Calculabilité
Cours 2 : machines de Turing

Kévin Perrot – L3 Info Aix Marseille Université – printemps 2022-23

Table des matières

3 Machines de Turing

Pour montrer qu'une fonction est calculable ou qu'un langage est décidable (distinction discutée en 3.2) il faut donner un algorithme correct. Pour montrer qu'une fonction n'est pas calculable ou qu'un langage est indécibable, il faut montrer que tout algorithme est incorrect, mais pour cela il faut d'abord définir l'ensemble des algorithmes (ce qui définit l'ensemble de ce qui est calculable/décibable). L'intérêt des machines de Turing est qu'elles définissent les algorithmes de façon intuitive et simple ! Imaginez devoir définir mathématiquement votre langage de programmation préféré dans ses moindres détails...

L'idée d'Alan Turing est inspirée du calculateur humain devant sa feuille [1] :
— feuilles découpées en cases : ruban ;
— crayon posé sur une case : tête de lecture/écriture, déplacement ;
— l'opérateur dispose d'une mémoire finie (son cerveau) : états.

3.1 Définitions

Définition 1. *Une* **machine de Turing** *(***MT***) déterministe est un 7-uplet*

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

où
— Q *est un ensemble fini : les* **états**,
— Γ *est un ensemble fini : l'***alphabet de ruban**,
— $\Sigma \subset \Gamma$ *est l'***alphabet d'entrée**,
— $\delta : (Q \setminus \{q_F\}) \times \Gamma \to Q \times \Gamma \times \{L, R\}$ *est la* **fonction de transition** *décrite ci-après,*
— $q_0 \in Q$ *est l'***état initial**,
— $B \in \Gamma \setminus \Sigma$ *est le* **symbole blanc**,
— $q_F \in Q$ *est l'***état final**.

Γ contient tous les symboles qui peuvent apparaître sur le ruban. En particulier $\Sigma \subset \Gamma$ car l'entrée est initialement écrite sur le ruban. On supposera que $Q \cap \Gamma = \emptyset$ pour qu'il n'y ait pas de confusion entre états et symboles du ruban.

La fonction de transition δ est une application partielle

de l'ensemble $(Q \setminus \{q_F\}) \times \Gamma$ dans l'ensemble $Q \times \Gamma \times \{L, R\}$.

Une transition

$$\delta(q, a) = (p, b, L)$$

signifie que dans l'état q et en lisant le symbole de ruban a, la machine passe dans l'état p, remplace a par b sur le ruban, et déplace la tête de lecture/écriture d'une cellule sur la gauche (L pour *left* et R pour *right*). Une application partielle peut être indéfinie pour certains arguments, auquel cas la machine n'a pas de mouvement suivant et s'arrête. En particulier, il n'y a pas de transition depuis l'état final q_F.

Pour décrire une machine de Turing, en général on la dessine sous la forme d'un automate dont les sommets sont les états (l'ensemble Q) et avec un arc de q vers p étiqueté $a \mid b, \leftarrow$ lorsque $\delta(q, a) = (p, b, L)$. Des exemples sont donnés en fin de document.

Initialement, le mot d'entrée est écrit sur le ruban et toutes les autres cellules contiennent le symbole blanc B. La machine est dans l'état q_0, et la tête de lecture/écriture est positionnée sur la lettre la plus à gauche de l'entrée. Il y a trois possibilités :

- **acceptation** si au cours des transitions la machine entre dans l'état final q_F (et donc s'arrête),
- **rejet** si au cours des transitions la machine s'arrête dans un état non final (s'il n'y a pas de mouvement suivant à réaliser),
- **rejet** si la machine ne s'arrête jamais.

Définition 2. *Une* **description instantanée** *(***DI***) d'une MT décrit sa configuration courante. C'est un mot*

$$uqav \in (\{\epsilon\} \cup (\Gamma \setminus \{B\})\Gamma^*)Q\Gamma(\{\epsilon\} \cup \Gamma^*(\Gamma \setminus \{B\})$$

avec $q \in Q$ l'état courant, $u, v \in \Gamma^*$ le contenu du ruban à gauche et à droite de la tête, respectivement, jusqu'au dernier symbole non blanc, et $a \in \Gamma$ le symbole de ruban actuellement sous la tête.

Définition 3. *Un* **mouvement**, *une* **transition**, *un* **déplacement** *de la MT à partir de la DI $\alpha = uqav$ vers la DI suivante β sera noté $\alpha \vdash \beta$. Plus précisément :*

1. *Si $\delta(q, a) = (p, b, L)$,*
   - *si $u = \epsilon$ alors $\beta = pBbv$ (potentiellement en supprimant les B à la fin de bv),*
   - *si $u = u'c$ avec $c \in \Gamma$ alors $\beta = u'pcbv$ (potentiellement en supprimant les B à la fin de bv).*
2. *Si $\delta(q, a) = (p, b, R)$,*
   - *si $v = \epsilon$ alors $\beta = ubpB$ (potentiellement en supprimant les B au début de ub),*
   - *si $v \neq \epsilon$ alors $\beta = ubpv$ (potentiellement en supprimant les B au début de ub).*
3. *Si $\delta(q, a)$ est indéfini alors aucun mouvement n'est possible depuis α, et α est une* **DI d'arrêt**. *Si $q = q_F$ alors α est une* **DI acceptante**.

Notation 4. *Notre modèle de MT est* **déterministe**, *ce qui signifie que pour tout α il y a au plus un β tel que $\alpha \vdash \beta$. Nous noterons*

$$\alpha \vdash^* \beta$$

si la MT change α en β en n'importe quel nombre d'étapes (0 inclus, auquel cas $\alpha = \beta$),
$\alpha \vdash^+ \beta$ *si la MT change α en β en au moins une étape, et*
$\alpha \vdash^i \beta$ *si la MT change α en β en exactement i étapes.*

Pour tout $w \in \Sigma^*$ nous pouvons définir la DI de départ correspondante

$$\iota_w = \begin{cases} q_0 w, & \text{si } w \neq \epsilon \\ q_0 B & \text{si } w = \epsilon. \end{cases}$$

3.2 Décider et calculer

Définition 5. *Le langage* **reconnu** *(ou* **accepté***) par la MT M est*

$$L(M) = \{w \mid w \in \Sigma^* \text{ et } \iota_w \vdash^* uq_F v \text{ avec } u, v \in \Gamma^*\}$$

Définition 6. *Un langage est* **semi-décidable** *s'il est reconnu par une machine de Turing. Un langage est* **décibable**, *s'il est reconnu par une machine de Turing qui s'arrête sur toutes les entrées.*

Attention à la différence ! Tout langage décidable est également semi-décidable.

Notation 7. *Le* **résultat** *du calcul de la MT M sur l'entrée w sera noté*

$$M(w) = \begin{cases} uv & \text{si } \iota_w \vdash^* uq_F v \text{ avec } u, v \in \Gamma^* \\ uav & \text{si } \iota_w \vdash^* uqav \text{ avec } u, v \in \Gamma^* \text{ et } \delta(q, a) \text{ non défini}\dagger \\ \uparrow & \text{si l'exécution ne termine pas.} \end{cases}$$

†potentiellement en supprimant les B à la fin de av.

Définition 8. *Une fonction $f : \Sigma^* \to \Gamma^*$ est* **calculable** *si et seulement si il existe une MT M telle que pour tout $w \in \Sigma^* : f(w) = M(w)$.*

Remarque 9. Décider (un langage) est équivalent à calculer (une fonction).
$\Leftarrow$ *Décider un langage L revient à calculer sa* **fonction caractéristique**

$$f_L : \begin{array}{rl} \Sigma^* & \to \{0, 1\} \\ w & \mapsto \begin{cases} 1 & \text{si } w \in L \\ 0 & \text{sinon.} \end{cases} \end{array}$$

$\Rightarrow$ *Calculer une fonction f revient à décider le langage*

$$L_f = \{(x, y) \mid y = f(x)\}.$$

Nous n'établirons pas de distinction très nette entre calculer et décider.
— Dans la vraie vie on dira plutôt calculer (plus parlant).
— Dans le monde des mathématiques on dira plutôt décider (plus simple à formaliser).
— **Récursif** est synonyme de calculable et décidable.

Remarque 10. *Le terme* **semi-décidable** *(définition 6) vient du fait que la machine de Turing qui semi-décide s'arrête pour toute entrée qui appartient au langage (on a à coup sûr la réponse si le mot appartient au langage car la machine atteindra un état final*

acceptant), mais ne s'arrêt pas obligatoirement sur les entrées qui n'appartiennent pas au langage (si le mot n'appartient pas au langage, la machine peut ne pas s'arrêter). Il y a donc une asymétrie entre les mots dans *et* en dehors *du langage. En lançant une telle machine sur un mot d'entrée dont on se demande s'il appartient au langage, on ne sait pas si la machine va s'arrêter et donner une réponse, mais on sait que si le mot est* dans *le langage alors la machine finira par nous donner la réponse en l'acceptant au bout d'un temps fini (mais a priori inconnu).*

Pour semi-décidable *on dit également* **récursivement énumérable** *car il est possible d'écrire (pour ces langages) une MT qui va, à partir d'une entrée vide, énumérer tous les mots du langage, un à un et sans en oublier aucun (dans n'importe quel ordre, possiblement en répétant plusieurs fois certains mots). Pour formaliser cette idée nous aurons un* état spécial d'énumération q_e *(quand on entre dans cet état c'est qu'on énumère le mot présent sur le ruban, par convention à la droite de la tête de lecture/écriture) tel que :*

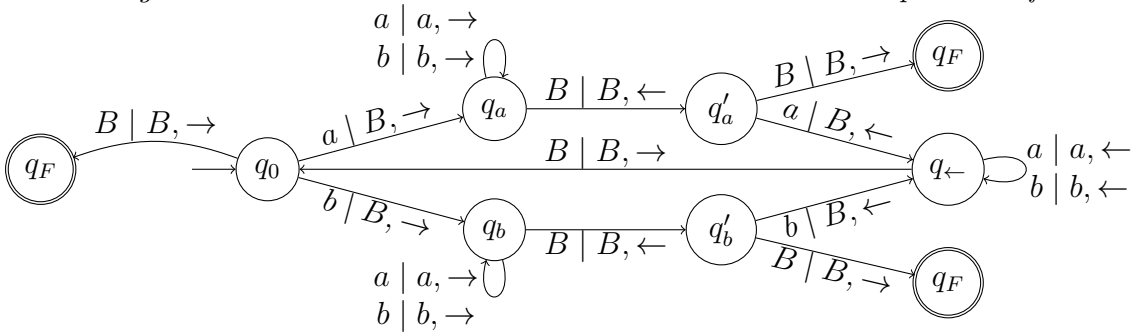
$$\textit{pour tout mot } w \in L, \textit{ il existe une étape } t \textit{ telle que } \iota_\epsilon \vdash^t w' q_e w.$$

Exemple 11. *Le langage suivant est décidable :*

$$\{w \in \{a,b\}^* \mid w \textit{ est un palindrome}\}$$

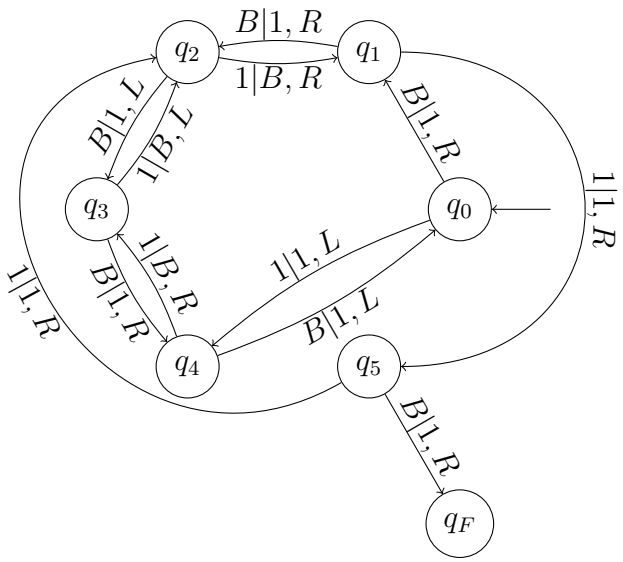
donc il existe une machine $M_{palindrome}$ qui le décide (répond oui/non sur toute entrée).

Rappelons qu'un palindrome est un mot qui se lit identiquement de gauche à droite, et de droite à gauche. Notre idée consiste à effacer les lettres du mot d'entrée de la gauche vers la droite, en vérifiant à chaque fois que la lettre correspondante à l'opposée du mot est identique (et en l'effaçant également). Nous avons un palindrome si et seulement si nous atteignons le mot vide. Par soucis de lisibilité nous écrivons plusieurs fois l'état q_F.



Les états q_a et q_b servent à parcourir le mot de gauche à droire jusqu'à la lettre correspondante. Les états q'_a et q'_b vérifient l'identité des lettres en début et fin du mot d'entrée (s'il comporte un nombre impair de lettres, alors la dernière lettre est comparée avec B). L'état $q_\leftarrow$ sert à revenir au début du mot.

Exemple 12. *Que se passe-t-il quand on lance la machine ci-dessous sur l'entrée vide ϵ (c'est-à-dire sur un ruban initial où chaque case contient le symbole B) ?*



Exemple 13. *Beau simulateur de machines de Turing :* `https://turingmachine.io/`.

3.3 Propriétés de clôture

Théorème 14. *Les propriétés suivantes sont vraies :*

1. *la famille des langages décidables est close par complémentation ;*
2. *les familles des langages décidables et semi-décidables sont closes par union et intersection ;*
3. *Un langage $L \subseteq \Sigma^*$ est décidable si et seulement si L et $\Sigma^* \setminus L$ sont semi-décidables.*

Idées de démonstration.

1. On veut prouver L décidable implique $\Sigma^* \setminus L$ décidable. Soit M la machine dont le langage est $L(M) = L$ et qui s'arrête toujours, nous allons construire une nouvelle machine M' dont le langage est $L(M') = \Sigma^* \setminus L$ et qui s'arrête toujours. Pour cela, on ajoute un puits global qui sera notre nouvel état final (toutes les transitions indéfinies de M mènent vers le nouvel état final de M', et l'état final de M n'a aucune transition dans M'). Ainsi, la machine M' s'arrête dans son état final à chaque fois que M s'arrêtait sur un état non final ($w \notin L(M) \Rightarrow w \in L(M')$). De plus, chaque fois que M s'arrêtait dans son état final, M' s'arrête également mais cet état n'est plus final ($w \in L(M) \Rightarrow w \notin L(M')$).
2. Concentrons nous sur l'intersection de deux lanages décidables, les autres démonstrations sont analogues. Soient $L_1 = L(M_1)$ et $L_2 = L(M_2)$. On veut construire une machine M' qui décide le langage $L(M') = L_1 \cap L_2$. Pour cela, sur une entrée w la machine M' simule (pour cela il suffit de modifier l'état final des machines simulées) :
   — M_1 sur l'entrée w (on sait que le calcul termine),
   — puis M_2 sur l'entrée w (on sait que le calcul termine),
   se souvient du résultat de chaque simulation (arrêt dans l'état final ou non), et va dans l'état final si et seulement si M_1 et M_2 acceptent w (sinon M' va dans un nouvel état non final à partir duquel aucune transition n'est possible).
3. Le sens $\Rightarrow$ est assez simple et laissé en exercice. Pour $\Leftarrow$, on construit une machine qui simule en parallèle les deux machines qui reconnaissent L et $L \setminus \Sigma^*$. Chacune

peut ne pas s'arrêter, mais puisque soit l'une soit l'autre accepte w, soit l'une soit l'autre entrera dans son état final, et nous pourrons alors :

— entrer dans notre état final si c'est la machine qui reconnait L qui est entrée dans son état final,

— entrer dans un nouvel état non final sans transition si c'est la machine qui reconnait $\Sigma^* \setminus L$ qui est entrée dans son état final.

□

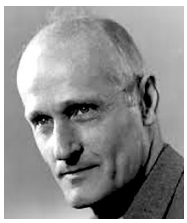
3.4 Un peu d'histoire


Cantor
(1845-1918)


Hilbert
(1862-1943)


Gödel
(1906-1978)


Church
(1903-1995)


Kleene
(1909-1994)


Turing
(1912-1954)


von Neumann
(1903-1957)

A la toute fin du XIX^{e} siècle, Georg Cantor définit les fondements de la théorie des ensembles, dont l'usage systématique (c'est-à-dire qui est utilisée dans tous les domaines) allait bouleverser les fondements de la logique mathématique. En 1900, pour fêter le passage au XX^{e} siècle, David Hilbert énonce 23 grands problèmes ouverts, dont le suivant : les propriétés qui s'expriment en langage mathématique sont-elles toutes décidables ? Si la réponse devait être affirmatives, les propriétés mathématiques valides seraient des théorèmes dérivables mécaniquement de quelques axiomes dans un système formel. Autrement dit : on pourrait remplacer les mathématicien·ne·s par des machines surpuissantes ! En 1931, Kurt Gödel met un terme à cette interrogation : il existe des propriétés mathématiques indécidables (dans tous les systèmes d'axiomes qui formalisent au moins l'arithmétique). Autrement dit : mathématicien·ne·s 1 - machines 0. Entre 1932 et 1936, Alsonso Church et Stephen Kleene proposent des modèles de calculs (le λ-calcul et les fonctions μ-récursives) qui semblent capturer la notion intuitive de fonctions calculables, mais il est un peu difficile de s'en convaincre... Notons tout de même que le λ-calcul est extrêmement minimaliste, ce qui rend sa compréhension mathématique fort intéressante : tout est capturé en quelques lignes de définition ! Indépendamment, en 1936, Alan Turing propose sa définition de machines. En 1937 il montre que la classe des fonctions λ-calculables est égale à la classe des fonctions programmables sur les machines de Turing. Les machines de Turing permettent de reformuler en termes intuitifs de calculs les résultats de Kurt Gödel (qui étaient exprimés en termes de démonstration). Avec l'aide de Von Neumann (et d'autres), les premiers ordinateurs programmables verront le jour quelques années plus tard !

La vie de Turing vaut le coup d'oeil (savez vous que le rôle de Turing durant la seconde guerre mondiale est resté secret d'Etat de nombreuses années ?).

e-penser (13') : `https://www.youtube.com/watch?v=7dpFeXV_hqs`

Références

[1] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42) :230–265, 1936.