

Chapitre 5

(Chapitre 3 de Sipser)

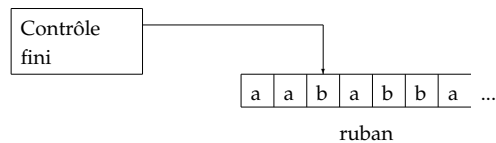
Church, Turing et compagnie

Motivation

- description de langages plus intéressants que les langages hors-contexte
- retour sur les fonctions calculables
- hypothèse de Church-Turing
- prélude à la théorie de la complexité

La machine de Turing (MT)

Du catalogue *Future Shop* de 1936 :



- Contrairement à l'AFD,
- la MT peut lire et écrire
 - sa tête est bidirectionnelle
 - son ruban est infini vers la droite.

Définition 5.1. (Sipser 3.3)

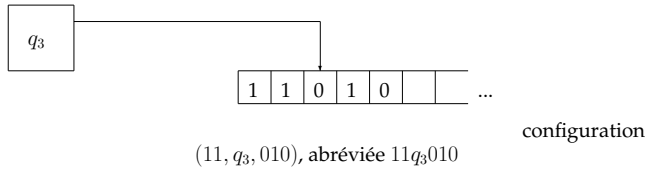
Une MT est un 7-tuplet $(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ où

- Q, Σ et q_0 sont comme pour l'AFD,
- q_a est l'état final acceptant,
- q_r est l'état final rejetant,
- Γ est l'alphabet de ruban,
 $\sqcup \in \Gamma$ et $\Sigma \subseteq \Gamma$,
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
est la fonction de transition.

▲

Définition 5.2. Une **configuration** d'une MT $(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ est un **triplet** (u, q, v) avec $q \in Q$ et $u, v \in \Gamma^*$. ▲

Indique que le préfixe pertinent du ruban est uv et que la tête pointe au premier symbole de v :



Remarque 5.3. La configuration initiale d'une telle MT sur entrée $w \in \Sigma^*$ est

$$(\varepsilon, q_0, w)$$

qu'on abrégie q_0w . ▲

Transitions d'une MT

La configuration $C = (u, q, av')$ mène à la configuration $C' = (ub, q', v')$ si $\delta(q, a) = (q', b, R)$.

La configuration $C = (u'd, q, av')$ mène à la configuration $C' = (u', q', dbv')$ si $\delta(q, a) = (q', b, L)$.

On dénote la **transition** de C à C' par $C \Rightarrow C'$.

On écrit $C \xRightarrow{*} C'$ si C mène à C' en 0 ou plus transitions successives.

Trois comportements possibles sur w

Définition 5.4. On dit que la MT

- **accepte** w ssi $(\varepsilon, q_0, w) \xrightarrow{*} (u, q_a, v)$,
- **rejette** w ssi $(\varepsilon, q_0, w) \xrightarrow{*} (u, q_r, v)$,
- **boucle** sur w ssi ni l'un ni l'autre des cas ci-dessus ne s'applique.

Remarque 5.5. La MT "rebondit" si sa tête de lecture tente de quitter le ruban. ▲

Définition 5.6. (Sipser 3.6)

Un langage Y est **décidable** s'il existe une MT qui, sur toute entrée w ,

- **accepte** si w est dans Y et
- **rejette** si w n'est pas dans Y . ■

Définition 5.7. (Sipser 3.5)

Un langage Y est **reconnaissable** s'il existe une MT qui, sur toute entrée w ,

- **accepte** si w est dans Y et
- **rejette ou boucle** si w n'est pas dans Y .

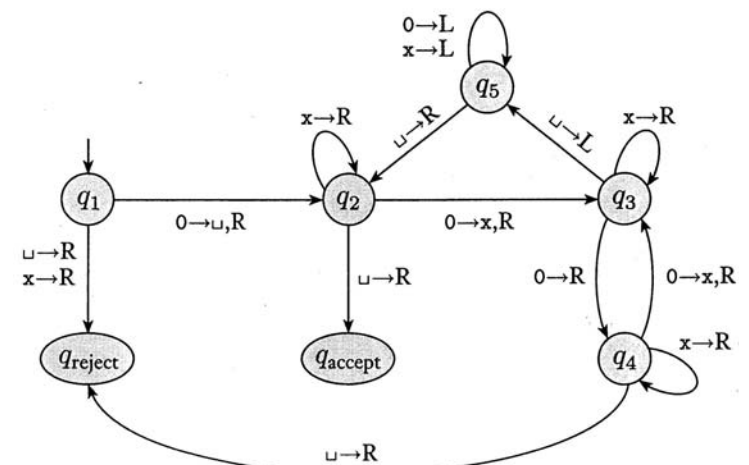
Exemple 5.8. (Sipser 3.7)

Le langage $Y = \{0^{2^n} : n \geq 0\} \subseteq 0^*$ est décidable.

Stratégie de haut niveau :

1. Si on ne trouve qu'un 0, accepter. ■
2. Si le nombre de 0s est impair, rejeter. ■
3. Effacer la moitié des 0s.
4. Retourner à l'étape 1.

Diagramme de la MT décidant $\{0^{2^n}\}$



Définition formelle

$(Q, \{0\}, \{0, \sqcup, x\}, \delta, q_1, q_a, q_r)$ où
 $Q = \{q_1, q_2, q_3, q_4, q_5, q_a, q_r\}$ et

δ	0	$\sqcup$	x
q_1			
q_2	(q_3, x, R)	$(q_a, \sqcup, R)$	(q_2, x, R)
q_3			
q_4			
q_5			
q_a	--	--	--
q_r	--	--	--



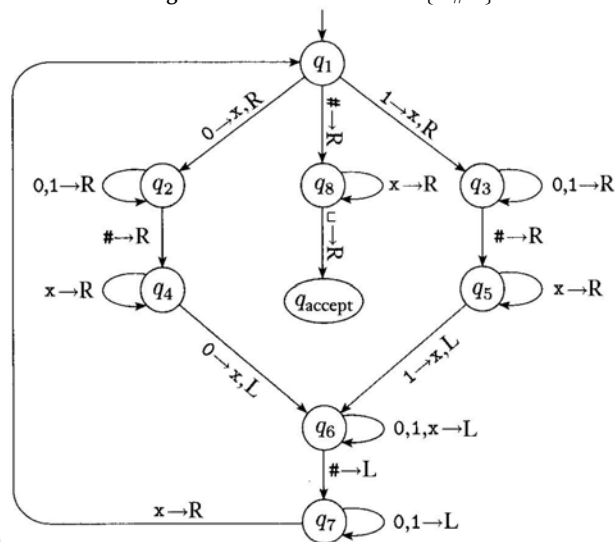
Exemple 5.9. (Sipser 3.9)

$Y = \{w\#w \mid w \in \{0, 1\}^*\}$ est décidable. ■

Stratégie :

1. Parcourir l'entrée afin de s'assurer qu'elle contient un seul #. Sinon, rejeter.
2. Vérifier que les positions ultimes de part et d'autre du # contiennent le même symbole. Si oui, les changer pour un x ; sinon, rejeter.
3. S'il ne reste plus que des x du côté gauche, vérifier s'il reste des symboles (autres que x) à droite. Si oui, rejeter ; sinon, accepter.

Diagramme de la MT décidant $\{w\#w\}$



Simulateurs de machines de Turing

Plusieurs amateurs se sont amusés à concevoir des interpréteurs graphiques de "programmes" de MTs, tel celui qui se trouve à <http://www.ironphoenix.org/tril/tm/>.

Techniques de construction de machines de Turing

- marquer
ex : $abbabc \rightarrow Abbabc$

- décaler
ex : $w \rightarrow \sqcup w$

- mémoriser dans les états
ex : le # dans $w\#w$, ou encore

$$\begin{aligned} qabx_1 \dots x_k &\Rightarrow \sqcup q_a b x_1 \dots x_k \\ &\Rightarrow \sqcup \sqcup q_{ab} x_1 \dots x_k \\ &\stackrel{*}{\Rightarrow} \sqcup \sqcup x_1 \dots x_k abq' \end{aligned}$$

- amalgamer plusieurs symboles : "pistes"

ex : nouveaux symboles de ruban de type

b
a
a

- utiliser des sous-routines
ex : expansion $abc \rightarrow a\sqcup b\sqcup c$
ex : multiplication unaire $1^n \# 1^m \rightarrow 1^{nm}$
ex : addition binaire $101 \# 110 \rightarrow 1011$



(<http://www-gap.dcs.st-and.ac.uk/history/PictDisplay/Turing.html>)

Scientifique, logicien, cerveau versus machine, cryptanalyste, athlète, inventeur de l'ordinateur, pionnier de l'IA, gai déclaré, etc.

Alan Mathison Turing, 1912-1954

Variante bi-infinie de la MT

Définition 5.10. Une **MT bi-infinie** est une MT dont le ruban est infini dans les deux sens. ▲

La configuration de départ sur entrée w est toujours $(\varepsilon, q_{\text{initiale}}, w)$.

Tout le reste demeure.

Proposition 5.11. Y est décidé (reconnu) par une MT M_2 bi-infinie ssi Y est décidé (reconnu) par une MT M_1 . ■

Aperçu de la preuve.

Si : (construire M_2 bi-infinie simulant MT M_1) ■

- marquer la case précédant w au départ
- faire comme M_1 par la suite
- mais simuler le rebond si la case marquée est revisitée.

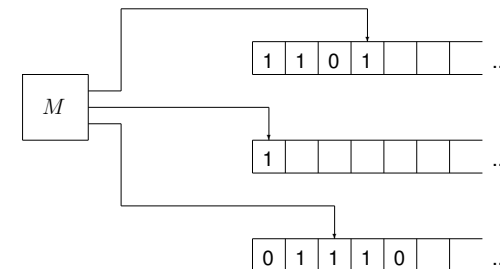
Seulement si : (construire MT M_1 simulant MT bi-infinie) ■

- marquer le début du ruban, et
- créer deux pistes virtuelles pour les deux moitiés du ruban bi-infini.



Variante multi-rubans

Définition 5.12. Une **MT à k rubans** est comme une MT sauf $\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, S, R\}^k$.



Dans la MT à k rubans,

- chaque ruban a une tête de lecture/écriture,
- chaque tête bouge indépendamment des autres,
- au départ, l'entrée se trouve sur le premier ruban et les autres rubans sont vides.

Théorème 5.13. (Sipser 3.13)

Si Y est décidé (reconnu) par une MT à k rubans M alors Y est décidé (reconnu) par une MT M' .

Aperçu de la preuve. M' utilise $2k$ "pistes" pour représenter une configuration de M .

Plus en détail, contenu typique du ruban de M' :

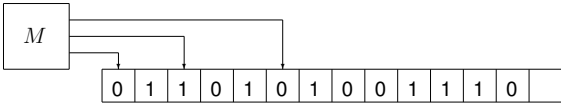
Ruban 5										X		
Lecteur 5	a	b	a	a	a	a	a	b	b	...	...	
Ruban 4						X						
Lecteur 4	b	a	b	a	b	b	b	a	b	...	...	
Ruban 3						X						
Lecteur 3	b	a	a	a	a	a	b	b	a	...	...	
Ruban 2				X								
Lecteur 2	a	b	b	a	a	b	a	b	b	...	...	
Ruban 1		X										
Lecteur 1	b	b	a	a	b	a	a	b	b	...	...	

Pour simuler un pas de la MT à k rubans :

1. M' lit de gauche à droite et mémorise les symboles sous les k têtes de M (états de M' de la forme $\langle q, m, \sigma_1, \dots, \sigma_k \rangle$: “ M est dans état q , m têtes vues, σ_i sous tête i ”)■
2. M' revient en simulant l’action des têtes de M (états requis pour ajuster la représentation d’une tête à la fois)■
3. M' mémorise le nouvel état de M . ▲

Variante à k têtes

Définition 5.14. Une MT à k têtes est comme une MT à k rubans avec ajustement de manière à correspondre à la situation :



■
Seule nouvelle technicalité : conflits d’écriture.
(Régler comme suit : en cas de conflit, la tête de plus petit numéro est celle qui écrit.) ▲

Proposition 5.15. Si Y est décidé (reconnu) par une MT à k têtes alors Y est décidé (reconnu) par une MT. ▲

Remarque 5.16. Souvent avantageux d’utiliser un modèle de MT plus “convivial” que la MT “standard”, sachant les deux modèles équivalents à l’égard de la décidabilité et de la reconnaissance. ▲

■
Ex : MT pour $\{w\#w : w \in \{a, b\}^*\}$ avec 2 têtes.

Variante multi-dimensionnelle

Définition 5.17. Une **MT à k dimensions** est une MT dont le “ruban” est un hypercube de dimension k bi-infini le long de chaque axe. ▲

Précisions :

- l’hypercube contient w au départ le long de son premier axe et tout le reste est blanc,
- l’unique tête peut se déplacer positivement ou négativement le long de chacun des k axes.

Théorème 5.18. Si Y est décidé (reconnu) par une MT multi-dimensionnelle alors Y est décidé (reconnu) par une MT.

Aperçu de la preuve. (pour 2 dimensions)

- Encoder ligne par ligne le rectangle qui contient des symboles non-blancs
 $aab \dots *a \sqcup b \dots * \dots * \sqcup ab \dots \#$
- une autre piste pour la position de la tête
- gauche/droite dans le rectangle : pas de problème
- haut/bas : un autre ruban pour compter la position dans la rangée où ira la tête
- si on quitte le rectangle : nouvelle colonne [décalage] ou nouvelle rangée [bloc de blancs].

▲

Variante non-déterministe

Définition 5.19. Pas de surprise : une **MT non-déterministe (MTND)** est comme la MT sauf pour $\delta : Q \times \Gamma \rightarrow \mathcal{P}\{Q \times \Gamma \times \{L, R\}\}$. ▲

Ajustements habituels :

- plusieurs transitions sont possibles à partir d’une configuration donnée.■
- w est accepté par la MTND ssi il existe une suite de transitions menant de (ε, q_0, w) à une configuration (u, q_a, v) .■
- $L(\text{MTND})$ est l’ensemble des mots acceptés.

Théorème 5.20. (Sipser 3.16)

Si L est décidé (reconnu) par une MTND, alors L est décidé (reconnu) par une MT.

Aperçu de la preuve.

- trois rubans : entrée, travail, chemin d'exécution (suite particulière de choix),
- énumérer tour à tour tous les chemins d'exécutions, en ordre de longueur croissante,
- pour chaque chemin : recopier l'entrée w sur le ruban de travail, reprendre du début la simulation de la MTND sur w en utilisant les choix prescrits par le chemin,
- accepter dès qu'une des simulations successives mène la MTND à son état acceptant. ▲

Langages et fonctions

À un langage Y sur alphabet Σ correspond sa fonction caractéristique χ_Y , définie comme suit :

$$\chi_Y : \Sigma^* \rightarrow \{0, 1\}$$
$$w \mapsto \begin{cases} 1 & \text{si } w \in Y \\ 0 & \text{sinon.} \end{cases}$$

Réciproquement, à tout problème de décision $f : \Sigma^* \rightarrow \{0, 1\}$ correspond $\{w : f(w) = 1\}$.

Une machine de Turing calcule une fonction

Définition 5.21.

On dit que $f : \Sigma^* \rightarrow \Sigma^*$ est calculée par une MT (déterministe) $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ si

- pour tout $w \in \text{Dom} f$, $(\varepsilon, q_0, w) \xRightarrow{*} (u, q_a, f(w))$ pour un certain $u \in \Gamma^*$, et
- pour tout $w \notin \text{Dom} f$, la MT M boucle sur w . ▲

Remarque 5.22. L'état final rejetant q_r ne joue plus de rôle lorsque la MT est utilisée pour calculer une fonction. ▲

Puissance de calcul des MTs

Jusqu'ou peut-on aller avec les MTs ?

Voyons quelques exemples.

Une machine de Turing $(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ est capable de ...

- décider ou reconnaître
 $\{w\$w\$w \mid w \in \{0, 1\}^*\}$



- calculer la fonction caractéristique d'un langage décidable

- calculer

$$f : \{0, 1\}^* \rightarrow \{0, 1\}^* \\ \text{BINAIRE}(m) \mapsto \text{BINAIRE}(m + 1)$$

Esquisse :

insérer le caractère sentinelle # en première position;
parcourir le ruban de gauche à droite jusqu'au dernier chiffre;
si ce chiffre est 0, alors le remplacer par 1;
sinon, remplacer tous les 1 consécutifs par des 0,
de droite à gauche, jusqu'au premier 0 rencontré,
et remplacer ce 0 par 1;
si on atteint la sentinelle avant de rencontrer un 0,
alors écrire 1 sur la première case;
supprimer la sentinelle si nécessaire.

- décider ou reconnaître
 $\{w_1\$w_2\$ \dots \$w_k \mid \text{les } w_i \in \{0, 1\}^* \text{ sont distincts}\}$



- calculer

$$f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \\ (k, m) \mapsto k + m$$

- calculer

$$f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \\ (k, m) \mapsto k \times m$$

- calculer

$$f : \mathbb{N} \rightarrow \{0, 1\} \\ m \mapsto \begin{cases} 0 & \text{si } m > 0 \text{ est premier} \\ 1 & \text{sinon} \end{cases}$$

- calculer $f \circ g$ sachant calculer f et g

- calculer $\overbrace{f \circ f \circ f \cdots \circ f}^m(w)$ sur entrée $\langle w, m \rangle$ sachant calculer f

- décider $\{\langle m \rangle \mid m \text{ est premier}\}$

MTs et programmes TANTQUE

Théorème 5.23. Soit

$$f : \mathbb{N} \rightarrow \mathbb{N}$$

une fonction calculée par un programme TANTQUE. Alors il existe une MT M qui calcule une fonction

$$f' : \{0, 1\}^* \rightarrow \{0, 1\}^*$$

telle que $f(x) = y$ implique $f'(x') = y'$, où x' et y' sont x et y en binaire.

Aperçu de la preuve. Pour simuler un programme TANTQUE, M doit pouvoir travailler avec des registres contenant des nombres entiers, c'est-à-dire entre autres :

- incrémenter la valeur d'un registre ;
- déterminer si deux registres sont égaux.

La première de ces tâches a déjà été vue, et la deuxième est laissée en exercice.

Les registres sont représentés sur le ruban de M et séparés par le caractère $\#$.

M commence par insérer les caractères $0\#$ au début du ruban, ce qui crée le registre r_0 et l'initialise à 0.

Les registres de travail $r_2, \dots, r_l$ sont créés et initialisés en ajoutant $\#0$ pour chacun d'eux à la fin du ruban.

Les branchements nécessaires à l'exécution du programme TANTQUE sont implantés à l'aide des états de M , c'est-à-dire que différents points dans le programme correspondent à différents états de M .

Lorsque la simulation est terminée, M efface les registres de travail, ramène la tête de lecture en première position, et passe à son état final. ▲

Théorème 5.24. Soit

$$f : \Sigma^* \rightarrow \Sigma^*$$

une fonction calculée par une MT $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$. Alors il existe un programme TANTQUE qui calcule une fonction

$$f' : \mathbb{N} \rightarrow \mathbb{N}$$

telle que $f(x) = y$ implique $f'(x') = y'$, où $x' = \text{Gödel}(x)$ et $y' = \text{Gödel}(y)$.

Aperçu de la preuve. On peut voir le ruban de M comme un tableau infini de caractères de Γ de la forme

$$(c_1, c_2, \dots, c_n, \sqcup, \sqcup, \dots).$$

Si on réserve l'entier 0 pour le caractère $\sqcup$, le ruban peut être représenté par un tableau infini d'entiers de la forme

$$(a_1, a_2, \dots, a_n, 0, 0, \dots),$$

ou, grâce au codage de Gödel, par l'entier

$$p_1^{a_1} p_2^{a_2} \dots p_n^{a_n}.$$

Le registre $r_1 =$ ruban de M .

Le registre $r_2 =$ position de la tête de lecture.

Le registre $r_3 =$ l'état de M encodé.

Chaque transition de M peut être simulée, dans les registres r_1, r_2 et r_3 , à l'aide entre autres des programmes TABLVAL et TABLASS déjà vus.

Lorsque la simulation est terminée, la portion du ruban de M qui contient $f(x)$ est copiée dans le registre r_0 , toujours en codage de Gödel. ▲

Remarque 5.25. Les théorèmes 5.23 et 5.24 s'appliquent aussi aux fonctions f partielles, où la MT et le programme bouclent sur w lorsque $f(w) = \uparrow$, et aux fonctions f à plusieurs variables entières. ▲

Remarque 5.26. Les théorèmes 5.23 et 5.24 nous permettent de décrire des MT comme des programmes TANTQUE, si cela est utile. ▲

Hypothèse de Church-Turing

La

notion intuitive d'algorithme

est capturée par la

notion formelle de machine de Turing.

Hypothèse de Church-Turing (énoncé équivalent)

La

notion intuitive d'algorithme

est capturée par

la notion formelle de programme TANTQUE.

Évidence accablante supportant l'hypothèse de Church-Turing

La machine de Turing est aussi équivalente

- à ses nombreuses variantes,
- au lambda-calcul,
- aux algorithmes de Markov,
- aux grammaires générales,
- à d'autres modèles bizarroïdes.



(<http://www-gap.dcs.st-and.ac.uk/history/PictDisplay/Church.html>)

Logicien de Princeton, inventeur du lambda-calcul (programmation fonctionnelle), publication en 1936 avant Turing, a ensuite dirigé le doctorat de Turing, etc.

Alonzo Church, 1903-1995

La MT et nous

Vrai ou faux ? On peut faire tourner un programme Java sur une machine de Turing.

Vrai par un argument intuitif :

toute fonction calculée par un programme Java est calculable au sens intuitif, et on fait appel à l'hypothèse de Church-Turing.

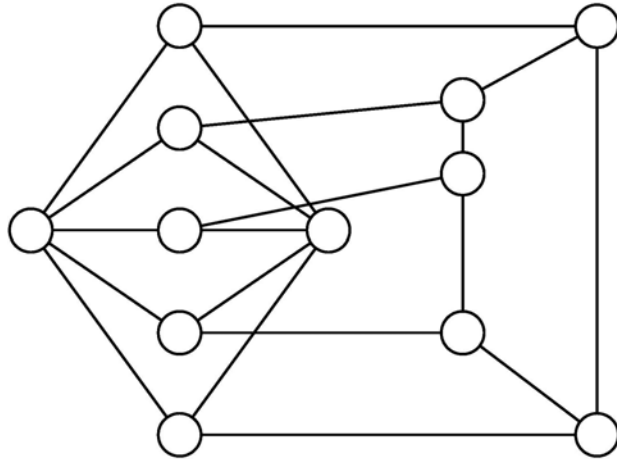
Vrai aussi par un argument formel détaillé qui reposerait sur :

- une suite de bits peut encoder :
nombres entiers, graphes, tableaux, listes, piles, structures complexes, bases de données, etc.
- ruban d'une mT $\approx$ mémoire rudimentaire,
- instructions d'une mT $\approx$ assembleur rudimentaire,
- fonction de transition $\approx$ programme.

Convainquez-vous !

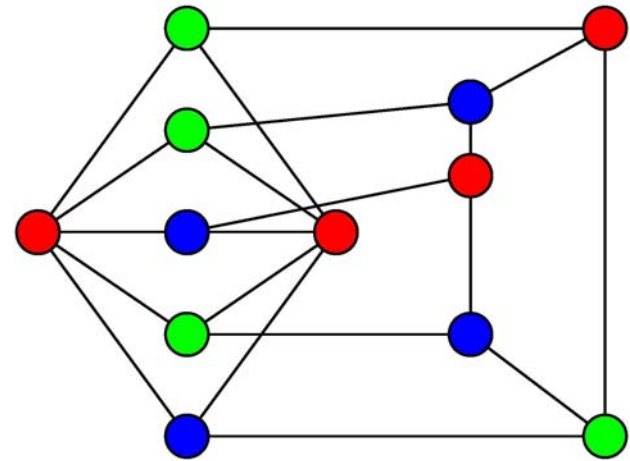
Convaincus, nous pourrions nous contenter de descriptions informelles d'algorithmes ou de machines de Turing.

Exemple : déterminer la 3-colorabilité



2105
ÉTÉ 2010

5 - Pg 57
Turing et compagnie



2105
ÉTÉ 2010

5 - Pg 58
Turing et compagnie

Exemple 5.27. Le langage des graphes 3-coloriables est **décidable**.

Aperçu de la preuve. Besoin d'une machine de Turing

$$(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$$

qui, lorsqu'un graphe G est placé sur son ruban, atteint à la fin de son calcul

$$\begin{cases} \text{l'état } q_a & \text{si } G \text{ est 3-coloriable} \\ \text{l'état } q_r & \text{sinon.} \end{cases}$$

2105
ÉTÉ 2010

5 - Pg 59
Turing et compagnie

Niveau de détail typique :

sur entrée $\langle G \rangle$, où G est un graphe de m sommets, la machine M

- engendre tour à tour les 3^m coloriages possibles
- pour chaque coloriage, M examine chaque paire de sommets adjacents de G
- si M découvre un coloriage qui attribue à chaque paire de sommets adjacents de G des couleurs distinctes alors M entre en q_a , sinon à la fin M entre en q_r .

2105
ÉTÉ 2010

5 - Pg 60
Turing et compagnie

Révision

- machine de Turing
- langage décidable / reconnaissable
- équivalence de différents modèles de machines de Turing
- Hypothèse de Church-Turing : [La machine de Turing capture la notion d'algorithme.](#)

À venir plus tard

Existe-t-il des problèmes qu'aucun algorithme ne peut résoudre ?

Existe-t-il des langages indécidables ?

Existe-t-il des langages non reconnaissables ?