913. MACHINES DE TURING. APPLICATIONS

I/ Machines de Turing

I/1) Définition et formalisation.

def 1.1 Une machine de Turing est un septuplet $(Q, \Sigma, \Gamma, E, q_0, F, \#)$ avec :
- $Q = \{q_0, \dots, q_m\}$ est un ensemble fini d'états.
- Σ est un alphabet fini d'entrée. Il ne contient pas $\#$
- Γ est l'alphabet (fini) de ruban. Il contient Σ et $\#$
- E est un ensemble fini de transitions $p, a \to q, b, x$ où $p, q \in Q$; $a, b \in \Gamma$ et $x \in \{\rightarrow, \leftarrow\}$ est un sens de déplacement de la tête de lecture
- $q_0 \in Q$ est l'état initial de la machine.
- $F \subset Q$ est l'ensemble des états finaux
- $\#$ est le symbole blanc. Il appartient à Γ

rq 1.2 Le ruban d'une telle machine est une suite de cases numérotées par les entiers naturels. Initialement, ce ruban contient le mot d'entrée suivi d'une infinité de symboles blancs et la tête de lecture est sur la case 0.

def 1.3 Une machine de Turing est dite déterministe si pour tous $q \in Q, a \in \Gamma$, il existe au plus une transition de la forme $p, a \to q, b, x$. Dans ce cas, on peut voir l'ensemble E comme une fonction notée δ et nommée fonction de transition.

def 1.4 Une configuration d'une machine de Turing est un triplet uqv où u est le contenu du ruban strictement à gauche de la tête de lecture, v est le contenu à droite de la tête, le premier symbole de v est sous la tête de lecture et q est l'état de la machine. Les symboles blancs à droite de v sont omis

ex 1.5 Exemple de configuration : annexe A

def 1.6 On nomme étape de calcul (ou dérivation) une paire C, C' de configurations, notée $C \to C'$ telle que
- soit $C = ucpav$ et $C' = uqcbv$ et $p, a \to q, b, \leftarrow$ est une transition
- soit $C = upav$ et $C' = ubqv$ et $p, a \to q, b, \rightarrow$ est une transition

def 1.6 On nomme calcul une suite de configurations successives $C_0 \to C_1 \to \dots \to C_k$. Un calcul est dit acceptant si la configuration C_0 est initiale, càd $C_0 = q_0 w$ où $w \in \Sigma^*$, et C_k est finale, càd $C_k = uqv$ où $q \in F$

def 1.7 On dit que $w \in \Sigma^*$ est accepté par une machine de Turing M si il existe un calcul acceptant de configuration initiale $q_0 w$. L'ensemble des mots acceptés par M est le langage accepté par M, noté $L(M)$.

ex 1.8 Machine de Turing acceptant $\{a^{(2^n)} \mid n \in \mathbb{N}\}$ sur $\Sigma = \{a\}$. Annexe B

def 1.9 Le langage L est décidé par la machine de Turing M si $L = L(M)$ et si pour tout $w \in \Sigma^*$ et tout calcul de configuration initiale $q_0 w$, M s'arrête

I/2) Extensions

prop 1.10 (Normalisation) Pour toute machine de Turing M, il existe une machine de Turing M' telle que
- $L(M) = L(M')$
- M' a deux états q_+ et q_- tels que $F' = \{q_+\}$, M' se bloque toujours en q_+ et q_- et M' ne se bloque qu'en q_+ et q_-

def 1.11 Une machine à ruban bi-infini est formellement identique à une machine de Turing mais son ruban est indicé par $\mathbb{Z}$

prop 1.12. Pour toute machine M à ruban bi-infini, il existe une machine de Turing qui accepte le même langage que M.

def 1.13 Une machine à $k \geq 1$ rubans est une machine disposant de k rubans chacun lu par une tête de lecture indépendante pouvant éventuellement ne pas se déplacer lors d'une transition.

prop 1.14 Pour toute machine M à k rubans, il existe une machine de Turing qui accepte le même langage que M

prop 1.15 Pour toute machine de Turing M, il existe une machine de Turing déterministe qui accepte le même langage que M.

Thèse de Church-Turing Les langages décidés par une machine de Turing sont exactement ceux reconnus par une procédure effective (algorithme)

II/ Calculabilité

rq 2.1 On peut confondre un problème de décision (dont la réponse est oui ou non) avec le langage des codages des instances positives pour ce problème.

II/1) R et RE

déf 2.2 On nomme langage récursif / décidable tout langage décidé par une machine de Turing et on note R leur ensemble.

déf 2.3 On nomme langage récursivement énumérable tout langage accepté par une machine de Turing et on note RE leur ensemble.

déf 2.4 On nomme $co\text{-}RE = \{\Sigma^* \setminus L \text{ où } L \in RE\}$.

prop 2.5 $R = RE \cap co\text{-}RE$ (donc $R \subset RE$) cf annexe C

rq 2.6 Il existe des langages qui ne sont pas dans RE

déf 2.7 Un énumérateur est une machine de Turing déterministe qui écrit sur un ruban de sortie des mots de Σ^* séparés par le symbole $\# \notin \Sigma$. La tête de lecture de cette machine ne se déplace jamais vers la gauche.

prop 2.8 Un langage $L \subset \Sigma^*$ est récursivement énumérable ssi il existe un énumérateur qui énumère exactement les mots de L

prop 2.9 Soit $L, L' \subset \Sigma^*$ des langages

- Si L et $L' \in RE$ alors $L \cup L'$ et $L \cap L' \in RE$ aussi
- Si L et $L' \in R$ alors $L \cup L'$, $L \cap L'$ et $\Sigma^* \setminus L$ aussi

II/2) Décidabilité, indécidabilité

déf 2.10 Pour toute machine M et mot w, on note $\langle M \rangle$, $\langle w \rangle$ et $\langle M, w \rangle$ les codages respectifs de M, w et (M, w)

déf 2.11 On définit le langage d'acceptation L_ϵ par $L_\epsilon = \{\langle M, w \rangle \mid w \in L(M)\}$

prop 2.12 Le langage L_ϵ est récursivement énumérable

rq 2.13 Une machine de Turing qui accepte L_ϵ se nomme machine universelle

th 2.14 Le langage L_ϵ n'est pas décidable.

cor 2.15 Le complémentaire de L_ϵ n'est pas récursivement énumérable

rq 2.16 Ce langage L_ϵ est associé au problème de décision suivant :
entrée = une machine M, 1 mot w ; sortie = oui si $w \in L(M)$, non sinon.
Il est très proche du problème de l'arrêt pour lequel :
entrée = une machine M, 1 mot w ; sortie = oui si un calcul de M sur w s'arrête. Ce dernier problème est lui aussi dans RE mais pas dans R

déf 2.17 Une fonction $f : \Sigma^* \to \Gamma^*$ est calculable si il existe une machine de Turing qui pour toute entrée w s'arrête avec $f(w)$ écrit sur son ruban.

déf 2.18 Soit A et B deux problèmes d'alphabets respectifs Σ_A et Σ_B associés respectivement aux langages L_A et L_B. Une réduction de A à B est une fonction calculable $red : \Sigma_A^* \longrightarrow \Sigma_B^*$ telle que $w \in L_A \Leftrightarrow red(w) \in L_B$. On note $A \leq B$ si A se réduit à B

th 2.19 Si $A \leq B$ et B est décidable alors A est décidable.

cor 2.20 Si $A \leq B$ et A est indécidable alors B est indécidable.

th 2.21 (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 DEV 1

ex 2.23 Avec P = être vide, on obtient que le problème
entrée = M une machine de Turing
sortie = oui si L(M) est vide, non sinon
est indécidable.

III / Complexité en temps d'une machine de Turing

III/1) Modèle

déf 3.1 Soit M une machine de Turing et $\gamma : q_0 w \longrightarrow C_1 \longrightarrow \cdots \longrightarrow C_m$ un calcul de M d'entrée w. Le temps $t_M(\gamma)$ de ce calcul est m. La complexité en temps pour une entrée w est le plus grand temps de calcul ayant $q_0 w$ pour entrée : $t_M(w) = \max_{\gamma} t_M(\gamma)$. La complexité en temps de la machine M est $t_M : n \longmapsto \max_{|w|=n} t_M(w)$.

rq 3.2 On s'intéresse au comportement asymptotique de la fonction t_M

rq 3.3 Le modèle de machine de Turing qu'on considère influe sur sa complexité en temps. Ainsi :

prop 3.4 Toute machine de Turing M à k rubans est équivalente à une machine à 1 ruban M' telle que $t_{M'}(n) = O(t_M^2(n))$

prop 3.5 Toute machine de Turing M est équivalente à une machine de Turing déterministe M' telle que $t_{M'}(n) = O(2^{t_M(n)})$

déf 3.6 Pour toute fonction $f : \mathbb{N} \longrightarrow \mathbb{R}^+$, on définit les classes

- TIME$(f(n))$ comme l'ensemble des langages décidés par une machine de Turing déterministe (éventuellement à plusieurs rubans) en temps $O(f(n))$
- NTIME$(f(n))$ comme l'ensemble des langages décidés par une machine de Turing (éventuellement à plusieurs rubans) en temps $O(f(n))$

déf 3.7 On définit les classes de complexité suivantes :

$$P = \bigcup_{k \geq 0} \text{TIME}(n^k), \quad NP = \bigcup_{k \geq 0} \text{NTIME}(n^k)$$

$$EXP = \bigcup_{k \geq 0} \text{TIME}\left(2^{(n^k)}\right) \quad NEXP = \bigcup_{k \geq 0} \text{NTIME}\left(2^{(n^k)}\right)$$

rq 3.8. Dit autrement, un langage L est de P (resp. NP) s'il existe une machine de Turing déterministe (resp. quelconque) qui décide L et dont le temps de calcul est polynomial en la taille de l'entrée

ex 3.9. Les problèmes de connexité d'un graphe ou d'accessibilité dans un graphe sont dans P

ex 3.10. Le problème SAT est dans NP

déf 3.11 Un vérificateur en temps polynomial pour un langage L est une machine de Turing déterministe V qui accepte le langage L(V) tel que $L = \{w \mid \exists c ; V(w,c) \in L(V)\}$ en temps polynomial en la longueur de w

déf 3.12 Un tel c se nomme certificat.

prop 3.13 Un langage L est dans NP ssi il existe un vérificateur en temps polynomial pour L

III/2) NP-complétude

déf 3.14 Une réduction polynomiale du problème A au problème B est une réduction (cf déf 2.18) pour laquelle la fonction calculable f est calculable en temps polynomial par une machine déterministe on note $A \leq_p B$

prop 3.15 Si $A \leq_p B$ et $B \in P$ alors $A \in P$. Si $A \leq_p B$ et $A \in NP$ alors $B \in NP$

déf 3.16 Un problème A est dit NP-dur si tout problème $B \in NP$ se réduit polynomialement à A, càd $B \leq_p A$

prop 3.17 Si A est NP-dur et $A \leq_p B$ alors B est NP-dur

déf 3.18 Un problème A est dit NP-complet s'il est NP-dur et dans NP

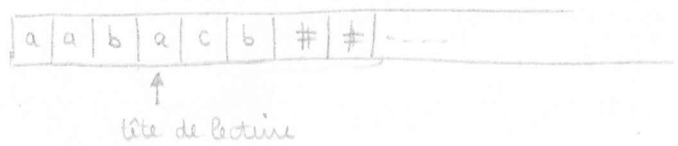
th 3.19 (Cook) SAT et 3SAT sont NP-complets DEV 2

rq 2SAT ∈ P

justifier de ne pas parler de la complexité en espace car elle est majorée par celle en temps (il faut le temps d'écrire)

Annexe A

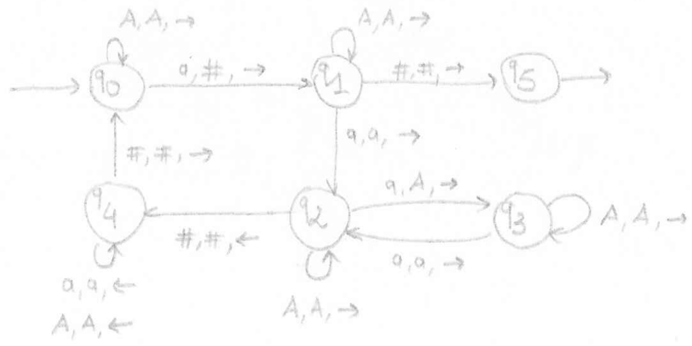
La configuration $aabq_1acb$ correspond à un ruban dans l'état suivant :



On ajoute à cette information le fait que la machine est dans l'état q_1

Annexe B

. On représente cette machine par un graphe



Le mot aa est accepté car le calcul

$q_0aa \rightarrow \#q_1a \rightarrow \#aq_2\# \rightarrow \#q_4a \rightarrow q_4\#a$

$\rightarrow \#q_0a \rightarrow \#\#q_1\# \rightarrow \#\#\#q_5$ est acceptant

Annexe C