913 Machines de Turing . Applications

But : Formaliser la notion de calcul par procédure effective.

I Machines de Turing

1) Définitions [WOL]

Description :

- ruban avec tête de lecture
- ensemble fini d'états

définition : Une machine de Turing (MT) est un septuplet

$$M = (Q, \Gamma, \Sigma, \delta, i, B, F)$$

avec :
- Q ensemble fini d'états
- Γ alphabet de ruban
- $\Sigma \subseteq \Gamma$ alphabet d'entrée
- $i \in Q$, état initial
- $F \subseteq Q$, ensemble des états accepteurs
- $B \in \Gamma \setminus \Sigma$, symbole "blanc"
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ fonction partielle de transition

Exemple : voir Annexe

Définition : Une configuration $(q, \alpha_1, \alpha_2) \in Q \times \Gamma^* \times (\Gamma^* \Gamma \setminus \{B\} \cup \varepsilon)$

q : état courant

α_1 : mot avant la tête de lecture

α_2 : mot entre la tête de lecture et l'infinité de blanc

Exécutions possibles : 3 cas :
- La machine s'arrête sur un état accepteur
- La machine bloque
- L'exécution est infinie

définition : Langage accepté par M

$L(M) = \{ w \in \Sigma^* \ / \ \text{l'exécution de } M \text{ sur } w \text{ s'arrête dans un état accepteur} \}$

définition : Langage décidé par M

M décide $L \subset \Sigma^*$ si M accepte L et M n'a pas d'exécutions infinies

Exemple : Dans le premier exemple $L(M) = a^n b^n$

2) Extensions [WOL]

- Ruban infini dans les deux sens :

prop : Tout langage accepté ou décidé par une MT à ruban infini dans les deux sens l'est par une MT classique.

- MT à deux rubans (et deux têtes de lecture) :

prop : Tout langage accepté ou décidé par une MT à deux rubans l'est par une MT classique.

- MT non déterministe :

définition : Une MT non déterministe est un septuplet $M = (Q, \Gamma, \Sigma, \Delta, i, B, F)$ avec $Q, \Gamma, \Sigma, i, B, F$ comme les MT et $\Delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{L, R\})$

prop : Tout langage accepté ~~ou~~ décidé par une MT non déterministe l'est par une MT déterministe

- MT universelle :

prop : Il existe une Machine de Turing universelle qui prend en entrée une description d'une MT, M et d'un mot w et qui simule l'exécution de M sur w.

II Thèse de Turing - Church

1) Fonctions Calculables [WOL]

définition : Une MT, M calcule $f : \Sigma^* \to \Sigma^*$ si pour tout $w \in \Sigma^*$ l'exécution de M sur w s'arrête en ayant $f(w)$ sur le ruban.

définition : Les fonctions μ-récursives sont les fonctions de $\mathbb{N}^k \to \mathbb{N}$ obtenues à partir des fonctions primitives récursives de base par : composition, récursion primitive et minimisation non bornée de prédicats sûrs

2) Thèse de Turing - Church [WOL]

Thèse : Les fonctions calculables par procédure effective sont les fonctions μ-récursives

prop : fonctions μ-récursives $\iff$ fonctions calculables par MT

III Calculabilité

1) Classes de décidabilité [CAR]

<u>définition</u> : R est l'ensemble des langages décidables par MT

<u>définition</u> : RE est l'ensemble des langages acceptés par MT

<u>prop</u> : $R \subseteq RE$

<u>proposition</u> : Si $L, L' \in R$ alors $L \cup L'$, $L \cap L'$ et $\bar{L} \in R$

<u>proposition</u> : Si $L, L' \in RE$ alors $L \cup L'$, $L \cap L' \in RE$

<u>proposition</u> : Si L et $\bar{L} \in RE$ alors L et $\bar{L} \in R$

2) Exemples de problèmes indécidables [WOL]

$(M_i)_{i \in \omega}$ énumération des MT sur un alphabet Σ

$(w_i)_{i \in \omega}$ énumération des mots de Σ^*

<u>Exemple</u> : $L_0 = \{ w \in \Sigma^* / \exists_i\, w = w_i$ et M_i n'accepte pas $w_i \}$

<u>prop</u> : $L_0 \notin RE$ et $\bar{L_0} \in RE$

<u>cor</u> : $R \subsetneq RE$

3) Réduction [CAR]

<u>définition</u> : On dit que $L_A \subset \Sigma_A^*$ se réduit à $L_B \subset \Sigma_B^*$ s'il existe une fonction calculable $f : \Sigma_A^* \longrightarrow \Sigma_B^*$ telle que $\forall w$, $w \in L_A \Leftrightarrow f(w) \in L_B$. On note $L_A \leq_m L_B$

<u>prop</u> : Soit L et L' tels que $L \leq_m L'$

alors $\begin{cases} L' \in R \Rightarrow L \in R \\ L \notin R \Rightarrow L' \notin R \end{cases}$

[WOL] <u>Exemple 1</u> : $LU = \{ \langle M, w \rangle / M$ une MT, $w \in \Sigma^*$ tels que M accepte $w \} \notin R$

[WOL] <u>Exemple 2</u> : Des problèmes d'arrêts indécidables :

- $H = \{ \langle M, w \rangle / M$ s'arrête sur $w \}$
- arrêt sur mot vide
- arrêt existentiel
- arrêt universel

<u>Exemple 3</u> : • Langage accepté vide

- $L_{\neq} = \{ \langle M, M' \rangle / L(M) \neq L(M') \}$

4) Théorème de Rice [WOL]

<u>Théorème</u> : (Rice) [DEV 1]

Toute propriété non triviale des langages récursivement énumérables est indécidable.

<u>Exemple</u> : Déterminer si $L(M) = \{$ mots miroirs de $L(M) \}$

IV Complexité

1) P et NP [CAR]

<u>définition</u> : Soit M une MT (déterministe ou non). Soit $w \in \Sigma^*$. On appelle temps de calcul de M sur w, notée $t_M(w)$ la longueur de la plus longue exécution finie de M sur w. 1 / si exe 00

<u>définition</u> : Soit M une MT, on appelle complexité en temps de M notée $t_M(n)$, la quantité :

$$t_M(n) = \max_{|w|=n} t_M(w)$$

<u>définition</u> : On dit que M une MT est polynomiale si il existe P polynôme tel que $\forall n$ $t_M(n) \leq P(n)$

<u>définition</u> : On définit la classe des langages P comme étant celle des langages décidés par une MT polynomiale déterministe

<u>Exemple</u> :

- le problème d'existence d'un chemin dans un graphe est P
- le problème de savoir si un mot appartient à un langage algébrique donné est un problème P.

<u>définition</u> : Soient $L_A \subseteq \Sigma_A^*$ et $L_B \subseteq \Sigma_B^*$. Une réduction polynomiale de L_A à L_B est une fonction $f : \Sigma_A^* \longrightarrow \Sigma_B^*$ calculable par MT en temps polynomial telle que :

$$\forall w \in \Sigma_A^*, \quad w \in L_A \Leftrightarrow f(w) \in L_B$$

Si une telle réduction existe, on notera alors $L_A \leq_p L_B$

prop : Soient $L_1 \leq_p L_2$. On a $L_2 \in P \Rightarrow L_1 \in P$

définition : On définit la classe des langages NP comme étant celle des langages acceptés par MT non déterministe en temps polynomial.

Exemple :

- Le problème de l'existence d'un circuit hamiltonien dans un graphe (ie d'un circuit passant par chaque noeud une et une seule fois) est NP.
- Le problème du voyageur de commerce est NP

définition : Soit L un langage. Un vérificateur en temps polynomial de L est une MT déterministe M qui prend en entrée des mots de la forme $\langle w, c \rangle$ et dont la complexité est polynomiale en $|w|$ et telle que $L = \{w \,/\, \exists c,\ \langle w,c\rangle \in L(M)\}$

prop : $L \in NP$ si et seulement si il existe un vérificateur en temps polynomial pour L

prop : $P \subseteq NP$

Rq : Le problème de savoir si $P = NP$ est ouvert

[WOL] **Rq** : Le cadre des machines de Turing a du sens en pratique car il est possible de simuler une machine RAM ~~à l'aide~~ d'une machine de Turing et vice-versa $\boxed{\text{polynomialement}}$.
La classe des langages P et NP pour machine RAM est donc la même que celle pour les MT.

2) NP-Complétude [CAR]

définition : Un langage L est dit NP-dur si $\forall L' \in NP$, on a $L' \leq_p L$

définition : Un langage L est dit NP-complet si : $L \in NP$ et L est NP-dur.

prop : Si il existe $L \in P$ et L NP-complet, alors $P = NP$

prop : Si L est NP-complet et $L \leq_p L'$, alors L' est NP-complet

Exemple : Le problème SAT est celui de savoir si une formule du calcul propositionnel est satisfiable

Théorème : (Cook-Levin) [DEV 2]
Le problème SAT est NP-complet

Exemple :

- 3-SAT est NP-complet
- 3-COL est NP-complet mais 2-COL $\in$ P
- Le problème de la couverture de sommets est NP-complet

3) PSPACE / NPSPACE [CAR]

définition : On appelle espace de calcul de M une MT sur w, notée $s_M(w)$ le nombre maximal de case utilisé par une exécution finie de M sur w.

définition : On définit la complexité en espace de M une MT, notée $s_M(n)$, la quantité : $s_M(n) = \max\limits_{|w|=n} s_M(w)$

définition : On définit la classe PSPACE comme étant celle des langages décidés par un MT ~~déte~~déterministe à complexité spatiale polynomiale.

définition : On définit la classe NPSPACE comme étant celle des langages acceptés par une MT non déterministe à complexité spatiale polynomiale.

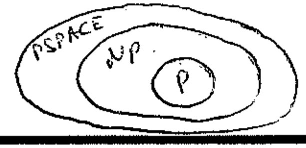
Exemple :

- Le problème de savoir si le langage accepté par un automate fini est Σ^* est PSPACE
- QSAT est PSPACE

Théorème : (Savitch) Soit $s : \mathbb{N} \to \mathbb{R}^+$, telle que $s(n) \geq n$ pour n assez grand
Toute machine non déterministe qui a une complexité en espace $o(s(n))$ est équivalente à une MT déterministe en espace $o(s^2(n))$

Cor : PSPACE = NPSPACE

Théorème : $NP \subseteq PSPACE$



Théorème de Rice

[WOLPER p 150 ou CARTON p 149]

On note $(w_i)_{i\in\mathbb{N}}$ et $(M_j)_{j\in\mathbb{N}}$ des énumérations des mots et des machine de Turing.

Définition $L_0 = \{ w \ / \ w = w_i$ et M_i n'accepte pas $w_i \}$
$LU = \{ \langle M, w\rangle \ / \ M$ accepte $w \}$

On notera $\overline{L_0}$ et $\overline{LU}$ les complémentaires.

> **Théorème de Rice**
> Toute propriété non triviale des langages récursivement énumerables est indécidable.

① Lemme. L_0 est indécidable. [WOLPER 142]

Preuve. Par l'absurde, supposons que $L_0 \in RE$
Alors L_0 est accepté par une machine de Turing notée M_k. Or :
- Si M_k accepte w_k, $w_k \notin L_0$ par définition de L_0 Absurde
- Si M_k n'accepte pas w_k, $w_k \in L_0$. Absurde.

② Lemme. $\overline{L_0}$ est indécidable [WOLPER 144]

Preuve. Si $\overline{L_0}$ était décidable alors L_0 le serait aussi. (Cf rappel)

> [PARENTHÈSE] Rappel
> Lemme Le complémentaire d'un langage de la classe R est un langage de la classe R.
> Preuve. Soit $L \in R$. Alors il existe une machine M qui décide L. On construit M' à partir de M en inversant les reponses données par M. M' décide $\overline{L}$.

③ Lemme. LU est indécidable. [WOLPER 146]

Preuve. On effectue une réduction à partir du langage $\overline{L_0}$
On doit trouver un algorithme qui décide $\overline{L_0}$ à l'aide d'une procédure qui déciderait le langage LU.

Supposons donc que LU est décidable. Considérons l'algorithme suivant, prenant en entrée un mot w.

→ déterminer i tq $w = w_i$

→ déterminer la MT M_i

→ applique la procédure de décision pour LU à $\langle M_i, w_i \rangle$.

→ si le résultat est positif, accepter w, sinon refuser w

Alors cet algorithme décide $\overline{L_0}$. Absurde.

Démonstration du théorème :

Considérons $L_P = \{\langle M \rangle / \mathcal{L}(M) \text{ vérifie } P\}$ le langage des codages des machines dont le langage vérifie P.

Quitte à remplacer P par sa négation, on peut supposer que le langage vide ne vérifie pas la propriété P.

Puisque, par hypothèse, P n'est pas triviale, il existe une machine de Turing M_P telle que $\mathcal{L}(M_P)$ vérifie P.

On effectue une réduction à partir de LU.

Soit $\langle M, w \rangle$ une instance de LU, on construit une machine M' qui a le comportement suivant.

- M' simule l'exécution de M sur w, sans tenir compte de son propre mot d'entrée x

- Si M accepte w, elle simule M_P sur x
- Si M n'accepte pas w, M' n'accepte aucun mot.

Cette réduction est correcte car :

Si M n'accepte pas w (ie $\langle M, w \rangle \notin LU$), alors $L(M') = \emptyset$ et ne satisfait pas la propriété P

Si M accepte w, alors $L(M') = L(M_P)$ qui satisfait P

Donc $\mathcal{L}(M')$ vérifie P ssi $\langle M, w \rangle \in LU$

L_P n'est pas décidable.

Remarque

Si tous les langages RE vérifient (ou ne vérifient pas) la propriété P, on construit facilement une MT qui accepte les codages $\langle M \rangle$ tq $L(M)$ vérifie P. il suffit d'accepter ou de rejeter tous les codages

Théorème de Cook

{WOLPER 184
{CARTON 189

Théorème Le problème SAT est NP-complet

Démonstration

★ SAT est dans NP

L'algorithme suivant est non déterministe polynomial et résoud le problème :

1. Générer de façon non déterministe une valuation
2. Vérifier que cette valuation rend la formule vraie.

★ Etablissons maintenant une réduction de n'importe quel problème NP à SAT

Soit A un problème NP, et soit M une machine de Turing non-déterministe qui décide A en temps polynomial.

OBJECTIF **Pour chaque entrée w de M, on va construire une instance de SAT, φ_w, de taille polynomiale en $|w|$ telle que φ_w est satisfaisable (ssi) w est acceptée par M.**

Notons $n = |w|$. Sans perte de généralité, on peut supposer que M accepte w (ssi) il existe une exécution de M sur w de longueur au plus n^k qui mène à un état accepteur. Une telle exécution peut être représentée par une suite de $n^k + 1$ configurations successives de la machine (quitte à rajouter des transitions inutiles lorsqu'on atteint une configuration finale en moins de n^k+1 étapes).

D'autre part, la machine M fonctionne en temps n^k, elle utilise $n^k + 1$ cellules du ruban puisqu'à chaque étape de l'exécution, la tête de lecture se déplace d'une seule case.

Tableau formé par les configurations (alphabet $\Gamma \cup Q = \Sigma$)

Config.	0	1	2	3	....	n^k+1
$C_0 =$	q_0	w_1	w_2	w_3	.....	#
$C_1 =$	w_1	q_1	w_2	w_3	...	#
$C_2 =$	w_1	w_2	q_2	w_3	.....	#
⋮						⋮
$C_{n^k} =$	....	...	....	.....		....

PRINCIPE **Le principe de la transformation vers SAT est de choisir un ensemble de variables propositionnelles telle qu'une valuation pour ces variables définisse le contenu du tableau.**

Définissions des variables propositionnelles

$\forall\ 0 \leq i \leq n^k + 1$,
$\forall\ 0 \leq j \leq n^k + 2$,
$\forall\ a \in \Sigma$

soit $x_{i,j,a}$ une variable qui code le fait que la case (i,j) contienne ou non le symbole a de Σ.

$\longrightarrow$ Le nombre de ces variables est $|\Sigma|(n^k+3)(n^k+2)$ (polynomial en n)

La formule φ_w se décompose en une conjonction :

$$\underbrace{\varphi_0}_{\substack{\text{code le fait que chaque} \\ \text{cellule contient un} \\ \text{unique symbole de } \Sigma.}} \wedge \underbrace{\varphi_1}_{\substack{\text{code que la } 1^{\text{er}} \\ \text{ligne est bien} \\ q_0 w \text{ (la configuration} \\ \text{initiale)}}} \wedge \underbrace{\varphi_2}_{\substack{\text{chaque ligne est} \\ \text{obtenue en appliquant} \\ \text{une transition de } M}} \wedge \underbrace{\varphi_3}_{\substack{\text{code que le calcul} \\ \text{est bien acceptant.}}}$$

φ_0 : garantit qu'au moins une des variables $x_{i,j,a}$ a la valeur 1 pour un $a \in \Sigma$ mais que $x_{i,j,a}$ et $x_{i,j,b}$ pour $a \neq b$ ne peuvent avoir la valeur 1 simultanément.

$$\varphi_0 = \bigwedge_{i,j} \left[\left(\bigvee_{a\in\Sigma} x_{i,j,a} \right) \wedge \left(\bigwedge_{\substack{a,a'\in\Sigma \\ a\neq a'}} \left(\neg x_{ija} \vee \neg x_{ija'} \right) \right) \right] \rightsquigarrow \boxed{XOR}$$

φ_1 : si on note $w = w_1 \dots w_n$.

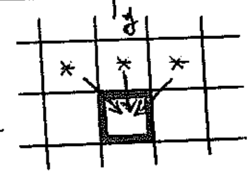
$$\varphi_1 = x_{0,0,q_0} \wedge \left(\bigwedge_{1\le k\le n} x_{0,k,w_k} \right) \wedge \left(\bigwedge_{n+1\le k\le n^k+1} x_{0,k,\#} \right)$$

φ_3 . au moins une des cases de la dernière ligne du tableau contient un état final.

$$\varphi_3 = \bigvee_{q\in F} \left(\bigvee_{0\le j\le n^k+1} x_{n^k,j,q} \right)$$

φ_2 : assure que chaque ligne est obtenue en appliquant une transition valide de M. on ne va pas chercher à l'expliciter, seulement à montrer que sa taille est polynomiale en n

Remarque.



la valeur d'une case (i, j) ne dépend que des trois cases $(i-1, j-1)$, $(i-1, j)$ et $(i-1, j+1)$ et de la transition effectuée par la machine pour passer de C_{i-1} à C_i

- Si dans ces trois cases, il n'y a que des symboles de bande, alors le contenu de la case (i, j) est le même qu'en $(i-1, j)$.
- Si l'état de la configuration C_{i-1} se trouve dans la case $(i-1, j)$ alors l'état de la configuration de C_i se trouve en $(i, j-1)$ ou $(i, j+1)$ suivant que la tête de lecture se déplace à gauche ou à droite.
- Si l'état de la configuration C_{i-1} se trouve dans la case $(i-1, j-1)$ ou $(i-1, j+1)$, le contenu de la case (i, j) dépend également du déplacement de la tête de lecture.

↳ Il suffit de regarder toutes les "fenêtres" de taille 2×3 du tableau.
Le nombre de tous les contenus possibles pour une telle fenêtre ne dépend que de $|\Sigma|$ et des transitions de M : ça ne dépend pas de n.
Le fait que 6 cases correspondent, s'écrit comme une conjonction de 6 variables $x_{i,j,a}$.
Le fait que toutes les parties de six cases du tableau correspondent à un des contenus possibles s'écrit comme une conjonction sur i, j d'une disjonction sur les différents contenus.
→ polynomial en n.