Complexité en état d'un automate

Master 1 - ITA : Analyse d'article

Encadré par Pascal Caron

Edouard HADDAG
Université de Rouen Normandie

3 février 2025

Présentation du papier

Présentation du papier

State complexity of some operations on binary regular languages [1]

Par **Galina Jirásková**, publié dans *Theoretical Computer Science, 330.2* en 2005

1. Jirásková 2005

Alphabet et Mot

Définition 1 (Alphabet)

Un « alphabet » Σ est un ensemble fini non-vide de symboles.

Alphabet et Mot

Définition 1 (Alphabet)

Un « alphabet » Σ est un ensemble fini non-vide de symboles.

Définition 2 (Mot)

Un « mot » w est une suite finie de symboles sur un alphabet Σ. La suite vide de symbole sera notée par ε.

Alphabet et Mot

Définition 1 (Alphabet)

Un « alphabet » Σ est un ensemble fini non-vide de symboles.

Définition 2 (Mot)

Un « mot » w est une suite finie de symboles sur un alphabet Σ. La suite vide de symbole sera notée par ε.

Exemple 3

$$\Sigma = \{a, b\}$$
$$w = abba$$

Alphabet et Mot

Définition 4 (Concaténation d'un mot)

L'opération $\cdot$ désigne la « concaténation » de deux mots.

Alphabet et Mot

Définition 4 (Concaténation d'un mot)

L'opération $\cdot$ désigne la « concaténation » de deux mots.

Exemple 5

$$u = aa \text{ et } v = bb$$

$$w = u \cdot v = aabb$$

Alphabet et Mot

Définition 6 (Le miroir d'un mot)

On notera $\overleftarrow{w}$, le « miroir » du mot w.

Alphabet et Mot

Définition 6 (Le miroir d'un mot)

On notera $\overleftarrow{w}$, le « miroir » du mot w.

Exemple 7

$$u = abcd$$
$$\overleftarrow{u} = dcba$$

Langage

Définition 8 (Langage)

Un « langage » L est un ensemble de mots sur un alphabet Σ.

Langage

Définition 8 (Langage)

Un « langage » L est un ensemble de mots sur un alphabet Σ.

Exemple 9

$$L = \{\varepsilon, aa, bb, abab\}$$

Langage

Définition 10 (La concaténation)

La « concaténation » est définie grâce à la concaténation des mots :

$$
\begin{aligned}
L_1 &= \{a, b\} \\
L_2 &= \{c, d\} \\
L_1 \cdot L_2 &= \{ac, ad, bc, bd\}
\end{aligned}
$$

Langage

Définition 11 (La copie n-ième)

On définit la « copie n-ième » d'un langage L noté L^n :

$$L_1 = \{a, b\}$$
$$L_1^2 = \{aa, ab, bb, ba\}$$

Langage

Définition 12 (L'étoile (de *Kleene*) d'un langage)

On peut définir « l'étoile » d'un langage notée L^* :

$$L^* = \bigcup_{i \geq 0} L^i$$

Langage

Définition 12 (L'étoile (de *Kleene*) d'un langage)

On peut définir « l'étoile » d'un langage notée L^* :

$$L^* = \bigcup_{i \geq 0} L^i$$

Exemple 13

$$L = \{a, b\}$$
$$L^* = \{\varepsilon, a, b, aa, ab, \ldots\}$$

Langage

Définition 14 (Le langage de l'ensemble des mots)

L'ensemble des mots possible sur l'alphabet Σ sera noté Σ^*.

Langage

Définition 14 (Le langage de l'ensemble des mots)

L'ensemble des mots possible sur l'alphabet Σ sera noté Σ^*.

Définition 15 (Le langage miroir)

On pourra noter $\overleftarrow{L}$, « le langage miroir » de L, qui sera le langage des miroirs des mots de L :

$$\overleftarrow{L} = \{\overleftarrow{w} \mid w \in L\}$$

Langage

Définition 14 (Le langage de l'ensemble des mots)

L'ensemble des mots possible sur l'alphabet Σ sera noté Σ^*.

Définition 15 (Le langage miroir)

On pourra noter $\overleftarrow{L}$, « le langage miroir » de L, qui sera le langage des miroirs des mots de L :

$$\overleftarrow{L} = \{\overleftarrow{w} \mid w \in L\}$$

Autres opérations

Les langages étant des ensembles, leurs opérations peuvent être étendues aux langages.

Langage

Définition 16 (Les langages rationnels)

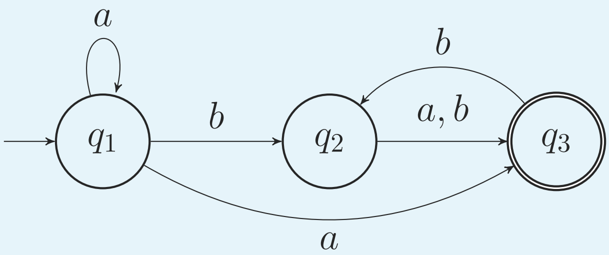
L'ensemble des « langages rationnels » sur Σ, noté $\mathrm{Rat}(\Sigma^*)$, est le plus petit ensemble de langages sur Σ qui vérifie :

- $\mathrm{Rat}(\Sigma^*)$ contient $\varnothing$ et $\{a\}$ avec $a \in \Sigma$.
- $\mathrm{Rat}(\Sigma^*)$ est fermé pour l'union, la concaténation et l'étoile.

Automate

Définition 17 (Automate)

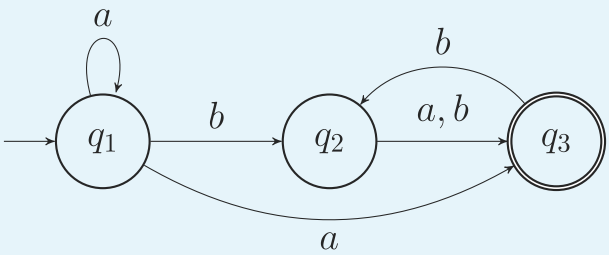
$M = (Q, I, F, \delta)$ avec $Q = \{q_1, q_2, q_3\}$, $I = \{q_1\}$ et $F = \{q_3\}$.



Automate

Définition 18 (L'acceptation d'un mot)

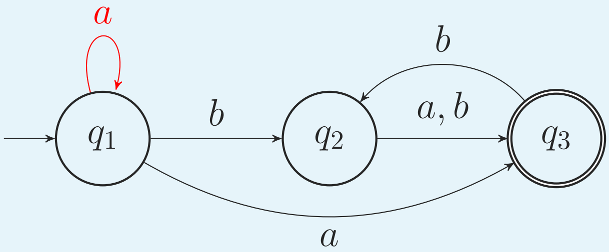
Le mot aba est « accepté » :



Automate

Définition 18 (L'acceptation d'un mot)

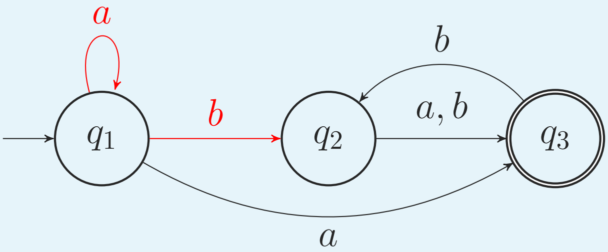
Le mot aba est « accepté » :



Automate

Définition 18 (L'acceptation d'un mot)

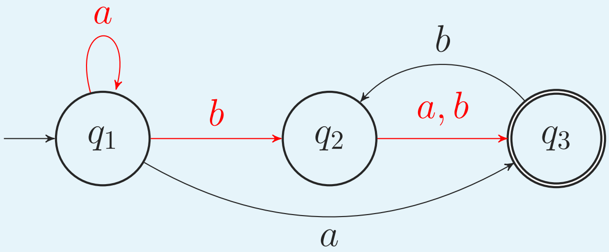
Le mot aba est « accepté » :



Automate

Définition 18 (L'acceptation d'un mot)

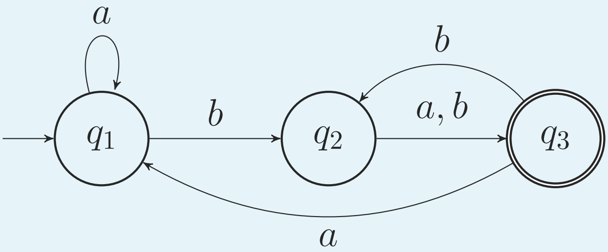
Le mot aba est « accepté » :



Automate

Définition 19 (Automate déterministe)

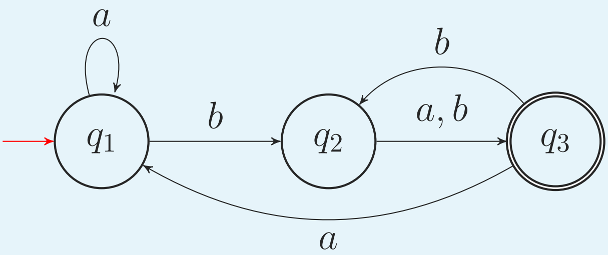
On parlera d'automate « déterministe » quand :



Automate

Définition 19 (Automate déterministe)

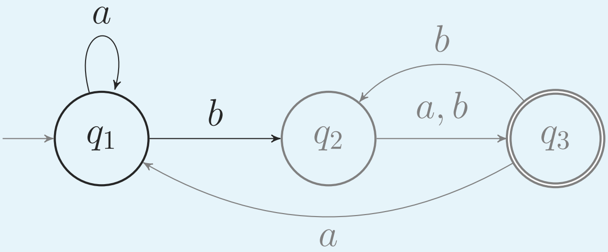
On parlera d'automate « déterministe » quand :



Automate

Définition 19 (Automate déterministe)

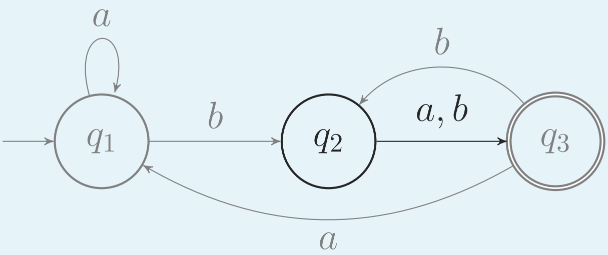
On parlera d'automate « déterministe » quand :



Automate

Définition 19 (Automate déterministe)

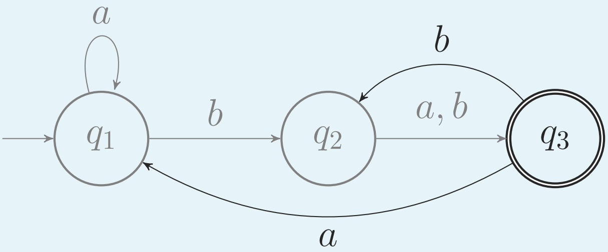
On parlera d'automate « déterministe » quand :



Automate

Définition 19 (Automate déterministe)

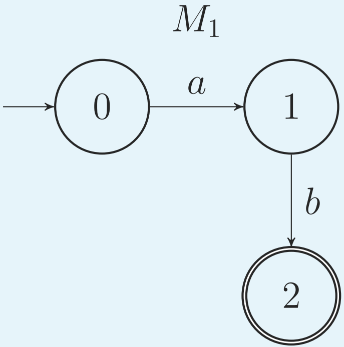
On parlera d'automate « déterministe » quand :



Automate

Définition 20 (Automate minimal)

L'automate M_1 est « minimal » du langage $L = \{ab\}$:



Lien entre les langages et les automates

Définition 21 (L'ensemble des langages reconnaisable)

On définira l'ensemble des langages de Σ^* « reconnaissable » par au moins un automate par $\mathrm{Rec}(\Sigma^*)$.

Lien entre les langages et les automates

Définition 21 (L'ensemble des langages reconnaisable)

On définira l'ensemble des langages de Σ^* « reconnaissable » par au moins un automate par $\mathrm{Rec}(\Sigma^*)$.

Théorème de *Kleene*

L'informaticien *Stephen C. Kleene* a montré en 1956 que :

$$\mathrm{Rec}(\Sigma^*) = \mathrm{Rat}(\Sigma^*)$$

Complexité en état

Définition 22

La « complexité en état » est une mesure d'un **langage**, elle est définie comme le nombre d'états de l'automate minimal du langage :

- Complexité en état déterministe (qu'on notera C_{det}).
- Complexité en état non déterministe (qu'on notera C_{ndet}).

Complexité en état

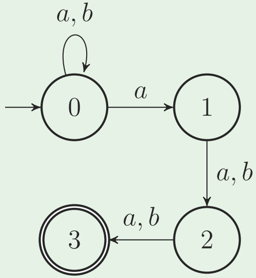
Exemple 23 (Définition d'une famille)

$$A_n = \Sigma^* \cdot a \cdot \Sigma^n$$

Complexité en état

Exemple 23 ($C_{\text{ndet}}(A_2) = 2 + 2$)

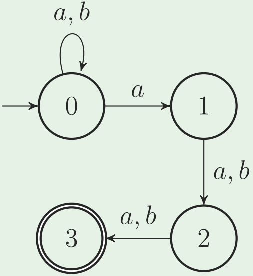
$$A_n = \Sigma^* \cdot a \cdot \Sigma^n$$

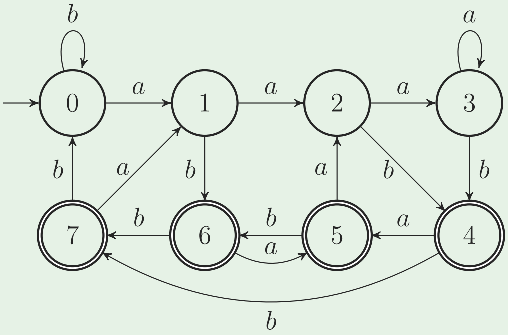


Complexité en état

Exemple 23 ($C_{\det}(A_2) = 2^{2+1}$)

$$A_n = \Sigma^* \cdot a \cdot \Sigma^n$$

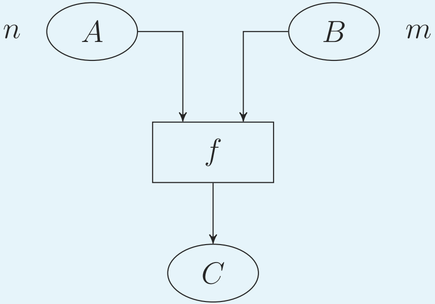




Complexité en état

Définition 24 (Complexité en état opérationnelle)

La fonction f à une complexité $g(n, m)$ si :



$$k \leq g(n, m)$$

Concaténation

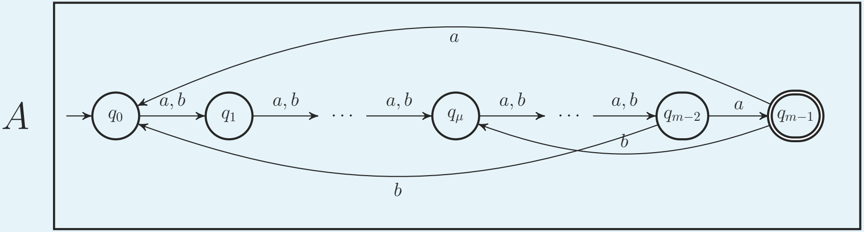
La concaténation

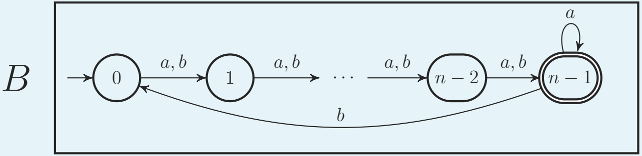
La complexité exacte de la concaténation

Nous allons montrer que la concaténation a une complexité déterministe « exacte » à $m2^n - 2^{n-1}$.

La concaténation

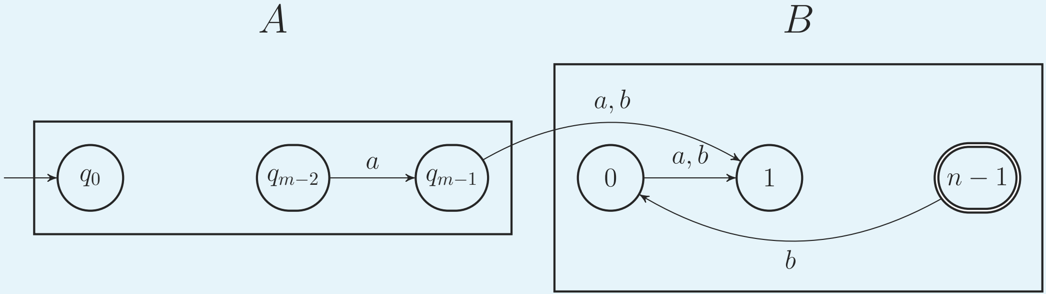
Définition des automes A et B ($\mu = (m - n + 1) \bmod n$)





La concaténation

Représentation de l'automate M



Conclusion

Conclusion

Or, la concaténation de deux langages rationnels à une complexité déterministe $m2^n - 2^{n-1}$. [2]

2. Yu 1997

Renverser

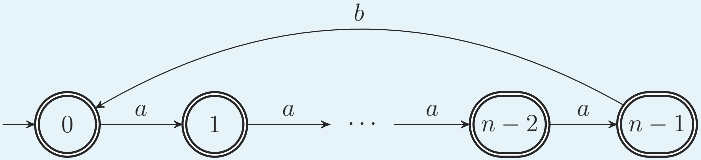
Renverser

La complexité exacte du renverser

Nous allons montrer que l'opération de « renverser » admet la complexité exacte de $n+1$ en considérant que l'automate résultant ne doit avoir qu'un seul état initial.

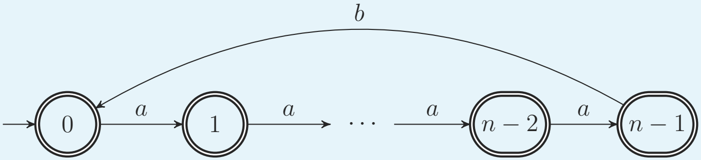
Renverser

Représentation de l'automate A



Renverser

Représentation de l'automate A



$M \in NFA(\Sigma)$

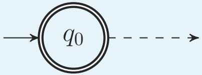
Soit l'automate minimal M qui reconnait le miroir du langage de l'automate A.

$$M = (Q, \{q_0\}, F, \delta)$$

Renverser

❶ Preuve

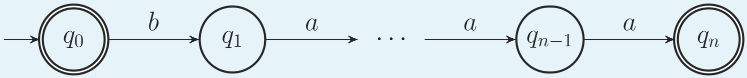
M doit nécessairement reconnaitre le mot vide, car l'automate A reconnait lui-même le mot vide.

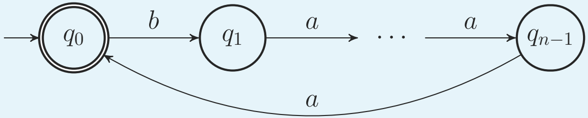


Renverser

❷ Preuve

L'automate A reconnait $a^{n-1}b$. Ainsi, il existe des états $q_1, q_2, \ldots, q_n$.

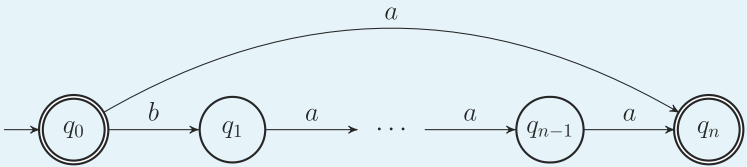




Renverser

❸ Preuve

M doit obligatoirement reconnaitre le mot a, mais pas le mot ba^n.



Renverser

Conclusion

Or, l'opération de renverser d'un langage rationnel à une complexité non déterministe de $n+1$.[3]

3. Holzer et Kutrib 2003

Conclusion

Conclusion

Conclusion

On a donc que pour un alphabet binaire :

- la concaténation de deux langages a une complexité de $m2^n - 2^{n-1}$,
- le renverser (à un état initial) d'un langage à une complexité de $(n+1)$,

Conclusion

Ouverture

On pourrait s'intéresser aux opérations :

Conclusion

Ouverture

On pourrait s'intéresser aux opérations :

- celle de permutation (*Twist*),

Conclusion

Ouverture

On pourrait s'intéresser aux opérations :

- celle de permutation (*Twist*),
- celle de grignotage (*Trognon*).

Bibliographie

- HOLZER, Markus et Martin KUTRIB (avr. 2003). « Nondeterministic Descriptional Complexity of Regular Languages ». In : *International Journal of Foundations of Computer Science* 14. DOI : 10.1142/S0129054103002199.
- JIRÁSKOVÁ, Galina (2005). « State complexity of some operations on binary regular languages ». In : *Theoretical Computer Science* 330.2. Descriptional Complexity of Formal Systems, p. 287-298. ISSN : 0304-3975. DOI : https://doi.org/10.1016/j.tcs.2004.04.011. URL : https://www.sciencedirect.com/science/article/pii/S0304397504006577.

Bibliographie

Yu, S. (1997). « Regular Languages ». In : *Handbook of Formal Languages, Vol. 1*. Sous la dir. de G. Rozenberg et A. Salomaa. Berlin, New York : Springer. Chap. 2, p. 41-110.