Début sept 2014

Machines de Turing. Applications.

913

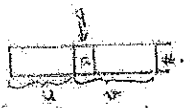
Motivation : Introduire un modèle de calcul représentant une abstraction des ordinateurs, autrement dit qui permette de calculer les fonctions calculables par procédure effective.

I. Machines de Turing

1) Définition [CAR] (Voir figure annexe pour un exemple)

Déf. 1 : Une machine de Turing est un septuplet $(Q, \Sigma, \Gamma, E, q_0, F, \#)$:
- $Q = \{q_0 \dots q_n\}$ est un ensemble fini d'états de contrôle.
- Σ est l'alphabet d'entrée (fini et ne contient pas $\#$) qui permet d'écrire la donnée initiale sur le ruban.
- Γ est l'alphabet de ruban (fini) : contient les symboles que l'on peut écrire sur le ruban, en particulier $\Sigma \cup \{\#\} \subset \Gamma$.
- E est un ensemble fini de transitions de la forme (p, a, q, b, x) où $p, q \in Q$, $a, b \in \Gamma$ et $x \in \{\triangleleft, \triangleright\}$. On la note $p, a \to q, b, x$.
- $q_0 \in Q$ est l'état initial.
- $F \subset Q$ est l'ensemble des états finaux.
- $\#$ est le symbole blanc, qui remplit au départ toutes les positions du ruban autres que celles contenant la donnée initiale.

Déf. 2 : Une configuration est l'état global de la machine à un instant donné, on la note $C = uqv$ où :



- $q \in Q$ est l'état de contrôle courant
- $w = uv \in \Gamma^*$ est le contenu du ruban, u est le contenu strictement à gauche de la tête de lecture et v le contenu à droite (les $\#$ ne sont pas écrits).

Déf. 3 : Une étape de calcul est une paire (C, C') de configurations notée $C \to C'$ telle que :
- Soit $C = ucpav$, $C' = uqcbv$ et $p, a \to q, b, \triangleleft \in E$
- Soit $C = upav$, $C' = ubqv$ et $p, a \to q, b, \triangleright \in E$.

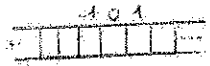
Déf. 4 : Un calcul est une suite de configurations successives $C_0 \to C_1 \to \dots \to C_k$. Il est acceptant si $C_0 = q_0 w$, $w \in \Sigma^*$ et $C_k = uqv$ avec $q \in F$

Déf. 5 : Le langage accepté par une machine M est :
$L(M) = \{w \in \Sigma^* /$ il existe un calcul acceptant partant de $C_0 = q_0 w\}$
Si de plus M est sans calcul infini, on dit qu'elle décide $L(M)$.

2) Variantes [CAR]

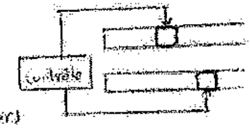
Déf. 6 : Deux machines M et M' sont dites équivalentes si $L(M) = L(M')$.

- Ruban bi-infini :



Prop. 7 : Toute machine à ruban bi-infini est équivalente à une machine classique.

- Machines à plusieurs rubans :



Prop. 8 : Toute machine à plusieurs rubans est équivalente à une machine classique.

- Machine déterministe : Une machine est déterministe si pour tout $(p, a) \in Q \times \Gamma$ il existe au plus un triplet $(q, b, x) \in Q \times \Gamma \times \{\triangleleft, \triangleright\}$ tel que $p, a \to q, b, x \in E$.

Prop. 9 : Toute machine M est équivalente à une machine déterministe M'. Si M est sans calcul infini, alors M' non plus.

II. Fonctions calculables [CAR]

Déf. 10 : Une fonction $f : \Sigma^* \to \Gamma^*$ est dite calculable s'il existe une machine de Turing qui pour toute entrée w s'arrête avec $f(w)$ écrit sur le ruban.

Prop. 11 : Les fonctions calculables sont exactement les fonctions μ-récursives. } DVPMT 1 (calculable $\Rightarrow$ récursive)

Thèse de Turing-Church : Tous les modèles de calcul sont équivalents ou plus faibles à celui des machines de Turing.

III. Décidabilité

1) Classes de décidabilité [CAR, WOL]

Soit Σ un alphabet fixe…

Def. 12 : R est l'ensemble des langages de Σ^* décidés par MT
RE est l'ensemble des langages de Σ^* acceptés par MT.
$co\text{-}RE = \{L \subseteq \Sigma^* / \overline{L} \in RE\}$

Prop. 13 : • $R \subseteq RE \cap co\text{-}RE$
• $L, L' \in R \Rightarrow L \cup L', L \cap L', \overline{L} \in R$
• $L, L' \in RE \Rightarrow L \cup L', L \cap L' \in RE$
• $L \in RE$ et $\overline{L} \in RE \Rightarrow L \in R$.

Prop. 14 : $L \in RE \Leftrightarrow$ il existe une MT déterministe appelée énumérateur qui écrit sur le ruban tous les mots de Σ^* séparés par un symbole $\$ \notin \Sigma$.

Prop. 15 : $A \subseteq \mathbb{N}^k$ est dans $RE \Leftrightarrow A$ est l'image d'une fonction μ-récursive. $A \in R \Leftrightarrow \mathbb{1}_A$ est μ-récursive

Prop. 16 [WOL] : $L \in RE \Leftrightarrow L$ est généré par une grammaire générale (de type 0)

2) Problèmes indécidables [CAR, WOL]

Def. 17 : Pour une MT M et $w \in \Sigma^*$ on note $\langle M, w \rangle$ un codage de la paire (M, w). On définit $LU = \{\langle M, w \rangle / w \in L(M)\}$.

Prop. 18 : $LU \in RE \setminus R$, par conséquent $\overline{LU} \notin RE$.

Def. 19 : Soient A et B deux problèmes de langages respectifs L_A et L_B définis sur des alphabets Σ_A et Σ_B. Une réduction de A à B est une fonction $f : \Sigma_A^* \to \Sigma_B^*$ calculable telle que $w \in L_A \Leftrightarrow f(w) \in L_B$. On note $A \leq_m B$.

Prop. 20 : Si $A \leq_m B$ et $B \in R$, alors $A \in R$.
Si $A \leq_m B$ et $A \notin R$, alors $B \notin R$.

Exemples de problèmes indécidables [WOL] : • Correspondance de Post (PCP)
• problème de l'arrêt : $L = \{\langle M, w \rangle / M$ s'arrête sur $w\}$
• arrêt sur mot vide, arrêt existentiel, arrêt universel
• langage accepté vide : $L_\emptyset = \{\langle M \rangle / L(M) = \emptyset\}$

Th. 21 (de Rice) : Pour tout $\mathcal{P} \subseteq RE$, $\mathcal{P} \notin \{RE, \emptyset\}$, le langage $\{\langle M \rangle / L(M) \in \mathcal{P}\}$ est indécidable. } DVPMT 2

Ex : • $L = \{\langle M \rangle / L(M) = L(M)^*$ (mots miroirs de $L(M)$)$\} \notin R$
• Sur un langage de programmation, l'ensemble des programmes qui comportent une division par 0 est indécidable.

IV. Complexité

1) Complexité temporelle [CAR]

Def. 22 : Soit M une MT. On note $t_M(w)$ pour $w \in \Sigma^*$ la longueur de la plus longue exécution finie de M sur w. On appelle complexité en temps de M, notée $t_M(n)$ ($n \in \mathbb{N}$) : $t_M(n) = \max_{|w| = n} t_M(w)$.

Def. 23 : Pour $f : \mathbb{N} \to \mathbb{R}^+$ on définit les classes :
$TIME(f(n)) = \{L \subseteq \Sigma^* / \exists M$ MT déterministe, M décide L et $t_M(n) = O(f(n))\}$
$NTIME(f(n)) = \{L \subseteq \Sigma^* / \exists M$ MT non déterministe, M décide L et $t_M(n) = O(f(n))\}$

Def. 24 : $P = \bigcup_{k \geq 0} TIME(n^k)$, $NP = \bigcup_{k \geq 0} NTIME(n^k)$

$EXPTIME = \bigcup_{k \geq 0} TIME(2^{n^k})$ $NEXPTIME = \bigcup_{k \geq 0} NTIME(2^{n^k})$

Pour une classe C, $co\text{-}C = \{L \mid \bar{L} \in C\}$.

Def. 25 : Un vérificateur en temps polynomial d'un langage L est une MT déterministe M prenant des entrées de la forme $\langle w, c \rangle$, de complexité temporelle polynomiale en $|w|$ et telle que $L = \{w \mid \exists c, \langle w, c\rangle \in L(M)\}$

Prop. 26 : $L \in NP \iff L$ admet un vérificateur en temps polynomial

Ex : $SAT \in NP$: un vérificateur prend $\langle \varphi, \nu \rangle$, φ formule, ν valuation, et accepte $\langle \varphi, \nu \rangle$ si et seulement si $\nu \models \varphi$.

Rem : $P = NP$ est un problème ouvert.

Rem : La classe P des MT correspond à la classe P des ordinateurs (machines RAM). [voir WOL, PAP]

2) NP-complétude [CAR, COR, ALG]

Def. 27 : Une réduction polynomiale de A vers B est une réduction f calculable en temps polynomiale. On note $A \leq_p B$.

Def. 28 : L est dit NP-difficile si : $\forall L' \in NP$ $L' \leq_p L$.
L est dit NP-complet si L est NP et NP-difficile.

Th. 29 (Cook) : SAT est NP-complet.

Prop. 30 : Si A est NP-difficile et $A \leq_p B$, B est NP-difficile.

Exemples de problèmes NP-complets : [COR, ALG]
3-SAT, clique, couverture de sommets, chemin hamiltonien, voyageur de commerce, sac à dos.

3) Complexité spatiale [CAR]

Def. 31 : Pour une MT M et $w \in \Sigma^*$, on note $S_M(w)$ le nombre maximal de cases de ruban utilisées lors d'une exécution finie de M sur w.

On appelle complexité spatiale de M, $S_M(n) = \max_{|w| \leq n} S_M(w)$ ($n \in \mathbb{N}$).

Def. 32 : Pour $f : \mathbb{N} \to \mathbb{R}^+$ on définit $SPACE(f(n))$, $NSPACE(f(n))$ de la même manière que $TIME(f(n))$, $NTIME(f(n))$, ainsi que PSPACE, NPSPACE, EXPSPACE, NEXPSPACE.

Prop. 33 : $S_M(n) \leq \max(t_M(n), n)$ et $t_M(n) \leq 2^{K \cdot S_M(n)}$ (où $K > 0$ ne dépend que de M)

Cor. 34 : $PSPACE \subset EXPTIME$, $NP \subset NPSPACE$.

Th. 35 (Savitch) Soit $s : \mathbb{N} \to \mathbb{R}^+$, $s(n) \geq n$ pour n assez grand. Toute MT non déterministe M telle que $S_M(n) = O(s(n))$ est équivalente à une MT déterministe M' telle que $S_{M'}(n) = O(s^2(n))$.

Cor. 36 : PSPACE = NPSPACE, EXPSPACE = NEXPSPACE.

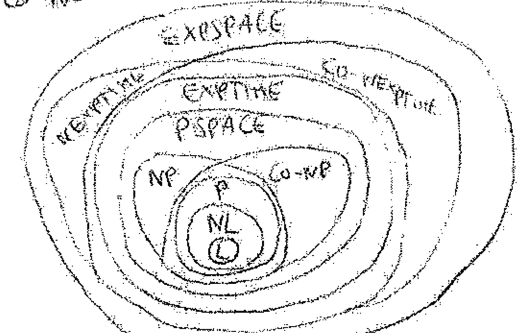
Ex : PATH, QSAT $\in$ PSPACE

Def. 37 : $L = SPACE(\log n)$, $NL = NSPACE(\log n)$, en ne considérant que des MT à deux rubans, un pour écrire l'entrée et un pour les calculs.

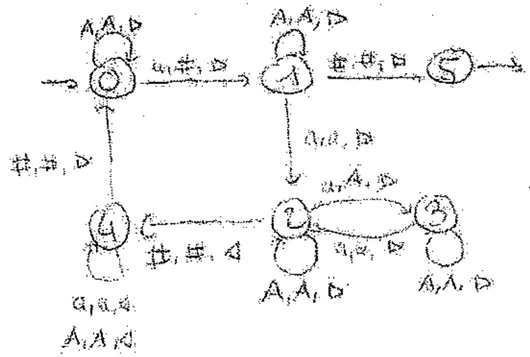
Ex : 2-SAT $\in$ NL.

Prop. 38 : NL = co-NL.

Conclusion :



Annexe : exemple de MT sur l'alphabet $\Sigma = \{a\}$ qui a pour langage $L = \{w \in \Sigma^* \mid |w| \text{ est une puissance de } 2\}$



$\Gamma = \{a, A, \#\}$

$Q = \{0, 1, 2, 3, 4, 5\}$

$q_0 = 0$

$F = \{5\}$

Exécution sur $w = aaaa$: 0aaaa# → #1aaa# → #a2aa# → #aA3a# → #aAa2# → #aA4a# → #a4Aa#
→ #a4Aa# → #4aAa# → 4#aAa# → #0aAa# → ##1Aa# → ##A1a# → ##Aa2#
→ ##A4a# → ##4Aa# → #4#Aa# → #0Aa# → ##A0a# → ##A#1# → ##A##5

Références : [CAR] Carton, Langages formels, Calculabilité et Complexité
[WOL] Wolper, Introduction à la calculabilité
[COR] Cormen, Algorithmique
[ALG] Dasgupta, Papadimitriou, Vazirani, Algorithms
[PAP] Papadimitriou, Computational Complexity

Autres développements possibles : Th. de Cook, Th. de Savitch