CS375: Logic and Theory of Computing

Fuhua (Frank) Cheng

Department of Computer Science

University of Kentucky

Table of Contents:

- **Week 1: Preliminaries (set algebra, relations, functions) (read Chapters 1-4)**
- **Weeks 2-5: Regular Languages, Finite Automata (Chapter 11)**
- **Weeks 6-8: Context-Free Languages, Pushdown Automata (Chapters 12)**
- **Weeks 9-11: Turing Machines (Chapter 13)**

Table of Contents (conti):

- **Weeks 12-13: Propositional Logic (Chapter 6), Predicate Logic (Chapter 7), Computational Logic (Chapter 9), Algebraic Structures (Chapter 10)**

6. Regular Languages & Finite Automata

- Constructing Efficient Finite Automata

Question: can we get a most efficient regular expression for a given NFA?

most compact

Why?

YES, in two steps:

{NFAs} ⟺ {DFAs}

First, transform the given NFA to a DFA

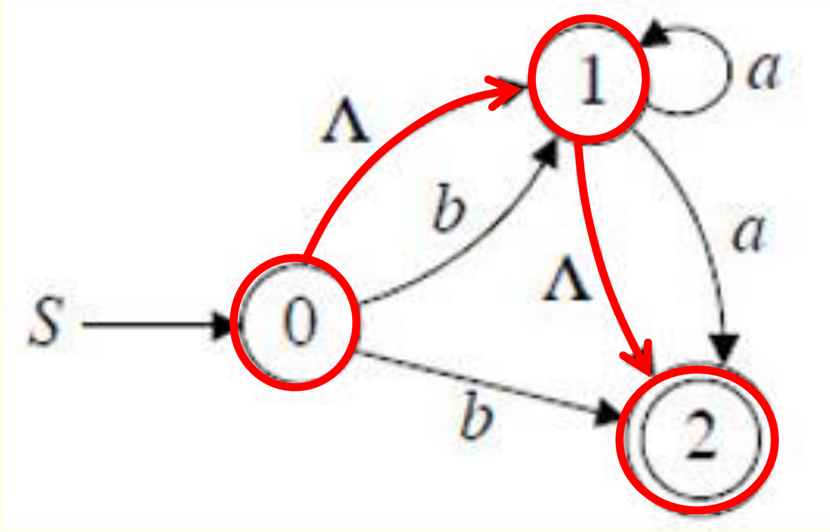
Then, transform the DFA to a minimum-state DFA

Constructing Efficient Finite Automata

Transforming an NFA to a DFA

The Λ-closure of a state *s*, denoted *Λ(s)* , is the set consisting of *s* together with all states that can be reached from *s* by traversing *Λ-edges*.

The *Λ-closure* of a set S of states, denoted *Λ(S)* , is the union of the *Λ-closures* of the states in S.



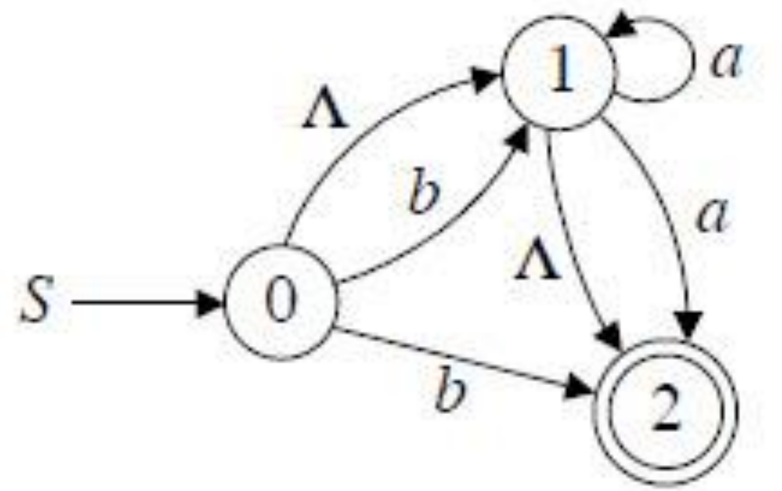
Λ(0)= ?

Λ(0)= {0, 1, 2}

Λ({0, 1})= Λ(0) U Λ(1)
= {0, 1, 2} U {1, 2}
= {0, 1, 2}

Constructing Efficient Finite Automata

The same NFA again:note



T_N	a	b	Λ
S 0	$\varnothing$	$\{1,2\}$	$\{1\}$
1	$\{1,2\}$	$\varnothing$	$\{2\}$
F 2	$\varnothing$	$\varnothing$	$\varnothing$

$$\Lambda(0) = \{0,1,2\} \qquad \Lambda(\{1,2\}) = \{1,2\}$$

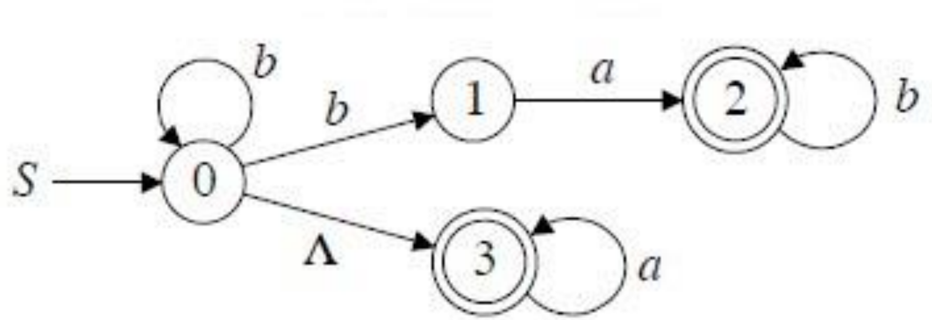
$$\Lambda(1) = \{1,2\} \qquad \Lambda(\{0,1,2\}) = \{0,1,2\}$$

$$\Lambda(2) = \{2\}$$

$$\boxed{\Lambda(\varnothing) = \varnothing}$$

Constructing Efficient Finite Automata

Example. Given the following NFA.



	T_N	a	b	Λ
S	0	$\varnothing$	$\{0, 1\}$	$\{3\}$
	1	$\{2\}$	$\varnothing$	$\varnothing$
F	2	$\varnothing$	$\{2\}$	$\varnothing$
F	3	$\{3\}$	$\varnothing$	$\varnothing$

To transform it to a DFA, we first construct Λ-closures of the states of the given NFA:

First:

$$\Lambda(0) = \{0,3\}$$

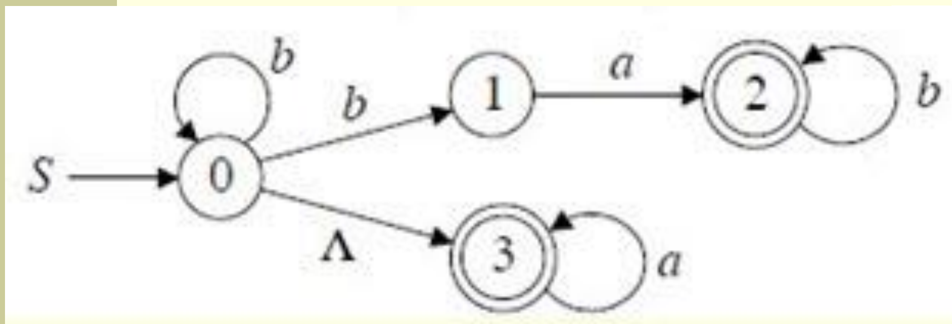
$$\Lambda(1) = \{1\}$$

$$\Lambda(2) = \{2\}$$

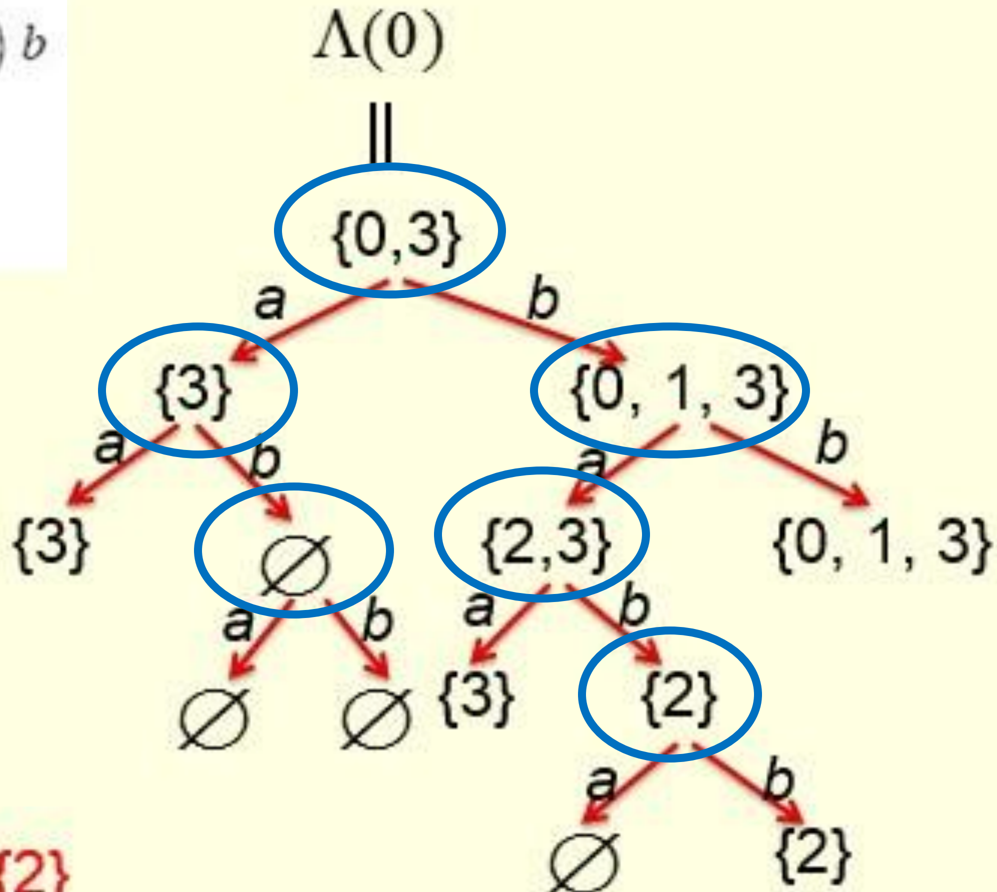
$$\Lambda(3) = \{3\}$$

Start state

Constructing Efficient Finite Automata



Then, how to get the States of the DFA?

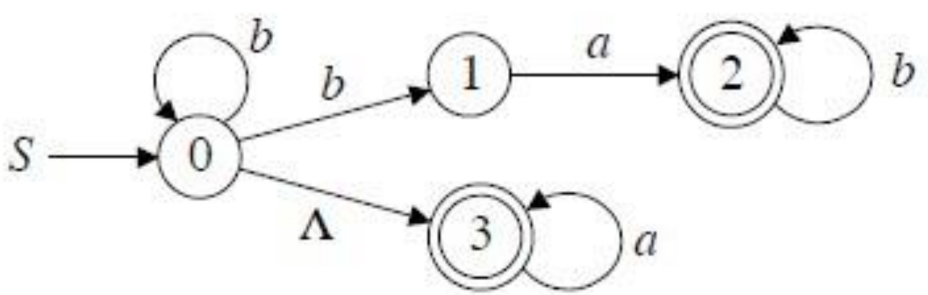


1. Build the tree on the right
2. Identify all distinct nodes

{0,3}, {3}, {0, 1, 3}, ∅, {2,3}, {2}

Constructing Efficient Finite Automata

Example. Given the following NFA.



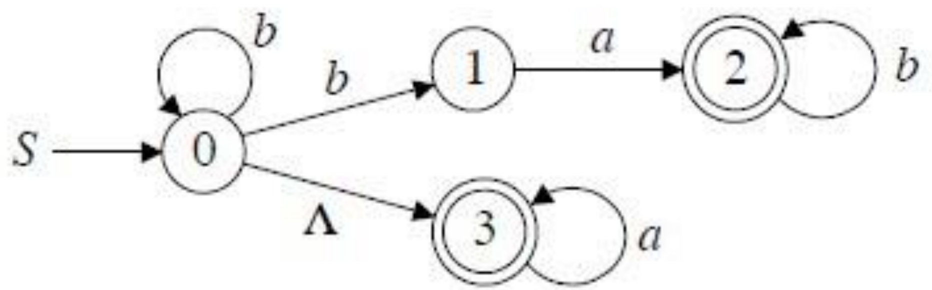
	T_N	a	b	Λ
S	0	$\varnothing$	$\{0, 1\}$	$\{3\}$
	1	$\{2\}$	$\varnothing$	$\varnothing$
F	2	$\varnothing$	$\{2\}$	$\varnothing$
F	3	$\{3\}$	$\varnothing$	$\varnothing$

$$\boxed{\begin{aligned}\Lambda(0) &= \{0,3\} \\ \Lambda(1) &= \{1\} \\ \Lambda(2) &= \{2\} \\ \Lambda(3) &= \{3\}\end{aligned}}$$

$$\boxed{\begin{aligned}T_D(\{0,3\}, a) &= \Lambda(\underline{T_N(0, a)} \cup \underline{T_N(3, a)}) \\ &= \Lambda(\underline{\varnothing} \cup \underline{\{3\}}) \\ &= \Lambda(\underline{\{3\}}) \\ &= \underline{\{3\}}\end{aligned}}$$

Constructing Efficient Finite Automata

Example. Given the following NFA.

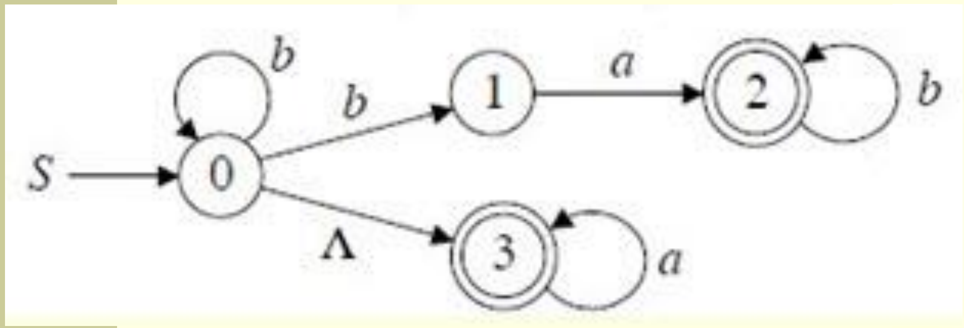


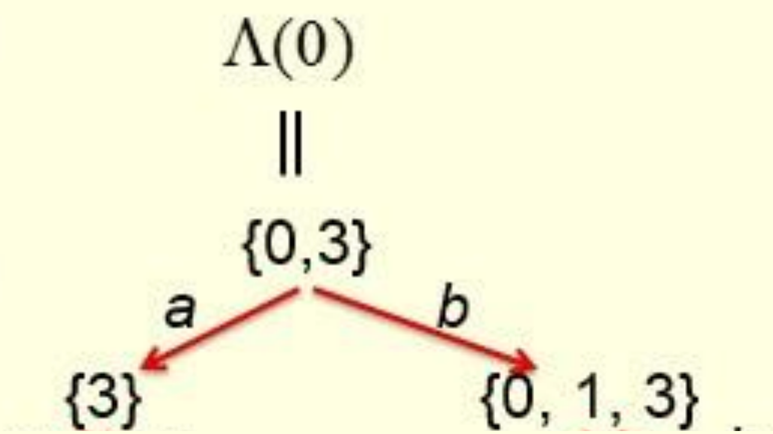
	T_N	a	b	Λ
S	0	$\varnothing$	$\{0, 1\}$	$\{3\}$
	1	$\{2\}$	$\varnothing$	$\varnothing$
F	2	$\varnothing$	$\{2\}$	$\varnothing$
F	3	$\{3\}$	$\varnothing$	$\varnothing$

$$\Lambda(0) = \{0,3\}$$
$$\Lambda(1) = \{1\}$$
$$\Lambda(2) = \{2\}$$
$$\Lambda(3) = \{3\}$$

$$\begin{aligned} T_D(\{0,3\}, b) &= \Lambda(\underline{T_N(0, b)} \cup \underline{T_N(3, b)}) \\ &= \Lambda(\underline{\{0,1\}} \cup \underline{\varnothing}) \\ &= \Lambda(\underline{\{0,1\}}) \\ &= \underline{\Lambda(0)} \cup \underline{\Lambda(1)} \\ &= \underline{\{0,3\}} \cup \underline{\{1\}} = \underline{\{0,1,3\}} \end{aligned}$$

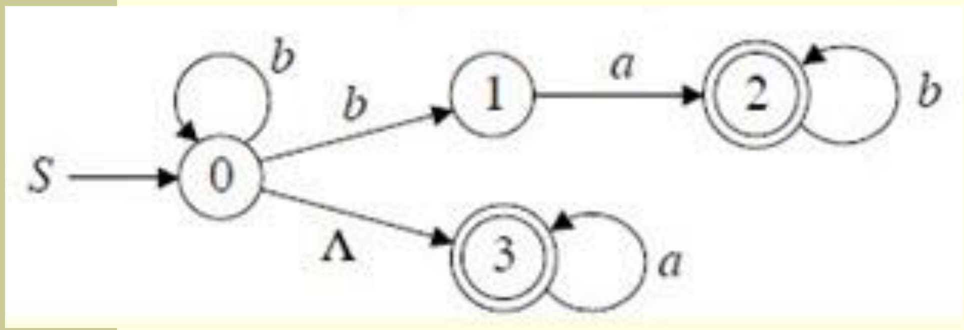
Constructing Efficient Finite Automata

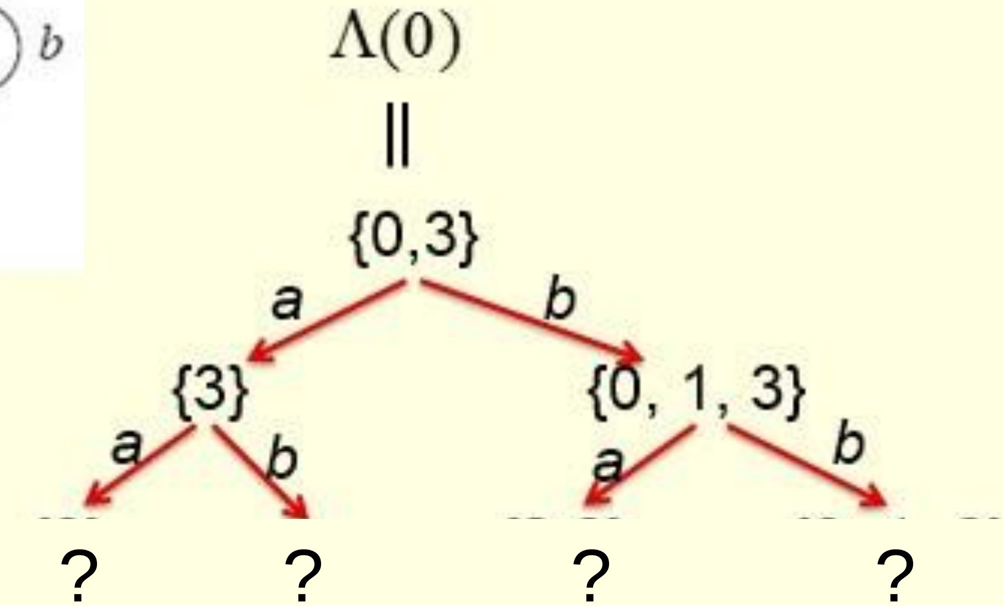




So, we have

Constructing Efficient Finite Automata

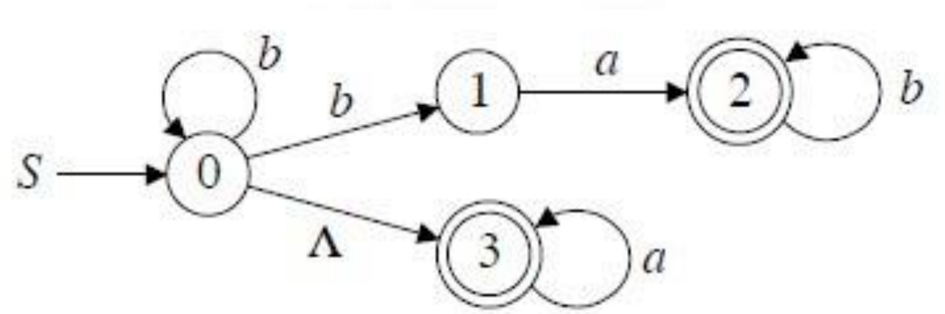




Then, build the next level of the tree:

Constructing Efficient Finite Automata

Example. Given the following NFA.



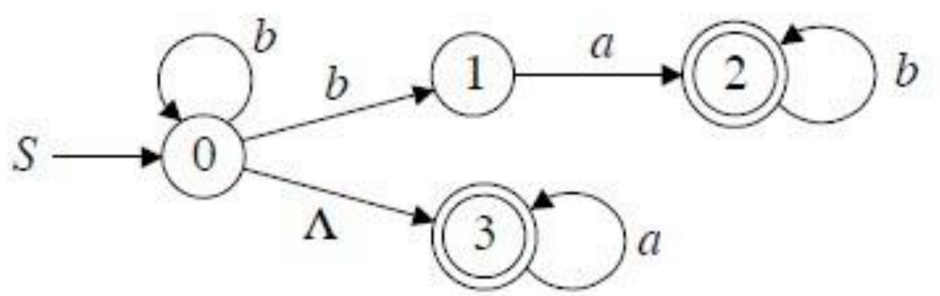
	T_N	a	b	Λ
S	0	$\varnothing$	$\{0, 1\}$	$\{3\}$
	1	$\{2\}$	$\varnothing$	$\varnothing$
F	2	$\varnothing$	$\{2\}$	$\varnothing$
F	3	$\{3\}$	$\varnothing$	$\varnothing$

$$\Lambda(0) = \{0,3\}$$
$$\Lambda(1) = \{1\}$$
$$\Lambda(2) = \{2\}$$
$$\Lambda(3) = \{3\}$$

$$\begin{aligned} T_D(\{3\}, a) &= \Lambda(T_N(3, a)) \\ &= \Lambda(\{3\}) \\ &= \Lambda(3) \\ &= \{3\} \end{aligned}$$

Constructing Efficient Finite Automata

Example. Given the following NFA.

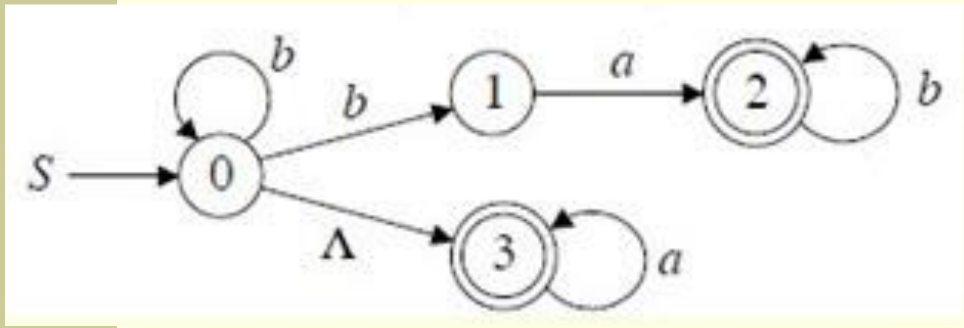


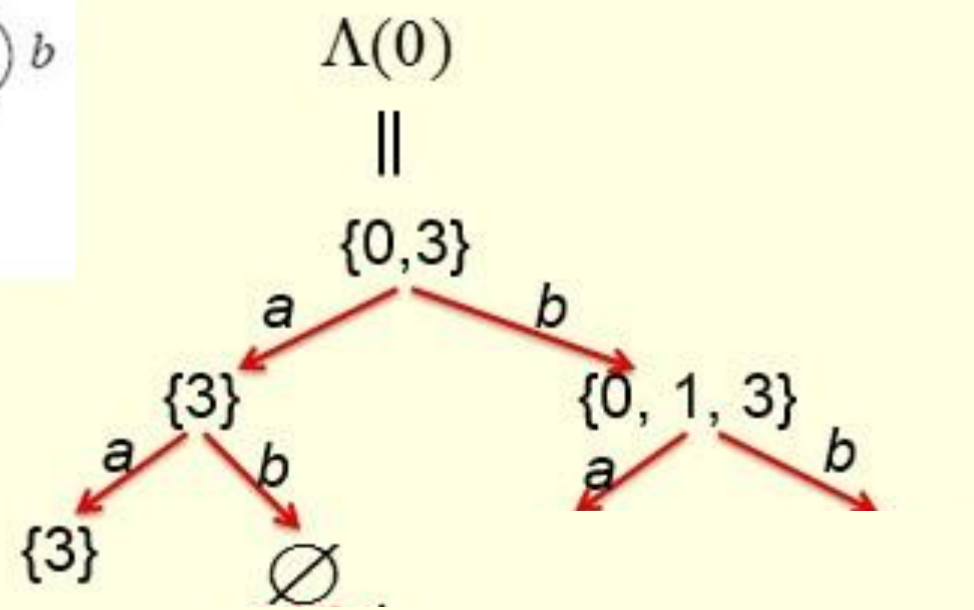
	T_N	a	b	Λ
S	0	$\varnothing$	$\{0, 1\}$	$\{3\}$
	1	$\{2\}$	$\varnothing$	$\varnothing$
F	2	$\varnothing$	$\{2\}$	$\varnothing$
F	3	$\{3\}$	$\varnothing$	$\varnothing$

$$\boxed{\begin{aligned}\Lambda(0) &= \{0,3\}\\ \Lambda(1) &= \{1\}\\ \Lambda(2) &= \{2\}\\ \Lambda(3) &= \{3\}\end{aligned}}$$

$$\boxed{\begin{aligned}T_D(\{3\}, b) &= \Lambda(\underline{T_N(3, b)})\\ &= \Lambda(\underline{\varnothing})\\ &= \underline{\varnothing}\end{aligned}}$$

Constructing Efficient Finite Automata

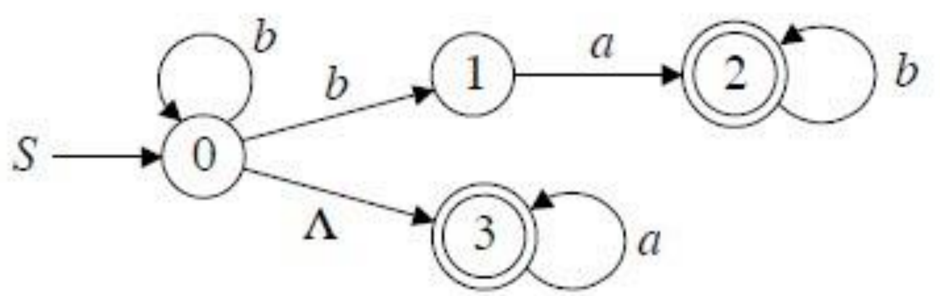




So we have the child nodes of {3}.
Then find the child nodes of {0, 1, 3}

Constructing Efficient Finite Automata

Example. Given the following NFA.



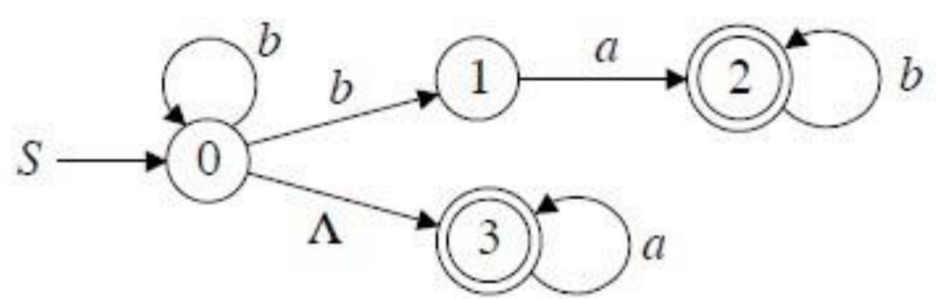
	T_N	a	b	Λ
S	0	$\varnothing$	$\{0, 1\}$	$\{3\}$
	1	$\{2\}$	$\varnothing$	$\varnothing$
F	2	$\varnothing$	$\{2\}$	$\varnothing$
F	3	$\{3\}$	$\varnothing$	$\varnothing$

$$\Lambda(0) = \{0,3\}$$
$$\Lambda(1) = \{1\}$$
$$\Lambda(2) = \{2\}$$
$$\Lambda(3) = \{3\}$$

$$\begin{aligned} T_D(\{0,1,3\}, a) &= \Lambda(T_N(0,a) \cup T_N(1,a) \cup T_N(3,a)) \\ &= \Lambda(\varnothing \cup \{2\} \cup \{3\}) \\ &= \Lambda(\{2,3\}) \\ &= \Lambda(2) \cup \Lambda(3) = \{2\} \cup \{3\} = \{2,3\} \end{aligned}$$

Constructing Efficient Finite Automata

Example. Given the following NFA.



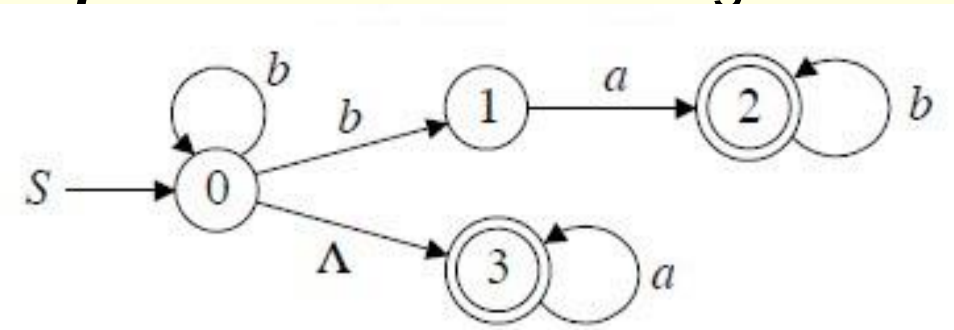
	T_N	a	b	Λ
S	0	$\varnothing$	$\{0, 1\}$	$\{3\}$
	1	$\{2\}$	$\varnothing$	$\varnothing$
F	2	$\varnothing$	$\{2\}$	$\varnothing$
F	3	$\{3\}$	$\varnothing$	$\varnothing$

$$\Lambda(0) = \{0,3\}$$
$$\Lambda(1) = \{1\}$$
$$\Lambda(2) = \{2\}$$
$$\Lambda(3) = \{3\}$$

$$\begin{aligned} T_D(\{0,1,3\}, b) &= \Lambda(T_N(0,b) \cup T_N(1,b) \cup T_N(3,b)) \\ &= \Lambda(\{0,1\} \cup \Phi \cup \Phi) \\ &= \Lambda(\{0,1\}) \\ &= \Lambda(0) \cup \Lambda(1) = \{0,3\} \cup \{1\} = \{0,1,3\} \end{aligned}$$

Constructing Efficient Finite Automata

Example. Given the following NFA.



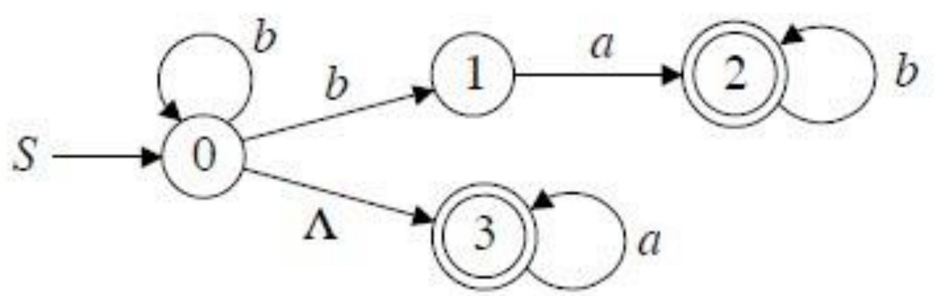
	T_N	a	b	Λ
S	0	$\varnothing$	$\{0, 1\}$	$\{3\}$
	1	$\{2\}$	$\varnothing$	$\varnothing$
F	2	$\varnothing$	$\{2\}$	$\varnothing$
F	3	$\{3\}$	$\varnothing$	$\varnothing$

$$\boxed{\begin{aligned}\Lambda(0) &= \{0,3\}\\ \Lambda(1) &= \{1\}\\ \Lambda(2) &= \{2\}\\ \Lambda(3) &= \{3\}\end{aligned}}$$

$$\boxed{\begin{aligned}T_D(\Phi, a) &= \Lambda(\underline{T_N(\Phi, a)})\\ &= \Lambda(\underline{\Phi})\\ &= \underline{\Phi}\end{aligned}}$$

Constructing Efficient Finite Automata

Example. Given the following NFA.



	T_N	a	b	Λ
S	0	$\varnothing$	$\{0, 1\}$	$\{3\}$
	1	$\{2\}$	$\varnothing$	$\varnothing$
F	2	$\varnothing$	$\{2\}$	$\varnothing$
F	3	$\{3\}$	$\varnothing$	$\varnothing$

$$\Lambda(0) = \{0,3\}$$

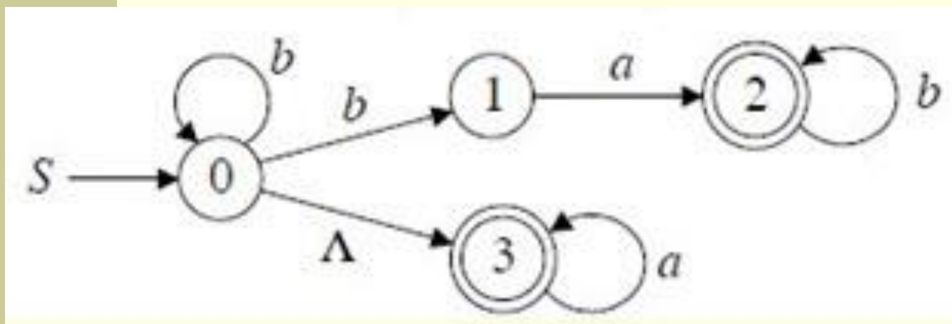
$$\Lambda(1) = \{1\}$$

$$\Lambda(2) = \{2\}$$

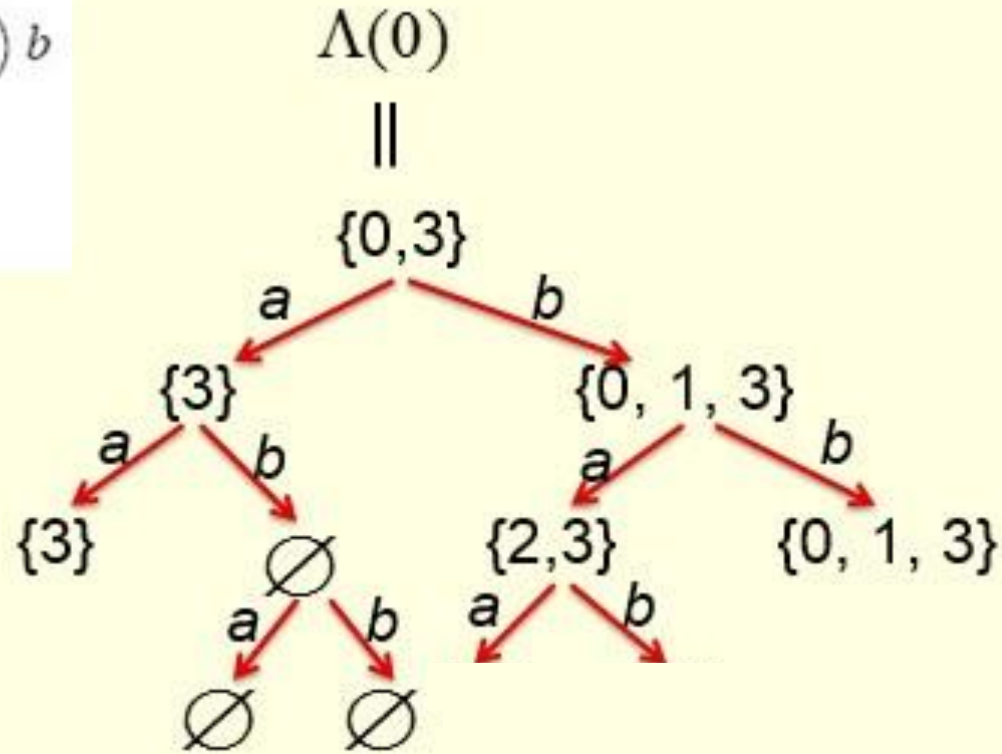
$$\Lambda(3) = \{3\}$$

$$\begin{aligned} T_D(\Phi, b) &= \Lambda(\underline{T_N(\Phi, b)}) \\ &= \Lambda(\underline{\Phi}) \\ &= \underline{\Phi} \end{aligned}$$

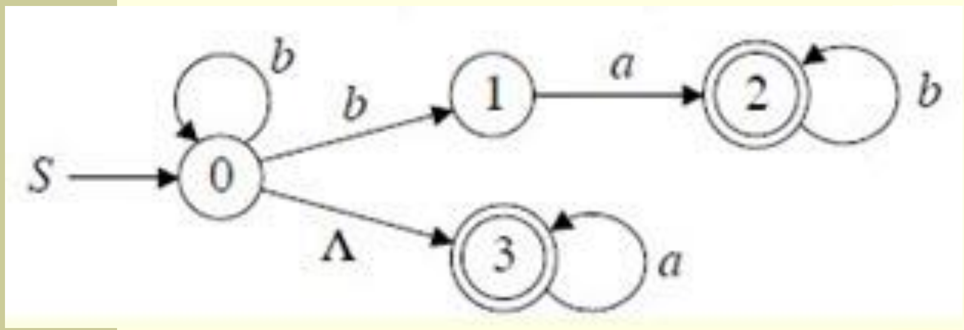
Constructing Efficient Finite Automata

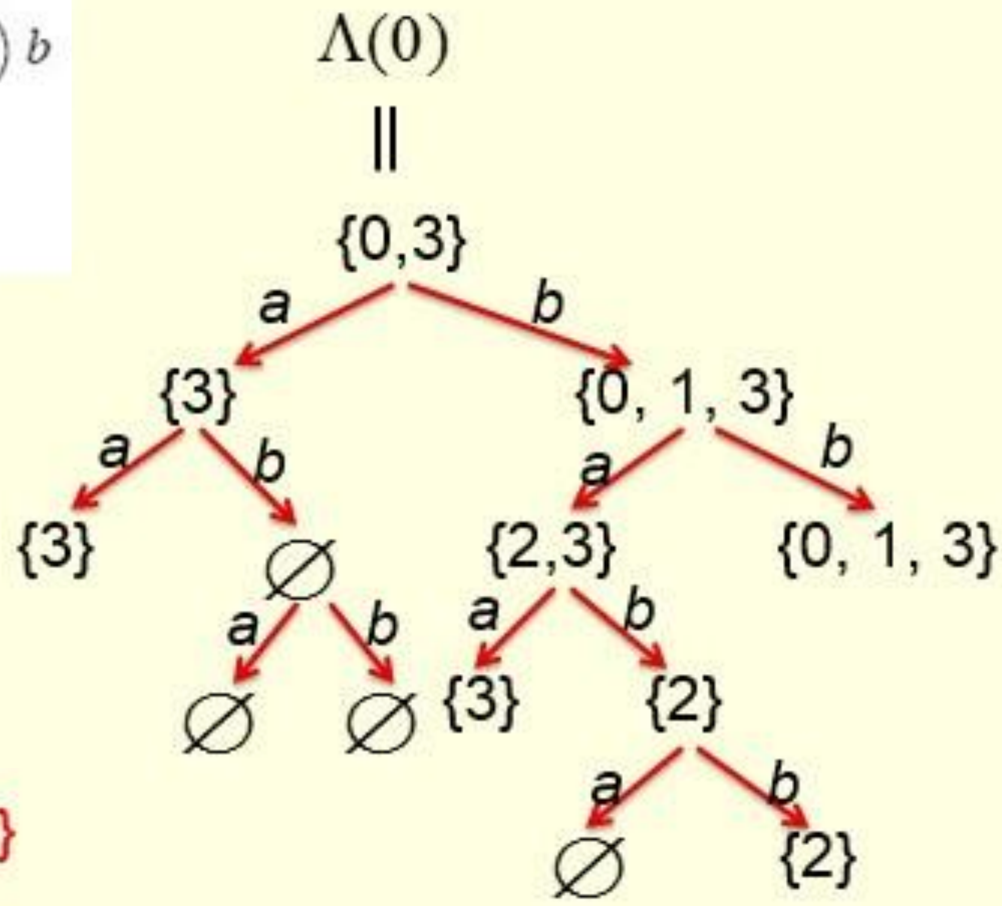


So we have child nodes of Φ now.



Eventually we would get the following tree:





Identify all distinct nodes:

{0,3}, {3}, {0, 1, 3}, ∅, {2,3}, {2}

Constructing Efficient Finite Automata

Construct DFA table T_D from NFA table T_N as follows:

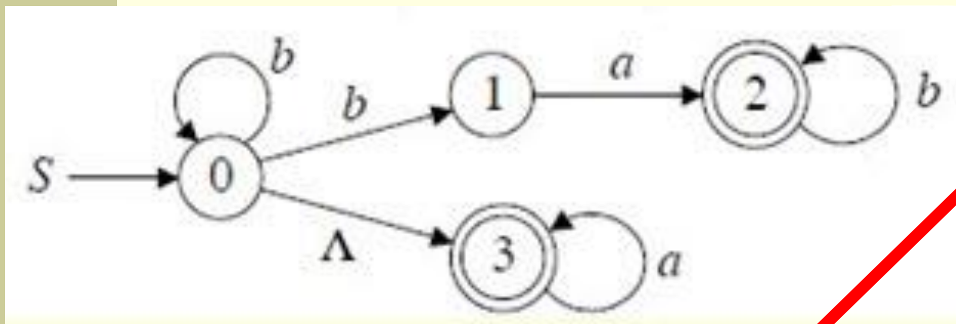
1. The start state of the DFA is $\Lambda(s)$, where s is the start state of the NFA.

2. If $\{s_1, \ldots, s_n\}$ is a DFA state and $a \in A$, then

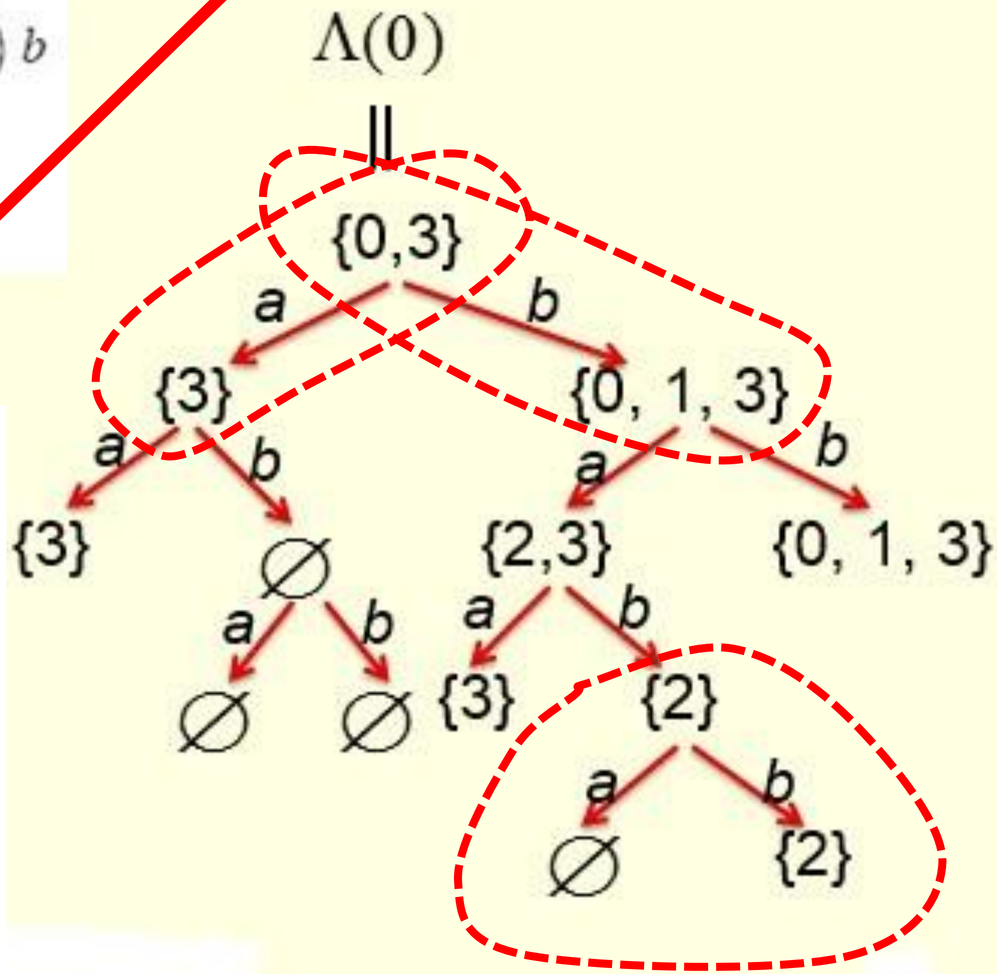
$$T_D(\{s_1, \ldots, s_n\}, a) = \Lambda(T_N(s_1, a) \cup \ldots \cup T_N(s_n, a)).$$

3. A DFA state is a final state if one of its elements is an NFA final state.

the transition table T_D



	T_D	a	b
S, F	$\{0, 3\}$	$\{3\}$	$\{0, 1, 3\}$
F	$\{3\}$	$\{3\}$	$\varnothing$
F	$\{0, 1, 3\}$	$\{2, 3\}$	$\{0, 1, 3\}$
F	$\{2, 3\}$	$\{3\}$	$\{2\}$
F	$\{2\}$	$\varnothing$	$\{2\}$
	$\varnothing$	$\varnothing$	$\varnothing$

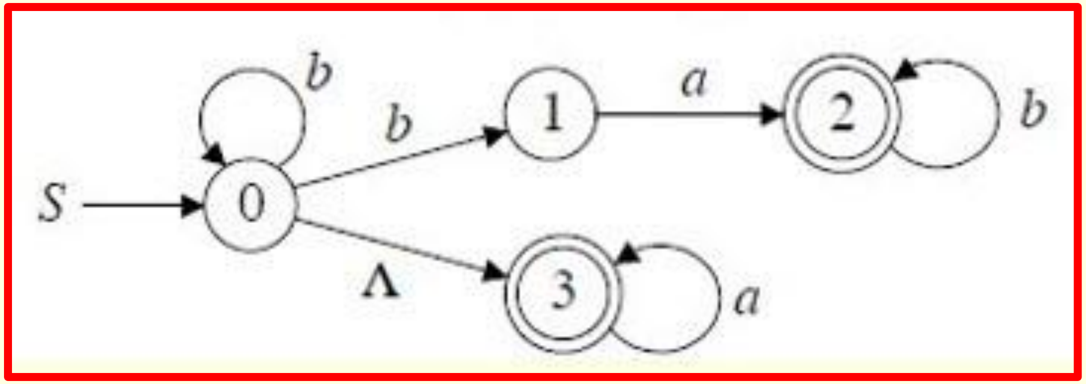


Constructing Efficient Finite Automata

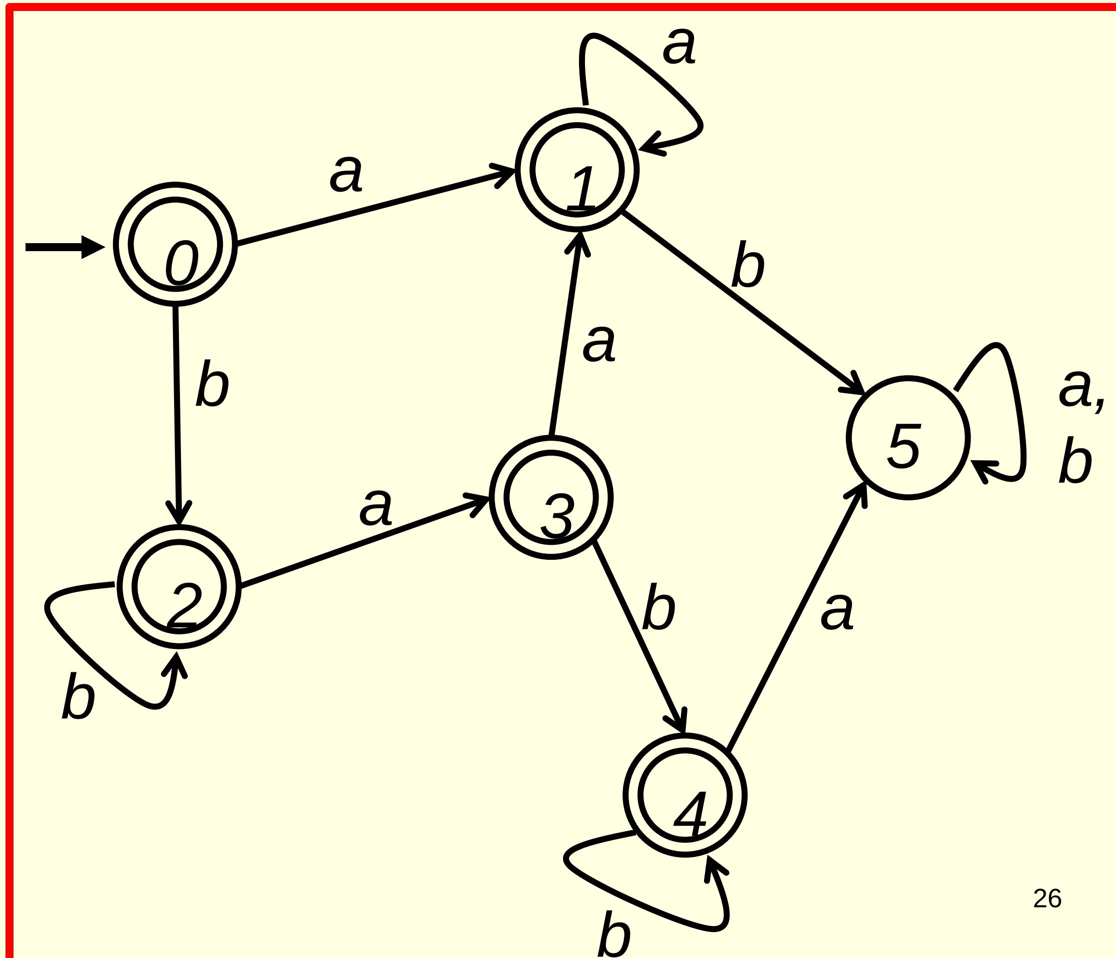
Write in simplified form after renumbering the states:

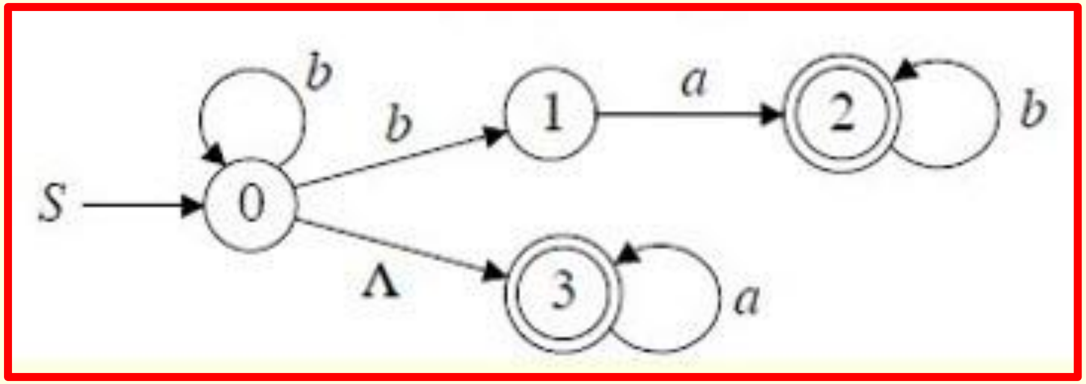
	T_D	a	b
S, F	$\{0, 3\}$	$\{3\}$	$\{0, 1, 3\}$
F	$\{3\}$	$\{3\}$	$\varnothing$
F	$\{0, 1, 3\}$	$\{2, 3\}$	$\{0, 1, 3\}$
F	$\{2, 3\}$	$\{3\}$	$\{2\}$
F	$\{2\}$	$\varnothing$	$\{2\}$
	$\varnothing$	$\varnothing$	$\varnothing$

	T_D	a	b
S, F	0	1	2
F	1	1	5
F	2	3	2
F	3	1	4
F	4	5	4
	5	5	5

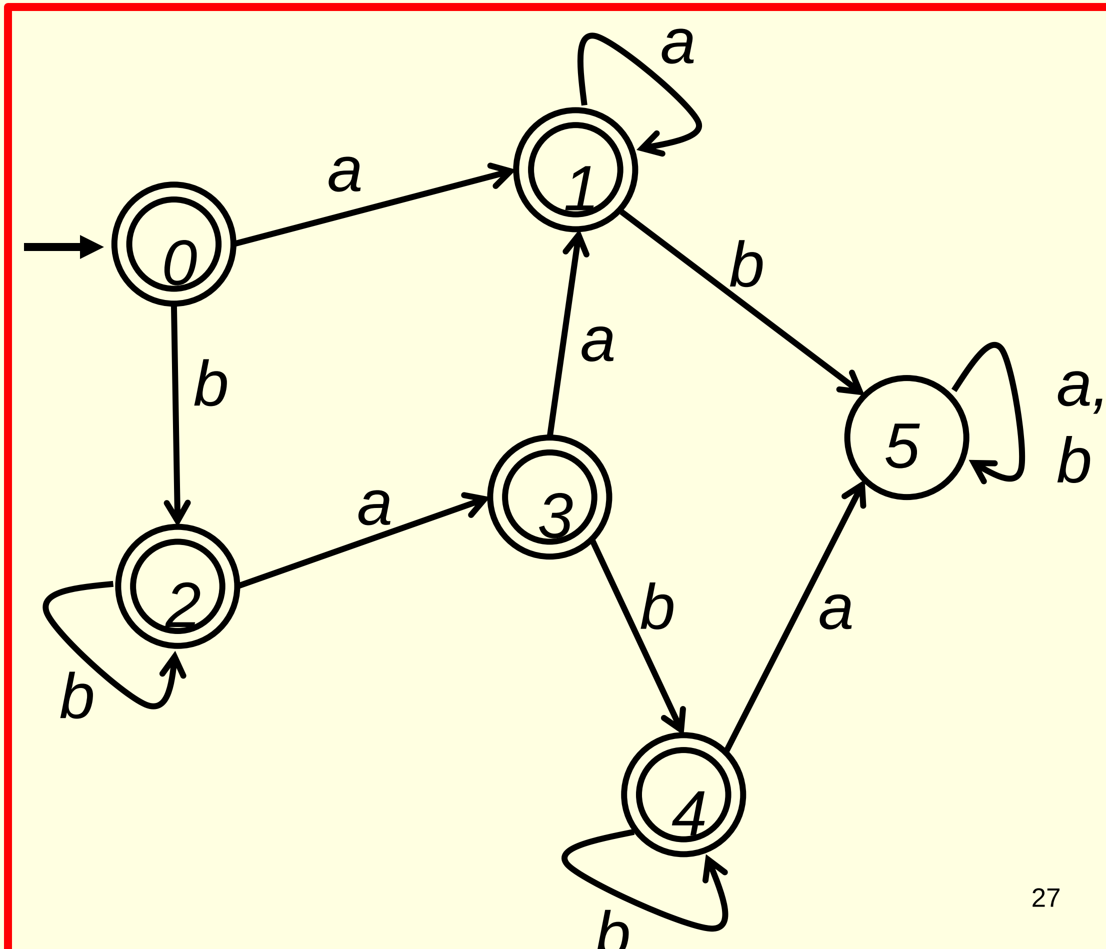


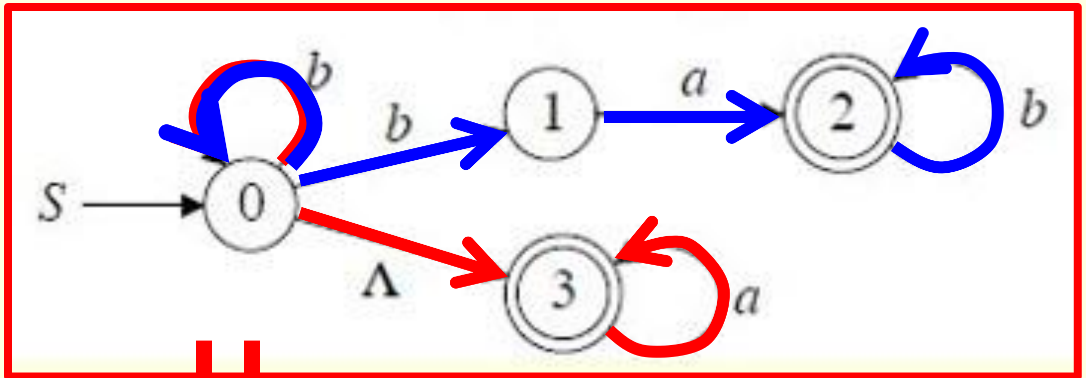
	T_D	a	b
S, F	0	1	2
F	1	1	5
F	2	3	2
F	3	1	4
F	4	5	4
	5	5	5

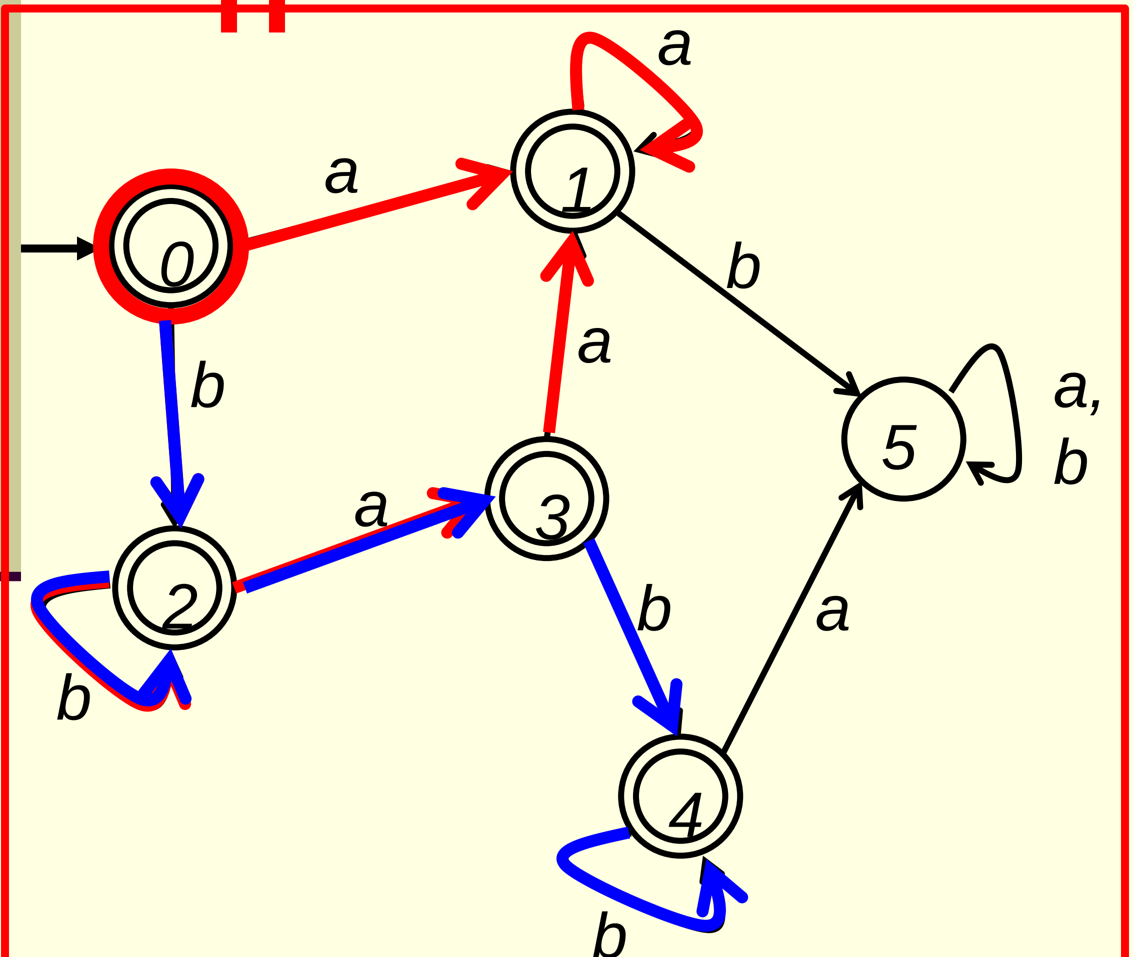




	T_D	a	b
S, F	0	1	2
F	1	1	5
F	2	3	2
F	3	1	4
F	4	5	4
	5	5	5







Language of NFA:

$b^*\Lambda a^* + b^*bab^*$

$= b^*a^* + b^*bab^*$

What is the language of the DFA?

Λ, aa^*, bb^*,
bb^*a, bb^*aa,
bb^*aaa^*,
bb^*ab, bb^*abb^*
$= a^*$, b^*, b^*a^*,
bb^*ab^*

$= b^*a^*$, bb^*ab^*

We need: $b^*a^* + b^*bab^*$

$\Lambda + aa^* = \Lambda + a(\Lambda + a + aa + aaa \ldots)$

$= \Lambda + a + aa + aaa + \ldots$

$= a^*$

$\Lambda + bb^* = b^*$

$bb^*a + bb^*aa + bb^*aaa^*$

$= bb^*a(\Lambda + a + aa^*)$

$= bb^*a(\Lambda + a + a + aa + aaa + \ldots)$

$= bb^*a(\Lambda + a + aa + aaa + \ldots)$

$= bb^*aa^*$

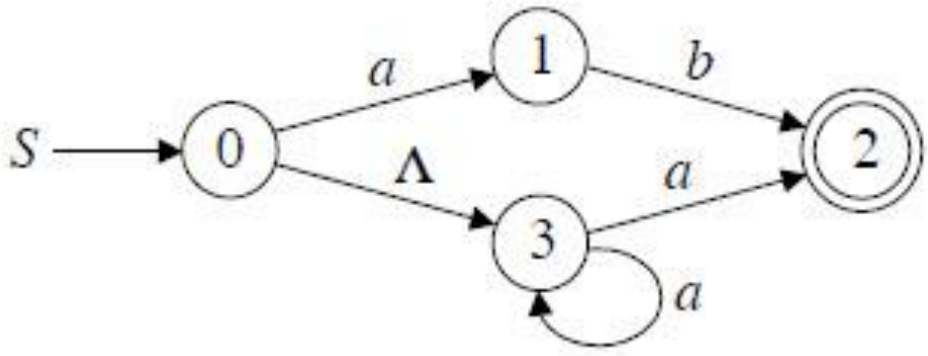
We need: *b*a* + b*bab**

$a^* + b^* + bb^*aa^* = \Lambda + aa^* + bb^*aa^* + b^*$
$= \Lambda + (\Lambda + bb^*)aa^* + b^*$
$= \Lambda + b^*aa^* + b^*$
$= \Lambda + b^*(aa^* + \Lambda)$
$= \Lambda + b^*a^*$

$bb^*a + bb^*ab + bb^*abb^* = bb^*a(\Lambda + b + bb^*)$
$= bb^*a(\Lambda + b + b + bb + bbb + \ldots)$
$= bb^*a(\Lambda + b + bb + bbb + \ldots)$
$= bb^*ab^*$

Constructing Efficient Finite Automata

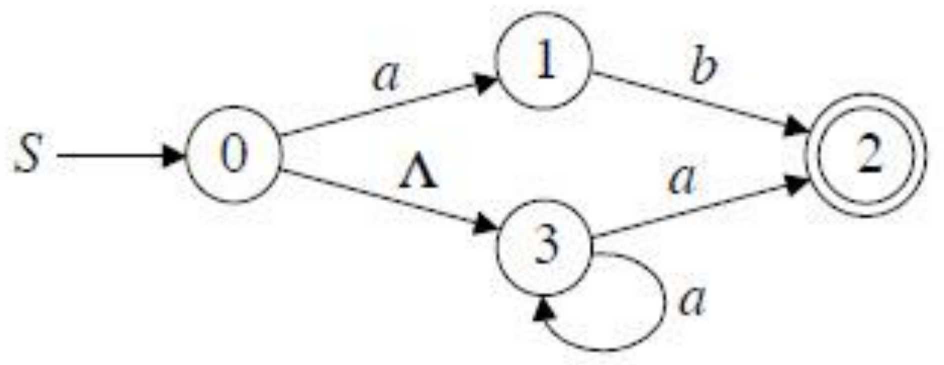
Example 2. Transform the following NFA to a DFA.



	T_N	a	b	Λ
S	0	$\{1\}$	$\varnothing$	$\{3\}$
	1	$\varnothing$	$\{2\}$	$\varnothing$
F	2	$\varnothing$	$\varnothing$	$\varnothing$
	3	$\{2, 3\}$	$\varnothing$	$\varnothing$

Constructing Efficient Finite Automata

Example 2. Transform the following NFA into a DFA.



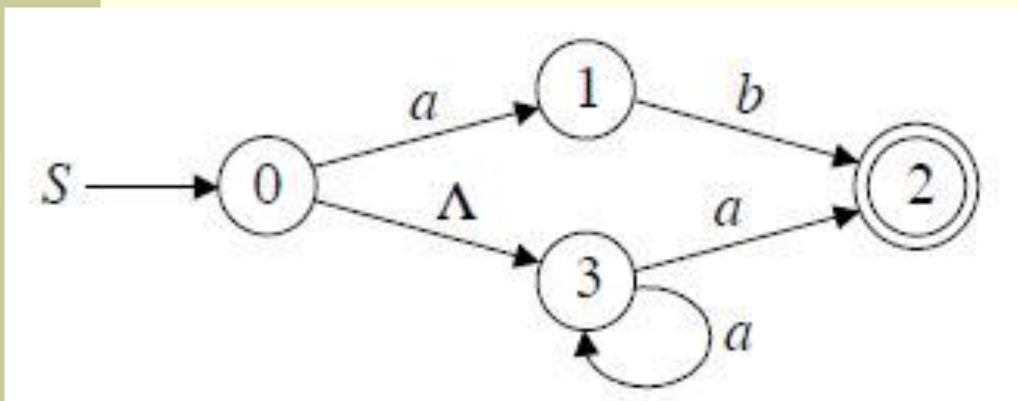
	T_N	a	b	Λ
S	0	$\{1\}$	$\varnothing$	$\{3\}$
	1	$\varnothing$	$\{2\}$	$\varnothing$
F	2	$\varnothing$	$\varnothing$	$\varnothing$
	3	$\{2, 3\}$	$\varnothing$	$\varnothing$

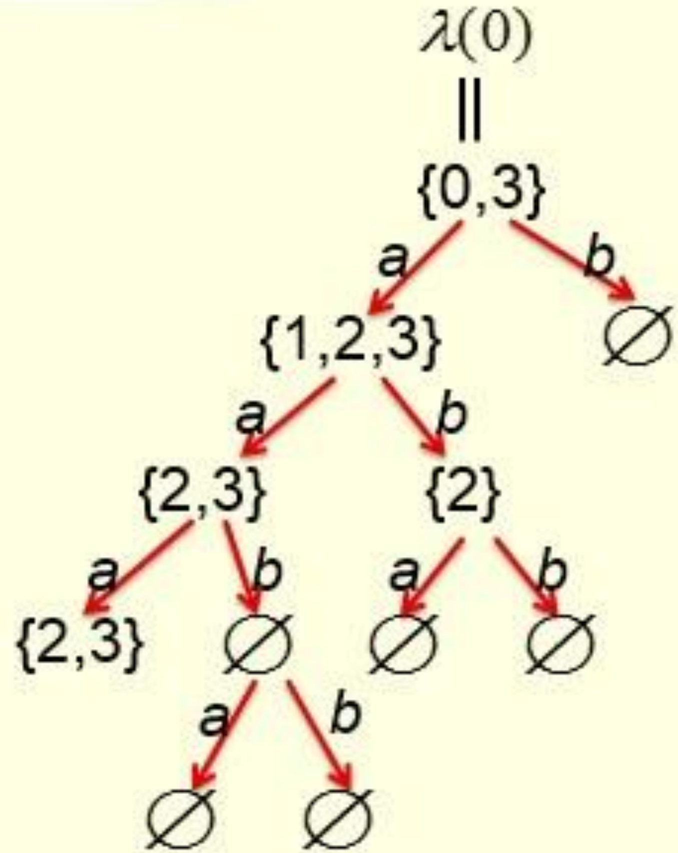
$\boxed{\text{First}}$

$$\boxed{\begin{aligned}\Lambda(0) &= \{0, 3\} \\ \Lambda(1) &= \{1\} \\ \Lambda(2) &= \{2\} \\ \Lambda(3) &= \{3\}\end{aligned}}$$

Constructing Efficient Finite Automata

Example 2. Transform the following NFA into a DFA.



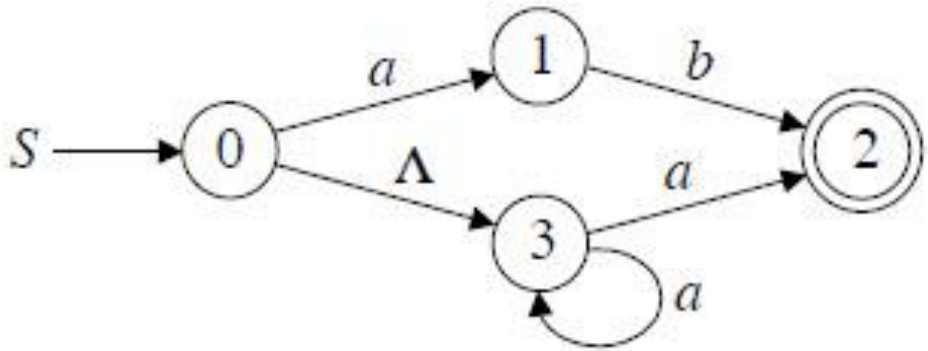


Then, build the tree

Distinct nodes are:
{0, 3}, {1,2,3}, {2,3}, {2}, ∅

Constructing Efficient Finite Automata

Example 2. Transform the following NFA into a DFA.



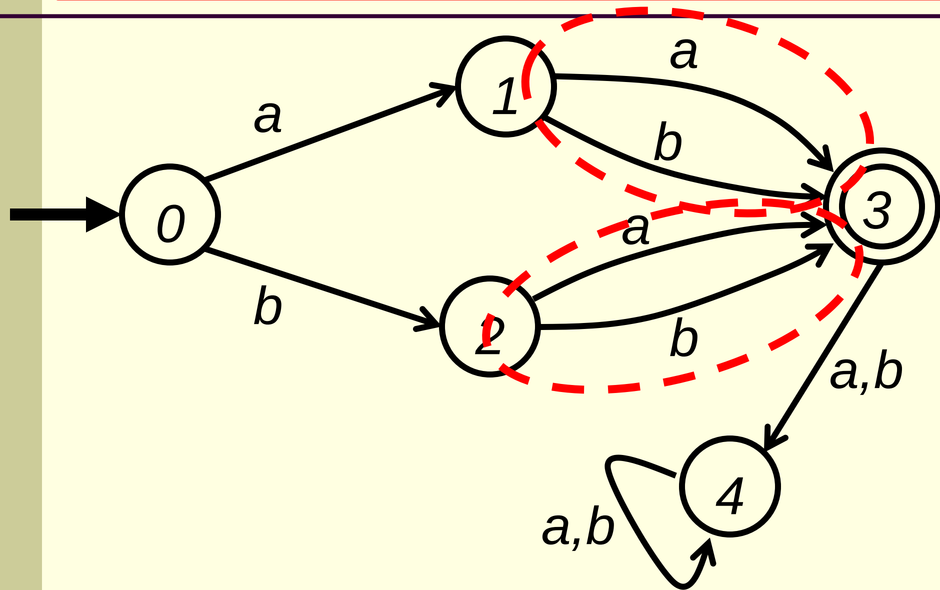
	T_N	a	b	Λ
S	0	$\{1\}$	$\varnothing$	$\{3\}$
	1	$\varnothing$	$\{2\}$	$\varnothing$
F	2	$\varnothing$	$\varnothing$	$\varnothing$
	3	$\{2, 3\}$	$\varnothing$	$\varnothing$

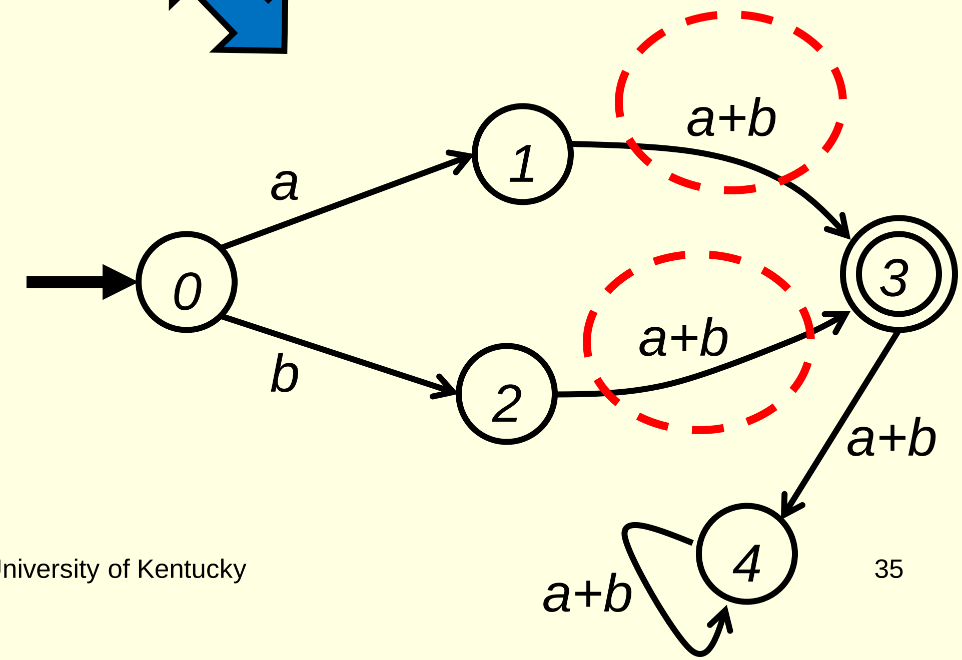
Hence, solution:

	T_D	a	b
S	$\{0, 3\}$	$\{1, 2, 3\}$	$\varnothing$
F	$\{1, 2, 3\}$	$\{2, 3\}$	$\{2\}$
F	$\{2, 3\}$	$\{2, 3\}$	$\varnothing$
F	$\{2\}$	$\varnothing$	$\varnothing$
	$\varnothing$	$\varnothing$	$\varnothing$

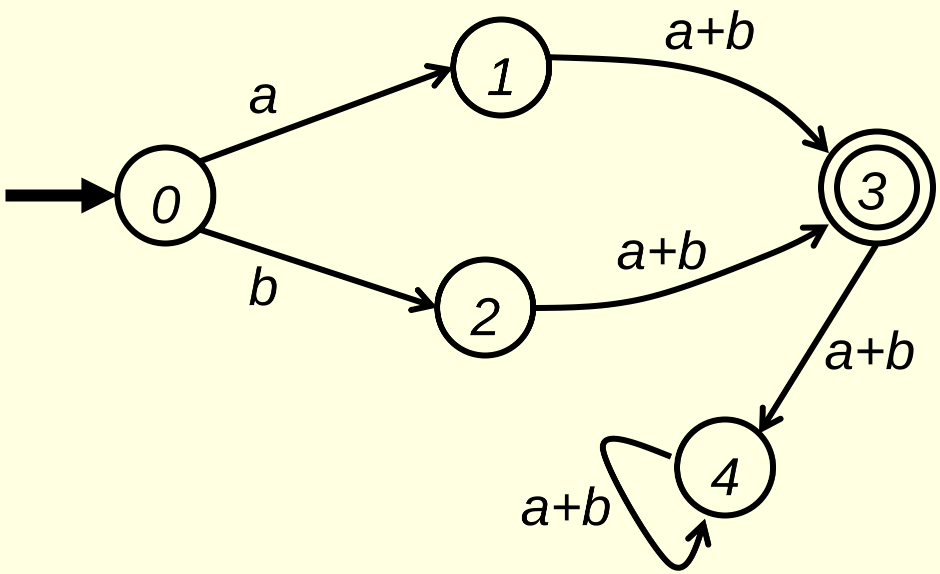
	T_D	a	b
S	0	1	4
F	1	2	3
F	2	2	4
F	3	4	4
	4	4	4

When can two states of a DFA be merged? (case 1)





When can two states of a DFA be merged? (case 1)

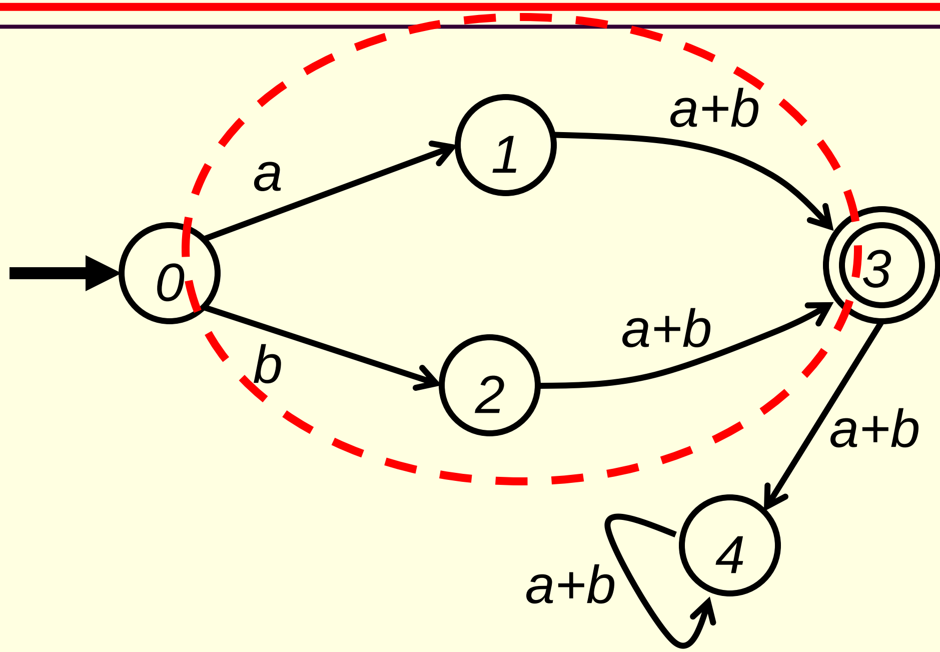


Expression of this DFA:

a(a+b) + b(a+b)

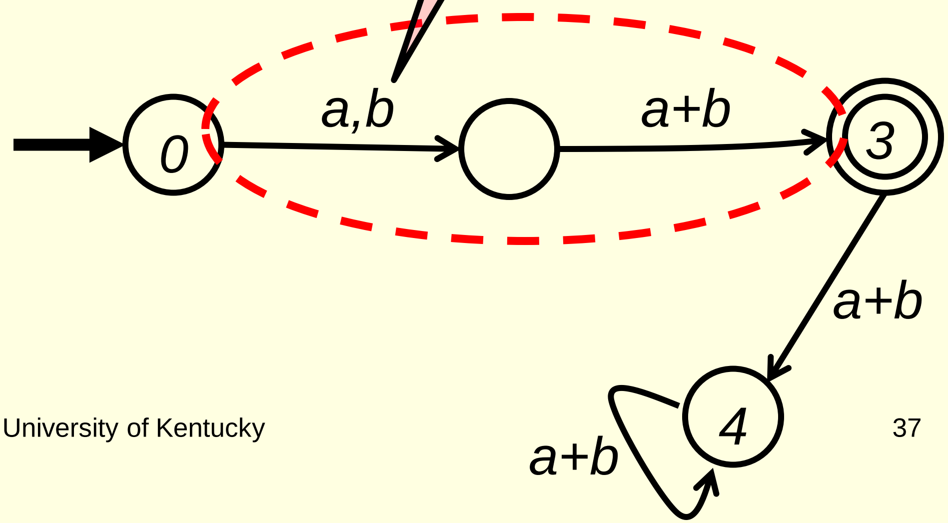
= (a+b)(a+b)

When can two states of a DFA be merged? (case 1)

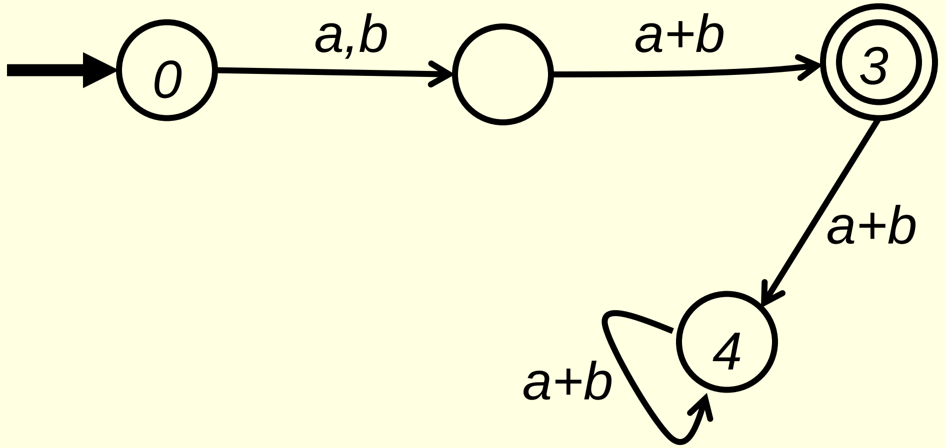


Same as a+b

Merging state 1 and state 2, we get



When can two states of a DFA be merged? (case 1)

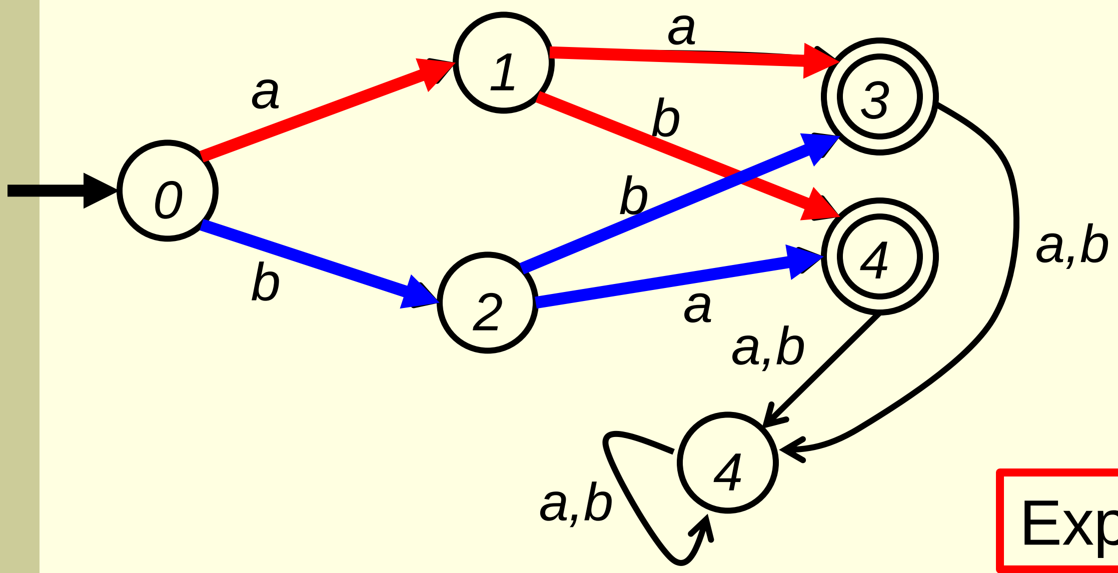


Expression of this DFA:

(a+b)(a+b)

Hence, two DFA states can be merged into a single state if for each letter of the alphabet, the corresponding edges of these two states point at the same state.

When can two states of a DFA be merged? (case 2)



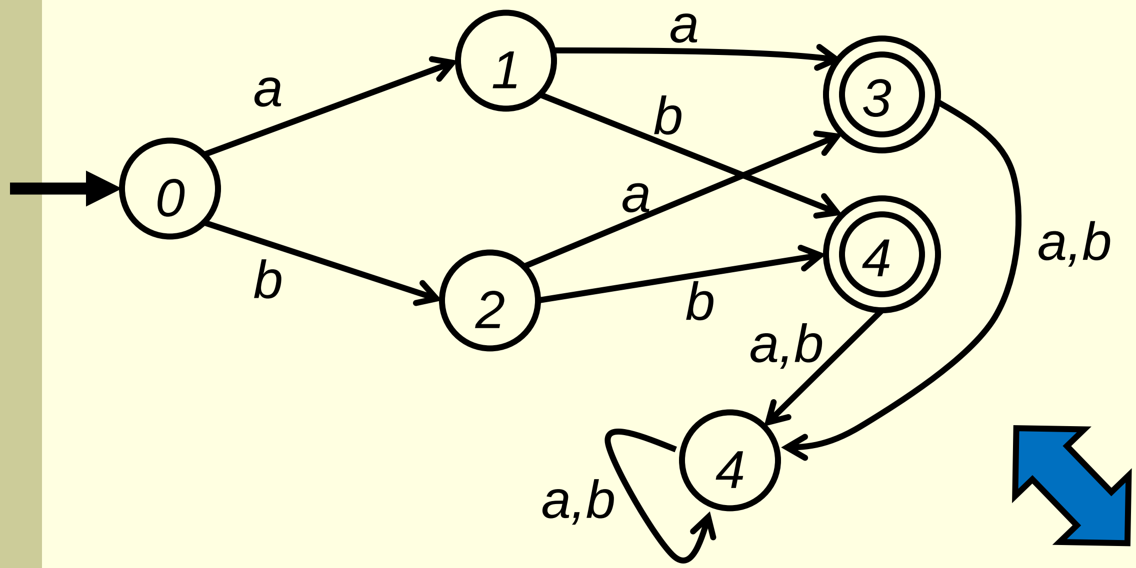
Expression of this DFA:

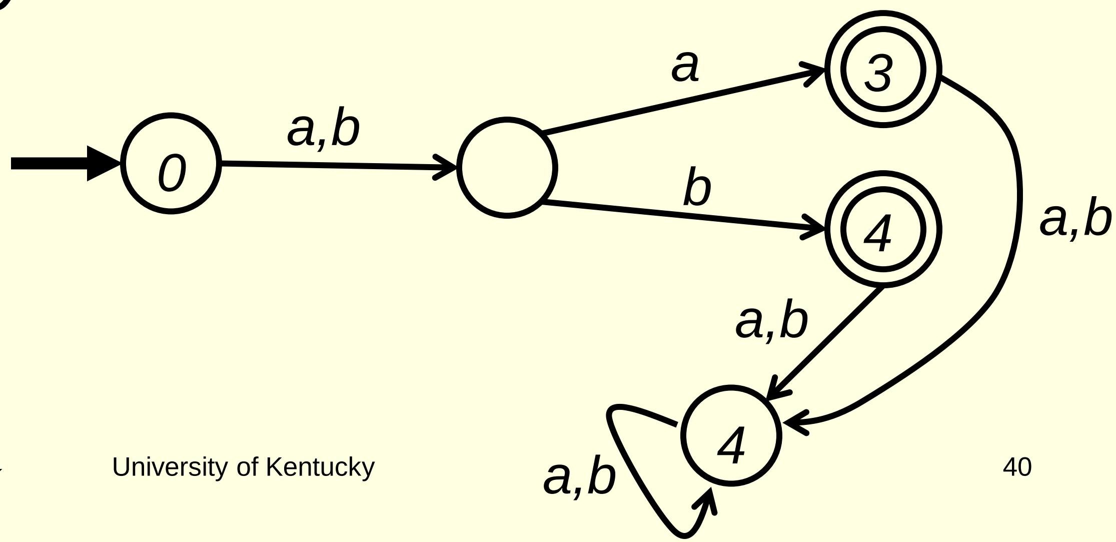
aa, ab, ba, bb

= a(a+b) + b(a+b)

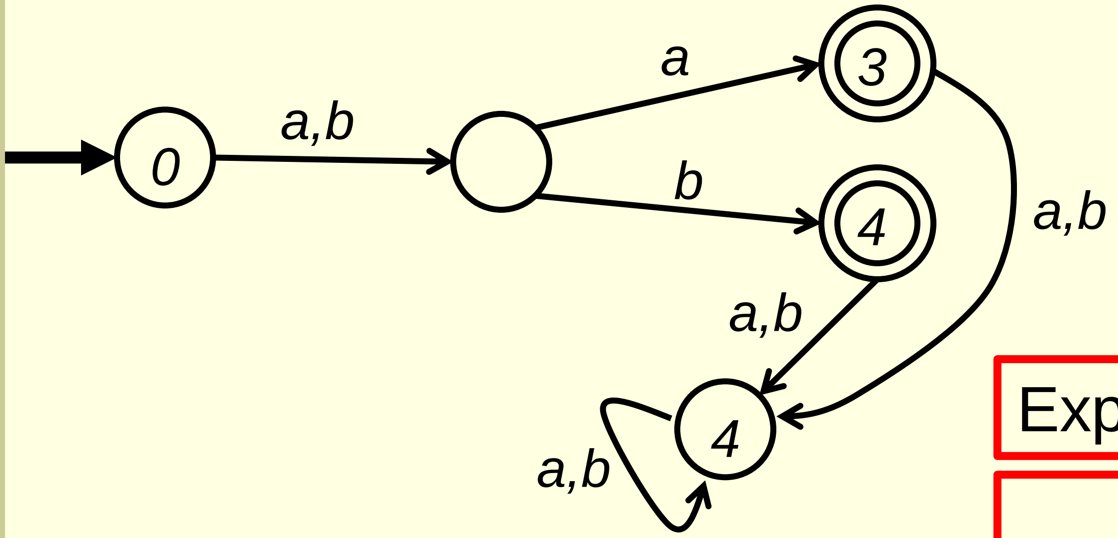
= (a+b)(a+b)

When can two states of a DFA be merged? (case 2)





When can two states of a DFA be merged? (case 2)



Expression of this DFA:

(a+b)a + (a+b)b

=(a+b)(a+b)

Hence, two DFA states can be merged into a single state if for each letter of the alphabet, the corresponding edges of these two states point at states in the same group.

Now, transform a DFA to a minimum-state DFA

Algorithm:

1. Construct the following sequence of sets of possible equivalent pairs of distinct states:

$$E_0 \supset E_1 \supset \cdots \supset E_k = E_{k+1}$$

where

$E_0 = \{(s, t) \mid s \text{ and } t \text{ are either both final or both non-final}\}$

and

$E_{i+1} = \{(s,t) \in E_i \mid (T(s, a), T(t, a)) \in E_i$ or $T(s, a) = T(t, a)$ for every $a \in A\}$

E_k represents the distinct pairs of equivalent states from which an equivalence relation ~ can be generated.

Note:

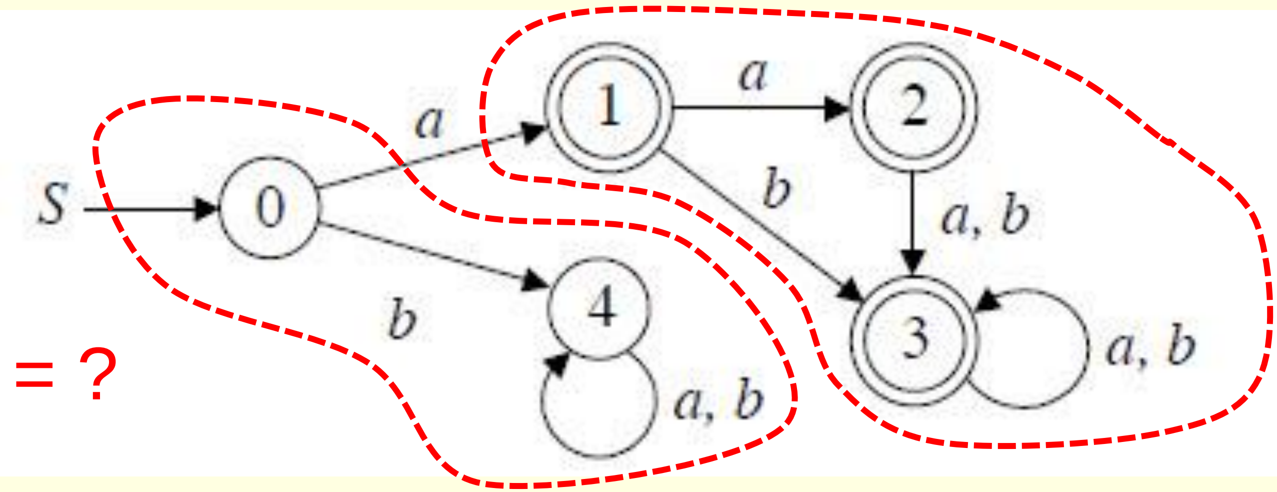
Each E_i *can be represented in 2 ways:*

1. *listed as a partition of the states of the given DFA*

2. *or listed as an equivalence relation on the states of the given DFA yielded by the above partition*

(Read slide 72 of the Notes “Preliminaries”)

Constructing Efficient Finite Automata



	T	a	b
S	0	1	4
F	1	2	3
F	2	3	3
F	3	3	3
	4	4	4

E_0 = ?

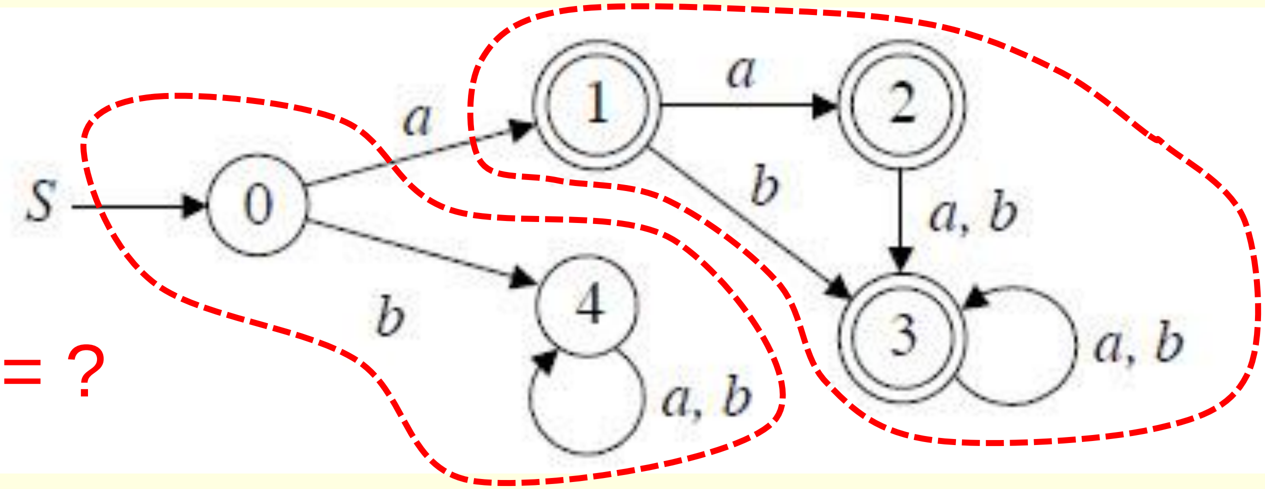
$E_0 = \{(s, t) \mid s \;\&\; t$ are either both final or both non-final$\}$

Theoretically

E_0 = { (0,4), (1,2), (1,3), (2,3) }

E_0 = { (0,0), (4,4), (0,4), (4,0), (1,1), (2,2), (3, 3), (1,2), (2,1), (1,3), (3,1), (2,3), (3,2) }

Constructing Efficient Finite Automata



	T	a	b
S	0	1	4
F	1	2	3
F	2	3	3
F	3	3	3
	4	4	4

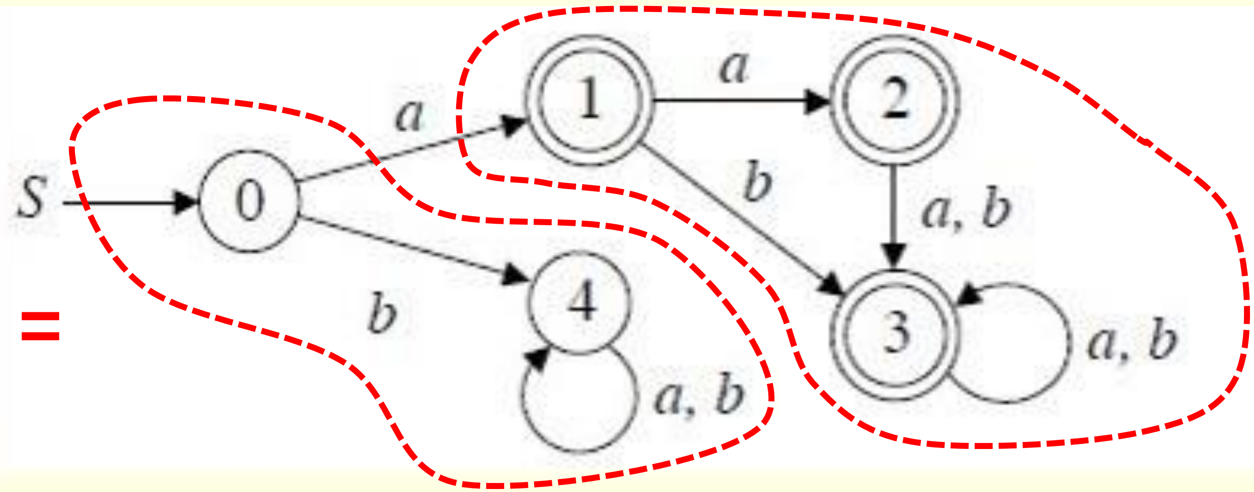
E_0 = ?

or

E_0 = { {0, 4} , {1, 2, 3} } , a partition of the states

Constructing Efficient Finite Automata

E_0 =



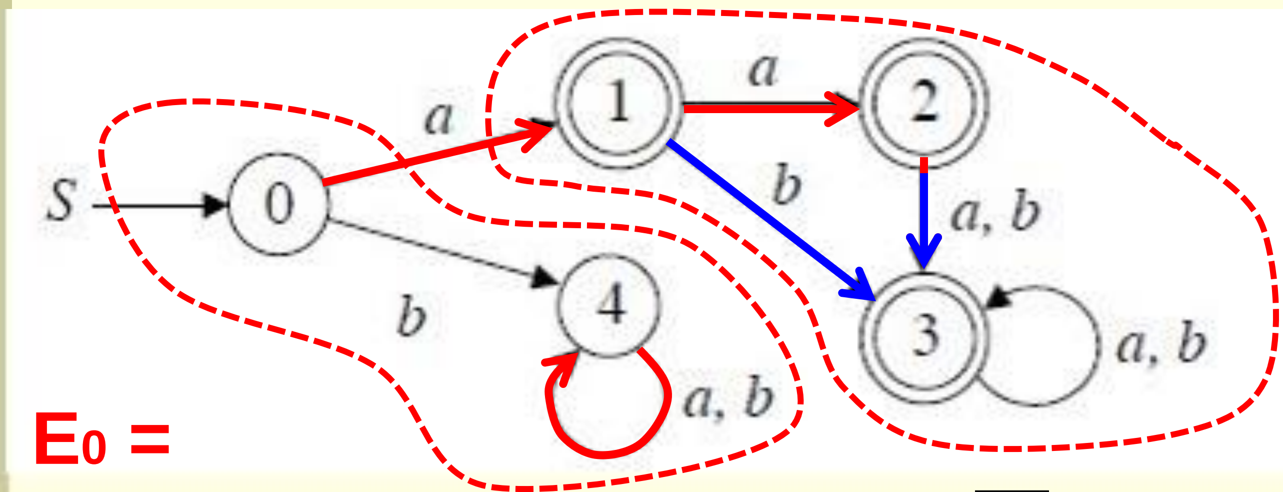
E_1 = ?

$E_{i+1} = \{(s, t) \in E_i \mid \underline{(T(s. a), T(t, a)) \in E_i}$ or $\underline{T(s, a) = T(t, a)}$ for every $a \in A\}$

To be a pair in E_{i+1}, s and t must be mapped to the same state or states in the same group in E_i by every a ϵ A.

Constructing Efficient Finite Automata

E_0 =



$1 \xrightarrow{a} 2$

$2 \xrightarrow{a} 3$

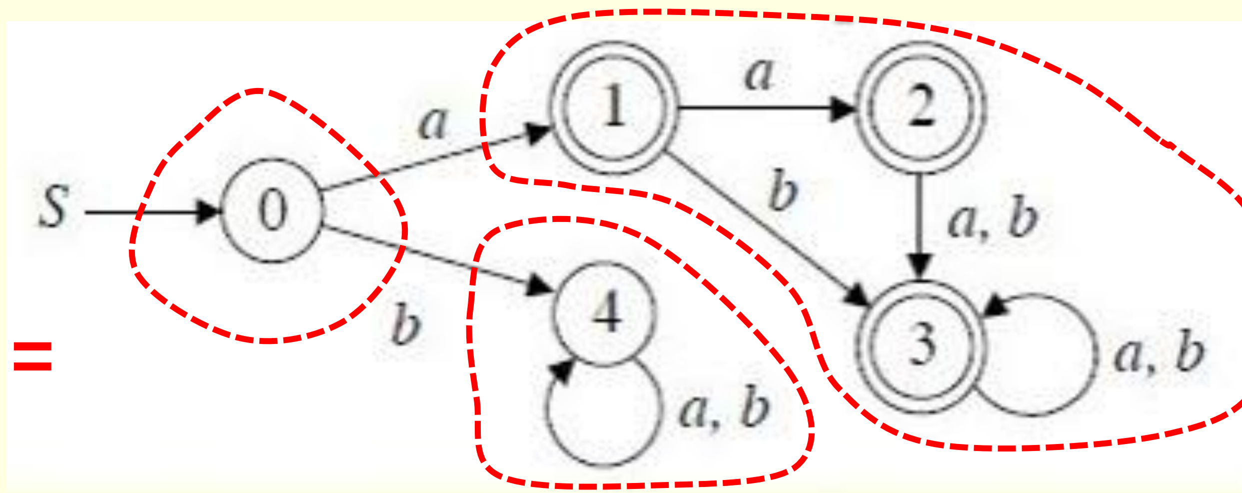
$1 \xrightarrow{b} 3$

$2 \xrightarrow{b} 3$

$0 \xrightarrow{a} 1$

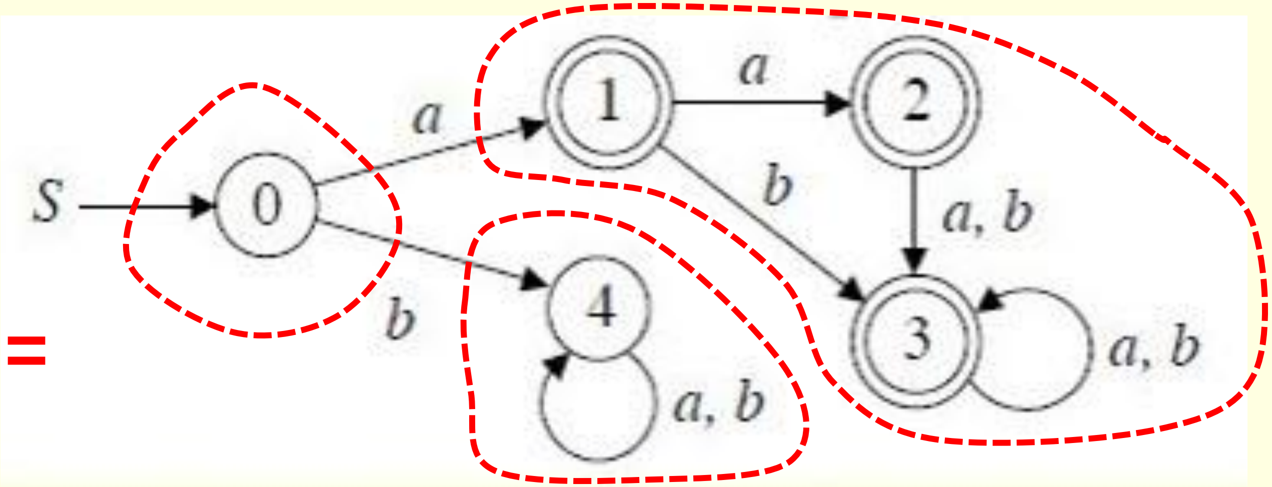
$4 \xrightarrow{a} 4$

E_1 =



Constructing Efficient Finite Automata

E_1 =



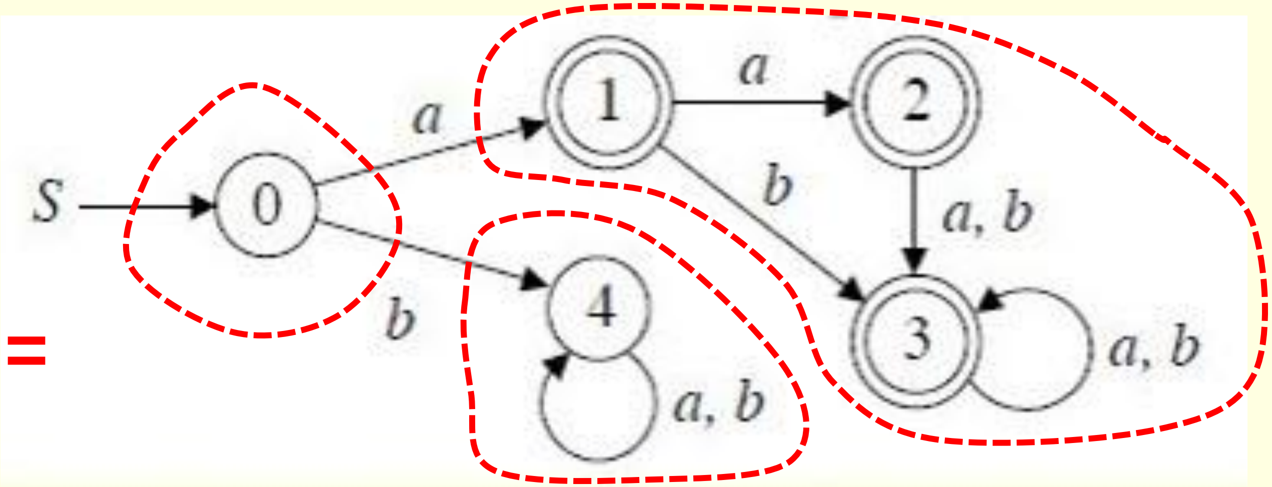
Theoretically,

E_1 = { (0,0), (4,4), (1,1), (2,2), (3, 3), (1,2), (2,1), (1,3), (3,1), (2,3), (3,2) }

or simply, E_1 = { (1,2}, (1,3), (2,3) }

Constructing Efficient Finite Automata

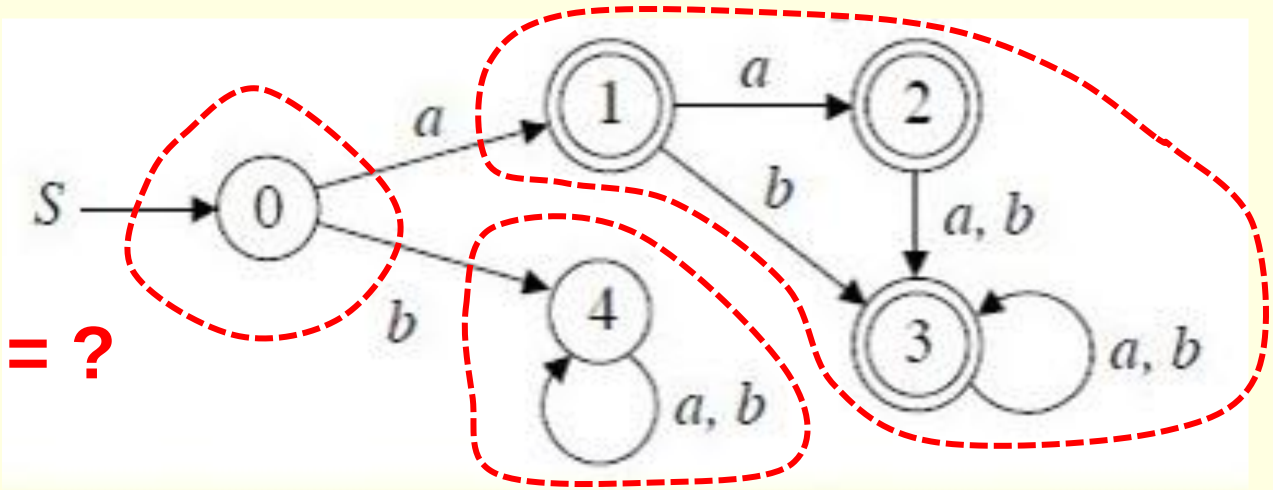
E_1 =



Or list E_1 as a partition of the states

$E_1 = \{ \{ 0 \}, \{ 4 \}, \{1, 2, 3\} \}$

Constructing Efficient Finite Automata



E_2 = ?

$$1 \xrightarrow{a} 2$$
$$2 \xrightarrow{a} 3$$
$$3 \xrightarrow{a} 3$$

One-element groups can not be further reduced and the three-element group will remain the same

Hence, $E_2 = E_1$

S is partitioned by {0}, {1, 2, 3}, {4}.

The minimum-state DFA has 3 states: [0], [4], [1]

Constructing Efficient Finite Automata

Algorithm: *Transform a DFA to a minimum-state DFA*

2. The equivalence classes form the states of the minimum state DFA with transition table T_{min} defined by

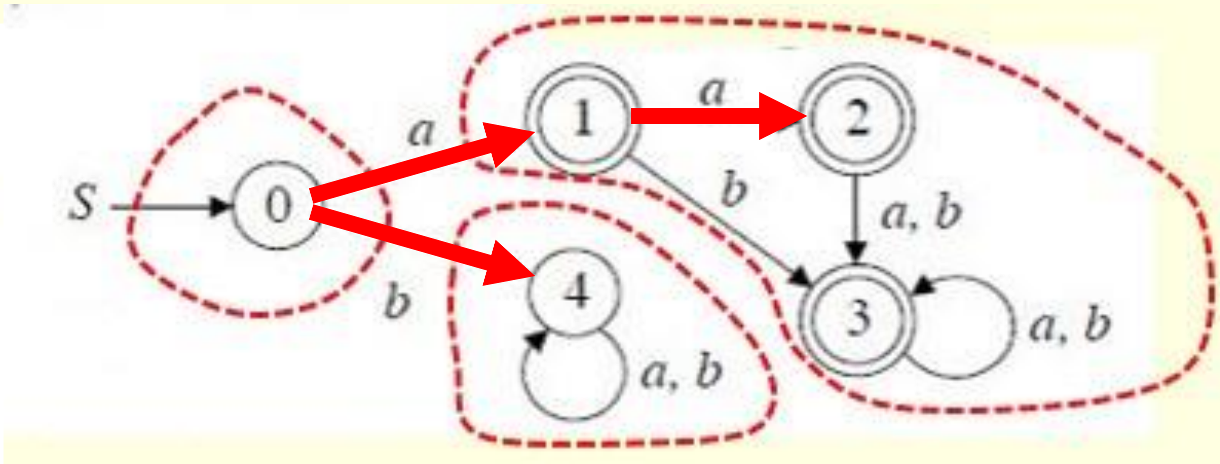
$$T_{min}([s], a) = [T(s, a)].$$

3. The start state is the class containing the start state of the given DFA.

4. A final state is any class containing a final state of the given DFA.

Constructing Efficient Finite Automata

Example. Transform the given DFA into a minimum-state DFA.



	T	a	b
S	0	1	4
F	1	2	3
F	2	3	3
F	3	3	3
	4	4	4

Min-state Table

	T_{Min}	a	b
S	[0]	[1]	[4]
F	[1]	[1]	[1]
	[4]	[4]	[4]

$T_{min}([s], a) = [T(s, a)]$

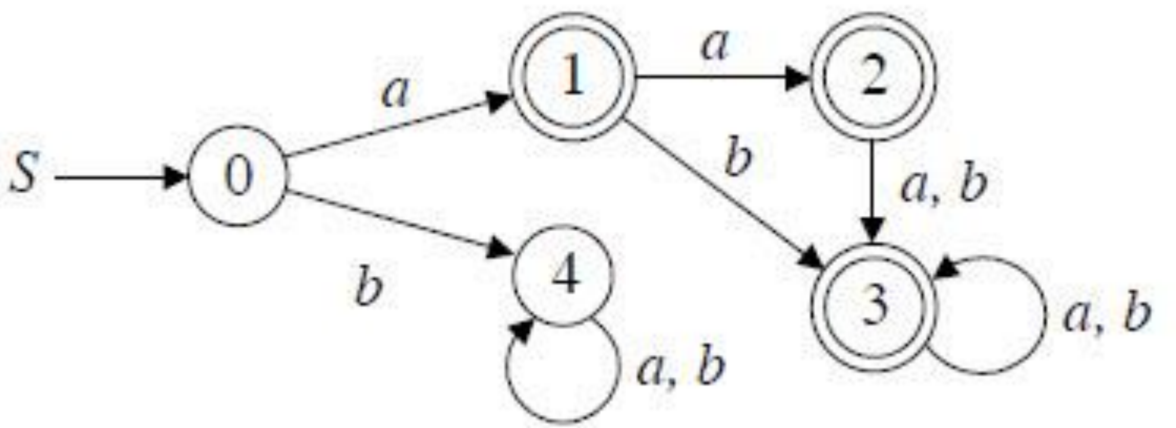
$T_{min}([0], a) = [T(0, a)] = [1]$

$T_{min}([0], b) = [T(0, b)] = [4]$

$T_{min}([1], a) = [T(1, a)] = [2] = [1]$

Constructing Efficient Finite Automata

Example. Transform the given DFA to a minimum-state DFA.



	T	a	b
S	0	1	4
F	1	2	3
F	2	3	3
F	3	3	3
	4	4	4

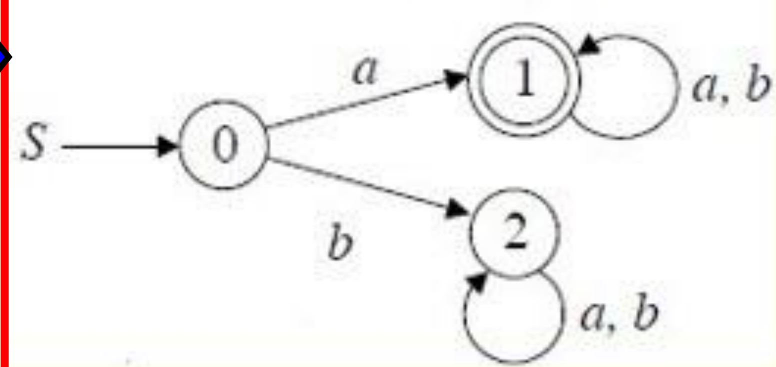
Min-state Table

	T_{Min}	a	b
S	[0]	[1]	[4]
F	[1]	[1]	[1]
	[4]	[4]	[4]

Renamed Table

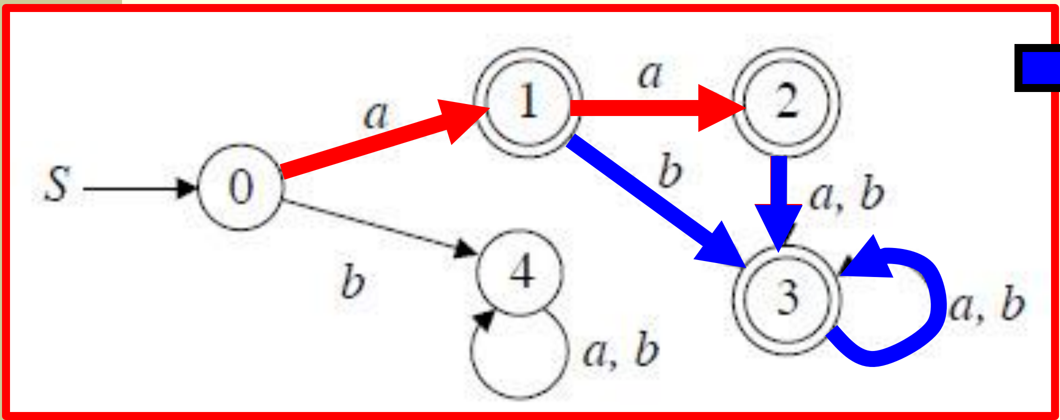
	T_{Min}	a	b
S	0	1	2
F	1	1	1
	2	2	2

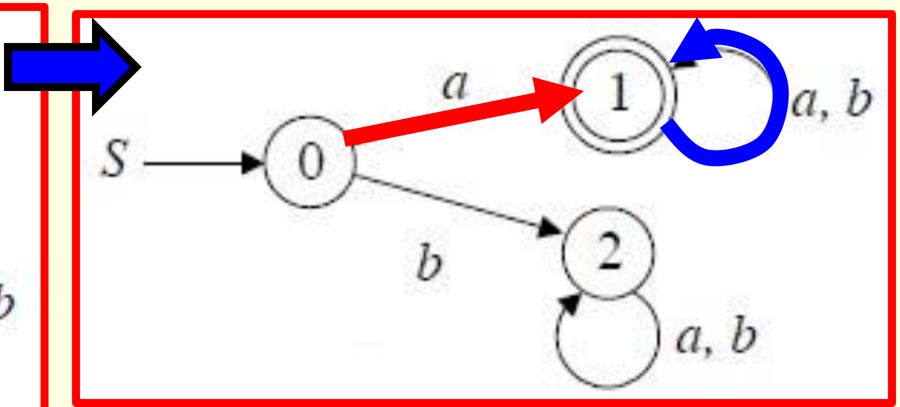
Min-state DFA graph



Constructing Efficient Finite Automata

Example. Transform the given DFA to a minimum-state DFA.





Question: What regular expression equality arises from the two DFAs?

Answer: *a + aa + (aaa + aab + ab)(a + b)* = a(a + b)*.*

Constructing Efficient Finite Automata

Prove : $a + aa + (aaa + aab + ab)(a + b)^* = a(a + b)^*$

$LHS = a + aa + (aaa + aab)(a + b)^* + ab(a + b)^*$

$= a + aa + aa(a + b)(a + b)^* + ab(a + b)^*$

$= a + aa + aa(\mathbf{a} + \mathbf{b})^+ + ab(a + b)^*$

$= a + aa(\Lambda + (\mathbf{a} + \mathbf{b})^+) + ab(a + b)^*$

$= a + aa(a + b)^* + ab(a + b)^*$

$= a + a(a + b)(a + b)^*$

$= a + a(\mathbf{a} + \mathbf{b})^+$

$= a(\Lambda + (\boldsymbol{a} + \boldsymbol{b})^+)$

$= a\,(a + b)^*$

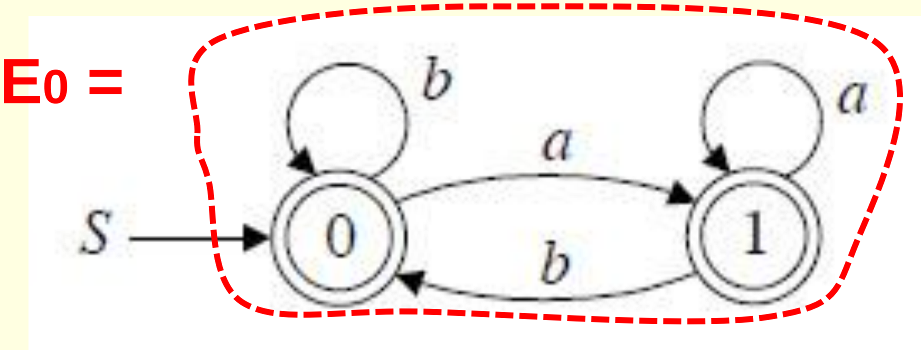
$= RHS$

$\Lambda = (a + b)^0$

Constructing Efficient Finite Automata

Question: Is the following DFA a minimum-state DFA?

E_0 =



Answer. *No.*

$E_0 = \{(0, 1)\}$,

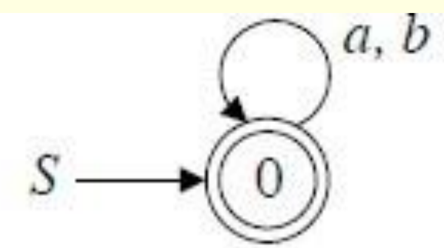
$E_1 = \{(0, 1)\} = E_0$.

The partition is the whole set of states {0, 1} = [0].

Therefore,

	T_{Min}	a	b
S, F	[0]	[0]	[0]

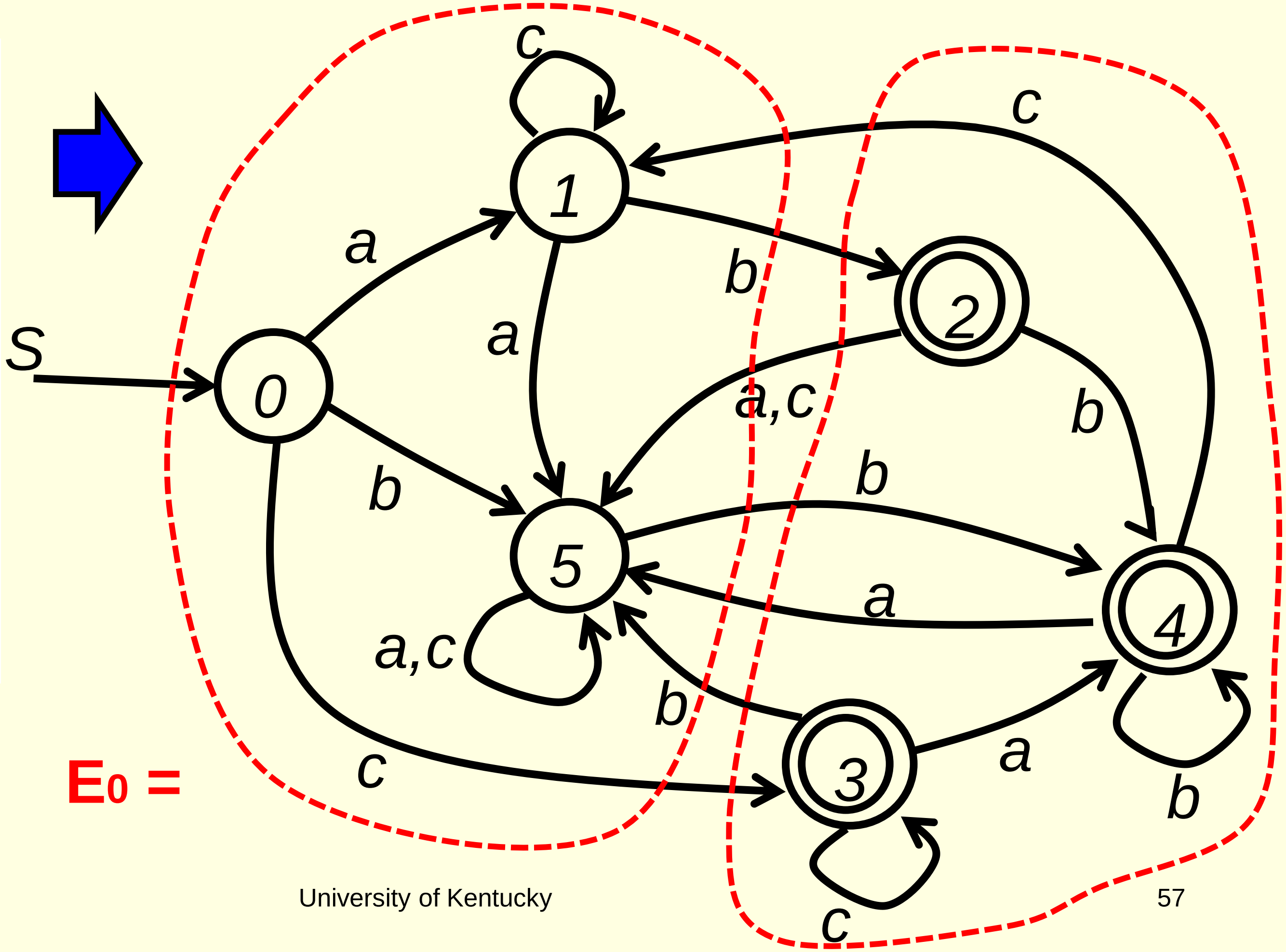
	T_{Min}	a	b
S, F	0	0	0



Constructing Efficient Finite Automata

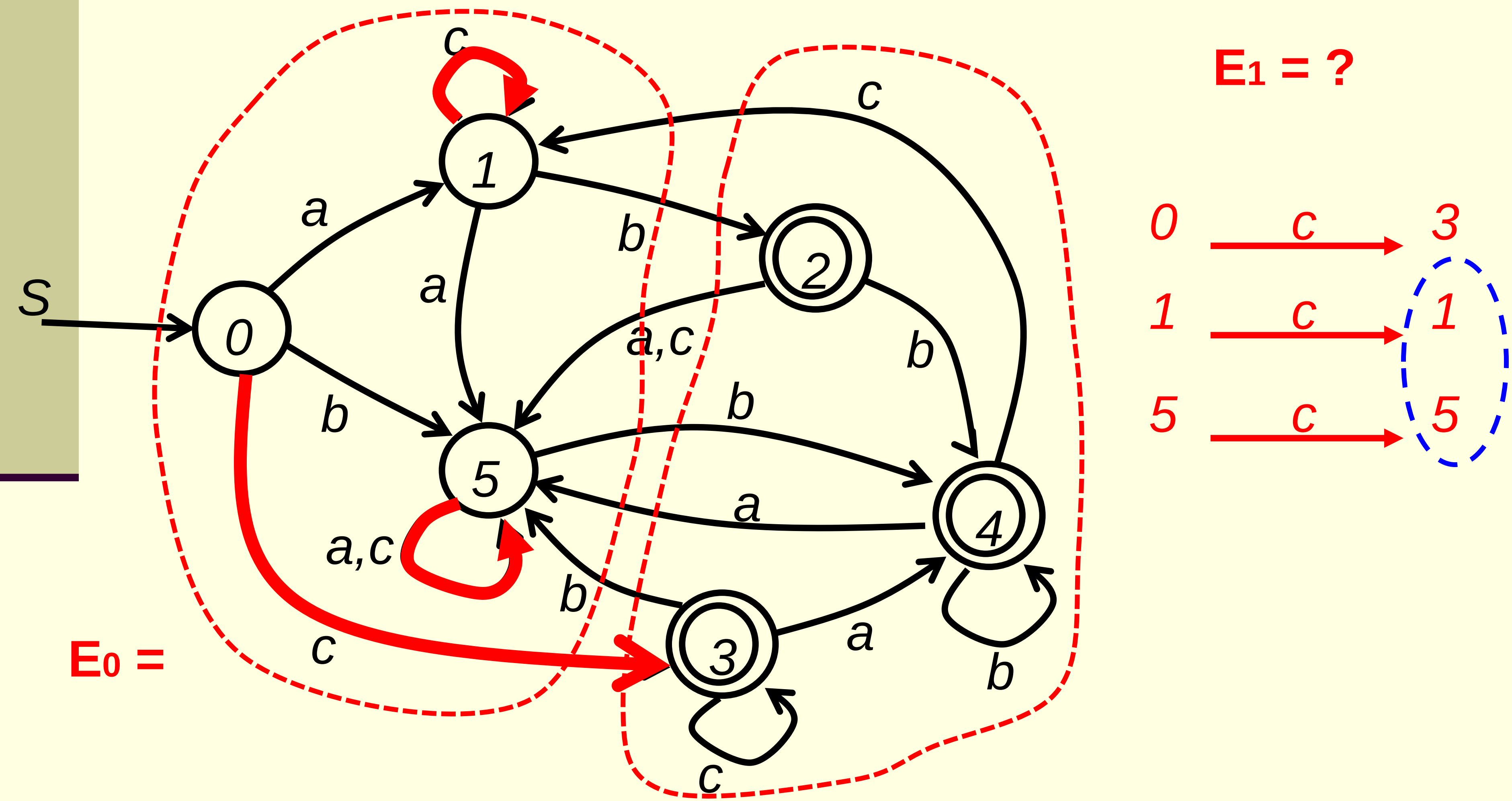
Question: Is the following DFA a minimum-state DFA?

	T	a	b	c
S	0	1	5	3
	1	5	2	1
F	2	5	4	5
F	3	4	5	3
F	4	5	4	1
	5	5	4	5



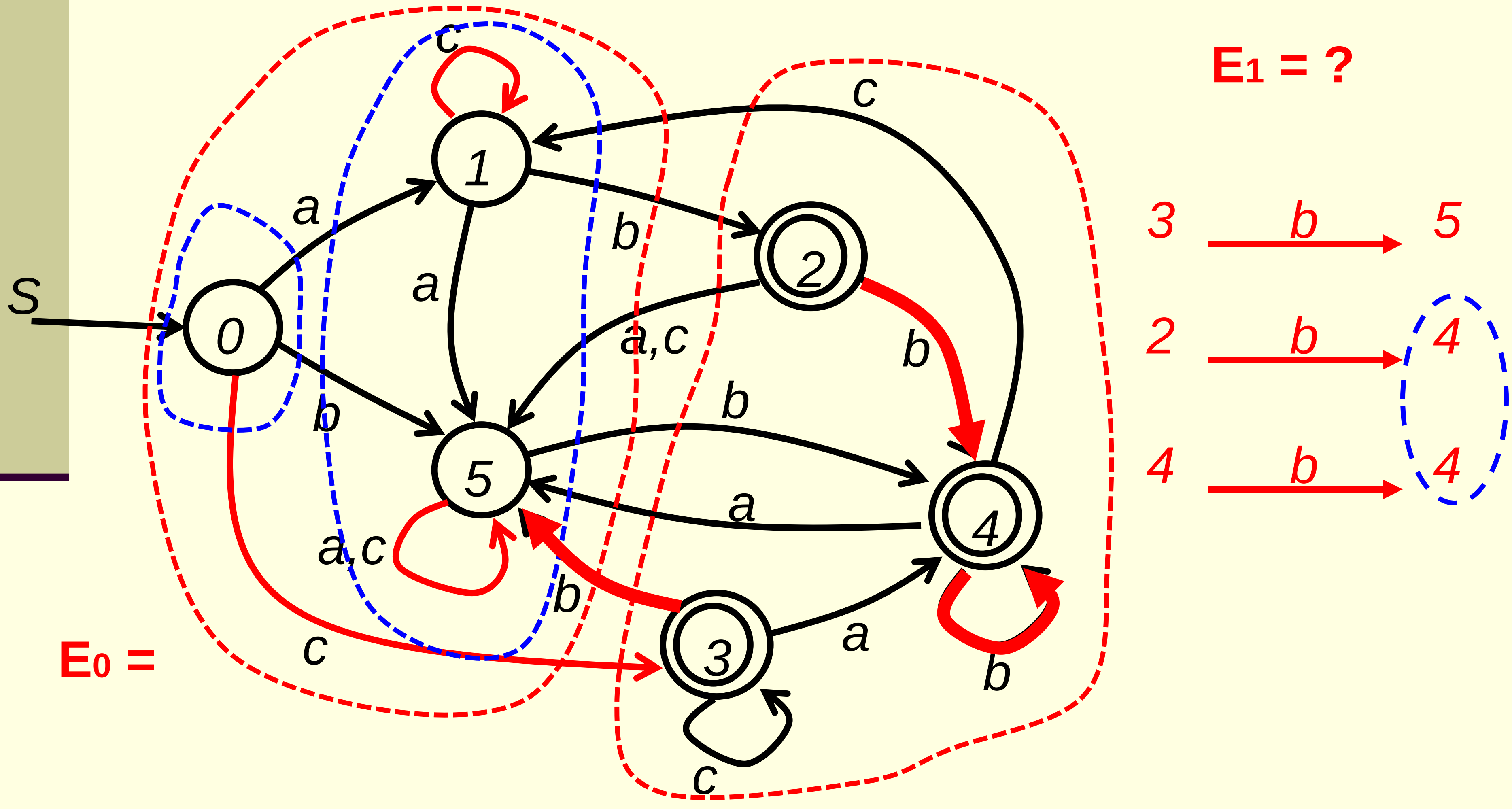
Constructing Efficient Finite Automata

Question: Is the following DFA a minimum-state DFA?



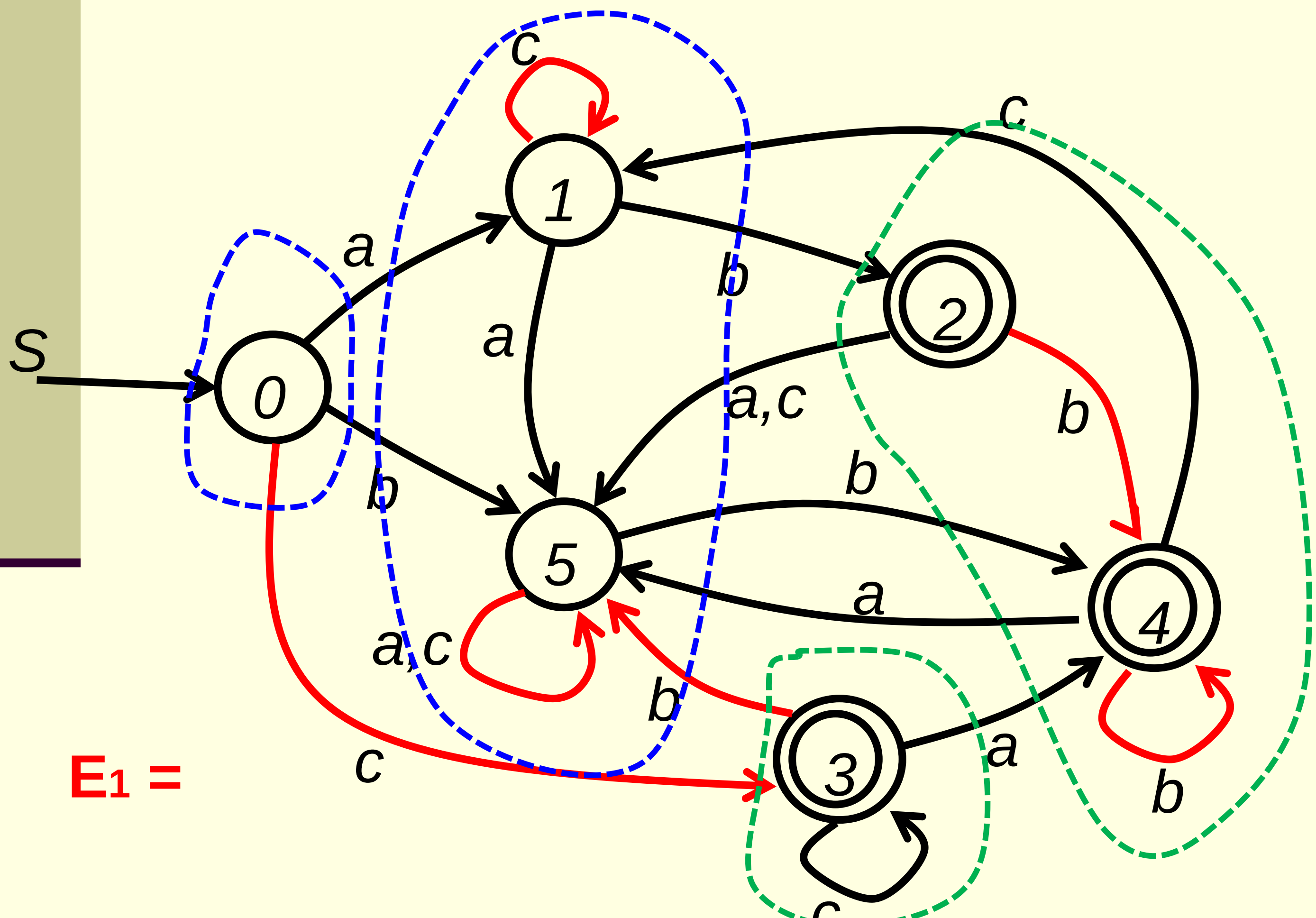
Constructing Efficient Finite Automata

Question: Is the following DFA a minimum-state DFA?



Constructing Efficient Finite Automata

Question: Is the following DFA a minimum-state DFA?



E_2 = ?

$E_2 = E_1$

The minimum-state DFA has 4 states : [0], [1], [2], [3].

E_1 =

Answer: NO

Constructing Efficient Finite Automata

Example.

S = {0, 1, 2, 3, 4, 5} : states of a DFA

0 : start state ;

2, 5 : final states.

The equivalence relation on S for a minimum-state DFA is generated by the following set of equivalent pairs of states: {(0, 1), (0, 4), (1, 4), (2, 5)}

Write down the states of the minimum-state DFA

Constructing Efficient Finite Automata

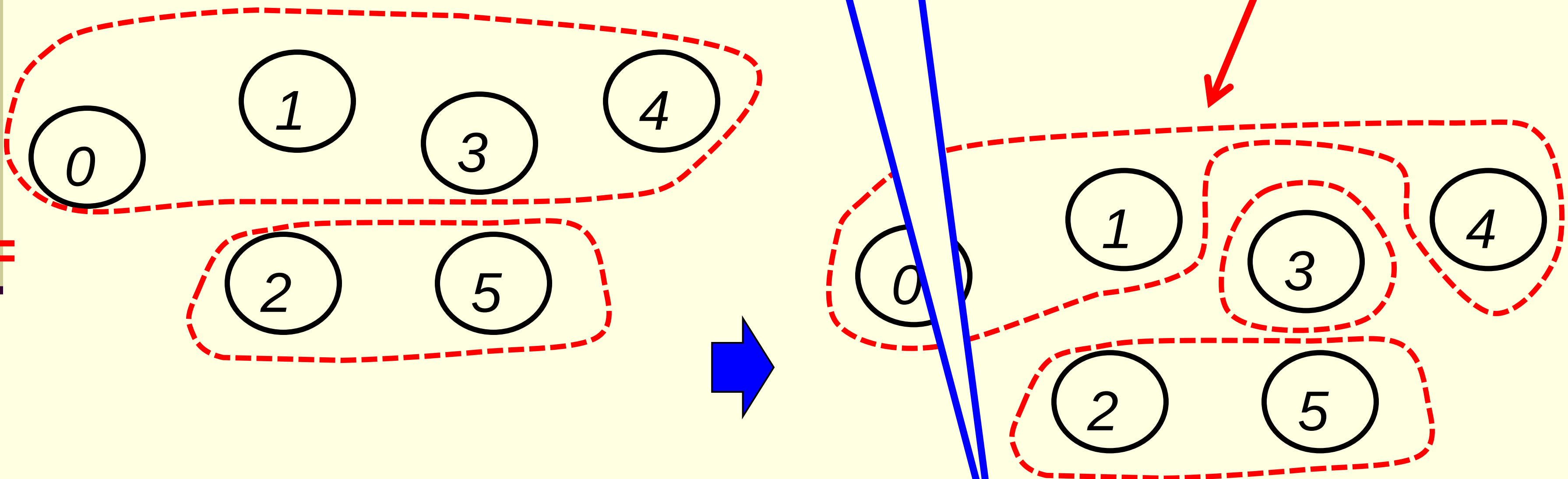
or : [0], [3], [2]

Example. S = {0, 1, 2, 3, 4, 5} ates of a DFA

0 : start state ; 2, nal states.

Equivalent pairs of sta : {(0,1), (0,4), (1,4), (2,5)}

E0 =



So, states of the minimum-state DFA: {0, 1, 4}, {3}, {2, 5}

End of Regular Language and Finite Automata III