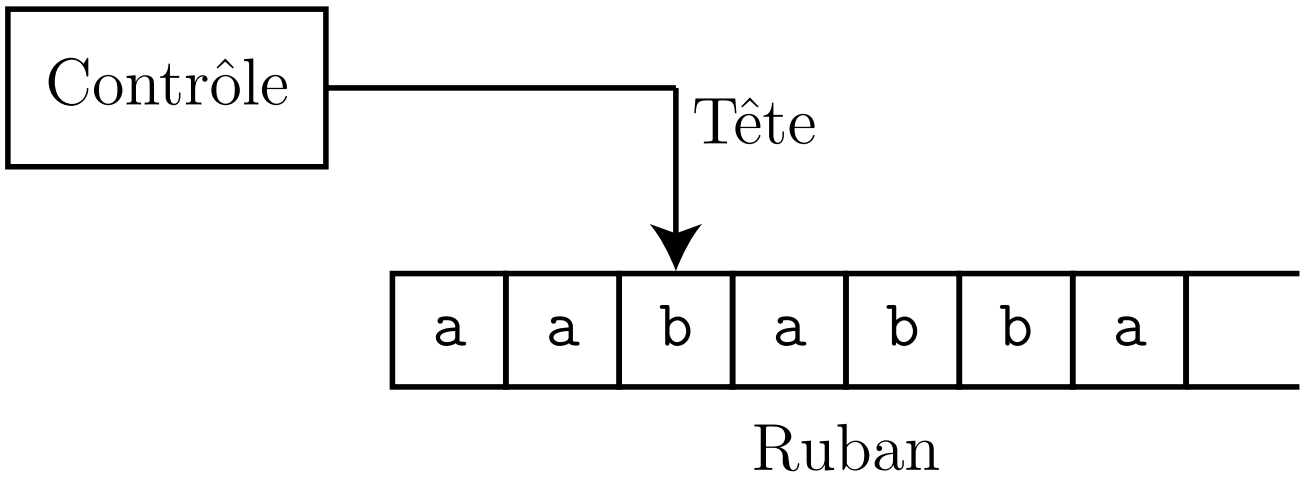
Chapitre 3

Thèse de Church-Turing

Les machines de Turing

Figure 3.1.



- Une machine de Turing peut lire le contenu du *ruban* et écrire sur celui-ci.
- Le *contrôle* passe d'un *état* à un autre à chaque étape du calcul ; il y a un nombre fini d'états possibles.
- À chaque étape de calcul, la *tête* de lecture et d'écriture peut bouger à gauche ou bouger à droite. Elle ne peut rester immobile que si elle est positionnée sur la première case à gauche.
- Le ruban est illimité vers la droite.
- Si le contrôle passe à l'état *final*, alors le calcul est terminé.

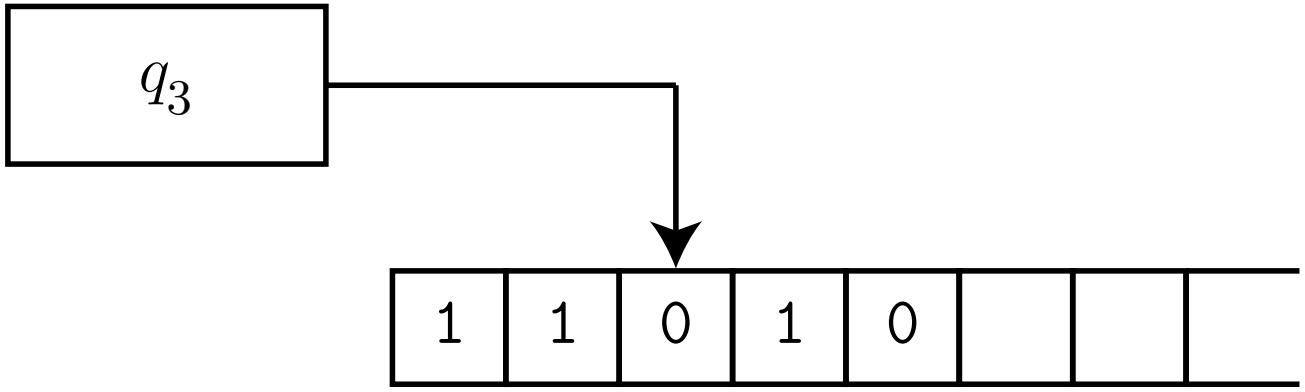
Définition 3.1. Une machine de Turing (MT) est un 6-tuplet $(Q, \Sigma, \Gamma, \delta, q_0, q_F)$ où :

- Q est un ensemble fini d'états ;
- Σ est l'alphabet ;
- Γ est l'alphabet de ruban, $\sqcup \in \Gamma$ et $\Sigma \subseteq \Gamma$;
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\langle \textsc{Gauche} \rangle, \langle \textsc{Droite} \rangle\}$ est la fonction de transition ;
- q_0 est l'état initial ;
- q_F est l'état final.

▲

Définition 3.2. Une configuration de la MT $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$ est un triplet (u, q, v) avec $q \in Q$ et $u, v \in \Gamma^*$, ce qui s'interprète en disant que le ruban de M contient uv, suivi d'une chaîne infinie de $\sqcup$, et la tête est au dessus du premier symbole de v. ▲

Figure 3.2.



La configuration $11q_3010$.

Remarque 3.3. Même si le ruban d'une MT est mathématiquement infini, nous dirons que le *contenu du ruban* est la chaîne finie $v \in \Gamma^*$, si le ruban est la concaténation de v et d'une chaîne infinie de $\sqcup$, et si le dernier caractère de v n'est pas $\sqcup$.

De la même façon, nous parlerons de l'*extrémité droite* du ruban, ou du *dernier caractère* du ruban, etc. ▲

Définitions 3.4. Pour $q, q' \in Q$, $u, v \in \Gamma^*$ et $x, x', z \in \Gamma$, on dit que la MT $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$

passe de la configuration $c = (uz, q, xv)$ à la configuration $c' = (u, q', zx'v)$

ou

de la configuration $c = (\varepsilon, q, xv)$ à la configuration $c' = (\varepsilon, q', x'v)$

si $\delta(q, x) = (q', x', \langle \textsc{Gauche} \rangle)$.

La machine M

passe de la configuration $c = (u, q, xv)$ à la configuration $c' = (ux', q', v)$

si $\delta(q, x) = (q', x', \langle \textsc{Droite} \rangle)$.

On dit qu'il s'agit d'une transition de c à c'.

Une telle transition est notée $c \vdash c'$.

On écrit $c \overset{*}{\vdash} c'$ si M passe de c à c' en 0, 1 ou plusieurs transitions successives.

▲

Définition 3.5. Pour $w \in \Sigma^*$, $w' \in \Gamma^*$ et $u \in \Gamma^*$, on dit que la MT $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$ retourne ou s'arrête sur le mot w' sur entrée w si

$$(\varepsilon, q_0, w) \overset{*}{\vdash} (u, q_F, w').$$

Si M n'atteint jamais son état final q_F à partir de la configuration (ε, q_0, w), alors on dit que M boucle sur entrée w. ▲

Définition 3.6. Soient une fonction

$$f \ : \ \Sigma^* \to \Sigma^*$$

et une MT

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_F).$$

On dit que f est calculée par M si pour tout $w \in \Sigma^*$ on a :

$$(\varepsilon, q_0, w) \overset{*}{\vdash} (u, q_F, f(w)) \quad \text{pour un certain } u \in \Gamma^*.$$

On dit aussi que M calcule ou implante la fonction f. ▲

Descriptions des MT

Les nombreux exemples de MT laissés en exercice donnent un aperçu des tâches qu'elles peuvent accomplir.

D'autre part, la plupart de ces tâches, même les plus simples, sont exécutées par des MT dont les définitions formelles sont abstruses et difficilement compréhensibles.

À partir de maintenant, nous décrirons les MT en langage naturel. Les descriptions seront schématiques, mais suffisamment claires et détaillées pour qu'on puisse sans difficulté donner la définition formelle correspondante, ou se convaincre qu'une telle définition existe bel et bien.

L'exemple suivant décrit une MT qui ajoute 1 à un nombre naturel écrit en binaire donné en entrée.

Exemple 3.7.

Prendre un nombre naturel en binaire en entrée;
insérer le caractère sentinelle # en première position;
parcourir le ruban de gauche à droite jusqu'au dernier chiffre;
si ce chiffre est `0`, alors le remplacer par `1`;
sinon, remplacer tous les `1` consécutifs par des `0`,
    de droite à gauche, jusqu'au premier `0` rencontré,
    et remplacer ce `0` par `1`;
    si on atteint la sentinelle avant de rencontrer un `0`,
        alors écrire `1` sur la première case;
supprimer la sentinelle si nécessaire.

▲

Les MT peuvent simuler les programmes TANTQUE.

Théorème 3.8. Soit

$$f \ : \ \mathbb{N} \to \mathbb{N}$$

une fonction calculée par un programme TANTQUE. Alors il existe une MT M qui calcule une fonction

$$f' \ : \ \{0,1\}^* \to \{0,1\}^*$$

telle que $f(x) = y$ implique $f'(x') = y'$, où x' et y' sont les représentations binaires de x et y respectivement.

Aperçu de la preuve. Pour que M puisse simuler un programme `TANTQUE`, elle doit pouvoir travailler avec des registres contenant des nombres entiers, c'est-à-dire entre autres :

- incrémenter la valeur d'un registre ;
- déterminer si deux registres sont égaux ou non.

La première de ces tâches est illustrée à l'exemple 3.7, et la deuxième est laissée en exercice.

Les registres seront présents sur le ruban de M sous forme binaire et séparés par le caractère $\#$.

La machine M commence par insérer les caractères `0#` au début du ruban, ce qui crée le registre r_0 et l'initialise à 0.

Les registres de travail $r_2, \ldots, r_l$ sont créés et initialisés en ajoutant `#0` pour chacun d'eux à la fin du ruban.

Les branchements nécessaires à l'exécution du programme `TANTQUE` sont implantés à l'aide des états de M, c'est-à-dire que différents points dans le programme correspondent à différents états de M.

Lorsque la simulation est terminée, M efface les registres de travail, ramène la tête de lecture en première position, et passe à son état final. ■

Remarque 3.9. Le théorème 3.8 peut être facilement généralisé aux cas où la fonction f peut prendre la valeur $\uparrow$, c'est-à-dire les cas où le programme peut boucler à l'infini, de même qu'aux cas où f est une fonction de plusieurs variables entières. ▲

Les programmes `TANTQUE` peuvent simuler les MT.

Théorème 3.10. Soit

$$f \ : \ \Sigma^* \to \Sigma^*$$

une fonction calculée par une MT $M = (Q, \Sigma, \Gamma, \delta, q_0, q_F)$. Alors il existe un programme `TANTQUE` qui calcule une fonction

$$f' \ : \ \mathbb{N} \to \mathbb{N}$$

telle que $f(x) = y$ implique $f'(x') = y'$, où les entiers x' et y' sont les représentations dans un codage de Gödel des chaînes x et y respectivement.

Aperçu de la preuve. On peut voir le ruban de M comme un tableau infini de caractères dans Γ de la forme

$$(c_1, c_2, \ldots, c_n, \sqcup, \sqcup, \ldots).$$

Si on réserve l'entier 0 pour le caractère $\sqcup$, le ruban peut être représenté par un tableau infini d'entiers de la forme

$$(a_1, a_2, \ldots, a_n, 0, 0, \ldots),$$

ou, grâce au codage de Gödel, par l'entier

$$p_1^{a_1} p_2^{a_2} \ldots p_n^{a_n}.$$

Le registre r_1 va donc contenir le ruban de M sous cette forme.

Le registre r_2 contient la position de la tête de lecture.

Le registre r_3 contient l'état de M, encodé dans un entier.

Chaque transition de M peut être facilement simulée, dans les registres r_1, r_2 et r_3, à l'aide entre autres des programmes `TABLVAL` et `TABLASS` vus au chapitre précédent.

Lorsque la simulation est terminée, la portion du ruban de M qui contient $f(x)$ est copiée dans le registre r_0, toujours en codage de Gödel. ■

Remarque 3.11. Le théorème 3.10 peut être facilement généralisé aux cas où M peut boucler à l'infini. ▲

Remarque 3.12. Les théorèmes 3.8 et 3.10 nous permettent de décrire des MT comme des programmes `TANTQUE`, si cela est utile. ▲

Les booléens

Définition 3.13. Une valeur booléenne ou un booléen est un élément de l'ensemble $\{\langle \text{VRAI} \rangle, \langle \text{FAUX} \rangle\}$.

On utilise aussi la convention que $\langle \text{VRAI} \rangle = 1$ et $\langle \text{FAUX} \rangle = 0$, ce qui permet de représenter une valeur booléenne par un bit. ▲

Définition 3.14. Pour un booléen x, $\neg x$, dit non x, ou la négation de x, est un booléen qui est $\langle \text{VRAI} \rangle$ si, et seulement si x est $\langle \text{FAUX} \rangle$. ▲

Définition 3.15. Pour x et y deux booléens, $x \wedge y$, dit x et y, ou la conjonction de x et y, ou le produit booléen de x et y, est un booléen qui est $\langle \text{VRAI} \rangle$ si, et seulement si $x = \langle \text{VRAI} \rangle$ et $y = \langle \text{VRAI} \rangle$. ▲

Définition 3.16. Pour x et y deux booléens, $x \vee y$, dit x ou y, ou la disjonction de x et y, ou la somme booléenne de x et y, est un booléen qui est $\langle \text{VRAI} \rangle$ si, et seulement si $x = \langle \text{VRAI} \rangle$ ou $y = \langle \text{VRAI} \rangle$. ▲

Expressions booléennes

Définition 3.17. Une expression booléenne est une expresion contenant des variables, des valeurs booléennes et les opérateurs $\neg$, $\wedge$ et $\vee$. ▲

Exemple 3.18.

$$(x_1 \vee x_2) \wedge ((\neg x_1 \vee (x_3 \wedge x_2)) \wedge \neg x_3)$$

est une expression booléenne contenant les variables x_1, x_2 et x_3.

Si $x_1 = 1$, $x_2 = 1$ et $x_3 = 1$, alors la valeur de l'expression est

$$\begin{aligned}
(1 \vee 1) \wedge ((\neg 1 \vee (1 \wedge 1)) \wedge \neg 1) &= 1 \wedge ((\neg 1 \vee (1 \wedge 1)) \wedge \neg 1) \\
&= 1 \wedge ((0 \vee (1 \wedge 1)) \wedge \neg 1) \\
&= 1 \wedge ((0 \vee 1) \wedge \neg 1) \\
&= 1 \wedge (1 \wedge \neg 1) \\
&= 1 \wedge (1 \wedge 0) \\
&= 1 \wedge 0 \\
&= 0
\end{aligned}$$

▲

Circuits booléens

Définition 3.19. Un circuit booléen est composé de variables d'input $x_1, \ldots, x_n$, de variables d'output $y_1, \ldots, y_m$, et de variables de travail $t_1, \ldots, t_k$.

Le circuit est un liste ordonnée d'équations $z = E$, où z est une variable de travail ou une variable d'output, et où E est une expression booléenne composée de variables d'input, de variables de travail, de variables d'output, et de constantes booléennes.

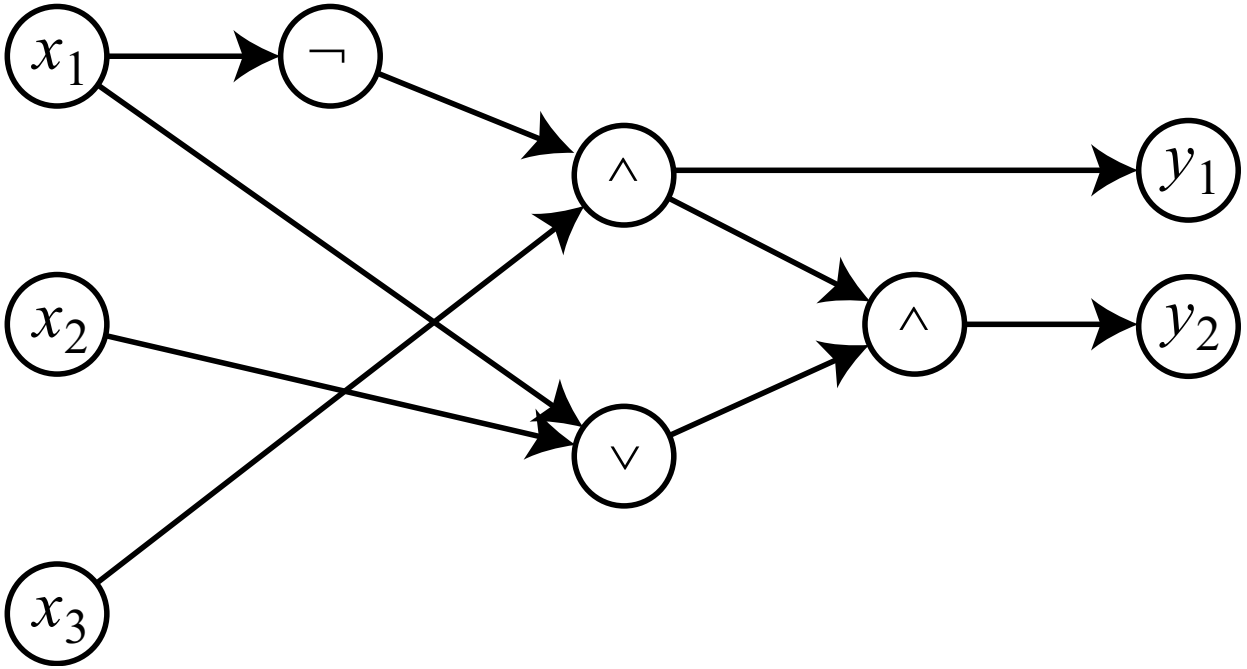
Dans l'équation $z = E$, on dit que l'expression E définit la variable z. Une variable ne peut être définie que par des variables d'input, des constantes ou des variables déjà définies.

Chaque variable d'output doit être définie au moins une fois. ▲

Remarque 3.20. On peut représenter un circuit booléen graphiquement par un circuit. ▲

Exemple 3.21. Deux représentations du même circuit :

$$\begin{aligned} t_1 &= x_1 \vee x_2 \\ y_1 &= \neg x_1 \wedge x_3 \\ y_2 &= t_1 \wedge y_1 \end{aligned}$$



Le circuit précédent donne

$$y_1 = \mathtt{1}, \quad y_2 = \mathtt{1},$$

si

$$x_1 = \mathtt{0}, \quad x_2 = \mathtt{1}, \quad x_3 = \mathtt{1}.$$

$$\begin{aligned} t_1 &= x_1 \vee x_2 = \mathtt{0} \vee \mathtt{1} = \mathtt{1} \\ y_1 &= \neg x_1 \wedge x_3 = \neg \mathtt{0} \wedge \mathtt{1} = \mathtt{1} \wedge \mathtt{1} = \mathtt{1} \\ y_2 &= t_1 \wedge y_1 = \mathtt{1} \wedge \mathtt{1} = \mathtt{1} \end{aligned}$$

▲

Théorème 3.22. Toute fonction booléenne $f(x_1, \ldots, x_n)$ peut être représentée par une expression booléenne, et donc par un circuit booléen.

Preuve. Le circuit est donné par une disjonction de termes

$$t_1 \vee t_2 \vee \ldots \vee t_k$$

où chaque t_i est une conjonction de toutes les variables, certaines variables pouvant être niées.

Pour chaque n-tuple $(a_1, \ldots, a_n)$ tel que $f(a_1, \ldots, a_n) = \langle \text{VRAI} \rangle$ on a exactement un terme. Ce terme prendra la valeur $\langle \text{VRAI} \rangle$ si, et seulement si $x_1 = a_1, \ldots, x_n = a_n$. ■

Exemple 3.23. Soit la fonction f définie par le tableau suivant :

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

alors l'expression booléenne correspondante est

$$(\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2 \wedge \neg x_3).$$

▲

Remarque 3.24. Le théorème 3.22 peut être facilement généralisé aux fonctions de n variables booléennes à m variables booléennes :

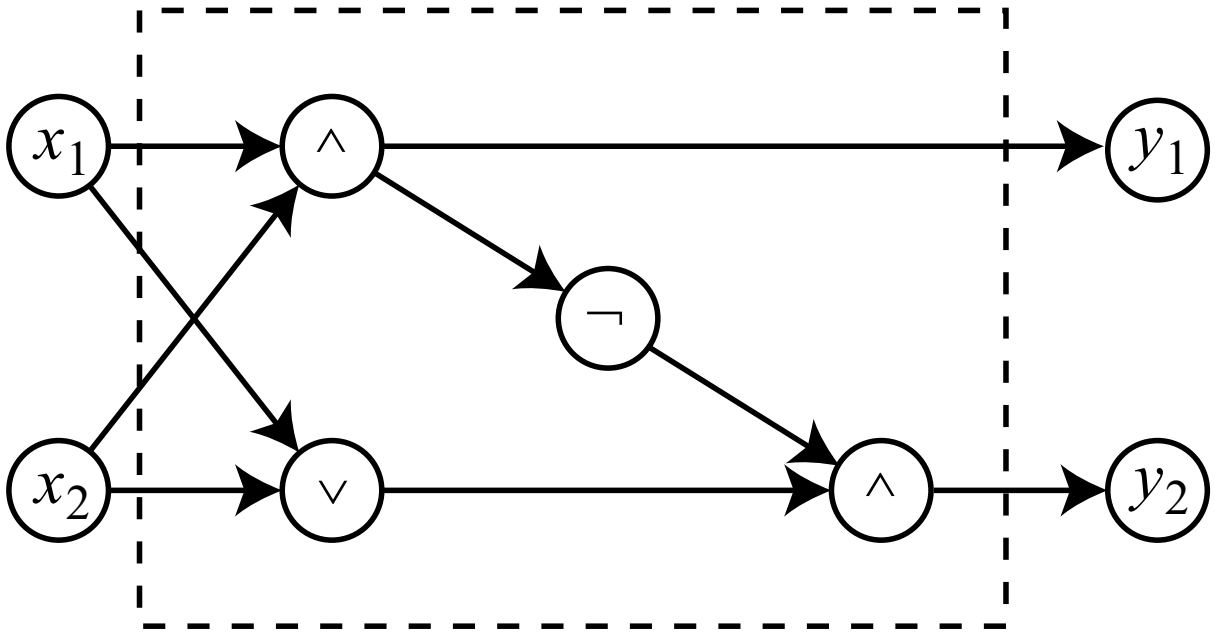
$$(y_1, \ldots, y_m) = f(x_1, \ldots, x_n).$$

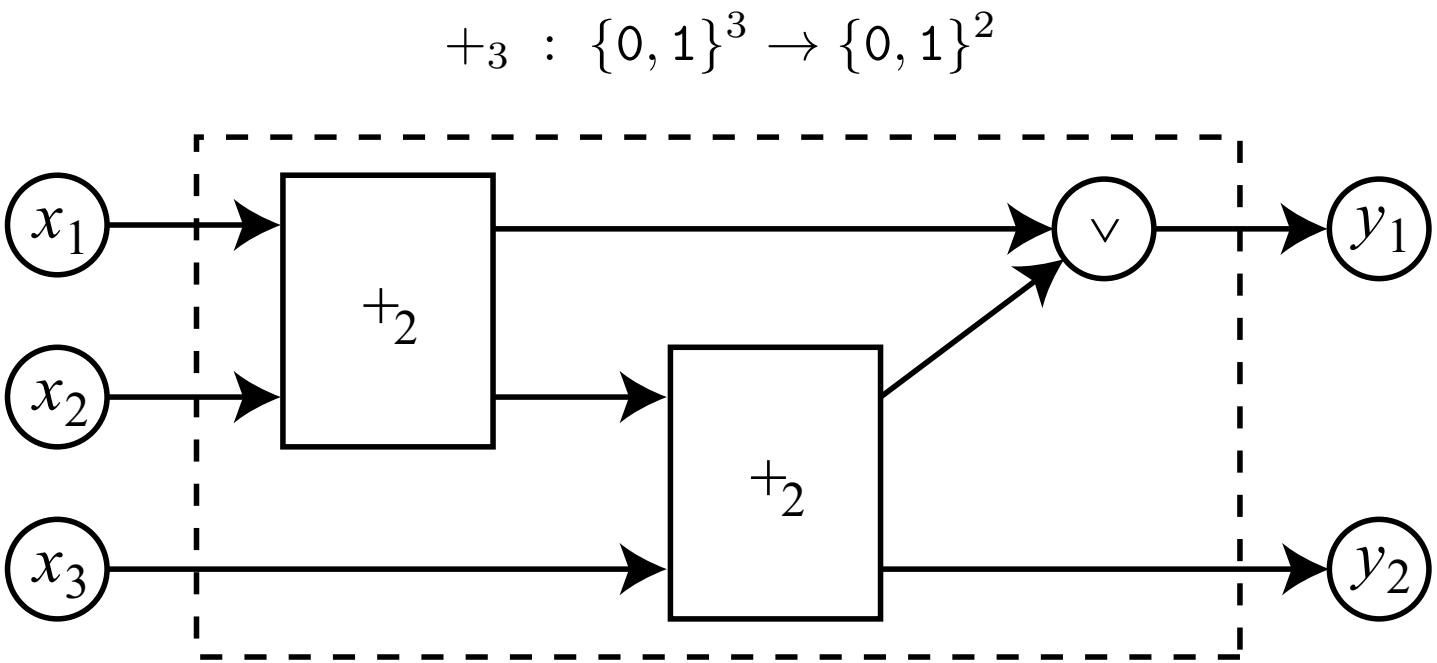
▲

Circuit booléen pour l'addition

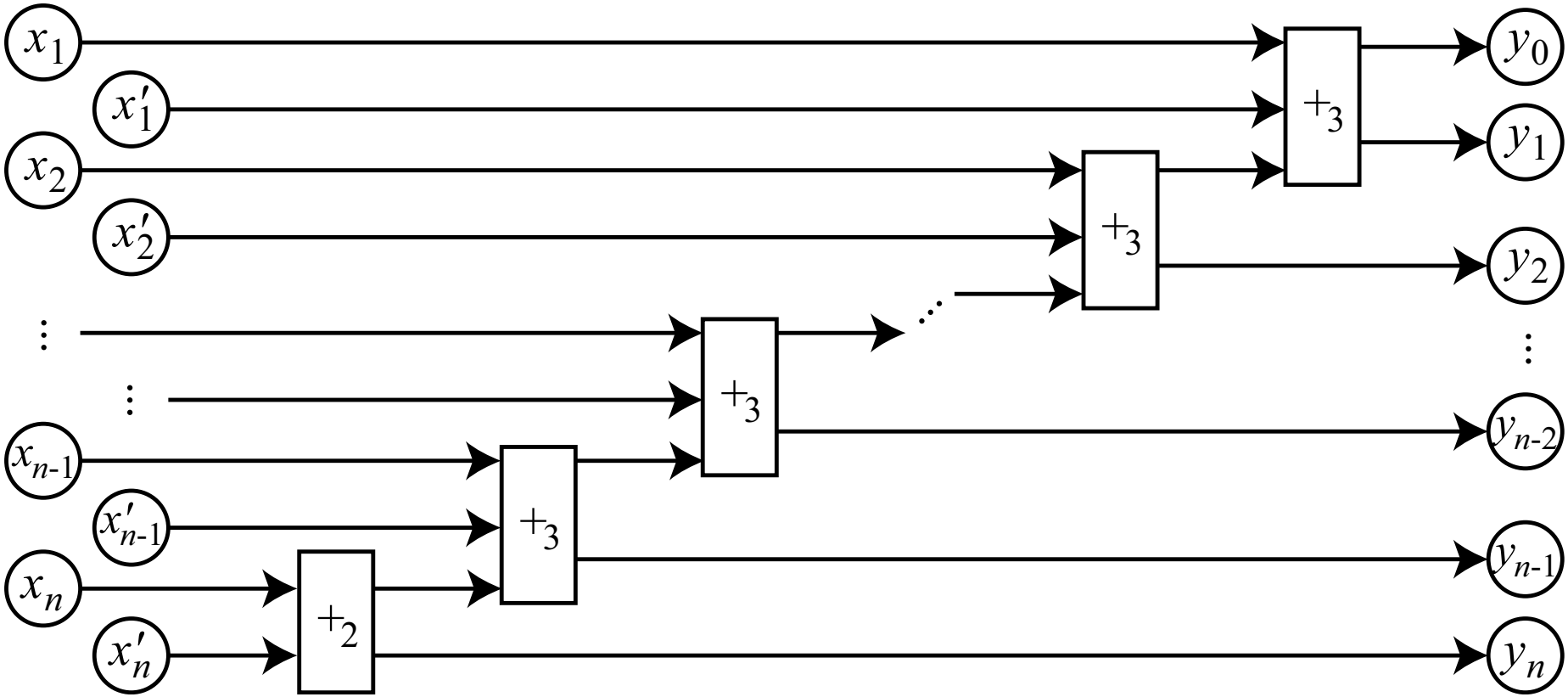
Exemple 3.25.

$$+_2 \ : \ \{0,1\}^2 \to \{0,1\}^2$$




$+_3 : \{0,1\}^3 \to \{0,1\}^2$
x_1
x_2
x_3
$+_2$
$+_2$
$\vee$
y_1
y_2

$$+ \ : \ \{0,1\}^{2n} \to \{0,1\}^{n+1}$$



▲

Le reste est en construction !

Thèse de Church-Turing

“Tout ce qui est intuitivement calculable par un algorithme est calculable par une machine de Turing.”

“Tout modèle de calcul raisonnable peut être simulé efficacement avec une machine de Turing.”

Fonctions calculables

Définition 3.26. Les fonctions calculées par les MT sont appelées calculables. ▲