# Modéles de Calcul - Machines de Turing

**Michaël Périn**

**Verimag – Polytech'Grenoble – Université Grenoble-Alpes**

**Préambule**

Pour bâtir ce cours je me suis principalement inspiré des ouvrages

- Pierre Wolper. *Introduction à la calculabilité.* Dunod, 2006. 3ème édition
- H. Common-Lundh, P. Schnoebelen, S. Haddad, P-A. Reynier, F-R. Sinot, and C. Sirangelo. *Note de cours de Calculabilité et Logique.* Polycopié en ligne de l'ENS Cachan, 2009

Le premier prend le temps d'expliquer pourquoi en informatique on s'intéresse à la question de décidabilité. Le second démontre les résultats du cours et apporte des compléments.

Je me suis efforcé de réduire au maximum les notations, de détailler les preuves et de les mener de la manière aussi simple que possible en mettant en jeu le minimum de concepts.

Ce cours contient les principaux résultats sur les machines de Turing, les réductions, les problèmes indécidables classiques et le théorême de Rice. Bref, ce qu'il faut de culture pour **transformer un programmeur en informaticien**.

Pour les néophytes, le meilleur ouvrage pour aborder les questions de décidabilité est sans doute la BD

- Apóstolos K. Doxiàdis, Christos Papadimitriou, Alecos Papadatos, and Annie Di Donna. *Logicomix.* Vuibert, BD 2010. 352 pages.

Pour se distraire après avoir relu votre cours, je vous conseille deux films :

- Morten Tyldum. «The Imitation Game», DVD 2014. Adaptation cinématographique de la biographie «Alan Turing ou l'énigme de l'intelligence» d'Andrew Hodges
- Gabe Ibáñez. «Autómata», DVD 2014. Une reflexion sur les capacités des machines

Pour aller plus loin et connaître les questions d'actualité en informatique fondamentale, je vous conseille

- Jean-Paul Delahaye. *Complexités : aux limites des mathématiques et de l'infomatique.* Belin/Pour la science, 2006

# Chapitre 1

# Machine de Turing

## 1.1 Rappels du module INF232 «Langages & Automates»

### 1.1.1 Langage

— Un langage $L$ est un .................... de mots écrits sur un alphabet donné, $\Sigma$.

— $\Sigma^*$ représente l'ensemble de ......... les mots .................... écrits avec des symboles de $\Sigma$.

— Tout langage $L$ sur $\Sigma$ est nécessairement ............. dans $\Sigma^*$, $L \subseteq \Sigma^*$

| Langage = Ensemble de mots écrits sur un alphabet $\Sigma$ |
|---|

### 1.1.2 Ne pas confondre le mot vide $\epsilon$ et le langage vide $\{\,\}$

Le **mot vide**, noté $\epsilon$ correspond au mot de longueur 0, ce n'est rien d'autre que la chaîne de caractères `""` en programmation. Il ne fait pas partie de l'alphabet $\Sigma$ ; on peut d'ailleurs le construire même si l'alphabet ne contient aucun symbole, puisque c'est un mot qui ne contient aucun symbole.

$$abc = a \cdot \epsilon \cdot \epsilon \cdot b \cdot \epsilon \cdot c = \texttt{"a"} + \texttt{""} + \texttt{""} + \texttt{"b"} + \texttt{""} + \texttt{"c"} = \texttt{"abc"}$$

### 1.1.3 Automates à nombre d'états fini et langages réguliers

1. Langage = Ensemble de mots sur un alphabet $\Sigma$
2. Langages .................. = $\mathscr{L}$(AEF déterministe) = $\mathscr{L}$(AEF ........-déterministe)

   clos par union, intersection, complémentaire, concaténation de langages $L_1 \bullet L_2$ et l'opérateur de Kleene ($L^*$)
3. Les langages ne sont pas tous réguliers. Par exemple, $\{a^n b^n \mid n \in \mathbb{N}\}$ n'est pas régulier. Ce résultat se prouve grâce au lemme de l'itération.

## 1.2 Automates à une pile et langages algébriques

Langage algébrique = Langage reconnu par un automate à une pile (AUP).

#### Exercice 1

Donnez un AUP déterministe qui reconnaît $\{a^n b^n \mid n \in \mathbb{N}\}$

#### Exercice 2

Donnez un AUP déterministe qui reconnaît le langage $\{\omega \in \{a,b\}^* \mid |\omega|_a = |\omega|_b\}$.

#### Exercice 3

Donnez un AUP déterministe qui reconnaît les palindromes sur $\Sigma = \{a, b\}$ dont le milieu est marqué par un $\bullet$.

**Exercice 4**

Donnez un AUP non-déterministe qui reconnaît les palindromes sur $\Sigma = \{a, b\}$.

**Remarque** Il n'existe pas d'AUP déterministe qui reconnaît les palindromes. Donc

$$\mathscr{L}(\text{AUP } \textit{déterministe}) \subset \mathscr{L}(\text{AUP } \textit{non-déterministe}).$$

Il existe des langages non-algébriques, *ie.* non reconnaissables par un AUP. Par exemple $\{ a^n b^n c^n \mid n \in \mathbb{N} \}$. On le montre au moyen d'un équivalent du lemme de l'itération pour les AUP.

## 1.3 Machine de Turing

Les machines de Turing sont une extension des automates à une pile. Voici les différences :

1. Alors qu'un AUP a le droit d'écrire et de lire **uniquement** le sommet de la pile, une MT peut .................. **à sa guise sa tête de lecture/écriture** dans la pile – qu'on appelle donc plus une pile mais **un** .............
2. La seconde différence est que **le mot à reconnaître est ............. sur le ruban**.

Les différences entre AUP et MT semblent mineures, pourtant ...

| |
|---|
| Ces extensions suffisent à donner aux MT la puissance du ...................... |

Encore plus surprenant ...

| |
|---|
| L'ajout d'une seconde pile aux AUP suffit à leur donner la .................... du calculable. |

**Preuve** On le montre expliquant comment simuler le fonctionnement d'une MT par un AUP à deux piles, cf. exercice de TD. $\square$

### 1.3.1 Représentation d'une machine de Turing

On considère des machines de Turing opérant sur **l'alphabet** $\Sigma = \{0, 1, \$\}$. Les symboles 0,1 servent à coder les données en binaire et et \$ est un symbole utile dans certains algorithmes pour **marquer une position** sur le ruban. Le symbole $\square$ dénote une **case** ................ du ruban : il **ne fait pas partie de l'alphabet**.

**Exemples :**

- $\epsilon$, 0101, 0\$01\$, \$, 00\$11\$00 sont des mots du langage $\Sigma^*$
- tandis que $\square$, $\square\square$, $\square 0 \square$, $0 \square 0$ et toute suite de symboles contenant un $\square$ ne sont pas des mots de $\Sigma^*$ puisque $\square \notin \Sigma$

Un **ruban vierge** est une ...................... infinie de .......... blanches : $\boxed{{}^\infty\square \mid \square \mid \square \mid \square \mid \square^\infty}$

**Exemples :**

- $\boxed{{}^\infty\square \mid 0 \mid 1 \mid 0 \mid \square^\infty}$ représente le mot 010 inscrit sur le ruban.
- $\boxed{{}^\infty\square \mid \square^\infty}$ représente le mot $\epsilon$ de longueur 0 inscrit sur le ruban.

Les **transitions des** MT sont de la forme

— $\mathbf{q} \xrightarrow{\ell/e} \mathbf{q'}$ : dans l'état $\mathbf{q}$, si on lit $\ell$, on écrit $e$ sur le ruban et on ............ dans l'état $\mathbf{q'}$

— $\mathbf{q} \xrightarrow{\ell:d} \mathbf{q'}$ : dans l'état $\mathbf{q}$, si on lit $\ell$, on effectue le ............................ $d$ de la tête de lecture/écriture et on passe dans ............ $\mathbf{q'}$. Les déplacements possibles sont $\{L, H, R\}$ pour *Left, Here, Right*. Un $L$ (resp. $R$) déplace la tête d'une case vers la gauche (resp. vers la droite).

— $\mathbf{q} \xrightarrow{\ell/e:d} \mathbf{q'}$ : dans l'état $\mathbf{q}$, si on lit $\ell$, on écrit $e$ sur le ruban *avant* d'effectuer le déplacement $d$ de la tête, puis on passe dans l'état ............ $\mathbf{q'}$

Parmi les **états d'une** MT on distingue un état ............. et potentiellement un état accepteur, un état erreur.

— $\searrow\!\bigcirc$ est l'**état initial**

— $\circledcirc$ est **l'état accepteur**

— les états non-accepteurs sont notés $\bigcirc$

— $\otimes$ est un état non-accepteur particulier, appelé **l'état erreur**

— un état est **terminal** $\iff$ il n'a **pas de transition sortante**, ce que l'on note $\bigcirc\!\not\to$

### 1.3.2 Notations mathématiques d'une MT

Pour définir précisement une MT $M$ il faut indiquer

— $\Sigma$, l'alphabet sur lequel elle opère

— $\mathscr{Q}$, l'ensemble des états utilisés dans les transitions. Par exemple, $\mathscr{Q} = \{\mathbf{q}_0, \mathbf{q}_1, \ldots\}$

— $\searrow\!\bigcirc$, son état initial

— $\delta : \mathscr{Q} \times \Sigma \cup \{\square\} \to \Sigma \cup \{\square\} \times \{L, H, R\} \times \mathscr{Q}$, sa fonction de transition.
$\delta$ associe à un état $\mathbf{q}$ et un symbole lu $\ell$, un triplet $(e, d, \mathbf{q'})$ qui représente le symbole à écrire, le déplacement de la tête et l'état d'arrivée.

— $\mathscr{A}cc \subseteq \mathscr{Q}$ est un ensemble d'état qui est soit vide, soit réduit à l'état accepteur, $\circledcirc$

— $\mathscr{E}rr \subseteq \mathscr{Q}$ est un ensemble d'état qui est soit vide, soit réduit à l'état erreur, $\otimes$

On définit une MT en donnant $M = (\Sigma, \mathscr{Q}, \searrow\!\bigcirc, \delta, \mathscr{A}cc, \mathscr{E}rr)$

**Définition 1 (Transitions)** *Il existe plusieurs notations équivalentes pour les transitions d'une* MT

$\delta(\mathbf{q}, \ell) = (e, d, \mathbf{q'})$ ou bien $\mathbf{q} \xrightarrow{\ell/e:d} \mathbf{q'}$ ou bien $(\mathbf{q}, \ell/e : d, \mathbf{q'})$

### 1.3.3 Conventions adoptées en MCAL

Il existe de nombreuses variantes – toutes équivalentes – des machines de Turing. Dans ce cours, on adopte les conventions suivantes :

(C1) On considére uniquement des MT **déterministes**, c'est-à-dire qu'aucun état $\mathbf{q}$ ne peut avoir deux ........................ différentes sur le même symbole lu.

**Exemples :**

- $(\mathbf{q}, \ell : H, \mathbf{q'})$ et $(\mathbf{q}, \ell : R, \mathbf{q'})$ n'est pas autorisé : deux actions ..................... pour $\delta(\mathbf{q}, \ell)$.
- $(\mathbf{q}, \ell/e, \mathbf{q'})$ et $(\mathbf{q}, \ell/e, \mathbf{q''})$ n'est pas autorisé : deux états ................. différents pour $\delta(\mathbf{q}, \ell)$.

(C2) Au delà d'un blanc vers la gauche et au delà d'un blanc vers la droite, il n'y que des ...............

Autrement dit **le ruban ne doit jamais avoir la forme suivante**

| | | | | |
|---|---|---|---|---|
| $^\infty\square$ | $\omega_1$ | $\square$ | $\omega_2$ | $\square^\infty$ |

(C3) Les machines de Turing ont un unique état initial ; au plus un état accepteur ; au plus un état erreur.

(C4) L'état accepteur $◎$ et l'état erreur $\otimes$ sont des états terminaux : $◎\not\to$, $\otimes\not\to$

### 1.3.4 Exemples de machines de Turing

**Exercice 5**

Donnez une machine de Turing $M_{\overleftarrow{\$}}$ qui recherche le marqueur \$ vers la gauche et termine dans un état $◎$ si elle a réussi et dans un état $\otimes$ sinon. On suppose que la MT démarre sur un symbole $\neq \square$. Que se passerait'il si la MT démarrait sur un $\square$ ?

**Exercice 6**

Donnez une machine de Turing $M_{\mathit{eff}}$ qui efface le ruban. On suppose que $M_{\mathit{eff}}$ commence sur un symbole $\neq \square$ du ruban.

## 1.4 Exécution d'une machine de Turing

Considérons une MT $M = (\Sigma, \mathscr{Q}, \searrow\!\bigcirc, \delta, \mathscr{A}cc, \mathscr{E}rr)$

**Définition 2** *Une **configuration** $(\omega_1, \mathbf{q}, \ell.\omega_2)$ contient les informations nécessaires pour exécuter la prochaine ..................... de la machine de Turing $M$ :*

- *son état courant :* $\mathbf{q}$
- *le contenu du ruban :* $\omega_1.\ell.\omega_2$ *désigne la partie intéressante du ruban*

| | | | | |
|---|---|---|---|---|
| $^\infty\square$ | $\omega_1$ | $\ell$ | $\omega_2$ | $\square^\infty$ |

- $\omega_1$ *désigne la partie du ruban situé à .............. de la tête,* $\omega_1 \in \Sigma^*$
- $\ell$ *est le symbole courant que lit la tête de lecture/écriture,* $\ell \in \Sigma$
- $\omega_2$ *désigne la partie du ruban situé à ............ de la tête,* $\omega_2 \in \Sigma^*$

*Une configuration* $(\omega_1, \mathbf{q}, \ell.\omega_2)$ *est un élément de* $\Sigma^* \times \mathscr{Q} \times \Sigma^+$.

**Définition 3 (Effet des différentes transitions sur une configuration)** *Le symbole* $\square$ *indique l'emplacement de la tête de lecture/écriture sur le ruban.*

1. *config :* $(\omega_1, \mathbf{q}, \ell.\omega_2)$ $\quad \mathbf{q} \xrightarrow{\ell/e} \mathbf{q'}$ $\quad (\omega_1, \mathbf{q'}, \ldots\ldots\ldots)$

   *ruban :* $\omega_1.\boxed{\ell}.\omega_2$ $\qquad \omega_1.\boxed{e}.\omega_2$

2. *config :* $(\omega_1, \mathbf{q}, \ell.\omega_2)$ $\quad \mathbf{q} \xrightarrow{\ell:H} \mathbf{q'}$ $\quad (\omega_1, \mathbf{q'}, \ldots\ldots\ldots)$

   *ruban :* $\omega_1.\boxed{\ell}.\omega_2$ $\qquad \omega_1.\boxed{\ldots}.\omega_2$

3. *config :* $(\omega_1, \mathbf{q}, \ell_1.\ell_2.\omega_2)$ $\quad \mathbf{q} \xrightarrow{\ell_1:R} \mathbf{q'}$ $\quad (\ldots\ldots\ldots\ldots, \mathbf{q'}, \ell_2.\omega_2)$

   *ruban :* $\omega_1.\boxed{\ell_1}.\ell_2.\omega_2$ $\qquad \omega_1.\ell_1.\boxed{\ldots\ldots}.\omega_2$

4. *config :* $(\omega_1.\ell_1, \mathbf{q}, \ell_2.\omega_2)$ $\quad \mathbf{q} \xrightarrow{\ell_2:L} \mathbf{q'}$ $\quad (\omega_1, \mathbf{q'}, \ldots\ldots\ldots\ldots\ldots\ldots)$

   *ruban :* $\omega_1.\ell_1.\boxed{\ell_2}.\omega_2$ $\qquad \omega_1.\boxed{\ldots\ldots}.\ell_2.\omega_2$

**Définition 4 (Configuration initiale)** *Pour exécuter une machine de Turing $M$ sur une **donnée d'entrée** $\omega$*

***CAS 1 : à partir d'un ruban vierge*** *ie. qui ne contient que des blancs* $\Box$.

1. *l'utilisateur inscrit* $\omega$ *sur le ruban*
2. *Il place la ........ de lecture/écriture sur le premier symbole du mot* $\omega$
3. *La* MT $M$ *démarre dans son état ............*

*Le ruban a l'aspect* $\begin{array}{c|c|c} \hline {}^{\infty}\Box & \omega & \Box^{\infty} \\ \hline & \uparrow & \end{array}$

***CAS 2 : la donnée*** $\omega$ ***a été inscrite sur le ruban par une*** MT ***précédente.***

1. *La* MT *précédente doit avoir positionné la tête de lecture/écriture sur le premier symbole du mot* $\omega$. *Il est parfois utile de faire précéder la donnée* $\omega$ *d'un marqueur* $ *pour indiquer de ne pas aller au delà de cette position afin de protéger des données écrites à gauche de* $\omega$.
2. *La* MT $M$ *démarre dans son état ............*

*Le ruban a l'aspect* $\begin{array}{c|c|c|c|c|c} \hline {}^{\infty}\Box & \dots & \$ & \omega & \Box & \Box^{\infty} \\ \hline & & & \uparrow & & \end{array}$

**Exercice 7** *(**)*

Donnez une machine de Turing $M_\$$ qui insère un $ avant la donnée $\omega$ sans détruire les éventuelles données situées à gauche de $\omega$.

**Exercice 8** *(**)*

Donnez une machine de Turing $M_{find}$ qui explore le ruban (dans les deux directions) à la recherche d'un mot écrit quelque part sur le ruban. Si le ruban n'est pas vierge, la MT doit s'arrêter sur le premier symbole du mot. Que se passe t'il si le ruban est vierge ?

**Exercice 9** *(***)*

Donnez une MT qui répare un ruban en concaténant les mots séparés par des $\Box$. Cette MT ne termine pas, sauf si on fixe une limite (disons $B$) sur le nombre maximal de $\Box$ entre deux mots.

**Définition 5 (Configuration terminale)** *Une **configuration*** $(\omega, \mathbf{q}, \omega')$ *est **terminale** si l'état* $\mathbf{q}$ *est **terminal**, ie.* $\mathbf{q}$ *n'a pas de transition sortante, ce qu'on indique par* $\textcircled{q}\not\to$. *L'exécution de* $M$ *est alors nécessairement .................. puisque* $M$ *.......................... effectuer de transition.*

**Définition 6 (Exécution)** *L'exécution d'une machine de Turing* $M$ *sur une donnée* $\omega$ *est une séquence de configurations* $(\omega_0, \mathbf{q}_0, \omega'_0) \xrightarrow[M]{} (\omega_1, \mathbf{q}_1, \omega'_1) \xrightarrow[M]{} \dots$, *commençant dans la configuration ..............* $(\epsilon, {\searrow}\bigcirc, \omega)$ *et passant d'une configuration à la suivante par une .................... de* $M$. *L'exécution peut s'arrêter sur une configuration .................... ou bien être infinie.*

**Définition 7 (Résultat d'une machine de Turing)** *Pour qu'une machine de Turing rende un ............... il faut que l'exécution **s'arrête** c'est-à-dire qu'elle atteigne une **configuration** ...................... .*
*On distingue 3 cas de terminaison et le cas des exécutions infinies :*

1. $M(\omega) = \mathbb{V}(\omega'_2)$ *si* $(\epsilon, {\searrow}\bigcirc, \omega) \xrightarrow[M]{}^{*} (\omega'_1, \circledcirc, \omega'_2)$ *et* $\circledcirc\not\to$

   **Interprétation** Le constructeur $\mathbb{V}$ indique que l'exécution s'est bien déroulée et que la MT $M$ s'est arrêtée dans un *état accepteur*. Le résultat du calcul est le mot $\omega'_2$ situé à droite de la tête de lecture.

2. $M(\omega) = \mathbb{F}(\omega'_2)$ *si* $(\epsilon, {\searrow}\bigcirc, \omega) \xrightarrow[M]{}^{*} (\omega'_1, \bigcirc, \omega'_2)$ *et* $\bigcirc\not\to$

   **Interprétation** Le constructeur $\mathbb{F}$ indique que l'exécution s'est bien déroulée et que la MT $M$ s'est arrêtée dans un *état non-accepteur*. Le résultat du calcul est le mot $\omega'_2$ situé à droite de la tête de lecture.

3. $M(\omega) = \mathbb{E}\mathtt{rr}$ *si* $(\epsilon, {\searrow}\bigcirc, \omega) \xrightarrow[M]{}^{*} (\omega'_1, \otimes, \omega'_2)$ *et* $\otimes\not\to$

**Interprétation** Le constructeur $\mathbb{E}\mathrm{rr}$ signale une erreur/exception. La MT $M$ s'est arrêtée dans un état erreur et il ne faut pas considérer le mot $\omega_2'$ situé à droite de la tête de lecture comme un résultat valide.

*4. $M(\omega) = ?$ si et seulement si $(\epsilon, \searrow\!\bigcirc, \omega) \xrightarrow[M]{}{}^{\infty}$ ie. que l'exécution ........................ indéfiniment.*

**Notation** On notera simplement $M(\omega) = \mathbb{V}$, $M(\omega) = \mathbb{F}$, $M(\omega) = \mathbb{E}\mathrm{rr}$ lorsque le résultat du calcul n'a pas d'importance et qu'on s'intéresse uniquement à l'état terminal ($\circledcirc$, $\bigcirc$, $\otimes$) de l'exécution.

## 1.5 Langage associé à une machine de Turing

Les AEF et AUP s'arrêtent forcément lorsqu'ils ont consommés toutes les lettres du ......... à reconnaître. Contrairement aux AEF et AUP, l'exécution d'un MT peut ne pas ................... Le ................ reconnu par une MT ne concerne que les exécutions qui *terminent* dans l'état *accepteur.*

**Définition 8** *On note $\mathscr{L}(M)$ le* ***langage r*** *...............* ***par/ a*** *.............* ***à une machine de Turing*** *$M$*

$$\mathscr{L}(M) = \{\omega \in \Sigma^* \mid M(\ldots) = \mathbb{V}\}$$

*C'est l'ensemble des mots $\omega$ pour lesquels l'exécution de $M$* ***s'arrête*** *dans un ........ ......................*
$\circledcirc$.

**Exercice 10** ***Non-terminaison***

Donnez une machine qui ne termine pas pour $\epsilon$ $(= {}^{\infty}\square \mid \square \mid \square^{\infty})$ et qui termine pour tous les autres mots. Donnez $\mathscr{L}(M)$, le langage accepté par votre machine $M$.

**Exercice 11** ***Non-terminaison***

Donnez une machine $M$ qui ne termine jamais. Donnez $\mathscr{L}(M)$, le langage accepté par $M$.

**Exercice 12** ***Reconnaissance de langages classiques***

Soit l'alphabet $\Sigma = \{a, b\}$. Pour chacun des langages suivants, donnez une MT qui le reconnaît $L_1 = \Sigma^*$, $L_2 = \emptyset$, $L_3 = \{\epsilon\}$, $L_4 = \{a^n b^n \mid n \in \mathbb{N}\}$, $L_5 = \Sigma \cup \{\omega.R(\omega) \mid \omega \in \Sigma^*\}$ où $R$ est l'opération qui renverse un mot et donc $L_5$ est l'ensemble des palindromes sur $\Sigma$.

## 1.6 Programmer avec des machines de Turing

L'objectif de cette section est de montrer qu'on peut simuler les langages de programmation usuels (tel que C) à l'aide des machines de Turing opérant sur une représentation binaire des données. Ainsi nous aurons montré que les machines de Turing peuvent simuler tous les comportements d'un ordinateur. La conséquence (au chapitre suivant) sera que ce qui est impossible pour une MT le sera pour un ordinateur.
Pour programmer nous avons besoin de représenter des fonctions, des prédicats de bases, des données (les entrées et les résultats), ainsi que les structures de contrôle des langages de programmation.

### 1.6.1 Représentation binaire des données

**Définition 9** *On définit $\mathbb{N}^* = \bigcup_{i \in \mathbb{N}} \mathbb{N}^i$ l'ensemble des listes d'entiers comme l'ensemble des ........................ de taille 0, 1, 2, etc.*

**Exemples :**

- $\mathbb{N}^2 = \mathbb{N} \times \mathbb{N}$ est l'ensemble des couples d'entiers $(n_1, n_2)$

- $\mathbb{N}^p$ est l'ensemble des vecteurs d'entiers à $p$ composantes $(n_1, \ldots, n_p)$
- $\mathbb{N}^1 = \{(0), (1), (2), \ldots\}$ contient les vecteurs à une seule .......................
- $\mathbb{N}^0$ contient un ............ vecteur : ()

Pour représenter les vecteurs d'entiers sur le ruban on a besoin de la représentation binaire des entiers et des symboles $(\,,\,)$ : soit 5 symboles en plus du $\square$. Pour représenter 6 symboles, il faut au minimum 3 bits puisque $2^2 < 6 < 2^3 = 8$. On utilise 3 bits pour coder nos 6 symboles. Notre alphabet de triplets de bits a donc 8 symboles. On a alors le droit à 2 symboles supplémentaires ; on choisit \$ et § qui serviront de marqueurs de position dans certains algorithmes. On considère finalement l'alphabet $\Sigma_8 = \{0, 1, (\,,\,,\,), \$, \S\} \cup \{\square\}$. On choisit la notation *little-endian* qui consiste à mettre les unités à gauche. Ainsi, la représentation binaire de 6, notée $[6]_2 = 011 = 0 \times 2^0 + 1 \times 2^1 + 1 \times 2^2$.

**Exemple :** le vecteur $(1, 2, 6)$ se notera

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $^{\infty}\square$ | ( | 1 | , | 0 | 1 | , | 0 | 1 | 1 | ) | $\square^{\infty}$ |

En TD vous verrez comment se ramener à un alphabet binaire $\Sigma_2 = \{\boxed{0}, \boxed{1}\}$.

**Exercice 13** ***Codage d'un vecteur d'entiers en binaire***

1. Donnez un codage dans l'alphabet $\Sigma_8 = \{0, 1, (\,,\,,\,), \$, \S\} \cup \{\square\}$ du vecteur $(0, 1, 2)$ en transformant les entiers en représentation binaire *little-endian* : ....................
2. Donnez un codage des symboles de l'alphabet $\Sigma_8 = \{0, 1, (\,,\,,\,), \$, \S\} \cup \{\square\}$ sous forme de triplet de symboles $\Sigma_2 = \{\boxed{0}, \boxed{1}\}$

| $\Sigma_8$ | $\to$ | $\Sigma_2 \times \Sigma_2 \times \Sigma_2$ |
|---|---|---|
| 0 | $\mapsto$ | $\boxed{0}\boxed{0}\boxed{0}$ |
| 1 | $\mapsto$ | $\boxed{1}\boxed{1}\boxed{1}$ |
| ( | $\mapsto$ | .............. |
| ) | $\mapsto$ | .............. |
| , | $\mapsto$ | .............. |
| \$ | $\mapsto$ | .............. |
| § | $\mapsto$ | .............. |
| $\square$ | $\mapsto$ | .............. |

3. Donnez le codage dans l'alphabet $\Sigma_2 \times \Sigma_2 \times \Sigma_2$ du vecteur $(0, 1, 2)$ :
   ..........................................................................................................
4. Donnez le mot $\omega \in \{\boxed{0}, \boxed{1}\}^*$ correspondant au codage de l'entrée $\$(0, 1, 2)\square$ :
   ..........................................................................................................
5. Décodez le résultat $\boxed{0}\boxed{0}\boxed{1}.\boxed{1}\boxed{1}\boxed{0}.\boxed{1}\boxed{1}\boxed{1}.\boxed{0}\boxed{1}\boxed{0}.\boxed{0}\boxed{0}\boxed{0}.\boxed{1}\boxed{1}\boxed{1}.\boxed{0}\boxed{1}\boxed{0}.\boxed{1}\boxed{1}\boxed{1}.\boxed{0}\boxed{0}\boxed{0}.\boxed{1}\boxed{1}\boxed{1}.\boxed{0}\boxed{1}\boxed{1}$ trouvé sur le ruban après l'exécution d'une machine de Turing sur l'entrée précédente. Donnez le résultat sous la forme d'un vecteur d'entiers naturels : ............

### 1.6.2 Les fonctions arithmétiques

**On s'intéresse aux fonctions** $f : \mathbb{N}^p \to \mathbb{N}^*$ qui opèrent sur *(la représentation binaire d')* un vecteur d'entiers $\vec{d} = (d_1, \ldots, d_p)$ représentant les $p$ données d'entrée et qui retourne *(la représentation binaire d')* un vecteur d'entiers $\vec{r} = (r_1, \ldots, r_q)$ de taille $q$ variable, représentants $q$ résultats entiers.

**Exemples :**

- *euclide* : $\mathbb{N}^2 \to \mathbb{N}^2 : (a, b) \mapsto (q, r)$ tels que $a = b \times q + r$ et $r < b$
- *inc* : $\mathbb{N}^1 \to \mathbb{N}^1 : (n) \mapsto (n + 1)$ où $(n)$ est un vecteur à une seule composante
- *sub* : $\mathbb{N}^2 \to \mathbb{N}^* : (n, m) \mapsto (n - m) \in \mathbb{N}^1$ si $n \geq m$ *et* $() \in \mathbb{N}^0$ sinon pour indiquer que le résultat n'est pas défini.
- *dfp* : $\mathbb{N}^1 \to \mathbb{N}^* : (n) \mapsto (p_1, \ldots, p_k)$ tels que $n = p_1 \times \ldots \times p_k$ et $p_i$ premiers
- *pi* : $\mathbb{N}^1 \to \mathbb{N}^1 : (n) \mapsto n^e$ décimale de $\pi$

**Exercice 14**

Donnez les MT $M_{inc}$ et $M_{dec}$ qui incrémente (resp. décrémente) un entier binaire *little-endian* inscrit sur le ruban.

Certaines de ces fonctions se définissent simplement en donnant directement la MT qui la réalise ; les autres sont définies à l'aide des MT de bases en utilisant les constructions du langage de programmation du §1.6.4.

**Notation** On n'écrira pas explicitement le contenu des vecteurs mais seulement $f(\vec{d}) = \vec{r}$ et on désignera par $\omega_d = [\vec{d}]_2$ et $\omega_r = [\vec{r}]_2$ les .................................. binaires des vecteurs $\vec{d}$ et $\vec{r}$.

### 1.6.3 Les prédicats

**Un prédicat** $P : \mathbb{N}^p \to Bool$ est un cas particulier de .................. $\mathbb{N}^p \to \mathbb{N}^*$ puisque

$$Bool = \{\mathbb{F}, \mathbb{V}\} \simeq \{(0), (1)\} \subset \mathbb{N}^*$$

**Exemples :**

- *nul ?* : $\mathbb{N}^1 \to Bool : (n) \mapsto \mathbb{V}$ si $n = 0$ *et* $\mathbb{F}$ sinon
- *inf ?* : $\mathbb{N}^2 \to Bool : (a, b) \mapsto \mathbb{V}$ si $a < b$ *et* $\mathbb{F}$ sinon

**Exercice 15**

Donnez une MT $M_{nul?}$ qui réalise le prédicat *nul ?* Attention, $\epsilon$ – représenté par le ruban $\boxed{{}^{\infty}\square \mid \square^{\infty}}$ – n'est pas un nombre. Il serait erronné de l'assimiler à 0.

### 1.6.4 Un langage impératif à base de machines de Turing (Projet 2025)

Pour montrer que les machines de Turing ont l'expressivité d'un langage de programmation impératif, on donne pour chaque construction du langage un moyen de construire une MT équivalente.

- **Les fonctions et instructions** de base sont des MT écrites à la main : $M_{\vec{\square}}$, $M_{inc}$, $M_{nul?}$, *etc.*
- **Les prédicats/tests** : une MT $M$ réalise/implémente un prédicat si elle s'arrête pour toute donnée binaire $\omega \in \{\boxed{0}, \boxed{1}\}^*$ et termine
  - soit sur ◎ on dit alors que $M(\omega)$ vaut $\mathbb{V}$
  - soit sur ○ on dit alors que $M(\omega)$ vaut $\mathbb{F}$.
  - soit sur ⊗ on dit alors que $M(\omega)$ vaut $\mathbb{E}\mathbb{r}\mathbb{r}$.
- **La conditionnelle** [if ($M_{cond}$) then $M_1$ else $M_2$] est une MT qui exécute $M_1$ si $M_{cond}$ termine dans l'état ◎ et qui exécute $M_2$ si $M_{cond}$ termine dans un état ○
- **La séquence** $[M_1 ; M_2]$ de deux MT est une MT qui exécute $M_1$ puis exécute $M_2$, cf. TD.
- **L'** .................. [while ($M_{cond}$) do $M_{body}$] est une MT qui exécute $M_{body}$ tant que $M_{cond}$ termine dans l'état ◎, cf. examen 2015.
- **Le calcul** $f(i_1, \ldots, i_d)$ consiste à appliquer la MT $M_f$ qui réalise la fonction $f$ sur la donnée $[(i_1, \ldots, i_d)]_2$ inscrite sur le ruban.
- **Les** .................. $i, j, \ldots$ sont représentées par des rubans séparés. Le ruban $B_i$ contient la valeur de la variable $i$.

| | | | |
|---|---|---|---|
| $B_i =$ | ${}^{\infty}\square$ | *valeur de i* | $\square^{\infty}$ |
| $B_j =$ | ${}^{\infty}\square$ | *valeur de j* | $\square^{\infty}$ |

$\vdots$

On verra en TD qu'une machine $M$ à plusieurs rubans peut être simulée par une MT $M'$ à un seul ruban, *ie.* que la MT $M'$ construit sur son ruban un résultat correspondant à celui que calcule $M$ sur ses multiples rubans. Les machines de Turing à un ruban ont donc le même pouvoir de calcul que les machines à plusieurs rubans, par contre elles sont plus lentes.

— **L'** ........................ $i := \omega$ consiste à recopier $\omega$ (soit une constante, soit le résultat d'un calcul, soit le contenu d'une variable) sur la bande $B_i$

— **L'** ...................... $\mathbb{E}\mathtt{rr}$ **:** Une MT $M_1$ qui s'arrête dans l'état $\otimes$ correspond à un signal d'erreur ou une exception. Seule instruction [`try` $M_1$ `catch` $M_2$] est capable de reprendre l'exécution.

On peut enrichir ce langage en ajoutant de nouvelles constructions, à condition d'expliquer comment réaliser la MT correspondant à cette construction.

**Exercice 16**

Définissez la construction [`try` $M_1$ `catch` $M_2$] qui exécute la MT $M_1$ et exécute la MT $M_2$ uniquement si l'exécution de $M_1$ atteint un état erreur $\otimes$ représentant une exception. Si l'exécution de $M_1$ se déroule normalement alors la machine $M_2$ n'est pas déclenchée.

**Exercice 17** ***

À l'aide de MT à deux rubans, définissez la construction [`transaction` $M$] qui exécute la MT $M$ et annule son effet (c'est à dire revient au ruban précédent) si cette exécution atteint l'état erreur $\otimes$.

### 1.6.5 Application

**Exercice 18** ***Énumération des mots de*** $\{0,1\}^*$

Donnez une MT $M_{next}$ qui à partir d'un mot $\omega \in \{0,1\}^*$ écrit sur le ruban un mot binaire différent, de sorte qu'en partant du ruban vide et en itérant les appels à $M_{next}$ on énumérere tous les mots de $\{0,1\}^*$. Notez que $M_{next} \stackrel{\text{def}}{=} M_{inc}$ qui incrémente un entier binaire ne convient pas car elle oublierait des mots binaires tels que $00, 010, 0100, 0010, \ldots$ qui contiennent des 0 non significatifs.

$$M_{next}(\epsilon) = 0,\ M_{next}(0) = 1,\ M_{next}(1) = 00,\ M_{next}(00) = 10,\ M_{next}(10) = 01,\ M_{next}(01) = 11,\ \ldots$$

**Indication :** On s'inspire de la MT $M_{inc}$ mais au lieu de transformer $11\ldots11$ en $00\ldots001$ comme le ferait $M_{inc}$ on transforme $11\ldots11$ en $00\ldots000$. Autrement dit $M_{next}$ se comporte comme $M_{inc}$ mais elle oublie la retenue lorsqu'on arrive sur le $\square$ qui suit le nombre binaire.

## Résumé

Nous avons vu que les machines de Turing permettent de définir un langage de programmation impératif comportant des expressions arithmétiques et booléennes, des séquences d'instructions : affectation, branchements conditionnels, itérations (while et for). Ce langage « while » est suffisamment évolué pour permettre de coder tout algorithme qu'on pourrait écrire dans des langages modernes.

Nous avons également vu qu'on pouvait coder dans l'alphabet binaire $\Sigma = \{\boxed{0}, \boxed{1}\}$ tous types de données : entiers, symboles, vecteurs, images, .... C'est le codage utilisé dans les ordinateurs pour représenter en mémoire les données.

Les MT sont capables de prouesses impresionnantes. Elles peuvent calculer les décimales de $\pi$ aussi loin que l'on veut, elles peuvent énumérer tous les mots binaires, ... Cependant, nous démontrerons dans le chapitre suivant – en raisonnant par l'absurde – que les fonctions $\mathbb{N} \rightarrow Bool$, en apparence simples, ne sont pas toutes réalisables par les machines de Turing.

# Chapitre 2

# Machine Chimique

À la fin des années 80, Le Métayer & Banâtre avec $\Gamma$ [BLM93] puis Berry & Boudol avec la *CHemical Abstract Machine* [BB92] ont proposé un modèle de calcul très simple qui modélise le calcul massivement parallèle (et même le parallèlisme maximal). Depuis, ce modèle a fait du chemin et a eu des utilisations très diverses [BFLM00]. Nous allons brièvement présenter leur proposition qui considère les réactions chimiques comme moteur du calcul. Nous utiliserons ce modèle au chapitre suivant pour décrire les bijections de Cantor.

## 2.1 Le modèle chimique

**Les algorithmes = des réactions chimiques** Les algorithmes sont définis sous forme de règles comme les réactions chimiques

| | | | | |
|---|---|---|---|---|
| forme générale | : | *réactifs* | $\xrightarrow{condition}$ | *produits* |
| chimie | : | $H,\ O,\ H$ | $\longrightarrow$ | $H_2O$ |
| algorithmique | : | $x, y$ | $\longrightarrow$ | $x + y$ |

**Les données = un amas de molécules** non ordonné où l'on trouve plusieurs occurences de la même molécule. Inspiré de la métaphore chimique, la mémoire sera un **multi-ensemble** de données, c'est-à-dire un « *ensemble* » au sens où les éléments ne sont pas ordonnés, mais « *multi* » puiqu'on autorise la présence de plusieurs occurences du même élément.

**Exemple :** $\{\!|1, 1, 0, 2, 0, 1|\!\}$ est un multi-ensemble à 6 éléments contenant 3 occurences de l'élément 1.

— **Les atomes** sont les éléments des type de base : $\mathbb{N}, Bool, \ldots$

— **Les molécules** sont des termes, c'est-à-dire des données complexes forgées à l'aide de constructeurs (notés en majuscule $\mathrm{A}, \mathrm{B}, \mathrm{C}, \ldots$) et d'atomes ou de plus petites molécules déjà formées.

**Exemple :** Avec un seul constructeur $\mathrm{C}(\_,\_)$ à deux arguments on peut manipuler

— **des couples d'entiers :**
Appliquée au multi-ensemble $\{\!|\mathrm{C}(0,0), \mathrm{C}(7,3)|\!\}$, la règle $\mathrm{C}(x, y) \xrightarrow{y<x} \mathrm{C}(y, x)$ ordonne les composantes des couples et donne $\{\!|\mathrm{C}(0,0), \mathrm{C}(3,7)|\!\}$

— **des couples de couples d'entiers :**
Appliquée au multi-ensemble à un seul élément $\{\!|\ \mathrm{C}(\mathrm{C}(0,0), \mathrm{C}(3,7))\ |\!\}$, la règle $\mathrm{C}(x, y) \longrightarrow x, y$ scindent les couples et donne $\{\!|\mathrm{C}(0,0),\ \mathrm{C}(3,7)|\!\}$ après une étape d'exécution et $\{\!|0, 0, 3, 7|\!\}$ au final.

— **Attention** à ne pas confondre les constructeurs $\mathrm{C}, \mathrm{S}, \mathrm{Z}, \ldots$ avec des fonctions $c : (x, y) \mapsto x^2 + y^2$, $\ s : x \mapsto x + 1$ ou des variables $z = 0$ :

**Exemples :**

- $\mathrm{C}(1, 2)$ n'est pas un calcul, c'est un terme *ie.* une structure constituée d'atomes, c'est une donnée inerte à ne pas confondre avec un appel de fonction $c(1, 2)$ qui retourne $1^2 + 2^2 = 5$
- $\mathrm{S}(\mathrm{S}(\mathrm{Z}))$ est un terme qui ne vaut rien d'autre que lui-même $\mathrm{S}(\mathrm{S}(\mathrm{Z}))$ à ne pas confondre avec le calcul $s(s(z))$ qui vaut $s(s(0)) = (0 + 1) + 1 = 2$

**Les conditions de la réaction** Les conditions des réactions s'écrivent à l'aide des prédicats usuels (par exemple, $x < y \ \wedge \ x \neq z$) auxquels on ajoute la reconnaissance de motif (*pattern matching*) puisque les molécules sont des termes.

**Exemple :**

« si le terme $t$ n'est pas un couple, l'éliminer » s'écrit $t \xrightarrow{t \neq \mathrm{C}(\_,\_)}$ .

« si le terme $t$ est un couple, le scinder en composantes mais sans répétition » s'écrit $\begin{cases} \mathrm{C}(x,y) & \xrightarrow{x \neq y} & x,\ y \\ \mathrm{C}(x,x) & \longrightarrow & x \end{cases}$

**Le principe de calcul**

1. Les réactions s'effectue en parallèle, simultanément, sur tous les réactifs qui satisfont leurs conditions et ce jusqu'à ce qu'aucune réaction ne puisse plus s'appliquer
2. Un élément réactif ne peut pas participer à plusieurs réactions simultanément
3. les réactifs sont consommés *ie.* retirés du multi-ensemble
4. les produits sont crées *ie.* ajoutés au multi-ensemble

### 2.1.1 Les limitations voulues de Gamma et l'application aux protocoles réseaux

La métaphore chimique s'applique encore pour définir ce qui est impossible en Gamma.

**On ne peut pas**

— **mettre des priorités entre les règles.** En chimie, les réactions s'effectuent sans demander la priorité ; si une réaction peut avoir lieu, elle a lieu.

— **demander si une règle a fini de s'appliquer.** En chimie, on ne sait jamais quand une solution a stabilisé ; le seul moyen serait d'attendre un temps très très très long. En Gamma c'est pareil, il se peut que la donnée $D$ qui réagira avec la donnée $D'$ ne l'ait pas encore croisée. Il faudrait donc attendre et attendre encore pour être sûr, mais alors on perdrait tout l'intérêt de la programmation parallèle.

— **définir l'opérateur de séquence «;».** La séquence de deux réactions « $r_1$ ; $r_2$ » signifie « lancer la réaction $r_1$ » puis « lancer la réaction $r_2$ » **quand $r_1$ a fini d'agir** : c'est impossible du fait des limitations précédentes.

— **tester l'absence d'une donnée**. En chimie on peut prouver la présence d'une molécule $m$ en introduisant un produit qui réagit avec la molécule $m$ pour créer quelque chose d'observable (un changement de couleur par exemple). En revanche on ne peut pas tester l'absence d'une molécule. Gamma reprend la même idée, on ne peut pas écrire de règle « $\neg m \rightarrow \ldots$ » car cela obligerait l'exécution à stopper toutes réactions pour examiner une par une les données du multi-ensemble afin de tester l'absence de $m$ ; on perdrait alors tout le parallèlisme.

La programmation chimique est donc un art subtil qui consiste à contourner ces limitations. Elle est utilisée pour définir des protocoles réseaux car c'est typiquement un cas dans lequel on n'a pas de garantie *de séquence, de terminaison, de priorité, de test d'absence (de réponse ou d'erreur)* et dans lequel on souhaite un parallèlisme maximal et sans blocage.

## 2.2 Applications

**Exercice 19**

1. Donnez le multi-ensemble résultat de l'application de la règle *sum* : $x, y \rightarrow x + y$ à $M = \{\!|1,1,0,2,0,1|\!\}$ : .......
2. Combien d'étapes de calcul faut-il pour arriver au résultat ? ..........................................................
3. Quel(s) résultat(s) obtient-on en appliquant la règle $x, y \rightarrow x - y$ au multi-ensemble $M$ ?
   Même question avec $x, y \rightarrow \frac{x}{y}$

SOLUTION

*Selon l'ordre des réactions on peut obtenir* $\{\!|-1|\!\}$ *ou* $\{\!|1|\!\}$ *car la soustraction n'est ni commutative* $(1,2 \to -1$ *alors que* $2,1 \to 1)$, *ni associative puisque* $(1-2)-3 = -4 \neq 2 = 1-(2-3)$. *Même remarque pour* $(1/2)/3 = \frac{1}{2\times 3} \neq \frac{3}{2} = 1/(2/3)$

## Exercice 20

1. Donnez une règle qui à partir de $\{\!|0|\!\}$ génère un multi-ensemble contenant tous les entiers : ............................
2. Donnez le multi-ensemble qu'on obtient après trois pas de calcul : ........................................
3. Donnez une règle qui à partir de $\{\!|\textsc{g}(0)|\!\}$ génère exactement $\mathbb{N} = \{\!|0,1,2,\ldots|\!\}$ sans répétition : .........................................
4. Ajoutez une règle avec condition pour effectuer le crible d'Eratosthène qui élimine tout nombre qui admet un diviseur strictement plus petit que lui et différent de 1 : ..................................................

## Exercice 21

On considère un constructeur $\textsc{t}$ à un argument et un multi-ensemble contenant des éléments de type $\textsc{t}(e)$.

1. Donnez deux règles qui permettent de compter le nombre d'éléments de type $\textsc{t}$ d'un multi-ensemble. Utilisez des constructeurs $\textsc{t}$' et CARD afin de ne pas compter plusieurs fois le même élément.
2. Appliquez vos règles au multi-ensemble $\{\!|\textsc{t}(1),\textsc{t}(1),\textsc{t}(0),\textsc{t}(0)|\!\}$ : .......................................................
3. Donnez des règles pour calculer le cardinal d'un multi-ensemble.

# Chapitre 3

# Calculabilité

## 3.1 Fonction calculable

Intuitivement, une fonction calculable est une fonction $f : \mathbb{N}^p \to \mathbb{N}^*$ dont le résultat peut être obtenu par un procédé automatisable. Cette définition est imprécise et il existe plusieurs définitions mathématiques des fonctions calculables. Nous allons éclaircir cette notion.

**Définition 10 (au sens de Turing)** *Les **fonctions calculables** sont les machines de Turing qui ..................... pour toute entrée binaire $\omega \in \{0,1\}^*$.*

### 3.1.1 Hypothèse de Church-Turing

De 1931 à 1936, Alan Turing présente ses machines, Jacques Herbrand & Kurt Gödel – les systèmes d'équations HG, Stephen Cole Kleene – les fonctions $\mu$-récursives et Alonso Church propose le $\lambda$-calcul. Ces propositions sont des tentatives très différentes pour définir des fonctions calculables ... Elles se sont avérées être équivalentes. La preuve de ces équivalences consiste à trouver un codage d'un système dans un autre et réciproquement.

> *Puisque des modèles de calcul très différents conduisent à une notion équivalente de fonctions calculables, il est raisonnable de penser – mais ce n'est qu'une h ................ e – que la notion de fonctions calculables est indépendante du modèle de calcul. Puisqu'on ne parvient pas à concevoir un modèle de calcul plus puissant que ceux déjà connus, on adopte ces modèles comme définitions de la notion de* fonctions calculables.

En théorie de la calculabilité, on emploie aussi le terme historique de *«fonctions récursives»*. On lui préferera le terme *fonctions calculables* afin de pas confondre avec la notion de *fonctions récursives en programmtion* qui sont des fonctions qui s'appellent elle-mêmes (on dit qu'elles font des appels récursifs).

### 3.1.2 Réalisation d'une fonction par une machine de Turing

**On s'intéresse aux fonctions de** $\mathbb{N}^p \to \mathbb{N}^*$ qui associent à un vecteur $\vec{d} = (d_1, \ldots, d_p)$ représentant les $p$ données d'entrée un vecteur $\vec{r} = (r_1, \ldots, r_q)$ représentants $q$ résultats entiers.

**Notation** On a montré au § 1.6.1 qu'on pouvait représenter un vecteur d'entiers $\vec{d}$ par un mot binaire $\omega_d = [\vec{d}]_2$. Désormais, au lieu de considérer des fonctions $f : \mathbb{N}^p \to \mathbb{N}^*$, on s'intéressera à la version $f : \Sigma^* \to \Sigma^*$ avec $\Sigma = \{0,1\}$ qui travaille en représentation binaire.

Au lieu de $f(\vec{d}) = \vec{r}$ on écrira $f([\vec{d}]_2) = [\vec{r}]_2$ ou bien $f(\omega_d) = \omega_r$ où $\omega_d = [\vec{d}]_2$ et $\omega_r = [\vec{r}]_2$.

**Définition 11** *La* MT $M_f$ ***réalise/implémente*** *la fonction* $f : \Sigma^* \to \Sigma^*$ *si*

— *l'.................... de* $M_f$ *s'arrête pour tout mot* $\omega_d \in \Sigma^*$

— $M_f(\omega_d) = \mathbb{V}(\omega_r)$ *chaque fois que* $f(\omega_d) = \omega_r$

— $M_f(\omega_d) = \mathbb{F}$ *si* $f$ *n'est pas ............. pour* $\omega_d$ *ie.* $f$ *ne rend pas de résultat pour l'entrée* $\omega_d$

> $f$ est **calculable** si et seulement si **il existe** une MT $M$ qui **réalise** la fonction $f$
> *ie.* $M$ ............... pour tout mot de $\Sigma^*$ et $M(\omega_d) = f(\omega_d)$

## 3.2 Ensemble dénombrable

### 3.2.1 Définition et théorème fondamental

**Définition 12** *Un ensemble* $E$ *est* ***dénombrable*** *s'il est en bijection avec* $\mathbb{N}$*, noté* $E \simeq \mathbb{N}$

**Théorème 1 (de Cantor-Bernstein)** *il existe une bijection* $E \simeq \mathbb{N}$ *si et seulement si il existe deux fonctions injectives* $g : \mathbb{N} \to E$ *et* $h : E \to \mathbb{N}$.

Attention, à ne pas conclure que $(g, h)$ forment une bijection. Le théorème n'affirme pas que $\forall e \in E,\ g(h(e)) = e$ ni que $\forall n \in \mathbb{N},\ h(g(n)) = n$ ; il garantie seulement qu'une .................... existe.

**Remaque** Cette définition mathématique autorise à prendre pour $g$ et $h$ des *fonctions non-calculables*.

**Définition 13 (Piqûre de rappel : injection, surjection, bijection)**

*(a) Une fonction* $g : A \to B$ *est .................... si et seulement si* $g(a_1) = g(a_1) \Rightarrow a_1 = a_2$ *ou de manière équivalente* $a_1 \neq a_2 \Rightarrow g(a_1) \neq g(a_2)$.

> **Remarque :** Les injections sont courantes et utiles en informatique : il s'agit de fonctions qui convertissent des données d'un format $A$ dans un format $B$ en prenant garde : (1) d'être applicable à toute donnée de format $A$ et (2) de ne pas convertir vers le même $b \in B$ deux données différentes $a_1, a_2 \in A$.

*(b) Une fonction* $g : A \to B$ *est ....................... si et seulement si* $\forall b \in B,\ \exists a \in A,\ g(a) = b$.

> **Remarque :** En terme informatique une surjection est une convertion du format $A$ vers le format $B$ qui garantit que toute donnée $b \in B$ possible provient bien de la convertion par $g$ d'une donnée $a \in A$

*(c) Une fonction* $g$ *est .................... si et seulement si elle est injective et surjective.*

> **Remarque :** Une convertion de format qui est injective et surjective est réversible.

### 3.2.2 Ensembles dénombrables

**Exercice 22** ***Application du théorème de Cantor-Bernstein***

Utilisez le théorème pour démontrer que $\mathbb{N} \simeq \mathbb{N} \times \mathbb{N}$.

SOLUTION

*On prend* $g : \mathbb{N} \to \mathbb{N} \times \mathbb{N} : n \mapsto (\,..\,,\,..\,)$*, c'est une fonction injective puisqu'elle vérifie* $n \neq i \Rightarrow (\,..\,,\,..\,) = g(n) \neq g(\,.\,) = (i, i)$.

*On pend pour* $h : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *la fonction* $(n, p) \mapsto ..^{n}\ ..^{p}$ *qui est injective puisque la décomposition d'un nombre en puissance de facteurs ................. est unique.* ***Conclusion :*** *d'après le théorème de Cantor-Bernstein, il existe donc une ................* $\mathbb{N} \simeq \mathbb{N} \times \mathbb{N}$ *mais le théorème de la ........... pas.*

---

**Proposition 1** *Les ensembles $\mathbb{N}, \mathbb{N}^2, \ldots, \mathbb{N}^i$ pour $i \in \mathbb{N}$ sont chacuns dénombrables. Encore plus surprenant : $\mathbb{N}^* = \bigcup_{i\in\mathbb{N}} \mathbb{N}^i$ est dénombrable ie. en bijection avec $\mathbb{N}$.*

> **Preuve** Les ensembles précédents sont tous infinis, mais pas plus grand que $\mathbb{N}$. Nous allons le démontrer à l'aide du principe de numérotation de Cantor qui permet de construire explicitment une bijection entre chacun de ces ensembles et $\mathbb{N}$, voir exercice ci-après. □

**Définition 14 (Liste d'entiers, cf. INF124)** *L'ensemble $\mathbb{N}^* = \bigcup_{i\in\mathbb{N}} \mathbb{N}^i$ est l'ensemble des vecteurs d'entiers de taille 0, 1, 2, .... On peut l'interpréter comme l'ensemble des **listes d'entiers** de taille variable : il suffit d'écrire les vecteurs $[n_1, n_2, n_3]$ au lieu de $(n_1, n_2, n_3)$*

**Exemples :**

- $\mathbb{N}^1 = \{(0), (1), (2), \ldots\}$ contient les vecteurs à une seule composante $\simeq \{[0], [1], [2], \ldots\}$
- $\mathbb{N}^0$ contient un unique vecteur : () que l'on note [ ] en caml

**Principe de numérotation de Cantor** On a montré grâce au théorème de Cantor-Bernstein que $\mathbb{N}^2$ l'ensemble des couples d'entiers est dénombrable. Par contre, ce théorème bien utile ne donne pas la bijection, il dit seulement qu'elle existe.
Georg Cantor (1845-1918) a proposé une définition constructive de la bijection $\mathbb{N} \simeq \mathbb{N} \times \mathbb{N}$.
Il définit un tableau $[0..\mathbb{N}[\times[0..\mathbb{N}[$ qui contient tous les couples d'entiers possibles : à la ligne $\ell$ et la colonne $c$ on trouve le couple $(\ell, c)$. On obtient un tableau à deux dimensions (infinie) de la forme :

| $\mathbb{N}\times\mathbb{N}$ | $c=0$ | 1 | 2 | 3 | 4 | $\ldots$ |
|---|---|---|---|---|---|---|
| $\ell=0$ | $(0,0)_{\simeq 0}$ | $(0,1)_{\simeq 2}$ | $(0,2)_{\simeq 5}$ | $(0,3)_{\simeq 9}$ | $(0,4)_{\simeq 14}$ | $\ldots$ |
| 1 | $(1,0)_{\simeq 1}$ | $(1,1)_{\simeq ..}$ | $(1,2)_{\simeq ..}$ | $(1,3)_{\simeq .....}$ | $(1,4)_{\simeq .....}$ | $\ldots$ |
| 2 | $(2,0)_{\simeq 3}$ | $(2,1)_{\simeq ..}$ | $(2,2)_{\simeq .....}$ | $(2,3)_{\simeq .....}$ | $(2,4)_{\simeq 25}$ | $\ldots$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ |

Pour attribuer un numéro unique à chaque couple, on parcourt le tableau suivant les anti-diagonales de Cantor en commençant par donner le numéro 0 au couple (0,0). Ce procédé de numérotation établit une bijection entre $\mathbb{N}$ et l'ensemble $\mathbb{N} \times \mathbb{N}$ des couples d'entiers.

**Construction de la bijection de Cantor** On peut définir en $\Gamma$ le programme qui construit la relation bijection $C_2$ qui associe un numéro de Cantor $n$ à chaque couple $(\ell, c)$.

Les deux réactions ci-dessous construisent $C_2(n, (\ell, c))$ si et seulement si $n \simeq (\ell, c)$ par la numérotation de Cantor.

Au départ le multi-ensemble est réduit à un élément $\{\!|C_2(0, (0, 0))|\!\}$ qui va servir à lancer la numérotation de Cantor des vecteurs d'entiers, de taille quelconque. La numérotation de Cantor des vecteurs de 2 entiers se code en GAMMA de la manière suivante :

$$\begin{cases} C_2(n, (\ell, c)) \xrightarrow{\ell\geq 1} C_2(n+1, (\ell \ldots\ldots, c \ldots\ldots)) \\ C_2(n, (0, c)) \longrightarrow C_2(n+1, (\ldots\ldots\ldots\ldots, 0)) \end{cases}$$

**Exercice 23** ***Généralisation du principe de numérotation de Cantor à $\mathbb{N}^*$ (à chercher)***

On a montré que $\mathbb{N}^2$ l'ensemble des vecteurs d'entiers de taille 2 est dénombrable.

1. À partir de $c_2$ donnez en Gamma un procédé de numérotation des vecteurs d'entiers $(i, j, k)$ vus comme des couples $((i, j), k)$.
2. En dénombrable que $\mathbb{N}^3$ forme un ensemble dénombrable.
3. En déduire par itération du procédé que l'ensemble des vecteurs d'entiers de taille $n \in \mathbb{N}$ est dénombrable.

**Proposition 2** *L'ensemble $\mathbb{N}^*$ des listes d'entiers est dénombrable, donc l'ensemble des chaînes de caractères est dénombrable, et donc l'ensemble des machines de Turing et celui des programmes est dénombrable.*

**Preuve : Montrons $\mathbb{N}^*$ dénombrable.** Pour construite l'ensemble des listes d'entiers, on range les vecteurs d'entiers de tailles variables dans un tableau $[0..\mathbb{N}[\times[0..\mathbb{N}[$.

1. À la ligne 1 on range les ............... $(i)$ de taille ..
2. À la ligne 2 on range les couples $(i, j)$ dans l'ordre croissant de leur .......................... de Cantor : le $n$ de $C_2(n, (i, j))$
3. À la ligne $\ell$ on range les vecteurs de taille .. dans l'ordre de la numérotation de ............ des $C_\ell$ et ainsi de suite.

On obtient un tableau de la forme :

| $\mathbb{N}$ | $n = 0$ | 1 | 2 | 3 | 4 | ... |
|---|---|---|---|---|---|---|
| $C_1$ | (0) | (1) | (2) | (3) | (4) | ... |
| $C_2$ | (0, 0) | (1, 0) | (0, 1) | (2, 0) | (1, 1) | ... |
| $\vdots$ | | | | | | |

On réapplique alors le ................ des anti-..................... de ............ afin d' ................ un numéroa à chaque ............. du tableau. On parvient ainsi à établir une bijection entre $\mathbb{N}$ et l'ensemble $\mathbb{N}^*$ des listes d'entiers. □

Ensembles dénombrables : $\mathbb{N}, \mathbb{N}^2, \ldots, \mathbb{N}^i, \ldots, \mathbb{N}^*$ = en bijection avec $\mathbb{N}$

### 3.2.3 Ensembles non-dénombrables : Principe de ................................... de Cantor

On peut montrer en utilisant le principe de diagonalisation de Cantor [1] que les ensembles suivants ne sont **pas dénombrables** :

— $\mathbb{N} \to \mathbb{N} = \{ f \mid f : \mathbb{N} \to \mathbb{N} \}$ = l'ensemble des fonctions à un ...................... entier et un résultat entier

— $\mathbb{N} \to Bool = \{p^? \mid p^? : \mathbb{N} \to Bool\}$ = l'ensemble des prédicats à un ...................... entier

— $\mathscr{P}(\mathbb{N}) = \{ S \mid S \subseteq \mathbb{N} \}$ = l'ensemble des .............. de $\mathbb{N}$, *ie.* l'ensemble de tous les .........-...................... de $\mathbb{N}$

— $[0, 1[\cap\mathbb{R}$ = l'ensemble des réels du ................. ouvert $[0, 1[$.

**Preuve : Montrons $\mathbb{N} \to \mathbb{N}$ non dénombrable.** Considérons une fonction $f : \mathbb{N} \to \mathbb{N}$. Elle est complètement ............ par un tableau $[0...\mathbb{N}[$ qui indique pour chaque entier $n$ la valeur $f(n)$ associée. On range alors les fonctions dans un tableau $[0..\mathbb{N}[\times[0..\mathbb{N}[$ comme suit :

| $\mathbb{N} =$ | 0 | 1 | 2 | 3 | 4 | 5 | ... |
|---|---|---|---|---|---|---|---|
| $nul = f_0$ | $\boxed{0}$ | 0 | 0 | 0 | 0 | 0 | ... |
| $id = f_1$ | 0 | $\boxed{1}$ | 2 | 3 | 4 | 5 | ... |
| $inc = f_2$ | 1 | 2 | $\boxed{3}$ | 4 | 5 | 6 | ... |
| $dbl = f_3$ | 0 | 2 | 4 | $\boxed{6}$ | 8 | 10 | ... |
| $f_4$ | .. | .. | .. | .. | $\boxed{f_4(4)}$ | .. | ... |
| $\vdots$ | | | | | | | |

1. Certains parlent de la «diagonale du ......» puisque les intuitions géniales de Cantor sur l'infini et ses paradoxes l'ont conduit plus d'une fois en hôpital psychiatrique avec interdiction formelle de refaire des maths. Heureusement qu'il n'a pas respecté l'injonction de ses médecins.

On peut donc repérer une fonction $\mathbb{N} \to \mathbb{N}$ par son .............. de ligne. Par exemple, la fonction identité serait $f_1$.
SUPPOSONS que l'ensemble $\mathbb{N} \to \mathbb{N}$ soit dénombrable c'est-à-dire en bijection avec $\mathbb{N}$. Alors il existe un tableau $[0..\mathbb{N}[\times[0..\mathbb{N}[$ comme le précédent qui décrit la ................ $\mathbb{N} \to (\mathbb{N} \to \mathbb{N}) : \ell \mapsto f_\ell$.

Autrement dit le tableau contient ........... les fonctions $\mathbb{N} \to \mathbb{N}$ : la ligne $\ell$ définit la fonction $f_\ell$.

Mais alors la fonction $g : \mathbb{N} \to \mathbb{N}$ définie par $g(n) = f_n(n)+1$ doit .................... dans le tableau à une certaine ligne, disons $\ell$, donc $g = f_\ell$.

> **Exemple :** La fonction $g$ correspond à la diagonale du tableau ........................ de 1. Dans le cas du tableau précédent, la fonction $g$ serait $g(0) = 0+1,\ g(1) = 1+1,\ g(2) = 3+1,\ g(3) = 6+1,\ g(4) = f_4(4)+1, \ldots$

Dans ce cas $g(\ell) = f_\ell(\ell)$ d'après l'............ précédente et $g(\ell) = f_\ell(\ell)+1$ par .................. de $g$ : CONTRADICTION.

**Conclusion :** On a supposé $\mathbb{N} \to \mathbb{N}$ dénombrable et on aboutit à une contradiction, donc $\mathbb{N} \to \mathbb{N}$ ................. dénombrable. □

**Exercice 24** ***(à chercher pour s'entraîner au cas où ça tomberait en ................)***

Les trois autres exemples se démontrent de la même manière : on construit un tableau $[0..\mathbb{N}[\times[0..\mathbb{N}[$ censé contenir tous les éléments (un par ligne) et on montre avec une construction diagonale de la forme $f_n(n)$ qu'on peut construite un élément qui n'est pas dans le tableau.

- Pour $\mathbb{N} \to \mathit{Bool}$ la ligne $\ell$ définit le prédicat $p_\ell$. La case à la ligne $\ell$ colonne $c$ contient le booléen [2] ($\mathbb{V}$ ou $\mathbb{F}$) qui correspond au résultat du prédicat $p_\ell$ pour l'entier $(c)$. Pour obtenir une contradiction on procède comme dans le cas $\mathbb{N} \to \mathbb{N}$ : on utilise la diagonale du tableau pour définir un prédicat qui n'apparait pas dans le tableau.
- Pour $\mathscr{P}(\mathbb{N})$ la ligne $\ell$ contient le sous-ensemble $S_\ell$ de $\mathbb{N}$ défini par sélection des colonnes : un 0 dans la colonne $c$ indique que $c \notin S_\ell$ un 1 dans la colonne $c$ indique que $c \in S_\ell$.
- Pour $\mathbb{R} \cap [0,1[$, la ligne $\ell$ contient le réel $r_\ell$ de la forme $\mathbf{0},d_0d_1d_2\ldots$ où chaque décimale $d_c$ est un entier entre 0 et 9 inscrit dans la colonne $c$.

## 3.3 Ensemble énumérable

**Définition 15** *Un ensemble $E$ est* ***énumérable***

(i) *s'il est* ***dénombrable****, c'est-à-dire en bijection avec $\mathbb{N}$.*
*Il existe donc une fonction surjective $\mathbb{N} \to E$, que l'on nommera get,*
*et qui vérifie $\forall e \in E,\ \exists n \in \mathbb{N},\ get(n) = e$*

(ii) *s'il existe une MT $M_{get}$ qui* ***réalise*** *la fonction get.*

**Exercice 25** $\Sigma^*$ ***est énumérable***

Considérons $\Sigma = \{0,1\}$. Montrez que $\Sigma^*$, l'ensemble des mots binaires, est énumérable en exhibant une MT $M_{get}$ qui réalise une fonction surjective de $\mathbb{N} \to \{0,1\}^*$.

SOLUTION

(i) *On a montré à l'exercice 23 que $\mathbb{N}^*$ était dénombrable. Or $\Sigma = \{0,1\} \subseteq \mathbb{N}$ donc $\Sigma^* \subseteq \mathbb{N}^*$ est lui aussi dénombrable.*

(ii) *On prouve que la fonction get : $\mathbb{N} \to \Sigma^* : [n]_2 = \omega.1 \mapsto \omega$ est surjective et on donne une MT $M_{get}$ qui réalise cette fonction. La fonction get n'est pas définie pour 0 mais on peut la définir : $get(0) = \epsilon$. Remarquez que get n'est pas une bijection puisque $get(0) = \epsilon = get(1)$ mais ça reste une surjection.*
*Montrons que get est une surjection : il faut montrer que pour tout élément $\omega \in \Sigma^*$ il existe un entier binaire $n \in [\mathbb{N}]_2$ tel que $get(n) = \omega$.*

2. 0 ou 1 si vous préférez

**Preuve** Pour atteindre par *get* l'élément $\omega$, on prend l'entier $n$ qui s'écrit $\omega.1$ en binaire. On a bien *get* $(n) = \omega$. $\square$

---

Ensemble $E$ énumérable = $E$ dénombrable et il existe une MT $M_{get}$ qui ............ la fonction .................... *get* : $\mathbb{N} \to E$ qui à $i$ associe le $i^e$ élement de $E$.

### Exercice 26 $\Sigma^* \times \Sigma^*$ *est énumérable*

Ce résultat nous sera utile au chapitre 4. Considérons $\Sigma = \{0, 1\}$. Montrez que $\Sigma^* \times \Sigma^*$, l'ensemble des couples de mots binaires, est énumérable en exhibant une MT $M_{get}$ qui réalise une fonction surjective de $\mathbb{N} \to \{0,1\}^* \times \{0,1\}^*$.

**Indication :** Utilisez la fonction $get_{\Sigma^*} : \mathbb{N} \to \Sigma^*$ de l'exercice 25 et la bijection de Cantor $C_2 : \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ définie à l'exercice 23.

SOLUTION

*On construit la fonction surjective get* : $\mathbb{N} \to \Sigma^* \times \Sigma^*$ *de la manière suivante*

$$\begin{array}{lccccc} get: & \mathbb{N} & \xrightarrow[bij]{C_2} & \mathbb{N} \times \mathbb{N} & \xrightarrow[surj]{get_{\Sigma^*}} & \Sigma^* \times \Sigma^* \\ & n & \mapsto & C_2(n) = (i,j) & \mapsto & \big(get_{\Sigma^*}(i), get_{\Sigma^*}(j)\big) \end{array}$$

---

## Résumé

Ce chapitre nous a appris que **ce qui est calculable par un procécédé automatique correspond à ce que peuvent faire les machines de Turing.**

Nous savions qu'on pouvait coder dans l'alphabet binaire $\Sigma = \boxed{0}, \boxed{1}$ tout type de données : entiers, symboles, vecteurs, images, .... Nous avons vu que **l'ensemble infini, $\Sigma^*$, de tous les mots binaires (toutes les données possibles) est énumérable.** Puisqu'une machine de Turing peut se représenter par un vecteur $(\Sigma, \mathcal{Q}, \searrow\!\bigcirc, \delta, \mathcal{A}cc, \mathcal{E}rr)$, on peut la représenter par le codage binaire de ce vecteur ; on détaillera ce codage au chapitre suivant. On en conclut que les MT sont des mots de $\Sigma^*$ ; elles sont donc dénombrables. On verra même que **les machines de Turing sont énumérables**.

Par ailleurs nous avons montré qu'**il existe des ensembles infinis, « trop infini » pour être énumérables** : par exemple, celui des prédicats $\mathbb{N} \to Bool$ et celui des fonctions $\mathbb{N} \to \mathbb{N}$.

La conclusion est qu'**il existe infiniment plus de fonctions que de machines de Turing :** il existe une infinité de fonction $\mathbb{N} \to \mathbb{N}$ non calculables, c'est-à-dire des fonctions pour lesquelles il n'y a pas de machine de Turing correspondante.

**Exercice 27** ***Connaissez-vous les définitions du cours ?***

On considère l'alphabet $\Sigma = \{\boxed{0}, \boxed{1}\}$

— Un ensemble est dénombrable si ...............................................................................

— une MT réalise un fonction $f : \Sigma^* \to \Sigma^*$ si .......................... et si ..........................

.........................................................................................................

— Une MT $M$ énumère un ensemble $E$ si elle .................................. une fonction ............ive *get* : ... → ... c'est-à-dire que tout .............. .. de ... il existe un .......... .. de ... tel que *get*$(n) = e$, et $M(\ldots\ldots\ldots) = \ \ldots ([..]_2)$.

— Un ensemble est énumérable si ...............................................................................

.........................................................................................................

— une fonction $f : A \to B$ qui convertit des données d'un format $A$ dans un format $B$ doit

(1) être ............ pour ................................................................, et

(2) ne pas convertir vers ............................ des données .................................

Autrement dit la fonction $f$ doit au minimum être ........ive, *ie.*

$$\forall \ldots\ldots\ldots \in \ldots, \ \ldots. \neq \ldots. \implies f(\ldots.) \ldots f(\ldots.)$$

— Donnez trois ensembles non-dénombrables : ..........................................................

— Donnez trois ensembles infinis dénombrables : ......................................................

— Une fonction est calculable si ................................ et ................................

.........................................................................................................

# Chapitre 4

# Décidabilité

Le chapitre précédent nous a appris qu'**il existe infiniment plus de fonctions que de machines de Turing :** il existe une infinité de fonctions $\mathbb{N} \to \mathbb{N}$ non-calculables, c'est-à-dire des fonctions pour lesquelles il n'y a pas de machine de Turing correspondante.

Soit, **mais lesquelles ? Comment les reconnaître ? Peut-on donner des exemples de fonctions non-calculables ?** Voici les questions qui vont guider la suite du cours. Dorénavant notre objectif sera de mieux cerner ce qui n'est pas calculable.

Pour apporter des réponses nous allons simplifier la question et **réduire la question de la calculabilité d'une fonction**

> « existe-t'il une MT capable de calculer le résultat $\omega_r = f(\omega_d)$ à partir de la donnée $\omega_d$ ? »

**à celle de la décidabilité d'une relation**, qui se contente de répondre $\mathbb{V}$ ou $\mathbb{F}$ lorsqu'on lui donne l'entrée $\omega_d$ ainsi qu'une proposition de résultat $\omega_r$

> « existe-t'il une MT qui reconnaît les couples $(\omega_d, \omega_r)$ tels que $\omega_r = f(\omega_d)$ ? »

Dans ce chapitre nous allons définir la décidabilité et établir une correspondance entre fonctions et relations.

## 4.1 Ensemble décidable

On s'intéresse à des ensembles d'éléments pris dans l'ensemble $\mathscr{U}$ de tous les éléments possibles, on appelle l'ensemble $\mathscr{U}$ l'**Univers.**

**Définition 16** *La **fonction indicatrice/caractéristique** d'un sous-ensemble $E \subseteq \mathscr{U}$, notée $\in^?_E$ est un prédicat $\mathscr{U} \to Bool$ qui **test l'appartenance à** $E$ des éléments $u$ de l'Univers. Il est défini par :*

$$\begin{array}{lll} \in^?_E (u) = \mathbb{V} & \iff & u \in E \\ \in^?_E (u) = \mathbb{F} & \iff & u \notin E \end{array}$$

**Définition 17 (ensemble décidable)** *Un ensemble $E \subseteq \mathscr{U}$ est **décidable** si et seulement si*

*(i) l'univers $\mathscr{U}$ est **énumérable***

*(ii) et si la .................. **indicatrice** $\in^?_E$ de $E$ est **calculable***

| Ensemble Décidable = Univers énumérable & Appartenance ...................... |
|---|

**Proposition 3** *Considérons l'alphabet $\Sigma = \{0,1\}$. **L'ensemble $\Sigma^*$ des mots binaires est** ....................*

**Preuve** On doit montrer $(i)$ et $(ii)$. Auparavant il faut préciser l'Univers dans lequel on se place. Puisque les éléments de $\Sigma^* = \{0,1\}^*$ sont des mots binaires on choisit $\mathscr{U} = \Sigma^*$. On a bien $\Sigma^* \subseteq \mathscr{U}$.

$(i)$ On a déjà montré à l'exercice 25 que $\Sigma^*$, l'univers est énumérable. Ce qui démontre $(i)$.

$(ii)$ $\in^?_{\Sigma^*}$, la fonction indicatrice de $\Sigma^*$ est calculable puisqu'il existe une MT qui termine toujours pour tout mot de l'univers et qui accepte ........ les mots de $\Sigma^*$ : c'est la MT réduite à une seul état ........... et ..................., $\to\circledcirc$.

On peut généraliser la proposition à n'importe quel alphabet $\Sigma = \{s_1, \ldots, s_{2^n}\}$ en codant les symboles par des vecteurs de $n$-bits comme dans l'exercice 13 et en considérant l'univers $\mathscr{U} = \{0,1\}^n$. □

## 4.2 Langage décidable (dit aussi *«langage récursif»*)

Dans la suite on s'intéresse aux cas particuliers des **langages** *ie.* des ....................... de mots écrits sur un **alphabet fini** $\Sigma$. Un langage est donc un sous-ensemble de l'univers $\mathscr{U} = \Sigma^*$ qui est é.....................e (cf. exercice 25). La définition 17 peut alors être simplifiée dans le cas des langages :

**Définition 18 (Langage décidable)** *Soit $\Sigma$ un alphabet fini.*
*Un langage $L \subseteq \Sigma^*$ est **décidable** si et seulement si $\in^?_L$ la **fonction** ........................ de $L$ est **calculable**.*

Langage Décidable = Alphabet fini (donc univers $\Sigma^*$ énumérable) & Appartenance ......................

**Vocabulaire historique** En théorie de la calculabilité, on emploie aussi le terme historique *«langage récursif»* pour langage décidable, en référence au fait que la fonction indicatrice est *«récursive»* au sens de calculable.

**Proposition 4 (Langage décidable, définition équivalente à la définition 17)** *Un langage $L$ est **décidable** s'il existe un machine de Turing $M_L$ dont l'exécution* ................. ***pour toute entrée** $\omega \in \Sigma^*$ avec $M_L(\omega) = \mathbb{V}$ si $\omega \in L$ et $M_L(\omega) = \mathbb{F}$ si $\omega \notin L$. On dit que la machine $M_L$ **décide** le langage $L$.*

**Preuve** La machine de Turing $M_L$ code la fonction ................... $\in^?_L$ puisque $M_L(\omega) = \mathbb{V} \iff \omega \in L$ et $M_L(\omega) = \mathbb{F} \iff \omega \notin L$. Mais alors, la fonction indicatrice de $L$ est .................. puisque $M_L$ s'arrête pour toute entrée $\omega \in \Sigma^*$. **Conclusion :** $L$ est donc un langage .................. d'après la définition 18. □

$M$ décide $L$ signifie $M$ ................ pour tous les mots de $\Sigma^*$ et
$M$ répond $\mathbb{V}$ si ............. et répond .... si $\omega \notin L$

Ne pas confondre « $M$ **décide le langage** $L$ » et « $M$ **reconnaît le langage** $L$ » ; les deux notions sont différentes. Une machine de Turing peut reconnaître un langage et ne pas le décider. Relisez les défintions pour découvrir cette subtile différence.

## 4.3 Calcul d'une fonction / Décision d'une relation

**Définition 19 (Relation binaire décidable)** *Soit $R \subseteq \Sigma^* \times \Sigma^*$ une relation binaire. La* MT *$M$* ***décide*** *la relation $R$ si et seulement si*

— *l'.................... de $M$ sur tout mot $(\omega_d, \omega_r) \in \Sigma^* \times \Sigma^*$ s'arrête*[1]

— *$M(\omega_d, \omega_r) = \mathbb{V}$ chaque fois que $(\omega_d, \omega_r) \in R$*

— *$M(\omega_d, \omega_r) = \mathbb{F}$ chaque fois que $(\omega_d, \omega_r) \notin R$*

### 4.3.1 Codage d'une fonction calculable sous la forme d'une relation décidable

Une fonction $f : \Sigma^* \to \Sigma^*$ peut être représentée par **une relation** $R_f \subseteq \Sigma^* \times \Sigma^*$ **c'est-à-dire un .................... de couples** $(\omega_d, \omega_r)$ tels que $f(\omega_d) = \omega_r$.

$$R_f = \{ (\omega_d, \omega_r) \mid f(\omega_d) = \omega_r \}$$

**Une relation $R_f$ peut aussi être représentée par un ..................** qui répond $\mathbb{V}$ si le couple $(\omega_d, \omega_r)$ appartient à l'ensemble $R_f$ et répond $\mathbb{F}$ sinon. Un tel prédicat est exactement la fonction ....................... de $R_f$.

Les notions suivantes sont équivalentes :

| | |
|---|---|
| $M_f(\omega_d) = \mathbb{V}(\omega_r)$ | MT *qui ................ la fonction $f$* |
| $\equiv$ | |
| $f(\omega_d) = \omega_r$ | *fonction $f$* |
| $\equiv$ | |
| $(\omega_d, \omega_r) \in R_f$ | *relation $R_f = \{ (\omega_d, \omega_r) \mid f(\omega_d) = \omega_r \}$* |
| $\equiv$ | |
| $M_{R_f}(\omega_d, \omega_r) = \mathbb{V}$ | MT *qui ........... la relation$R_f$* |

**Proposition 5 (Équivalence entre Calculabilité et Décidabilité)**

*La fonction $f : \Sigma^* \to \Sigma^*$ est calculable $\iff$ La relation binaire $R_f \subseteq \Sigma^* \times \Sigma^*$ est décidable.*

**Que dit cette proposition ?** Que si une fonction $f$ est calculable, sa relation $R_f$ est décidable (et réciproquement). On peut donc réduire l'étude des fonctions ..................... à l'étude des relations binaires ..................... En fait c'est principalement l'implication ($\Rightarrow$) qui nous intéresse « $f$ **calculable** $\Longrightarrow$ $R_f$ **décidable** » ou plutôt sa version équivalente sous forme **contraposé** :[2]

$$\boxed{R_f \text{ non-décidable} \Longrightarrow f \text{ non-calculable}}$$

Rappelons que notre objectif est d'exhiber des fonctions non-calculables. L'implication précédente dit qu'il suffit d'exhiber des ensembles $R_f$ non-décidables.

On ne s'en servira pas par la suite mais on démontre au passage la **réciproque**[3] « $f$ **calculable** $\Longleftarrow$ $R_f$ **décidable** » ce qui prouve l'équivalence entre les notions de calculabilité et décidabilité.

1. On exclut ainsi le cas $M(\omega_d, \omega_r) = ?$. Il ne reste alors que deux cas possibles.
2. La ..................... est simplement l'application de l'équivalence logique $A \Rightarrow B \;\equiv\; \neg B \Rightarrow \neg A$
3. Ne pas confondre réciproque et contraposé. La .................... de $A \Rightarrow B$ est $B \Rightarrow A$ : elle ne se déduit pas de $A \Rightarrow B$.

### 4.3.2 Preuve de l'équivalence (Proposition 5) par une double réduction

La preuve de la propostion 5 n'est pas compliquée : elle ne fait pas appel à des concepts mathématiques profonds et c'est un exemple de raisonnement courant en informatique appelé *réduction* : il s'agit de « réduire une question à une question plus simple ».

**Preuve**

($\Rightarrow$) Montrons $f$ **calculable** $\Rightarrow$ $R_f$ **décidable :** On suppose que la fonction $f$ **est calculable**, ce qui signifie[4] qu'il existe une MT $M_f$ qui termine pour tout mot $\omega$ de $\Sigma^*$ et telle $M_f(\omega_d) = \mathbb{V}(\omega_r)$ si $f$ est définie pour $\omega_d$ et $f(\omega_d) = \omega_r$. On doit montrer que **l'ensemble** $R_f \subseteq \Sigma^* \times \Sigma^*$ **est décidable.** Pour cela, on doit montrer deux choses :

($i$) on doit montrer que l'univers $\mathscr{U} \stackrel{\text{def}}{=} \Sigma^* \times \Sigma^*$ est énumérable (voir exercice 26).

($ii$) on doit montrer qu'il existe une machine de Turing $M_{R_f}$ qui décide le langage $R_f \stackrel{\text{def}}{=} \{(\omega_d, \omega_r) \mid f(\omega_d) = \omega_r\}$, c'est-à-dire[5] qu'on doit exhiber une MT qui termine pour toute entrée $(\omega, \omega')$ de $\mathscr{U}$, qui répond $\mathbb{V}$ si $(\omega, \omega') \in R_f$ et qui répond $\mathbb{F}$ si $(\omega, \omega') \notin R_f$.

**Pour montrer qu'une telle machine de Turing existe, on va la construire :** On définit $M_{R_f}$ comme une machine de Turing à deux bandes ; on a vu qu'il était possible ensuite de la transformer en une MT classique à une bande. Pour interroger le prédicat $M_{R_f}$ sur le couple $(\omega_d, \omega_r)$ on écrit $\omega_d$ sur une première bande $B_d$ et $\omega_r$ sur la seconde bande $B_r$. On lance $M_f$ sur la bande $B_d$, l'exécution s'arrête et écrit le résultat de $\omega_r = f(\omega_d)$ sur la bande $B_d$. On utilise ensuite la MT $M_{eq}$ pour tester si les bandes $B_d$ et $B_r$ sont identitques. $M_{eq}$ termine dans un état $\circledcirc$ si $B_d = B_r$ et dans $\otimes$ sinon.

$$M_{R_f}(\omega_d, \omega_r) \stackrel{\text{def}}{=} [B_d := \omega_d \ ; \ B_r := \omega_r \ ; \ B_d := M_{..}\ (B_d) \ ; \ M_{....}\ (B_d, B_r)]$$

($\Leftarrow$) Montrons $R_f$ **décidable** $\Rightarrow$ $f$ **calculable :** On suppose que la relation $R_f$ **est décidable**, ce qui signifie qu'il existe une MT $M_{R_f}$ qui termine pour tout mot $(\omega_d, \omega_r)$ de $\Sigma^* \times \Sigma^*$ et répond $\mathbb{V}$ si $(\omega_d, \omega_r) \in R_f$ et répond $\mathbb{F}$ sinon. Montrons qu'**on peut alors constuire une machine de Turing** $M_f$ qui pour chaque entrée $\omega_d$ ............ le résultat de $f(\omega_d)$.

On construit une machine de Turing $M_f$ à ........ bandes $B_d, B_r, B_n$ qu'on pourrait ....................... en une MT classique à ...................... Puisque $\Sigma^*$ est ....................... (voir l'exercice 25), on peut utliser la MT $M_{.......}$ : $[\mathbb{N}]_2 \to \Sigma^*$ pour énumérer les mots de $\Sigma^*$.

$M_f$ calcule $f$ de la manière suivante, pas efficace du tout mais qui .............. et donne le résultat attendu :

(1) On inscrit $\omega_d$ sur la bande $B_d$ et $0$ sur la bande $B_n$

(2) On recopie la bande $B_n$ sur la bande $B_r$.

(3) On applique $M_{\textit{get}}$ sur $B_r$, on obtient un mot de $\Sigma^*$.

(4) On applique $M_{R_f}$ sur $(B_d, B_r)$.

(5) Si $M_{R_f}$ répond $\mathbb{V}$ alors, par définition de $R_f$, le contenu de $B_r$ est le .............. de $f(\omega_d)$ cherché.

(6) Si $M_{R_f}$ répond ... alors on passe au mot ............. de $\Sigma^*$ : on applique $M_{inc}$ sur la bande $B_n$ et on .............. à l'étape (2).

$$M_f(\omega_d) \stackrel{\text{def}}{=} \left[\begin{array}{l} B_d := \omega_d \ ; \\ B_n := 0 \ ; \\ \text{do } [B_r := M_{......}\ (B_n) \ ; \ B_n := M_{......}\ (B_n)] \text{ until } (M_{......}\ (B_d, B_r)) \ ; \\ \text{return } B_r \end{array}\right]$$

4. par définition de « fonction calculable ».
5. d'après la définition de « MT qui décide »

Pour garantir la terminaison de cette MT, il faut montrer que pour chaque entrée $\omega_d \in \Sigma^*$ il existe un résultat $\omega_r$ qui correspond au résultat de $f$. Examinons le cas d'une fonction $\hat{f} : \Sigma^* \setminus \{\,\Omega\,\} \to \Sigma^*$ qui ne serait pas définie si le mot en entrée est $\Omega$. Dans ce cas, on ne parviendrait jamais à trouver le résultat $\omega_r$ correspondant à l'entrée $\omega_d = \Omega$ puisque le résultat de $\hat{f}$ n'est pas défini pour $\Omega$. Il n'y aurait donc aucun couple $(\Omega, \ldots)$ dans $R_{\hat{f}}$ et la MT qu'on a définie chercherait donc indéfiniment un couple $(\Omega, \omega)$ qui satisfasse $R_{\hat{f}}$ et ne terminerait pas.

Pour résoudre ce problème on impose que les fonctions soient définies pour toute entrée, quitte à forcer une fonction $\hat{f}$ à rendre un symbole spécifique ($\epsilon$ ou $\perp$) pour les entrées où elle n'est pas définie. On compléte alors la fonction $\hat{f}$ pour l'étendre à tout $\Sigma^*$ de la façon suivante : $\hat{f}(\Omega) = \epsilon$ ou $\hat{f}(\Omega) = \perp$ selon la convention choisie.

Dans ce cours, on choisira de retourner $\epsilon$ mais pour être plus rigoureux il faudrait créer un caractère nouveau $\perp$ et étudier les fonctions $\Sigma^* \to \Sigma^* \cup \{\,\perp\,\}$ Ce qui ne change pas fondamentalement les preuves mais complique un peu la présentation. $\square$

**Résumé**

Le raisonnement de la section précédente se nomme **réduction** : On a réduit l'étude des fonctions calculables par les MT à la question un peu plus simple[6] des langages décidables par les MT. On reverra la notion de réduction lorsqu'on abordera le problème de l'arrêt d'une machine de Turing.

## 4.4 La MT universelle, un interpréteur de machines de Turing

A.Turing a défini les MT afin d'étudier les limites du calculable : ce qui est faisable par un ordinateur, ce qui ne le sera jamais quel que soit l'ordinateur ou le langage de programmation utilisé.
Les raisonnements qui lui ont permis d'exhiber des problèmes concrets qui n'ont pas de solution informatique reposent sur l'utilisation d'une MT universelle $U$ capable d'exécuter toute machine de Turing $M$ sur un mot $\omega$. Nous allons donc présenter le codage qui permet de représenter par un mot binaire $m$, une MT $M$ *ie.* $m = [M]_2$. Ensuite nous présenterons le principe de fonctionnement de la MT universelle $U$. C'est une MT à deux bandes qui prend en premier argument la représentation binaire d'une MT $M$ inscrite sur la bande $B_2$ sous la forme d'un mot binaire $m$, puis prend en second argument un mot $\omega \in \{0,1\}^*$ inscrit sur la bande $B_1$. La MT universelle $U$ est conçue de sorte que

> *l'exécution de $U(m)(\omega)$ donne le même résultat (soit $\mathbb{V}$, soit $\mathbb{F}$, soit ?) que l'exécution de $M$ sur $\omega$.*

Ce qu'on résume par l'égalité :

$$U(m) \simeq M \text{ si } m = [M]_2$$
$$\text{En particulier, } U(m)(\omega) = M(\omega) \quad \text{et} \quad U(m)(\omega) \to^{\infty} \text{ si et seulement si } M(\omega) \to^{\infty}$$

**Remarque :** $m$ représente la description de $M$, *ie.* c'est un programme (pas très lisible mais il décrit bien un algorithme). $U$ est donc un **interpréteur** qui lit le programme $m$ et l'exécute sur $\omega$, tandis que $M$ peut être considérée comme une **version compilée** de $m$. $M(\omega)$ correspond alors à un appel au **programme exécutable** $M$ sur la donnée $\omega$ passée en ligne de commande.

### 4.4.1 Représentation binaire d'une machine de Turing

Considérons une MT $M = (\Sigma, \mathscr{Q}, \mathbf{q}_i, \delta)$ avec $\Sigma = \{0, 1, \square, \$\}$.
Pour coder cette représentation de $M$ sur le ruban on va considérer un alphabet $\Sigma_U$ contenant les symboles de $\Sigma$ et enrichi de nouveaux symboles :

— A,E,Q, (, :, ) afin de représenter les états quelconque par Q, accepteur par A, rejet par R.

6. La question est plus simple puisque dans le cas de la décision, on ne se préoccupe pas de ce qu'il y a d'écrit sur le ruban après l'exécution, mais uniquement de l'arrêt de la MT et, le cas échéant, de l'état dans lequel elle s'est arrêtée : soit $\circledcirc$, soit $\cancel{\otimes}$ (= $\bigcirc$ ou $\otimes$).

**Exemples :**

- l'état 5 sera représenté par (A : 101) ce qui donne

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $^\infty\square$ | ( | A | : | 1 | 0 | 1 | ) | $\square^\infty$ |

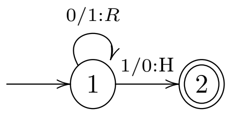
  sur le ruban
- l'état $\otimes$ sera représenté par (E : 0)
- l'état 2 sera représenté par (Q : 01).

— L,H,R afin de représenter les déplacements $d \in \{$L,H,R$\}$ de la tête lors des transitions.

**Exemple :** La transition $1 \xrightarrow{\ell/e:d} 2$ avec $\ell, e \in \Sigma$ sera représentée sur le ruban par

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ( | Q | : | 1 | ) | $\ell$ | $e$ | $d$ | ( | A | : | 0 | 1 | ) |

— Les transitions sont inscrites les unes derrière les autres. La séparation entre transitions est indiquée par la succession des cases | ) | ( |.

**Exemple :** La MT 

sera représentée par la séquence de transitions

(Q : 1)01R(Q : 1) (Q : 1)10H(A : 01)

Pour obtenir le mot $m$ décrivant $M$ on fait précéder la séquence des transitions de l'état initial suivi du symbole \$, on obtient alors le mot [7]

$m =$

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| (Q : 1) | \$ | (Q : 1) | 0 | 1 | R | (Q : 1) | (Q : 1) | 0 | 1 | R | (A : 01) |

**Définition 20** *(codage binaire des machine de Turing) On note* $\mathscr{M} = \{m \in \{0,1\}^* \mid m = [M]_2,\ M \in \text{MT}\}$ *l'ensemble des mots binaires qui correspondent à des* MT *sur l'alphabet* $\{0, 1, \$\} \cup \{\square\}$.

**Remarque :** Si on lance l'exécution de $U$ sur un couple $(m, \omega)$ où $m \notin \mathscr{M}$, c'est-à-dire lorsque $m$ ne correspond pas à codage valide de MT alors le comportement de $U$ est imprévisible. Néanmoins il est possible d'étendre la MT $U$ pour qu'une première phase vérifie si le mot $m$ correspond bien à un codage de MT.

### 4.4.2 Fonctionnement de la machine de Turing universelle

On définit la MT $U$ comme une machine à deux bandes. L'exécution de $U(m)(\omega)$ doit simuler l'exécution de $M$ sur le mot $\omega$.

— La bande $B_2$ contient le mot $m$ correspondant à la représentation binaire de $M$, cf. § 4.4.1.

— La bande $B_1$ contiendra la configuration courante de $M$ *ie.* $(\omega_L, \mathbf{q}, \ell.\omega_R)$ où $\omega_L$ est la partie du ruban de $M$ située à gauche de $T_M$ (la tête de $M$), $\ell$ est le symbole courant sur lequel pointe $T_M$, et $\omega_R$ est la partie du ruban située à droite de $T_M$. On représente la configuration $(\omega_L, \mathbf{q}, \ell.\omega_R)$ de la manière suivante sur la bande $B_1$ où l'état $\mathbf{q} = (t : n)$ avec $t \in \{$A,R,Q$\}$ est le statut de l'état, $n \in \mathbb{N}$ son numéro d'état et $[n]_2$ la représentation binaire de $n$.

$B_1 =$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $^\infty\square$ | $\omega_L$ | ( | $t$ | : | $[n]_2$ | ) | $\ell$ | $\omega_R$ | $\square^\infty$ |
| | | | | | | | $\uparrow$ $T_M$ | | |

Par souci de lisibilité on désignera désormais par | $(\mathbf{q}_n)$ | un état | ( | $t$ | : | $[n]_2$ | ) |.

7. Pour rester lisible, on ne fait pas apparaître les séparations en cases des états.

**Exécution de** $U(m)(\omega)$

1. Au départ $B_1$ contient le mot $\omega$ et $B_2$ contient le mot $m = [M]_2 = (\mathbf{q}_i)\$ \underbrace{\ldots\delta\ldots}_{transitions}$ .

   Par convention, les têtes $T_1, T_2$ des bandes 1 et 2 sont positionnées sur le symbole le plus à gauche de chaque bande :

$$B_1 = \begin{array}{c|c|c} \hline {}^{\infty}\square & \omega & \square^{\infty} \\ \hline & \uparrow & \\ & T_1 & \end{array} \text{ et } B_2 = \begin{array}{c|c|c|c|c} \hline {}^{\infty}\square & (\mathbf{q}_i) & \$ & \delta & \square^{\infty} \\ \hline & \uparrow & & & \\ & T_2 & & & \end{array}$$

2. On commence par modifier la bande 1 pour qu'elle contienne la **configuration initiale de l'exécution** $(\epsilon, \mathbf{q}_i, \omega)$ et pas seulement le mot $\omega$. Pour cela on recopie au début de $B_1$ l'**état initial** de $M$, inscrit au début de la bande $B_2$.

$$B_1 = \begin{array}{c|c|c|c} \hline {}^{\infty}\square & (\mathbf{q}_i) & \omega & \square^{\infty} \\ \hline & \uparrow & & \\ & T_1 & & \end{array} \text{ et } B_2 = \begin{array}{c|c|c|c|c} \hline {}^{\infty}\square & (\mathbf{q}_i) & \$ & \delta & \square^{\infty} \\ \hline & & \uparrow & & \\ & & T_2 & & \end{array}$$

3. Considérons le cas général auquel on aboutit après exécution de plusieurs transitions.

$$B_1 = \begin{array}{c|c|c|c|c|c} \hline {}^{\infty}\square & \omega_L & (\mathbf{q}) & \ell & \omega_R & \square^{\infty} \\ \hline & & \uparrow & & & \\ & & T_1 & & & \end{array} \text{ et } B_2 = \begin{array}{c|c|c|c|c} \hline {}^{\infty}\square & (\mathbf{q}_i) & \$ & \delta & \square^{\infty} \\ \hline & & & \uparrow & \\ & & & T_2 & \end{array}$$

   On parcourt les transitions $\delta$ sur $B_2$ jusqu'à trouver la transition à exécuter c'est-à-dire celle qui commence par $\boxed{(\mathbf{q})\,|\,\ell}$.

4. Si aucune transition de $\delta$ ne correspond à $\boxed{(\mathbf{q})\,|\,\ell}$, alors la MT $U$ s'arrête : son résultat ($\mathbb{V}$, $\mathbb{F}$ ou $\omega'$) dépend de l'état $\mathbf{q}$ (accepteur, rejet, quelconque) et le mot résultat $\omega'$ correspond au contenu de la bande $B_1$ lorsqu'on a effacé $(\mathbf{q})$ ce qui fait passer de la configuration terminale au **mot résultat** .

5. Si on trouve dans la partie $\delta$ de $B_2$ une transition $(\mathbf{q}) \xrightarrow{\ell/e:d} (\mathbf{q}')$ alors on modifie la configuration sur $B_1$ :
   (a) on remplace $\ell$ de $B_1$ par $e$,
   (b) on remplace l'état courant par $(\mathbf{q}')$
   (c) on simule le déplacement $d$ de la tête $T_M$ en décalant l'état $(\mathbf{q}')$ d'un symbole vers la ............ pour $d =$ L et vers la .............. pour $d =$ R.

   Ensuite,
   (a) on déplace la tête $T_1$ sur l'état courant $(\mathbf{q}')$ et la tête $T_2$ sur le début de $\delta$
   (b) on poursuit l'exécution de $U$ en reprenant à l'étape 3

La machine de Turing universelle que l'on a décrit garantie les propriétés suivantes :

| | | | | |
|---|---|---|---|---|
| $U(m)(\omega) = \mathbb{V}$ | *ie.* | $U(m)(\omega) \to^* \circledcirc$ | *si et seulement si* | $M(\omega) \to^* \circledcirc$ |
| $U(m)(\omega) = \mathbb{F}$ | *ie.* | $U(m)(\omega) \to^* \bigcirc\!\not\to$ | *si et seulement si* | $M(\omega) \to^* \bigcirc\!\not\to$ |
| $U(m)(\omega) = \mathbb{E}\mathrm{rr}$ | *ie.* | $U(m)(\omega) \to^* \otimes$ | *si et seulement si* | $M(\omega) \to^* \otimes$ |
| $U(m)(\omega) = ?$ | *ie.* | $U(m)(\omega) \to^{\infty}$ | *si et seulement si* | $M(\omega) \to^{\infty}$ |
| $U(m)(\omega) = \omega'_1.\omega'_2$ | *ie.* | $U(m)(\omega) \to^* \omega'_1.\omega'_2$ *sur* $B_1$ | *si et seulement si* | $M(\omega) \to^* \omega'_1.\omega_2$ |

**Conclusion** La machine de Turing universelle $U$ est une MT à deux rubans, définie sur l'alphabet $\Sigma_U = \{\square, \$, (, \text{A}, \text{E}, \text{Q}, :, 0, 1), \text{L}, \text{H}, \text{R}\}$. On a vu au TD n° 2 qu'on pouvait se ramener à une MT classique à un seul ruban et on a vu au TD n° 1 qu'on pouvait se ramener à un alphabet réduit à $\{\boxed{0}, \boxed{1}\}$. On est donc capable au moyen de ces 2 transformations de construire un MT classique à un ruban travaillant sur l'alphabet $\{\boxed{0}, \boxed{1}\}$ qui réalise la machine de Turing universelle $U$. Par contre il est extrémement pénible d'appliquer ces transformations à la main. Les projets signalés dans les pages de ce cours visent à programmer ces transformations afin de pouvoir générer la version classique de la machine de Turing universelle.

## 4.5 Les projets MT

**(2015)** Représentation et exécution de MT classique sur $\Sigma = \{\Box, \$, 0, 1, \ldots\}$ cf. chapitre 1.

**(2015)** Représentation et exécution de MT à deux rubans, cf. TD n° 2.

**(2017)** Transformation d'une MT classique définie sur $\Sigma$ en une MT équivalente définie sur $\{\Box, \$, 0, 1\}$, cf. TD n° 1.

**(20 ? ?)** Transformation d'une MT à 2 rubans en MT classique, cf. TD n° 2.

**(20 ? ?)** Représentation d'une MT classique par un mot sur l'alphabet $\Sigma_U$, cf. § 4.4.1.

**(20 ? ?)** Réalisation de la machine de Turing universelle $U$ et transformation en une MT classique sur $\{\Box, \$, 0, 1\}$, cf. § 4.4.2.

# Chapitre 5

# Indécidabilité, premiers exemples

Le chapitre précédent nous a appris qu'**on pouvait ramener l'étude des fonctions calculables** à l'étude des **langages décidables**, c'est-à-dire des ensembles de mots (ou couples de mots) pour lesquels il existe des machines de Turing capables de décider si un mot appartient à l'ensemble ou non. Évidemment cela exige que l'exécution de la machine de Turing sur le mot termine.

Au chapitre précédent nous avons également conçu une **machine de Turing universelle** $U$ qui, à partir de la description binaire $m$ d'une MT $M$, permet d'exécuter $M$. La machine $U$ est un interpréteur de machines de Turing puisque

$$U(m)(\omega) = M(\omega) \;\; \textit{si} \;\; m = [M]_2$$

Nous allons dans ce chapitre exhiber **deux cas concrets de langages indécidables**, qui s'appuient sur la machine $U$. À chaque fois il s'agit d'un langage $L$ pour lequel il existe pourtant une MT $M$ qui répond $\mathbb{V}$, en un temps fini, si on lui donne un mot $\omega$ qui appartient à $L$. Un néophyte pourrait croire – à tort – que si on dispose d'une telle machine on sait alors tester si un mot appartient à $L$ : il suffit d'exécuter $M$ sur $\omega$ et observer si elle répond $\mathbb{V}$.

**L'effet de l'indécidabilité** apparaît lorsqu'on appelle $M$ avec un mot $\omega$ n'appartenant pas à $L$. Évidemment, lorsqu'on fait appel à la machine $M$ c'est qu'on ne sait pas si $\omega$ appartient à $L$, c'est justement ce qu'on veut tester. Si, pour certains mots n'appartenant pas à $L$, la MT $M$ peut répondre $\mathbb{F}$ dans un temps fini ; pour d'autres mots n'appartement pas à $L$, la MT $M$ peut ne pas terminer et c'est là le problème.

**Pourquoi la machine $M$ n'est pas utilisable en pratique ?** Supposons qu'on appelle $M(\omega)$ pour tester si $\omega \in L$ et que la réponse se fait attendre ... longtemps.

- Est-ce parce que $\omega$ appartient $L$ mais que le calcul pour le découvrir prend du temps ?
- Est-ce parce que $\omega$ n'appartient pas à $L$ et que la machine $M$ est partie dans un calcul qui ne finira jamais ?

Que faire dans ce cas ? Faut-il attendre 10 minutes de plus ou interrompre le calcul ? Vous avez tous fait l'expérience d'un ordinateur qui ne répond plus aux commandes, avec un ventilateur en marche et avec un processeur qui chauffe. Est-il en train de faire quelque chose d'important qu'il ne faut surtout pas interrompre sous peine de perdre des données ? ou bien, est-il parti dans un calcul inutile qu'il faut arrêter ? **L'indécidabilité vous met face au même dilemne.**

La suite du cours va établir une liste de problèmes apparemment simples qui se révèlent indécidables. Dans ce chapitre nous commençons par les deux premiers cas, qui donnent naissance aux autres.

Ce chapitre commence par introduire le vocabulaire utilisé dans les ouvrages classiques sur la décidabilité.

## 5.1 Indécidabilité : vocabulaire et définitions

**Rappel 1** *Soit $\Sigma$ un alphabet fini. $\Sigma^*$ est l'ensemble de tous les mots écrits sur l'alphabet $\Sigma$. C'est un ensemble énumérable. Un* ***langage*** *est un sous-ensemble de $\Sigma^*$. Dans ce chapitre on considère $\Sigma = \{0, 1\}$.*

**Définition 21 (Langage récursivement énumérable = reconnaissable par une MT)** *Un langage $L$ est* ***récursivement énumérable*** *si ................ une MT $M$ qui reconnaît $L$*

— *ie.* $\mathscr{L}(M) = L$

— *ie.* $M(\omega)$ *s'arrête sur un état accepteur si et seulement si* $\omega \in L$.

— *ie.* $M(\omega) = \mathbb{V} \iff w \in L$

**Remarque :** En revanche, le comportement de $M$ n'est pas certain pour les mots $\omega \notin L$ : $M$ peut ne pas ................. ou bien s'arrêter sur l'état rejet.

**Définition 22 (Langage co-récursivement énumérable)** *Un langage $L$ est* ***co-récursivement énumérable*** *si et seulement si son langage complémentaire $\overline{L}$ est récursivement énumérable, ie. $\overline{L}$ est ................................*

**Proposition 6 (Langage décidable)** *Un langage est* ***décidable*** *au sens de la définition 18*

$\Leftrightarrow$ *il est récursivement énumérable et co-récursivement énumérable*

$\Leftrightarrow$ ***le langage et son complémentaire*** *sont* ***reconnaissables****.*

**Preuve**

($\Rightarrow$) Ce sens de l'implication est facile à montrer. Il suffit d'appliquer les définitions.

($\Leftarrow$) $L$ est récursivement énumérable signifie qu'il existe une MT $M_1$ telle que $\mathscr{L}(M_1) = L$.
$L$ est co-récursivement énumérable signifie qu'il existe une MT $M_2$ telle que $\mathscr{L}(M_2) = \overline{L}$.
Notez que rien ne dit que $M_1 = M_2$. On doit supposer qu'il s'agit de machines de Turing différentes.
On doit montrer que $L$ est décidable, or la définition 18 dit : «Un langage $L$ est décidable s'il existe **une** MT qui termine pour tout mot de $\omega \in \{0, 1\}^*$ sur un état accepteur si $\omega \in L$ et sur un état rejet si $\omega \notin L$». On doit donc construire à partir de $M_1$ et $M_2$ une unique MT $M$ qui accepte $L$ et rejette $\overline{L}$.

**L'idée** On constuit la MT $M$ de la manière suivante : $M(\omega)$ copie le mot $\omega$ sur deux bandes $B_1$ et $B_2$ puis effectue alternativement une transition de $M_1$ sur $B_1$ et une transition de $M_2$ sur $B_2$ jusqu'à ce que l'une des deux machines s'arrêtent.

— Si $M_1$ s'arrête dans l'état $\circledcirc$ alors, par définition, $M_1(\omega) = \mathbb{V}$ *ie.* $\omega \in \mathscr{L}(M_1) = L$ donc $M$ passe dans un état $\circledcirc$

— Si $M_2$ s'arrête dans l'état $\circledcirc$ alors, par définition, $M_2(\omega) = \mathbb{V}$ *ie.* $\omega \in \mathscr{L}(M_2) = \overline{L}$ donc $M$ passe dans un état $\cancel{\circledcirc}$

**Ordonanceur** Pour construire $M$ on a besion d'un ordonanceur (*scheduler* en anglais), noté ${}_1||_1$, qui effectue alternativement une transition de chaque machine, cf. exercice 28.

$$M(\omega) \stackrel{\text{def}}{=} \left[ \begin{array}{l} B_1 := \omega \; ; \; B_2 := \omega \; ; \\ (m_1, B_1)\,{}_1||_1(m_2, B_2) \; ; \\ \text{if ( } \textit{état courant } m_1 = \circledcirc \text{ ) then } \rightarrow \circledcirc \text{ else } \rightarrow \cancel{\circledcirc} \end{array} \right]$$

□

**Exercice 28** *Ordonanceur*

Étant donné deux machines de Turing décrites par leur codage binaire $m_1$ et $m_2$, donnez une MT qui effectue alternativement une transition de $m_1$ sur $B_1$ et une transition de $m_2$ sur $B_2$ jusqu'à l'arrêt d'une des deux machines.

**Proposition 7 (Langage indécidable = non-décidable)** *Un langage $L$ est* ***indécidable***

$\Leftrightarrow \neg($ *L est reconnaissable .... $\overline{L}$ est ..............................)*

$\Leftrightarrow$ ***le langage** L* ***.....*** ***son complémentaire*** $\overline{L}$ *est* ***non-reconnaissable***

**Exercice 29**

La Proposition 7 est la négation de la Proposition 6. Complétez et donnez une interprétation, en français, de la dernière formulation logique.

$$\begin{array}{ll} \neg & \big(\exists M \in \text{MT}, \forall \omega \in \Sigma^*,\ M(\omega) = \mathbb{V} \Longleftrightarrow \omega \in L\big) \\ \equiv & ..M \in \text{MT},\ ..\omega \in \Sigma^*,\ \neg\big(\big(M(\omega) = \mathbb{V} \ldots . \omega \in L\big) \ldots\ldots \big(\omega \in L \Rightarrow M(\omega) = \mathbb{V}\big)\big) \\ \equiv & ..M \in \text{MT},\ ..\omega \in \Sigma^*,\ \neg\big(M(\omega) = \mathbb{V} \Rightarrow \omega \in L\big) \ldots\ldots \neg\big(\omega \in L \ldots . M(\omega) = \mathbb{V}\big) \\ \equiv & \forall M \in \text{MT},\ \exists \omega \in \Sigma^*,\ \big(M(\omega) = \mathbb{V}\ \wedge\ \omega \notin L\big)\ \bigvee\ \big(\omega \in L\ \wedge\ M(\omega) \neq \mathbb{V}\big) \end{array}$$

**Exercice 30**

Soit $\overline{L}$ un langage indécidable. Appliquez les définitions du cours afin de montrer que $L$ est indécidable.

## 5.2 L'ensemble des codages binaires des machines de Turing

Dans ce chapitre on considère l'ensemble $\mathcal{M} \times \{0,1\}^*$ des couples $(m,\omega)$ formés d'un mot binaire $\omega$ et du codage binaire $m$ d'une MT. C'est un ensemble énumérable, donc un langage et on peut alors s'intéresser à la décidabilité de ses sous-ensembles.

**Proposition 8 (L'ensemble $\mathcal{M}$ des codages binaires de machine de Turing est décidable)**

$$\mathcal{M} = \{m \in \{0,1\}^* \mid m = [M]_2,\ M \in \text{MT}\} \textit{ est } \textbf{\textit{un langage décidable}}.$$

**Preuve : Que doit-on montrer ?**

1. **que $\mathcal{M}$ est un langage, *ie.* un sous-ensemble d'un ensemble énumérable :** Il suffit de remarquer que les codages binaires des machines de Turing sont des mots de $\{0,1\}^*$ qui est énumérable.
2. **qu'il existe une** MT $M_{\mathcal{M}}$ **capable de décider si un mot** $\omega \in \{0,1\}^*$ **appartient ou non à $\mathcal{M}$, *ie.* décider si** $\omega$ **est bien le codage d'une** MT **:** On sait que tout langage régulier est reconnaissable par une MT (cf. exercice de traduction d'un automate (à nombre) d'états fini en MT). Il nous suffit donc de montrer que les codages binaires de MT forment un langage régulier.
   — Le codage binaire des états peut-être décrit par une expression régulière $\mathit{ExpReg}_{\textit{état}} \stackrel{\text{def}}{=} \texttt{"("} \cdot \{\text{A} \mid \text{R} \mid \text{Q}\} \cdot \texttt{":"} \cdot \{0,1\}^* \cdot \texttt{")"}$
   — Notre codage binaire des transitions $\mathbf{q} \xrightarrow{\ell/e:d} \mathbf{q}'$ est reconnaissable par l'expression régulière $\mathit{ExpReg}_{\tau} \stackrel{\text{def}}{=} \mathit{ExpReg}_{\textit{état}} \cdot \Sigma \cdot \Sigma \cdot \{\text{L,H,R}\} \cdot \mathit{ExpReg}_{\textit{état}}$
   — Le codage binaire d'une MT par « $\mathbf{q}_{init} \cdot \$ \cdot$ *listes des transitions* » est reconnaissable par l'expression régulière $\mathit{ExpReg}_{\mathcal{M}} \stackrel{\text{def}}{=} \mathit{ExpReg}_{\textit{état}} \cdot \$ \cdot \big(\mathit{ExpReg}_{\tau}\big)^*$, d'où $\mathcal{M} = \mathscr{L}(\mathit{ExpReg}_{\mathcal{M}})$
   — Toute expression régulière correspond à un AEF et tout AEF est réalisable par une MT donc il existe un AEF $A_{\mathcal{M}}$ équivalent à $\mathit{ExpReg}_{\mathcal{M}}$ et une MT $M_{\mathcal{M}}$ équivalente à $A_{\mathcal{M}}$

**Conclusion :** $\mathcal{M} = \mathscr{L}(\mathit{ExpReg}_{\mathcal{M}}) = \mathscr{L}(A_{\mathcal{M}}) = \mathscr{L}(M_{\mathcal{M}})$ est reconnaissable pour une MT $M_{\mathcal{M}}$ □

On donne maintenant deux exemples de langages indécidables : le langage universel et le langage des exécutions finies. Il en existe de nombreux autres dont l'indécidabilité est une conséquence de l'un ces deux cas.

## 5.3 Indécidabilité : premier exemple, preuve directe

**Proposition 9 (Le langage universel n'est pas décidable)** *Le* ***langage universel*** $L_U$ *est l'ensemble défini par*

$$L_U = \{(m,\omega) \in \mathcal{M} \times \{0,1\}^* \mid m = [M]_2,\ M(\omega) = \mathbb{V}\}$$

*C'est l'ensemble des couples* $(m,\omega)$ *tels que la* ................. *m accepte le* ........ $\omega$*.*

*(i)* $L_U$ *est* ................................................................, *ie. reconnaissable par une* MT

*(ii)* $L_U$ *n'est pas* **.... -récursivement énumérable**, *ie.* $\overline{L_U}$ *n'est pas reconnaissable par une* MT

*(iii)* $L_U$ *n'est* **pas décidable**.

**Preuve**

**(i)** On doit montrer qu'il existe une MT qui reconnaît $L_U$ : la MT cherchée c'est $U$. En effet, $\mathscr{L}(U) \stackrel{\text{def}}{=} \{\, (m,\omega) \mid \ \dots (m,\omega) = \mathbb{V} \}$ par définition du langage reconnu par une MT.

or $U(m,\omega) = U(m)(\omega) = M(\omega)$ avec $m = [M]_2$ par définition de la $U$
donc $\mathscr{L}(U) = \{\, (m,\omega) \mid M(\omega) = \dots,\ m = \dots\dots\dots \} \stackrel{\text{def}}{=} L_U$ d'après la définition de $L_U$.

**Conclusion :** $\mathscr{L}(U) = L_U$ ce qui signifie que la machine .................... $U$ reconnaît le langage ................. $L_U$.

**(ii)** On va montrer par .......................... qu'il n'existe pas de MT qui ....................... $\overline{L_U}$.

**Que représente** $\overline{L_U} \stackrel{\text{def}}{=} (\mathcal{M} \times \{0,1\}^*) \setminus L_U$ **?** Les éléments de $\overline{L_U}$ sont les couples $(m,\omega)$ que la machine universelle $U$ n'accepte pas.

$$\overline{L_U} = \{\, (m,\omega) \in \mathcal{M} \times \{0,1\}^* \mid U(m)(\omega) \neq \mathbb{V} \}$$

**Preuve de (ii) par contradiction :** Supposons qu'il ........... une MT $M_{\overline{L_U}}$ qui reconnaisse $\overline{L_U}$. On peut l'utiliser pour construire une MT $M_C(\omega) \stackrel{\text{def}}{=} M_{\overline{L_U}}(\omega,\omega)$ qui duplique le mot binaire $\omega$ pour en faire un couple et exécute $M_{\overline{L_U}}$ sur ce couple.

$$
\begin{array}{rcll}
\mathscr{L}(M_C) & = & \{\omega \mid M_C(\omega) = \mathbb{V}\} & \textit{par définition du langage } \dots\dots\dots\dots \textit{ par une } \text{MT} \\
 & = & \{\omega \mid M_{\overline{L_U}}(\omega,\omega) = \mathbb{V}\} & \textit{par définition de } M_C \\
 & = & \{\omega \mid (\omega,\omega) \in \overline{L_U}\} & \textit{par définition du langage reconnu par } \dots\dots\dots\dots \\
 & & & \textit{donc } \omega \textit{ n'est pas un mot quelconque de } \{0,1\}^* \textit{ mais un élément de } \mathcal{M} \\
 & = & \{m \in \ \dots\dots \mid (m,m) \in \overline{L_U}\} & \\
 & = & \{m \in \ \dots\dots \mid U(m)(m) \neq \mathbb{V}\} & \textit{par définition de } \overline{L_U} \\
\mathscr{L}(M_C) & = & \{m \in \ \dots\dots \mid m = [M]_2,\ M(m) \neq \mathbb{V}\} & \textit{par définition de la } \dots\dots \textit{ universelle}
\end{array}
$$

$\mathscr{L}(M_C)$ est donc l'ensemble des mots binaires de $\mathcal{M}$ qui correspondent à *des* MT *qui n'accepte pas, en tant que ........, leur ....................... binaire, ie.* $M(m) \to^* \otimes \ \vee \ M(m) \to^\infty$.

***Exhibons la contradiction :*** Considérons maintenant $m_c$ le codage binaire de la MT $M_C$ que l'on vient de .....................

**On peut alors se demander si $m_c$ appartient à $\mathscr{L}(M_C)$ ?**

$$
\begin{array}{rcl}
m_c \in \mathscr{L}(M_C) & \iff & m_c \in \{m \in \mathcal{M} \mid m = [M]_2,\ M(m) \neq \mathbb{V}\} \ \textit{par définition de } \mathscr{L}(M_C) \\
 & \iff & M_c(m_c) \neq \mathbb{V} \ \ \textit{puisque} \ \ m_c = [M_c]_2
\end{array}
$$

Ainsi,

$(\dagger) \quad m_c \in \mathscr{L}(M_C) \iff M_C(m_c) \neq \mathbb{V}$

Par ailleurs,

$(\ddagger) \quad m_c \in \mathscr{L}(M_C) \iff M_C(m_c) = \mathbb{V}$ *par définition du langage* ................. *par une* MT

Les équivalences $(\dagger)$ et $(\ddagger)$ donnent la CONTRADICTION cherchée.

***Conclusion :*** En supposant qu'il existait une MT qui reconnaît $\overline{L_U}$ nous aboutissons à une contradiction. Donc $\overline{L_U}$ est indécidable, ce qui termine la preuve de *(ii)*.

**(iii)** D'après la propostion 6 un langage $L$ est décidable si et seulement si $L$ et $\overline{L}$ sont reconnu par une MT. $\overline{L_U}$ n'étant pas reconnaissable par une MT, cf. *(ii)*. $L_U$ n'est pas décidable.

□

**Exercice 31** ***Preuve de l'indécidabilité de*** $L_{EF}$

En suivant exactement le même schéma de preuve que pour le langage universel, complétez la preuve que $L_{EF}$ est reconnaissable par un MT et que $\overline{L_{EF}}$ n'est pas reconnaissable par une MT.

## 5.4 Indécidabilité : second exemple, preuve directe

**Proposition 10 (Le langage des exécutions finies n'est pas décidable)** *Le* ***langage des exécutions finies*** $L_{EF}$ *est l'ensemble défini par*

$$L_{EF} = \{ (m,\omega) \in \mathcal{M} \times \{0,1\}^* \mid U(m)(\omega) \not\to \infty \}$$

*C'est l'ensemble des ................ $(m,\omega)$ tels que l'........................ de la machine $m$ termine quand on ............................ le ........ $\omega$.*

*(i)* $L_{EF}$ *est* ***récursivement énumérable****, ie. reconnaissable ..........................*

*(ii)* $L_{EF}$ *n'est* ***pas co-récursivement énumérable****, ie. .......... n'est pas reconnaissable.*

**Preuve**

**(i) Montrons $L_{EF}$ reconnaissable :** Montrons qu'il existe une MT $M_{EF}$ qui reconnaît $L_{EF}$, *ie.* $\mathscr{L}(M_{EF})$ = ........., *ie.* $M_{EF}(m,\omega)$ = .... $\iff (m,\omega) \in L_{EF}$, *ie.* $M_{EF}(m,\omega)$ = .... $\iff$ $U(m)(\omega) \not\to$ .....

La MT $M_{EF}$ doit s'arrêter dans un état accepteur pour tout couple $(m,\omega)$ de $L_{EF}$, c'est-à-dire pour les couples qui correspondent à des ..................... finies. $M_{EF}$ consiste à exécuter $U(m)(\omega)$ – le résultat nous importe peu – puis à passer dans l'état accepteur ◎. Puisque les couples de $L_{EF}$ sont précisement les couples pour lesquels l'exécution de $U$ .............., on a la garantie que la MT $M_{EF}$ ci-dessous ................... dans l'état ◎ pour les couples de ..........

$$M_{EF}(m,\omega) \stackrel{\text{def}}{=} [\ U(m)(\omega)\,;\to\circledcirc\ ]$$

**(ii) Montrons $\overline{L_{EF}}$ non-reconnaissable :** On va montrer par .......................... qu'il ....................... de MT qui ....................... $\overline{L_{EF}}$.

**Que représente $\overline{L_{EF}} \stackrel{\text{def}}{=} (\mathcal{M} \times \{0,1\}^*) \setminus L_{EF}$ ?** Les éléments de $\overline{L_{EF}}$ sont les couples $(m,\omega)$ sur lesquels que la machine universelle $U$ ne .............. pas.

$$\overline{L_{EF}} \stackrel{\text{def}}{=} (\mathcal{M} \times \{0,1\}^*) \setminus L_{EF} = \{ (m,\omega) \in \mathcal{M} \times \{0,1\}^* \mid U(m)(\omega) .......... \}$$

**Preuve de (ii) par contradiction :** Supposons qu' .............. une MT $M_{\overline{EF}}$ qui ....................... $\overline{L_{EF}}$. On peut l'utiliser pour .................... une MT $M_C(\omega) \stackrel{\text{def}}{=} M_{\overline{EF}}(\omega,\omega)$ qui ................ le mot binaire $\omega$ pour en faire un couple et .............. $M_{\overline{EF}}$ sur ce couple.

$$\begin{array}{rcll}
\mathscr{L}(M_C) & = & \{\omega \mid .................................\} & \textit{par définition du langage reconnu par une MT} \\
& = & \{\omega \mid M_{\overline{EF}}(\omega,\omega) = \mathbb{V}\} & \textit{par définition de } M_C \\
& = & \{\omega \mid ........... \in \overline{L_{EF}}\} & \textit{par définition du langage reconnu par une MT} \\
& & & \textit{donc } \omega \textit{ n'est pas un mot quelconque de } \{0,1\}^* \textit{ mais un élément de } \mathcal{M} \\
& = & \{m \in \mathcal{M} \mid (m,m) \in .........\} & \\
& = & \{m \in \mathcal{M} \mid U(m)(m) ..........\} & \textit{par définition de } \overline{L_{EF}} \\
\mathscr{L}(M_C) & = & \{m \in \mathcal{M} \mid m = ........., \ M(m) \to \infty\} & \textit{par définition de la} \\
& & & ................................
\end{array}$$

$\mathscr{L}(M_C)$ est donc l'ensemble des mots binaires de $\mathcal{M}$ qui correspondent à *des* MT *qui ne s'arrête pas lorsqu'on les exécute sur leur ...................... binaire.*

***Exhibons la contradiction :*** Considérons maintenant $m_c$ le codage binaire de la MT $M_C$ que l'on vient de construire. **On peut alors se demander si $m_c$ appartient à $\mathscr{L}(M_C)$ ?**

$$\begin{array}{rcl} m_c \in \mathscr{L}(M_C) & \iff & m_c \in \{ m \in \mathcal{M} \mid m = [M]_2,\ M(m) \to \infty \} \textit{ par définition de } \mathscr{L}(M_C) \\ & \iff & \dots\dots\dots\dots\dots\dots \to \infty \ \textit{ puisque } \ m_c = [M_c]_2 \end{array}$$

Ainsi, (†) $m_c \in \mathscr{L}(M_C) \iff M_C(m_c) \to \infty$

Par ailleurs, par définition du langage ................ par une MT, on a aussi l'équivalence :

$$(\ddagger) \quad m_c \in \mathscr{L}(M_C) \iff M_C(m_c) = \mathbb{V} \iff M_C(m_c) \to^* \circledcirc$$

Les équivalences (†) et (‡) donnent la CONTRADICTION cherchée puisque l'exécution $M_C(m_c)$ est censée terminer (dans l'état $\circledcirc$) d'après (‡), et ne pas terminer d'après (‡).

***Conclusion :*** En supposant qu'il existait une MT qui reconnaît $\overline{L_{EF}}$ nous aboutissons à une contradiction. Donc $\overline{L_{EF}}$ est indécidable, ce qui termine la preuve de $(ii)$.

□

## 5.5 D'où vient l'indécidabilité ?

Certrains pensent que l'indécidabilité apparaît du fait que les MT peuvent avoir des exécutions infinies : ils ont tort. Si vous relisez la preuve précédente vous verrez que l'existence d'exécution infinie ne joue aucun rôle. L'argument principal qui conduit à l'indécidabilité est la possibilité qu'une MT $M$ prenne en argument son propre code $m$. On retrouve un argument de diagonalisation à la Cantor puisque parmi les couples $(m, \omega)$ on s'intéresse au couple $(m, m)$, auxquelles on applique $U$ pour obtenir $U(m)(m) = M(m)$.

Cette capacité à s'auto-analyser, s'auto-modifier, s'auto-répliquer apporte à l'informatique la capacité de s'auto-vérifier, de s'adapater, de créer des virus, *etc*. La contre-partie de cette puissance est que la plupart des questions non-triviales qu'on se pose sur les programmes *(terminent-ils ? sont-ils bogués ?)* ne sont pas décidables ... par des programmes.

# Chapitre 6

# Principe de réduction & applications

## 6.1 Principe de réduction

Le **principe de réduction** permet de **relier la décidabilité / l'indécidabilité de deux langages**, disons $L$ et $L'$ au moyen d'une traduction qui réduit la question d'appartenance à $L$ à la question de l'appartenance à $L'$. **Prenez garde au sens des implications de la proposition 12.**

**Définition 23 (Diagramme de réduction)** *Soit $L$ un langage sur l'alphabet $\Sigma$ et $L'$ un langage sur l'alphabet $\Sigma'$. Un **diagramme de réduction de $L$ à $L'$** est un **schéma** comme suit, **accompagné***

$$\begin{array}{ccc} \omega \in \Sigma^* & \xrightarrow[\text{traduction}]{M_R} & R(\omega) = \omega' \in \Sigma'^* \\ \omega \in L & \underset{(\ddagger)}{\Longleftrightarrow} & R(\omega) \in L' \end{array}$$

1. *d'une* MT *$M_R$ qui termine **pour tout mot** $\omega \in \Sigma^*$ et traduit $\omega$ en un mot de $\Sigma'^*$.*
   *On note $R(\omega)$ le résultat de $M_R(\omega)$. La traduction doit être bien choisie de sorte qu'on ait l'équivalence ($\ddagger$) qui relie l'appartenance d'un mot à $L$ à l'appartenance de sa traduction à $L'$*
2. *et de la **preuve de l'équivalence** ($\ddagger$)*

**Proposition 11 (Diagramme de réduction complémentaire)** *Le même diagramme de réduction est valable pour les complémentaires de $L$ et $L'$.*

$$\begin{array}{ccc} \omega \in \Sigma^* & \xrightarrow[\text{traduction}]{M_R} & R(\omega) = \omega' \in \Sigma'^* \\ \omega \in \overline{L} & \underset{(\ddagger)}{\Longleftrightarrow} & R(\omega) \in \overline{L'} \end{array}$$

**Preuve** Il suffit de montrer que l'équivalence ($\ddagger$) est valide pour les complémentaires de $L$ et de $L'$.

$$\begin{array}{rccc} & \omega \in L & \underset{(\ddagger)}{\Longleftrightarrow} & R(\omega) \in L' \\ (\ddagger) \textit{ est une équivalence donc} & \neg(\ldots\ldots\ldots\ldots) & \Longleftrightarrow & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ & \equiv \omega \notin L & & \equiv \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ & \equiv \ldots\ldots\ldots\ldots & \underset{(\ddagger)}{\Longleftrightarrow} & \ldots\ldots\ldots\ldots\ldots \overline{L'} \end{array}$$

□

**Proposition 12** *Un diagramme de réduction comme celui de la définition 23 permet de conclure :*

| | | | | |
|---|---|---|---|---|
| *(i)* | *$L$ reconnaissable* | $\Longleftarrow$ | *$L'$ reconnaissable* | |
| *(ii)* | *$L$ décidable* | $\Longleftarrow$ | *$L'$ décidable* | |
| *(iii)* | *$L$ non-reconnaissable* | …….. | *$L'$ non-reconnaissable* | *: c'est la contraposé de (i)* |
| *(iv)* | $L$ …………………… | $\Longrightarrow$ | $L'$ …………………… | *: c'est la contraposé de (ii)* |

*(v)* (i), (ii), (iii), (iv) *sont valables en remplaçant $L$ et $L'$ par leurs complémentaires $\overline{L}$ et $\overline{L'}$*

**Preuve**

$(i)$ **Le principe de réduction repose sur la construction suivante :** Supposons $L'$ reconnaissable par une MT $M'$ et utilisons $M'$ pour construire une MT $M$ qui reconnaît $L$.

L'hypothèse « $M'$ reconnaît $L'$ » signifie $M'(\omega') = \mathbb{V} \underset{(1)}{\iff} \omega' \in L'$.

Prenons $M \stackrel{\text{def}}{=} [M_R\,;M']$. **On doit montrer** $M(\omega) = \mathbb{V} \stackrel{?}{\iff} \omega \in L$.

$$\begin{array}{ccc}
M(\omega) = \mathbb{V} & \stackrel{\text{OK}}{\iff} & w \in L \\
\Updownarrow \text{ def} & & \Uparrow \\
\overbrace{\hphantom{\omega \xrightarrow{\quad\quad} R(\omega) \longrightarrow \circledcirc}} & & \text{d'après } \ldots\ldots \\
\omega \xrightarrow[\quad\quad]{\ldots\ldots\ldots} \underbrace{R(\omega) \xrightarrow{\ldots\ldots} \circledcirc}_{M'(R(\omega))=\ldots.} & \underset{(1)}{\iff} & \Downarrow \\
 & & R(\omega) \in \ \ldots.
\end{array}$$

$(ii)$ **On doit montrer** ... reconnaissable ... ... reconnaissable. L'hypothèse « $L'$ décidable » signifie $L'$ ............................ et .... reconnaissable. D'après $(i)$ $L'$ reconnaissable implique $L$ reconnaissable. Il reste à montrer $\overline{L}$ reconnaissable : il suffit d'appliquer $(i)$ au diagramme des complémentaires de la proposition 11, pour obtenir « .................................. implique .................................. ». Mais alors $L$ est ................. puisque $L$ et $\overline{L}$ sont reconnaissables.

$(iii)$ et $(iv)$ s'obtiennent comme contraposé de $(i)$ et $(ii)$ : $B \Longleftarrow A \equiv$ *par contraposé* $\equiv \neg B \Longrightarrow \neg A$

> **Preuve par contradiction** Montrons que de $A \Longrightarrow B$ et de $\neg B$ on peut déduire $\neg A$. Supposons $A$. En utilisant l'implication $A \Longrightarrow B$, on peut en déduire $B$. Ce qui contredit l'hypothèse $\neg B$. Conclusion : En supposant $A$ on aboutit à une contradiction, ce qui prouve $\neg A$. □

$(iii)$ peut aussi se démontrer directement par contradiction

> Supposons $L'$ reconnaissable (c'est $A$) alors on peut construire $M \stackrel{\text{def}}{=} [M_r\,;M']$ qui reconnaît $L$ comme au $(i)$, donc $L$ serait reconnaissable (c'est $B$). Or, $(iii)$ fait l'hypothèse que $L$ est non-reconnaissable (c'est $\neg B$), on aboutit donc à une contradiction. Conclusion : $L'$ est non-reconnaissable (c'est $\neg A$).

$(iv)$ peut aussi se démontrer directement : **On doit montrer** $L'$ indécidable *ie.* $L'$ ..... $\overline{L'}$ non-reconnaissable. L'hypothèse « $L$ indécidable » signifie que $L$ .... $\overline{L}$ est non-reconnaissable. Examinons les deux cas :

(a) Si c'est $L$ qui est non-reconnaissable alors $L'$ est non-reconnaissable d'après ........ .

(b) Si c'est $\overline{L}$ qui est non-reconnaissable : il suffit d'appliquer ........ au diagramme des ................................. de la proposition 11, pour obtenir « $\overline{L}$ non-reconnaissable implique $\overline{L'}$ non-reconnaissable ».

Dans les deux cas, $L'$ est indécidable puisque soit $L'$, soit $\overline{L'}$ est non-reconnaissable.

$(v)$ D'après la proposition 11 le diagramme de réduction est valide pour les complémentaires de $L$ et $L'$. Les résultats précédents $(i)$, $(ii)$, $(iii)$, $(iv)$ qui reposent sur le diagramme de réduction sont donc applicables à $\overline{L}$ et $\overline{L'}$.

□

## 6.2 Applications du principe de réduction

### 6.2.1 Réduction de $L_{EF}$ à $L_{TT}$

**Proposition 13 (L'ensemble des machines de Turing qui Terminent Toujours est indécidable)** *Le langage des machines de Turing qui terminent toujours $L_{TT}$ est défini par*

$$L_{TT} = \{m' \in \mathcal{M} \mid \forall \omega' \in \{0,1\}^*,\ U(m',\omega') \not\to \infty\}$$

*C'est l'ensemble des .................................. de* MT *dont les exécutions .................... .......................... le mot binaire en entrée. Le langage $L_{TT}$ est ...................... puisque son complémentaire $\overline{L_{TT}}$ n'est pas reconnaissable.*

**Remarques**

1. $\overline{L_{TT}}$ est l'ensemble des codages binaires $m$ de MT tels qu'il existe au moins une entrée sur laquelle l'exécution de $m$ ne termine pas.

$$\overline{L_{TT}} = \mathcal{M} \setminus L_{TT} = \{m' \in \mathcal{M} \mid \exists \omega' \in \{0,1\}^*,\ U(m',\omega') \to \infty\}$$

2. On va montrer qu'**on peut réduire la question** de l'appartenance à $\overline{L_{EF}}$ à celle de l'appartenance à $\overline{L_{TT}}$. On pourrait alors conclure, d'après le $(iv)$ de la proposition 12, que $L_{TT}$ est indécidable en s'appuyant sur le fait connu que $L_{EF}$ est indécidable. Nous allons être plus précis et montrer pourquoi $L_{TT}$ est indécidable : la raison est que $\overline{L_{TT}}$ est non-reconnaissable.

3. L'ensemble $\overline{L_{EF}}$ des exécutions infinies est l'ensemble des couples $(m,\omega)$ tels que l'exécution de la MT $m$ sur le mot $w$ ne termine pas.

$$\overline{L_{EF}} = (\mathcal{M} \times \{0,1\}^*) \setminus L_{EF} = \{(m,\omega) \in \mathcal{M} \times \{0,1\}^* \mid U(m,\omega) \to \infty\}$$

**Preuve de la proposition 13 – montrons que $\overline{L_{TT}}$ est non-reconnaissable**

$\overline{L_{EF}}$ est un ensemble de couples $(m,\omega) \in \mathcal{M} \times \{0,1\}^*$ et $\overline{L_{TT}}$ est un ensemble de codages binaires de MT $m \in \mathcal{M}$. Un **diagramme de réduction de $\overline{L_{EF}}$ à $\overline{L_{TT}}$** est donc de la forme

$$\begin{array}{rcl} (m,\omega) \in \mathcal{M} \times \Sigma^* & \xrightarrow[traduction]{M_R} & R(m,\omega) = m' \in \mathcal{M} \\ (m,\omega) \in \overline{L_{EF}} & \underset{(\ddagger)}{\Longleftrightarrow} & R(m,\omega) \in \overline{L_{TT}} \end{array}$$

**Que doit-on faire ?**

1. **On doit définir une** MT $M_R$ qui traduit en un temps fini tout couple $(m,\omega)$ en une MT $m' \stackrel{\text{def}}{=} R(m,\omega)$.
2. **On doit montrer** que l'exécution de $M_R$ termine pour tout couple $(m,\omega)$ de $\mathcal{M} \times \{0,1\}^*$.
3. **On doit ensuite démontrer l'équivalence** $(\ddagger)$ c'est-à-dire que l'exécution $U(m,\omega)$ ne termine pas si et seulement si la MT $R(m,\omega)$ ne termine pas pour une certaine entrée.

**Allons-y**

1. Étant donné $(m,\omega)$ avec $\omega = s_1.s_2.\ldots.s_n$, **la MT $M_R$ constuit la réduction** $m' \stackrel{\text{def}}{=} R(m,\omega)$ de la façon suivante :

$$m' = M_R(m,\omega) \stackrel{\text{def}}{=} \underbrace{[\ M_{\mathit{eff}}\,;\mathbf{q}_0 \xrightarrow{\square/s_1:R} \mathbf{q}_1 \xrightarrow{\square/s_2:R} \ldots \xrightarrow{\square/s_n:R} \mathbf{q}_n\,; M_{\overleftarrow{\$}}\ ]_2\,; m}_{\textit{codage binaire d'une MT}}$$

À partir de $m$ et $\omega$, la $M_R$ produit un codage binaire $m'$ d'une MT qui .......... la donnée inscrite sur le ruban, puis ........... le mot $\omega = s_1.s_2.\ldots.s_n$ symbole par symbole à la place, puis .................. sur le symbole \$ de début de ruban, et enfin ......... la machine $m$ sur ce ruban (donc sur le mot $\omega$). La machine $m'$ ainsi produite ignore la donnée d'entrée inscrite sur le ruban et exécute en fait $U(m,\omega)$.

2. **La MT $M_R$ termine pour tout couple** $(m,\omega) \in \mathcal{M} \times \{0,1\}^*$ : en effet, elle laisse $m$ intacte sur le ruban ; elle copie le codage binaire de $[\ M_{\mathit{eff}}\,;\mathbf{q}_0 \xrightarrow{\square/s_1:R} \mathbf{q}_1 \xrightarrow{\square/s_2:R} \ldots \xrightarrow{\square/s_n:R} \mathbf{q}_n\,; M_{\overleftarrow{\$}}\ ]$ avant $m$ or la taille de ce codage binaire dépend uniquement de la taille de $\omega$ qui est un mot fini.

3. **Démontrons l'équivalence** (‡) Examinons l'exécution de $R(m,w)$ sur un mot d'entrée $\omega' = \ell_1.\dots.\ell_k$ à $k$ symboles :

$$\underset{\uparrow}{\$}.\underbrace{\ell_1.\dots.\ell_k}_{\textit{entrée}} \xrightarrow{M_{\textit{eff}}} \underset{\uparrow}{\$}.\square \xrightarrow{\square/s_1:R} \xrightarrow{\square/s_2:R} \dots \xrightarrow{\square/s_n:R} \underbrace{s_1.\dots.s_n}_{\omega}.\underset{\uparrow}{\square} \xrightarrow{M_{\overleftarrow{\$}}} \underset{\uparrow}{\$}.\omega \xrightarrow{U(m)} \dots$$

Puisque le mot d'entrée est fini (de taille $k$), la MT $M_{\textit{eff}}$ termine (en $2k$ étapes) ; les $n$ transitions qui inscrivent $\omega$ sur le ruban terminent en $n$ étapes ; la MT $M_{\overleftarrow{\$}}$ retrouve le \$ de début de ruban en $n+1$ étapes. Toutes ces instructions terminent et **l'exécution ne peut éviter la dernière instruction qui consiste à exécuter $U(m)$ sur le mot $\omega$.**

Finalement, **exécuter** $m' \overset{\text{def}}{=} R(m,\omega)$ **sur le mot** $\omega'$ **revient à exécuter** $U(m,\omega)$**.** Utilisons cette observation pour démontrer (‡) :

($\Rightarrow$) $(m,\omega) \in \overline{L_{EF}}$ signifie que l'exécution $U(m,\omega)$ .............................., donc il existe

........................... entrée (en fait .......... ) sur laquelle $m' \overset{\text{def}}{=} R(m,\omega)$ ..............................

et donc $R(m,\omega) \in \overline{L_{TT}}$.

($\Leftarrow$) $R(m,\omega) \in \overline{L_{TT}}$ signifie que il existe au moins une entrée $\omega'$ sur laquelle $m' \overset{\text{def}}{=} R(m,\omega)$ ne termine pas. Or, .......................... $\omega'$ l'exécution de $m'$ ........ inévitablement

......................... $\$MU(m,\omega)$. Donc, c'est $U(m,\omega)$ la cause de la non-terminaison

et donc $(m,\omega) \in \overline{L_{EF}}$.

□

### 6.2.2 Réduction de $L_{EF}$ à PCP

En TD vous verrez un autre exemple de réduction de $L_{EF}$ vers un probléme très différent « l'existence d'une solution au Problème des Correspondances de Post (PCP)».

**Remarques**

1. Dans les exemples de langages indécidables traités jusqu'à présent on a pris soin de bien identifier la raison de l'indécidabilité : est-ce $L$ qui est non-reconnaissable ? ou bien $\overline{L}$ ? ou bien les deux ?
2. **En pratique** on se moque de savoir lequel de $L$ ou $\overline{L}$ est non-reconnaissable puisqu'il faut que le **problème soit décidable pour pouvoir donner un algorithme qui termine toujours et réponde** $\mathbb{V}$ **ou** $\mathbb{F}$.
3. Un **algorithme** qui répond $\mathbb{V}$ quand l'entrée $\omega \in L$ mais qui **peut boucler** quand l'entrée $\omega \in \overline{L}$ est intéressant du point de vue théorique mais peu utile en pratique. En effet, lorsqu'on lui donne un mot $\omega$ et qu'il n'a toujours pas répondu au bout d'1 min... de 10 min... d'1h... de 10h... que conclure ? Peut-être que $\omega \in L$ mais que cette vérification demande beaucoup de calcul et qu'il faut attendre encore un peu pour avoir la réponse, ou bien la MT est en train de boucler ; on n'a aucun moyen de le savoir.
4. Dans la suite nous nous contentons donc d'indiquer si le problème est décidable ou indécidable, sans préciser en cas d'indécidabilité si c'est le langage ou son complémentaire qui est non-reconnaissable.

# Chapitre 7

# Théorème de Rice & applications

## 7.1 Théorème de Rice

### 7.1.1 Version classique du théorème de Rice

**Définition 24 (Langages reconnaissables)** *Étant donné un alphabet $\Sigma$. On note $\mathcal{L}$ l'ensemble des langages reconnaissables par les machines de Turing :*

$$\mathcal{L} = \{\mathscr{L}(M) \mid M \in \textsc{mt}\}$$

**Théorème 2 (de Rice, version langage)** *Soit $\mathcal{P}_\mathcal{C}$ un sous-ensemble non-trivial de $\mathcal{L}$, défini comme l'ensemble des langages reconnaissables qui satisfont une **condition non-triviale** $\mathcal{C}$, ie.*

$$\mathcal{P}_\mathcal{C} = \{L \in \mathcal{L} \ \mid \ \mathcal{C}(L)\}$$

*alors **l'appartenance d'un langage à $\mathcal{P}_\mathcal{C}$ est indécidable, ie. $\mathcal{P}_\mathcal{C}$ ou $\overline{\mathcal{P}_\mathcal{C}}$ est non-reconnaissable**.*

**Remarques**

1. La condition $\mathcal{C}$ est un prédicat sur les langages, *ie.* $\mathcal{C} : \mathcal{L} \rightarrow Bool$.
2. Une condition est triviale si elle ne dépend pas de son paramètre. Il n'y a donc que deux conditions $\mathcal{C}$ triviales : celle qui vaut toujours $\mathbb{V}$, dans ce cas $\mathcal{P}_\mathcal{C} = \mathcal{L}$ et celle qui vaut toujours faux, dans ce cas $\mathcal{P}_\mathcal{C} = \emptyset$.
3. la condition $\mathcal{C}$ est non-triviale $\Leftrightarrow$ l'ensemble $\mathcal{P}_\mathcal{C}$ est non-trivial,
   *ie.* $\mathcal{C} \neq \mathbb{V}$ et $\mathcal{C} \neq \mathbb{F}$ *ie.* $\mathcal{P}_\mathcal{C} \neq \mathcal{L}$ et $\mathcal{P}_\mathcal{C} \neq \emptyset$.

Avant de donner la preuve de ce théorème, pour en comprendre la portée et les conséquences nous allons le reformuler en terme de machines de Turing.

### 7.1.2 Version machine de Turing du théorème de Rice

Au lieu de considérer $\mathcal{L}$, l'ensemble des langages reconnaissables, on peut énoncer le théorème de Rice en considérant $\mathcal{M}$, l'ensemble des codes de machines de Turing. C'est cette version que nous utiliserons dans les exemples et que nous allons démontrer.

**Rappel** $m$ est le codage binaire de la MT $M$ signifie $m = [M]_2$ et $U(m) = M$ et $U(m)(\omega) = M(\omega)$

**Définition 25 (Ensemble de Rice)** *Un ensemble de Rice, noté $\mathcal{P}_\mathcal{C}$ est un sous-ensemble de $\mathcal{M}$, défini comme l'ensemble des codes $m$ de* MT *qui satisfont une **condition non-triviale** $\mathcal{C} : \mathcal{L} \rightarrow Bool$ **sur le langage reconnu par** $m$, ie. un $\mathcal{P}_\mathcal{C}$ est de la forme*

$$\{m \in \mathcal{M} \ \mid \ \mathcal{C}(\mathscr{L}(U(m)))\}$$

**Remarque :** $\mathscr{L}(U(m))$ désigne le langage reconnu par la MT de code $m$.

$$\begin{aligned}\text{En effet, } \mathscr{L}(U(m)) &= \{\omega \in \Sigma^* \mid U(m)(\omega) = \mathbb{V}\} \\ &= \{\omega \mid M(\omega) = \mathbb{V}\} \textit{ puisque } U(m)(\omega) = M(\omega) \textit{ où } m = [M]_2 \\ &= \mathscr{L}(M)\end{aligned}$$

**Théorème 3 (de Rice, version** MT**)** *Soit $\mathcal{P}_\mathcal{C}$ un ensemble de Rice défini par une* ***condition non-triviale*** *$\mathcal{C} : \mathcal{L} \to Bool$* ***sur le langage reconnu par un code*** *$m$* ***de machine de Turing****, ie.*

$$\mathcal{P}_\mathcal{C} = \{ m \in \mathcal{M} \ \mid \ \mathcal{C}(\mathscr{L}(U(m))\}$$

*Alors* ***l'appartenance d'une*** *MT $m$* ***à $\mathcal{P}_\mathcal{C}$ est indécidable, ie. $\mathcal{P}_\mathcal{C}$ ou $\overline{\mathcal{P}_\mathcal{C}}$ est non-reconnaissable.***

### 7.1.3 Applications du théorème de Rice

1. $\mathcal{P}_1 = \{m \mid U(m)(1) = \mathbb{V}\}$ correspond à la condition $\mathcal{C} : \mathcal{L} \to Bool$ définie par $\mathcal{C}(L) \stackrel{\text{def}}{=} 1 \in L$. D'après le Théorème de Rice, $\mathcal{P}_1$ est indécidable ; en fait, $\overline{\mathcal{P}_1}$ est non-reconnaissable. En français cela donne : L'ensemble des machines de Turing qui acceptent le mot binaire 1 est indécidable.
2. On peut généraliser l'exemple précédent à n'importe quel mot $\omega$. L'ensemble des MT $\mathcal{P}_\omega = \{m \mid U(m)(\omega) = \mathbb{V}\}$ correspond à la condition $\mathcal{C}(L) \stackrel{\text{def}}{=} \omega \in L$. D'après le Théorème de Rice, $\mathcal{P}_\omega$ est indécidable ; en fait, $\overline{\mathcal{P}_\omega}$ est non-reconnaissable. En français cela donne : Étant donné un mot $\omega$, l'ensemble $\mathcal{P}_\omega$ des MT qui acceptent le mot binaire $\omega$ est indécidable.
3. $\mathcal{P}_{reg} = \{m \mid \exists Aut \in \text{AEF},\ \mathscr{L}(Aut) = \mathscr{L}(U(m))\}$ correspond à la condition $\mathcal{C}(L) \stackrel{\text{def}}{=} \exists Aut \in \text{AEF},\ \mathscr{L}(A) = L$. D'après le Théorème de Rice, $\mathcal{P}_{reg}$ est indécidable. En français cela donne : L'ensemble des machines de Turing régulières, c'est-à-dire les MT dont le langage est régulier, est indécidable.
4. $\mathcal{P}_\emptyset = \{m \mid \mathscr{L}(U(m)) = \emptyset\}$ correspond à la condition $\mathcal{C}(L) \stackrel{\text{def}}{=} L = \emptyset$. D'après le Théorème de Rice, $\mathcal{P}_\emptyset$ est indécidable. En français cela donne : L'ensemble des machines de Turing qui n'acceptent aucun mot est non-reconnaissable.
5. $\mathcal{P}_{fini} = \{m \mid |\mathscr{L}(U(m))| \in \mathbb{N}\}$ correspond à la condition $\mathcal{C}(L) \stackrel{\text{def}}{=} |L| \in \mathbb{N}$ D'après le Théorème de Rice, $\mathcal{P}_{fini}$ est indécidable. En français cela donne : L'ensemble des machines de Turing dont le langage est fini, est indécidable.

### 7.1.4 Mais alors que reste-t'il de décidable ?

Les propriétés décidables sur $\mathcal{M}$ sont celles qui ne s'intéressent pas seulement sur le résultat, la terminaison ou le langage reconnu mais qui porte aussi sur la structure des MT (ou la longueur de l'exécution).

**Exemples :**

- $\{m \mid U(m)(\omega) \xrightarrow{\leq 1000} \omega'\}$ l'ensemble des MT dont l'exécution sur $\omega$ termine en moins de 1000 pas de calcul.
- L'ensemble des MT qui ne contiennent pas de boucle.
- L'ensemble des MT dont les boucles sont bornées par des constantes, *ie.* for i = $N..M$ avec $N$ et $M$ constant.

**Remarques**

1. Puisque les MT correspondent aux fonctions calculables, elles représentent tous les algorithmes, indépendamment du langage de programmation choisi. **Les raisonnements sur les limitations des MT s'appliquent donc aux algorithmes.**
2. **La plupart des propriétés intéressantes sont indécidables.** La source du problème est qu'il n'existe pas d'algorithme capable de décider pour tout programme qu'on lui donnerait en entrée s'il termine ou non.
3. Toutefois, en restreignant la forme des programmes qu'on prend en entrée, on est capable de décider certaines classes de programmes. On parvient ainsi à mieux cerner la frontière entre décidable et indécidable. On sait par exemple quelles restrictions imposer sur les langages de programmation afin

(a) d'éliminer les fuites de mémoire et les segmentation fault – grâce au ramasse-miettes (*garbage collector* en anglais) ;

(b) d'éliminer les erreurs de type ou d'écrasement mémoire – grâce aux typages forts comme en CAML ou ADA ;

(c) de garantir la terminaison des programmes par un typage encore plus fort que celui de CAML

Malgré cela les programmeurs continuent de programmer dans des langages qui n'offrent aucune garantie et ils se retrouvent à devoir passer 60% de leur temps à tester et à debogguer.

4. D'autre part, on utilise en pratique des algorithmes semi-décidables (qui peuvent ne pas terminer) en leur ajoutant un minuteur qui interrompt l'exécution au delà d'un certain délai. Si l'algorithme termine avant le délai imparti il donne une réponse ; si l'algorithme est interrompu il conclut par "je ne sais pas".

### 7.1.5 Théorème de Rice et MT équivalentes

**Définition 26 (Équivalence de machines de Turing)** *Deux* MT $M_1$ *et* $M_2$ *sont équivalentes si et seulement si* $\mathscr{L}(M_1) = \mathscr{L}(M_2)$ *ie. si elles reconnaissent le même langage.*

**Remarques**

1. L'équivalence portent sur **les résultats** des exécutions de $M_1$ et $M_2$ et pas sur leur code ; ce qu'on pourrait énoncer de manière caricaturale par «***en calculabilité****, peu importe le programme, du moment qu'il donne le résultat souhaité*».
2. Au contraire, ***en complexité****, pour un même résultat final, on s'intéresse au code le plus efficace*. Dans ce cas on cherchera à comparer les codes.

**Rappel** $m$ est le codage binaire de la MT $M$ signifie $m = [M]_2$ et $U(m) = M$ et $U(m)(\omega) = M(\omega)$

**Proposition 14 (Ensemble de Rice et MT équivalentes)** *Étant donné un ensemble de Rice* $\mathcal{P_C}$*, par définition* $\mathcal{P_C} \stackrel{def}{=} \{m \in \mathcal{M} \mid \mathcal{C}(\mathscr{L}(U(m)))\}$ *où la condition* $\mathcal{C}$ *porte sur le langage reconnu par* $m$

***si*** *des codages binaires* $m$ **et** $m'$ ***représentent des*** MT ***équivalentes****,*
*ie.* $\mathscr{L}(U(m)) = \mathscr{L}(U(m'))$
***alors*** $m \in \mathcal{P_C} \Longrightarrow m' \in \mathcal{P_C}$

**Preuve**

$$m \in \mathcal{P_C} \Longrightarrow \mathcal{C}(\underbrace{\mathscr{L}(U(m))}_{=\mathscr{L}(U(m'))}) \Longrightarrow \mathcal{C}(\mathscr{L}(U(m'))) \Longrightarrow m' \in \mathcal{P_C}$$

Autrement dit, $m \in \mathcal{P_C}$ signifie $\mathcal{C}(\mathscr{L}(U(m))) = \mathbb{V}$ par définition de $\mathcal{P_C}$. Prenons maintenant une MT $m'$ équivalente à $m$, *ie.* $\mathscr{L}(U(m')) = \mathscr{L}(U(m))$. Forcément $\mathcal{C}(\mathscr{L}(U(m'))) = \mathcal{C}(\mathscr{L}(U(m))) = \mathbb{V}$ et donc $m' \in \mathcal{P_C}$. □

**Exercice 32** ***Indécidabilité de l'équivalence de machines de Turing***

Le théorème de Rice permet de montrer qu'il n'existe pas de MT (*ie.* de programme) capable de dire si deux MTs (ou programmes) fournis en paramètre sont équivalents (c'est-à-dire qu'ils retournent le même résultat sur toutes les entrées possibles).

**Q1. Complétez (répondez sur votre copie).** Deux MT $M_1$ et $M_2$ sont équivalentes si et seulement si ..................................................................

**Q2.** Étant donnée une MT $M'$, montrez, à l'aide du théorème de Rice, que l'ensemble $\mathcal{P} \stackrel{def}{=} \{m \mid U(m) \equiv M'\}$ des machines de Turing équivalentes à $M'$ est indécidable.

### 7.1.6 Preuve du théorème de Rice

Soit $\mathcal{P_C}$ un ensemble non-trivial de Rice. Montrons que $\mathcal{P_C}$ est indécidable par réduction à $\mathcal{P_C}$ d'un langage connu pour être ..........................

**Remarques préliminaires**

1. Soit $\mathcal{P_C}$ un ensemble non-trivial de Rice. $\mathcal{P_C}$ non-trivial signifie que $\mathcal{P_C} \neq \mathcal{M}$ et $\mathcal{P_C} \neq \emptyset$. On en déduit (1) $\exists m_1 \notin \mathcal{P_C}$ et (2) $\exists m_2 \in \mathcal{P_C}$ . En effet, si (1) est faux alors $\mathcal{P_C} = \mathcal{M}$, ce qui contredit $\mathcal{P_C}$ *non-trivial* et si (2) est faux alors $\mathcal{P_C} = \emptyset$, ce qui contredit aussi $\mathcal{P_C}$ *non-trivial* .
2. La MT $M_\emptyset \overset{\text{def}}{=}\!\!\rightarrow$  reconnaît le langage vide
3. La preuve comporte une subtilité qui nous oblige à distinguer deux cas :

   **(cas 1)** $M_\emptyset \notin \mathcal{P_C}$ : on montre $\mathcal{P_C}$ indécidable en réduisant la question $\overset{?}{\in} L_{EF}$ à $\overset{?}{\in} \mathcal{P_C}$

   **(cas 2)** $M_\emptyset \in \mathcal{P_C}$ : on se ramène au (cas 1) en considérant l'indécidabilité de $\overline{\mathcal{P_C}}$ qui alors ne contient pas $M_\emptyset$ puisque $m_\emptyset \in \mathcal{P_C}$.

#### Première partie de la preuve (cas 1)

**Preuve du théorème de Rice (cas 1) en supposant** $M_\emptyset \notin \mathcal{P_C}$ On prouve l'indécidabilité de $\mathcal{P_C}$ par **réduction de** $L_{EF}$ **à** $\mathcal{P_C}$. Rappelons que $L_{EF}$ est l'ensemble des couples (machine,mot) tels que l'exécution de la machine sur ce mot est finie :

$$L_{EF} = \{\, (m,\omega) \in \mathcal{M} \times \{0,1\}^* \mid U(m)(\omega) \not\rightarrow \infty \}$$

Considérons le diagramme de réduction

$$\begin{array}{ccc} (m,\omega) \in \mathcal{M} \times \{0,1\}^* & \xrightarrow[\textit{traduction}]{M_R} & R(m,\omega) \in \mathcal{M} \\ \underbrace{(m,\omega) \overset{?}{\in} L_{EF}} & \underset{(\ddagger)}{\Longleftrightarrow} & \underbrace{R(m,\omega) \overset{?}{\in} \mathcal{P_C}} \\ \textit{indécidable} & \Longrightarrow & \ldots\ldots\ldots\ldots\ldots\ldots\ldots \end{array}$$

**Que doit-on faire ?**

(a) On doit donner une MT qui traduit tout couple $(m,\omega)$ de $\mathcal{M} \times \{0,1\}^*$ en une MT, notée $R(m,\omega)$

(b) On doit montrer l'équivalence ($\ddagger$).

**Allons-y !**

(a) **Définition de la traduction :** Puisque $\mathcal{P_C}$ est non-trivial, $\mathcal{P_C} \neq \emptyset$ donc il existe (au moins) une MT $m_1$ qui appartient à $\mathcal{P_C}$ – cf. remarque préliminaire 1. On utilise $m_1$ pour définir $R(m,\omega)$ :

$$\underbrace{\boxed{R(m,\omega)}}_{\text{MT}}(\omega_1) \overset{\text{def}}{=} [\ \text{if } U(m)(\omega) \text{ then } U(m_1)(\omega_1) \text{ else } U(m_1)(\omega_1)\ ]$$

$R(m,\omega)$ est une MT à 4 bandes $B_1 = \omega_1,\ B_2 = m_1,\ B_3 = \omega,\ B_4 = m$. Par défaut le mot d'entrée $\omega_1$ est inscrit sur la bande $B_1$. La MT $R(m,\omega)$ fait appel à la machine universelle $U$ pour exécuter $m$ sur $\omega$ ; puis (*si cette exécution termine*) elle exécute $m_1$ sur le mot d'entrée $\omega_1$, toujours grâce à $U$.

(b) **Montrons l'équivalence** ($\ddagger$)

($\Rightarrow$) **On doit montrer que si** $\boxed{(m,\omega) \in L_{EF}}$ **alors** $\boxed{R(m,\omega) \in \mathcal{P_C}}$ **:**

$\boxed{(m,\omega) \in L_{EF}}$ signifie que l'exécution $U(m)(\omega)$ termine mais alors l'exécution de $R(m,w)$ sur le mot d'entrée $\omega_1$ se comporte comme celle de $U(m_1)(\omega_1)$ ; sauf qu'elle est précédée par un calcul $U(m)(\omega)$ qui termine et dont on ignore le résultat. On en déduit que $\mathscr{L}(R(m,\omega)) = \mathscr{L}(U(m_1))$. Autrement dit le langage de $R(m,\omega)$ est le même que celui de $m_1$ et puisque $m_1 \in \mathcal{P_C}$ la proposition 14 permet de conclure que $\boxed{R(m,\omega) \in \mathcal{P_C}}$.

($\Leftarrow$) **On doit montrer que si** $R(m,\omega) \in \mathcal{P_C}$ **alors** $(m,\omega) \in L_{EF}$ **:**

Il est équivalent[1] de **montrer la contraposé : si** $\boxed{(m,\omega) \notin L_{EF}}$ **alors** $\boxed{R(m,\omega) \notin \mathcal{P_C}}$.

$\boxed{(m,\omega) \notin L_{EF}}$ signifie $U(m)(\omega) \to \infty$ *ie.* l'exécution de $U(m)(\omega)$ ne termine pas, mais alors – quel que soit le mot $\omega_1$ – l'exécution de $R(m,\omega)$ sur $\omega_1$ ne termine pas non plus puisqu'elle attend indéfiniment le résultat de l'évaluation de la condition du if $U(m)(\omega)$... Ne terminant pas la MT $R(m,\omega)$ n'accepte pas le mot $\omega_1$. Notez que ce raisonnement est valide quel que soit le mot $\omega_1$ et donc $\mathscr{L}(R(m,\omega)) = \emptyset$.

Mais alors $R(m,\omega) \equiv M_\emptyset$ puisque $\mathscr{L}(R(m,\omega)) = \emptyset = \mathscr{L}(M_\emptyset)$. Et comme on a supposé $M_\emptyset \notin \mathcal{P_C}$ (cas 1) alors $\boxed{R(m,\omega) \notin \mathcal{P_C}}$ d'après la proposition 14.

□

**Seconde partie de la preuve (cas 2)** La preuve exploite le résultat de l'exercice 30 page 35 qu'on rappelle ici.

$\overline{L}$ indécidable $\iff$ $L$ indécidable

**Preuve du théorème de Rice, suite et fin, en supposant** $M_\emptyset \in \mathcal{P_C}$ **(cas 2)** D'après l'exercice 30 il suffit de montrer que $\overline{\mathcal{P_C}}$ est indécidable pour conclure que $\mathcal{P_C}$ est indécidable.

1. D'après la remarque préliminaire 1. On sait qu'il existe une MT $m_2 \notin \mathcal{P_C}$ et donc $\boxed{m_2 \in \overline{\mathcal{P_C}}}$
2. De même, puisque la MT $M_\emptyset$ appartient à $\mathcal{P_C}$ $(cas\ 2)$, elle n'appartient pas à son complémentaire. Donc $\boxed{M_\emptyset \notin \overline{\mathcal{P_C}}}$
3. On peut alors montrer l'indécidabilité de $\overline{\mathcal{P_C}}$ par réduction de $L_{EF}$ à $\overline{\mathcal{P_C}}$ en recopiant la preuve du (cas 1) dans laquelle il suffit de remplacer $\mathcal{P_C}$ par $\overline{\mathcal{P_C}}$ et $m_1 \in \mathcal{P_C}$ par $m_2 \in \overline{\mathcal{P_C}}$.

□

1. $A \Rightarrow B \equiv \neg A \Rightarrow \neg B$