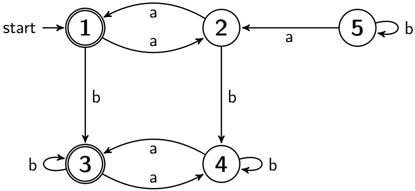
R4.A.12 Automates et Langages

Thibault Godin ; Lucie Naert

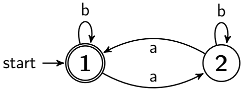
IUT Vannes, Département informatique

Minimisation d'automate



Théorème

Soit L un langage rationnel. Alors il existe un unique DFA ayant un nombre minimal d'état et reconnaissant L



remarque : l'existence d'un automate reconnaissant le langage et ayant un nombre minimal d'état n'est pas surprenante, c'est l'unicité qui importe ici.

Minimisation d'automate : Algorithme de Moore

A un DFA complet $\mathcal{A} = (Q, \Sigma, \delta, i_0, F)$ dont on a supprimé les états non-accessibles

$\mathcal{A}_{min}$ est construit avec une suite de partitions des états de plus en plus fines.

$\rightsquigarrow$ relations d'équivalence $\equiv_h$ données par :

$$\forall p, q \in Q, p \equiv_0 q \Longleftrightarrow (p \in F \text{ et } q \in F) \text{ ou } (p \notin F \text{ et } q \notin F)$$
$$\forall h \geq 0, \forall p, q \in Q, p \equiv_{h+1} q \Longleftrightarrow p \equiv_h q \text{ et } (\forall a \in \Sigma, \delta(qp, a) \ \equiv_h \delta(q, a))$$

En d'autres termes, on commence avec une partition à 2 ensembles, F et $Q \setminus F$, puis on dit que deux états sont équivalents au rang $h+1$ s'ils sont équivalents au rang h et que toutes les transitions mènent dans des états 2 à 2 équivalents.

$$x \xrightarrow{i} y$$

$$x' \xrightarrow{i} y'$$

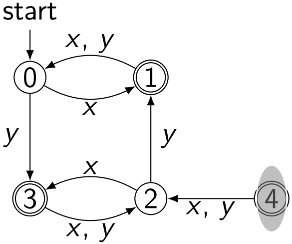
$$\forall x, x' \in Q$$

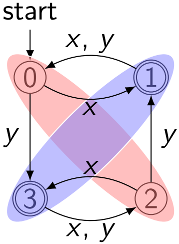
$$x \equiv_0 x' \iff x, x' \in F \text{ ou } x, x' \in Q \setminus F \,,$$

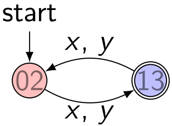
$$\forall k \geqslant 1$$

$$x \equiv_{k+1} y \iff x \equiv_k x' \quad \wedge$$

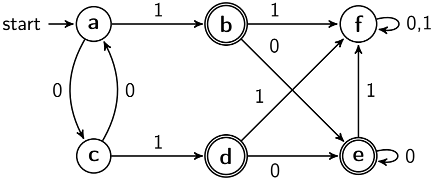
$$\forall i \in \Sigma \colon \delta_i(x) \equiv_k \delta_i(x) \,.$$







Minimisation d'automate : Algorithme de Moore



Minimiser l'automate et en déduire le langage reconnu.

Minimisation d'automate : Algorithme de Moore

Corollaire

L'équivalence de deux langages rationnels est décidable (en temps potentiellement exponentiel)

Motivation

Qu'est-ce qu'on ne peut *pas* faire avec les langages rationnels/reguliers/reconnaissables ?

Lemme de l'étoile

Lemme de l'étoile

Soit L un langage rationnel sur Σ. Il existe un entier N tel que pour tout mot $\boldsymbol{w} \in L$ tel que $|\boldsymbol{w}| \geq N$, il existe $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \Sigma^*$ vérifiant :

- $\boldsymbol{w} = \boldsymbol{xyz}$
- $0 < |\boldsymbol{y}| \leq N$
- $\boldsymbol{xy}^n\boldsymbol{z} \in L$ pour tout entier $n \geq 0$.

Lemme de l'étoile

Lemme de l'étoile (variante)

Soit L un langage rationnel sur Σ. Il existe un entier N tel que tout mot $\boldsymbol{w} \in L$ tel que $|\boldsymbol{w}| \geq N$, il existe $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \Sigma^*$ vérifiant :

- $\boldsymbol{w} = \boldsymbol{xyz}$
- $|\boldsymbol{xy}| \leq N$ et $|\boldsymbol{y}| > 0$
- $\boldsymbol{xy}^n\boldsymbol{z} \in L$ pour tout entier $n \geq 0$.

Langages non-rationnels

Lemme de l'étoile (contraposée partielle)

Soit L un langage sur Σ, tel que, pour tout entier N , il existe $\boldsymbol{w} \in L$ tel que $|\boldsymbol{w}| \geq N$, et pour tout $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \Sigma^*$ vérifiant

- $\boldsymbol{w} = \boldsymbol{xyz}$
- $0 < |\boldsymbol{y}| \leq N$

il existe n tel que $\boldsymbol{x}\boldsymbol{y}^n\boldsymbol{z} \notin L$;

alors L n'est pas régulier.

Exemple : Le langage $L = \{a^n b^n\}$ n'est pas régulier.

Soit N un entier, considérons $\boldsymbol{w} = a^N b^N$. En utilisant la variante, toutes les factorisations possibles de $\boldsymbol{w}$ avec $|\boldsymbol{xy}| \leq N$ sont de la forme $\boldsymbol{x} = a^{N-i}$ $\boldsymbol{y} = a^i$ avec $i > 0$. Mais alors $\boldsymbol{x}\boldsymbol{y}^2\boldsymbol{z} \notin L$, donc L n'est pas rationnel.

Langages non-rationnels

Montrer que les langages suivants ne sont pas rationnels :

1. Des palindromes sur $\{a, b\}$
2. $L = \{a^p |\ p \text{ est un nombre premier}\}$
3. Des expressions bien parenthésées (langages de Dyck)

Grammaires

Une manière inductive de définir/reconnaître $L = \{a^n b^n\}$

- $\varepsilon \in L$
- $\boldsymbol{u} \in L \iff a\boldsymbol{u}b \in L$

Notion de *grammaire* (règles de construction)

$$S \rightarrow aSb|\varepsilon$$

Grammaires

Donner une grammaire pour les langages suivants :

1. Des palindromes sur $\{a, b\}$
2. Des expressions bien parenthésées (langages de Dyck)

Encore plus général

Classe	Nom	Automate	Grammaire	Exemple
type-0	récursivement énumérable	Machine de Turing	$\gamma \to \alpha$	ensemble des machines de Turing qui terminent
type-1	contextuelle	Machine de Turing à ruban linéairement borné	$\alpha A \beta \to \alpha \gamma \beta$	$\{a^n b^n c^n \mid n > 0\}$
type-2	hors-contexte	automate à pile	$A \to \alpha$	$\{a^n b^n \mid n > 0\}$
type-3	régulier	automates finis	$A \to \mathrm{a} \mid B$	$\{a^n \mid n > 0\}$

avec [1] A, B des symboles non terminaux, a un symbole terminal, α, β une chaîne de symboles terminaux et/ou non-terminaux et γ une chaîne non vide symboles terminaux et/ou non-terminaux

1. adapté de `https://en.wikipedia.org/wiki/Chomsky_hierarchy`

Encore plus général

Il existe des langages qui ne sont même pas de type-0

La classe des langages récursifs (importante en informatique) est strictement comprise entre le type-0 et le type-1

Le langage $\{a^p | p \text{ est premier}\}$ est de type-1 : on peut vérifier qu'un mot est ou non dans le langage en utilisant seulement un ruban de la longueur du mot.

On peut définir de nombreux modèles intermédiaires qui peuvent être intéressants du point de vu théorique ou pratique

Pour la compilation, on se place généralement dans la classe type-2