INF 302 : Langages & Automates

Chapitre 4 : Opérations sur les automates déterministes et fermeture des langages à états

Yliès Falcone

ylies.falcone@univ-grenoble-alpes.fr — www.ylies.fr

Univ. Grenoble-Alpes, Inria

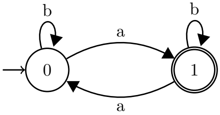
-

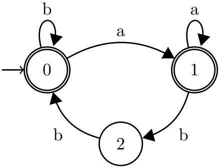
Laboratoire d'Informatique de Grenoble - www.liglab.fr

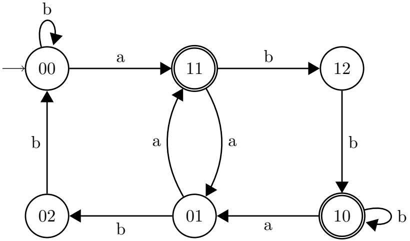
Équipe de recherche LIG-Inria, Corse - team.inria.fr/corse/

Année Académique 2021 - 2022

Intuition et objectifs







- Correspondance entre opérations sur les langages et opérations associées sur les automates.

opérations sur automate	**opérations sur langage**
négation	complémentation
produit	intersection

- Fermeture de l'ensemble des langages à états.

Fermeture de EF par complémentation et intersection

Fermeture de EF par **complémentation**

Soit A un AD.

1. Le langage $\Sigma^* \setminus L(A)$ est-il reconnaissable par un AD ?
2. Si oui, peut-on construire de manière effective un automate qui reconnaît $\Sigma^* \setminus L(A)$?

Fermeture de EF par **intersection**

Soient A et B deux ADs.

1. Le langage $L(A) \cap L(B)$ est-il reconnaissable par un AD ?
2. Si oui, peut-on construire de manière effective un automate qui reconnaît $L(A) \cap L(B)$?

Nous pourrons répondre de manière *affirmative* à toutes ces questions.

Plan Chap. 4 - Opérations sur les automates déterministes et fermeture des langages à états

Plan Chap. 4 - Opérations sur les automates déterministes et fermeture des langages à états

Complétion d'automates

Intuition

Soit $A = (Q, \Sigma, q_{\text{init}}, \delta, F)$ un AD qui reconnaît un langage (noté $L(A)$).

Effet de la complétion : un *AD **complet*** qui reconnaît $L(A)$.

Objectifs de la complétion :

- travailler avec des automates complets pour certaines transformations,
- raisonner sur des automates complets est parfois plus simple.

Idée de la complétion :

1. Ajouter un *nouvel état* à Q. Cet état est appelé *état puits*.
2. Diriger toutes les transitions non définies dans A vers l'état puits.

◁ Complétion d'un automate

Complétion d'automates

Définition

Soit $A = (Q, \Sigma, q_{\mathrm{init}}, \delta, F)$ un AD qui reconnaît un langage (noté $L(A)$).

Définition (Complétion d'un automate)

L'automate *complété* de A est $C(A) = (Q \cup \{q_p\}, \Sigma, q_{\mathrm{init}}, C(\delta), F)$ tel que

- $q_p \notin Q$ et
- $C(\delta) : Q \cup \{q_p\} \times \Sigma \to Q \cup \{q_p\}$ est une *application* définie par :

$$C(\delta)(q, a) \stackrel{\mathrm{def}}{=} \begin{cases} \delta(q, a) & \text{pour tout } (q, a) \in \mathrm{dom}(\delta) \\ q_p & \text{sinon} \end{cases}$$

Correction de l'opération de complétion

$$L(A) = L(C(A))$$

Idée de la preuve

Montrer que A et $C(A)$ acceptent les mêmes mots en considérant les exécutions des automates. Voir exercices.

Plan Chap. 4 - Opérations sur les automates déterministes et fermeture des langages à états

Négation d'un automate

Soit $A = (Q, \Sigma, q_{\text{init}}, \delta, F)$ un AD *complet*.

Définition (Négation d'un AD complet)

La négation de A est l'automate $A^c = (Q, \Sigma, q_{\text{init}}, \delta, Q \setminus F)$.

L'opération de négation est aussi appelée opération de complémentation.

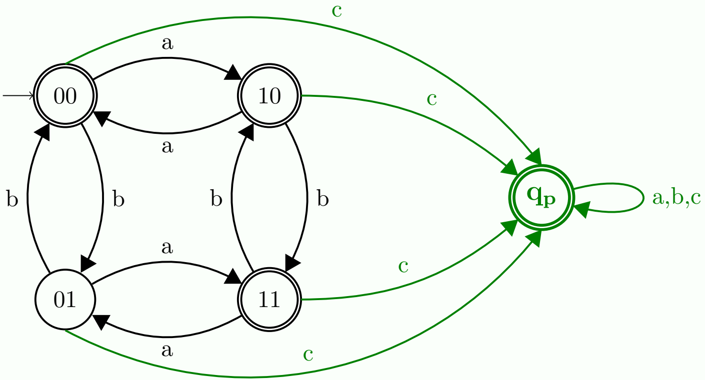
Procédure de complémentation d'un AD (quelconque) A :

1. **Construire** $C(A)$ — la version complète de A.
2. Inverser les états accepteurs et non-accepteurs dans $C(A)$.

Négation d'un automate : exemple

Exemple (Négation d'un automate)

Sur $\Sigma = \{a, b, \mathbf{c}\}$:
un nombre **impair** de *a* ou un nombre **pair** de *b*
ou un *c*



Correction de l'opération de négation et fermeture par complémentation

Soit $A = (Q, \Sigma, q_{\mathrm{init}}, \delta, F)$ un AD *complet*.

Correction de la procédure de complémentation

$$L(A^c) = \Sigma^* \setminus L(A).$$

Idée de la preuve

Montrer qu'un mot accepté par A n'est pas accepté par A^c et vice-versa en utilisant l'exécution de ces mots. Voir exercices.

Corollaire

La classe EF des langages à états est fermée par **complémentation**.

Plan Chap. 4 - Opérations sur les automates déterministes et fermeture des langages à états

Produit d'automates

Considérons deux ADs : $A = (Q^A, \Sigma, q^A_{\rm init}, \delta^A, F^A)$ et $B = (Q^B, \Sigma, q^B_{\rm init}, \delta^B, F^B)$.

Objectifs du produit d'automate :

- construire un automate qui accepte les mots reconnus par les deux automates (*à la fois*). Le langage reconnu par l'automate produit est donc l'*intersection* des langages des automates passés en paramètres ;
- réaliser cette construction de manière *compositionnelle*.

◁ Produit de deux automates

Définition (Produit d'automates)

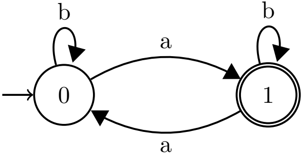
L'automate produit de A et de B est $A \times B = (Q, \Sigma, q_{\rm init}, \delta, F)$ où :

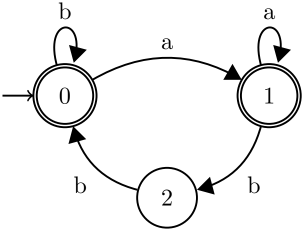
- $Q = Q^A \times Q^B$
- $q_{\rm init} = (q^A_{\rm init}, q^B_{\rm init})$
- $\delta : (Q^A \times Q^B) \times \Sigma \rightarrow (Q^A \times Q^B)$ est telle que :

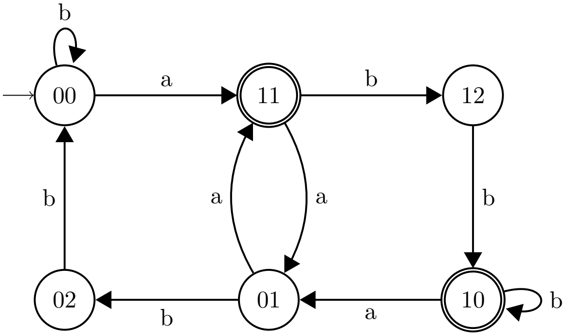
$$\delta\big((q^A, q^B), a\big) = \big(\delta^A(q^A, a), \delta^B(q^B, a)\big)$$

- $F = F^A \times F^B$.

Produit d'automates : exemple







Correction de l'opération de produit et fermeture par intersection

Soient $A = (Q^A, \Sigma, q_0^A, \delta^A, F^A)$ et $B = (Q^B, \Sigma, q_0^B, \delta^B, F^B)$ deux ADs.

Correction de l'opération de produit

$$L(A \times B) = L(A) \cap L(B).$$

Idée de la preuve

Pour montrer $L(A \times B) = L(A) \cap L(B)$, on doit montrer :

1. $L(A \times B) \subseteq L(A) \cap L(B)$, cad :
   - $L(A \times B) \subseteq L(A)$,
   - $L(A \times B) \subseteq L(B)$ et
2. $L(A) \cap L(B) \subseteq L(A \times B)$

Pour montrer 1), à partir de l'exécution d'un mot accepté par $A \times B$, déduire l'exécution sur A et B.
Pour montrer 2), construire l'exécution sur $A \times B$ d'un mot accepté par A et par B, puis utiliser les critères d'acceptation.
Voir exercices.

Corollaire

La classe EF des langages à états est fermée par **intersection**.

Plan Chap. 4 - Opérations sur les automates déterministes et fermeture des langages à états

Résumé du chapitre 4 : opérations sur les automates et fermeture des langages à états

Opérations sur les automates déterministes et fermeture des langages à états

- Calcul de l'automate *complété* (même langage).
- Calcul de l'automate *complémentaire* (complémentaire d'un langage).
- Calcul de l'automate *produit* de deux automates (intersection de langages).
- *Fermeture* des langages à états par *complémentation* et *intersection*.

En TD

- Donner les algorithmes pour procédures de complétion et complémentation (après le chapitre 5).
- Définir des procédures permettant de calculer des automate reconnaissant l'union et le ou exclusif des langages d'automates passés en paramètres.
- ...