CS 6041
Theory of Computation

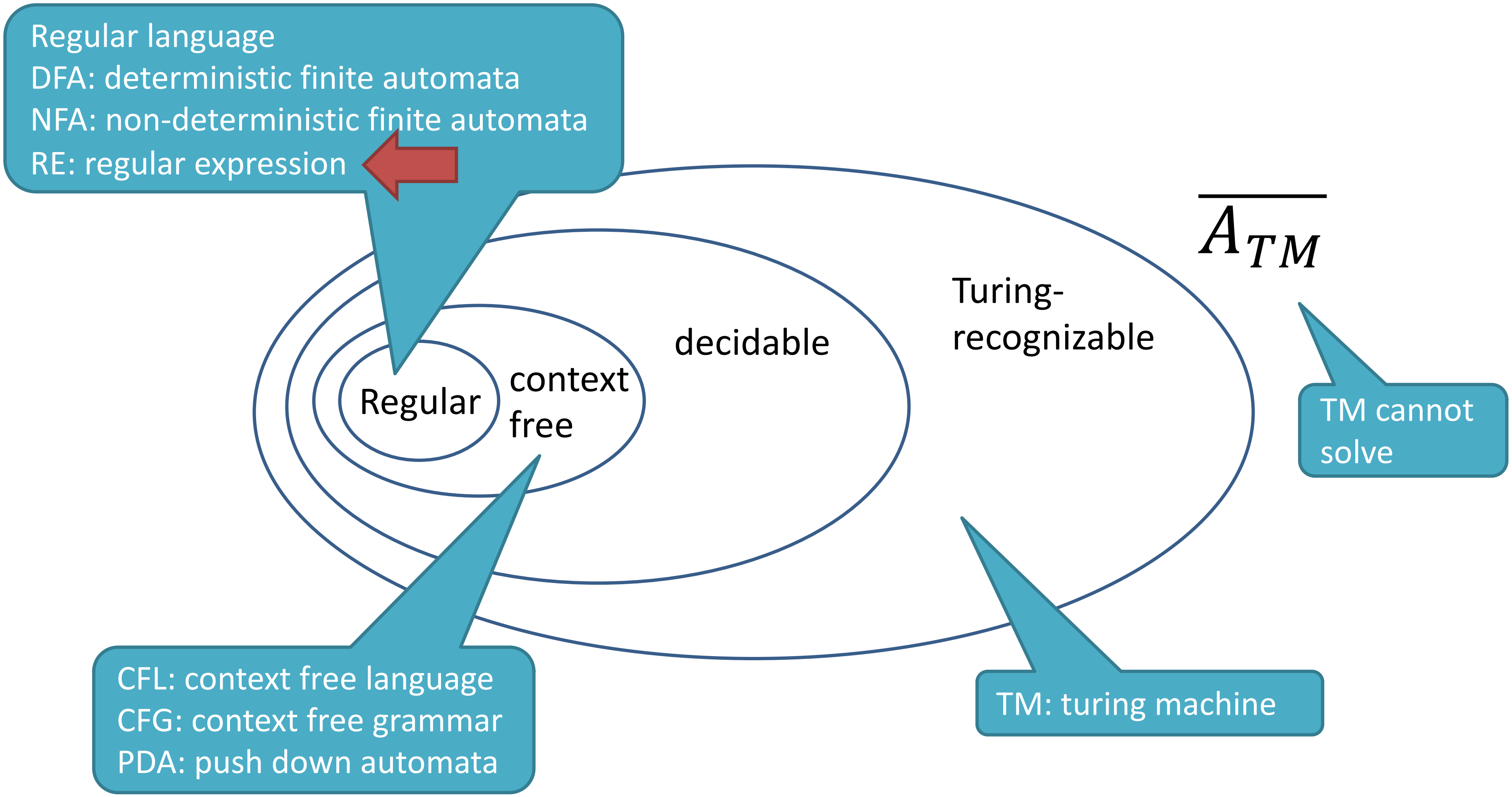
Regular expression

Kun Suo

Computer Science, Kennesaw State University

https://kevinsuo.github.io/

Where are we now?



Outline

- Regular expression
  - Definition
  - Example
- Equivalence with DFA/NFA
  - Regular expression $\Rightarrow$ Regular language
  - Regular expression $\Leftarrow$ Regular language

Regular expression

- Regular expressions are those describing languages by using regular operations (*Union, Concatenation, Star, Complement, Boolean, etc*.)

- Example:

$(0 \cup 1)0^*$

$= (\{0\} \cup \{1\})\{0\}^*$ //add bracket

$= \{0,1\}\{0\}^*$ //comma = union

Regular expression

- $\Sigma=\{0,1\}$
  - $(0\cup1)^* = \{0,1\}^* = \Sigma^*$

- Σ is any alphabet
  - Σ describes the language consisting of all strings of length 1 over this alphabet
  - Σ^* describes the language consisting of all strings over that alphabet

Regular expression

- What is Σ^*1? -> {w | w…}
  - describes the language that contains all strings that end in a 1

- What is $(0\Sigma^*) \cup (\Sigma^*1)$? -> {w | w…}
  - describes all strings that start with a 0 or end with a 1

Definition of regular expression

- R is regular expression if R is
  - a, where $a \in \Sigma$, length is 1;
  - ε , length is 0;
  - $\emptyset$;
  - Union: $(R_1 \cup R_2)$, where R_1 and R_2 are all regular expressions;
  - Concatenation: (R_1R_2), where R_1 and R_2 are all regular expressions;
  - Star: $(R_1{}^*)$, where R_1 is regular expression.

Priority:
$* > \circ > \cup$

- L(R): the language of R
  - $L(1\Sigma^*)$: language that starts with 1

Regular expression → Description

- Let $\Sigma=\{0,1\}$
  - 0^*10^* = { w | w contains a single 1 }
  - $\Sigma^*1\Sigma^*$ = { w | w has at least one 1 }
  - $\Sigma^*001\Sigma^*$ = { w | w contains the substring 001 }
  - $(\Sigma\Sigma)^*$ = { w | w is a string of even length}
  - $(\Sigma\Sigma\Sigma)^*$ = { w | the length of w is a multiple of 3 }

Regular expression → Description

- Let Σ={0,1}
  - $01 \cup 10 = \{ 01, 10 \}$
  - $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$
  - $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$
  - $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$

  = { w | w starts and ends with the same symbol }

What is the description for this RE?

Some special regular expression

- Let $\Sigma=\{0,1\}$
  - $1^*\emptyset = \emptyset$
  - $\emptyset^* = \{\varepsilon\}$
  - $R\cup\emptyset = R$
  - $R\emptyset = \emptyset$
  - $R\cup\varepsilon = R\cup\{\varepsilon\}$
  - $R\varepsilon = R$

Regular expression for numbers

- $\{+,-,\varepsilon\}(D^* \cup D^*.D^*)$, where $D=\{0,1,2,3,4,5,6,7,8,9\}$
  - 72
  - 3.14159
  - +7.
  - -.01

Description→ Regular expression

- Let $\Sigma=\{0,1\}$

{ w | w contains exactly two 0s}	$1^*01^*01^*$
{ w | w contains at least two 0s }	$\Sigma^*\ 0\Sigma^*\ 0\Sigma^*$
{ w | w begins with a 1 and ends with a 0}	$1\Sigma^*\ 0$
{ w | w is a string which does not contain substring 10}	0^*1^*

Description→ Regular expression

- Let Σ={0,1}

{ w | w contains exactly two 0s}	1*01*01*
{ w | w contains an even number of 0s }	(1*01*01*)*
{ w | w contains exactly two 1s}	0*10*10*
{ w | w contains an even number of 0s, or contains exactly two 1s}	(1*01*01*)* U 0*10*10*

Outline

- Regular expression
  - Definition
  - Example
- Equivalence with DFA/NFA
  - Regular expression $\Rightarrow$ Regular language
  - Regular expression $\Leftarrow$ Regular language

Equivalence with DFA/NFA

- **Theorem: A language is regular if and only if some regular expression describes it.**

- Lemma1:

  Regular expression $\Rightarrow$ Regular language.

- Lemma2:

  Regular expression $\Leftarrow$ Regular language.

Regular expression ⇒ Regular language

- Proof

  Create an equivalent NFA for regular expression

Definition:

R is regular expression if R is

- a
- ε
- $\emptyset$
- $R_1 \cup R_2$
- $R_1 R_2$
- $R_1{}^*$

⇒ Create NFA for each case

Regular expression ⇒ Regular language

- Proof

Create an equivalent NFA for regular expression

Case 1: a

$R=a,\ a\in\Sigma.$

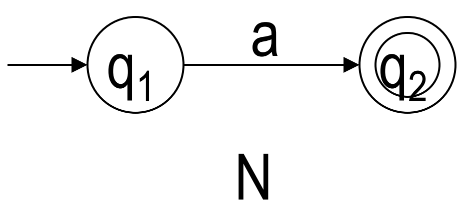
$L(\ R\ )=\{a\},$

$N=(\{q_1,q_2\},\Sigma,\delta,q_1,\{q_2\}),$

$\delta(q_1,a)=\{q_2\},$

$\delta(r,b)=\varnothing$, if $r\neq q_1$ or $b\neq a$.

Can you draw the NFA?



Regular expression ⇒ Regular language

- Proof

Create an equivalent NFA for regular expression

Case 2: ε

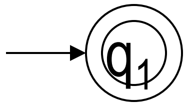
$R=\varepsilon$.

$L(\ R\ )=\{\varepsilon\}$,

$N=(\{q1\},\Sigma,\delta,q1,\{q1\})$,

$\forall r,\forall b,\ \delta(r,b)=\varnothing$.

Can you draw the NFA?



N

Regular expression ⇒ Regular language

- Proof

Create an equivalent NFA for regular expression

Case 3: empty set

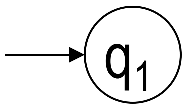
$R=\emptyset$.

$L(R)=\emptyset$,

$N=(\{q_1\},\Sigma,\delta,q_1,\emptyset)$,

$\forall r, \forall b,\ \delta(r,b)=\emptyset$.

Can you draw the NFA?



N

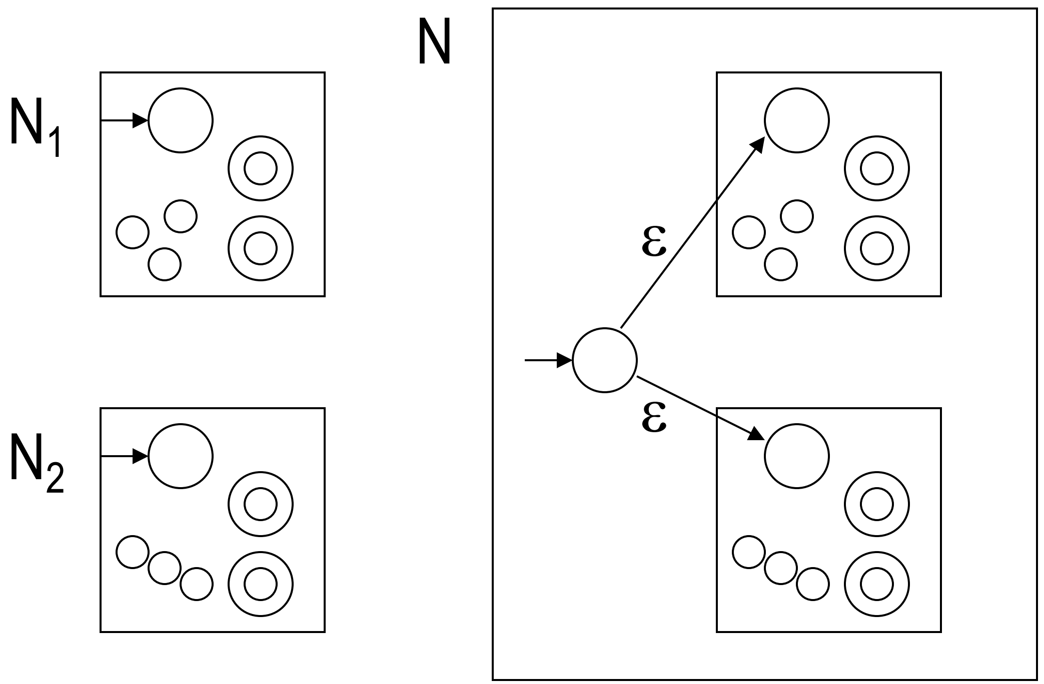
Regular expression ⇒ Regular language

- Proof

Create an equivalent NFA for regular expression

Case 4: $R=(R_1 \cup R_2)$,

Can you draw the NFA?



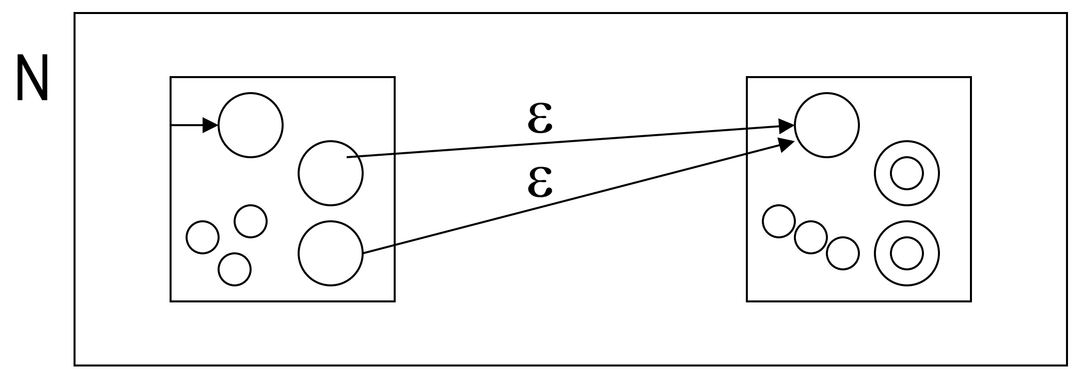
Regular expression ⇒ Regular language

- Proof

Create an equivalent NFA for regular expression

Case 5: $R=(R_1R_2)$,

Can you draw the NFA?



Add all accept states in N_1 to start state of N_2

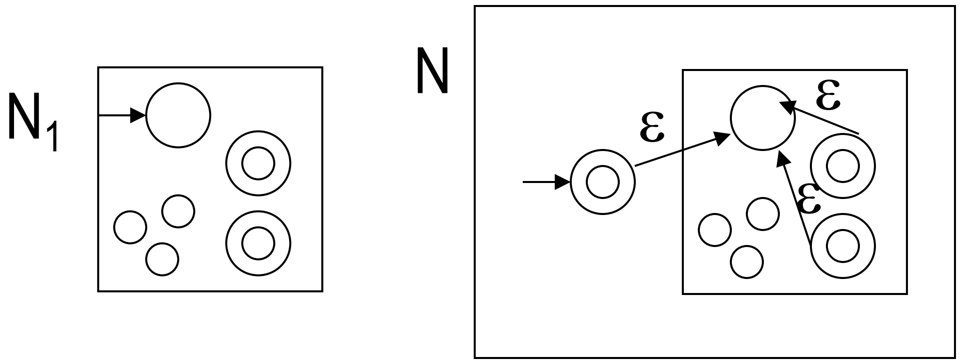
Regular expression ⇒ Regular language

- Proof

Create an equivalent NFA for regular expression

Case 6: $R=(R_1{}^*)$,

Can you draw the NFA?



Add all accept states to start state

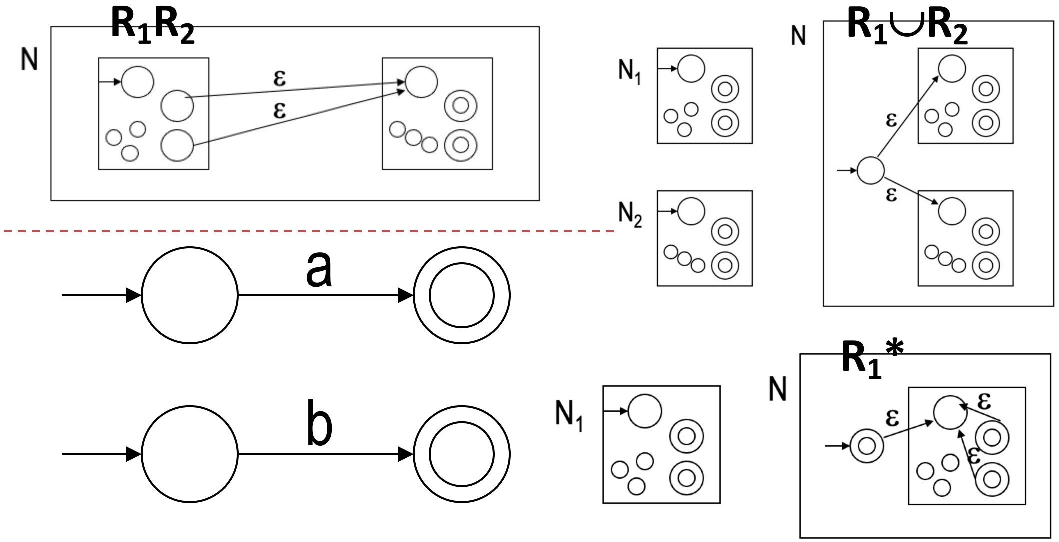
Equivalence with DFA/NFA

- **Theorem: A language is regular if and only if some regular expression describes it.**

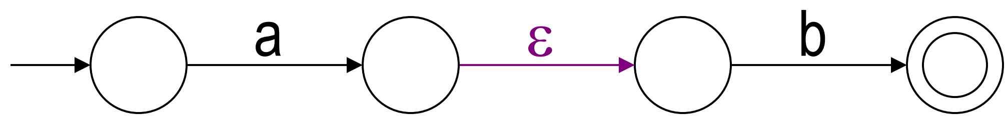
- Lemma1: (**proved**)

  Regular expression $\Rightarrow$ Regular language (NFA).

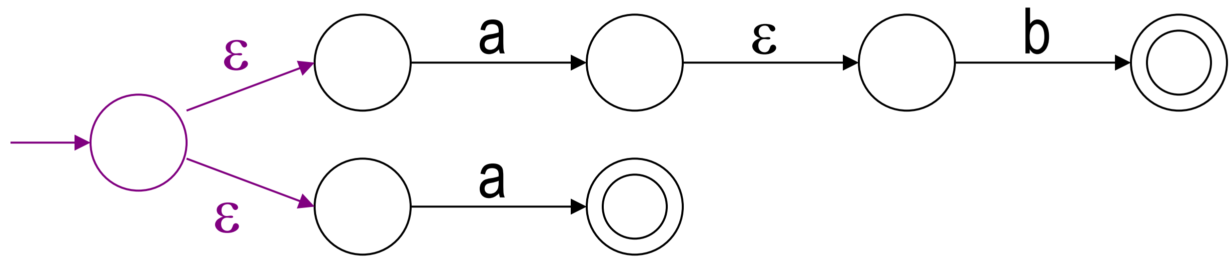
RE ⇒ RL(NFA)



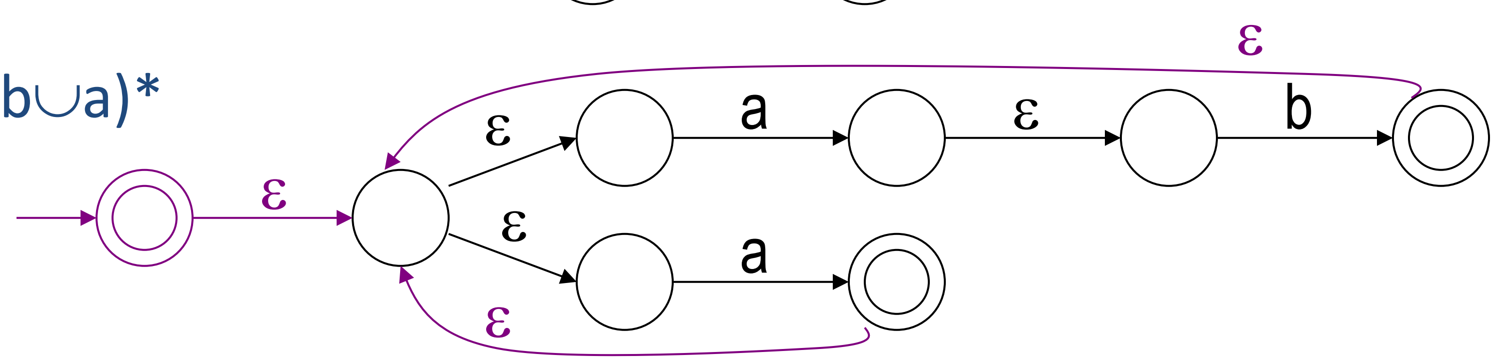
- Create (ab∪a)*
  1. a
  2. b



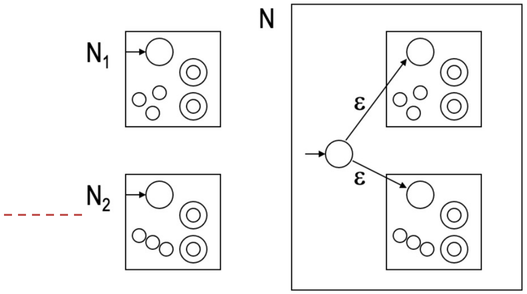
  3. ab

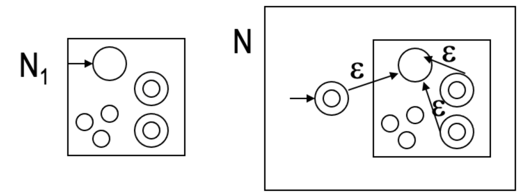


  4. ab∪a
  5. (ab∪a)*

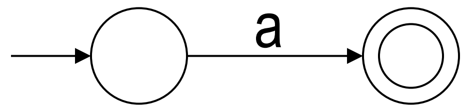


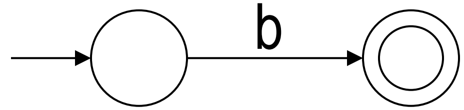
RE ⇒ Regular language (NFA)



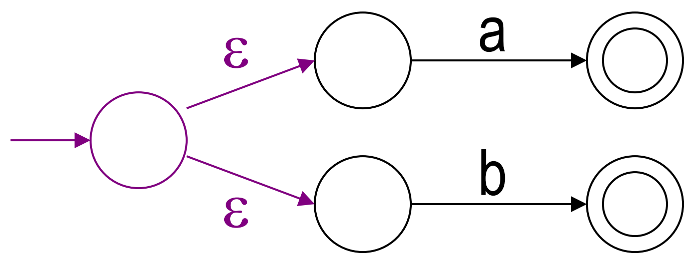


- Create (a∪b)*aba
  - a

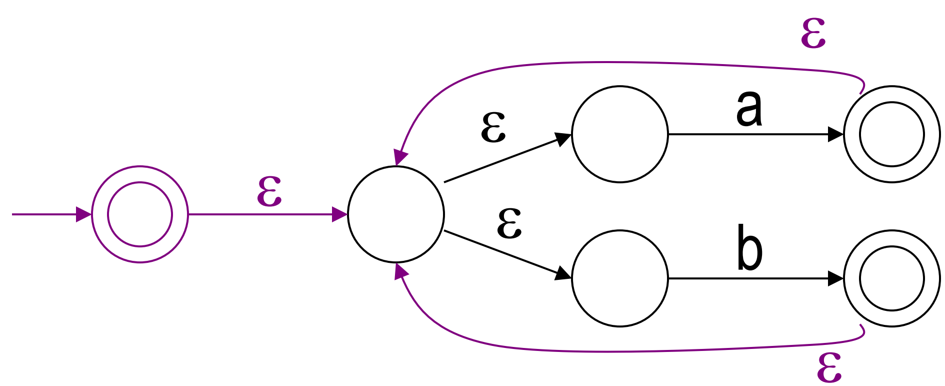




  - b
  - a∪b

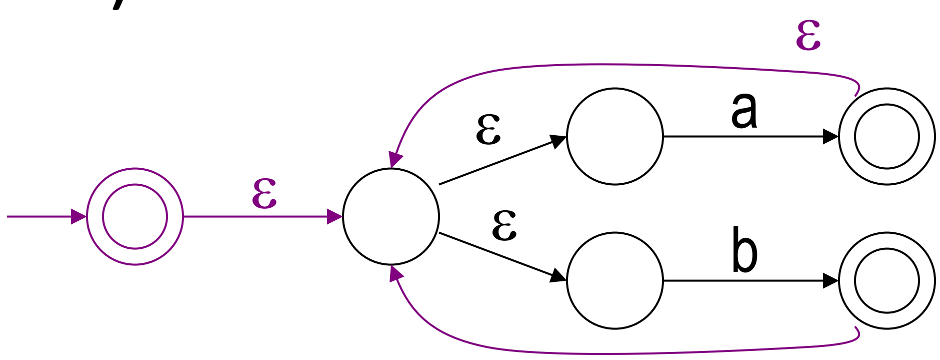


  - (a∪b)*

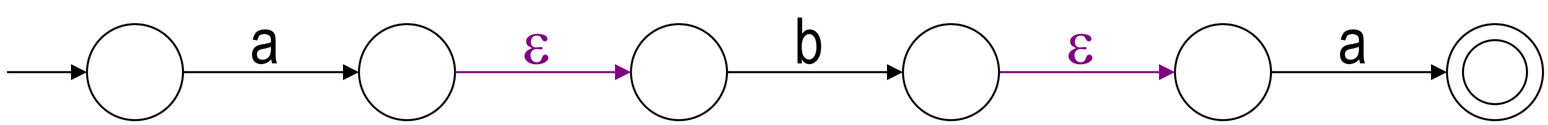


RE ⇒ Regular language (NFA)

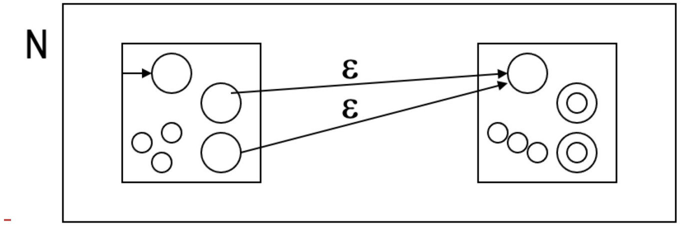
- Create (a∪b)*aba
  - (a∪b)*



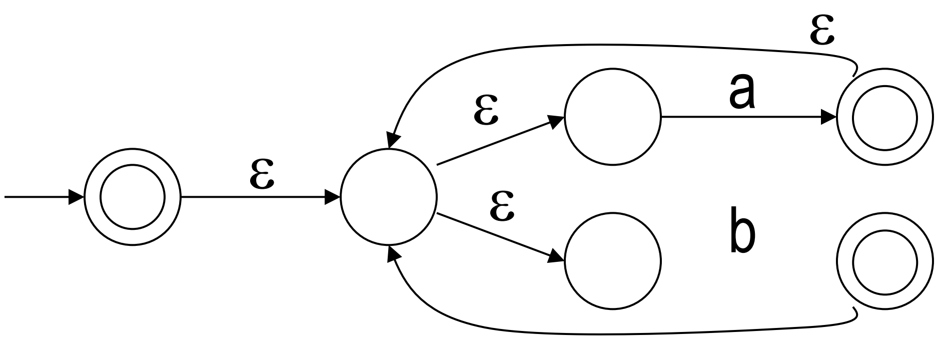
  - aba

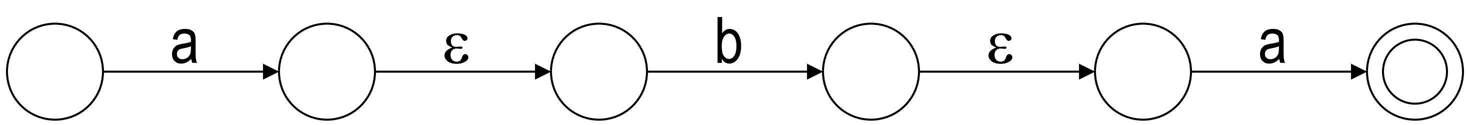


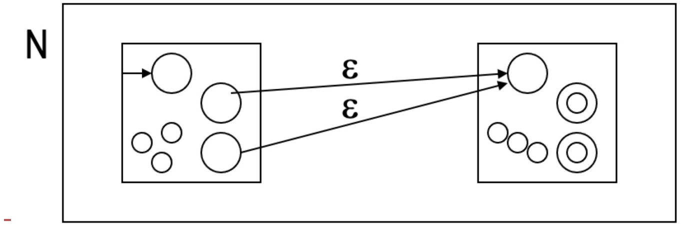
RE ⇒ Regular language (NFA)



- Create (a∪b)*aba

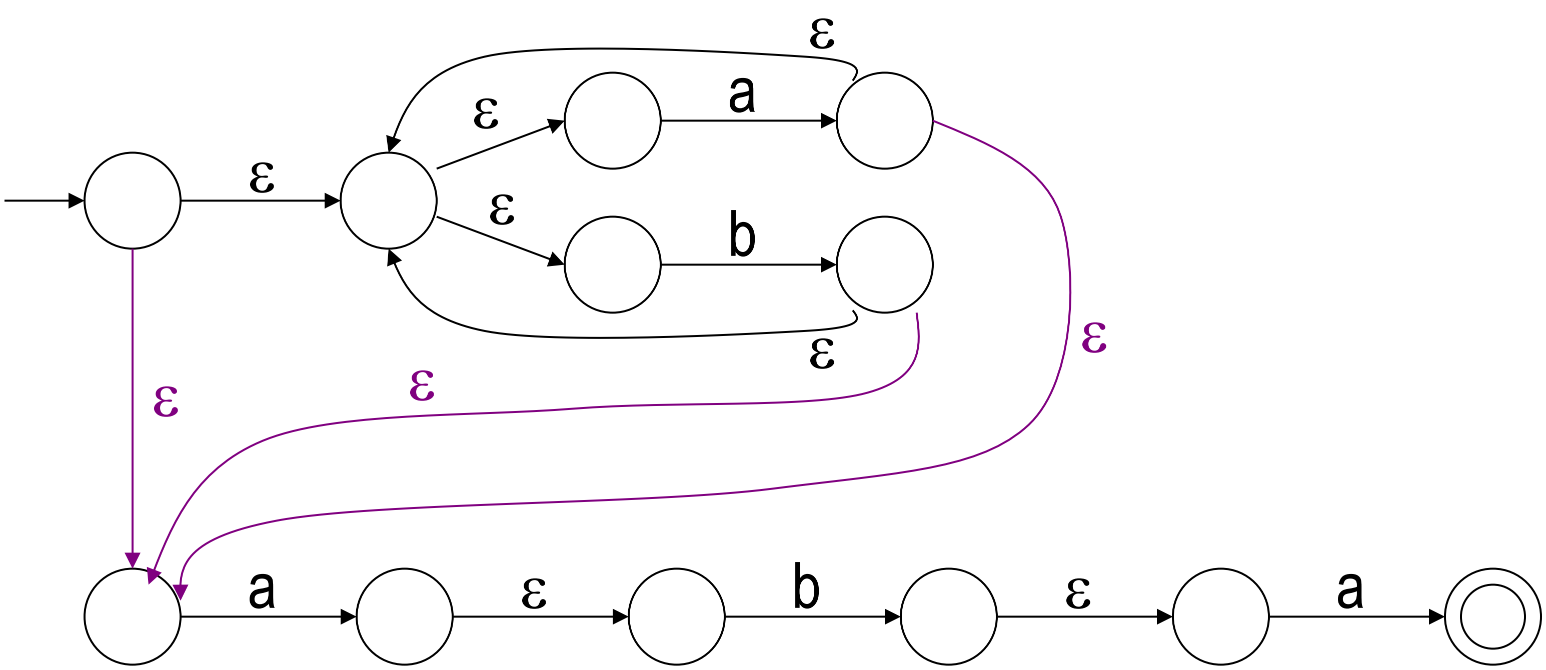




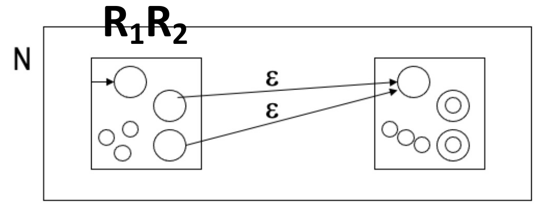


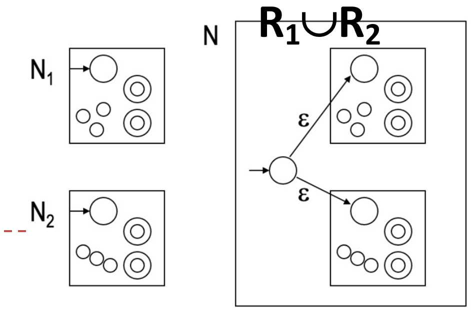
RE ⇒ Regular language (NFA)

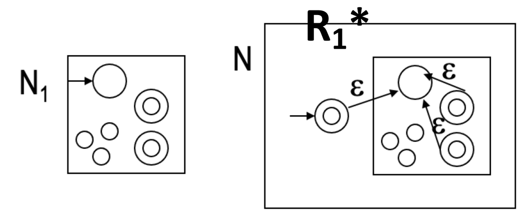
- Create (a∪b)*aba



Practice:
RE ⇒ RL(NFA)

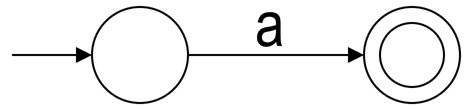




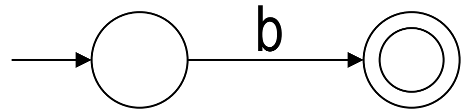


- Create $(ab) \cup a^*$

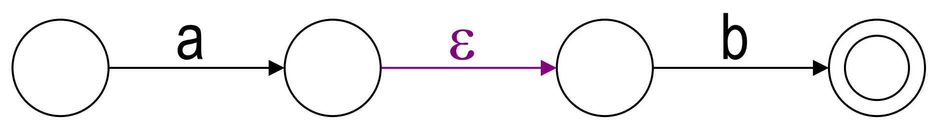
1. a



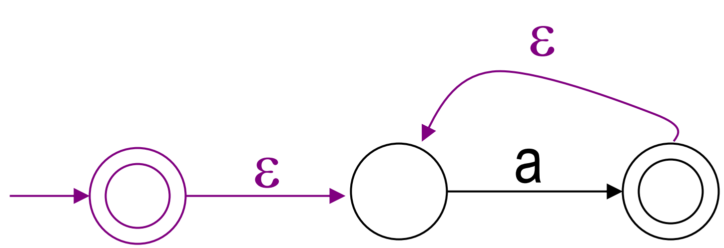
2. b



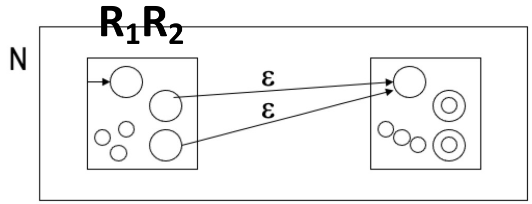
3. ab

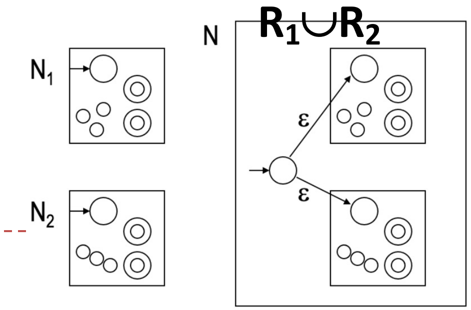


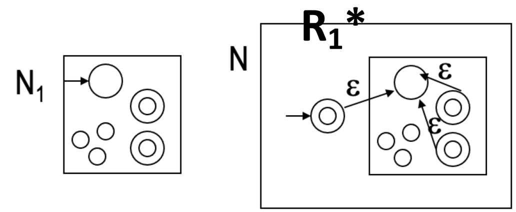
4. a*



Practice:
RE ⇒ RL(NFA)

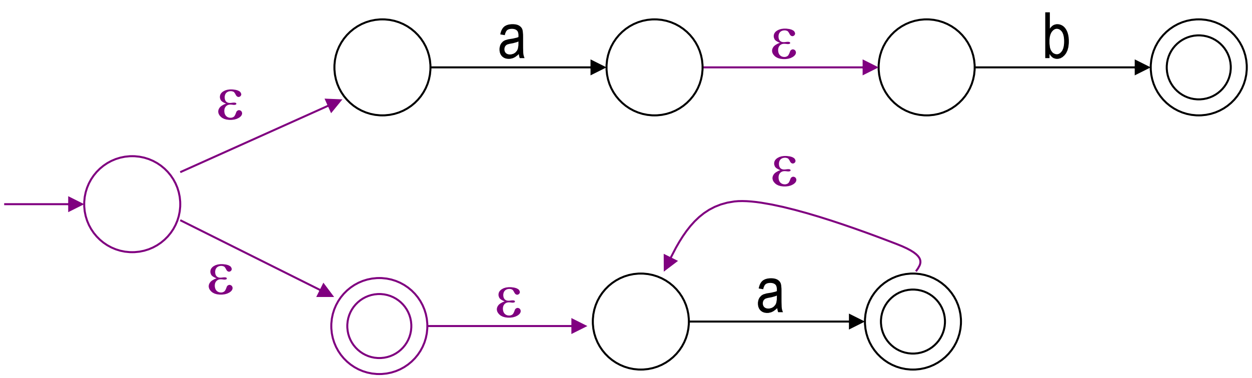






- Create (ab)∪a*

ab ∪ a*



Regular expression ⇐ Regular language

- Proof

Definition a language is called a regular language if some finite automaton (DFA/NFA) recognizes it

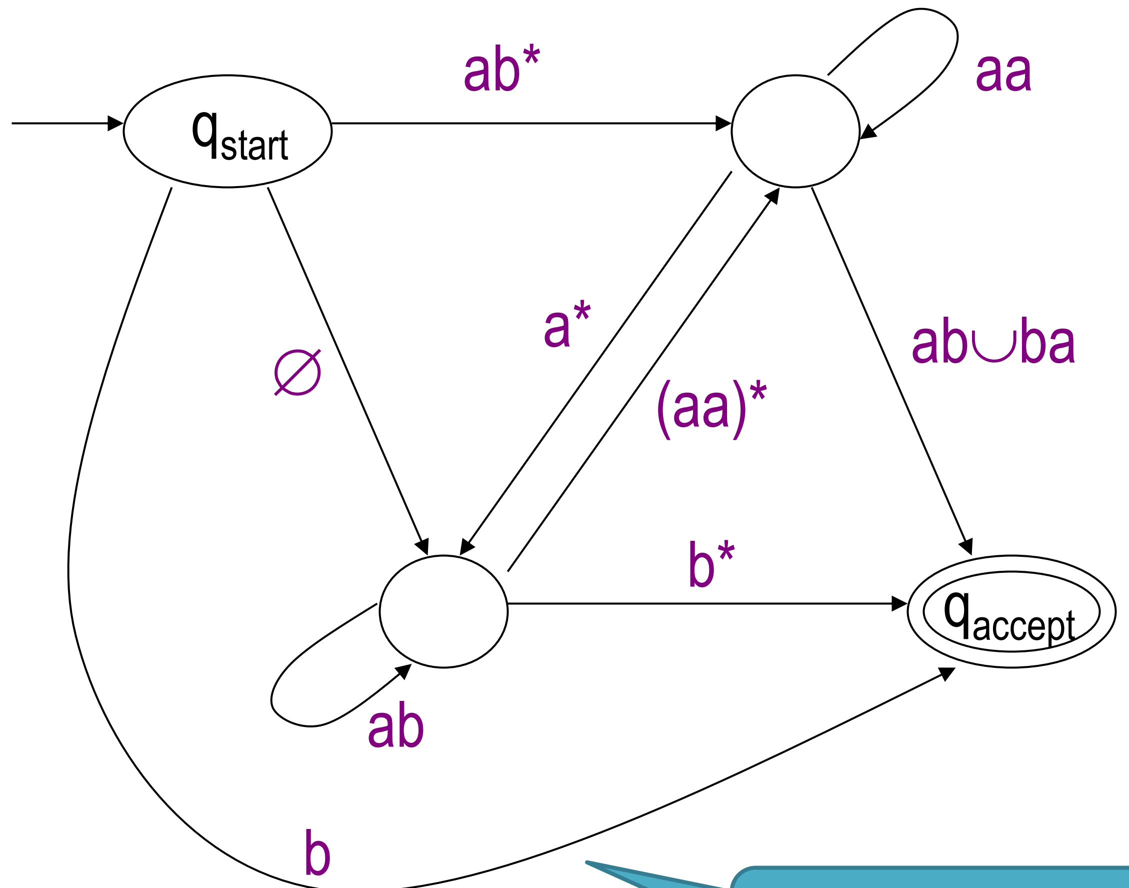
Idea: RL=DFA/NFA ⇒ ? ⇒ Regular expression

Generalized nondeterministic finite automaton, GNFA

1, create an equivalent GNFA based on DFA

2, use GNFA to create an equivalent RE

Generalized nondeterministic finite automaton

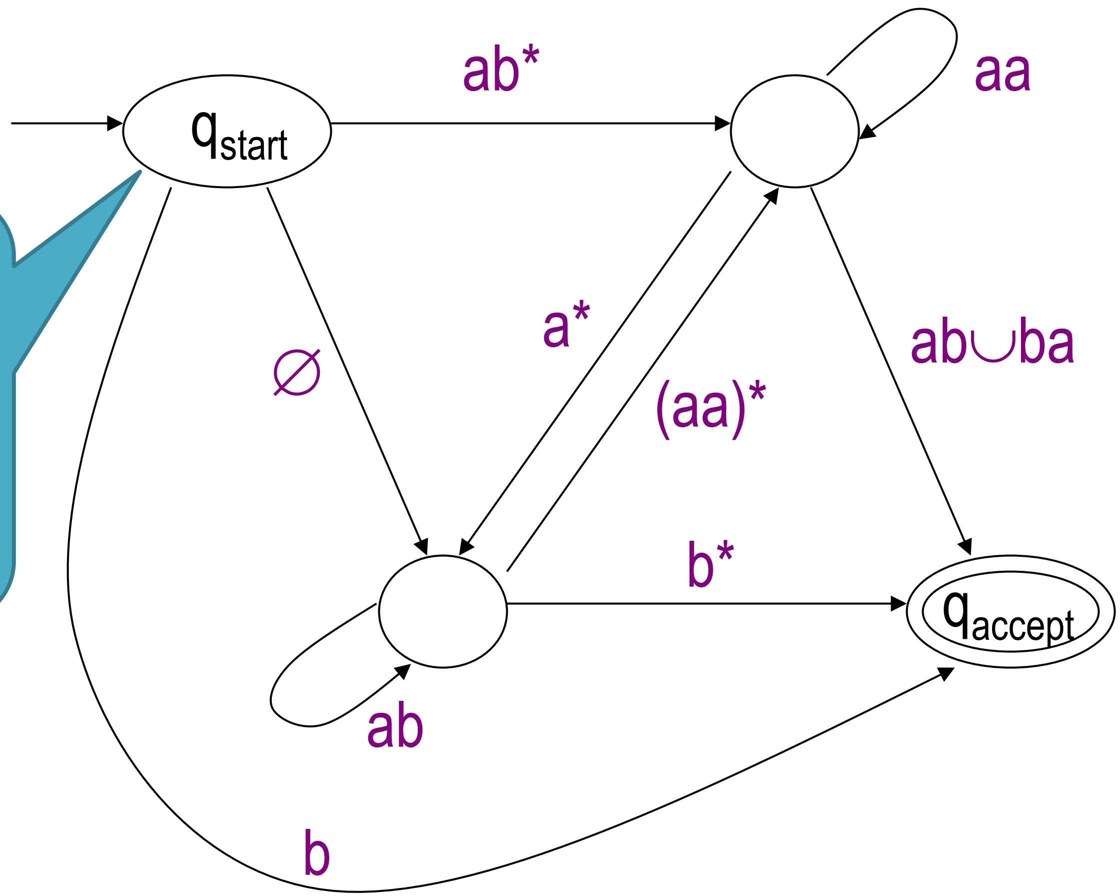


Input is not symbol, but regular expression

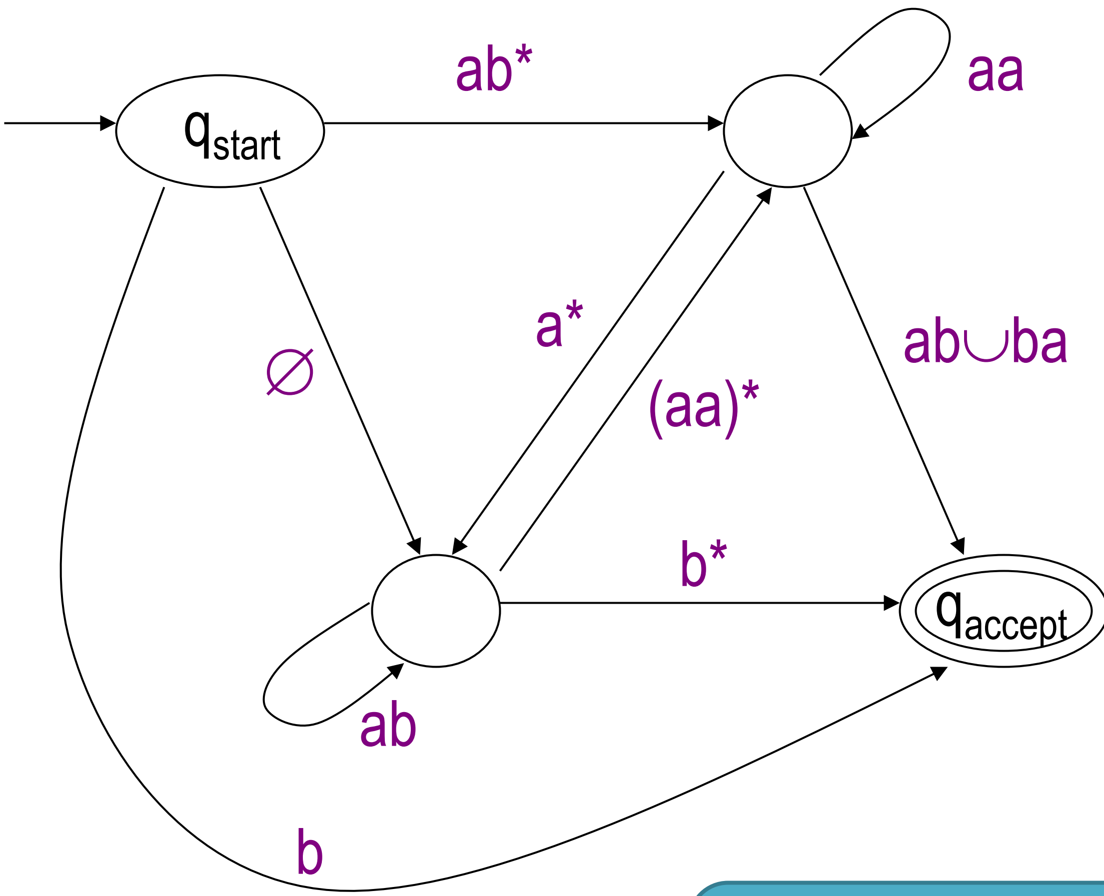
Generalized nondeterministic finite automaton

Start state:
1, can access all other states

2, no other states can access to it

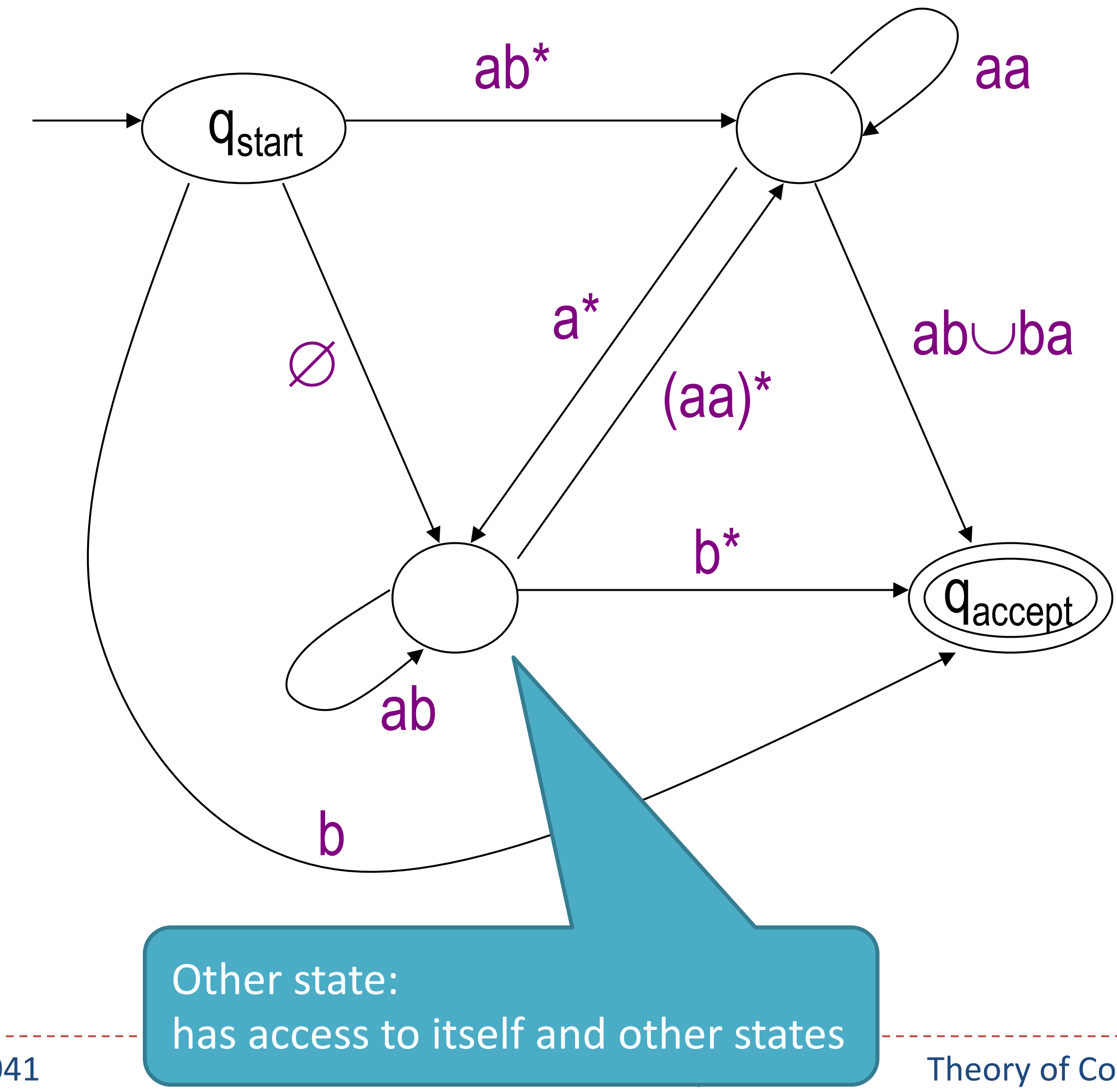


Generalized nondeterministic finite automaton



Accept state:
1, unique and different from start state
2, cannot access to other states
3, all other states can access to it

Generalized nondeterministic finite automaton



Definition of GNFA

- GNFA is a five tuple $(Q,\Sigma,\delta,q_{start},q_{accept})$
  - Q is finite set of states
  - Σ is input alphabet
  - $\delta:(Q\text{-}\{q_{accept}\})\times(Q\text{-}\{q_{start}\})\rightarrow R$

    is transition functions, means from $(Q\text{-}\{q_{accept}\})$ to $(Q\text{-}\{q_{start}\})$

    with input R
  - q_{start} is the start state
  - q_{accept} is the accept state

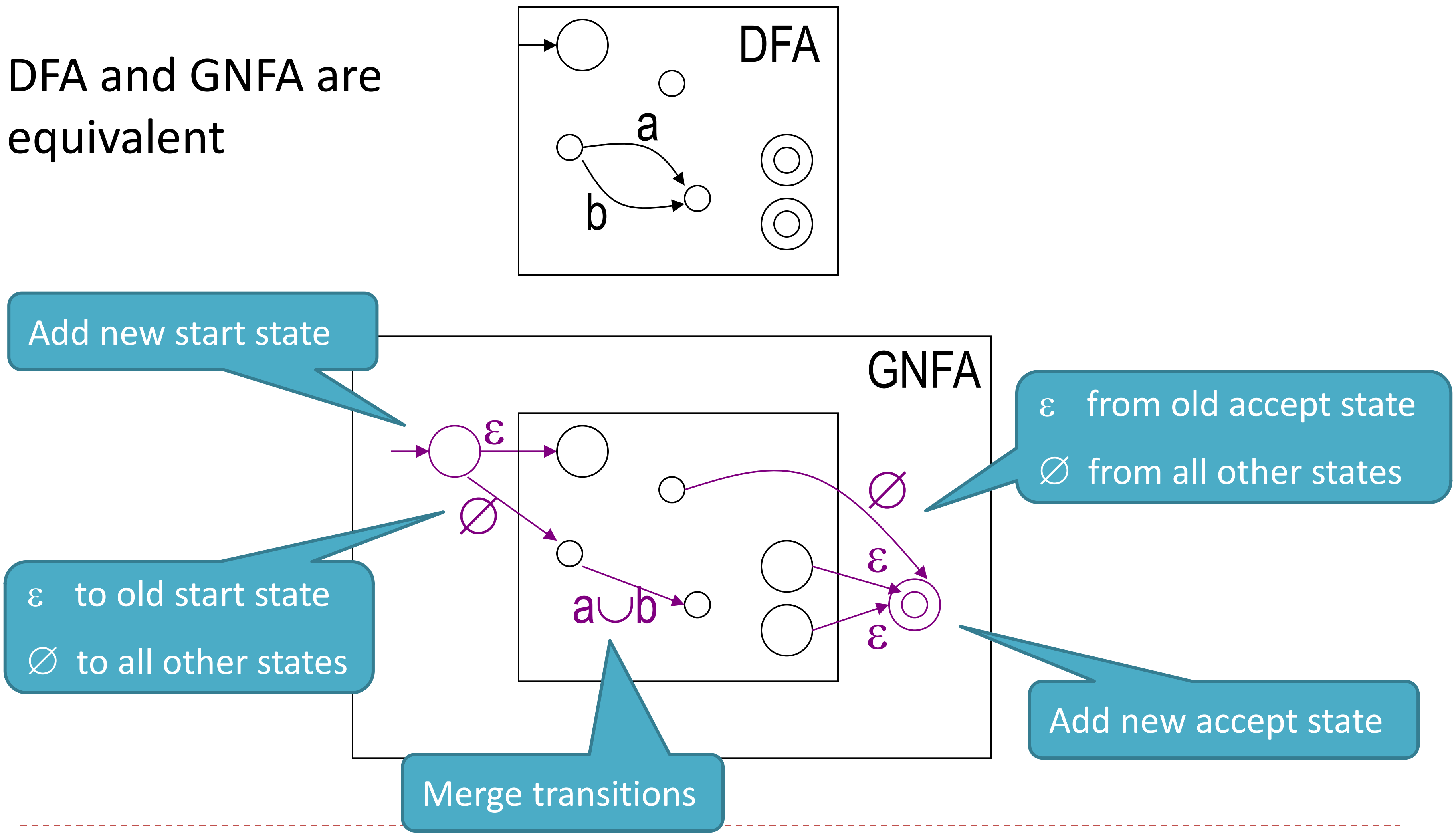
Computation on GNFA

- Input $w=w_1w_2\ldots w_k$, $w_i \in \Sigma^*$
- Computation: for state sequence $q_0,q_1,\ldots,q_k$
  - $q_0=q_{start}$ is the start state
  - $\forall i$, $w_i \in L(R_i)$, $R_i=\delta(q_{i-1},q_i)$
- Accept:
  - $q_k=q_{accept}$ is accept state

R_i	1^*0^*
⟒	
w_i	110

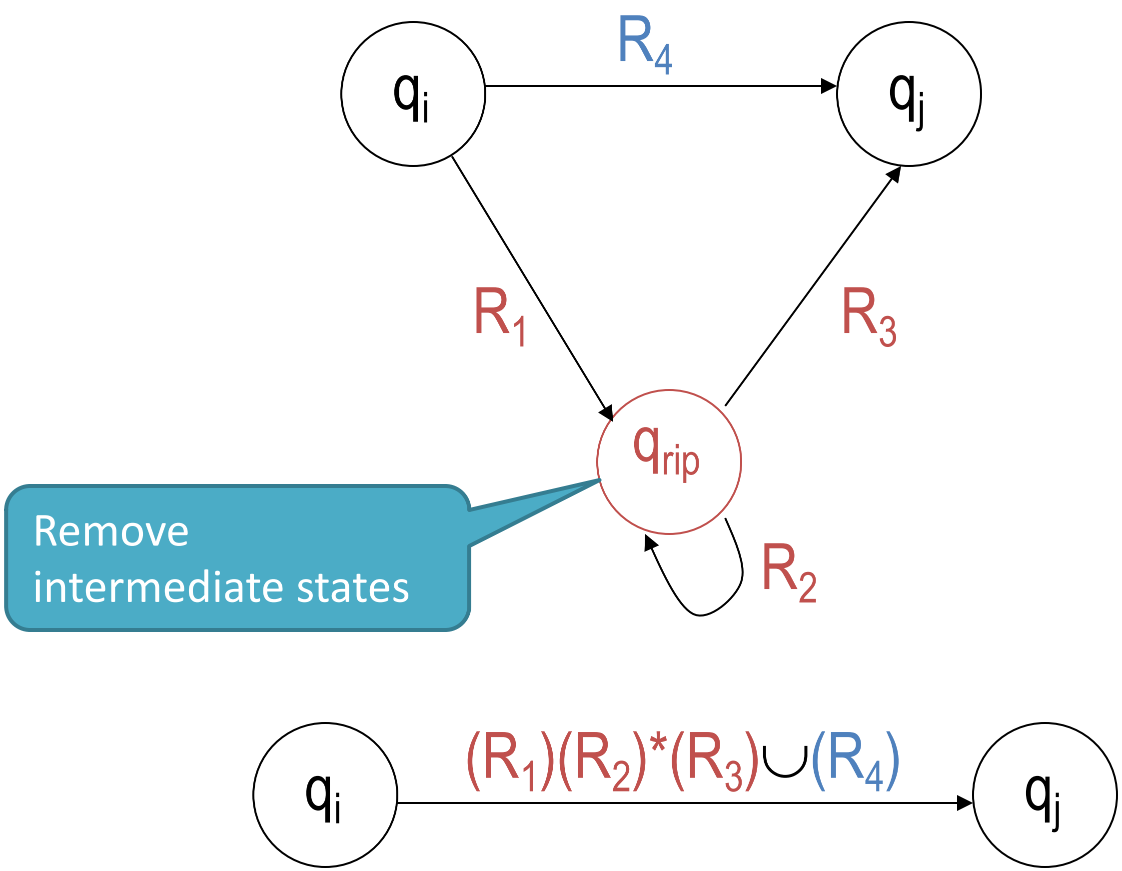
DFA/NFA ⇒ GNFA

DFA and GNFA are equivalent

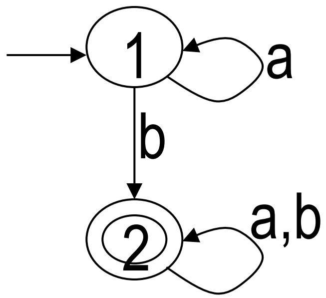


GNFA ⇒ Regular expression

- Change the number of states in GNFA to 1



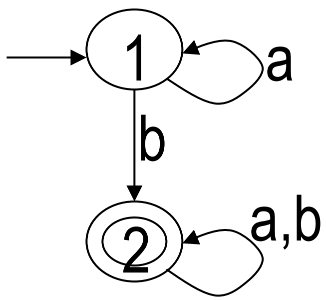
Example: DFA - -> Regular expression



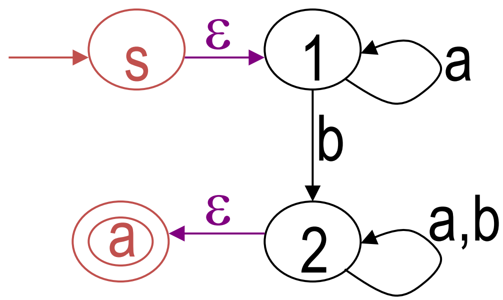
DFA

Example: DFA - -> Regular expression

Add new start and accept state



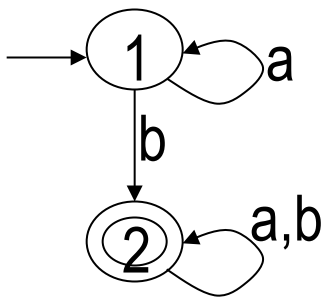
DFA



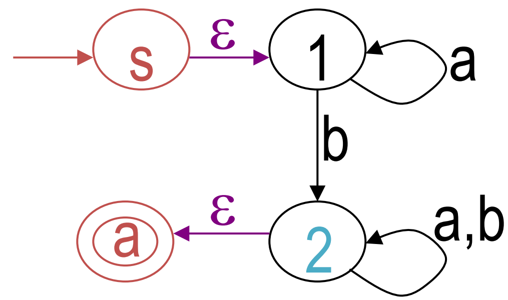
GNFA

Remove intermediate state {2}. Can you draw the new figure?

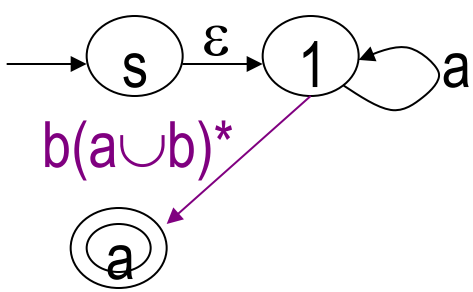
Example: DFA - -> Regular expression



DFA

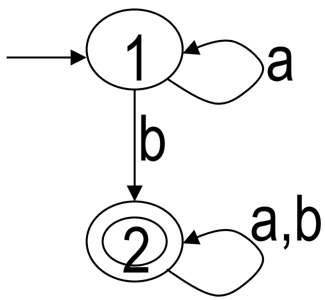


GNFA

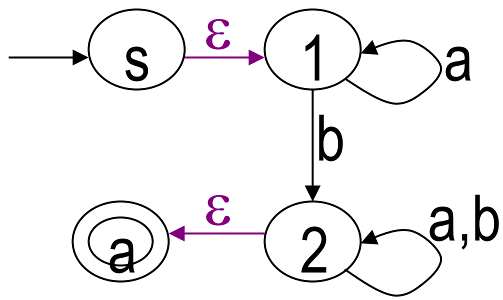


Remove intermediate state {1}. Can you draw the new figure?

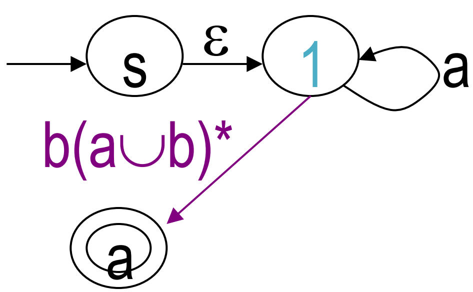
Example: DFA - -> Regular expression

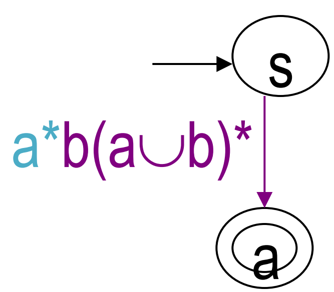


DFA

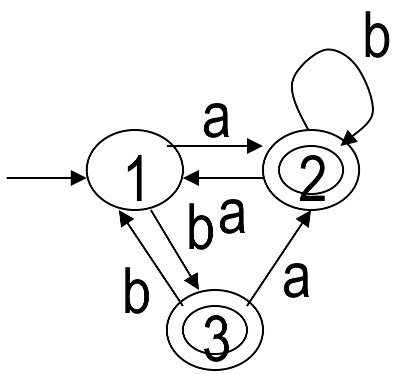


GNFA





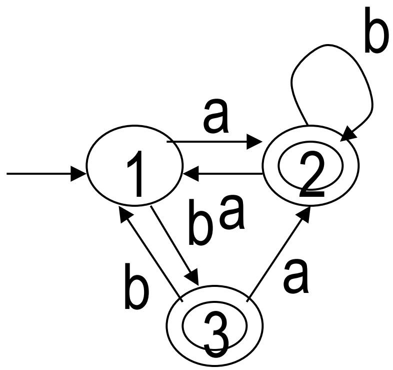
Example: DFA - -> Regular expression



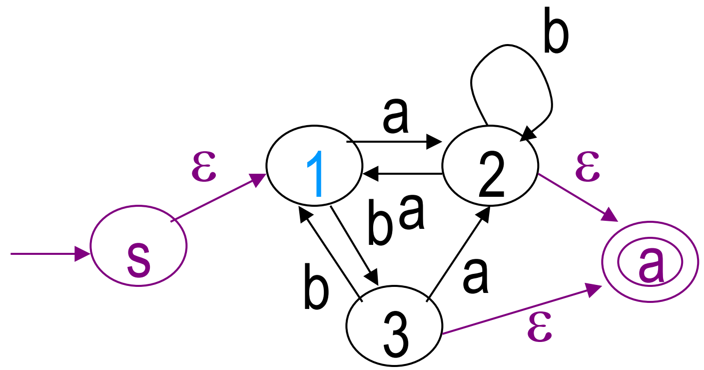
DFA

Can you draw GNFA? Adding new state and accept states.

Example: DFA - -> Regular expression



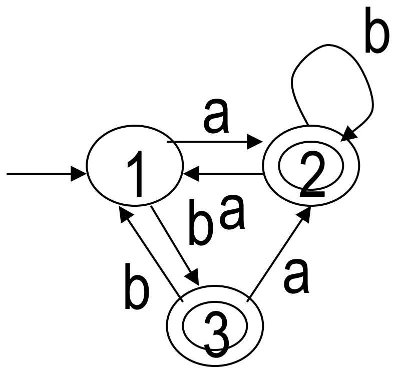
DFA



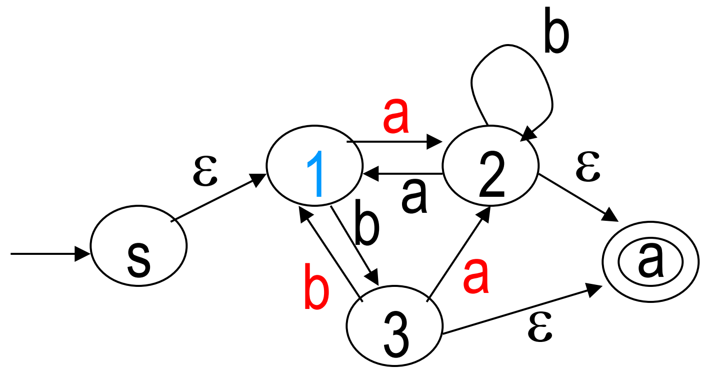
GNFA

Remove intermediate state {1}. Can you draw the new figure?

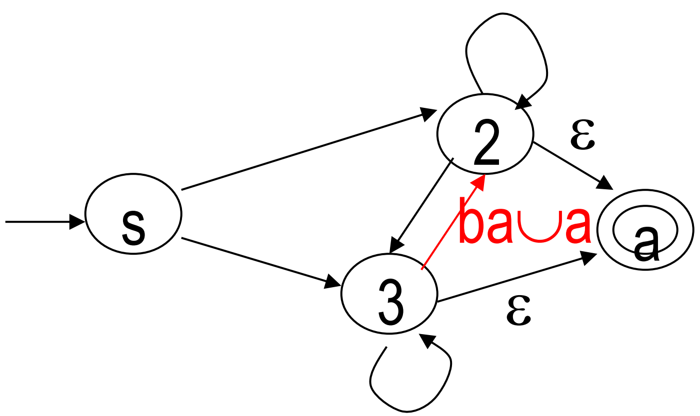
Example: DFA - -> Regular expression



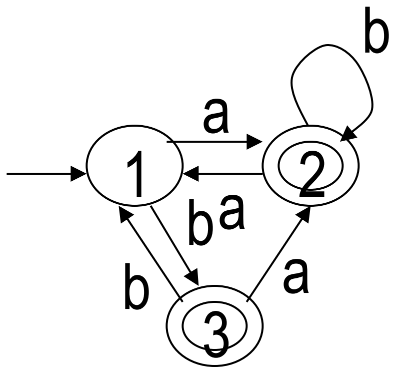
DFA



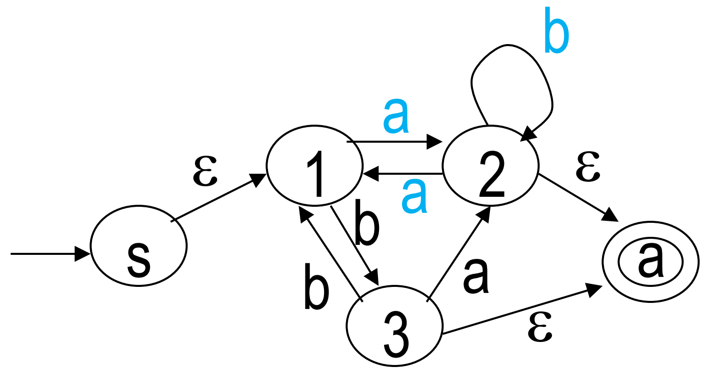
GNFA



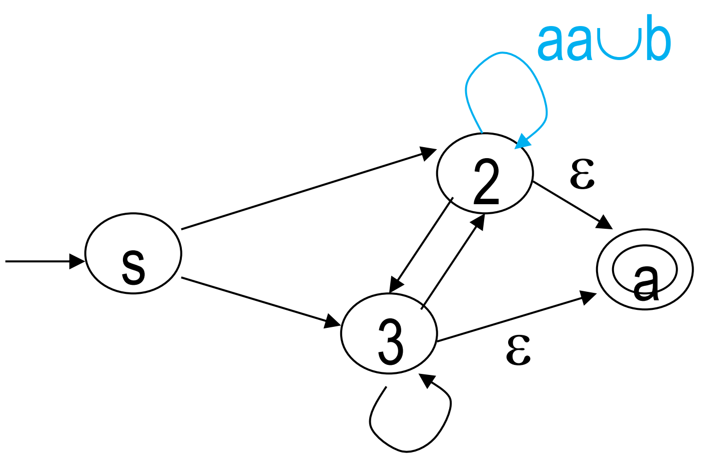
Example: DFA - -> Regular expression



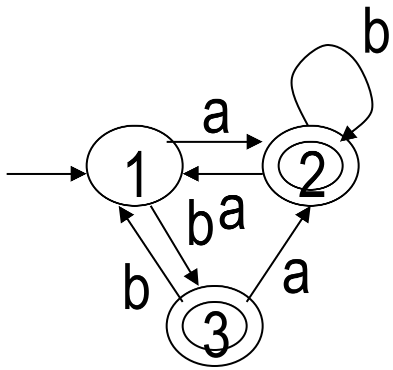
DFA



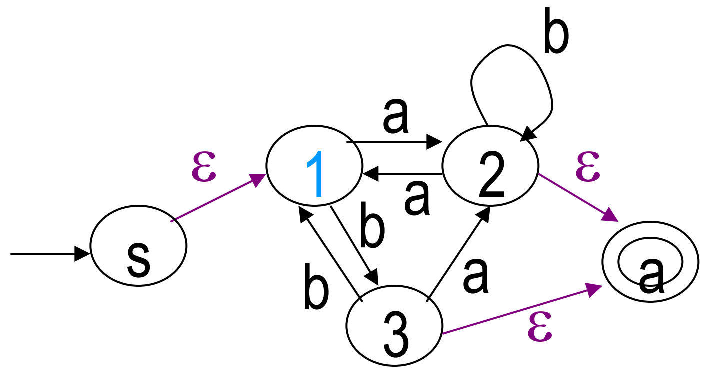
GNFA



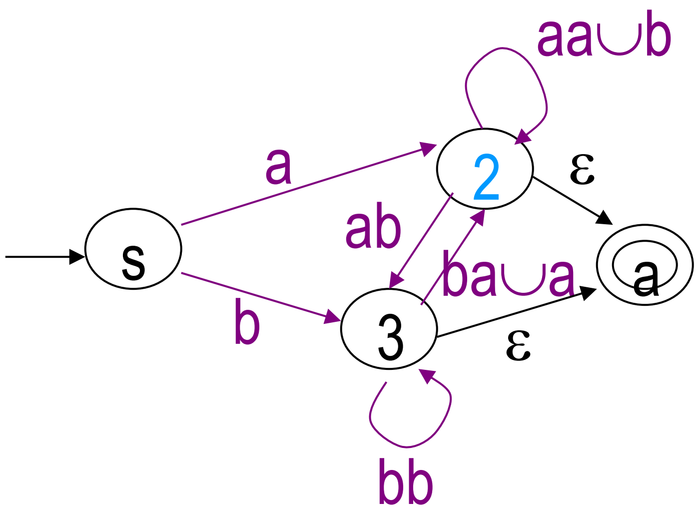
Example: DFA - -> Regular expression



DFA

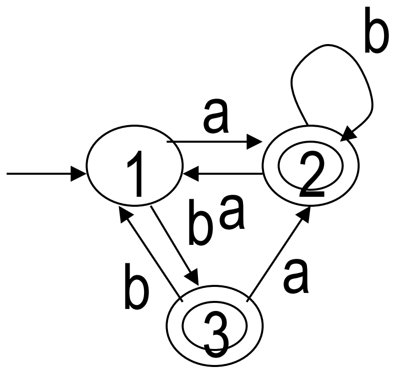


GNFA

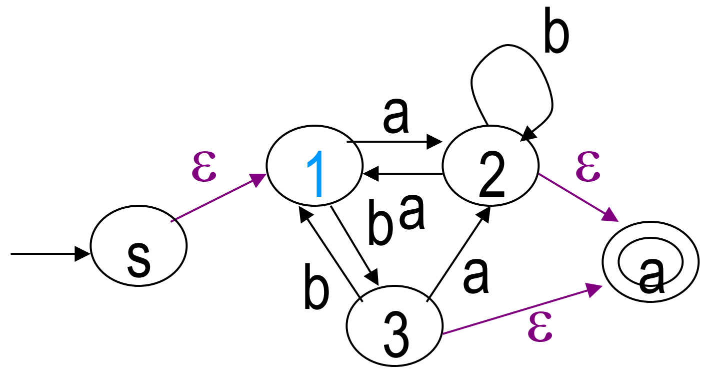


Remove intermediate state {2}. Can you draw the new figure?

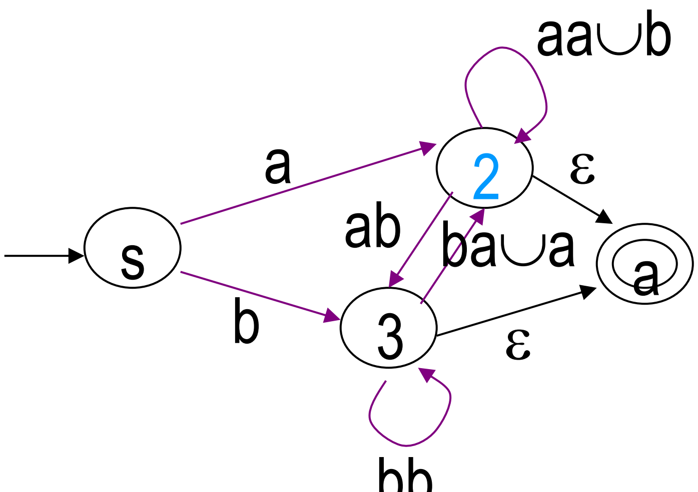
Example: DFA - -> Regular expression

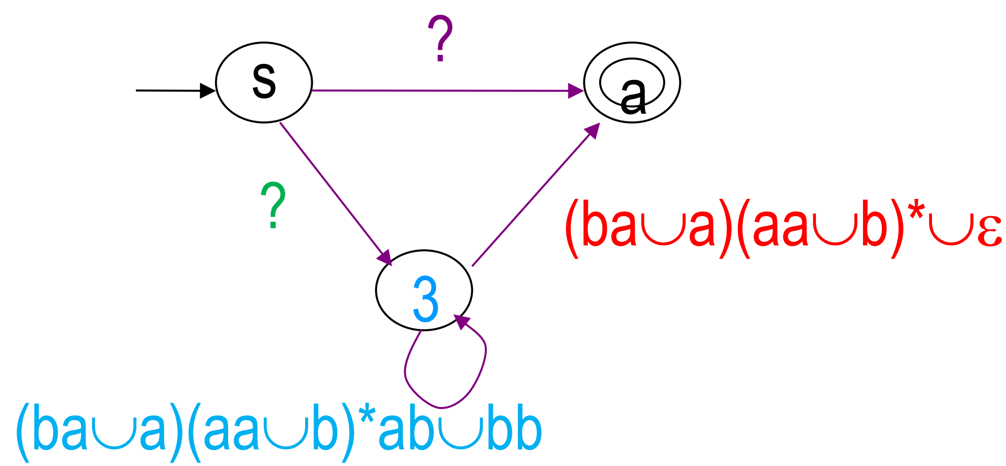


DFA

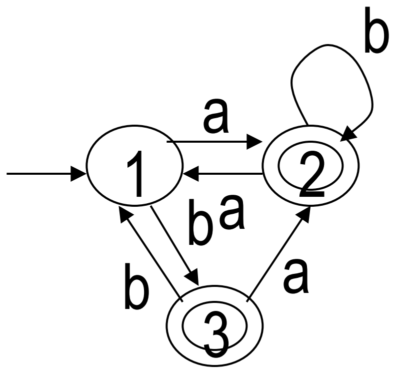


GNFA

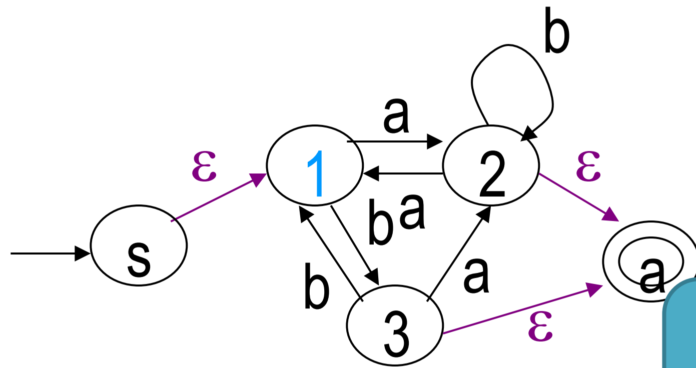




Example: DFA - -> Regular expression

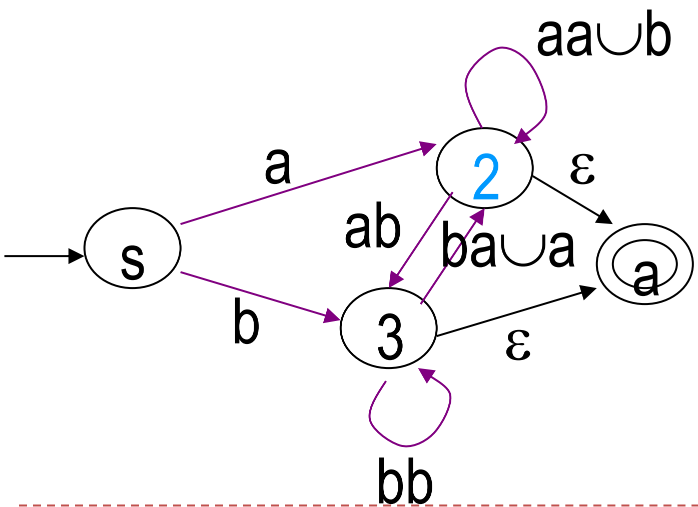


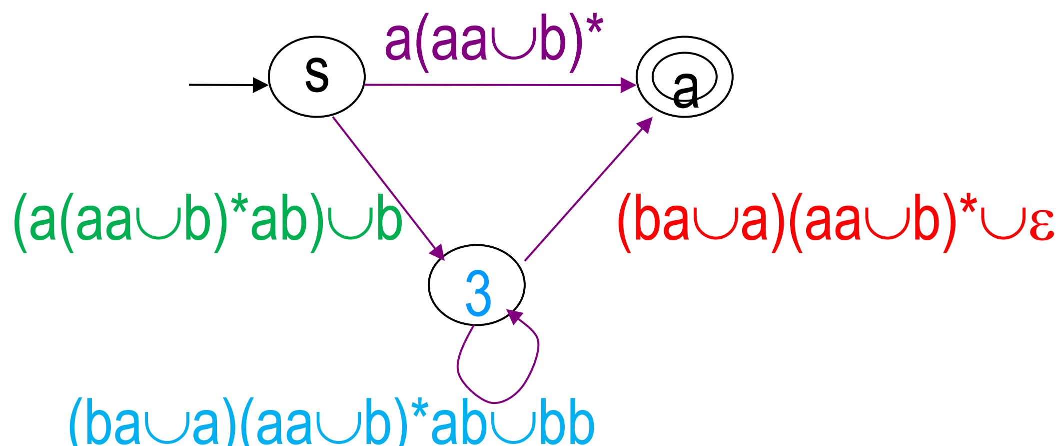
DFA



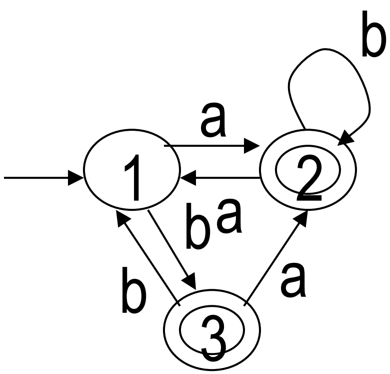
GNFA

Remove intermediate state {3}. Can you draw the new figure?

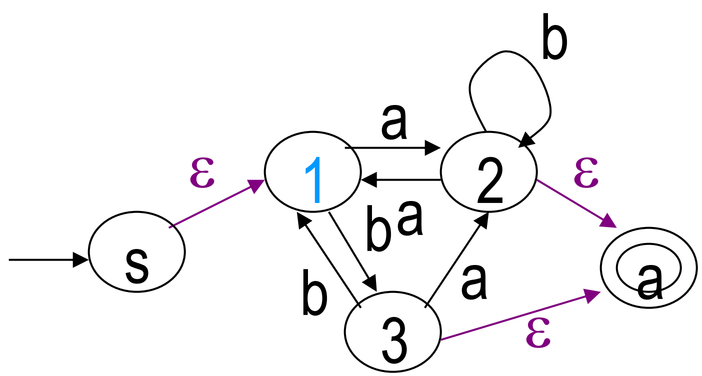




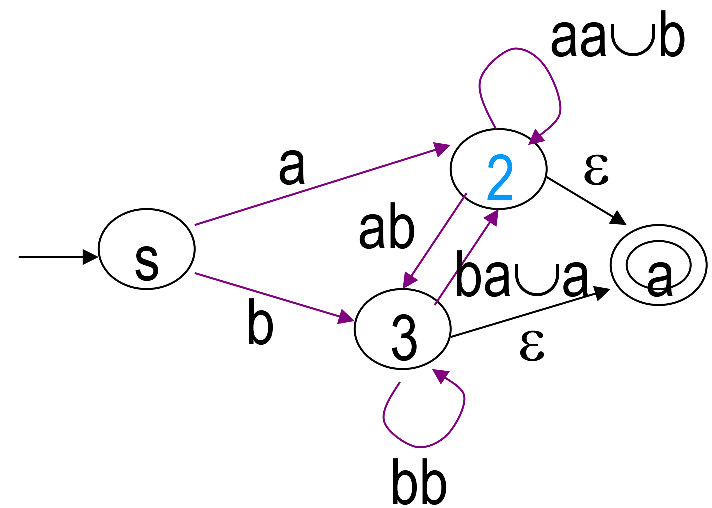
Example: DFA - -> Regular expression

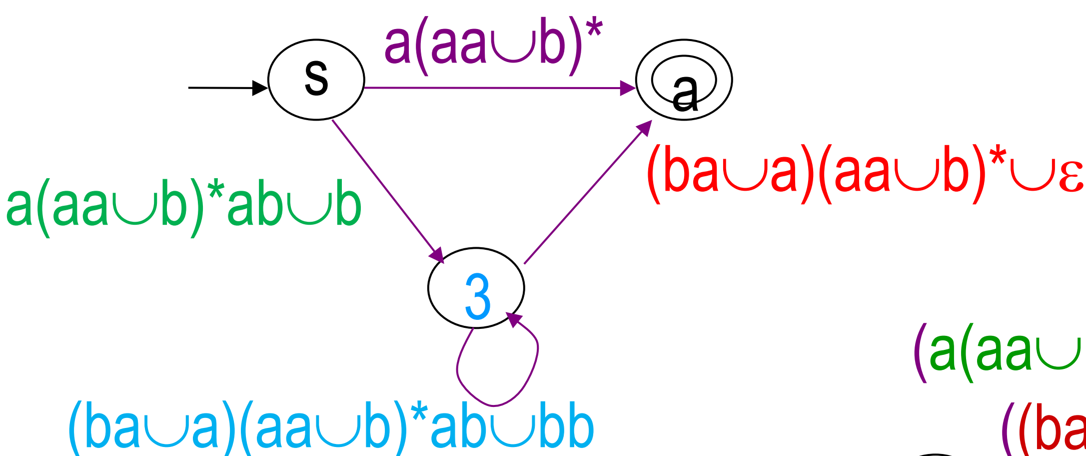


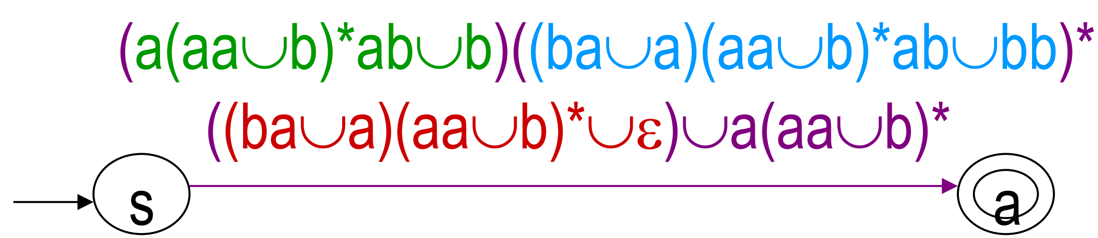
DFA



GNFA

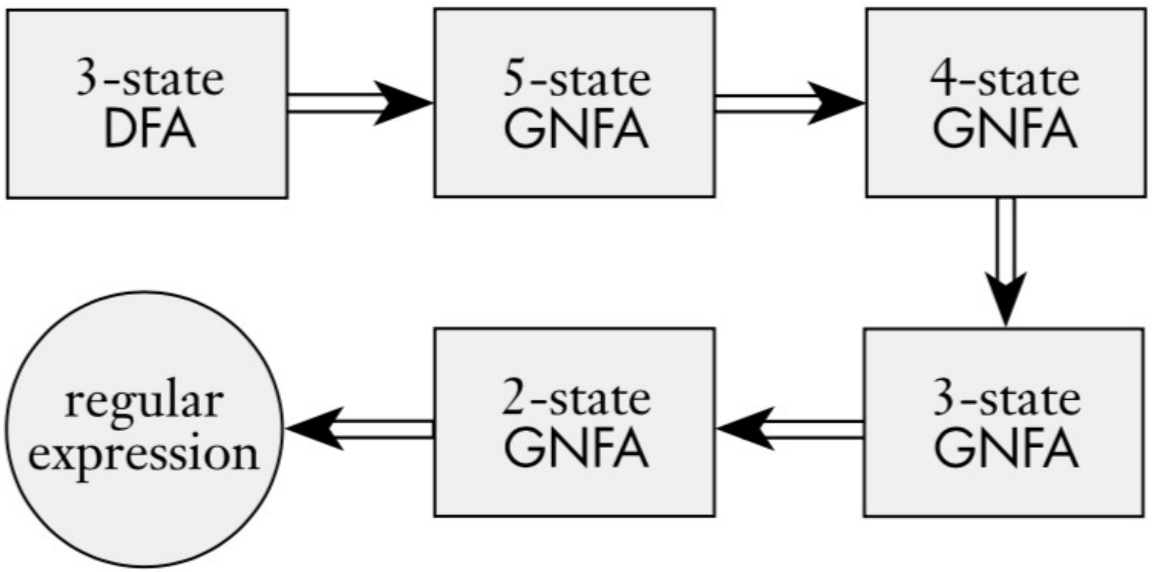






DFA ⇒ GNFA ⇒ Regular expression

Add start/accept state



Regular language <==> Regular expression

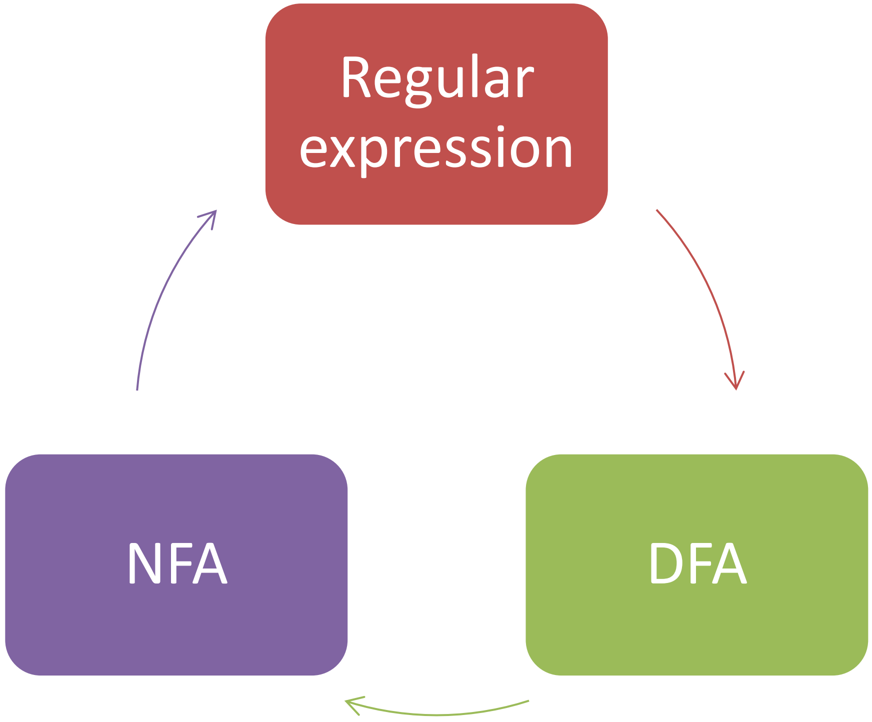
- **Theorem: A language is regular if and only if some regular expression describes it.**

- Regular language ==> Regular expression

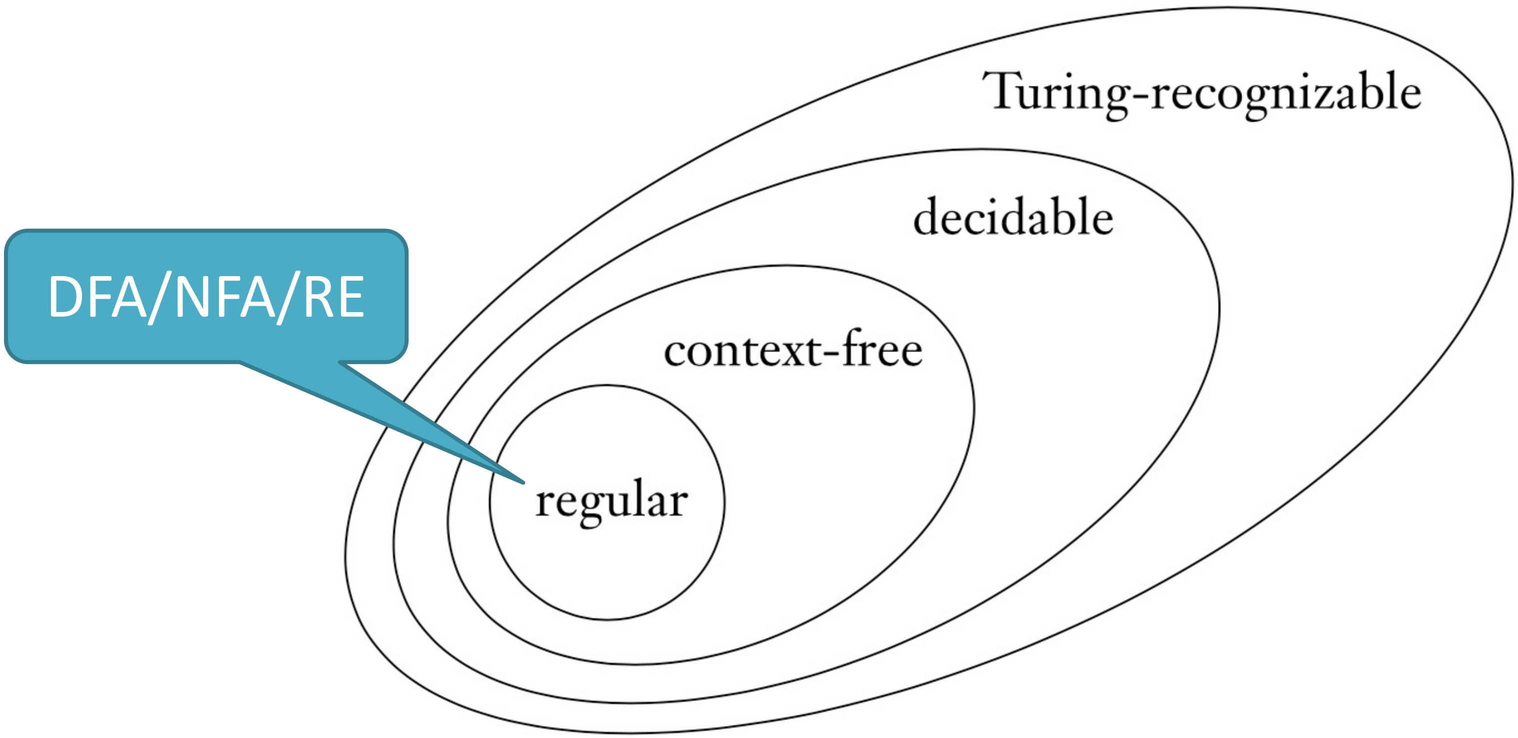
- Regular language <== Regular expression

Regular language: DFA, NFA, Regular expression

- A language is regular if some <u>deterministic finite automaton</u> recognizes it
- A language is regular if and only if some <u>nondeterministic finite automaton</u> recognizes it
- A language is regular if and only if some <u>regular expression</u> describes it



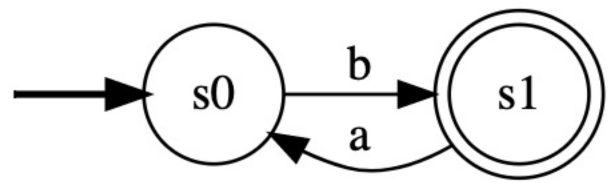
Regular language in big picture



DFA/NFA → RE web tool

- http://ivanzuzak.info/noam/webapps/fsm2regex/

#states
s0
s1
s2
#initial
s0
#accepting
s1
#alphabet
a
b
#transitions
s0:b>s1
s1:a>s0



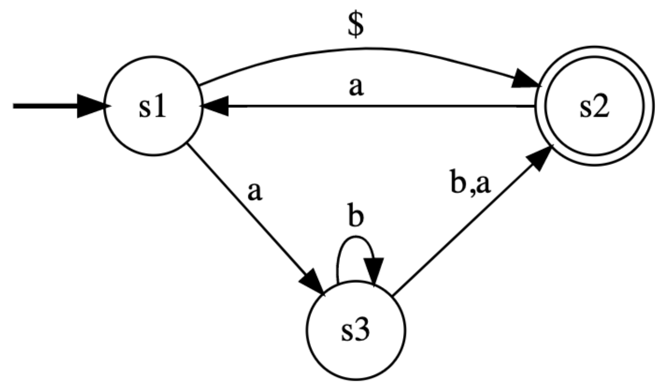
b+($+ba)(ba)*b

the $ character representing the empty string
p+ : Any string containing one or more p's.

DFA/NFA → RE web tool

- http://ivanzuzak.info/noam/webapps/fsm2regex/

#states
s1
s2
s3
#initial
s1
#accepting
s2
#alphabet
a
b
#transitions
s1:$>s2
s1:a>s3
s2:a>s1
s3:b>s3
s3:b>s2
s3:a>s2



$+aa*(b(b+aaa*b)*(a+a(a+aa*(a+$+b))+b+$)+a+$+b)+a

the $ character representing the empty string
p+ : Any string containing one or more p's.

Conclusion

- Regular expression
  - Definition
  - Example

- Equivalence with DFA/NFA
  - Regular expression $\Rightarrow$ Regular language
  - Regular expression $\Leftarrow$ Regular language