Motivations :

* Les machines de Turing sont un modèle abstrait des ordinateurs
* Elles sont la base des théories de calculabilité et de la complexité.

I - Les machines de Turing : un modèle de calcul formel.

Définition : [2, p. 115] Un modèle de calcul est un système qui associe à une entrée x, une sortie y en un nombre fini d'opérations élémentaires.

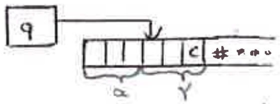
A - Vocabulaire autour des machines de Turing. [5, p. 103]

Définition : Une machine de Turing déterministe est décrite formellement par un septuplet $M=(Q,\Gamma,\Sigma,\delta,s,\#,F)$ où

* Q est un ensemble fini d'états ;
* Γ est l'alphabet de ruban ;
* $\Sigma \subseteq \Gamma$ est l'alphabet d'entrée ;
* $s \in Q$ est l'état initial ;
* $F \subseteq Q$ est l'ensemble des états acceptants ;
* $\# \in \Gamma \setminus \Sigma$ est le symbole blanc ;
* $\delta : Q\times\Gamma \to Q\times\Gamma\times\{\leftarrow;\rightarrow\}$ la fonction de transition.

Définition : Une configuration est un élément de $\Gamma^* \times Q \times (\varepsilon \cup \Gamma^*(\Gamma\setminus\{\#\}))$

Exemple :



$\rightsquigarrow (\alpha, q, \gamma)$ avec $c \neq \#$

Application : [2, p. 132] Un automate fini est une machine de Turing parcourant une seule fois le ruban sans écrire (machine de Turing sans mémoire).

Définition : Soit $c=(\alpha a, q, b\gamma)$ une configuration. La configuration suivante est

* Si $\delta(q,b) = (q',b',\rightarrow)$, $c' = (\alpha a b', q', \gamma)$
* Si $\delta(q,b) = (q',b',\leftarrow)$, la configuration suivante est $c' = (\alpha, q', a b'\gamma)$

On note $c \vdash c'$.

Définition : L'exécution d'une machine de Turing sur un mot w est la suite de configurations $(\varepsilon,s,w) \vdash_M c_1 \vdash_M \cdots$ maximale telle qu'elle soit infinie, ou se terminant sur un état acceptant, ou sur un état dont la configuration suivante n'est pas définie.

Définition : Le langage $L(M)$ accepté par une machine M de Turing est l'ensemble des mots w tels que
$(\varepsilon, s, w) \vdash^* (\alpha, p, \gamma)$ avec $p \in F$.

On dit que le langage $L(M)$ est décidé si M n'a pas d'exécution infinie.

B - Les machines de Turing calculent. [4, p. 59]

Principe : Les machines de Turing peuvent calculer les fonctions élémentaires de l'arithmétique.

Exemples : voir annexe (Figure 1)

(*)

II - Justifications de la thèse de Church.

Thèse de Church : [5, p. 109] Les langages reconnus par un algorithme sont les langages décidés par une machine de Turing.

A - Les extensions d'une machine de Turing décident les mêmes langages.

* Machine de Turing à rubans multiples [5, p. 112] (à k rubans).

Fonction de transition : $\delta : Q\times\Gamma^k \to Q\times\Gamma^k\times\{\leftarrow;\rightarrow\}^k$

Configuration c : $c \in (\Gamma^*)^k \times Q \times (\varepsilon \cup \Gamma^*(\Gamma\setminus\{\#\}))^k$

Simulation par une machine à 1 ruban : les cases du $i$ème ruban sont représentées par les cases de position i modulo k.

* Machine de Turing à ruban bi-infini [5, p. 110]

Configuration c : $c \in (\varepsilon \cup (\Gamma\setminus\{\#\})\Gamma^*) \times Q \times (\varepsilon \cup \Gamma^*(\Gamma\setminus\{\#\}))$

Simulation par une machine à 2 rubans : on fixe une case d'indice 0. Le premier ruban représente la partie droite du ruban bi-infini et le second la partie gauche.

(*) Définition : Fonction calculable par une machine de Turing

* Machine de Turing non-déterministe [5, p. 114]

Relation de transition : $\delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{\leftarrow ; \rightarrow\})$

Configuration suivante : la machine choisit parmi l'ensemble des triplets obtenus par la relation de transition.

Langage accepté : Un mot w est accepté si il existe une suite de choix définissant une exécution acceptante. Le langage accepté est l'ensemble des mots acceptés.

Simulation par une machine déterministe à 3 rubans : le premier ruban contient l'entrée en lecture seule. Le second permet de lister les différentes suites de choix possibles (finies). Le troisième simule la machine non-déterministe en effectuant les choix indiqués par le second ruban.

B - Les autres modèles de calcul décident les mêmes langages.

* Fonctions récursives [5, p. 131]

Définition : Le langage $\mathcal{L}_{\mu R}$ des expressions des fonctions μ-récursives est défini par induction :

- 0 est une expression d'arité 0 ;
- σ est une expression d'arité 1 ;
- π_i^n où $n \in \mathbb{N}^*$ et $i \in \{1, \dots, n\}$ est d'arité n ;
- Si F est une expression d'arité n, et $G_1, \dots, G_n$ sont des expressions d'arités p, alors $o(F, G_1, \dots, G_n)$ est une expression d'arité p ;
- Si F est une expression d'arité n et G une expression d'arité $n+2$, alors $rec(F,G)$ est une expression d'arité $n+1$.
- Si F est une expression d'arité $p+1$, alors $mu(F)$ est une expression d'arité p.

Définition : Avec les notations précédentes, on définit la fonction $[\![\ ]\!] : \mathcal{L}_{\mu R} \to \bigcup_{k \in \mathbb{N}} \mathcal{F}_{partielle}(\mathbb{N}^k, \mathbb{N})$ par induction :

- $[\![0]\!] : / \longmapsto 0$
- $[\![\sigma]\!] : x \longmapsto x+1$
- $[\![\pi_i^n]\!] : (x_1, \dots, x_n) \mapsto x_i$
- $[\![o(F, G_1, \dots G_n)]\!] : \vec{x} \longmapsto [\![F]\!]([\![G_1]\!](\vec{x}), \dots, [\![G_n]\!](\vec{x}))$
- $[\![rec(F,G)]\!] = g$ où $g : \mathbb{N}^{n+1} \to \mathbb{N}$ et $(\vec{x}, k) \longmapsto \begin{cases} [\![F]\!](\vec{x}) & \text{si } k = 0 \\ [\![G]\!](\vec{x}, k-1, g(\vec{x}, k-1)) & \text{sinon} \end{cases}$
- $[\![mu(F)]\!] : \vec{x} \mapsto \mu i . [\![F]\!](\vec{x}, i)$

où $\mu i . [\![F]\!](\vec{x}, i) = \begin{cases} \text{le plus petit } i \in \mathbb{N} \text{ tq } [\![F]\!](\vec{x}, i) \neq 0 \\ \text{non défini si un tel } i \text{ n'existe pas} \end{cases}$

Définition : Une fonction $f : \mathbb{N}^k \to \mathbb{N}$ est une fonction μ-récursive si une expression de $\mathcal{L}_{\mu R}$ la dénote.

Exemple : $plus(n_1, n_2) = \begin{cases} \pi_1^2(n_1, n_2) & \text{si } n_2 = 0 \\ \sigma(\pi_3^3(n_1, n_2 - 1, plus(n_1, n_2 - 1))) & \text{sinon} \end{cases}$

Théorème : Les fonctions μ-récursives sont exactement celles calculables par une machine de Turing.

$\boxed{\text{DEV} \Leftarrow}$

Remarque : On peut redéfinir la thèse de Church avec des fonctions μ-récursives.

* λ-calcul. [2, p. 185].

Définition : L'ensemble L des termes du λ-calcul est le plus petit ensemble tel que :

* les variables $x, y, z, \dots$ sont des termes ;
* si u et v sont des termes, (uv) est un terme ;
* si x est une variable et t un terme, $\lambda x\ t$ est un terme.

Le terme (uv) représente l'application de la fonction u à l'argument v. Le terme $\lambda x\ t$ représente la fonction qui, à la variable x, associe le terme t : ce terme est obtenu par abstraction sur la variable x.

Exemple : Codage des entiers

- $0 = \lambda(f\, x) . x$
- $1 = \lambda(f\, x) . f x$
- $2 = \lambda(f\, x) . f(f(x))$
- $n = \lambda(f\, x) . \underbrace{f(f(\dots(f x)\dots))}_{f\ n \text{ fois}} = \lambda(f\, x) . f^n x$ avec f itérée n fois.

Ici f représente la fonction successeur.

Théorème : Les fonctions μ-récursives sont exactement celles qui sont représentables par un terme du λ-calcul.

Corollaire : Les fonctions calculables par une machine de Turing sont exactement celles qui sont représentables par un terme du λ-calcul.

C - La théorie de la calculabilité. [5, p. 139]

Définition : Une machine de Turing universelle peut simuler n'importe quelle machine de Turing. [5, p. 116]

Définition : RE = { L | L est un langage accepté par une machine de Turing }

R = { L | L est un langage décidé par une machine de Turing }

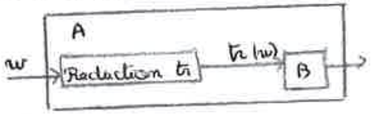
Problème : Arrêt

entrée : une machine de Turing M déterministe, un mot w.

sortie : oui si $M(w)$ s'arrête ; non, sinon.

Théorème : Le problème de l'Arrêt est dans RE et est indécidable.

Définition : Une réduction d'un problème A à un problème B est une fonction tr calculable telle que pour toute instance w : w est instance positive de A ssi $tr(w)$ est une instance positive de B. On dit que A se réduit à B si une telle réduction existe.



Théorème : Si A se réduit à B, alors :

* B décidable implique A décidable
* A indécidable implique B indécidable.

Problème : Pavage de Wang.

entrée : une famille(*) de tuiles de la forme , avec a, b, c, d appartenant à un ensemble fini.

sortie : oui si le jeu de tuiles permet de paver le plan ; non sinon.

(*) finie

Application : Le problème du pavage de Wang est indécidable.

Théorème [3, p. 162] (Rice). Pour toute propriété non triviale P sur les langages récursivement énumérables, le problème de savoir si le langage $L(M)$ d'une machine de Turing M vérifie P est indécidable.

III - Les machines de Turing classent les fonctions calculables

Définition : Un arbre de calcul depuis (ϵ, s, w) est l'arbre de racine (ϵ, s, w) tel que les fils de tout noeud C l'ensemble de ses configurations suivantes.

Définition : Soit $f : \mathbb{N} \to \mathbb{N}$, la machine de Turing M décide L en temps f si M décide L et $\forall w$, la hauteur de l'arbre de calcul depuis (ϵ, s, w) est $\leq f(|w|)$. On dit que M décide L en espace f si M décide L et $\forall w$, au plus $f(|w|)$ cases du ruban ont été utilisées.

Définition :

* $TIME(f) = \{ L \mid L$ est décidé par une machine de Turing déterministe en temps $O(f(n)) \}$.
* $NTIME(f) = \{ L \mid L$ est décidé par une machine de Turing non-déterministe en temps $O(f(n)) \}$.
* $SPACE(f) = \{ L \mid L$ est décidé par une machine de Turing déterministe en espace $O(f(n)) \}$
* $NSPACE(f) = \{ L \mid L$ est décidé par une machine de Turing non déterministe en espace $O(f(n)) \}$

Définition : - $P = \bigcup_k TIME(x \mapsto x^k)$ - $NP = \bigcup_k NTIME(x \mapsto x^k)$

- $PSPACE = \bigcup_k SPACE(x \mapsto x^k)$ - $NPSPACE = \bigcup_k NSPACE(x \mapsto x^k)$

Théorème : [3, p. 219] (Savitch). Soit $f : \mathbb{N} \to \mathbb{R}^+$ telle que $f(n) \geq n$, pour n assez grand. Toute machine de Turing déterministe qui décide en espace $f(n)$ est équivalente à une machine de Turing non-déterministe en espace $O(f^2(n))$. DEV

Corollaire : PSPACE = NPSPACE.

Définition : L est NP-dur ssi tout problème dans NP se réduit en temps polynomial à L. Si, de plus, L est dans NP, L est dit NP-complet.

Problème : SAT

entrée : φ une formule du calcul propositionnel

sortie : oui si φ est satisfiable ; non sinon.

Théorème [3, p. 203] (Cook). Le problème SAT est NP-complet.

Définition : * L = SPACE(log n)

* NL = NSPACE(log n)

Remarque : Pour définir une machine de Turing qui décide en espace logarithmique, il nous faut deux rubans : un en lecture seule et le second de travail borné avec l'entrée logarithmiquement.

Ouverture : On peut parler de la relation grammaire, machine de Turing via la hiérarchie de Chomsky.

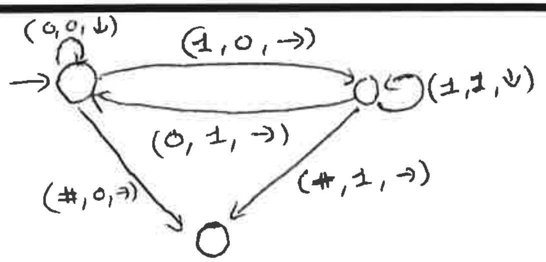


Figure 1 : Multiplication par 2.

Références :

[1] Girard, Turing : La machine de Turing

[2] Rougemont, Lassaigne, Logique et fondements de l'informatique. Logique du 1er ordre, calculabilité et λ-calcul.

[3] Carton, Langages formels

[4] Stern, Fondements mathématiques de l'informatique

[5] Wolper, Introduction à la calculabilité