Théorème de Rice
version 1.2
le 8 avril 2021

S.V.P. ME SIGNALER TOUTE FAUTE DE FRAPPE, GRAMMAIRE, ORTHOGRAPHE OU LOGIQUE; LES FAUTES DE LOGIQUE VOUS APPORTENT DES POINTS SUPPLÉMENTAIRES POUR LA NOTE FINALE. TOUTE SUGGESTION D'UNE MEILLEURE TRADUCTION D'UN TERME ANGLAIS SERA ÉGALEMENT BIENVENUE.

Soit X un ensemble et $P \subseteq X$. On appelle P une *propriété*. Cela peut parraître bizarre, mais pensez aux exemples simples: avoir des cheveux noirs veut dire appartenir à la partie de l'ensemble d'humains qui contient tous les humains aux cheveux noirs; une voiture est bleue si et seulement si elle appartient à l'ensemble $B \subseteq V$, où V est l'ensemble de voitures. Etc.
Une propriété P est *non-triviale* si $\emptyset \neq P \neq X$, c'est-à-dire, il y existe $x \in P$ et $y \in X \setminus P$.

Soit $\mathcal{L}_{rec}$ l'ensemble de langages sur $\{0, 1\}$ reconnaissables (donc $L \in \mathcal{L}_{rec}$ quand il existe un machine de Turing qui le reconnaît). Une propriété $\mathcal{S}$ de $\mathcal{L}_{rec}$ est *décidable* (*reconnaissable*) si le langage $L_{\mathcal{S}} = \{\langle M \rangle : L(M) \in \mathcal{S}\}$ l'est. Notons que $\mathcal{S}$ est décidable si et seulement si $\overline{\mathcal{S}} = \{\langle M \rangle : L(M) \notin \mathcal{S}\}$ l'est. Pour le voir, il suffit de se souvenir de notre convention que tout mot sur $\{0, 1\}$ est le code d'une machine de Turing; si le mot n'est pas de format correspondant à une vraie machine de Turing, son langage est vide. Si on veut se passer de la convention, alors il faut récrire les définitions un peu : $L_{\mathcal{S}} = \{x : x \text{ est le code d'une machine de Turing } M \text{ et } L(M) \in \mathcal{S}\}$, et $\overline{\mathcal{S}} = \{x : x \text{ est le code d'une machine de Turing } M \text{ et } L(M) \notin \mathcal{S}\}$. Il devrait alors être facile de voir même dans ce cas que $\mathcal{S}$ est décidable si et seulement si $\overline{\mathcal{S}} = \{\langle M \rangle : L(M) \notin \mathcal{S}\}$ l'est (c'est un bon test de compréhension).

Théorème 1 *(Rice) Toute propriété non triviale de langages reconnaissables est indécidable.*

Démonstration. On va donner deux preuves différentes mais semblable. Toutes les deux reposent sur la construction algorithmique (donc réalisable par une machine de Turing) d'une machine de Turing $M_{\langle M,\, w\rangle}$ à partir d'une machine de Turing M et son mot d'entrée w de façon à ce que $L(M_{\langle M,\, w\rangle}) \in \mathcal{S}$ si et seulement si $w \in L(M)$.

Preuve I.

Sans perte de généralité, on peut supposer que $\emptyset \notin \mathcal{S}$ - sinon, on considère $\overline{\mathcal{S}}$, par la remarque ci-haut. Puisque $\mathcal{S}$ est une propriété non-triviale, il existe un langage $L \in \mathcal{S}$ et une machine de Turing M_L qui le reconnaît (i.e. telle que $L(M) = L$). Etant donné une machine de Turing M et un mot $w \in \{0,1\}^*$, soit $M_{\langle M,\ w\rangle}$ la machine de Turing qui démarre avec x sur son ruban mais l'ignore d'abord at simule M sur w, pour simuler ensuite M_L sur x si $w \in L(M)$ (donc M_L n'est pas utilisé si M ne s'arrête pas en acceptant w). On a alors $L(M_{\langle M,\ w\rangle}) = L$ si $w \in L(M)$ et $L(M_{\langle M,\ w\rangle}) = \emptyset$ sinon. En d'autre mots, $L(M_{\langle M,\ w\rangle}) \in S$ si et seulement si $w \in L(M)$.
Supposons qu'une machine de Turing $M_{\mathcal{S}}$ existe qui décide $L_{\mathcal{S}}$. Elle sera alors capable de décider si le langage de $M_{\langle M,\ w\rangle}$ est dans $\mathcal{S}$ ou non, i.e. si $w \in L(M)$ ou non. Avec ceci on construit donc une machine M_u qui décide le langage universel L_u qui fonctionne comme ceci : sur l'entrée $\langle M,\ w\rangle$, elle produit le code $\langle M_{\langle M,\ w\rangle}\rangle$ et elle simule $M_{\mathcal{S}}$ sur $\langle M_{\langle M,\ w\rangle}\rangle$. Mais une telle machine ne peut pas exister, L_u est indécidable. Donc $L_{\mathcal{S}}$ n'existe pas.

Preuve II.

Puisque $\mathcal{S}$ est une propriété non-triviale, il existe un langage $L \in \mathcal{S}$ est une machine de Turing M_L qui le reconnaît (i.e. telle que $L(M) = L$). Il existe également un langage $\overline{L} \notin \mathcal{S}$ et une machine de Turing $M_{\overline{L}}$ qui le reconnaît.
Etant donné une machine de Turing M et un mot $w \in \{0,1\}^*$, soit $M_{\langle M,\ w\rangle}$ la machine de Turing qui démarre avec x sur son ruban et simule en parallèle $M_{\overline{L}}$ et M_L sur x. Si $x \in L$, elle simule M sur w et accepte x si $w \in L(M)$; si $x \in \overline{L}$, elle accepte (notons que $L \cap \overline{L} = \emptyset$, par définitions, donc soit $x \in L$, soit $x \in \overline{L}$, mais pas les deux; bien sur, il se peut que $x \notin L \cup \overline{L}$). On a alors $L(M_{\langle M,\ w\rangle}) = L$ si $w \in L(M)$ et $L(M_{\langle M,\ w\rangle}) = \overline{L}$ sinon. En d'autre mots, $L(M_{\langle M,\ w\rangle}) \in S$ si et seulement si $w \in L(M)$.
Supposons qu'une machine de Turing $M_{\mathcal{S}}$ existe qui décide $L_{\mathcal{S}}$. Elle sera alors capable de décider si le langage de $M_{\langle M,\ w\rangle}$ est dans $\mathcal{S}$ ou non, i.e. si $w \in L(M)$ ou non. Avec ceci on construit donc une machine M_u qui décide le langage universel L_u qui fonctionne comme ceci : sur l'entrée $\langle M,\ w\rangle$, elle produit le code $\langle M_{\langle M,\ w\rangle}\rangle$ et elle simule $M_{\mathcal{S}}$ sur $\langle M_{\langle M,\ w\rangle}\rangle$. Mais une telle machine ne peut pas exister, L_u est indécidable. Donc $L_{\mathcal{S}}$ n'existe pas.