Chapitre 3
Évaluation des expressions régulières et automates finis

Jean Privat

Université du Québec à Montréal

INF5000 — Théorie et construction des compilateurs
Automne 2013

Évaluation d'une expression régulière

Soit une expression régulière définissant un langage

- Une chaîne appartient-elle au langage ?
- Rechercher les sous-chaînes appartenant au langage ?

Questions non triviales

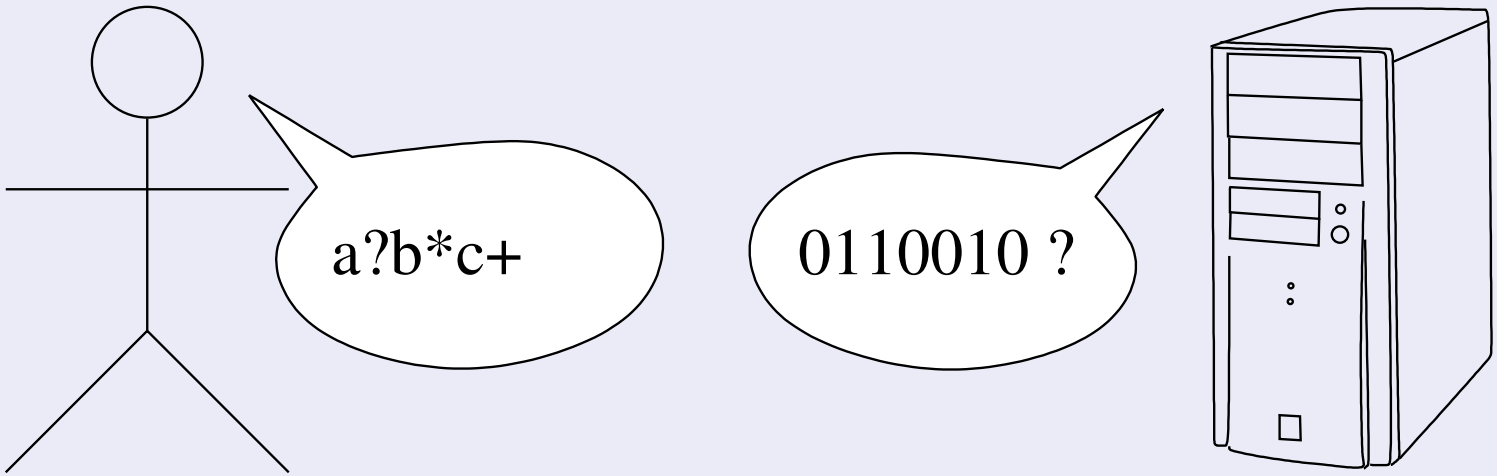
Évaluation d'une expression régulière

Soit une expression régulière définissant un langage

- Une chaîne appartient-elle au langage ?
- Rechercher les sous-chaînes appartenant au langage ?

Questions non triviales

- Même pour un ordinateur



Approche naïve

Comment évaluer ?

- `a*ba*ba*`
- `(ab|b)*a?`

Approche naïve

Comment évaluer ?

- `a*ba*ba*`
- `(ab|b)*a?`

Les langages sont-ils les mêmes ?

- `a+ba*|a*ba+`
- `a+ba+|a+b|ba+`

Approche naïve

Comment évaluer ?

- `a*ba*ba*`
- `(ab|b)*a?`

Les langages sont-ils les mêmes ?

- `a+ba*|a*ba+`
- `a+ba+|a+b|ba+`

Quel est l'expression régulière la plus rapide à évaluer ?

- Qu'est-ce qui impacte les performances ?

Outils nécessaires

Structures de données

- Automates finis

Algorithmes

- Transformation d'automates
- Évaluation d'automates

Automate = Graphe

Transitions = arcs

- Orientés
- Étiquetés par un caractère de l'alphabet ou par ε (epsilon)

États = nœuds

- Un état de départ
- Un ensemble d'états d'acceptation (éventuellement vide)

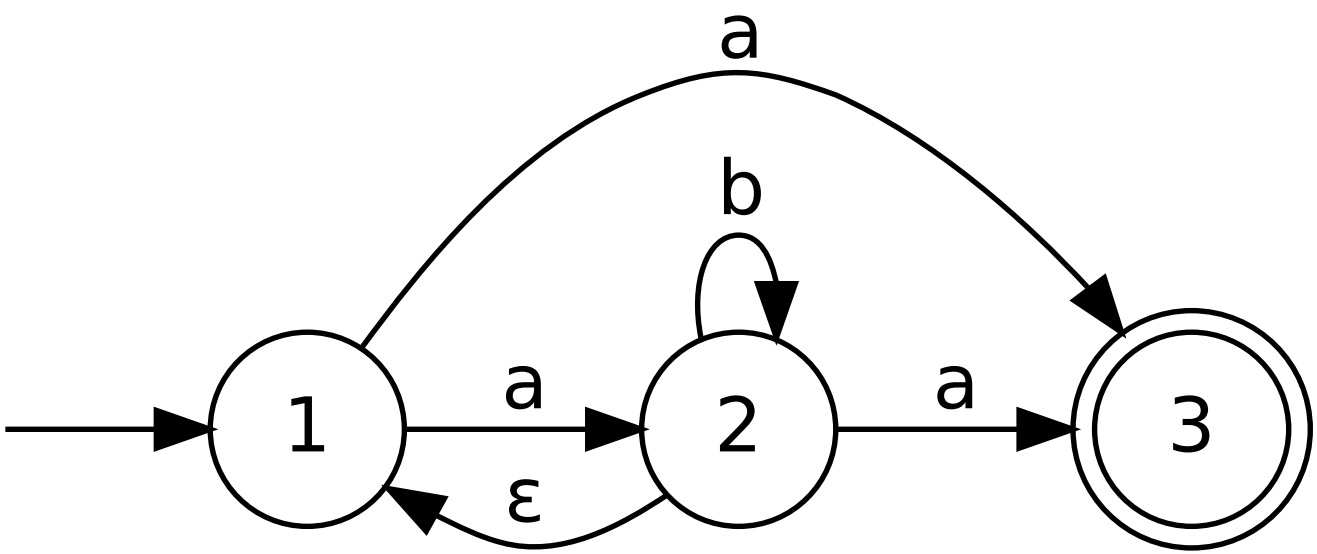
Automate fini

- Nombre fini d'états (et de transitions)

Automate fini non déterministe (NFA)

Règle : pas de règle

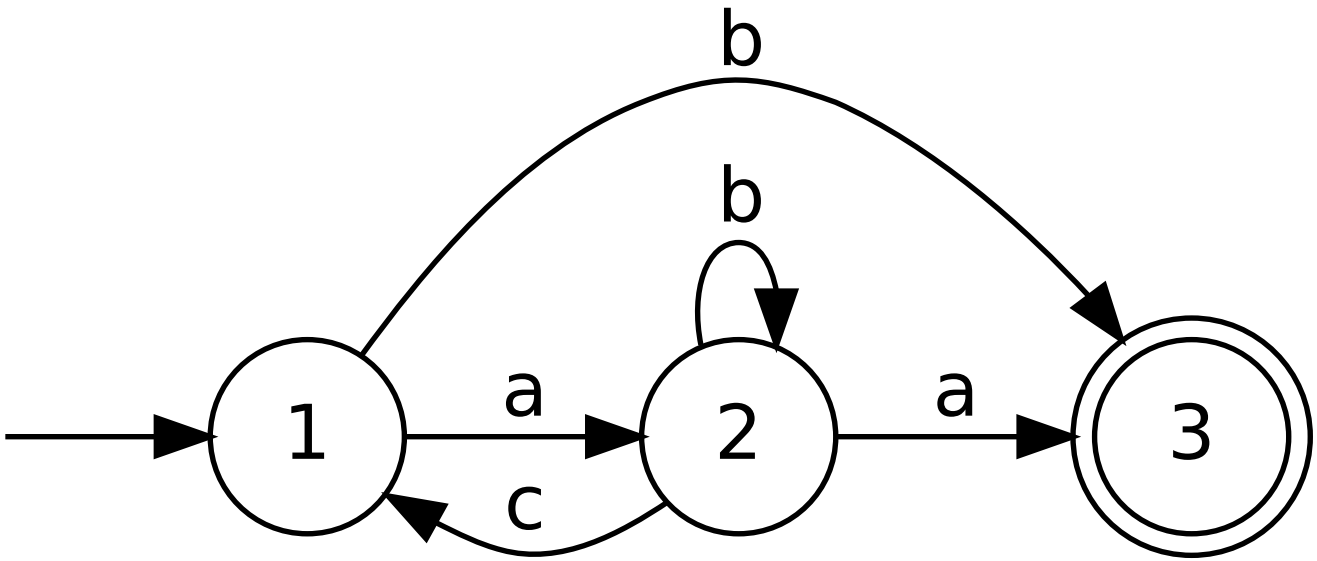
- Pas de restriction sur les étiquettes des arcs



Automate fini déterministe (DFA)

Règles

- Au plus un arc sortant pour une étiquette
- Pas d'étiquette ε



Langages réguliers

NFA et DFA définissent un langage

- L'ensemble des chemins partant d'un état de départ vers un état d'acceptation

NFA, DFA et expression régulières

- Reconnaissent la même classe de langages :
  $\Rightarrow$ les langages réguliers

Évaluation d'automates

Soit un automate fini définissant un langage

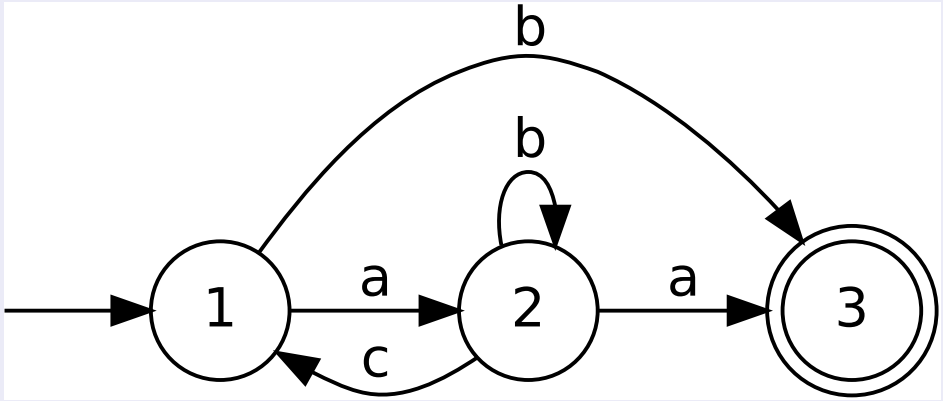
- Une chaîne appartient-elle au langage ?

Facile

- Il suffit de trouver un chemin
- Encore plus facile avec un DFA (algorithme linéaire)

Évaluation d'automates : Exercice

Soit le DFA

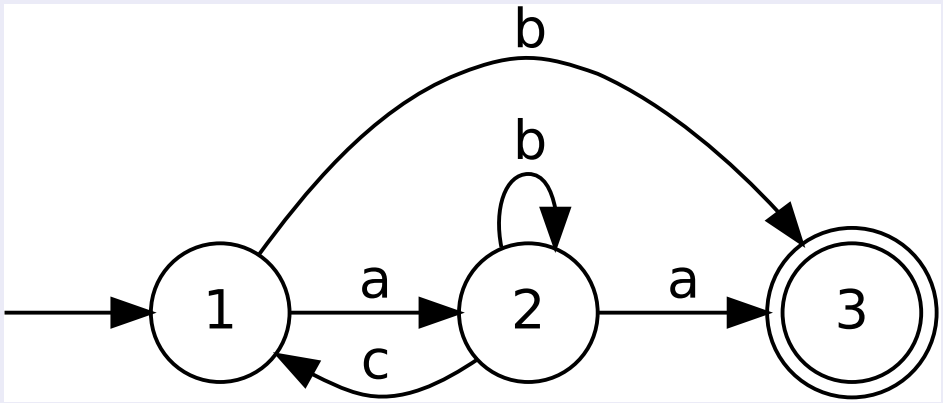


Quelles chaînes sont reconnues parmi

- aa
- acabcb
- acc
- abbc

Évaluation d'automates : Exercice

Soit le DFA



Quelles chaînes sont reconnues parmi

- aa : $1 \xrightarrow{a} 2 \xrightarrow{a} 3 \xrightarrow{\$} OK$
- acabcb : $1 \xrightarrow{a} 2 \xrightarrow{c} 1 \xrightarrow{a} 2 \xrightarrow{b} 2 \xrightarrow{c} 1 \xrightarrow{b} 3 \xrightarrow{\$} OK$
- acc : $1 \xrightarrow{a} 2 \xrightarrow{c} 1 \xrightarrow{c}$ PAS OK
- abbc : $1 \xrightarrow{a} 2 \xrightarrow{b} 2 \xrightarrow{b} 2 \xrightarrow{c} 1 \xrightarrow{\$}$ PAS OK

Évaluation d'expression régulières

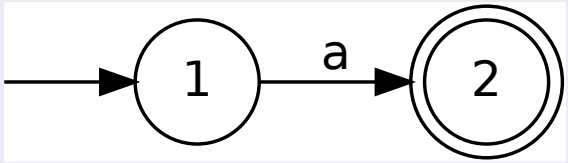
Trois étapes

- Transformation RE $\rightarrow$ NFA
- Transformation NFA $\rightarrow$ DFA
- Évaluation du DFA

Transformation d'expression régulières $\rightarrow$ NFA

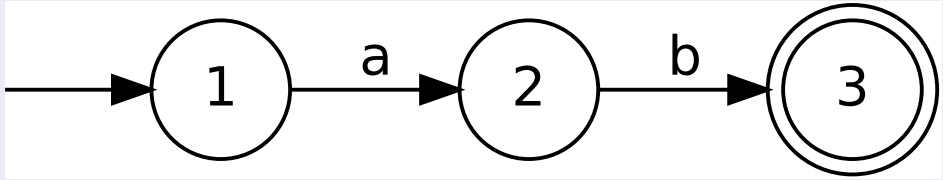
Atome (caractère ou ε)

- a



Concaténation

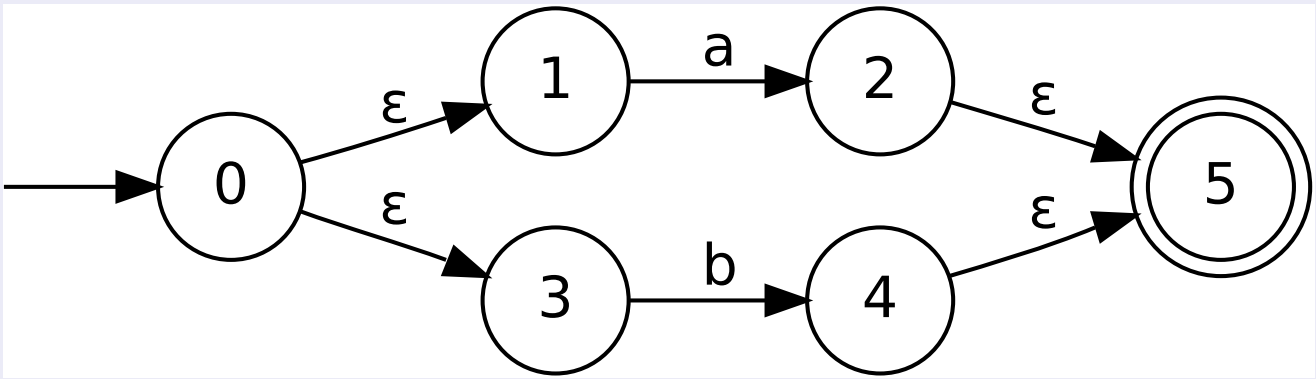
- ab



Transformation d'expression régulières → NFA

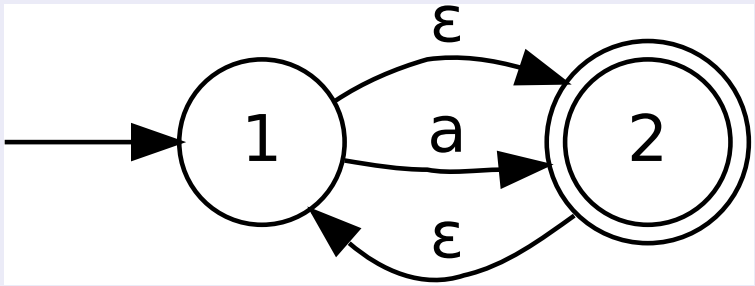
Alternation

- a|b



Étoile

- a*



RE $\rightarrow$ NFA : Exercice 1

Écrire le NFA de l'expression régulière

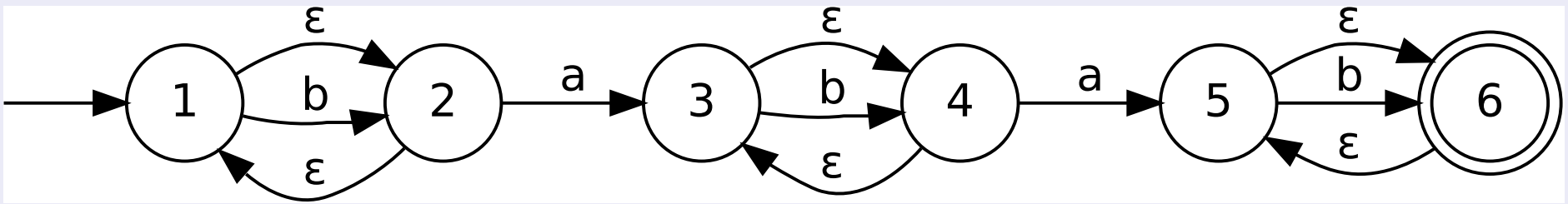
- `b*ab*ab*`

RE $\rightarrow$ NFA : Exercice 1

Écrire le NFA de l'expression régulière

- b*ab*ab*

Solution



RE $\rightarrow$ NFA : Exercice 2

Écrire le NFA de l'expression régulière

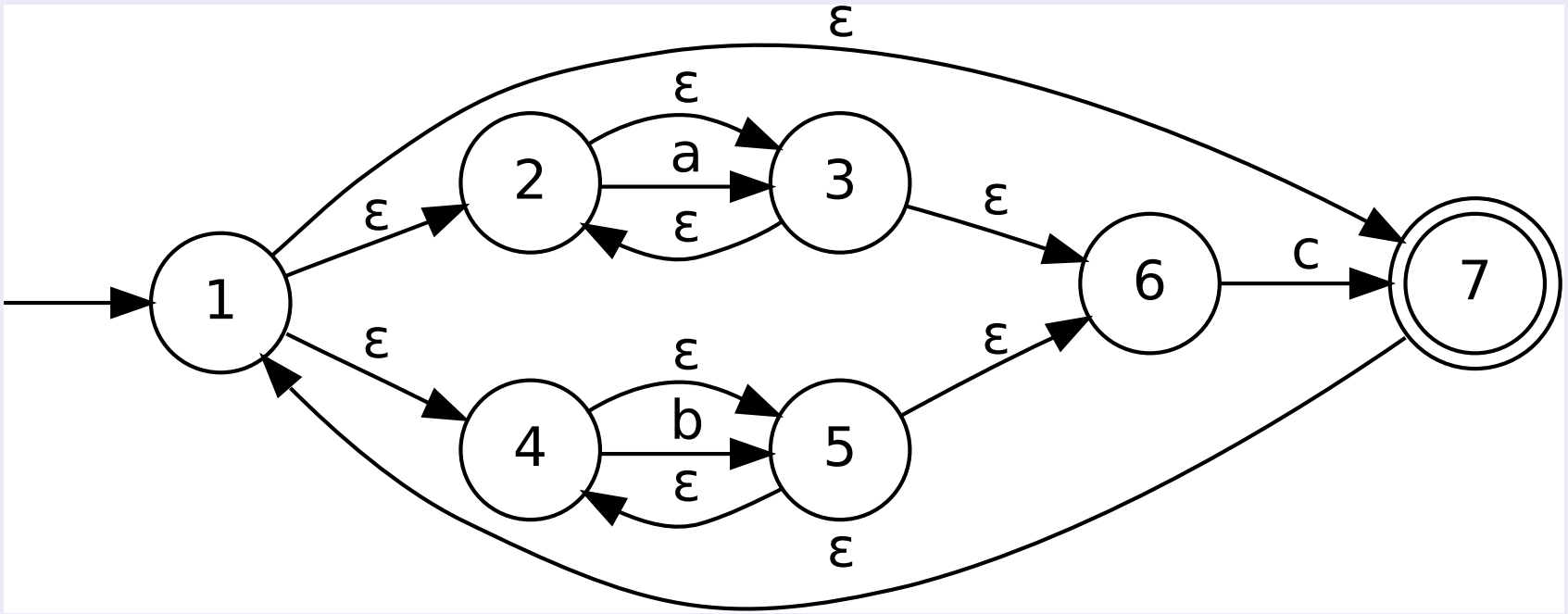
- `((a*|b*)c)*`

RE $\rightarrow$ NFA : Exercice 2

Écrire le NFA de l'expression régulière

- `((a*|b*)c)*`

Solution



RE $\rightarrow$ NFA : Exercice 3

Écrire le NFA de l'expression régulière

- `a(bc)?d`

RE $\rightarrow$ NFA : Exercice 3

Écrire le NFA de l'expression régulière

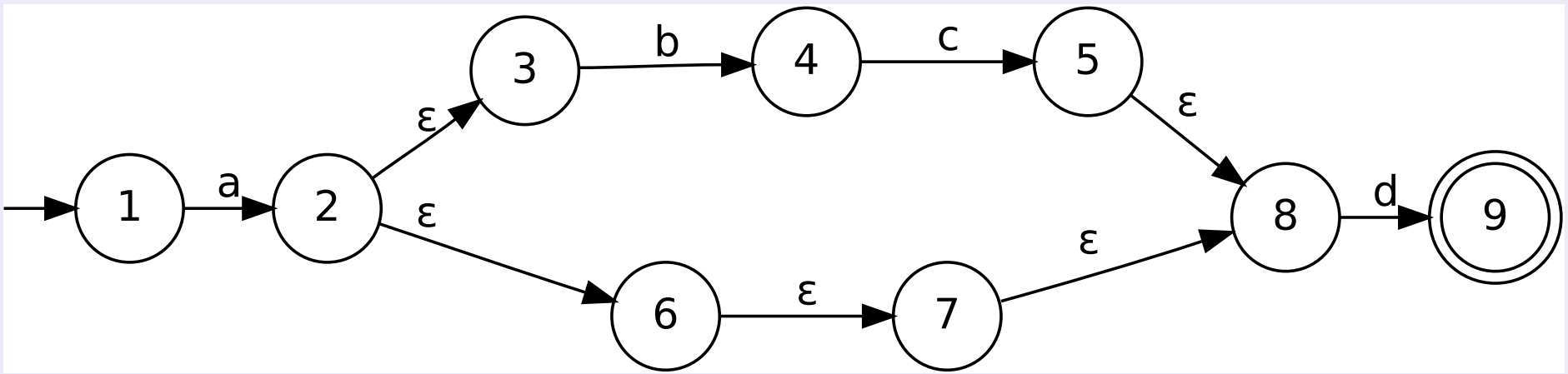
- `a(bc)?d` $\equiv$ `a(bc|)d`

RE $\rightarrow$ NFA : Exercice 3

Écrire le NFA de l'expression régulière

- `a(bc)?d` $\equiv$ `a(bc|)d`

Solution



Transformation NFA $\rightarrow$ DFA

Idée

- Simuler en parallèle tous les chemins
  $\Rightarrow$ un état du DFA $\approx$ n états du NFA.

Risque

- Au pire, DFA exponentiellement plus grand que NFA
- Mais suffisamment rare en pratique

Outils sur les NFA

$\varepsilon fermeture(E)$

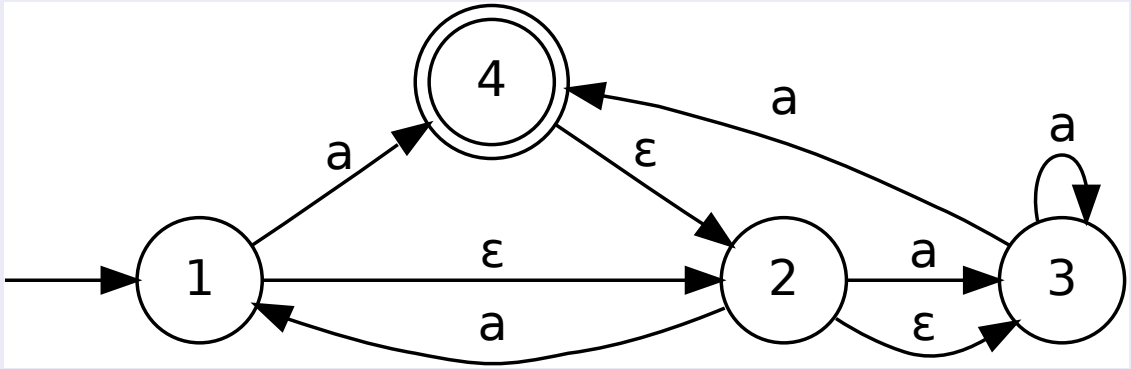
- L'ensemble des états atteignable par 0, 1, ou plusieurs transitions ε à partir d'un état de E.

$trans(E, c)$

- L'ensemble des états atteignable par une seule transition c à partir d'un état de E.

Outils sur les NFA : Exercice

Soit le NFA

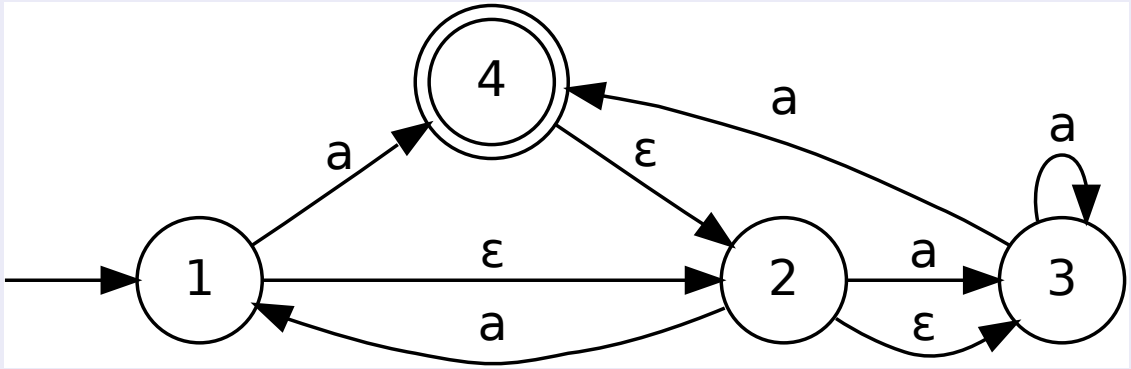


Déterminer $F = \varepsilon fermeture(E)$ et $T = trans(E, \mathrm{a})$ pour les ensembles E suivants

- $E = \{1\}$
- $E = \{1, 2\}$
- $E = \{3\}$
- $E = \{4\}$

Outils sur les NFA : Exercice

Soit le NFA



Déterminer $F = \varepsilon fermeture(E)$ et $T = trans(E, \mathrm{a})$ pour les ensembles E suivants

- $E = \{1\} : F = \{1, 2, 3\}\,;\ T = \{4\}$
- $E = \{1, 2\} : F = \{1, 2, 3\}\,;\ T = \{1, 3, 4\}$
- $E = \{3\} : F = \{3\}\,;\ T = \{3, 4\}$
- $E = \{4\} : F = \{2, 3, 4\}\,;\ T = \emptyset$

NFA $\rightarrow$ DFA : Algorithme

Données : Un NFA N

Résultat : Un DFA D définissant le même langage que N

$E = \varepsilon fermeture(depart(N))$;

ajouter E comme état de départ de D (sans le marquer);

tant que un état E de D est non marqué **faire**

  marquer E;

  pour chaque caractère c de l'alphabet **faire**

    $F = \varepsilon fermeture(trans(E, c))$;

    si F n'est pas un état de D **alors**

      ajouter l'état F à D (sans le marquer);

      si un élément de F est un état d'acceptation de N **alors**

      F est un état d'acceptation de D

    fin

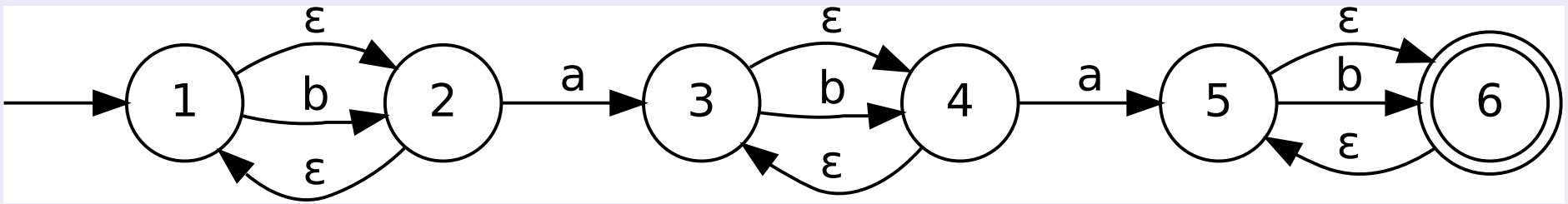
    ajouter la transition $E \xrightarrow{c} F$ à D;

  fin

fin

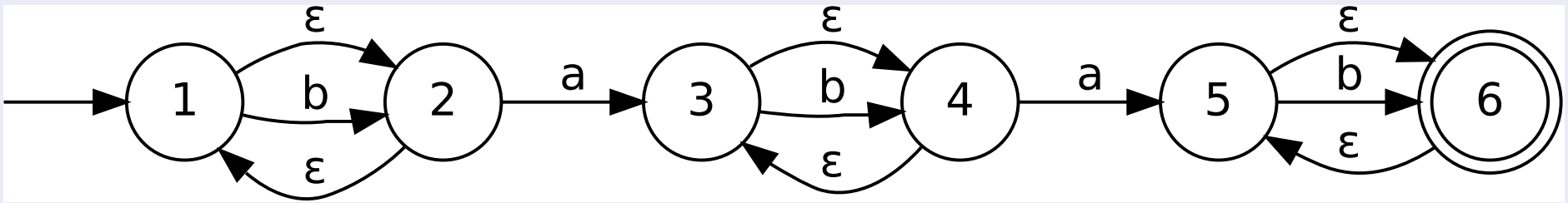
NFA $\rightarrow$ DFA : Exercice 1

Transformer en DFA le NFA

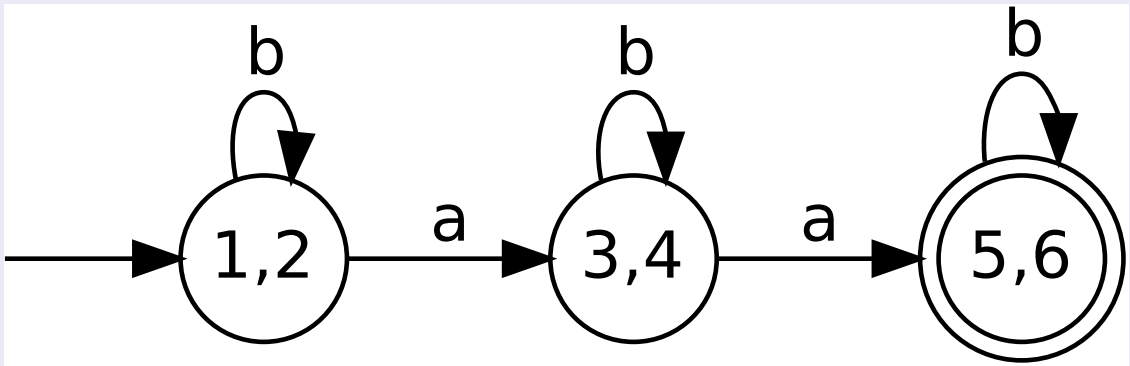


NFA $\rightarrow$ DFA : Exercice 1

Transformer en DFA le NFA

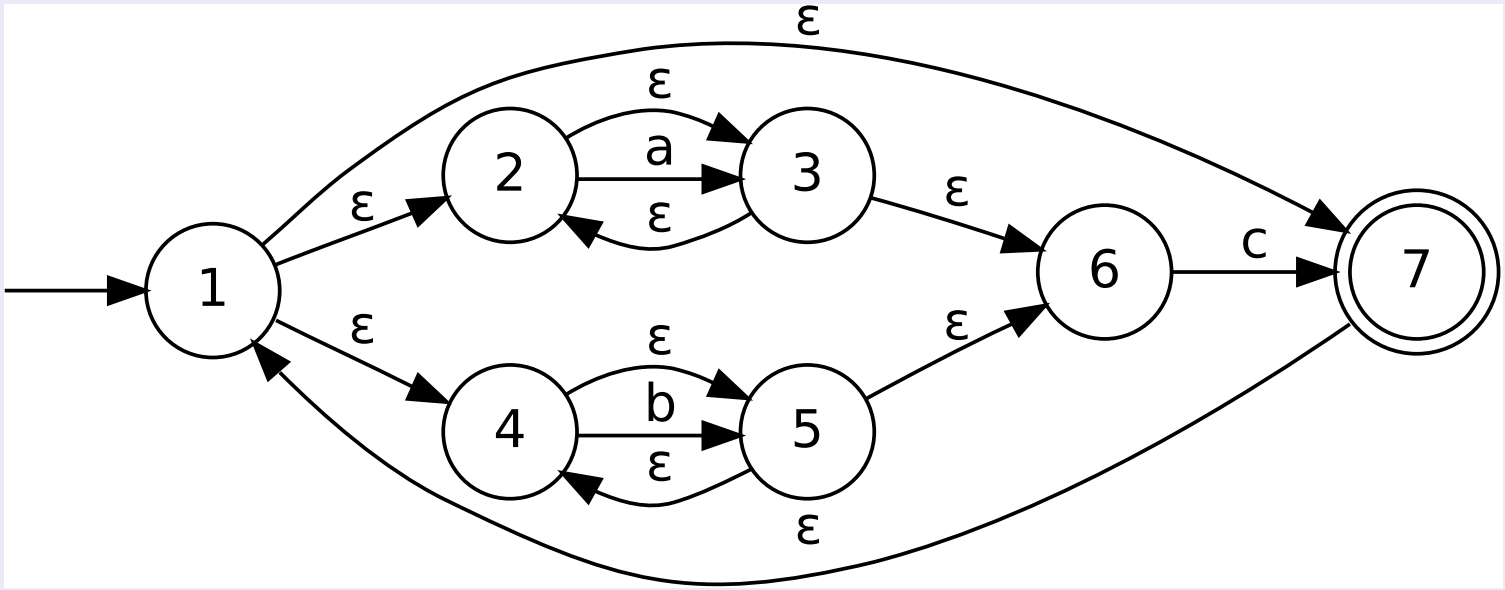


Solution



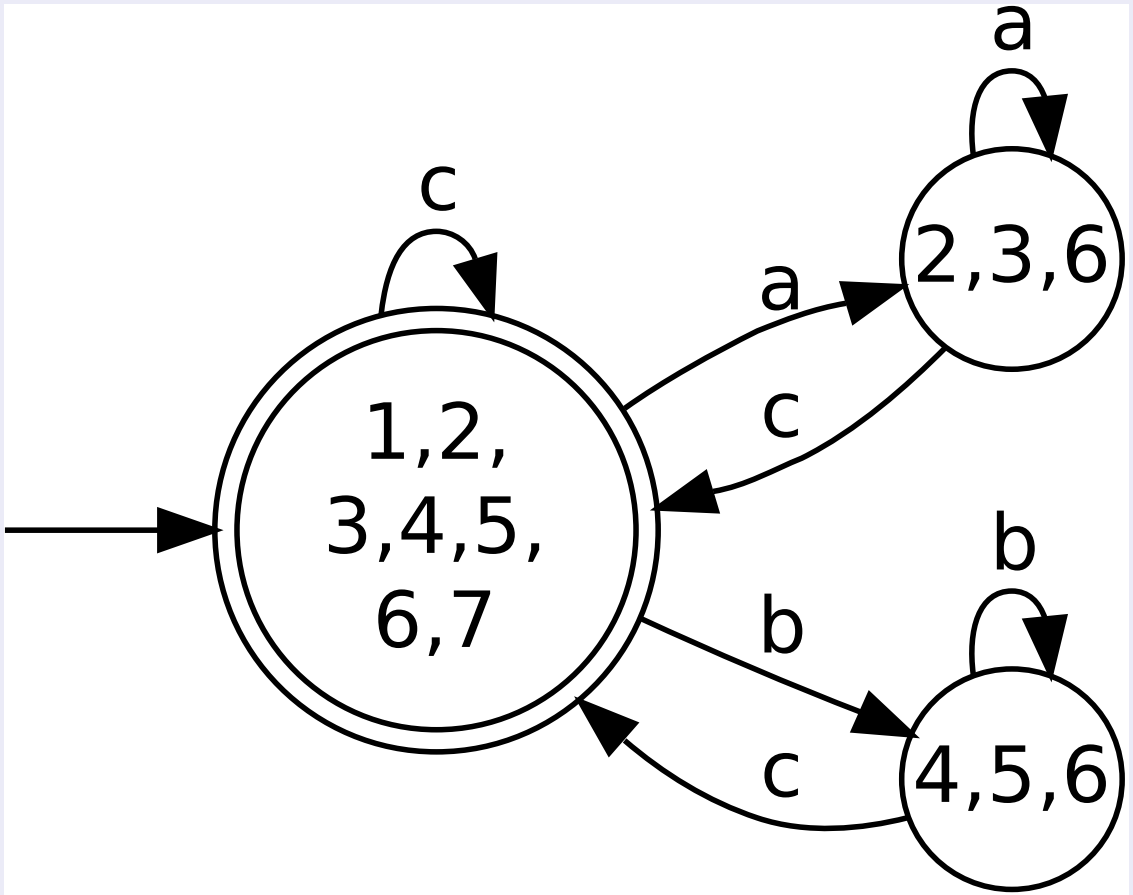
NFA $\rightarrow$ DFA : Exercice 2

Transformer en DFA le NFA



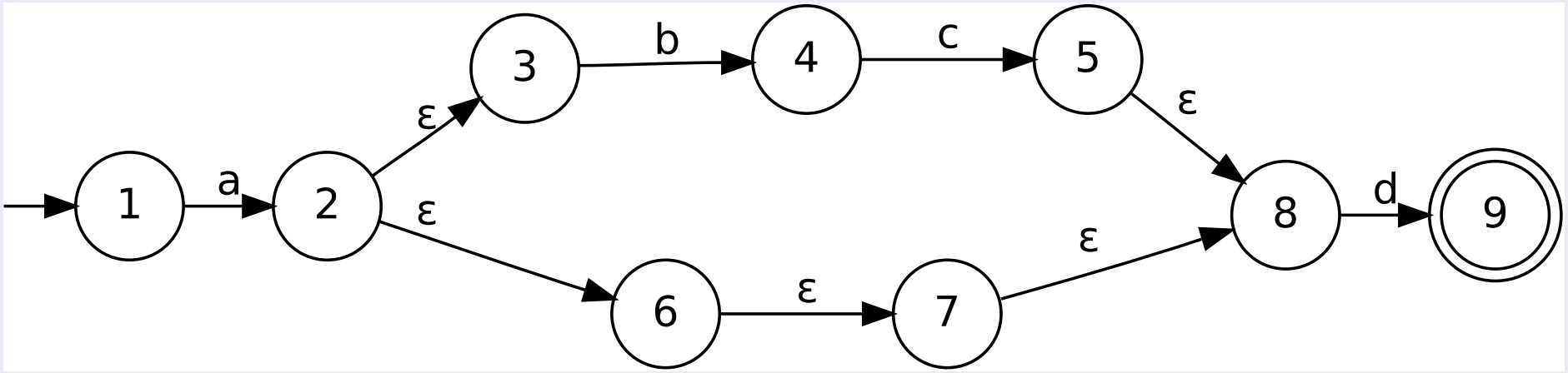
NFA $\rightarrow$ DFA : Exercice 2

Solution



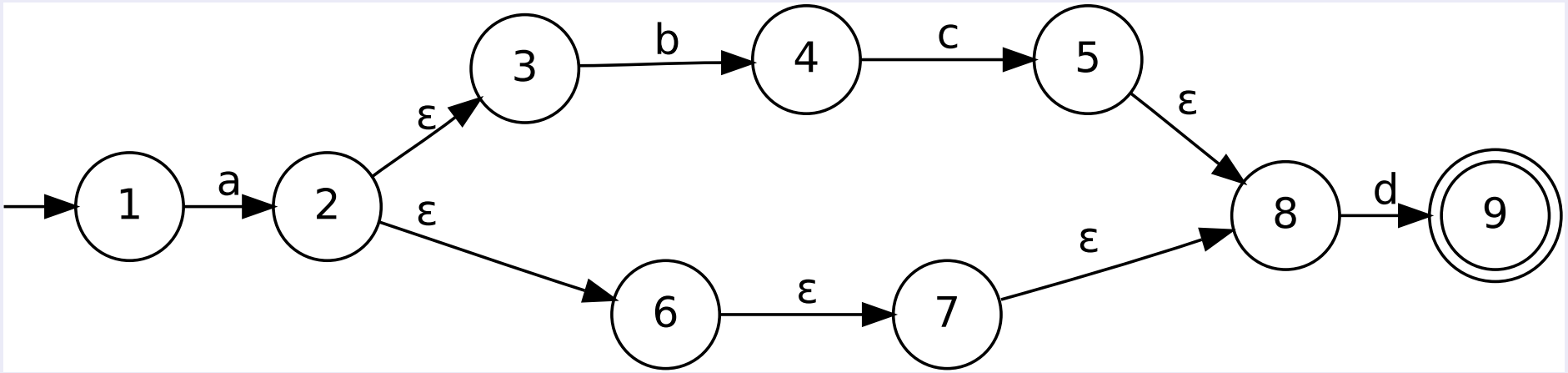
NFA $\rightarrow$ DFA : Exercice 3

Transformer en DFA le NFA

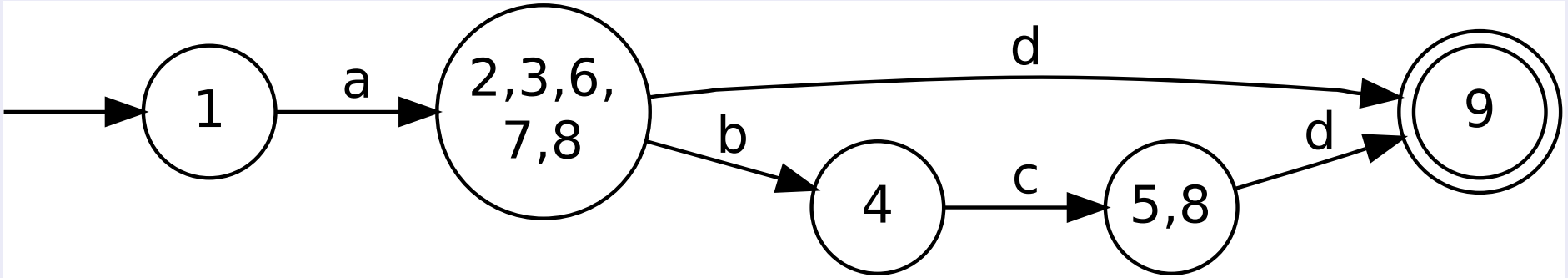


NFA $\rightarrow$ DFA : Exercice 3

Transformer en DFA le NFA



Solution



Autres algorithmes pratiques

Minimisation DFA

- La même chose avec moins d'états
- Théorème : DFA minimum unique
- Corollaire : Permet de déterminer l'équivalence d'expressions régulières

DFAisation paresseuse

- Construire et évaluer le DFA en même temps

Transformation DFA $\rightarrow$ Expression régulière

- Prouve l'égalité de la classe de langages
- Pas forcément utile en pratique