Complexité en état opérationnel de langage rationnel

Master 1 - ITA : Application Informatique
Encadré par Pascal Caron

Edouard HADDAG
Université de Rouen Normandie

3 juin 2025

Langage

Définition 1 (Langage)

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

Langage

Définition 1 (Langage)

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

Exemple 2

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

Langage

Définition 1 (Langage)

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

Exemple 2

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

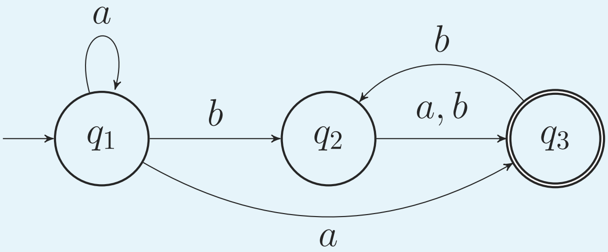
Définition 3 (Le langage de l'ensemble des mots)

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

Automate

Définition 4 (Automate)

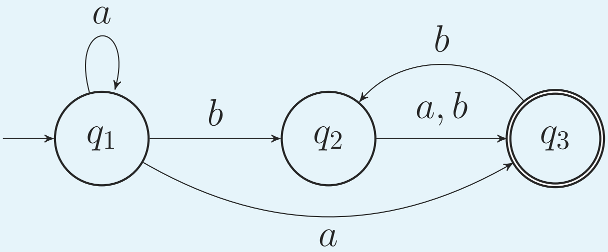
$M = (Q, I, F, \delta)$ avec $Q = \{q_1, q_2, q_3\}$, $I = \{q_1\}$ et $F = \{q_3\}$.



Automate

Définition 5 (L'acceptation d'un mot)

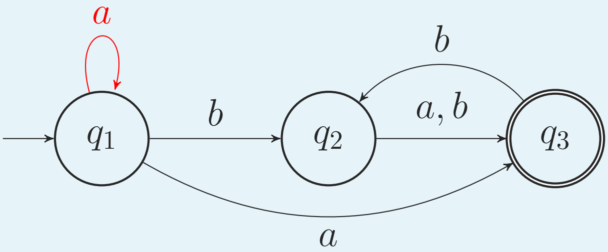
Le mot aba est « accepté » :



Automate

Définition 5 (L'acceptation d'un mot)

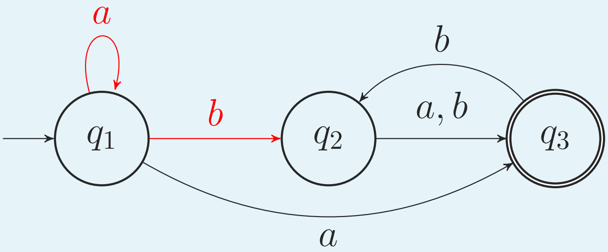
Le mot aba est « accepté » :



Automate

Définition 5 (L'acceptation d'un mot)

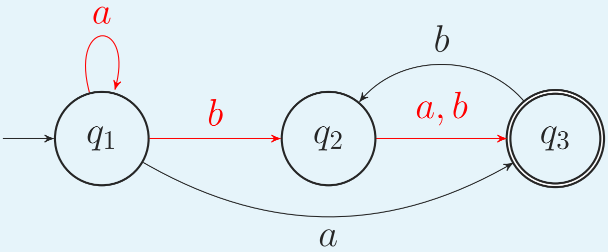
Le mot aba est « accepté » :



Automate

Définition 5 (L'acceptation d'un mot)

Le mot aba est « accepté » :

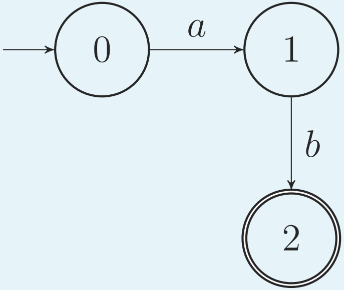


Automate

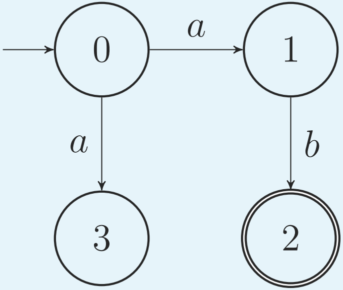
Définition 6 (Automate minimal)

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

M_1



M_2



Lien entre les langages et les automates

Définition 7 (Langage rationnel)

Les langages « rationnels » sont les langages reconnaissables par au moins un automate.

Complexité en état

Définition 8

La « complexité en état » est une mesure d'un **langage**, elle est définie comme le nombre d'états de l'automate minimal du langage. Qui sera noté $\mathcal{C}(L)$.

Complexité en état

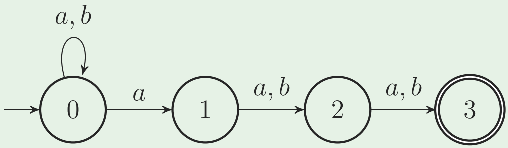
Exemple 9 (Définition d'une famille)

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

Complexité en état

Exemple 9 ($\mathcal{C}(A_2) = 2 + 2$)

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



Complexité en état

Définition 10 (Complexité en état opérationniel)

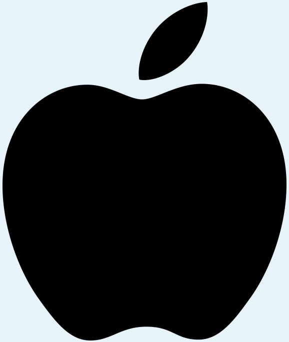
La fonction f à une complexité $g(n)$ si :

$$n \quad A \longrightarrow \boxed{f} \longrightarrow C \quad k \leq g(n)$$

Le trognon d'un langage

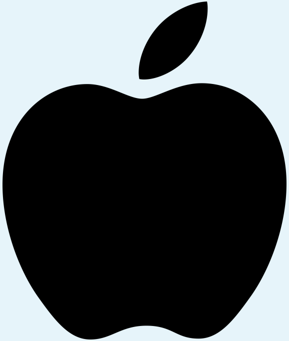
Définition

Le trognon d'un mot



Définition

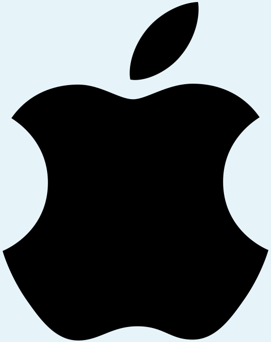
Le trognon d'un mot



$abcdcba$

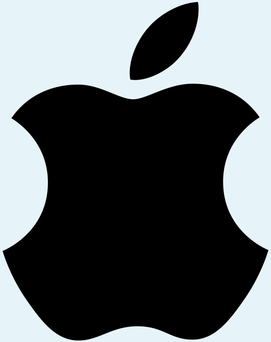
Définition

Le trognon d'un mot



Définition

Le trognon d'un mot



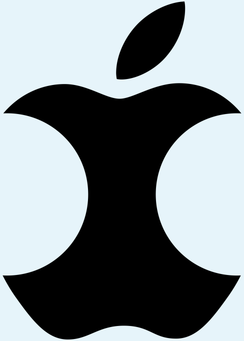
$$ab\cdot cdc\cdot ba$$

Définition

Le trognon d'un mot

Définition

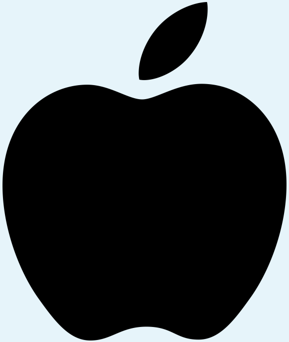
Le trognon d'un mot



$$abc \cdot d \cdot cba$$

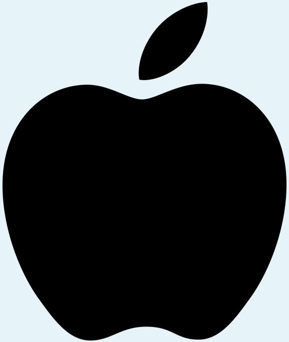
Définition

Le trognon d'un mot



Définition

Le trognon d'un mot



$$\varepsilon \cdot abcdcba \cdot \varepsilon$$

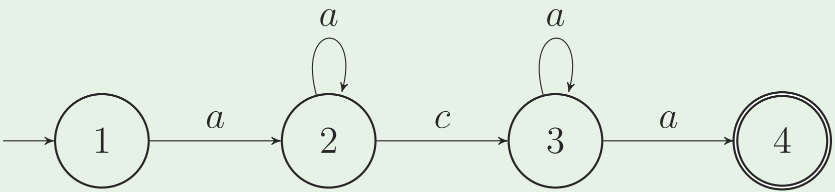
Définition

Définition 11 (Le trognon d'un mot)

Le trognon d'un langage sera alors l'union des trognons des mots qu'il le compose.

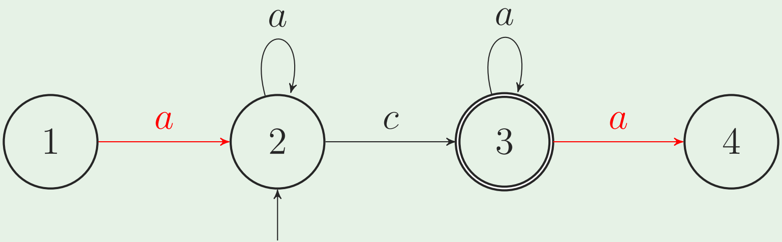
Algorithme de grignotage

Exemple 12 ($L(M) = a \cdot a^* \cdot c \cdot a^* \cdot a$)



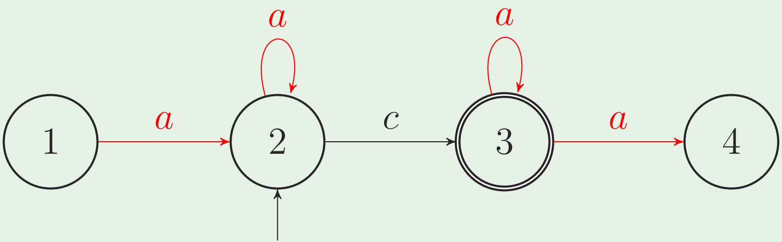
Algorithme de grignotage

Exemple 13 (M grignoté de a)



Algorithme de grignotage

Exemple 14 (M grignoté de aa)



Algorithme de grignotage

Définition 15

L'automate `nibbling`(M) sera donc l'union des automates grignotés distincts.

Algorithme de grignotage

Définition 15

L'automate `nibbling`(M) sera donc l'union des automates grignotés distincts.

Compléxité en état de notre algorithme

Le nombre d'états de l'union de deux automates est égal à la somme des nombres d'états des automates.

Algorithme de grignotage

Définition 15

L'automate `nibbling`(M) sera donc l'union des automates grignotés distincts.

Compléxité en état de notre algorithme

Le nombre d'états de l'union de deux automates est égal à la somme des nombres d'états des automates.

Les automates grignotés ont comme forme (Q, I, F, δ) avec Q et δ les mêmes et comme changement I et F qui sont deux ensembles non vides.

Algorithme de grignotage

Définition 15

L'automate `nibbling`(M) sera donc l'union des automates grignotés distincts.

Compléxité en état de notre algorithme

Le nombre d'états de l'union de deux automates est égal à la somme des nombres d'états des automates.

Les automates grignotés ont comme forme (Q, I, F, δ) avec Q et δ les mêmes et comme changement I et F qui sont deux ensembles non vides.

Ce qui nous donne comme complexité finale : $n(2^n - 1)^2$ états.

Langage permuté

Définition

Définition 16 ($Twist(L)$)

Le langage $Twist(L)$ sera défini comment l'ensemble des mots du langage de L tel qu'on aura échangé les lettres aux indices $2k$ et $2k+1$ avec $k \in \mathbb{N}$.

Définition

Définition 16 ($Twist(L)$)

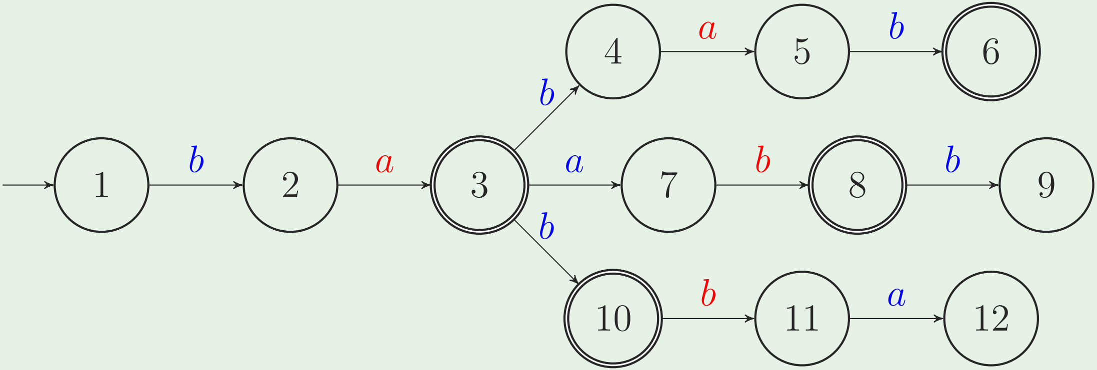
Le langage $Twist(L)$ sera défini comment l'ensemble des mots du langage de L tel qu'on aura échangé les lettres aux indices $2k$ et $2k+1$ avec $k \in \mathbb{N}$.

Exemple 17

$$L = \{\varepsilon, a, ab, abcd\}$$
$$Twist(L) = \{\varepsilon, a, ba, badc\}$$

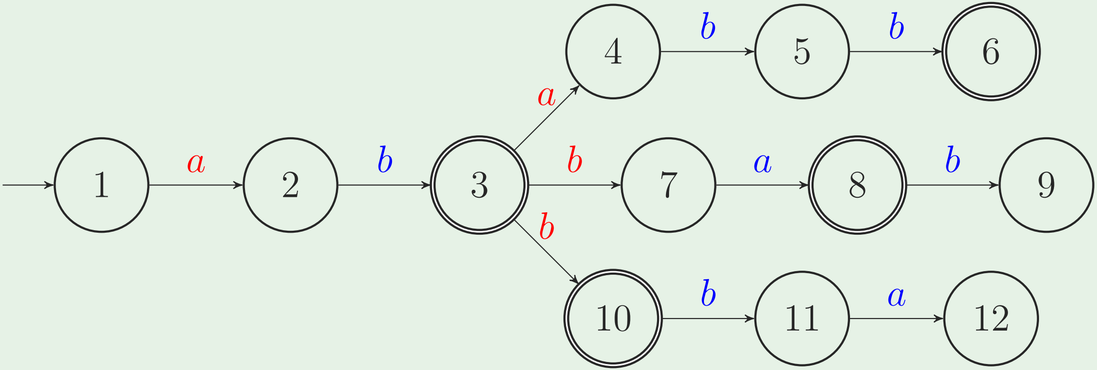
Algorithme twister

Exemple 18 (Soit l'automate M suivant)



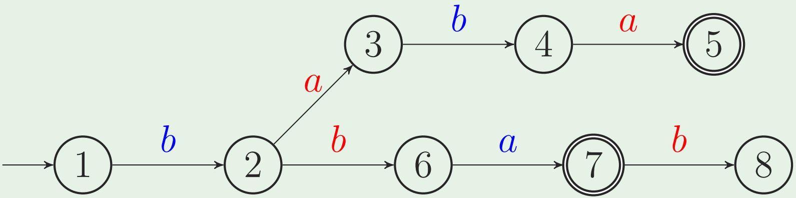
Algorithme twister

Exemple 18 (Soit l'automate M suivant)



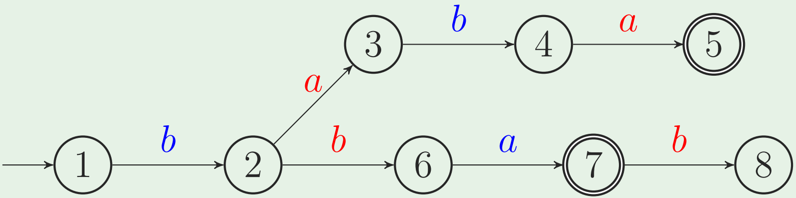
Algorithme twister

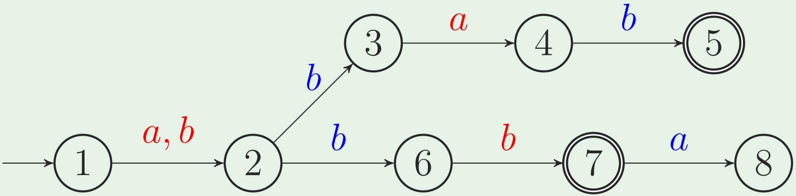
Exemple 19 (Soit l'automate N suivant)



Algorithme twister

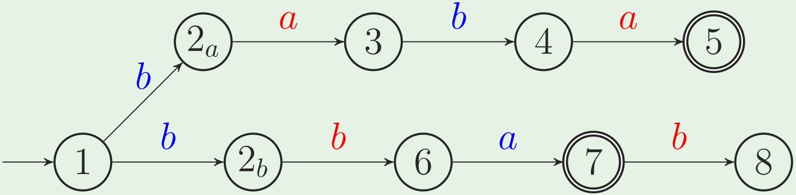
Exemple 19 (Soit l'automate N suivant)





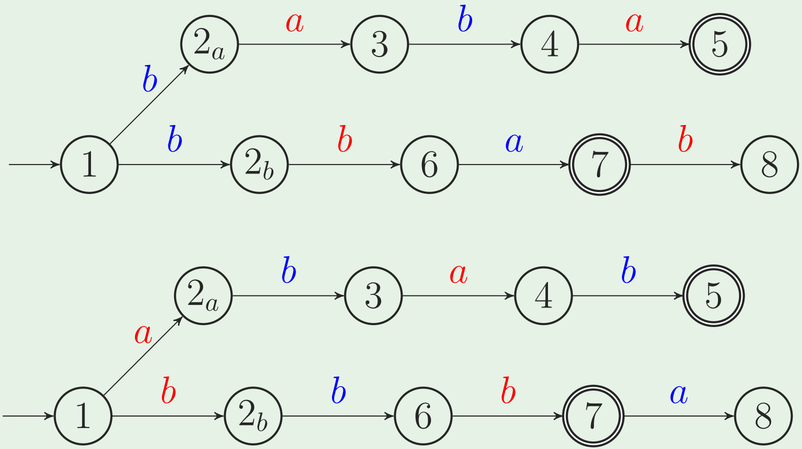
Algorithme twister

Exemple 19 (Soit l'automate N suivant)



Algorithme twister

Exemple 19 (Soit l'automate N suivant)



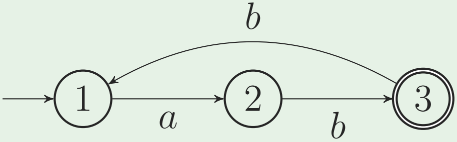
Algorithme twister

Automate quelconque

Si on ajoute les cycles, on devra alors supposer que tous nœuds intermédiaire se trouve dans le cas du nœud 2, on devra donc le dupliquer.

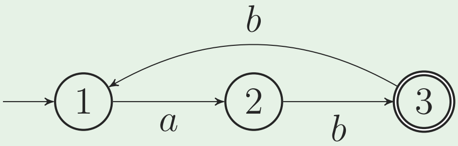
Algorithme twister

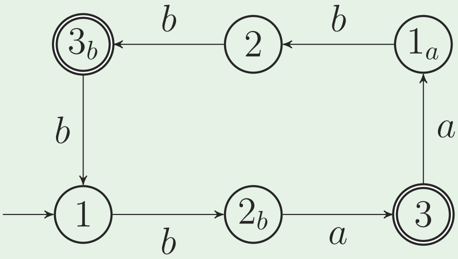
Exemple 20



Algorithme twister

Exemple 20





Algorithme twister

Compléxité en état de notre algorithme

Dans le pire cas, notre algorithme dupliquera tous les états de l'automate.

Algorithme twister

Compléxité en état de notre algorithme

Dans le pire cas, notre algorithme dupliquera tous les états de l'automate.

Sachant que chaque état peut être dupliqué étant de voir qu'il y a de symbole de l'alphabet plus un.

Algorithme twister

Compléxité en état de notre algorithme

Dans le pire cas, notre algorithme dupliquera tous les états de l'automate.

Sachant que chaque état peut être dupliqué étant de voir qu'il y a de symbole de l'alphabet plus un.

Alors, la complexité dans le pire cas de notre algorithme est $n(|\Sigma| + 1)$.

Conclusion

Conclusion

Conclusion

On vient donc de faire deux algorithmes qui permettent de calculer les automates reconnaissant le trognon et le langage permuté.

Conclusion

Conclusion

On vient donc de faire deux algorithmes qui permettent de calculer les automates reconnaissant le trognon et le langage permuté.

Pour autant ça ne veut pas dire que ces opérations sur les langages ont les mêmes complexités que nos algorithmes, ce n'est qu'une borne supérieure.