Computing Theory

COMP 147 (4 units)

Chapter 1: Regular Languages
Regular Expressions

Announcements

- Assignment1 due Saturday
- Assignment2
- Quiz2 on Tuesday
  - Closure properties of Regular languages
  - Regular expressions

Regular Expressions

To describe a regular language, L_a, we could

give a DFA, D, such that $L(D) = L_a$,

or an NFA, N, such that that $L(N) = L_a$.

This is not always convenient.

Regular expressions give us a more compact, more readable, notation for the same languages.

Variations on regex appear everywhere that strings need to be specified: compilers, search tools, editing tools …

Regular Expressions

- The value of a regular expression is a language
  - a regex defines a set of strings over some alphabet Σ (for example $\Sigma = \{0,1\}$)
- The operators allowed in a regex are: union $\cup$, concatenation $\circ$ and star $*$
- Any of these operations on regex produces regex.
- Precedence order: $*$ $\circ$ $\cup$
  - parentheses override precedence
- Example: $(0 \cup 1) \circ 0^*$ ($\circ$ often omitted and sometimes | instead of $\cup$)

Formal Definition

Inductive definition

R is a regular expression if R is

1. a, where $a \in \Sigma$
2. $\boldsymbol{\varepsilon}$
3. $\varnothing$
4. $R_1 \cup R_2$
5. $R_1 \circ R_2$
6. $R_1{}^*$

where R_1 and R_2 are regular expressions

$\boldsymbol{\varepsilon}$ is the set containing only the empty string

$\varnothing$ is an empty set

Shorthand Notation

- R_1R_2 represents $R_1 \circ R_2$

- R^+ represents RR^*
  - $R^+ \cup \varepsilon = R^*$

- R^k represents $\underbrace{RRR \ldots R}_{k\ times}$

- Σ represents all strings of length 1 over Σ
  - Σ represents a, where $a \in \Sigma$

Examples

0^*10^* $\quad\{w \mid w \text{ contains a single } 1\}$

$\Sigma^*1\Sigma^*$ $\quad\{w \mid w \text{ has at least a single } 1\}$

$\Sigma^*(str)\Sigma^*$ $\quad\{w \mid w \text{ contains } str \text{ as a substring}\}$

$1^*(01^+)^*$ $\quad\left\{\begin{array}{c} w \mid \text{every } 0 \text{ in } w \text{ is followed} \\ \text{by at least a single } 1 \end{array}\right\}$

$(\Sigma\Sigma)^*$ $\quad\{w \mid w \text{ is of even length}\}$

Examples

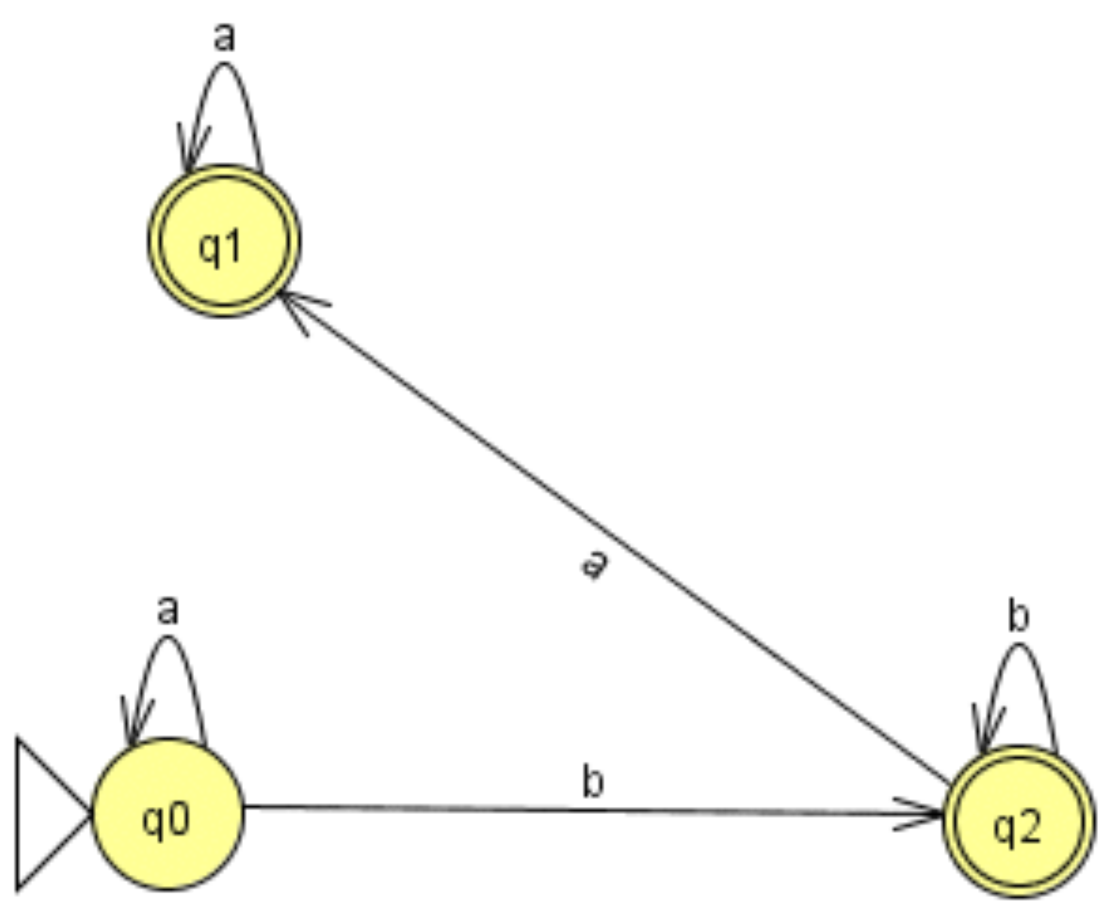
$$0\Sigma^{*}0 \cup 1\Sigma^{*}1 \cup 0 \cup 1$$

all strings starting and ending with the same symbol

Additional Examples

$$a^*b^+a^*$$

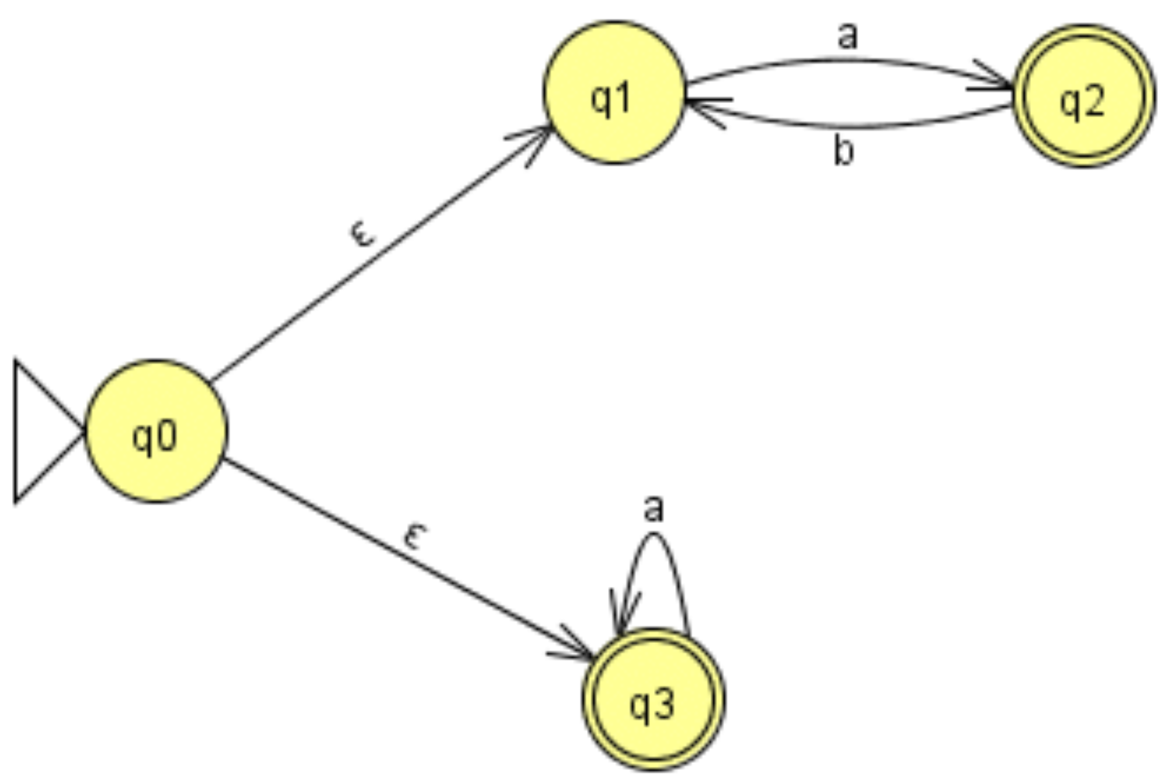
What is the corresponding finite automaton?



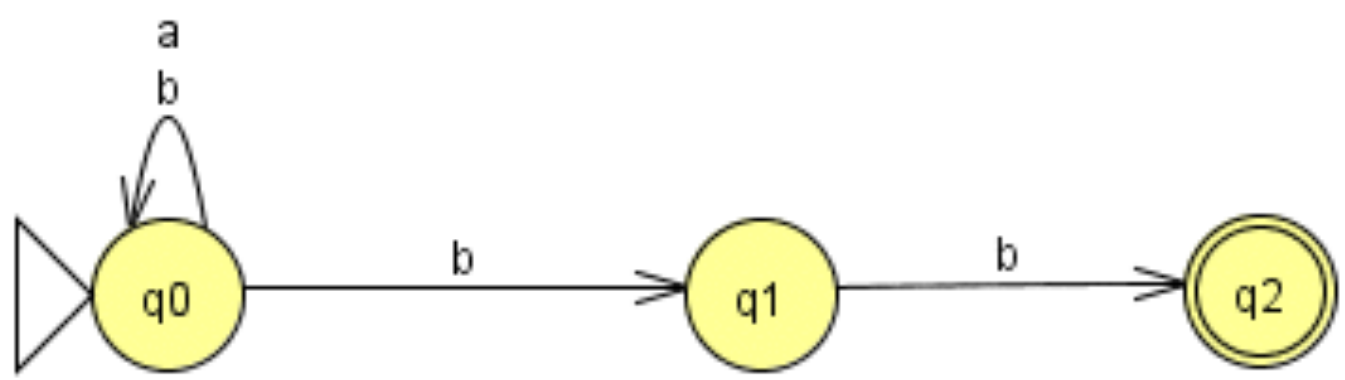
Additional Examples

$$a(ba)^* \cup a^*$$

What is the corresponding finite automaton?

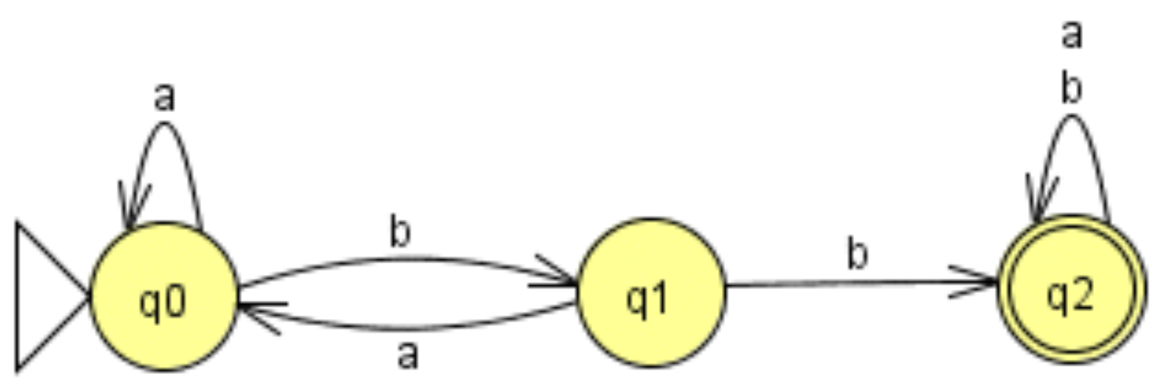


Additional Examples



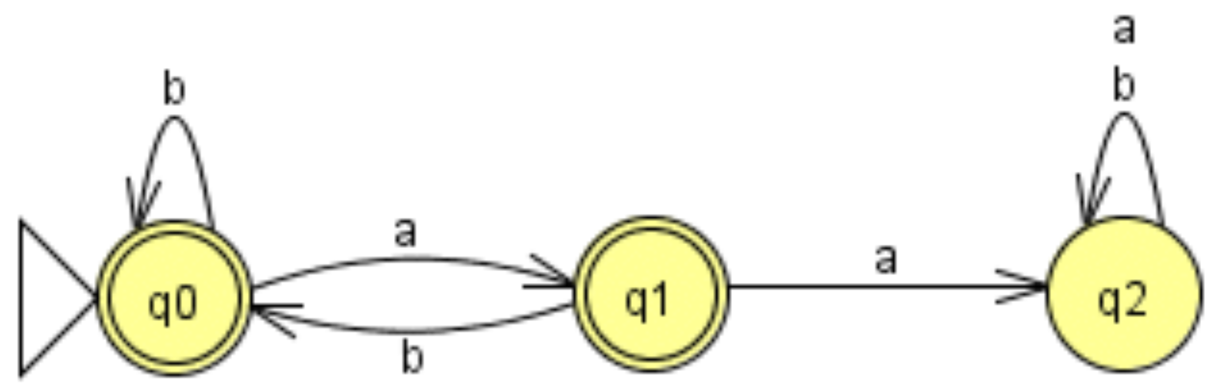
What is the corresponding regular expression?

Additional Examples



What is the corresponding regular expression?

Additional Examples



What is the corresponding regular expression?

Equivalence With Finite Automata

Regular expressions and finite automata are equivalent in their descriptive power.

<u>**Theorem**</u>

A language is regular **if and only if** it can be described by a regular expression.

Equivalence of FAs and Regex

- For alphabet Σ, let
  $L(DFA) = \{\beta \mid \beta \text{ is a DFA over } \Sigma\}$
  $L(NFA) = \{\beta \mid \beta \text{ is a NFA over } \Sigma\}$
  $L(RE) = \{\beta \mid \beta \text{ is a regular expression over } \Sigma\}$

  Then

  $L(DFA) \leftrightarrow L(NFA) \leftrightarrow L(RE)$

Regex / FA Equivalence

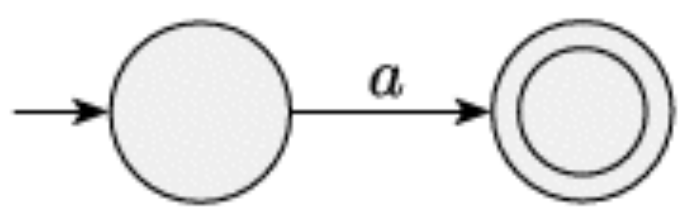
- Theorem: A language is regular if and only if some regular expression describes it.
  - Lemma: If a language is described by a regular expression, then it is regular
    - Proof by construction: show how to build an NFA from a regex
  - Lemma: If a language is regular, then it is described by a regular expression
    - Proof by construction: show how to convert a DFA to a regex

Construct NFA from Regex

- Convert regex R into NFA N.
    - There are 6 cases (see formal definition of a regex)

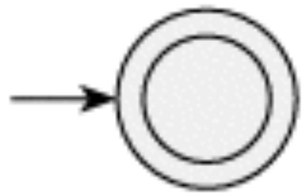
- Case 1:
  $R = a \in \Sigma$
  $L(R) = \{ a \}$

  $N =$

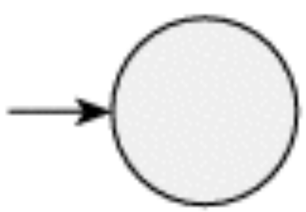


Construct NFA from Regex

- Case 2:
  $R = \varepsilon, L(R) = \{ \varepsilon \}$

  $N =$ 

- Case 3:
  $R = \emptyset, L(R) = \{ \ \} = \emptyset$

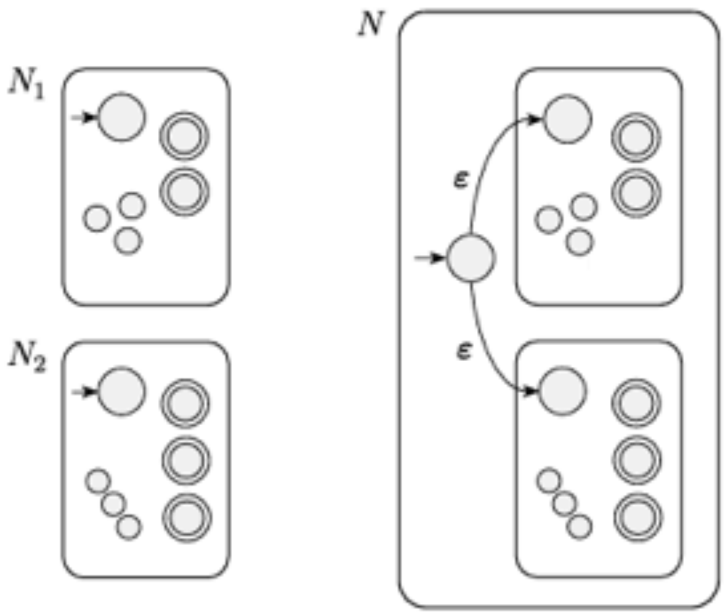
  $N =$ 

Construct NFA from Regex

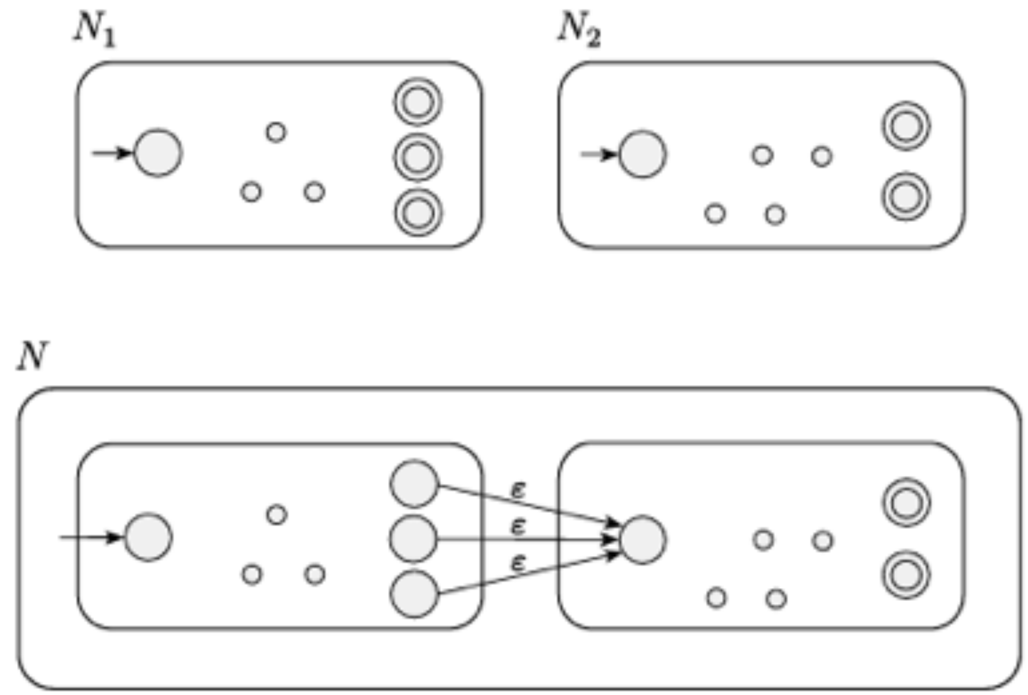
- Case 4:
  $R = R_1 \cup R_2, L(R) = L(R_1) \cup L(R_2)$
- Case 5:
  $R = R_1 \circ R_2, L(R) = L(R_1) \circ L(R_2)$
- Case 6:
  $R = R_1{}^*, L(R) = L(R_1)^*$
  - In these cases, use the NFA construction techniques from the closure proofs for $\cup$, $\circ$ and $*$.

Construct NFA from Regex

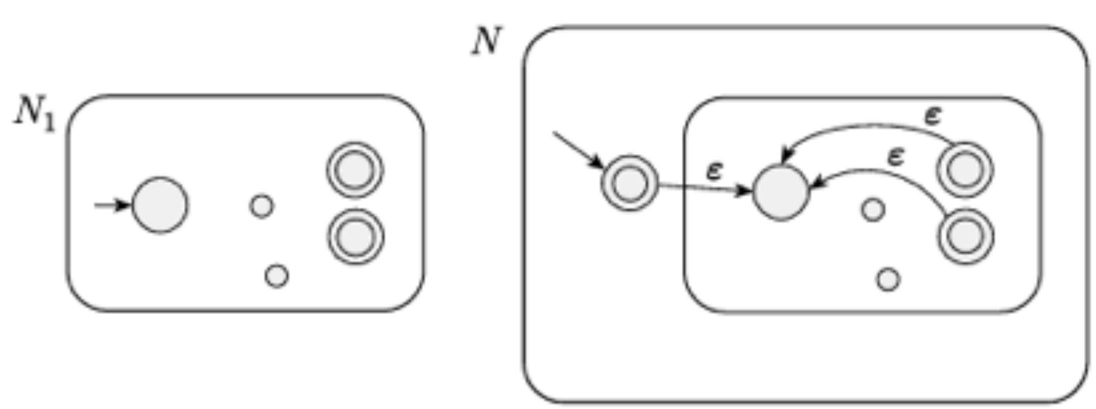
Case 4: $R = R_1 \cup R_2$



Case 5: $R = R_1 \circ R_2$

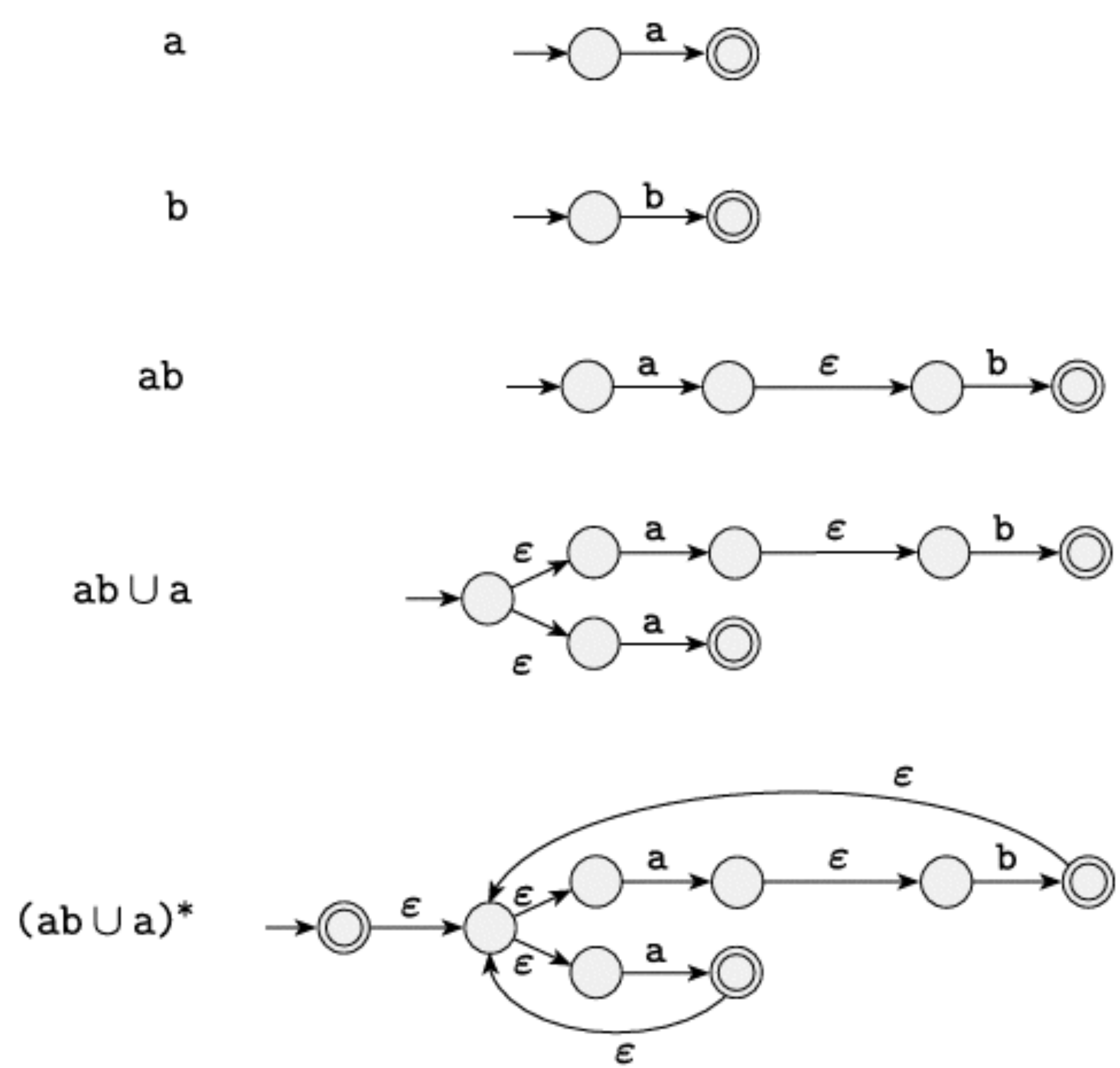


Case 6: $R = R_1{}^*$



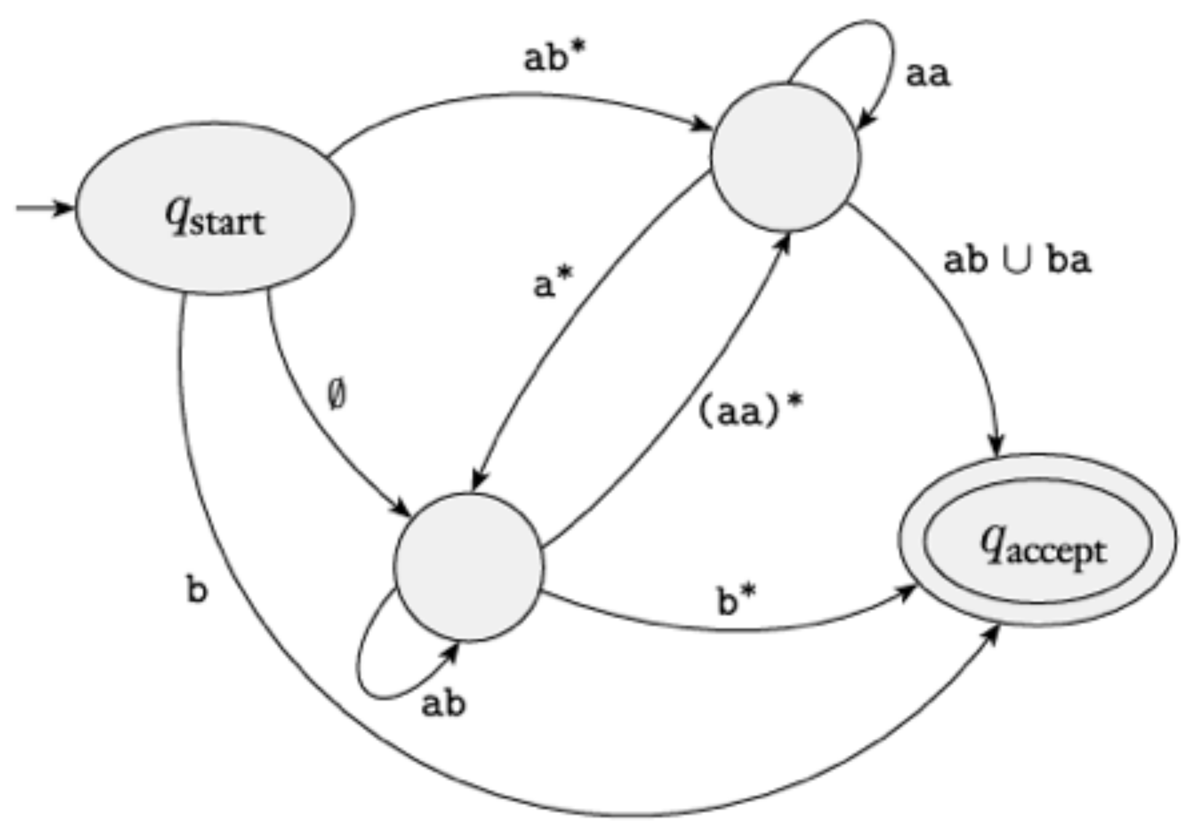
Regex to NFA Example

- R = (ab ∪ a)*



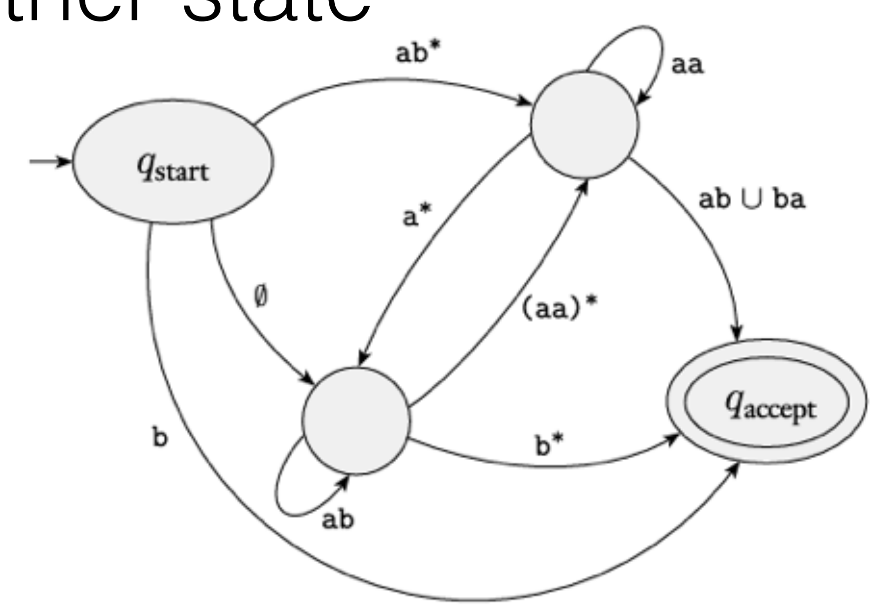
Construct Regex from DFA

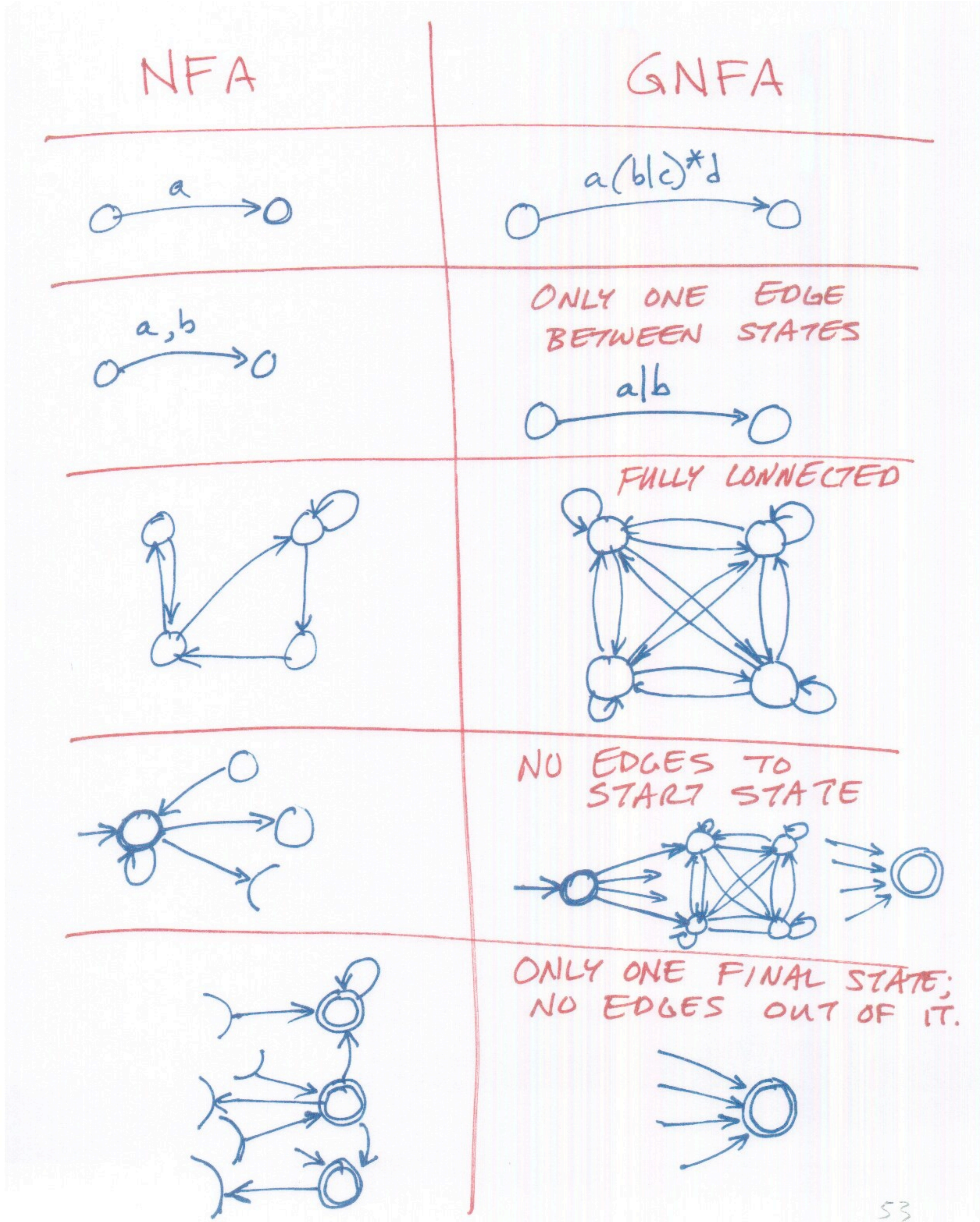
- Convert DFA D into regex R.
  - We'll use a new type of FA as an intermediate step
  - GNFA: generalized nondeterministic finite automata
  - GNFA allow regex as transition labels



GNFA Special Form

- A GNFA is similar to an NFA, except:
  - GNFA allow regex as transition labels
- For the current proof, add following conditions:
  - One start state with transitions to all other states
  - One accept state with transitions from all other states
  - Start state and accept state are different
  - Transitions from every state to every other state (excluding start and accept)
  - Every state has a self transition (excluding start and accept)



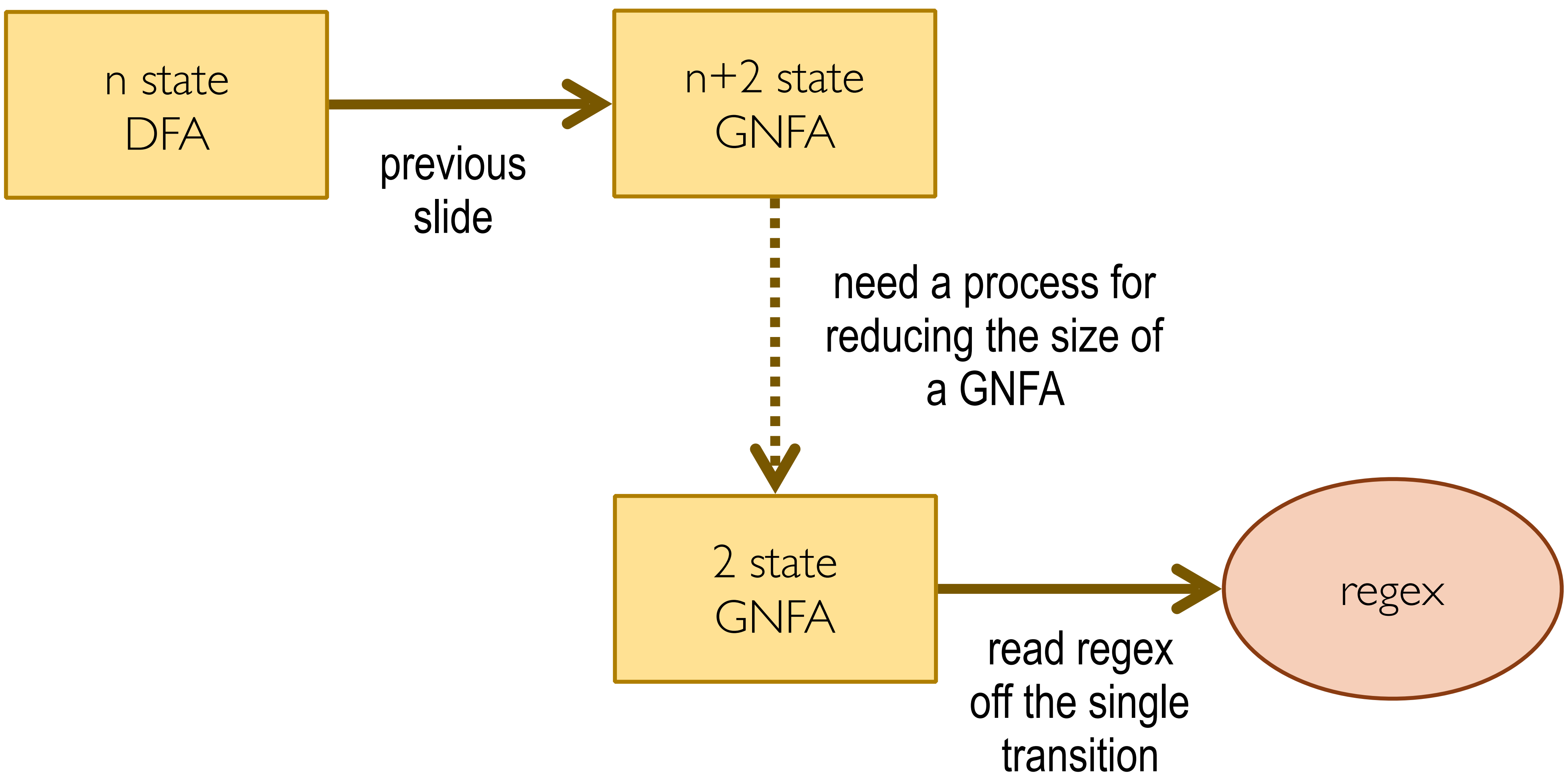

NFA
GNFA
a
a(b|c)*d
a,b
ONLY ONE EDGE BETWEEN STATES
a|b
FULLY CONNECTED
NO EDGES TO START STATE
ONLY ONE FINAL STATE; NO EDGES OUT OF IT.
53

DFA to Regex (step 1: DFA to GNFA)

- Add new start state with **ε** transition to old start state.
- Add new accept state with **ε** transitions from all old accept states
- For any pair of states that has multiple transitions, replace with transition labeled with union of previous labels

(Example on white board)

DFA to Regex

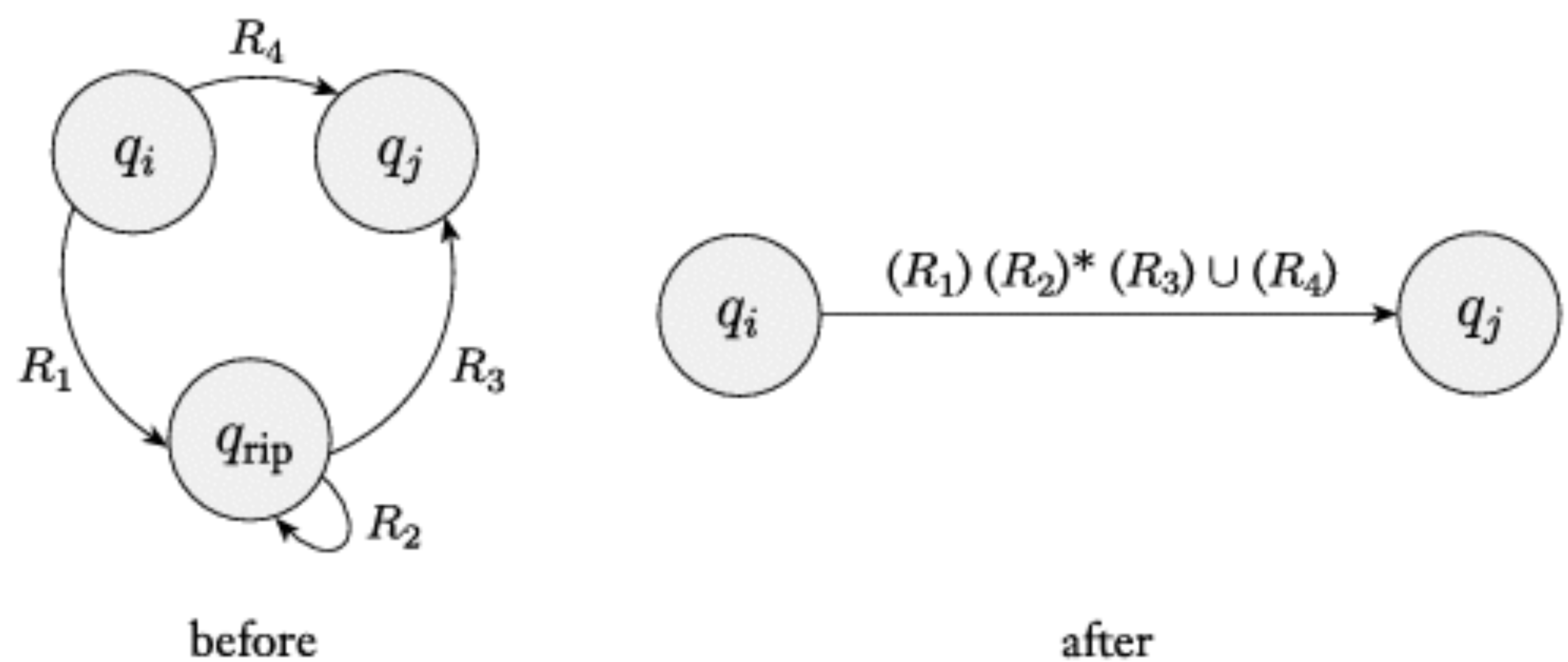

n state
DFA
previous
slide
n+2 state
GNFA
need a process for
reducing the size of
a GNFA
2 state
GNFA
read regex
off the single
transition
regex

Ripping a state from a GNFA

- Select a state at random (do not select start or accept states)
- Let's call it q_{rip}
- Rip the state out of the GNFA
- Remove q_{rip} and all edges to/from it
- Modify the other transition edges so that the machine accepts the same language

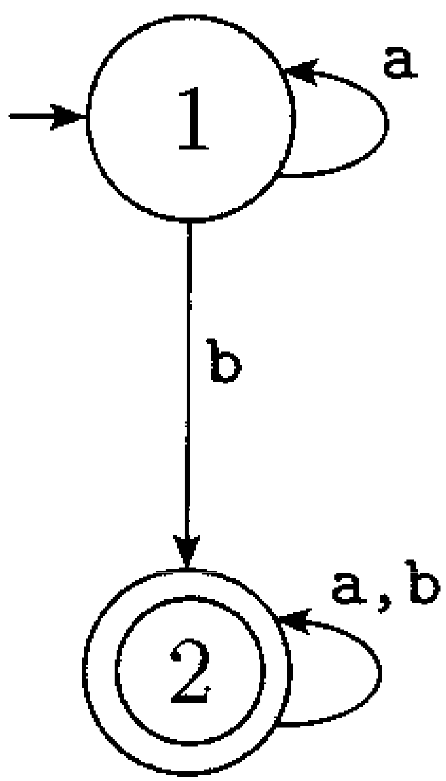
Ripping a state from a GNFA

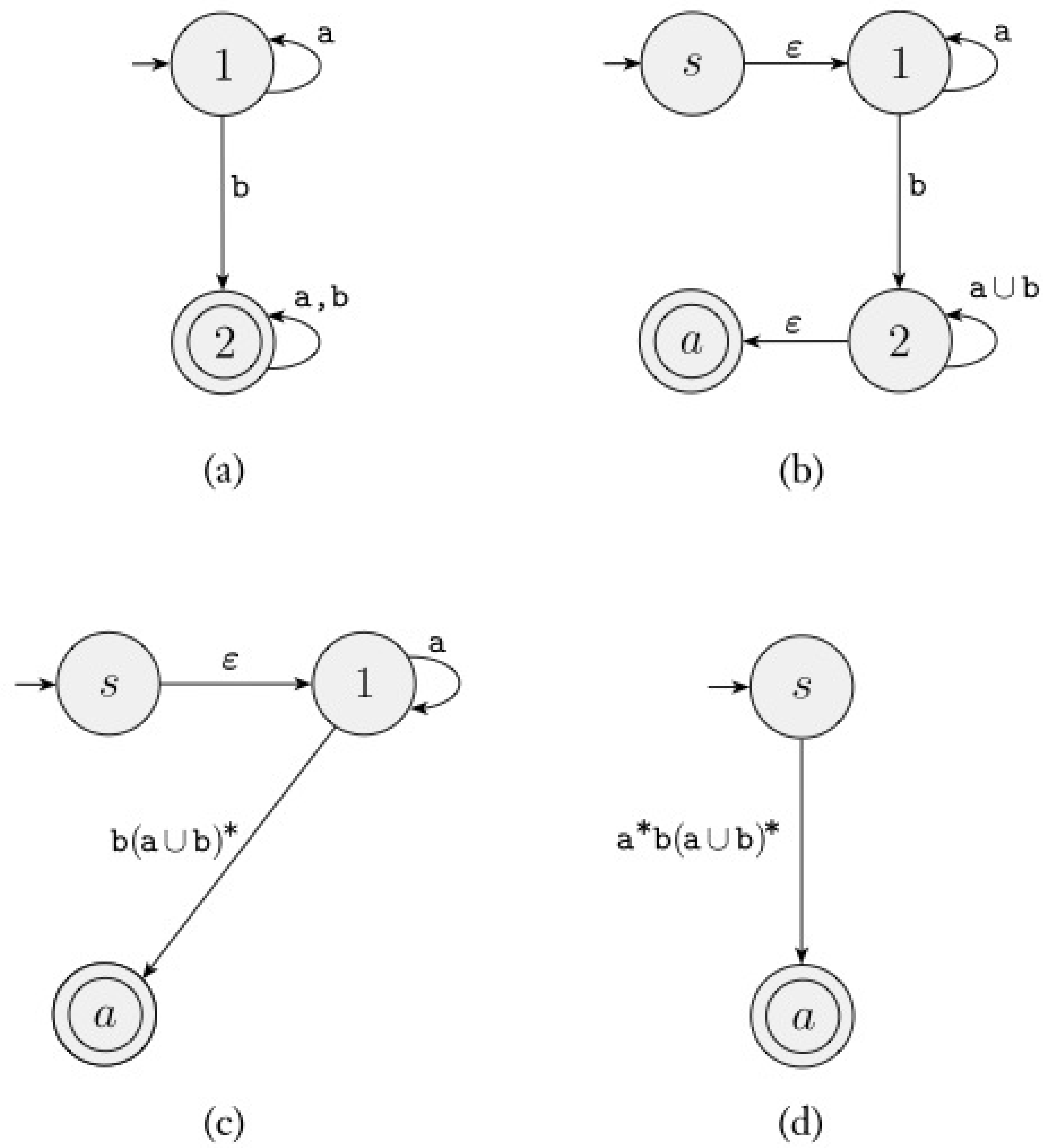
- To reduce the size of the GNFA, we'll rip out states, one at a time, and repair the transitions
  - Example:



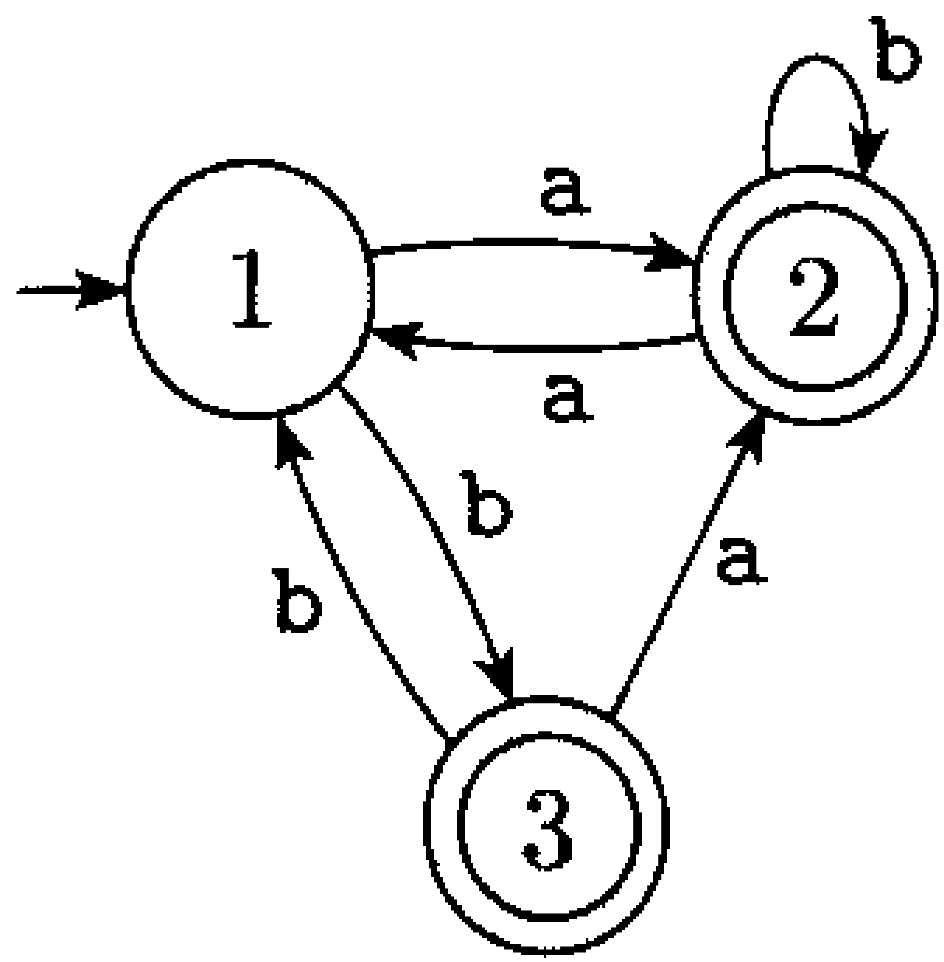
  - If we formalize this into an algorithm, we'll complete our proof

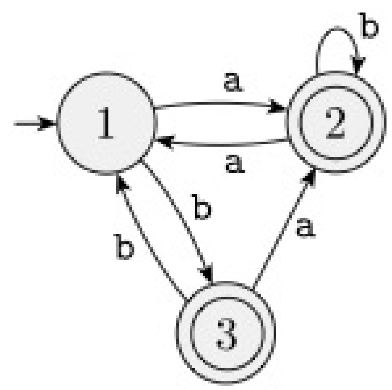
Examples on Board



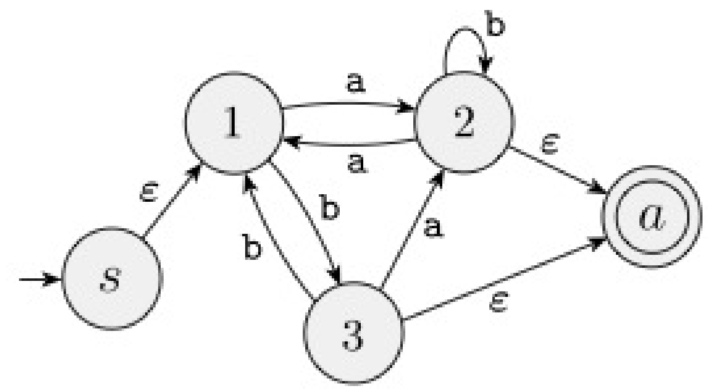

1
a
b
2
a,b
(a)
s
ε
1
a
b
2
a∪b
ε
a
(b)
s
ε
1
a
b(a∪b)*
a
(c)
s
a*b(a∪b)*
a
(d)

Examples on Board

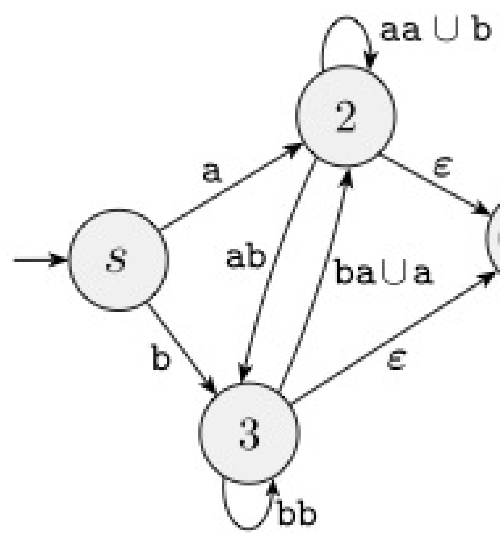



1
2
3
a
a
b
b
a
b

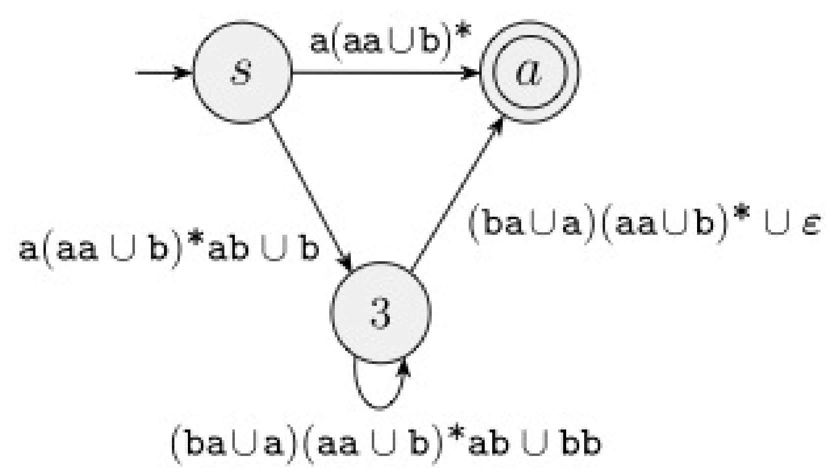
(a)


s
1
2
3
a
ε
a
a
ε
b
b
a
b
ε

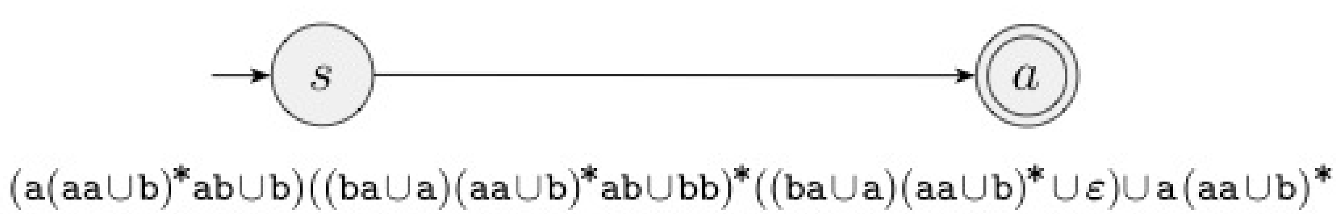
(b)


aa ∪ b
2
s
a
a
ab
ba∪a
ε
b
3
ε
bb

(c)


s
a(aa∪b)*
a
a(aa ∪ b)*ab ∪ b
(ba∪a)(aa∪b)* ∪ ε
3
(ba∪a)(aa ∪ b)*ab ∪ bb

(d)


s
a
(a(aa∪b)*ab∪b)((ba∪a)(aa∪b)*ab∪bb)*((ba∪a)(aa∪b)*∪ε)∪a(aa∪b)*

(e)