Théorie des langages

Alexis Nasr

Machines de Turing

- Définies par Alan Turing en 1936
- Proches des automates finis mais avec une mémoire infinie et à accès direct.
- Modèle plus proche d'un ordinateur.
- Une machine de Turing (MT) peut faire tout ce qu'un ordinateur peut faire.

Généralités

- La mémoire de la MT est matérialisée par une bande de lecture/écriture.
- Elle possède une tête de lecture/écriture pouvant se déplacer vers la gauche et vers la droite.
- Au départ, la bande contient le mot à reconnaître et possède des $\square$ dans toutes les autres cases.

Caractéristiques

- Une MT peut lire et écrire sur la bande de lecture/écriture.
- La tête de lecture/écriture peut se déplacer vers la droite et vers la gauche.
- La bande de lecture écriture est infinie.
- Lorsque la MT atteint l'état d'acceptation ou l'état de rejet, elle s'arrête et accepte ou rejette le mot.
- Si la MT n'atteint pas l'état d'acceptation ou de rejet, elle peut continuer indéfiniment.

Exemple

Principe d'une machine reconnaissant $L = \{m\sharp m \mid m \in \{0,1\}^*\}$

1. Parcourt le mot pour vérifier qu'il possède un $\sharp$ unique. Si ce n'est pas le cas, va dans l'état de rejet.
2. Fait des allers-retours entre les deux occurrences de m pour vérifier qu'elles contiennent bien le même symbole. Si ce n'est pas le cas, va dans l'état de rejet. Les symboles sont éliminés au fur et à mesure qu'ils sont vérifiés.
3. Lorsque tous les symboles au gauche de $\sharp$ ont été éliminés, vérifie qu'il ne reste plus de symboles à droite de $\sharp$. Si c'est le cas, va dans l'état d'acceptation, sinon va dans l'état de rejet.

Exemple d'exécution

$$
\begin{array}{ccccccccccccccccc}
\downarrow & & & & & & & & & & & & & & & & \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & \sharp & 0 & 1 & 1 & 0 & 0 & 0 & 0 & \square & \ldots \\
 & \downarrow & & & & & & & & & & & & & & & \\
x & 1 & 1 & 0 & 0 & 0 & 0 & \sharp & 0 & 1 & 1 & 0 & 0 & 0 & 0 & \square & \ldots \\
 & & & & & & & & \downarrow & & & & & & & & \\
x & 1 & 1 & 0 & 0 & 0 & 0 & \sharp & x & 1 & 1 & 0 & 0 & 0 & 0 & \square & \ldots \\
\downarrow & & & & & & & & & & & & & & & & \\
x & 1 & 1 & 0 & 0 & 0 & 0 & \sharp & x & 1 & 1 & 0 & 0 & 0 & 0 & \square & \ldots \\
 & \downarrow & & & & & & & & & & & & & & & \\
x & x & 1 & 0 & 0 & 0 & 0 & \sharp & x & 1 & 1 & 0 & 0 & 0 & 0 & \square & \ldots \\
 & & & & & & & & & & & & & & & \downarrow & \\
x & x & x & x & x & x & x & \sharp & x & x & x & x & x & x & x & \square & \ldots
\end{array}
$$

Représentation graphique

$$1 \xrightarrow{a \to b, D} 2$$

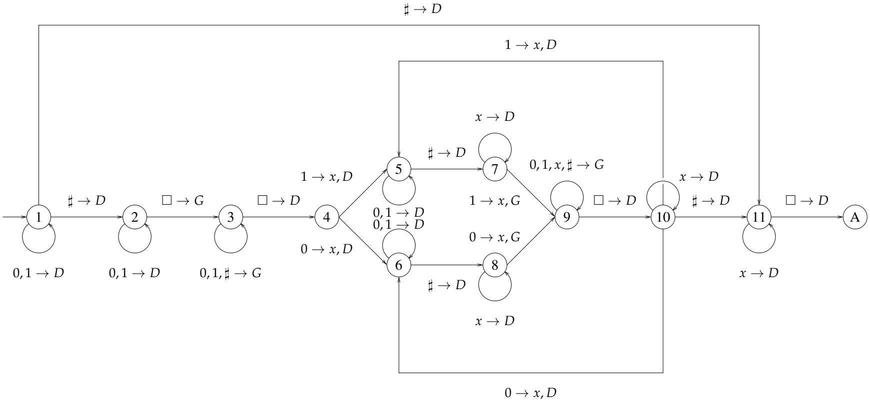
La machine est en 1, la tête de lecture est sur a, elle :

- écrit un b sur la bande (à la place du a),
- décale la tête de lecture d'une case vers la droite (D)
- va en 2.

cas particuliers

- $a \to G$: la machine n'écrit rien.
- $a, b \to c, D$: la machine lit un a ou un b.

Exemple



Toutes les transitions manquantes mènent à l'état de rejet.

Définition

Une MT est un 7-uplet $\langle Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R \rangle$ où :

- Q est l'ensemble des états,
- Σ est l'alphabet de l'entrée (qui ne contient pas le symbole spécial $\Box$),
- Γ est l'alphabet de la bande ($\Box \in \Gamma$ et $\Sigma \subseteq \Gamma$),
- δ est la fonction de transition :

$$\delta : Q \times (\Gamma \cup \{\varepsilon\}) \to Q \times \Gamma \times \{D, G\}$$

- $q_0 \in Q$ est l'état initial,
- $a_A \in Q$ est l'état d'acceptation,
- $a_R \in Q$ est l'état de rejet, avec $q_R \neq q_A$.

Configurations et mouvement

$M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R \rangle$

- Configuration : $(u, q, av) \in \Gamma^* \times Q \times \Gamma^*$ où :
  - q représente l'état courant de l'unité de contrôle
  - u est la partie de la bande se trouvant à gauche de la tête.
  - v est la partie de la bande se trouvant à droite de la tête.
  - a est le symbole se trouvant sous la tête.
- Configuration initiale : (ε, q_0, m) où m est le mot à reconnaître
- Configuration d'acceptation : (u, q_A, v)
- Configuration de rejet : (u, q_R, v)
- Mouvement : $(ua, q_i, bv) \vdash (u, q_j, acv)$ si $(q_j, c, G) \in \delta(q_i, b)$.

Exemple d'exécution

	Configuration			Transition	Destination
	$(\square$	1	$011000\sharp011000)$	$0 \to D$	1
$\vdash$	$(0$	1	$11000\sharp011000)$	$1 \to D$	1
		$\dots$			
$\vdash$	$(011000$	1	$\sharp011000)$	$\sharp \to D$	2
$\vdash$	$(011000\sharp$	2	$011000)$	$0 \to D$	2
		$\dots$			
$\vdash$	$(011000\sharp011000$	2	$\square)$	$\square \to G$	3
$\vdash$	$(011000\sharp011000$	3	$0)$	$0 \to G$	3
		$\dots$			
$\vdash$	$(\square$	3	$\square011000\sharp011000)$	$\square \to D$	4
$\vdash$	$(\square$	4	$011000\sharp011000)$	$0 \to x, D$	6
		$\dots$			
$\vdash$	$(x11000$	6	$\sharp011000)$	$\sharp \to D$	8
$\vdash$	$(x11000\sharp$	8	$011000)$	$0 \to x, G$	9
$\vdash$	$(x11000$	9	$\sharp x11000)$	$\sharp \to G$	9
		$\dots$			
$\vdash$	$(\square$	9	$\square x11000\sharp x11000)$	$\square \to$	10
$\vdash$	$(\square$	10	$x11000\sharp x11000)$	$x \to D$	10
$\vdash$	$(x$	10	$11000\sharp x11000)$	$1 \to x, D$	5
$\vdash$	$(xx$	5	$1000\sharp x11000)$	$1 \to D$	5
		$\dots$			

Langages Turing reconnaissables

- L'ensemble des mots reconnus par une MT $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R \rangle$, est le langage de M, noté $L(M)$

$$L(M) = \{m \in \Sigma^* | (\varepsilon, q_0, m) \overset{*}{\vdash} (u, q_A, v)\}$$

- Un langage est Turing reconnaissable (on dit aussi ou récursivement énumérable) si et seulement s'il existe une MT qui le reconnaît.

Non Déterminisme

- Fonction de transition non déterministe

$$\delta : Q \times (\Gamma \cup \{\varepsilon\}) \rightarrow \wp(Q \times \Gamma \times \{D, G\})$$

- Pour toute MT A non déterministe, il existe une MT A' telle que $L(A) = L(A')$.
- Pourquoi ?
  - On peut simuler le comportement d'une MT N avec une MT déterministe D
  - D parcourt les branches de l'arbre des configurations de N
  - si D tombe sur une configuration d'acceptation, elle accepte

Langages Turing décidables

- Trois possibilités lorsqu'on essaie de reconnaître un mot à l'aide d'une MT.
  - L'exécution se termine et le mot est accepté.
  - L'exécution se termine et le mot est rejeté.
  - L'exécution ne se termine pas.
- Une MT dont l'exécution se termine pour tout mot est appelée un décideur.
- Si une MT accepte tous les mots du langage L et rejette tous le autres, on dit qu'elle décide L.
- On dit qu'un langage est Turing décidable (ou simplement décidable) si et seulement s'il existe une MT qui le décide.
- La Turing décidabilité est une condition plus forte que la Turing reconnaissance : tout langage Turing décidable est Turing reconnaissable.

Exemples de langages décidables

- $L = \{m\sharp m \mid m \in \{0,1\}^*\}$
- $L = \{0^{2^n} \mid n \geq 0\}$

Thèse de Church-Turing

Tout traitement réalisable par un algorithme peut être accompli par une machine de Turing.

Problème = langage

- Soit un problème de décision (réponse oui/non) dont les instances sont encodées par des mots définis sur un alphabet Σ.
- L'ensemble de tous les mots sur Σ peut être partitionné en deux ensembles :
  - ceux qui codent des instances du problème pour lesquels la réponse est oui (instances positives).
  - ceux qui codent des instances du problème pour lesquels la réponse est non et ceux qui ne représentent pas des instances du problème (instances négatives)
- Un problème peut alors être représenté par le langage constitué de l'ensemble de ses instances positives.
- Résoudre le problème c'est décrire le langage des instances positives.

Autres langages (problèmes) décidables

$A_{AD} =$
$\{\langle A, m\rangle \mid A$ est un automate fini déterministe et m un mot reconnu par $A\}$

- on construit une machine de Turing M qui :
- étant donné $\langle A, m\rangle$ où A est un automate déterministe et m un mot
- vérifie que A est bien formé $A = \langle Q, \Sigma, \delta, q_0, F\rangle$
- simule le comportement de A pour m :
  - écrit sur la bande la configuration courante : $\langle q, i\rangle$ où q est l'état courant de l'automate et i la position courante dans m
  - initialise de la configuration courante $q = q_0$ et $i = 1$
  - met à jour la configuration courante grâce à la fonction de transition
  - si $i = |m|$
    - si $q \in F$ va à l'état d'acceptation
    - sinon va à l'état de rejet

Autres langages (problèmes) décidables

$A_{GHC} =$
$\{\langle G, m\rangle \mid G$ est une grammaire hors-contexte qui génère le mot $m\}$

- solution naïve : tenter toutes les dérivations possibles à partir de S à la recherche de la dérivation menant à m
- ne marche pas car si $m \notin L(G)$ et $L(G)$ est inifini, M ne s'arrêtera pas
- on transforme G sous forme normale de Chomsky (CNF)
- une grammaire en CNF définit une dérivation de longueur $2n - 1$ pour un mot de longueur n
- M essaye toutes les dérivations de longueur $2n - 1$
  - si l'une d'entre elles génère m va à l'état d'acceptation
  - sinon va à l'état de rejet

Langages non décidables

- Certains langages ne sont pas décidables

Le problème de l'arrêt

- Etant donné la MT M et un mot m, tester si M accepte m.

$$A_{MT} = \{\langle M, m\rangle \mid M \text{ est une MT qui accepte } m\}$$

- Le langage A_{MT} est indécidable.
- Le langage A_{MT} est récursivement énumérable, il suffit de simuler M sur m et accepter $\langle M, m\rangle$ si M accepte m.
- Problème : il se peut que sur l'entrée m, M ne s'arrête pas.

Conséquence du problème de l'arrêt

- Il n'existe pas de programme informatique qui prendrait comme entrée le code d'un programme informatique quelconque et qui grâce à la seule analyse de ce code ressortirait VRAI si le programme s'arrête et FAUX sinon.
- En d'autres termes, on ne peut construire un compilateur capable de déterminer dans tous les cas si le programme bouclera indéfiniment ou non.

Autres langages non décidables

- $EQ_{GHC} = \{\langle G, H \rangle \mid G$ et H sont des grammaires hors-contexte et $L(G) = L(H)\}$
- $EQ_{MT} = \{\langle M_1, M_2 \rangle \mid M_1$ et M_2 sont deux MT et $L(M_1) = L(M_2)\}$
- $AMB_{GHC} = \{\langle G \rangle \mid G$ est une grammaire hors-contexte ambiguë$\}$
- $REG_{MT} = \{\langle M \rangle \mid M$ est une MT et $L(M)$ est un langage régulier$\}$

Sources

- Michael Sipser *Introduction to the Theory of Computation* PWS Publishing Company, 1997.
- John Hopcroft, Rajeev Motwani, Jeffrey Ullman *Introduction to Automata Theory, Languages and Computation*, 2ème édition Pearson Education International, 2001.
- John Aho, Jeffrey Ullman *The Theory of Parsing, Translation and Compiling, Vol I : Parsing* Prentice-Hall, 1972