# Thèse de Church-Turing

**Exercice 1.** *Reconnaître, décider, calculer*

1. Montrer que si un algorithme $A$ *décide* un langage $L$, alors $A$ le *reconnaît* également.
2. Soit $\mathcal{M}_c$ une machine de Turing qui *calcule* une fonction totale $f : \Sigma^* \to \{0,1\}$. Montrer qu'on peut construire une machine de Turing $\mathcal{M}_d$ qui *décide* $L_f = \{w \in \Sigma^* : f(w) = 1\}$.
3. On applique la même construction qu'à la question précédente pour une machine $\mathcal{M}_c$ qui calcule une fonction *partielle* $f : \Sigma^* \to \{0,1\}$. Que fait la machine $\mathcal{M}_d$ obtenue ?
4. Inversement, soit $\mathcal{M}_d$ une machine de Turing qui *décide* un langage $L$. Montrer qu'on peut construire une machine $\mathcal{M}_c$ qui *calcule* la fonction caractéristique $\chi_L$ de $L$.

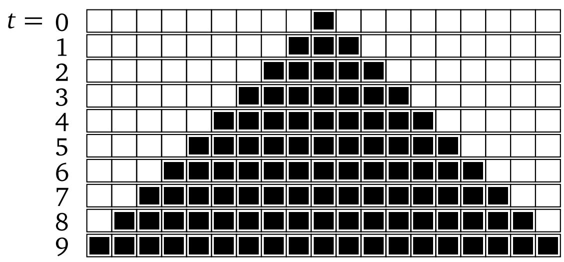
**Exercice 2.** *Machines RAM*

On considère des machines RAM de taille de mot $n$.

1. Écrire le code d'une machine RAM qui étant donné $x$ et $y$ calcule $x - y \bmod 2^n$.
2. Écrire le code d'une machine RAM qui étant donné $x$ et $y$ calcule $x \times y \bmod 2^n$.
3. On s'intéresse à des machines dont les entrées et sorties sont dans $\{0,1\}$ : quelque soit la taille de mot $n$, on suppose que les valeurs des entrées sont dans $\{0,1\}$ (sans avoir à le vérifier).
   - **i.** Écrire le code d'une machine RAM qui étant $x \in \{0,1\}$ calcule $\neg x$.
   - **ii.** Écrire le code d'une machine RAM qui étant donné $x, y \in \{0,1\}$ calcule $x \wedge y$.
   - **iii.** Écrire le code d'une machine RAM qui étant donné $x, y \in \{0,1\}$ calcule $x \vee y$.
   - **iv.** En déduire toute fonction booléenne $f : \{0,1\}^k \to \{0,1\}$ peut être calculée par une machine RAM.

**Exercice 3.** *Automates cellulaire unidimensionnels*

Un automate cellulaire (unidimensionnel) consiste en un ruban constitué d'une infinité de cases, chacune possédant un symbole d'un alphabet $\Sigma$, et d'une fonction de transition locale $f : \Sigma^3 \to \Sigma$ qui indique comment chaque cellule doit évoluer à chaque étape en fonction de son symbole et des symboles de ses deux voisines. Formellement, les cases du ruban sont indexées par les entiers relatifs : $R = (r_i)_{i\in\mathbb{Z}}$. Initialement, le ruban est dans la configuration initiale $R^{(0)} = (r_i^{(0)})_{i\in\mathbb{Z}}$. Une étape de calcul fait passer de la configuration $R^{(t)}$ à $R^{(t+1)}$ définie par $r_i^{(t+1)} = f(r_{i-1}^{(t)}, r_i^{(t)}, r_{i+1}^{(t)})$ pour tout $i$. On suppose que la fonction de transition est totale.
**Exemple.** On définit l'automate sur l'alphabet $\Sigma = \{0,1\}$ par $f(0,0,0) = 0$ et $f(a,b,c) = 1$ si $(a,b,c) \neq (0,0,0)$. On dessine ci-dessous les configurations $R^{(0)}$ à $R^{(9)}$ où $r_0^{(0)} = 1$ et $r_i^{(0)} = 0$ si $i \neq 0$. On représente $r_i^{(t)} = 0$ par une case blanche et $r_i^{(t)} = 1$ par une case noire.




1. On s'intéresse à des automates où $\Sigma = \{0,1\}$ et on utilise la même représentation que dans l'exemple. Pour chacune des tables de transitions, représenter les configurations $R^{(0)}$ à $R^{(9)}$ où $r_0^{(0)} = 1$ et $r_i^{(0)} = 0$ so $i \neq 0$.
   - **i.** $f(1,0,0) = f(0,0,1) = 1$ et $f(a,b,c) = 0$ sinon ;
   - **ii.** $f(0,1,0) = f(1,0,0) = f(0,1,1) = 1$ et $f(a,b,c) = 0$ sinon ;
   - **iii.** $f(1,0,0) = f(0,1,1) = f(0,1,0) = f(0,0,1) = 1$ et $f(a,b,c) = 0$ sinon.

Notre objectif est de démontrer l'équivalence entre le modèle des automates cellulaires unidimensionnels et les machines de Turing. Pour cela, on va faire deux hypothèses sur les automates cellulaires : d'une part, il existe un symbole spécial « blanc » $\square$ (le 0 dans nos exemples) et la configuration initiale ne contient qu'un nombre fini de symboles $\neq \square$ (l'entrée) ; d'autre part, $f(\square, \square, \square) = \square$.

**2.** Justifier l'intérêt de ces hypothèses.

**3.** On veut simuler une machine $\mathcal{M} = (Q, \Sigma, q_0, \delta)$. L'idée est que l'alphabet de l'automate cellulaire contienne non seulement $\Sigma$, mais également, pour chaque état $q \in Q$ et symbole $x \in \Sigma$, un symbole $x_q$.

  **i.** Décrire formellement l'alphabet de l'automate cellulaire.

  **ii.** Considérons une transition $\delta(q, x) = (q', y, \longleftrightarrow)$ de la machine de Turing. Traduire cette transition en des règles $f(x_q, \cdot, \cdot)$, $f(\cdot, x_q, \cdot)$ et $f(\cdot, \cdot, x_q)$, en fonction du type de déplacement.

  **iii.** Comment doit-on définir $f(x, y, z)$ si $x,\ y,\ z \in \Sigma$ ?

**4.** La fonction de transition locale d'un automate cellulaire est totale, ce qui implique que l'automate ne termine jamais. La table de transition de la machine n'est pas totale, donc la question précédente laisse encore des transitions à compléter.

  **i.** Supposons que $\mathcal{M}$ calcule une fonction $f$. Proposer une façon de compléter la fonction $f$ pour la rendre totale, et une convention pour définir le résultat du calcul de l'automate cellulaire.

  **ii.** Supposons que $\mathcal{M}$ reconnaît un langage $L$. Proposer une façon de compléter la fonction $f$ pour la rendre totale, et une convention pour affirmer qu'un mot en entrée de l'automate cellulaire appartient ou non à $L$.

**5.** Expliquer comment simuler un automate cellulaire par une machine de Turing, toujours avec les deux hypothèses ci-dessus.