Automates et Calculabilité

– Les Machines de Turing –

Leçon 10 : 5 jan 2004

Uwe Nestmann

EPFL-I&C

Plan du cours

1 — 20 oct 2003: introduction

2 — 27 oct 2003: techniques de preuves

3 — 3 nov 2003: K 2+3+4: chaînes, ensembles (réguliers), automates finis

4 — 10 nov 2003: K 5+6: non-détérminisme (AFN), "subset construction"

5 — 17 nov 2003: K 7+8: filtrage des motifs & expressions régulieres

6 — 24 nov 2003: K 9+A: expressions régulieres & algèbres de Kleene

7 — 1 dec 2003: K 10+11+12: homomorphismes, limitations & lemme de gonflement

8 — 8 dec 2003: K 13+14: minimisation

9 — 15 dec 2003: K 15+16: Myhill-Nerode

---

10 — 5 jan 2004: K 28+29: machines de Turing & thèse de Church-Turing

11 — 12 jan 2004: K 31+32: machines universelles et diagonalisation

12 — 19 jan 2004: K 33+34: réduction & théorème de Rice

13 — 26 jan 2004: K 36+37+I: autres formalismes, λ-calcul & programmes basés sur while

14 — 2 fév 2004: K 38+39+K: l'incomplétude selon Gödel

Résumé

Nous avons appris :

- les concepts d'***automates*** : AFD, AFN, et AFN$_\epsilon$
- que ces trois concepts sont équivalents :
  ils acceptent les mêmes langages, les ***langages réguliers***
- qu'ils sont équivalents au concept des ***expressions régulières*** et au filtrage par certains motifs
- comment passer d'un concept à l'autre
- comment ***minimiser*** les automates
- que l'ensemble des langages réguliers
  est ***clos*** sous beaucoup de constructions standards
- qu'il existe des langages ***non-réguliers***, et
- comment démontrer qu'un langage n'est pas régulier
  (par le lemme de ***gonflement*** et/ou par ***Myhill-Nerode***).

Description informelle (I)

Une machine de Turing (**MdT**)
est composée des éléments suivants :

- un ***ensemble fini d'états***
  (initial, accepteur, rejettant, autres) ;
- une ***mémoire infinie*** sous la forme d'un ***ruban*** divisé en cases ; chaque case du ruban peut contenir un symbole d'un *alphabet de ruban* ;
- une ***tête de lecture & écriture***, se déplaçant sur le ruban ;
- une ***fonction de transition*** qui pour tout état de la machine et tout symbole se trouvant sous la tête de lecture précise :
  - l'état suivant ;
  - un nouveau caractère à écrire par la tête ;
  - un sens de déplacement de la tête ;

Description informelle (II)

L'***exécution*** d'une MdT se fait comme suit :

- ☐ Initialement,
  - le ***mot (fini!) d'entrée*** se trouve au début du ruban ; les autres cases du ruban contiennent toutes un *symbole blanc* ;
  - la tête est sur la première case du ruban ;
  - la machine se trouve dans l'état initial.
- ☐ A chaque étape de l'exécution, la machine :
  - lit le symbole se trouvant sous sa tête,

  et, comme défini par la ***fonction de transition*** :
  - remplace ce symbole par un autre symbole (ou le même)
  - déplace sa tête à gauche ou à droite, et
  - change l'état de la machine.

Comparaison informelle avec les automates

Comme les automates,

- les MdT servent à accepter des langages ;
- les MdT fonctionnent avec un nombre fini d'états, des états initiaux et des états accepteurs.

Contrairement aux automates,

- les MdT disposent d'une ***mémoire non bornée***
  … ce qui les rend plus expressives ;
- la longueur finie d'un mot d'entrée ne garantit pas qu'une MdT s'***arrêt***era, parce que
  - la tête peut de déplacer dans les ***deux sens***, et
  - le ***mot*** sur le ruban ***change*** pendant l'exécution ;
- les MdT disposent (dans la version de Kozen) aussi d'un ensemble d'états rejettants.

Définition formelle

<u>Définition:</u> Une MdT est un 9-tuple

$$M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$$

- Q est un ensemble fini d'états ;
- Σ est *l'alphabet d'entrée* ;
- Γ est *l'alphabet du ruban*, avec $\Sigma \subseteq \Gamma$;
- $\vdash$ $(\in \Gamma \setminus \Sigma)$ marque le début (la position 0) du ruban ;
- $\sqcup$ $(\in \Gamma \setminus \Sigma)$ est le *symbole blanc* ;
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{ L, R \}$ est la fonction de transition ;
- $s \in Q$ est l'état initial ;
- $t \in Q$ est l'état accepteur ;
- $r \in Q$ est l'état rejettant.

(Il existe beaucoup de définitions alternatives pour une MdT.)

Fonction de transition (I)

La signification de

$$\delta(p, a) = (q, b, d)$$

et la suivante :

si

- la machine est dans l'état p, et
- le symbole sous la tête est un a,

alors la machine

- remplace le symbole a sous la tête par le symbole b,
- puis déplace d'une case la tête dans la direction d, et
- passe dans l'état q.

Fonction de transition (II)

Il faut rajouter des contraintes pour “stabiliser” la définition.

- “La fin du ruban doit rester stable.”

$$\forall p \in Q : \quad \exists q \in Q : \quad \delta(p, \vdash) = (q, \vdash, R)$$
$$\Big(\ \forall p, q, a, b, d \in \ldots : \quad (\ \delta(p, a) = (q, b, d) \wedge b = \vdash \ ) \implies a = \vdash \ \Big)$$

- “Une fois accepté/rejeté, toujours accepté/rejeté.”

$$\forall b \in \Gamma : \quad \exists c \in \Gamma : \quad \exists d \in \{\, L, R \,\} : \quad \delta(t, b) = (t, c, d)$$
$$\forall b \in \Gamma : \quad \exists c \in \Gamma : \quad \exists d \in \{\, L, R \,\} : \quad \delta(r, b) = (r, c, d)$$

Configurations des MdT

Notation: Comment noter le ruban qui est non-borné ?
Nous utilisons la notation générale a^ω (resp. Γ^ω)
pour dénoter (resp. l'ensemble) des ***chaînes de longueur infinie*** constituées exclusivement de concatenations du symbole a (resp. de symboles arbitraires de l'alphabet Γ).
$\rightarrow$ Comparer avec a^* (et resp. Γ^*).

Définition: Une ***configuration*** d'une MdT est composée

- de l'état actuel de la machine $(\in Q)$,
- du contenu actuel du ruban $(\in \{\vdash y \sqcup^\omega \mid y \in \Gamma^* \})$,
- de la position de la tête sur le ruban $(\in \mathbb{N})$,

elle est donc un élément de $Q \times \{\vdash y \sqcup^\omega \mid y \in \Gamma^* \} \times \mathbb{N}$.

Pour un mot arbitraire d'entrée $x \in \Sigma^*$,
la ***configuration initiale*** est $(s, \vdash x \sqcup^\omega, 0)$.

Relation de transition

Notation:

- Si $z \in \Gamma^\omega$ et $n \in \mathbb{N}$, alors z_n dénote le $n^{\text{ème}}$ symbole de z.
- Si $z \in \Gamma^\omega$ et $n \in \mathbb{N}$, alors $\mathrm{s}_b^n(z)$ dénote le mot z où le $n^{\text{ème}}$ symbole a été remplacé par b.

Exemple: $\mathrm{s}_b^4(\vdash baa\underline{a}cabca\sqcup^\omega) = \vdash baa\underline{b}cabca\sqcup^\omega$

Définition:
Soit (p, z, n) une configuration d'une MdT M.
Nous définissons la relation de ***transition entre configurations*** par :

$$(p, z, n) \to_M^1 \begin{cases} (q, \mathrm{s}_b^n(z), n{-}1) & \text{if } \delta(p, z_n) = (q, b, L) \\ (q, \mathrm{s}_b^n(z), n{+}1) & \text{if } \delta(p, z_n) = (q, b, R) \end{cases}$$

(Par les contraintes sur δ, si $\mathrm{s}_b^n(z)$ et $n = 0$ alors $z_0 = \vdash$ et $b = \vdash$).

Acceptation

Définition: Soit α, β, γ des configurations d'une MdT M.
La ***fermeture reflexive et transitive*** de $\rightarrow_M^1$ est définie par :

- ☐ $\alpha \rightarrow_M^0 \alpha$.
- ☐ $\alpha \rightarrow_M^{n+1} \beta$ s'il existe γ tel que $\alpha \rightarrow_M^n \gamma \rightarrow_M^1 \beta$.
- ☐ $\alpha \rightarrow_M^* \beta$ s'il existe $n \geq 0$ tel que $\alpha \rightarrow_M^n \beta$.
- ☐ $\alpha \rightarrow_M^+ \beta$ s'il existe $n > 0$ tel que $\alpha \rightarrow_M^n \beta$.

Définition:
Soit $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$ une MdT.
M ***accepte*** le mot $x \in \Sigma^*$, s'il existe $y \in \Gamma^*$ et $n \in \mathbb{N}$ tel que

$$(s, \vdash x \sqcup^\omega, 0) \quad \rightarrow_M^* \quad (t, y, n)$$

$L(M)$ dénote l'ensemble des mots qui sont acceptés par M.

Arrêt

Définition:
Soit $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$ une MdT.

M ***rejette*** le mot $x \in \Sigma^*$, s'il existe $y \in \Gamma^*$ et $n \in \mathbb{N}$ tel que

$$(s, \vdash x \sqcup^\omega, 0) \quad \rightarrow_M^* \quad (r, y, n)$$

M ***s'arrête*** pour le mot $x \in \Sigma^*$, si M *accepte ou rejette* x.

M est ***totale***, si elle *s'arrête pour tout mot* $x \in \Sigma^*$.

NB : Il est possible que M ne s'arrête pas pour un mot $x \in \Sigma^*$.
Dans ce cas, on dit que la machine *boucle*.

Exemple : accepter $\{ a^n b^n \mid n \geq 0 \}$

Non-régulier !

Idée (pour la programmation d'une MdT appropriée) :

1. vérifier que le mot d'entrée est de la forme $a^* b^*$;
2. marquer la fin du mot, par exemple, avec le symbole $\dashv \in \Gamma$;
3. par traversées successives du mot,
   (a) à chaque traversée de droite à gauche,
   effacer le premier b, puis le premier a que l'on lit ;
   (b) à chaque traversée de gauche à droite,
   effacer le premier a, puis le premier b que l'on lit ;
4. si un symbole attendu lors d'une traversée n'apparaît pas
   $\rightarrow$ rejetter ;
5. si la dernière traversée ne permet de lire que des $\sqcup$
   $\rightarrow$ accepter.

Exemples … cas de test

rejetter le mot $aaba$:

$$
\begin{array}{ll}
 & \underline{\vdash} aaba\sqcup^\omega \\
\rightarrow^+ & \vdash aab\underline{a}\sqcup^\omega \\
\rightarrow^+ & \text{état } r
\end{array}
$$

rejetter le mot aab :

$$
\begin{array}{ll}
 & \underline{\vdash} aab\sqcup^\omega \\
\rightarrow^+ & \vdash aa\underline{b}\dashv\sqcup^\omega \\
\rightarrow^+ & \vdash a\underline{a}\sqcup\dashv\sqcup^\omega \\
\rightarrow^+ & \vdash \underline{a}\sqcup\sqcup\dashv\sqcup^\omega \\
\rightarrow^+ & \vdash \sqcup\underline{\sqcup}\sqcup\dashv\sqcup^\omega \\
\rightarrow^+ & \vdash \sqcup\sqcup\sqcup\underline{\dashv}\sqcup^\omega \\
\rightarrow^+ & \text{état } r
\end{array}
$$

accepter le mot $aaabbb$:

$$
\begin{array}{ll}
 & \underline{\vdash} aaabbb\sqcup^\omega \\
\rightarrow^+ & \vdash aaabb\underline{b}\dashv\sqcup^\omega \\
\rightarrow^+ & \vdash aaab\underline{b}\sqcup\dashv\sqcup^\omega \\
\rightarrow^+ & \vdash a\underline{a}\sqcup bb\sqcup\dashv\sqcup^\omega \\
\rightarrow^+ & \vdash \sqcup\underline{a}\sqcup bb\sqcup\dashv\sqcup^\omega \\
\rightarrow^+ & \vdash \sqcup a\sqcup\sqcup\underline{b}\sqcup\dashv\sqcup^\omega \\
\rightarrow^+ & \vdash \sqcup a\sqcup\underline{\sqcup}\sqcup\sqcup\dashv\sqcup^\omega \\
\rightarrow^+ & \vdash \underline{\sqcup}\sqcup\sqcup\sqcup\sqcup\sqcup\dashv\sqcup^\omega \\
\rightarrow^+ & \vdash \sqcup\sqcup\sqcup\sqcup\sqcup\sqcup\underline{\dashv}\sqcup^\omega \\
\rightarrow^+ & \text{état } t
\end{array}
$$

Exemple … déterminer les états

s	travaille sur	$\vdash \; \Sigma^* \; \sqcup^\omega$
q_1	travaille sur	$\vdash \; a^*b \; \Sigma^* \; \sqcup^\omega$
q_2	travaille sur	$\vdash \; \sqcup^* \; a^* \; \sqcup^* \; b^* \; \sqcup^* \; \dashv \; \sqcup^\omega$
q_3	travaille sur	$\vdash \; \sqcup^* \; a^* \; \sqcup^* \; b^* \; \sqcup^* \; \dashv \; \sqcup^\omega$
q_4	travaille sur	$\vdash \; \sqcup^* \; a^* \; \sqcup^* \; b^* \; \sqcup^* \; \dashv \; \sqcup^\omega$
q_5	travaille sur	$\vdash \; \sqcup^* \; a^* \; \sqcup^* \; b^* \; \sqcup^* \; \dashv \; \sqcup^\omega$
q_6	travaille sur	$\vdash \; \sqcup^* \; a^* \; \sqcup^* \; b^* \; \sqcup^* \; \dashv \; \sqcup^\omega$
q_7	travaille sur	$\vdash \; \sqcup^* \; a^* \; \sqcup^* \; b^* \; \sqcup^* \; \dashv \; \sqcup^\omega$
t	travaille sur	$\vdash \; \sqcup^* \; \dashv \; \sqcup^\omega$ ou presque …

On tombe en état r quand on est en
ou on attend un
mais on lit un

q_2	q_3	q_5	q_6
$b, \sqcup$	$a, \sqcup$	$a, \sqcup$	$b, \sqcup$
a	$\vdash$	b	$\dashv$

Exemple : accepter $\{\, ww \mid w \in \{a, b\}\,\}$

Non-régulier !

Idée (pour la programmation de la MdT) :

1. compter le nombre de symboles du mot d'entrée.
   Si ce nombre est impair, rejetter immédiatement;
2. marquer la fin du mot par le symbole $\dashv$
3. par traversées successives du mot,
   (a) chercher et séparer les deux moitiés en marquant le premier symbole non-marqué vu à chaque traversée.
   (b) vérifier symbole par symbole que les deux moitiés sont identiques en effaçant les symboles correspondants à l'aide de la mémoire ...

Bon exercice : définir la MdT qui effectue cet algorithme ...

Ensembles récursifs

Définition:
Soit $A \subseteq \Sigma^*$ un langage sur l'alphabet Σ.

A est ***récursivement énumerable*** (***r.e.***)
s'il existe une MdT M sur Σ tel que $L(M) = A$.

A est ***co-r.e.*** si son complément $\overline{A}$ est r.e..

A est ***récursif***
s'il existe une MdT ***totale*** M sur Σ tel que $L(M) = A$.

Exemple: $\{ a^n b^n \mid n \geq 0 \}$ est-il

- ☐ r.e. ?
- ☐ co-r.e. ?
- ☐ récursif ?

Récursivité et clôture sous complément

Proposition:

Soit $A \subseteq \Sigma^*$ un langage sur l'alphabet Σ.
Si A est récursif, alors son complément $\overline{A}$ est aussi récursif.

N.B.: Ceci n'est pas le cas pour les langages r.e. !

Idée de la démonstration :

- Si A est récursif,
  alors il existe une MdT *totale* M telle que $L(M) = A$.
- Si M est totale,
  alors M accepte $x \in \Sigma^*$ ssi M ne rejette pas x.
- Donc, et comme M est déterministe, on peut échanger les états t et r pour accepter $\overline{A}$ à la place de A.
- Cette MdT est totale et, donc, $\overline{A}$ est récursif. C.Q.F.D.

Récursivité et r.e.

Proposition:

Soit $A \subseteq \Sigma^*$ un langage sur l'alphabet Σ.
Si A est r.e. et co-r.e., alors A est récursif.

Idée de la démonstration :

- Soit M une MdT qui accepte A.
- Soit $\overline{M}$ une MdT qui accepte $\overline{A}$.
- On peut construire une MdT N qui travaille simultanément/alternativement comme M et $\overline{M}$.
- N accepte si la partie simulant M accepte.
- N rejette si la partie simulant $\overline{M}$ accepte.
- N accepte A et est totale par construction. C.Q.F.D.

Décidabilité et semi-décidabilité

Définition:
Soit P une propriété sur les mots x d'un alphabet Σ.

P est ***décidable*** ssi $\{ x \mid P(x) \}$ est récursif
P est ***semi-décidable*** ssi $\{ x \mid P(x) \}$ est r.e.

Lemme:
Soit $A \subseteq \Sigma^*$ un langage sur l'alphabet Σ.

A est ***récursif*** ssi "$x \in A$" est décidable
A est ***r.e.*** ssi "$x \in A$" est semi-décidable

Tout langage récursif est r.e., mais l'inverse n'est pas vrai.
Il existe des langages r.e. qui ne sont pas récursifs.

Donc, il existe des propriétés/problèmes qui sont semi-décidables, mais pas décidables. *Existe-t-il aussi des propriétés/problèmes qui ne sont même pas semi-décidable ?*

Thèse de Church-Turing

Le concept de MdT formalise la notion de procédure effective : toute procédure effective peut être calculée par une MdT.
On ne peut pas démontrer cette thèse (elle ne constitue donc pas un théorème). Mais, on peut tenter de la justifier :

- ☐ Toutes les tentatives d'extension du concept de MdT pour lui donner une plus grande expressivité ont raté ; chaque fonction représentable dans l'extension l'était déjà dans le modèle de base. Exemple :
  - MdT avec rubans multiples,
  - MdT avec ruban infini dans les deux sens,
  - MdT nondéterministe, . . .
- ☐ Toutes les autres tentatives de formalisation de la notion de procédure effective sont équivalentes (ou moins puissantes).
  Nous en considérerons deux autres dans le cours. . .