) \to (q, e, \Delta)$ de l'état $p \in \mathbb{N}$ avec la lecture de $l \in \{0, 1\}$ qui passe à l'état $q \in \mathbb{N}$, écrit $e \in \{0, 1\}$ et effectue le déplacement $\Delta \in \{G, D\}$ par

$$\#\text{bin}(p)\#\ l\ \#\text{bin}(q)\#\ e\Delta$$

où $\text{bin}(p)$ et $\text{bin}(q)$ sont les représentations binaires des entiers p et q.

On représente ensuite l'ensemble des transitions de M par

$$\$T_1T_2T_3T_4...T_n\$.$$

Notre machine universelle U est d'alphabet $\{0, 1, G, D, \$, \#\}$. À partir de la donnée de l'ensemble des transitions de M sur le second ruban, de la représentation binaire de l'état initial de M sur le troisième ruban et de la donnée d'un mot sur le premier ruban, la machine U simule le calcul de M de la façon suivante :

1. A la lecture de 0 ou 1 sur le premier ruban elle recherche en lisant sons troisième ruban la transition correspondante au calcul par M.
2. Une fois cette transition trouvée, elle simule cette transition en écrivant sur son troisième ruban l'état d'arrivée, en écrivant le caractère 0 ou 1 defini par la transition sur son premier ruban et en déplaçant la tête de lecture sur son premier ruban suivant toujours cette transition. Dans le cas où l'état d'arrivée est 0 (respectivement 1) elle se met dans son état refusant (respectivement acceptant) et s'arrête.

Remarque. Les choix ci-dessus du codage, de l'alphabet pour cela et du nombre de rubans utilisés ne sont qu'un exemple possible. Par des codages supplémentaires, on peut considérer une machine universelle à un seul ruban et qui a elle-même l'alphabet $\{0, 1\}$.

14.4.3. Problème de l'arrêt. Le problème de l'arrêt est le suivant : étant donnée une machine de Turing M et un mot w sur son alphabet, est-ce que le calcul de M à partir de ce mot w s'arrête ou non. Nous allons voir que ce problème est indécidable mais tout d'abord montrons qu'il est semi-décidable. Pour cela nous devons choisir une formalisation de ce problème : nous n'allons considérer que des machines de Turing d'alphabet $\{0, 1\}$ et considérer leurs codes (codes des ensembles des transitions) vu précédemment. Notons que l'ensemble de ces codes qui sont sur l'alphabet $\Sigma = \{0, 1, G, D, \$, \#\}$ forme un langage rationnel. Pour tout tel code e, l'on note M_e la machine de Turing correspondante. Sous cette formalisation le problème de l'arrêt correspond au langage sur Σ

$$L_{ar} = \{ex : \text{ le calcul de la machine de Turing } M_e \text{ s'arrête en partant de la configuration } \uparrow x\}.$$

Il est assez clair que la machine universelle précédente semi-décide ce langage.

Montrons maintenant que le problème de l'arrêt est indécidable. Nous allons pour cela utiliser un procédé diagonal. Tout d'abord il est facile de se rammener au langage $\{0, 1\}$, par exemple en codant les lettres de $\Sigma = \{0, 1, G, D, \$, \#\}$ de la façon suivante :

$$\begin{array}{rcl} 0 & : & 000 \\ 1 & : & 001 \\ G & : & 010 \\ D & : & 011 \\ \$ & : & 100 \\ \# & : & 101 \end{array}$$

Notons ϕ la fonction de codage. Supposons que L_{ar} soit décidable. Alors nous obtenons une contradiction par un procédé diagonal classique, en construisant une machine de Turing M' sur l'alphabet $\{0, 1\}$ qui à partir de la configuration initiale $\uparrow u$, s'arrête si et seulement si les deux conditions suivantes sont réalisées :

1. le mot u s'écrit sous la forme $\phi(e)u'$ pour un mot e qui est le code d'une machine de Turing ;
2. si la première condition est vérifiée avec le code e alors la machine M_e ne s'arrête pas sur le mot $\phi(e)$ (c'est-à-dire, $e\phi(e)$ n'appartient pas au langage L_{ar}).

La contradiction est alors obtenue en utilisant le code e' de la machine M' et en faisant opérer M' sur la configuration $\uparrow \phi(e')$. En effet :

1. ou bien M' atteint un état d'arrêt. Dans ce cas la seconde condtion n'est pas vérifiée et donc M' n'aurait pas du s'arrêter ;
2. ou bien M' ne s'arrête pas. Mais alors les deux condtions sont vérifiées et cette machine aurait du s'arrêter.

Pour construire M', commençons par l'introduction d'une telle machine à plusieurs rubans. Le premier ruban utilise l'alphabet $\{0, 1\}$ (où 0 correspond aux cases vides). Voici son fonctionnement :

1. On met la donnée u sur le premier ruban ;
2. Cette machine décode par bloc de trois lettres de u, le code e d'une machine Turing qu'elle écrit sur un second ruban. Si elle tombe sur un bloc de trois lettres ne correspondant pas à aucune lettre de Σ, elle déplace alors indéfiniment sa tête de lecture vers la droite.

3. La reconnaissance éventuelle d'un code e se fait simultanément sur le second ruban par la simulation de l'automate correspondant. La machine passe à l'étape suivant si elle a reconnu un code, sinon elle ne s'arrête pas.
4. Si la machine se trouve à cette étape en ayant trouvé le code e d'une machine de Turing, alors sur son second ruban se trouve e et sur le premier ruban un mot $\psi(e)u'$. Elle utilise alors la machine décidant le langage L_{ar} pour déterminer si M_e s'rarrête sur $\psi(e)$ ou non. Dans le cas négatif notre machine s'arrête, dans le cas positif, elle déplace indéfiniment sa tête de lecture à droite.

Remarquons maintenant que cette machine peut être simulée par une machine sur l'alphabet $\{0,1\}$ et avec un seul ruban, cela donne notre machine M'. (Il suffit d'utiliser un procédé analogue à la réduction de deux rubans à un seul. Notons qu'il faut faire un peu attention dans cette simulation qui consiste à dupliquer les cases d'un ruban, pour conserver entrée la donnée sous la même forme.)

14.4.4. Le castor affairé. On considère dans cette partie uniquement des machines de Turing d'alphabet $\Sigma_+ = \{0,1\}$. Nous mettons ces machines en compétition : le but est d'écrire le plus grand nombre de 1 et de s'arrêter ceci en partant de la configuration initiale où le ruban est vide. Pour cette compétition, on range ces machines par catégories selon leurs nombres d'états. Pour tout entier n, il n'y a, à numérotation près des états, qu'un nombre fini de machine de Turing avec n états sur l'alphabet $\{0,1\}$. Il existe donc une (ou plusieurs) machine à n états qui écrit le plus grand nombre de 1 en s'arrêtant : on notera $\mathrm{Cast}(n)$ ce nombre maximal de 1.

Nous allons montrer que l'on obtient ainsi une fonction qui n'est pas calculable.

Tout d'abord vérifions que cette fonction est strictement croissante : soient n et $\mathtt{Castor}_n$ une machine de Turing écrivant $\mathrm{Cast}(n)$ 1 sur ruban puis s'arrêtant. Avec un état supplémentaire, on peut considérer une machine qui simule la machine $\mathtt{Castor}_n$ et quand elle se trouve dans l'état final de celle-ci, reste dans cette état tant qu'elle lit des 1 en se déplaçant à droite. Dès qu'elle lit un 0 eslle écrit un 1 et se place dans son nouvel état qui remplace l'état final précédent. Cette machine à $n+1$ états écrira donc un 1 de plus.

14.4.5 Proposition. *Pour toute fonction calculable* $f : \mathbb{N} \to \mathbb{N}$, $f(n) < \mathrm{Cast}(n)$ *pour* n *assez grand. Donc* Cast *n'est pas calculable.*

Démonstration. Remarquons tout d'abord que la fonction $g(n) = \sum_{i=0}^{n} f(n)$ est également calculable. On peut donc supposer que f est croissante. Comme la fonction produit est calculable, c'est également le cas de la fonction carrée. Il suit que la fonction qui à n associe $f(n^2)$ est calculable. Considérons maintenant une machine de Turing M calculant cette fonction représentée en base 1. Soit k son nombre d'états. Soit n un entier quelconque. Il existe une machine de Turing à $n+1$ qui, à partir de la configuration vide, aboutit à la configuration $\uparrow$ 111...111 avec exactement n 1, c'est-à-dire la représentation base 1 de n sur son ruban. En composant cette machine avec M, on obtient une machine de Turiing à $n+k+1$ états qui, à partir de la configuration vide, écrit $f(n^2)$ 1. Par définition de Cast on a donc

$$f(n^2) \leq \mathrm{Cast}(n+k+1).$$

Maintenant, pour n assez grand on a

$$n+k+1 < (n-1)^2$$

d'où

$$f(n^2) < \mathrm{Cast}((n-1)^2).$$

Par croissance des deux fonctions, il suit que pour tout m assez grand

$$f(m) < \mathrm{Cast}(m).$$

□