Machines de Turing.

On travaille avec des machines à $k \geq 1$ rubans. Les rubans sont semi-infinis à droite et sur chaque ruban, on a une tête de lecture qui lit le contenu d'une case.

À chaque étape,

- ▷ la machine M lit les k caractères situés sous les têtes de lecture $(a_1, \ldots, a_k)$;
- ▷ en fonction de son état interne $q \in Q$, et des caractères lus, M remplace chaque a_i par a_i', déplace les têtes de lectures (d'au plus un case à droite ou à gauche), et passe dans un état $q' \in Q$.

1 Définitions.

Définition 1. Formellement, une *machine de Turing* à k rubans est un triplet (Γ, Q, δ) où

- ▷ Γ est l'alphabet du ruban ;
- ▷ Q est un ensemble fini d'états ;
- ▷ $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{\leftarrow, \rightarrow, \mathtt{I}\}^k$ la fonction de transition.

Remarque 1. On fixe

- ▷ un *ruban d'entrée* (généralement supposé en lecture uniquement),
- ▷ un *ruban de sortie* (généralement supposé en écriture uniquement),
- ▷ un état initial q_{initial} ;
- ▷ un état final q_{final} ;

- un symbole blanc $\square \in \Gamma$;
- un symbole de départ $\triangleright \in \Gamma$;
- un alphabet d'entrée $\Sigma \subseteq \Gamma \setminus \{\triangleright, \square\}$. [1]

Au départ, M est dans l'état q_{initial}, le ruban d'entrée contient le mot infini $\triangleright x \square^\infty$ où $x \in \Sigma^\star$ est l'entrée.

Définition 2. On dit que M *calcule* la fonction $f : \Sigma^\star \to \Sigma^\star$ si, pour toute entrée $x \in \Sigma^\star$, le calcul de M termine et $f(x)$ est écrit sur le ruban de sortie.

On dit qu'un langage $L \subseteq \Sigma^\star$ est *reconnu* si on peut calculer la fonction $\mathbb{1}_L : \Sigma^\star \to \{\mathtt{0}, \mathtt{1}\} \subseteq \Sigma$. On peut aussi avoir un état q_{accepte} acceptant et un état q_{rejet} de rejet.

Remarque 2 (Variantes). On peut considérer un modèle avec des rubans bi-infinis. C'est un modèle équivalent (*c.f.* TD).

On peut considérer une machine avec $\Gamma = \{\mathtt{0}, \mathtt{1}, \mathtt{2}, \mathtt{3}, \square, \triangleright\}$, c'est-à-dire un alphabet plus gros. Ce modèle est équivalent avec l'association

$$\mathtt{0} \leftrightarrow \mathtt{00} \qquad \mathtt{1} \leftrightarrow \mathtt{01} \qquad \mathtt{2} \leftrightarrow \mathtt{10} \qquad \mathtt{3} \leftrightarrow \mathtt{11}.$$

Définition 3. Un langage $L \subseteq \Sigma^\star$ est *reconnu en temps* $T(n)$ par une machine M si

- M reconnaît L ;
- sur toute entrée x de taille n, la machine M s'arrête en au plus $T(n)$ étapes de calcul.

Définition 4. On dit que L est dans la classe $\mathrm{DTIME}(f(n))$ s'il existe une machine (à plusieurs rubans) qui reconnaît L en temps $\mathrm{O}(f(n))$. On supposera toujours avoir $f(n) \geq n + 1$. [2]

1. Généralement, on prendra $\Sigma \subseteq \{\mathtt{0}, \mathtt{1}\}$.
2. Il faut, au moins, lire l'entrée.

Définition 5. La classe P est l'ensemble des langages reconnaissables en temps polynomial :

$$\mathsf{P} = \bigcup_{\alpha \geq 1} \mathrm{DTIME}(n^\alpha).$$

Théorème 1. Si $L \in \mathrm{DTIME}(f(n))$ alors L peut être reconnu en temps $\mathrm{O}(f(n)^2)$ sur une machine à un ruban.

Théorème 2 (Simulation efficace). Pour toute machine M fonctionnant en temps $T(n)$, il existe une machine M' à deux rubans qui fonctionne en temps $\mathrm{O}(T(n) \log T(n))$ telle que $M(x) = M'(x)$ pour toute entrée $x \in \Sigma^\star$.

Définition 6. Étant donné x, y, on définit l'encodage du couple (x, y) comme le mot $\langle x, y\rangle = \mathtt{1}^{|x|}\mathtt{0}xy$.

Définition 7. Une *machine universelle* U prend en entrée des coules $\langle x, \alpha\rangle$ (où α est le code d'une machine M_α et $x \in \Sigma^\star$) et simule la machine M_α sur l'entrée x. Autrement dit, pour tout x et tout α, on a

$$U(\langle x, \alpha\rangle) = M_\alpha(x).$$

Théorème 3. Il existe une machine de Turing universelle U telle que, sur tout entrée $\langle x, \alpha\rangle$, si M_α s'arrête sur x en t étapes, alors la machine U s'arrête sur $\langle x, \alpha\rangle$ en au plus $ct \log t$, où c ne dépend que de x.

Preuve. ▷ *Cas d'une machine à deux rubans.* On montre d'abord que si M_α est une machine à deux rubans, alors U peut simuler M_α en temps linéaire. En effet, on utilise quatre rubans (figure 1) :

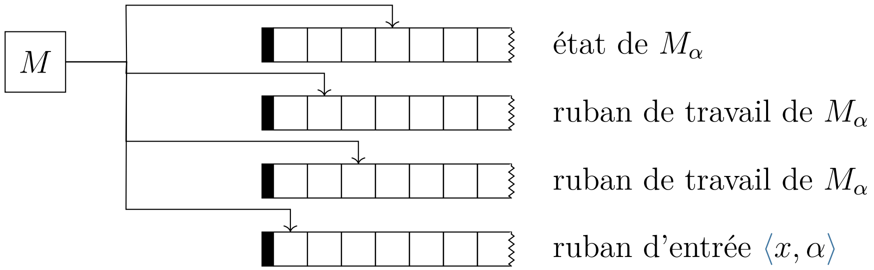

Figure 1 | *Construction de U dans le cas à deux rubans*

– deux rubans pour stocker les deux rubans de M_α ;
– un ruban qui stocke l'état de M_α ;
– le ruban d'entrée qui contient $\langle x, \alpha \rangle$.

Pour faire une étape de calcul de M_α, la machine U doit déterminer $\delta(q, a, b)$, mais la complexité de cette opération est cachée dans la constante.

▷ *Cas général.* Par le théorème de simulation efficace, on peut construire une machine M_β à deux rubans équivalente et on peut simuler M_β en temps linéaire, et on obtient bien la complexité attendue.

□

2 Non déterminisme.

Définition 8. Une *machine de Turing non-déterministe* est un triplet (Γ, Q, δ) où

$$\delta : Q \times \Gamma^k \to \wp(Q \times \Gamma^k \times \{\leftarrow, \rightarrow, \mathrm{I}\}^k),$$

où l'on distingue deux états q_{accepte} et q_{rejet}.

Une entrée $x \in \Sigma^\star$ est *acceptée* s'il existe un chemin d'exécution acceptant sur l'entrée x.

Définition 9. On note $\mathrm{NTIME}(f(n))$ l'ensemble des langages acceptés par une machine de Turing non-déterministe fonctionnant en temps $\mathrm{O}(f(n))$ sur toute entrée de taille n et tout chemin de calcul.

Définition 10. Un langage L est dans NP s'il existe une machine non-déterministe M fonctionnant en temps polynomial tel que L est l'ensemble des entrées acceptées par M. Ainsi,

$$\mathsf{NP} = \bigcup_{\alpha \geq 1} \mathrm{NTIME}(n^\alpha).$$

Théorème 4 (Définition alternative de NP). Un langage L est dans NP s'il existe un polynôme p et $A \in \mathsf{P}$ tel que, pour toute entrée $x \in \{\mathtt{0}, \mathtt{1}\}^\star$,

$$x \in L \quad\iff\quad \exists y \in \{\mathtt{0}, \mathtt{1}\}^\star, \quad \langle x, y\rangle \in A.$$

On dit que y *certifie* que x est dans L.

3 NP-complétude.

Définition 11. On dit qu'un problème A *se réduit à* B *en temps polynomial* s'il existe une fonction $f : \Sigma^\star \to \Sigma^\star$ calculable en temps polynomial telle que, pour toute entrée x,

$$x \in A \quad\iff\quad f(x) \in B.$$

On note alors $A \leq_{\mathrm{P}} B$ ou $A \leq_{\mathrm{M}} B$.[3]

Remarque 3. Si $A \leq_{\mathrm{P}} B$ et $B \leq_{\mathrm{P}} C$ alors $A \leq_{\mathrm{P}} C$ par composition des réductions.

3. Personnellement, je noterai parfois $f : A \leq_{\mathrm{P}} B$ où f est une fonction de réduction.

Définition 12 (NP-complétude). On dit que A est NP-*complet* dès lors que

- ▷ $A \in \mathsf{NP}$;
- ▷ A est NP-*dur*, c'est-à-dire $B \leq_{\mathrm{P}} A$ pour tout $B \in \mathsf{NP}$.

Remarque 4. Pour montrer qu'un problème $A \in \mathsf{NP}$ est NP-complet, il suffit de montrer que $B \leq_{\mathrm{P}} A$ où B est un problème NP-complet. En effet, on a alors, pour tout $C \in \mathsf{NP}$,

$$C \leq_{\mathrm{P}} B \leq_{\mathrm{P}} A.$$

Il suffit donc d'avoir un problème NP-complet comme « point de départ ». Généralement, on choisit Sat ou 3-Sat, mais dans ce cours on partira de CircuitSat (défini au prochain chapitre).

Sat	**Entrée.** Une formule booléenne ϕ **Sortie.** La formule ϕ est-elle satisfiable ?

3-Sat	**Entrée.** Une formule booléenne ϕ sous 3-CNF **Sortie.** La formule ϕ est-elle satisfiable ?