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

- Finite Automata

algorithm

Can a machine recognize a regular language?

Yes

Deterministic Finite Automaton (DFA)

A finite digraph over an alphabet A (vertices are called states).

Each state emits one labeled edge for each letter of *A*. One state is defined as the start state and several states may be final states.

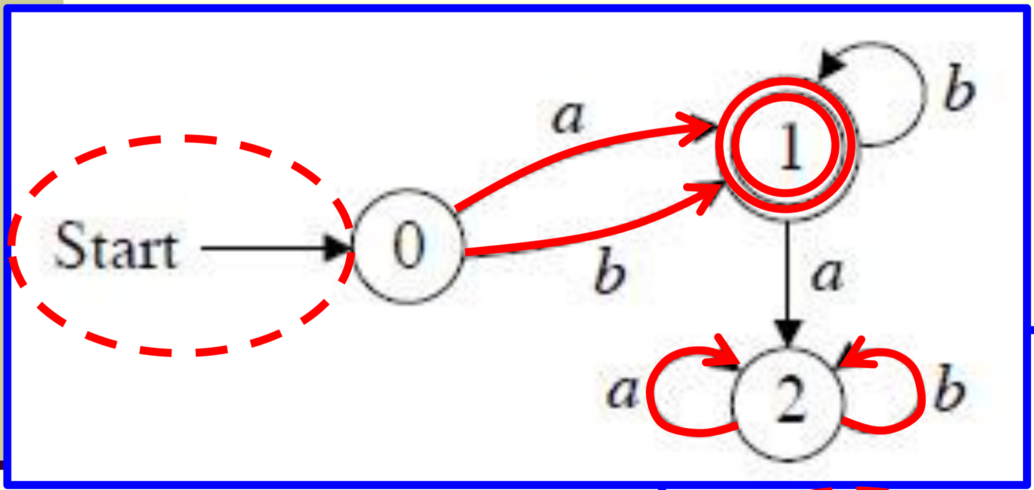
indicated by double circles

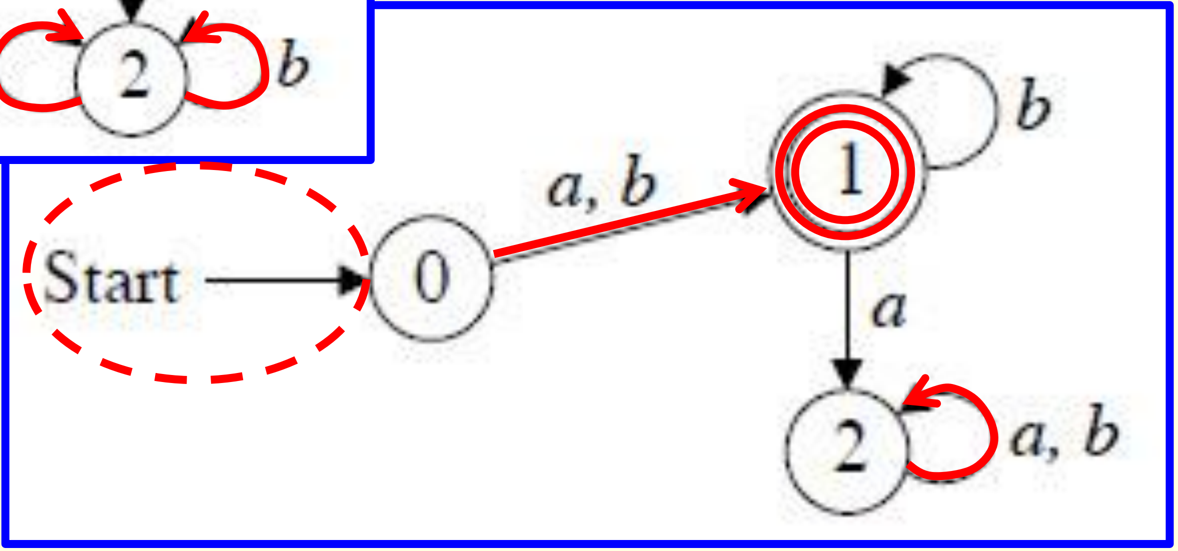
6. Regular Languages & Finite Automata

- Finite Automata

Example.

Either one is acceptable





A={a, b}

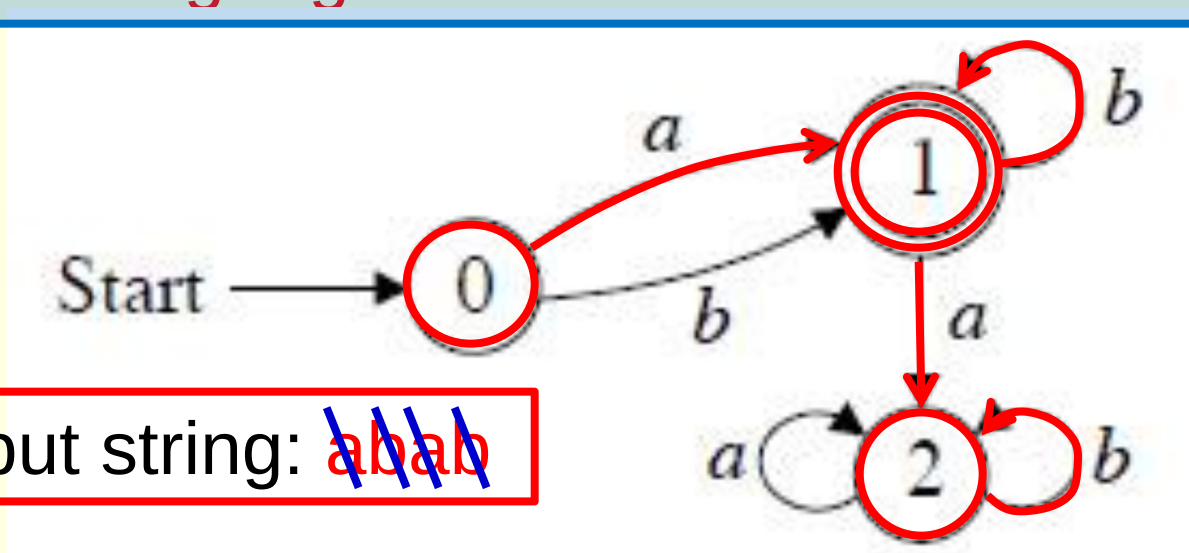
6. Regular Languages & Finite Automata

- Finite Automata

The **execution** of DFA for input string $w \in A^*$ begins at the **start state** and follows a **path** whose edges concatenate to w.

The DFA **accepts** w if the path ends in a **final state**. Otherwise the DFA **rejects** w.

The **language of a DFA** is the set of **accepted strings**.

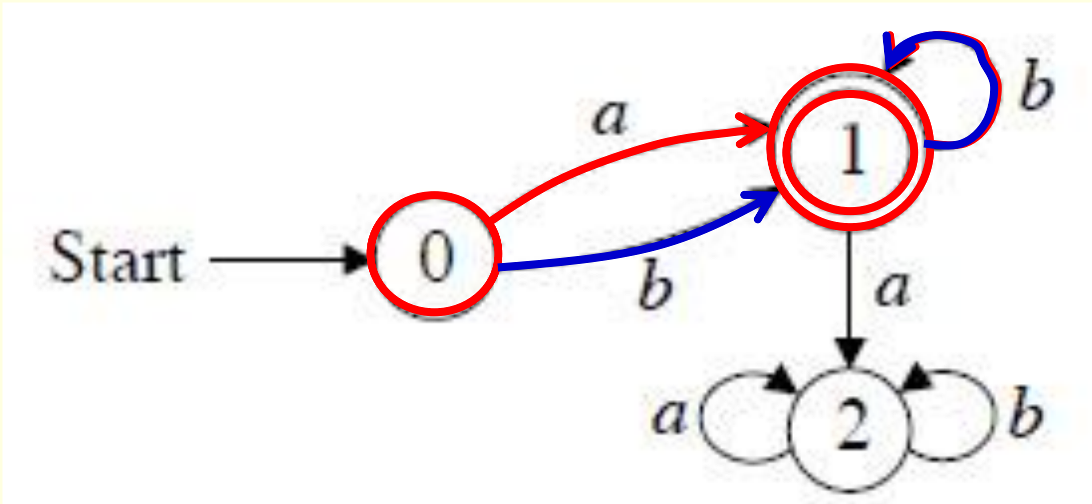


Input string: ~~abab~~

an empty string will enter the start state but the empty set will not.

6. Regular Languages & Finite Automata

- Finite Automata



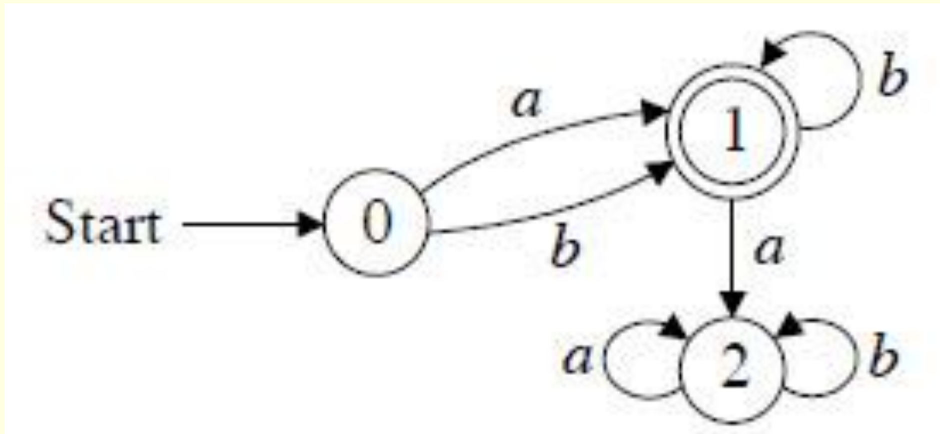
Example. The example DFA accepts the strings

a, b, $\boxed{ab}$, bb, $\boxed{abb}$, bbb, …, $\boxed{ab^n}$, $\boxed{bb^n}$, …

The language of the DFA is $\{ ab^n, bb^m \mid n \in N, m \in N \}$

6. Regular Languages & Finite Automata

– Finite Automata



Example. The example DFA accepts the strings

$a, b, ab, bb, abb, bbb, \ldots, ab^n, bb^n, \ldots$

The regular expression of the language of the DFA is

$\boxed{(a+b)b^*}$ Why?

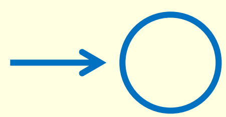
$a+b$ $(a+b)b$ $(a+b)b^2$ $(a+b)b^n$

6. Regular Languages & Finite Automata

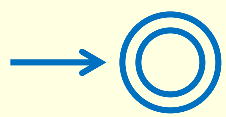
- Finite Automata

Theorem (Kleene) The class of regular languages is exactly the same as the class of languages accepted by DFAs.

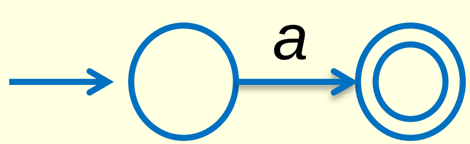
Proof. Need three lemmas or by induction.



accepts $\varnothing$



accepts $\{\Lambda\}$

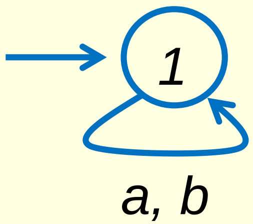


accepts $\{a\}$

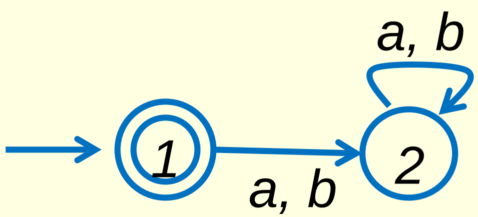
6. Regular Languages & Finite Automata

- Finite Automata

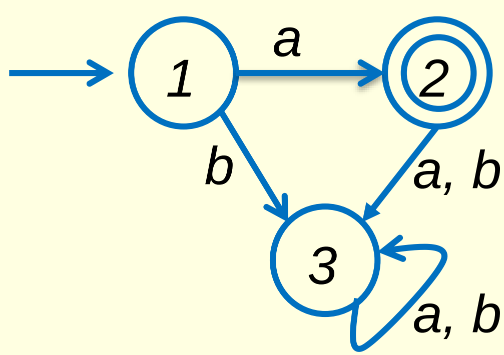
Specifically, say A = {a, b}, then



accepts $\varnothing$



accepts $\{\Lambda\}$



accepts $\{a\}$

6. Regular Languages & Finite Automata

- Finite Automata

Theorem (Kleene) The class of regular languages is exactly the same as the class of languages accepted by DFAs.

Proof. (conti.)

Inductive step: prove that if L_1 and L_2 are accepted by DFAs, then $\boxed{L_1 \cup L_2}$, $\boxed{L_1 L_2}$ and $\boxed{L_1^*}$ are accepted by DFAs.

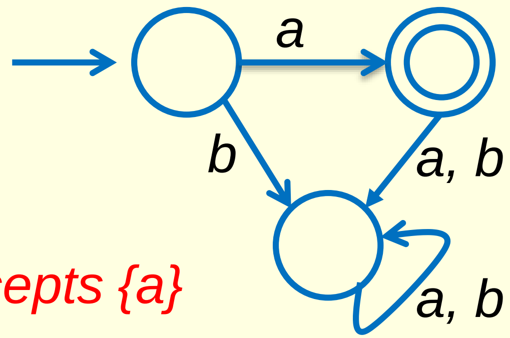
Since any regular language is obtained from $\{\Lambda\}$ and $\{a\}$ for any symbol a in the alphabet A by using union, concatenation and Kleene star operations, that

together with the basis step would prove the theorem.

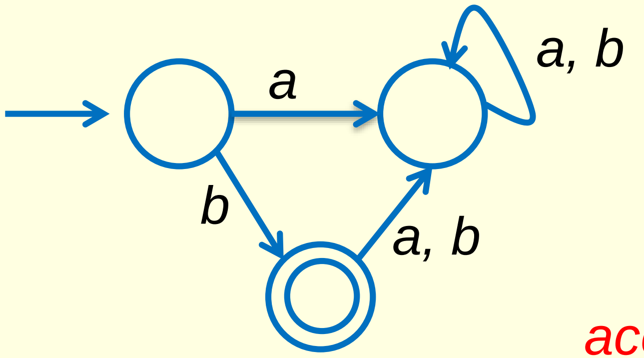
6. Regular Languages & Finite Automata

- Finite Automata

For instance, if L_1 = {a} and L_2 = {b}

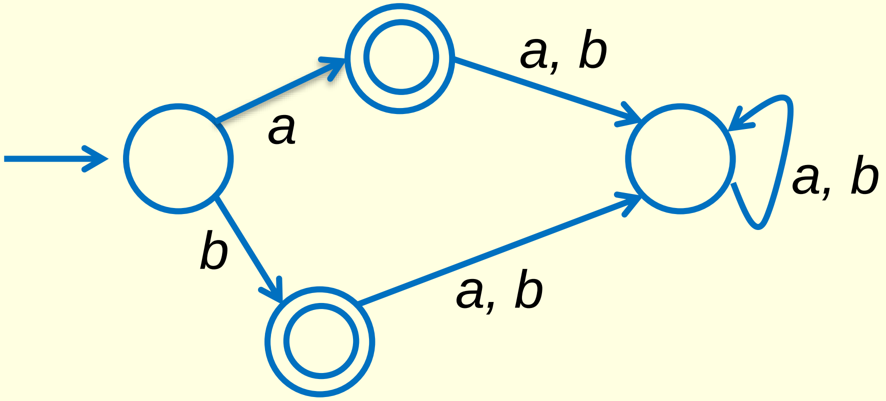


accepts {a}



accepts {b}

then

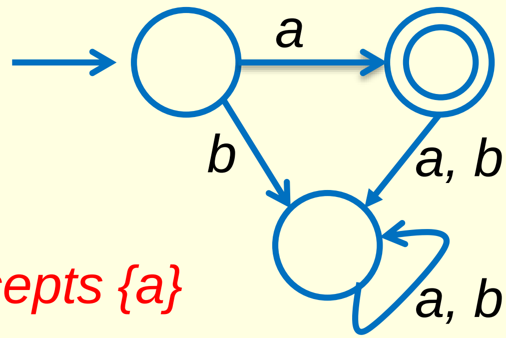


accepts {a, b}

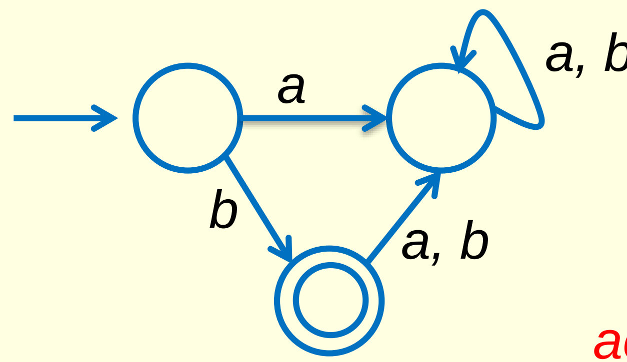
6. Regular Languages & Finite Automata

- Finite Automata

For instance, if L_1 = {a} and L_2 = {b}

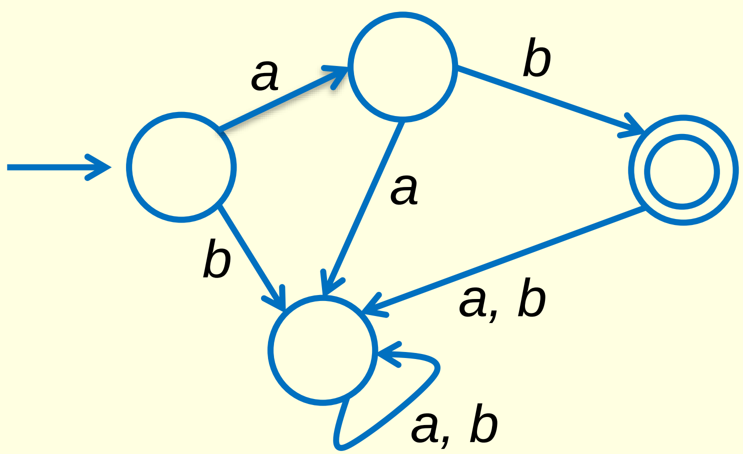


accepts {a}



accepts {b}

then

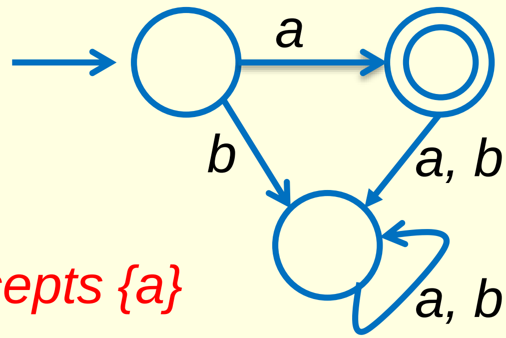


accepts {ab}

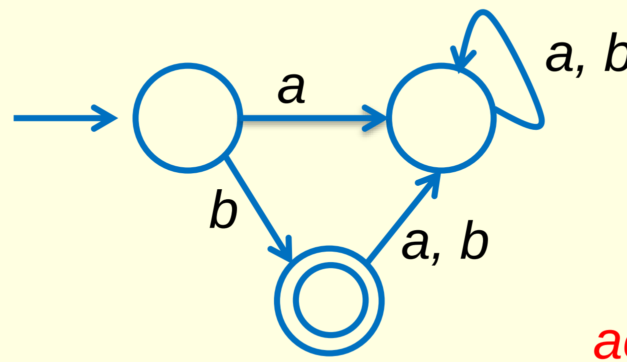
6. Regular Languages & Finite Automata

- Finite Automata

For instance, if L_1 = {a} and L_2 = {b}

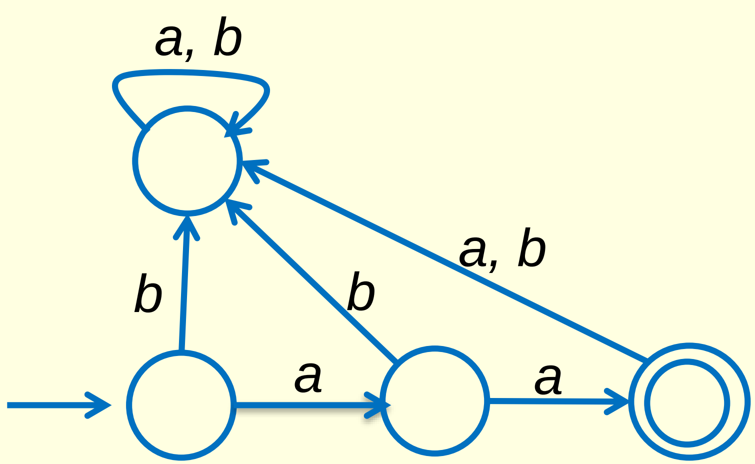


accepts {a}



accepts {b}

then

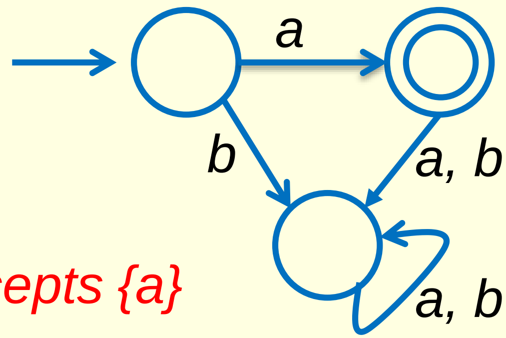


accepts {aa}

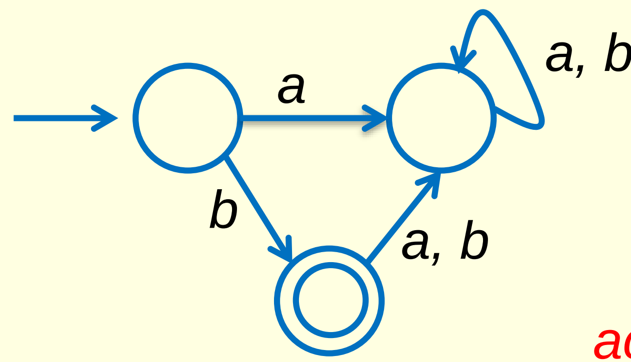
6. Regular Languages & Finite Automata

- Finite Automata

For instance, if L_1 = {a} and L_2 = {b}

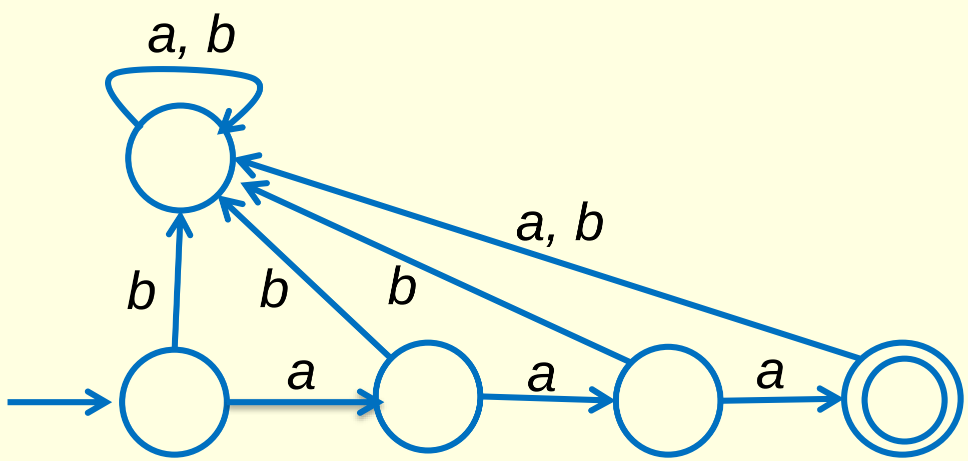


accepts {a}



accepts {b}

or



accepts {aaa}

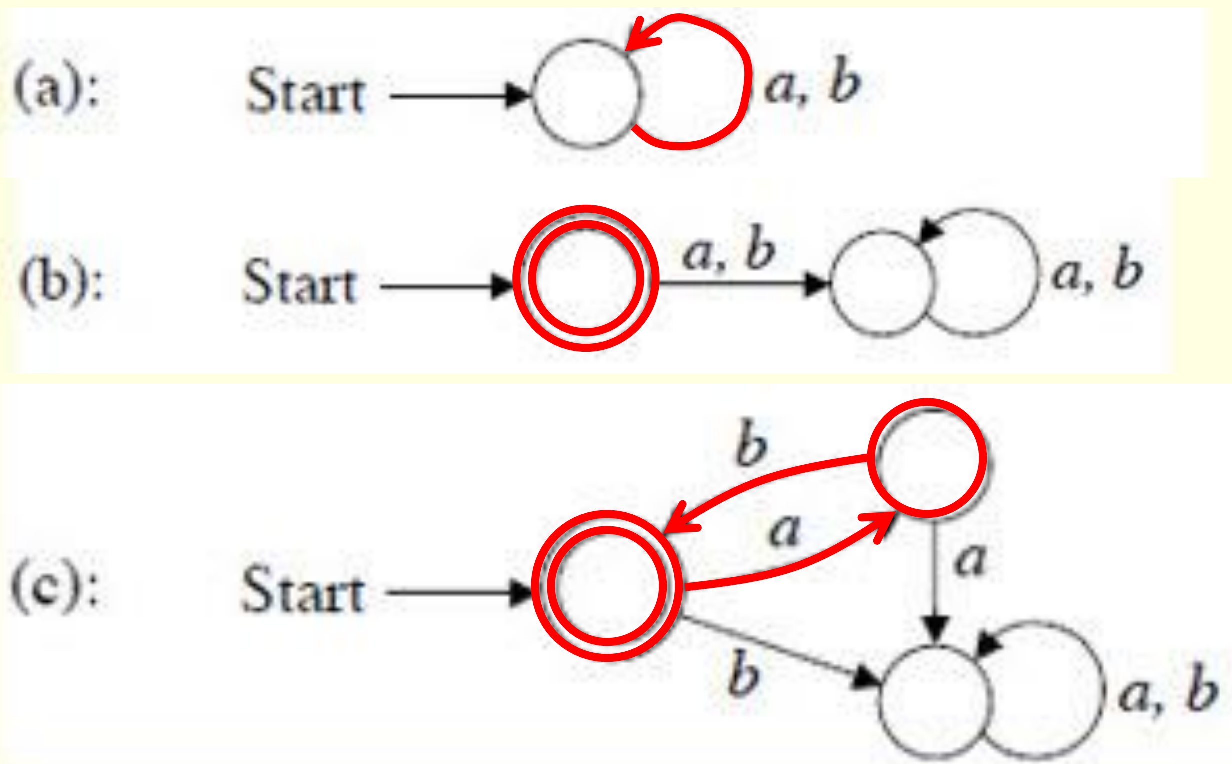
6. Regular Languages & Finite Automata

- Finite Automata

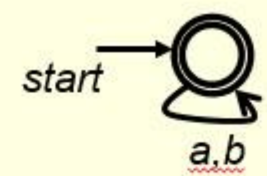
Example. Find a DFA for each language over the alphabet {a,b}.

(a) ∅. (b) {Λ}. (c) $\{ (ab)^n \mid n \in \mathbf{N} \}$, which has regular expression (ab)*.

Solution:



Would this DFA work?

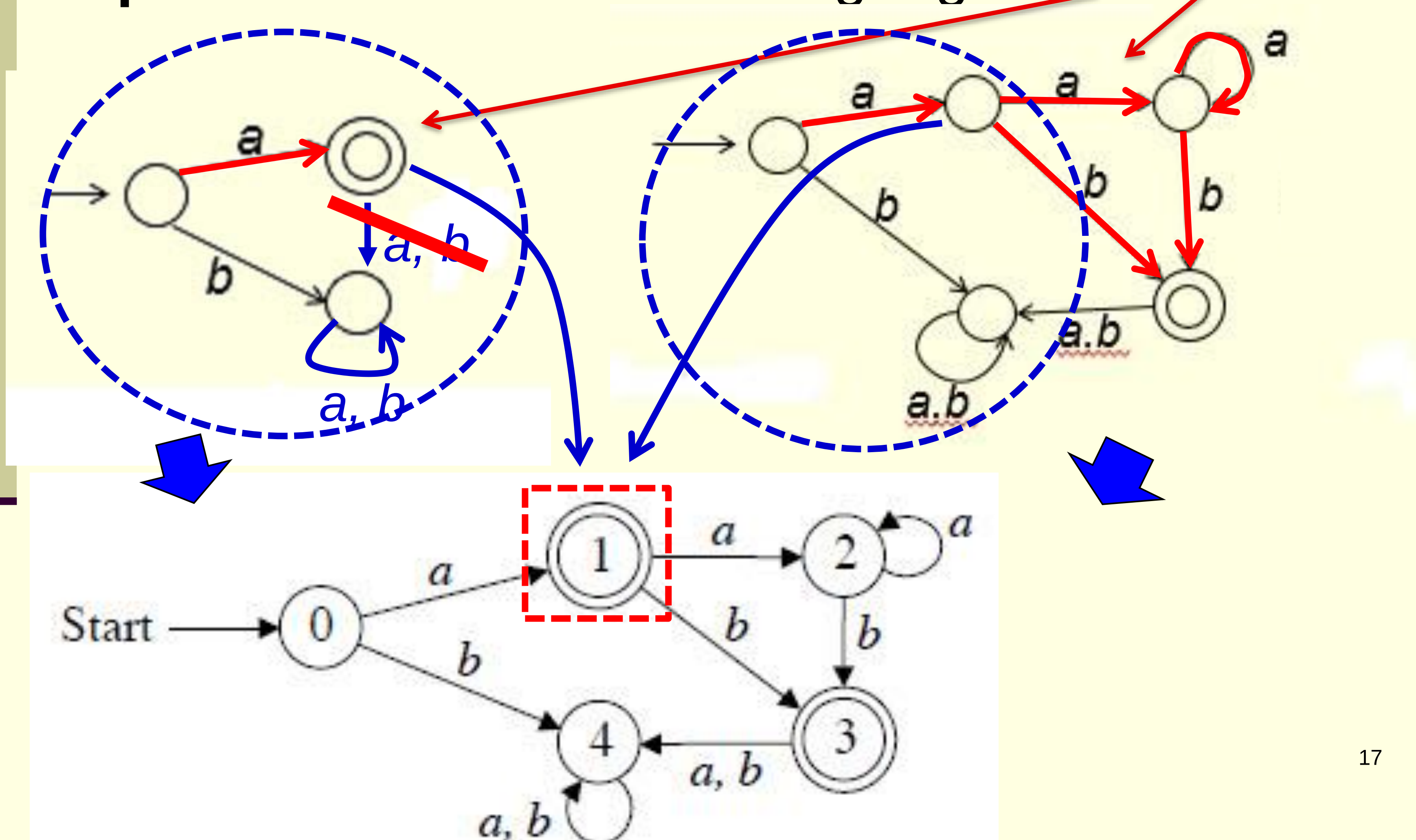


No, it accepts {a, b}*

6. Regular Languages & Finite Automata

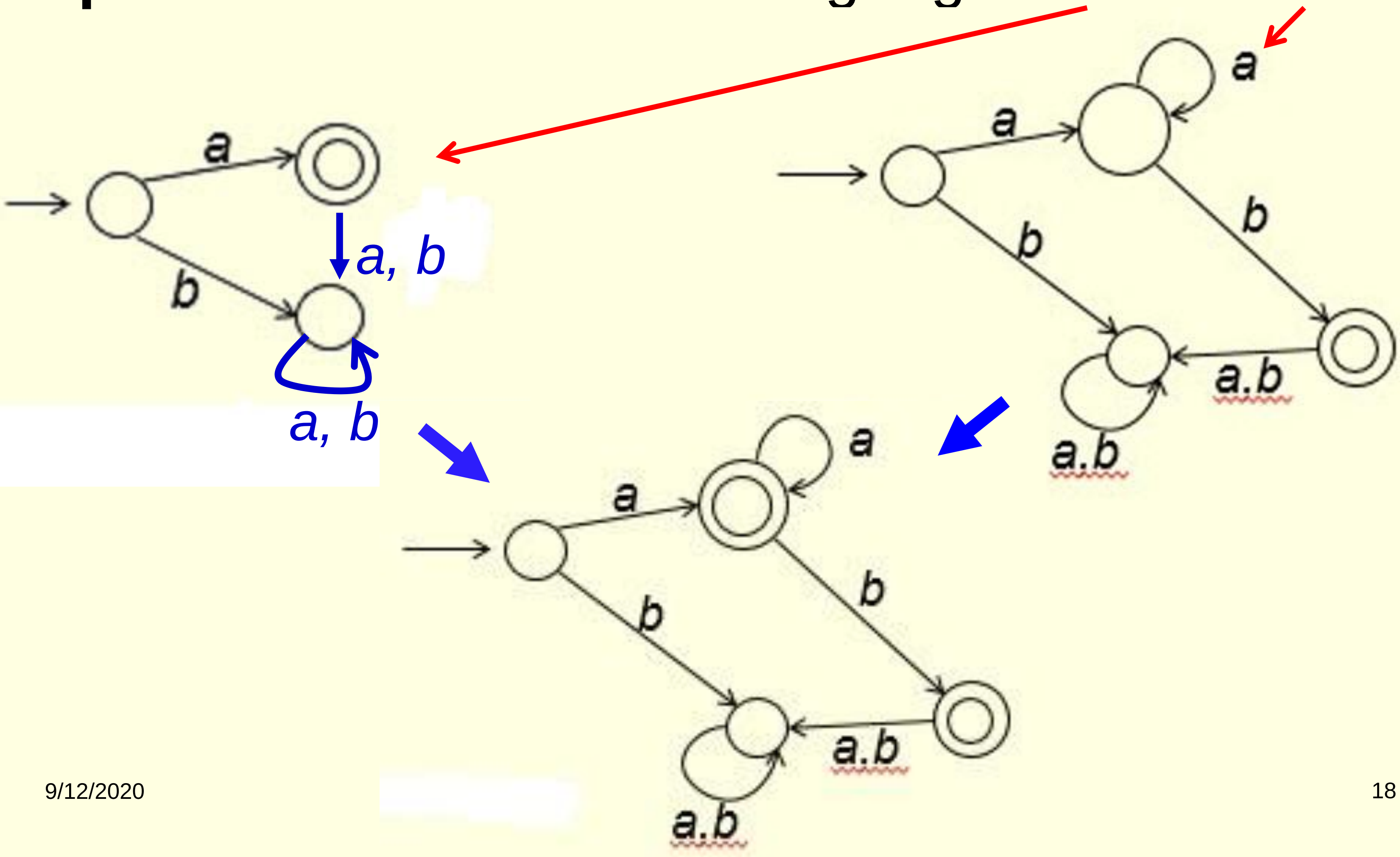
- Finite Automata

Example. Find a DFA for the language of $a + aa^*b$.



What is the problem with the following approach?

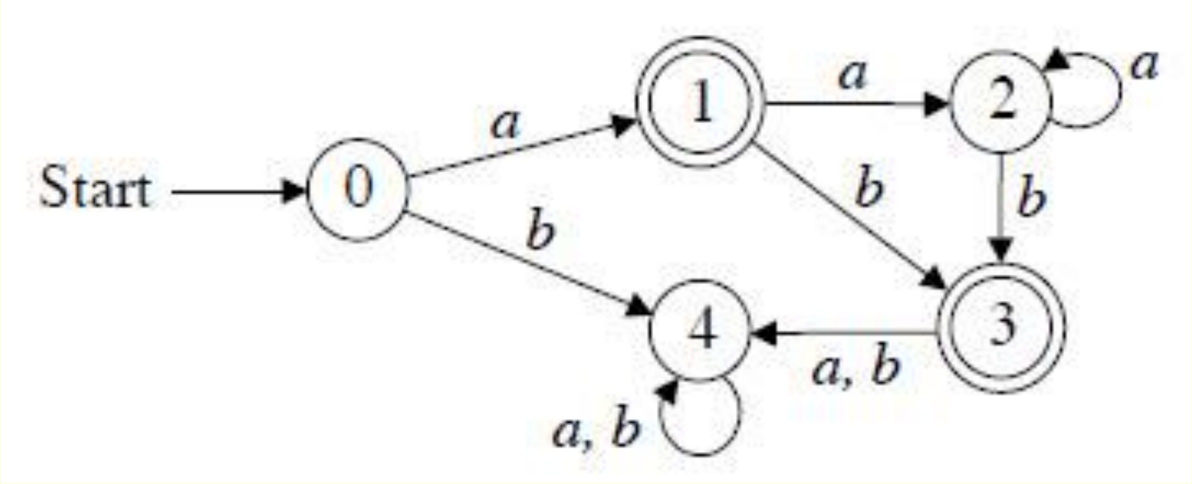
Example. Find a DFA for the language of *a + aa*b*.



What is the problem with the following approach?

Example. Find a DFA for the language of *a + aa*b.*

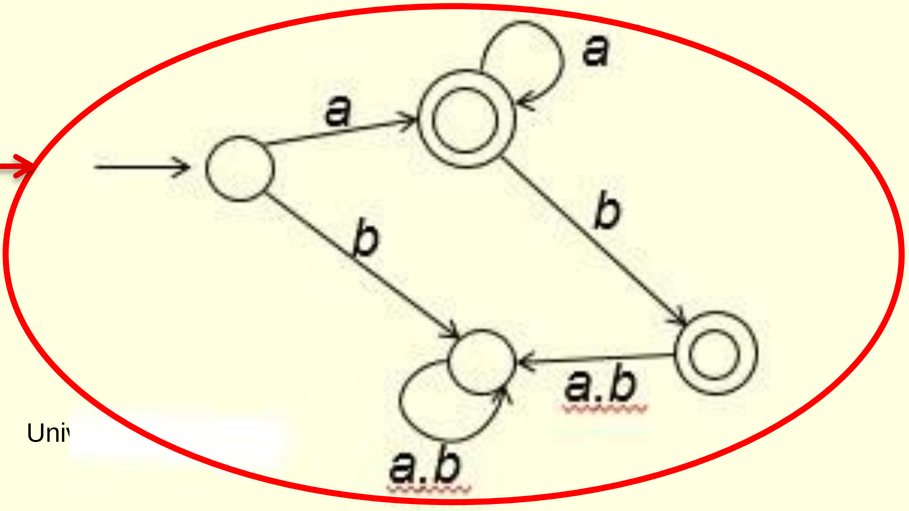
Solution:



Question:

Would this DFA work?

aa is accepted by this DFA, but …



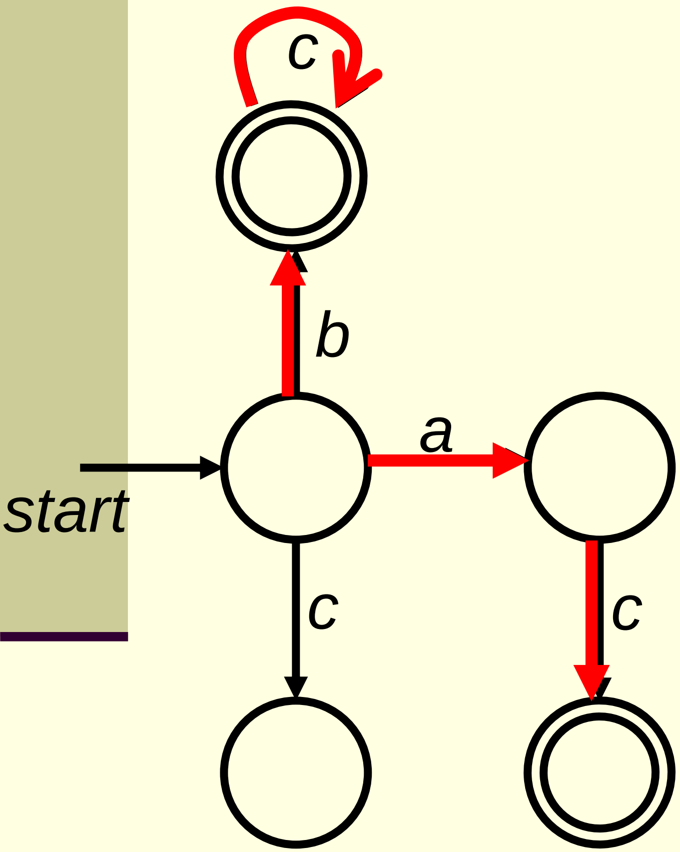
6. Regular Languages & Finite Automata

- Finite Automata

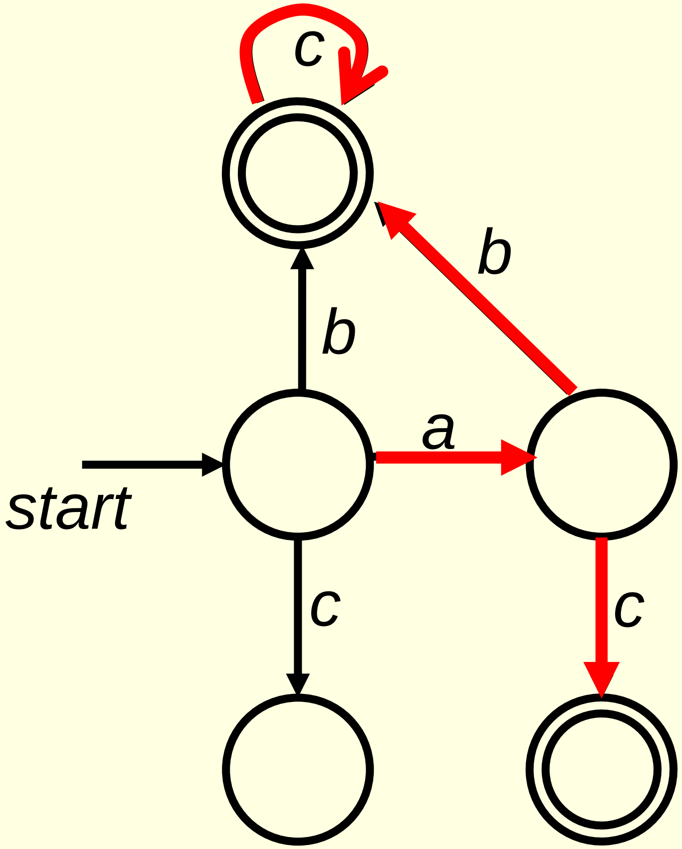
Loops are dangerous because if a path contains a loop then the length of that path is not unique!

You use an internal or external loop in a DFA only if you want to recognize an expression with variable length.

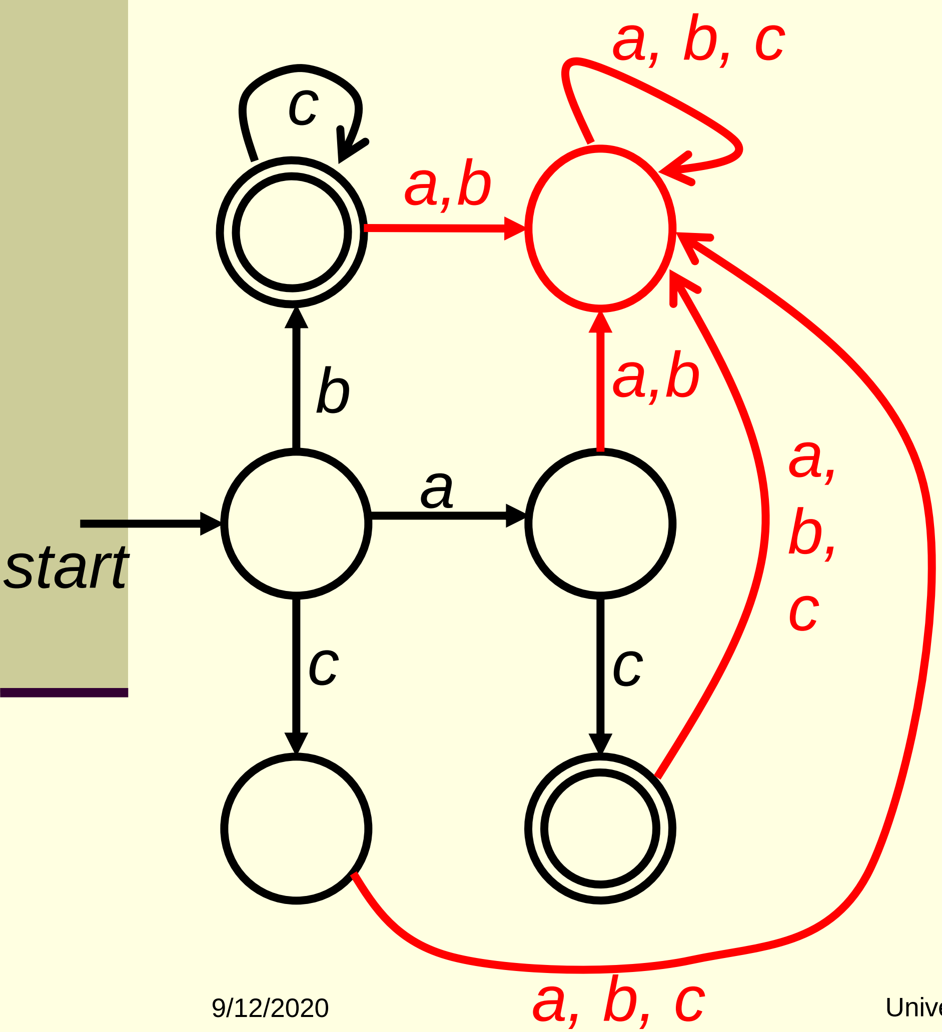
A DFA that recognizes *bc* + ac*



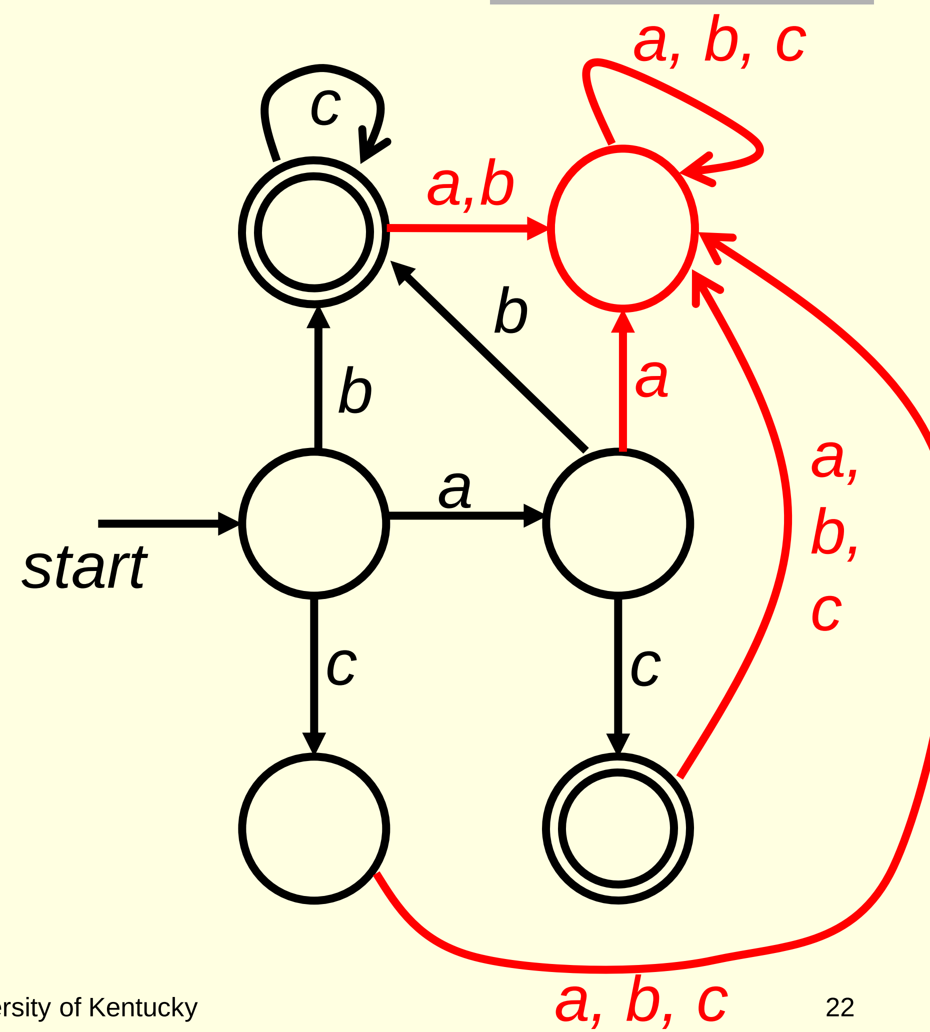
A DFA that recognizes *bc* + abc* + ac*



A DFA that recognizes *bc* + ac*

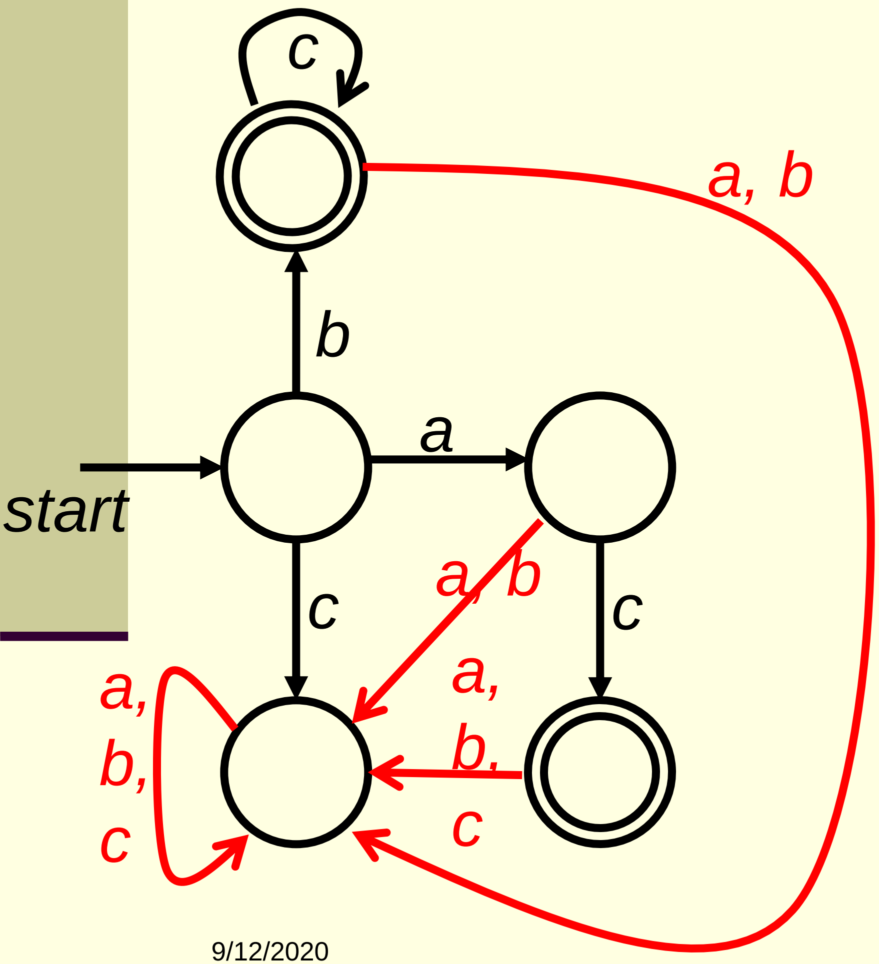


A DFA that recognizes *bc* + abc* + ac*



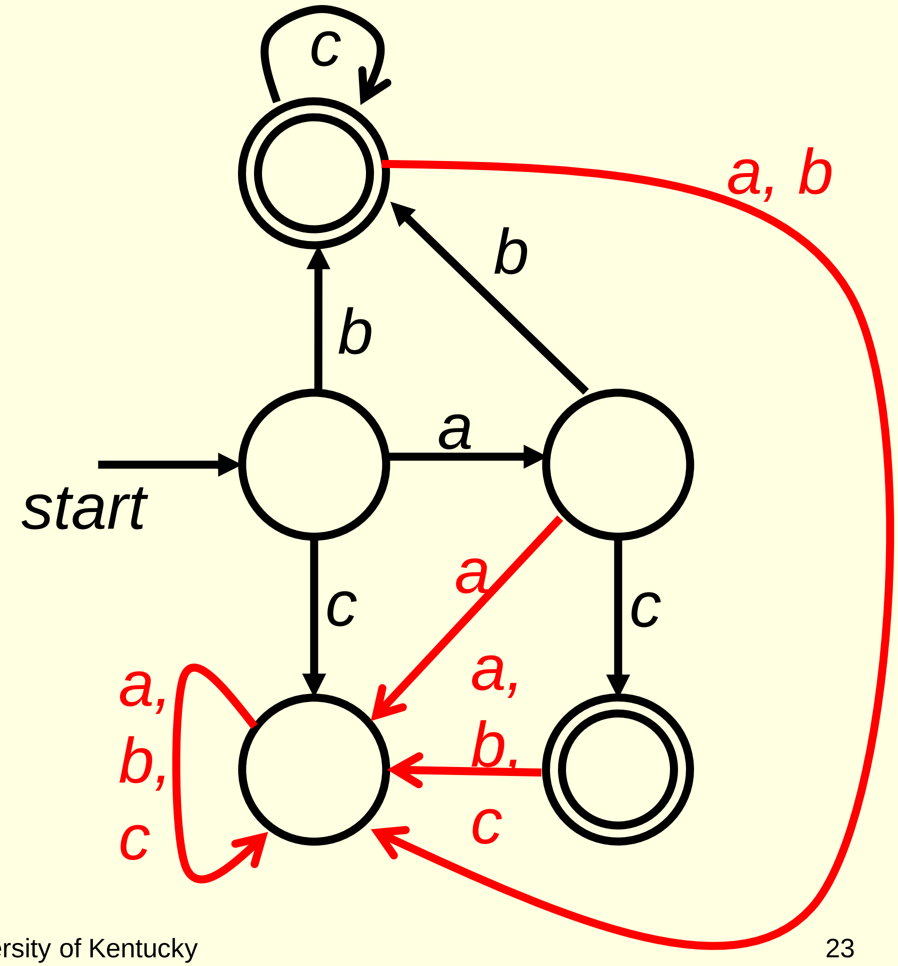
A DFA that recognizes

bc + ac*



A DFA that recognizes

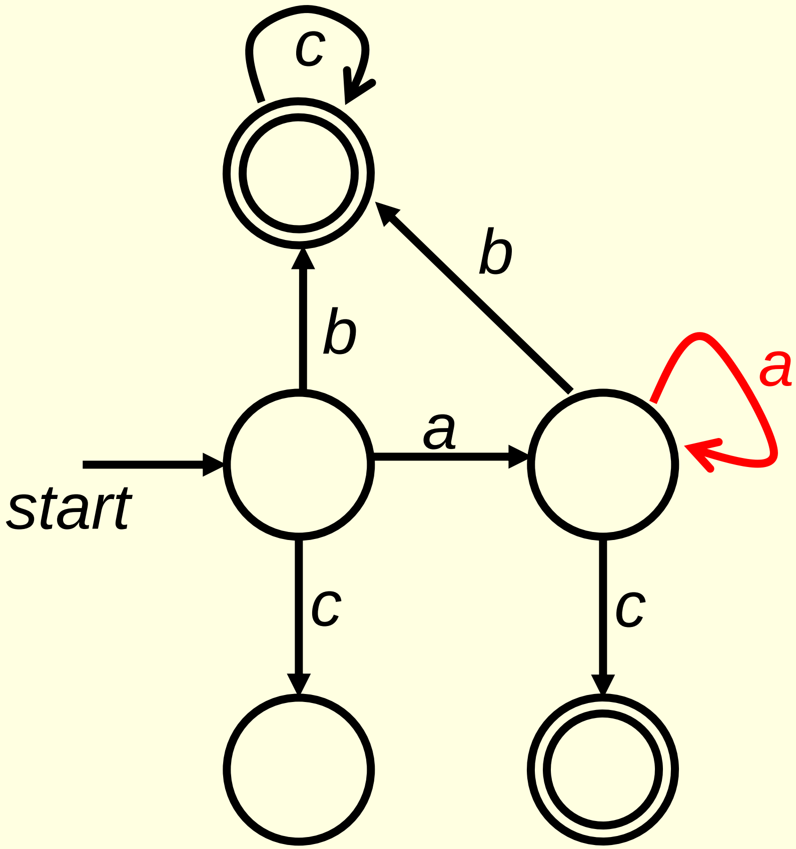
bc + abc* + ac*



To make each of these FA's a DFA, you either create a new state or use a non-final state as the sink of all the remaining edges of the FA.

That state should not have outgoing edges.

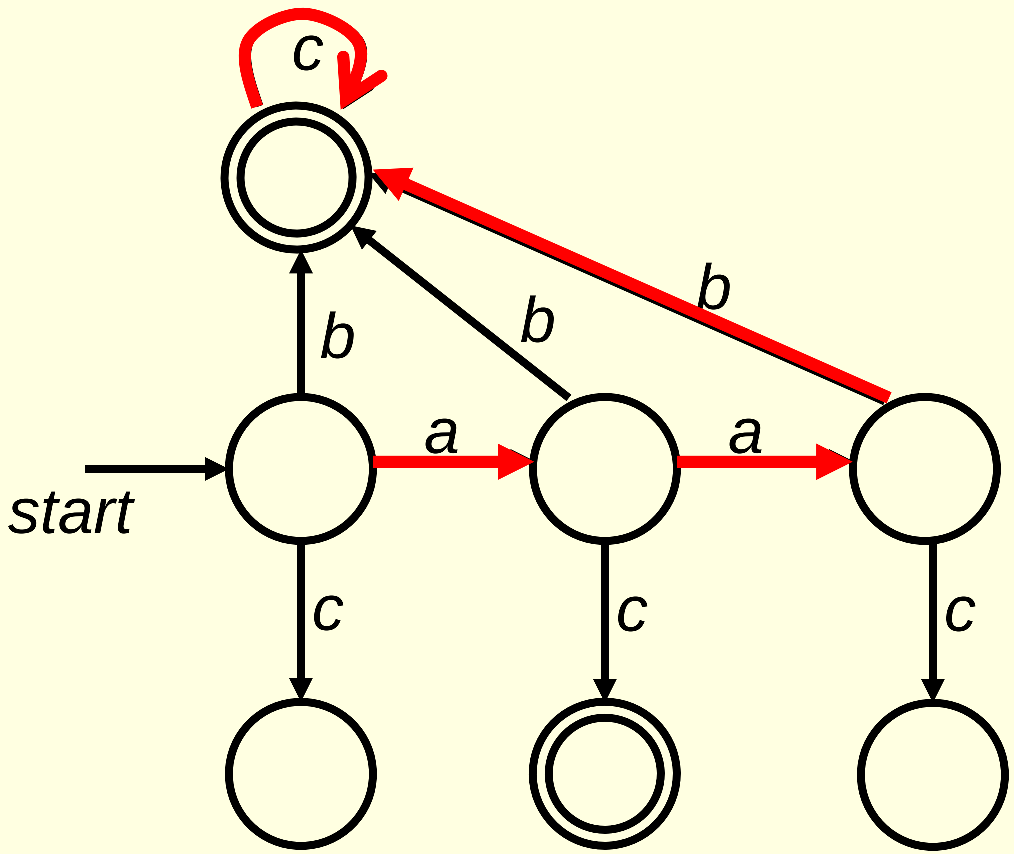
Would the following DFA recognize *a*bc* + ac ?*



In addition to a*bc* and ac, does it recognize anything else?

Yes, such as *aac, aaac, …*

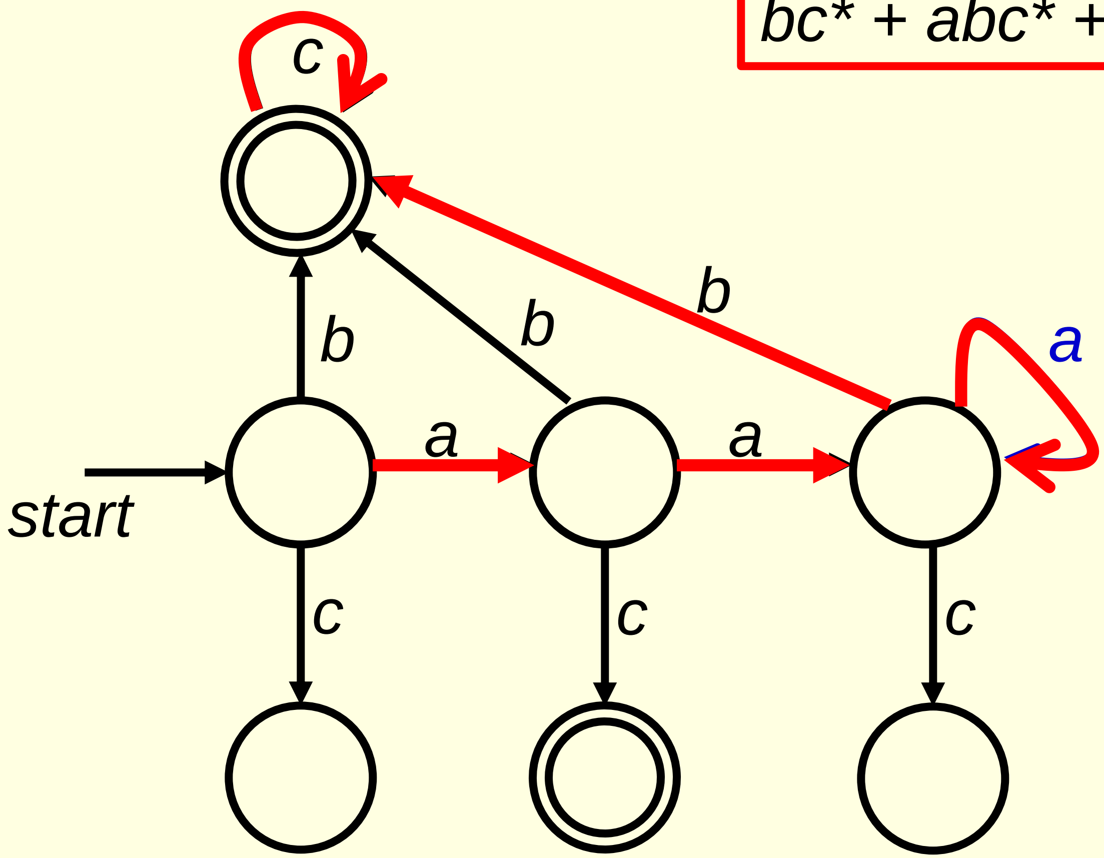
A DFA that recognizes $bc^* + abc^* + a^2bc^* + ac$



A DFA that recognizes *a*bc* + ac*

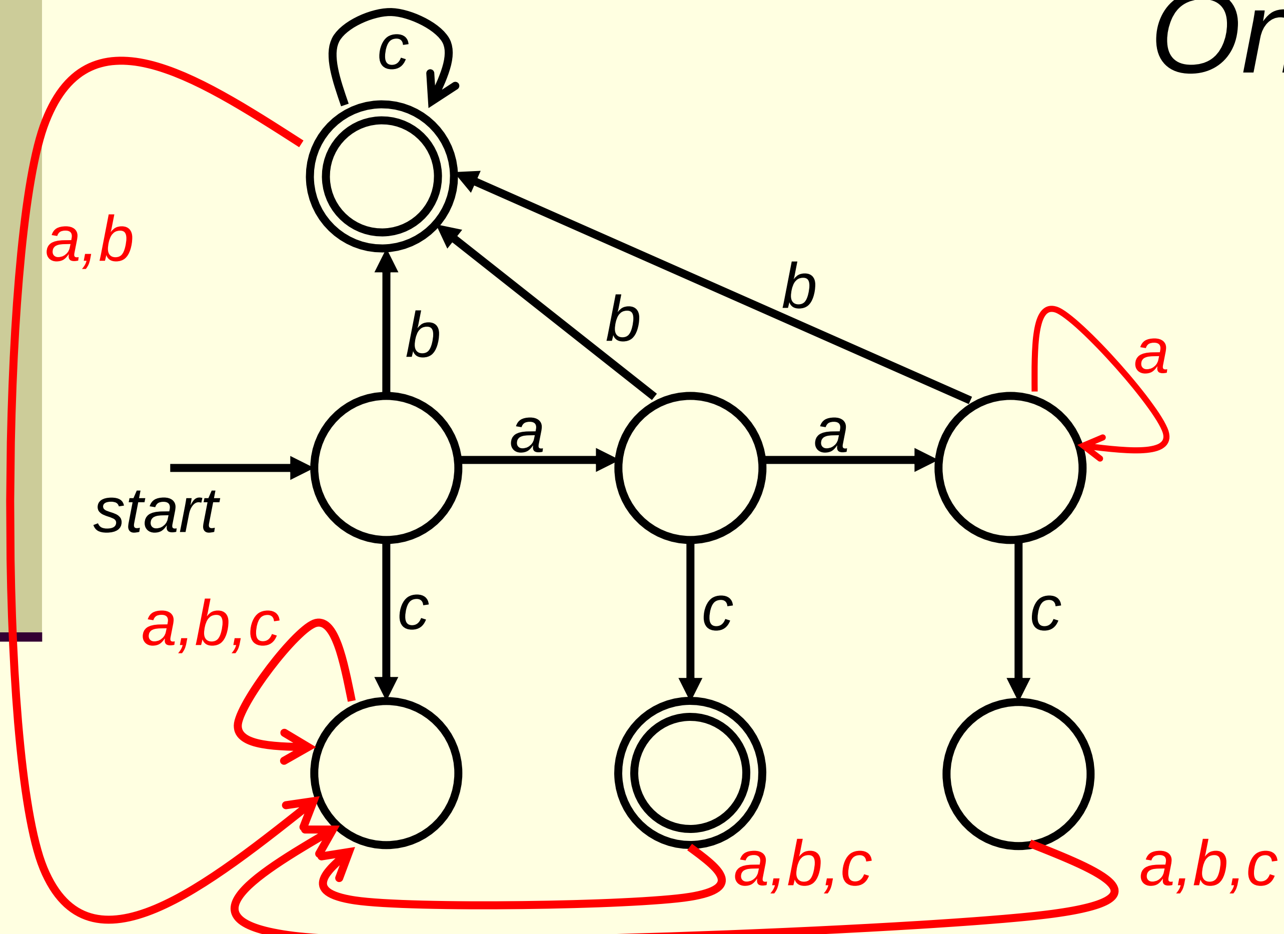
$bc^* + abc^* + a^2bc^* + a^nbc^*$

$n \geq 3$



How to make this FA a real DFA?

A DFA that recognizes *a*bc* + ac*



One option:

a real DFA now

6. Regular Languages & Finite Automata

- Finite Automata

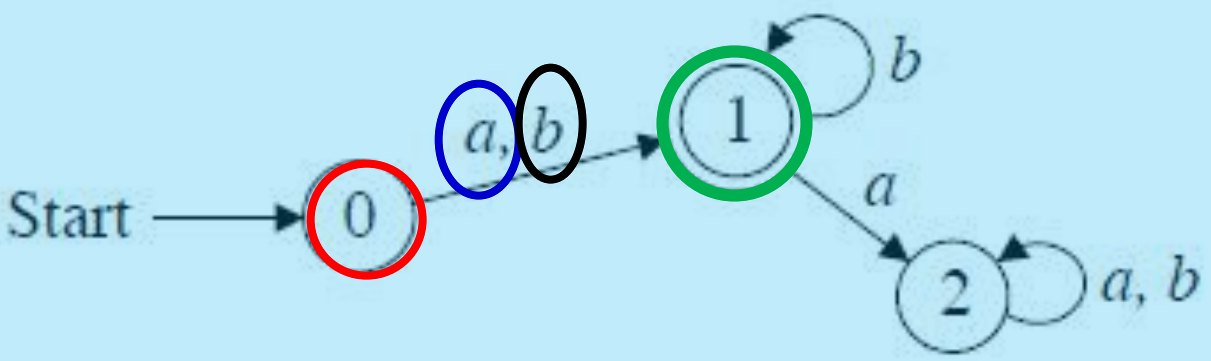
Table Representation of a DFA

DFA over *A* can be represented by a transition function

$$T : States \times A \rightarrow States,$$

where $T(i, a)$ is the state reached from state i along the edge labeled a, and we mark the start and final states.

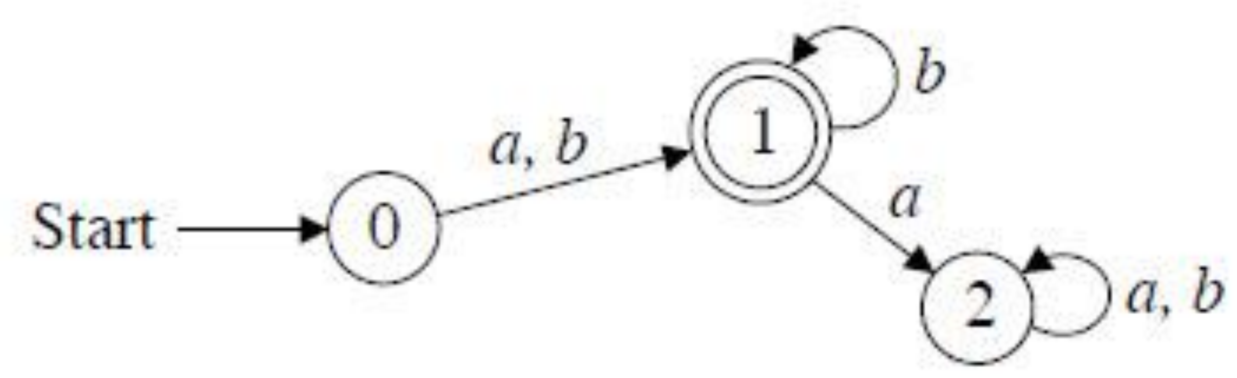
Example:



	T	a	b
start	0	1	1
final	1	2	1
	2	2	2

6. Regular Languages & Finite Automata

- Finite Automata



	T	a	b
start	0	1	1
final	1	2	1
	2	2	2

Note: T can be extended to $T : States \times A^* \rightarrow States$

by $T(i, \Lambda) = i$, $T(i, aw) = T(T(i, a), w)$ $a \in A$, $w \in A^*$

Question: $T(0, bba) = ?$

or

$T(0, bba) = T(1, ba) = T(1, a) = T(2, \Lambda) = 2.$

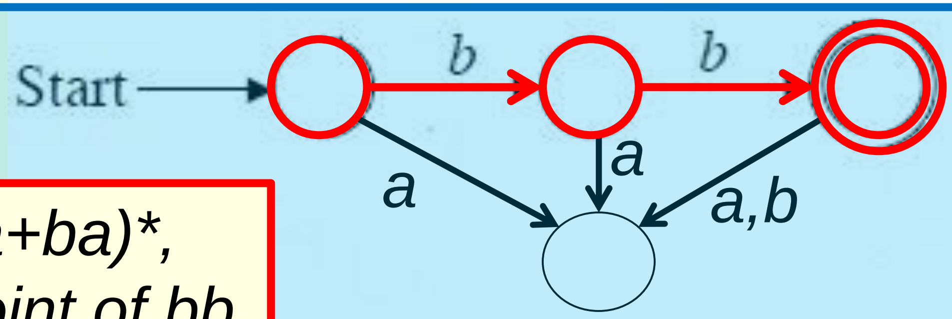
6. Regular Languages & Finite Automata

- Finite Automata

Example. Find a DFA to recognize *(a + ba)*bb(a + ab)*.*

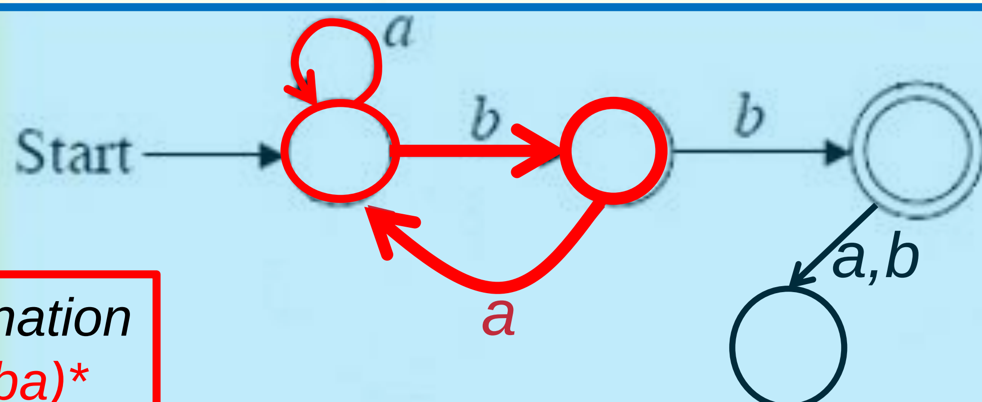
A solution:

{bb}



After excuting (a+ba),*
return to start point of bb

$\{(a+ba)^{*}bb\}$



any combination of a and (ba)**

6. Regular Languages & Finite Automata

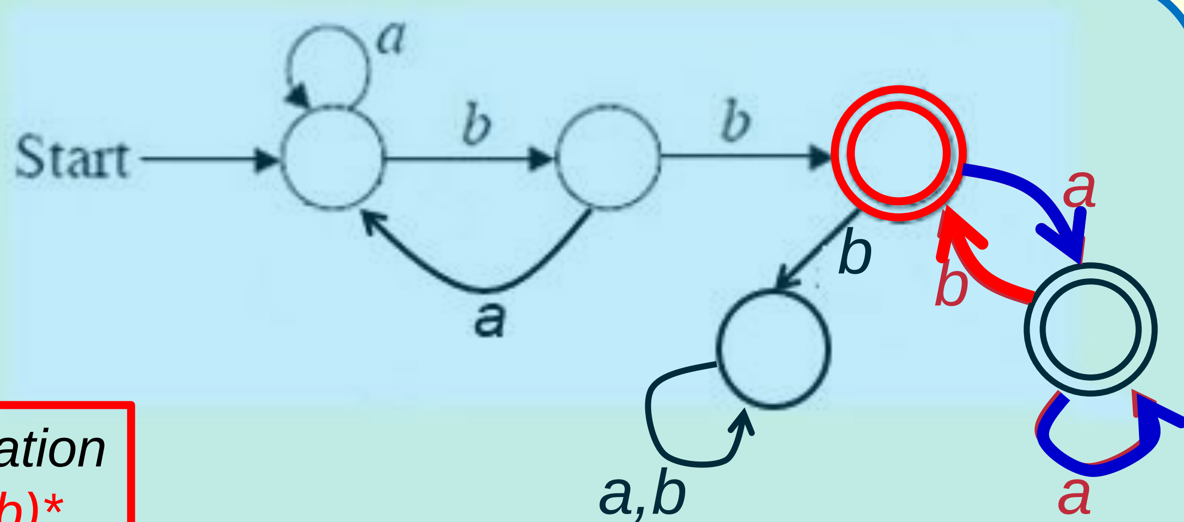
- Finite Automata

Example (conti). Find a DFA to recognize *(a + ba)*bb(a + ab)**.

A solution:

$$\{(a+ba)^{*}bb(a+ab)^{*}\}$$

any combination of a and (ab)**

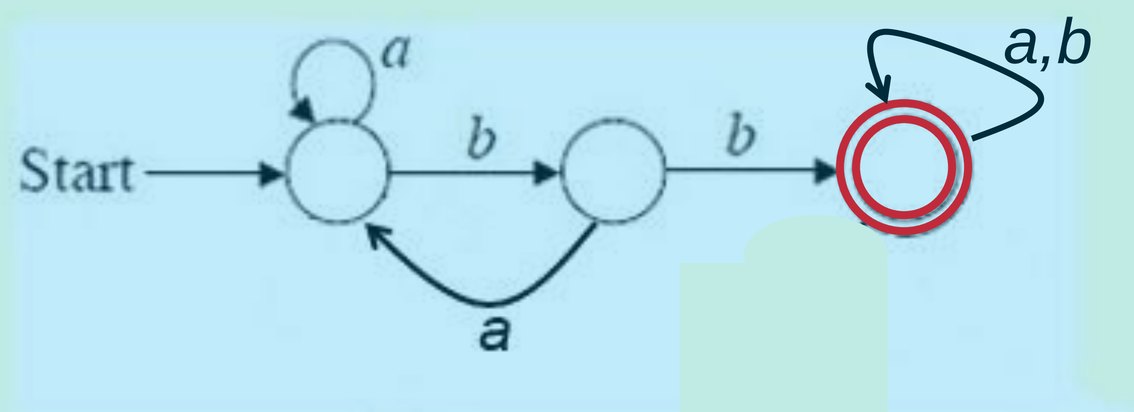


Would the following approach work?

Example (conti). Find a DFA to recognize $(a + ba)^*bb(a + ab)^*$.

A solution:

$$\{(a+ba)^*\underline{bb}(a+ab)^*\}$$



any combination of a^* and $(ab)^*$

6. Regular Languages & Finite Automata

- Finite Automata

Nondeterministic Finite Automata (NFA)

An NFA over an alphabet *A* is similar to a DFA except that

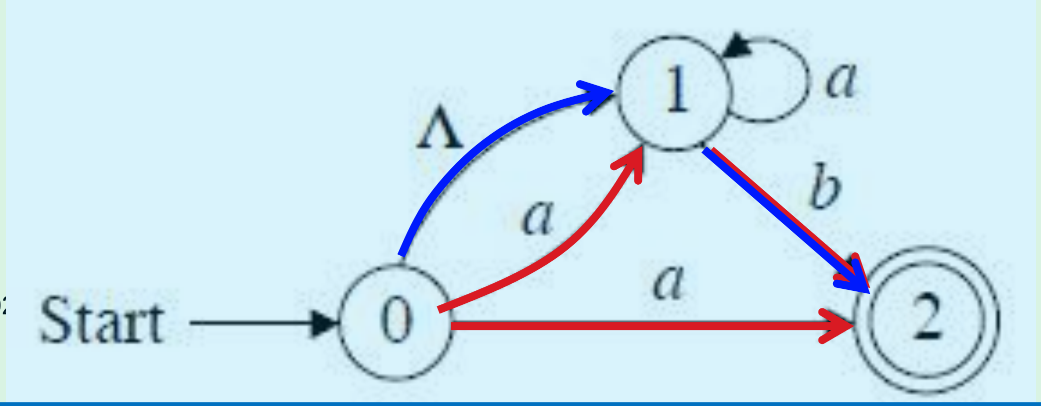
Λ-edges are allowed,

there is no requirement to emit edges from a state, and

multiple edges with the same letter can be emitted from a state.

Example. The following NFA recognizes the language of

$a + aa^*b + a^*b$.



a ?

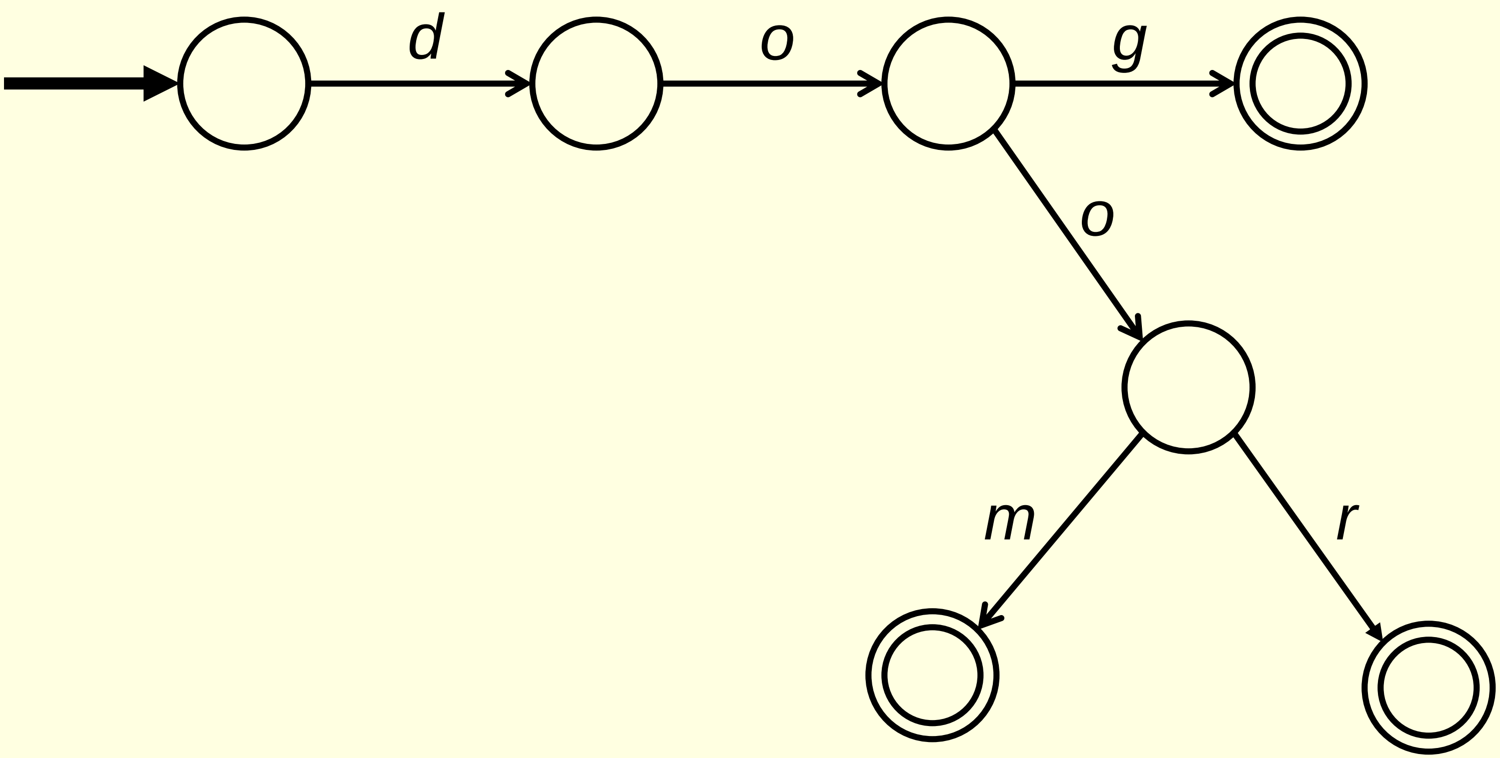
*aa*b*: *ab*, *aab* ?

*a*b*: *b*, *ab* ?

Intuitive examples

A = { d, g, m, o, r}

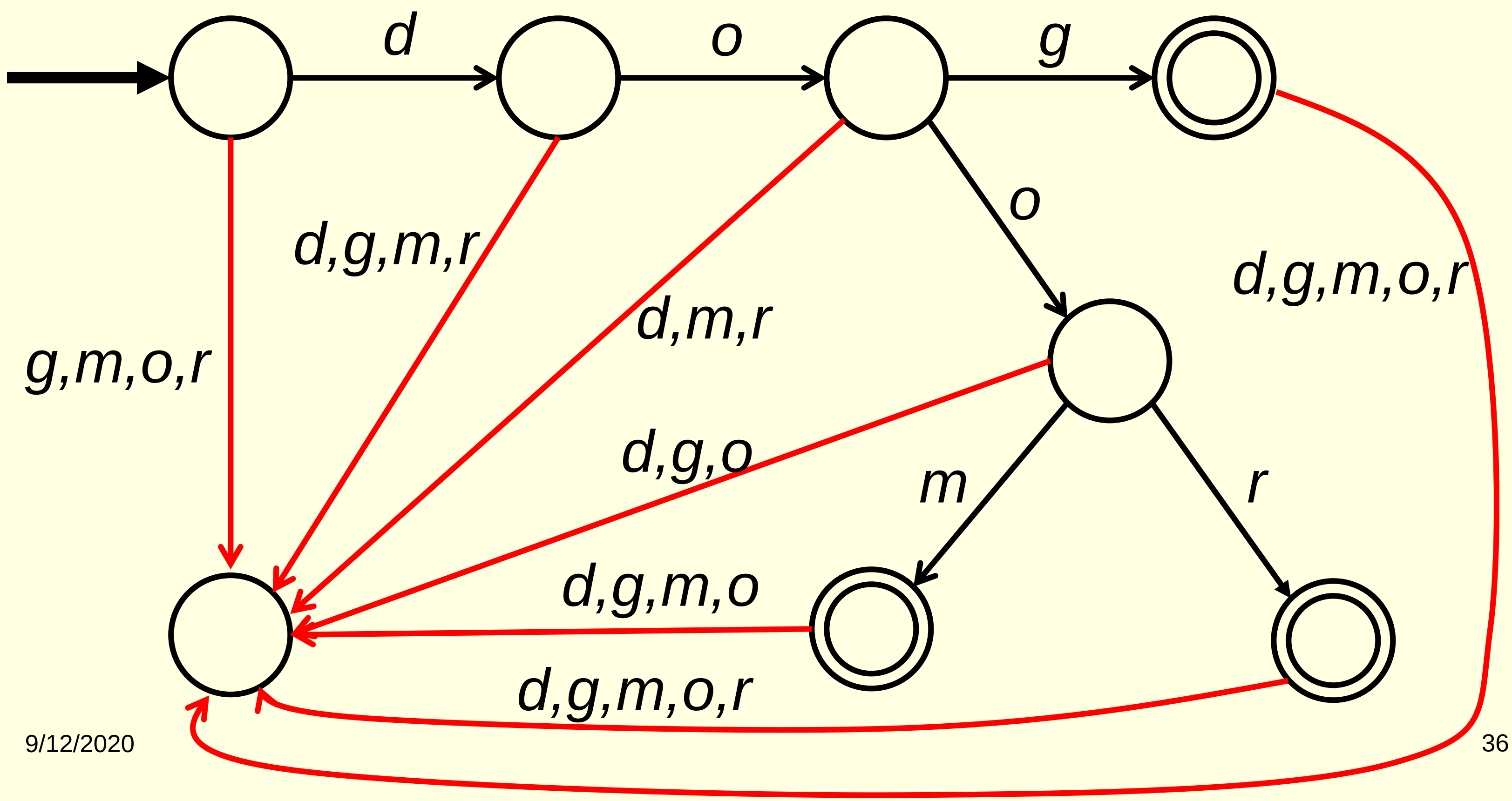
NFA



Intuitive examples

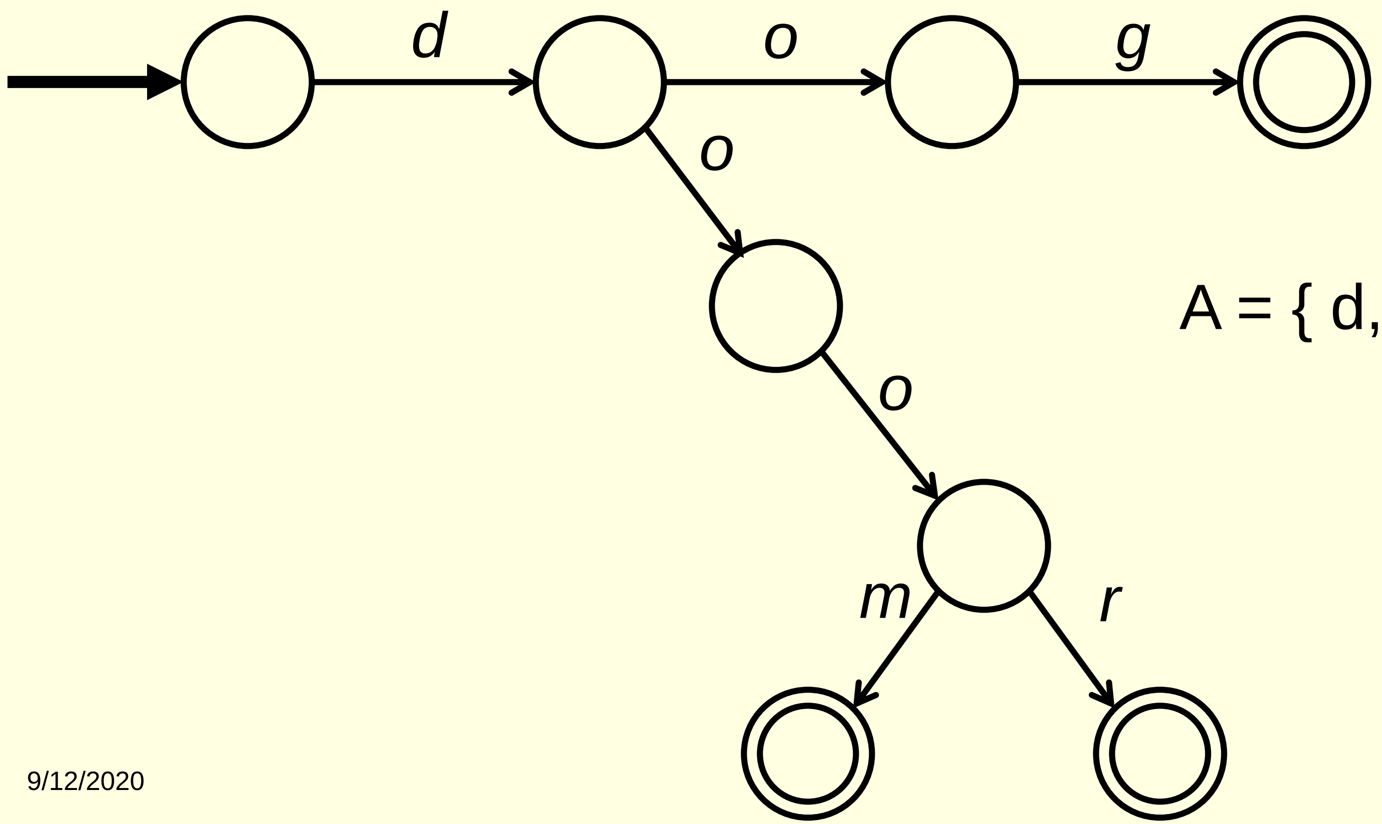
DFA

A = { d, g, m, o, r}



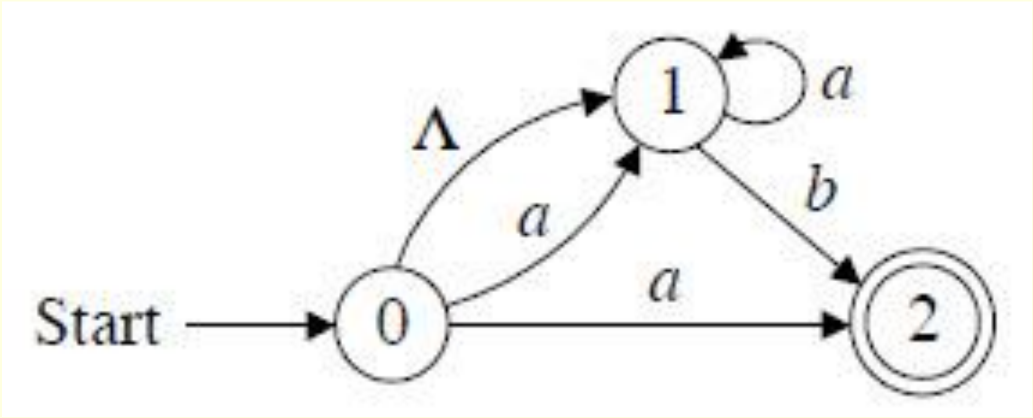
Intuitive examples

Actually, the NFA can also be defined as follow:



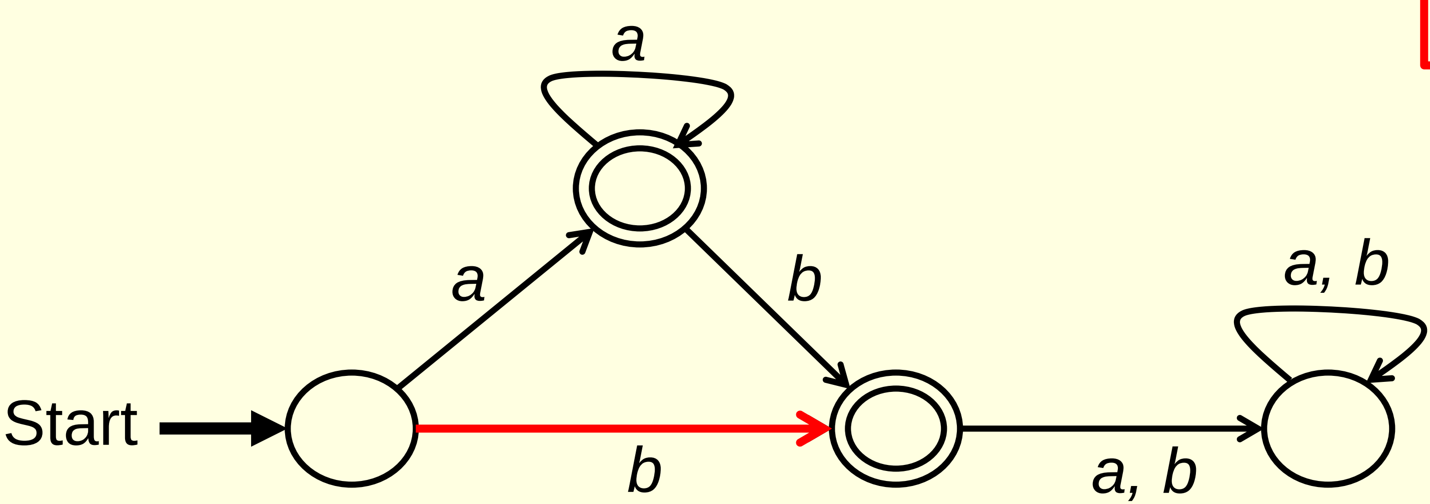
A = { d, g, m, o, r}

Example. NFA for the language of $a + aa^*b + a^*b$.



DFA for the language of $a + aa^*b + a^*b$.

a, b, ab, aab, aaab, …



DFA and NFA are equivalent concept.

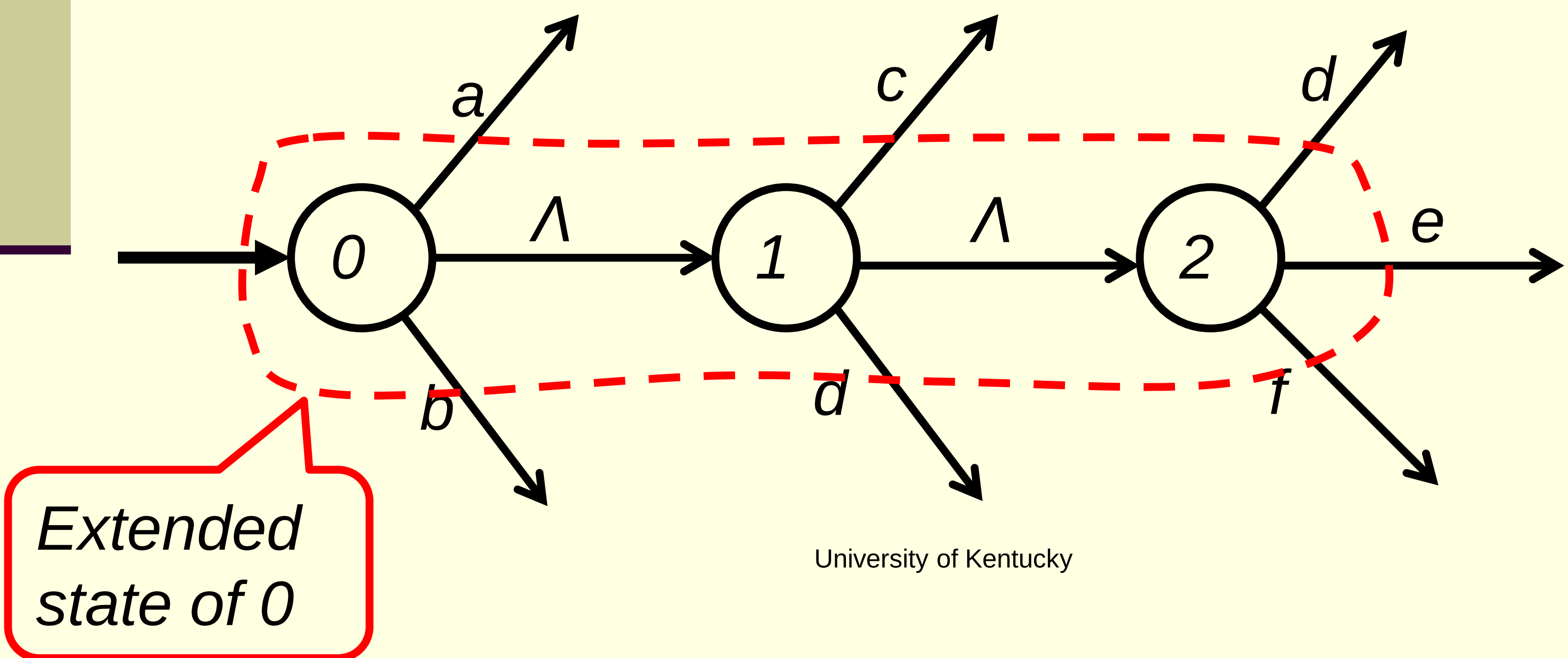
A DFA is also an NFA.

But actually for every NFA there's a corresponding DFA that accepts the same language, but if the NFA has n states, the DFA could have $O(2^n)$ states.

So we work with NFAs because they're usually a lot smaller than DFAs.

A few points about NFA's.

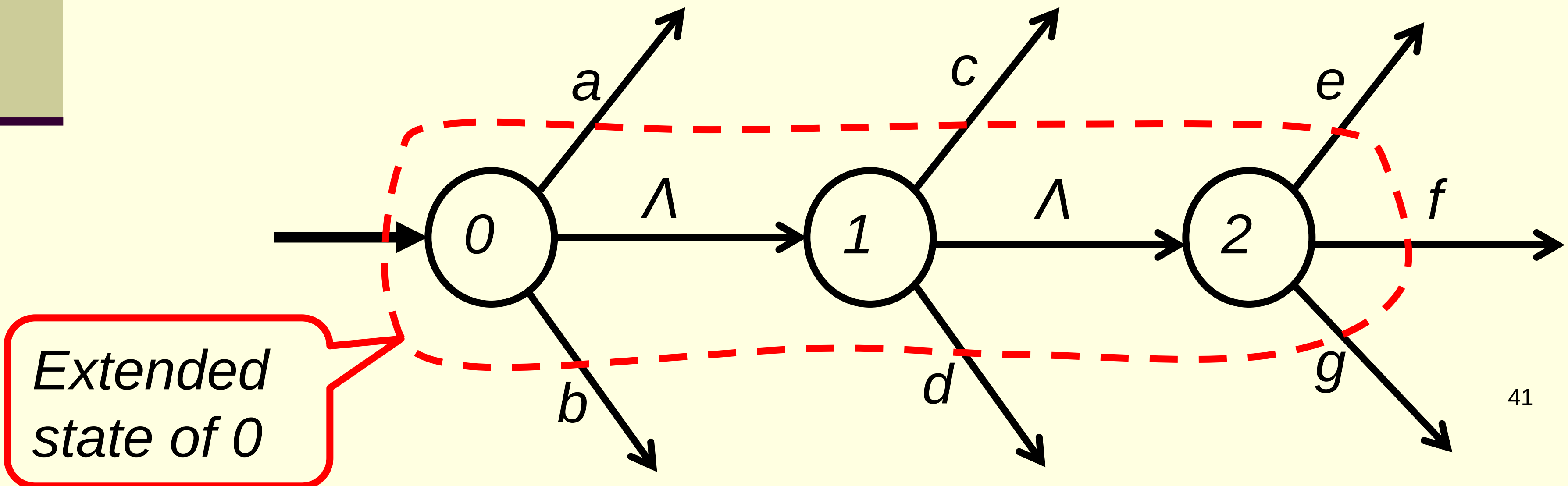
The existence of Λ-edges implicitly creates the concept of an extended state (a multiple-node state) of a given state.



Edges a, b, c, d, e, f, g are edges of the extended state of 0.

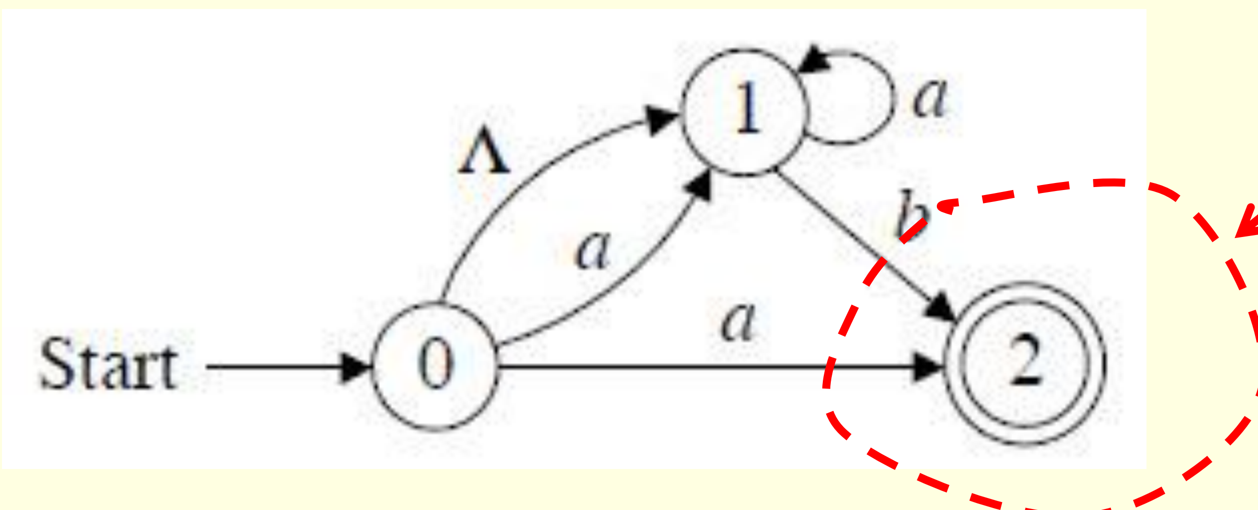
Each edge of the extended state of 0 can be used by state 0 through some Λ-edges.

So essentially state 0 has 7 edges to use even though there are only 2 real edges emitted from state 0.



For an NFA, only edges really needed for its function have to be designed.

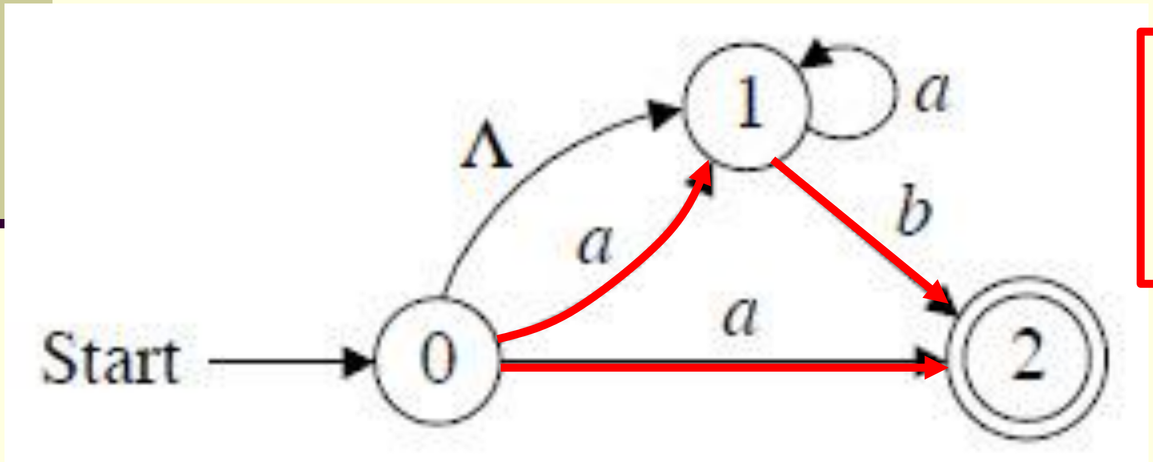
NFA for the language of $a + aa^*b + a^*b$.



We don't need to design any edges for state 2.

Why do we need "multiple edges with the same letter"?

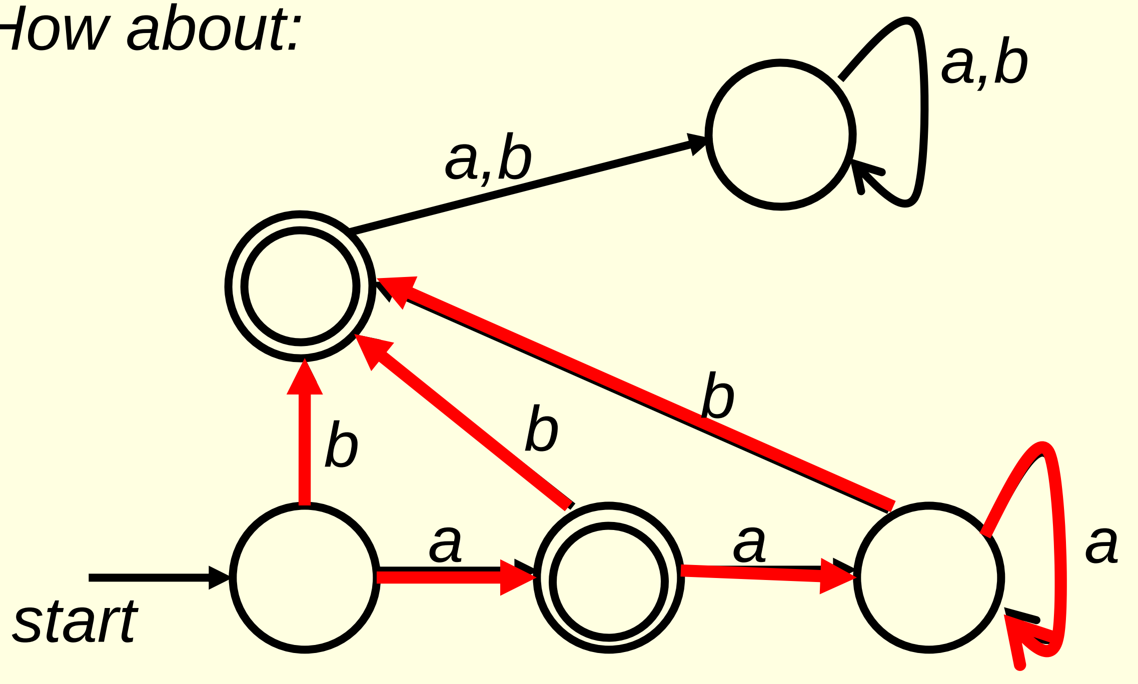
NFA for the language of *a* + *aa*b* + *a*b*.



A letter can be used as the lead symbol for disjoint paths

Question: can you think of a DFA that would recognize *a+ aa*b + a*b* ?

How about:



a b ab aab aaab

6. Regular Languages & Finite Automata

- Finite Automata

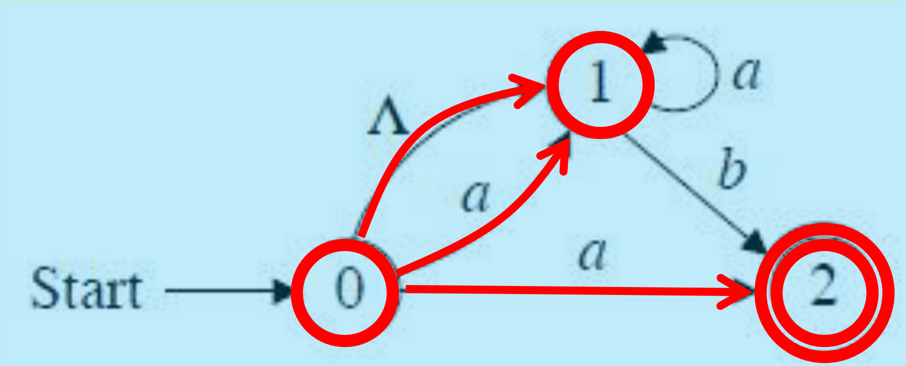
Table representation of NFA

An NFA over *A* can be represented by a function

$$T : \underline{States \times A \cup \{\Lambda\}} \rightarrow \underline{power(States)},$$

where *T(i, a)* is the set of states reached from state *i* along the edge(s) labeled *a*, and we mark the start and final states.

Example:



	T	a	b	Λ
start	0	{1, 2}	∅	{1}
	1	{1}	{2}	∅
final	2	∅	∅	∅

6. Regular Languages & Finite Automata

– Finite Automata

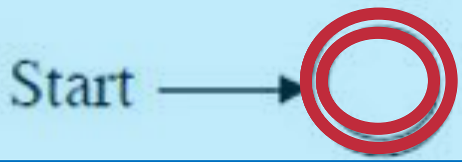
the proof is similar to Kleene's theorem

Theorem (Rabin and Scott): The class of regular languages is exactly the same as the class of languages accepted by NFAs.

Questions. Find an NFA for each of the languages over *{a, b}*.

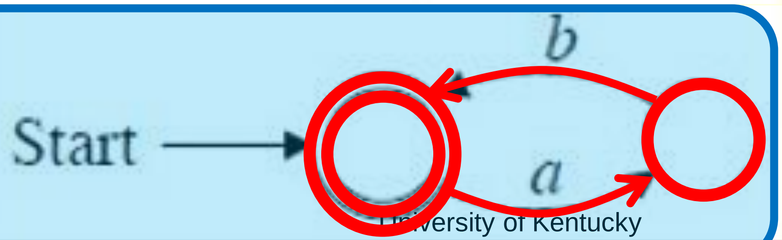
(a) $\emptyset$

(b) $\{\Lambda\}$

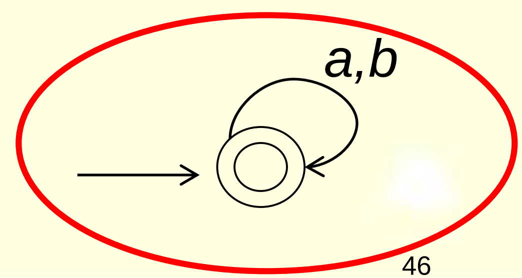


$\{a^*, b^*, (ab)^*\}$

(c) $(ab)^n$



= ?



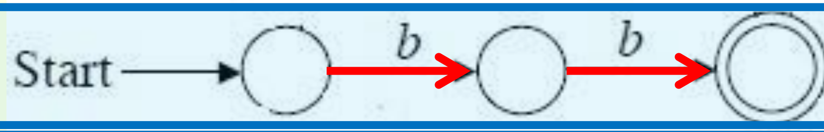
6. Regular Languages & Finite Automata

– Finite Automata

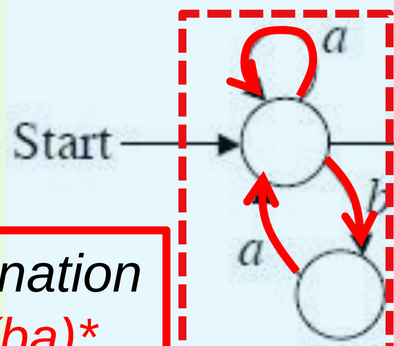
Example. Find an NFA to recognize $(a + ba)^*bb(a + ab)^*$.

A solution:

$\{bb\}$



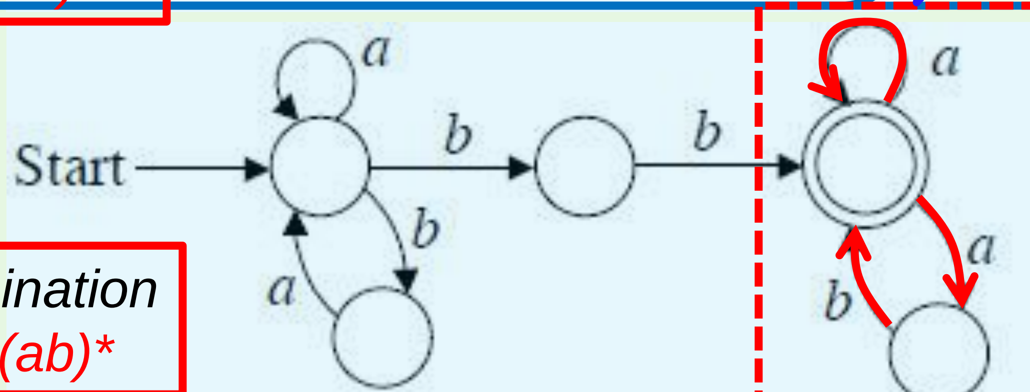
$\{(a+ba)^*bb\}$



any combination of a and (ba)**

OK for an NFA, not for a DFA

$\{(a+ba)^*bb(a+ab)^*\}$



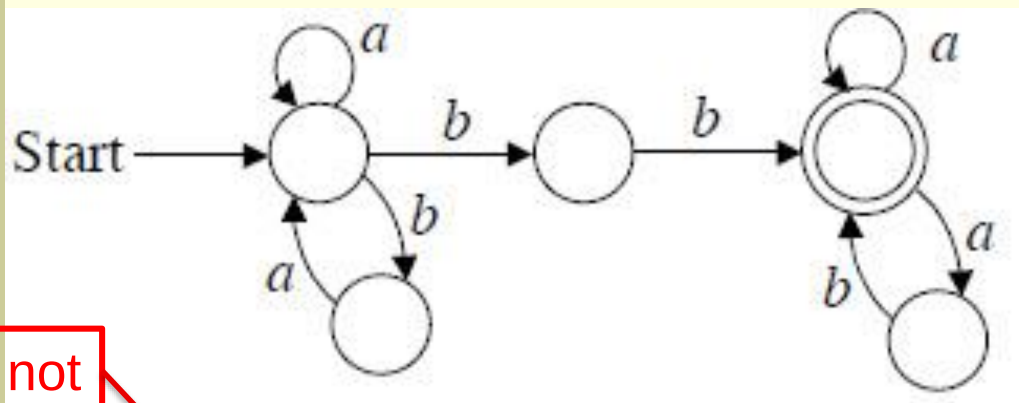
any combination of a and (ab)**

6. Regular Languages & Finite Automata

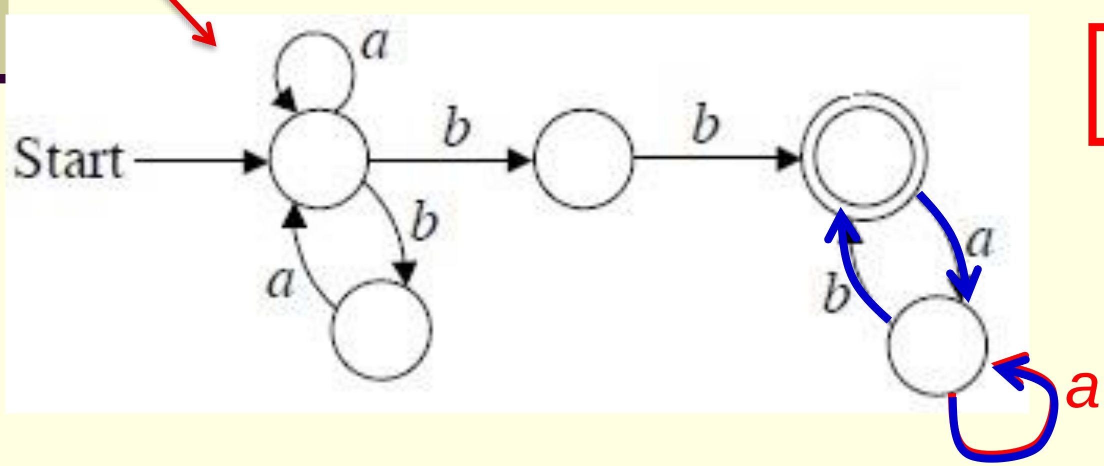
- Finite Automata

Example (conti). Find an NFA to recognize *(a + ba)*bb(a + ab)**.

A solution:



But not



$$\{(a+ba)^* bb(aa^*b)^*\}$$

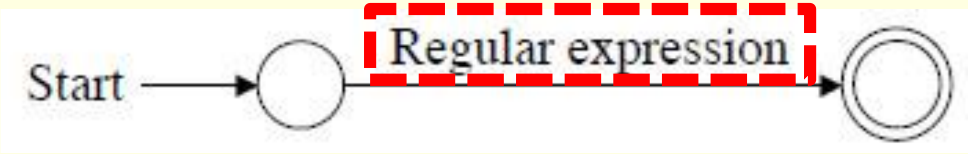
6. Regular Languages & Finite Automata

- Finite Automata

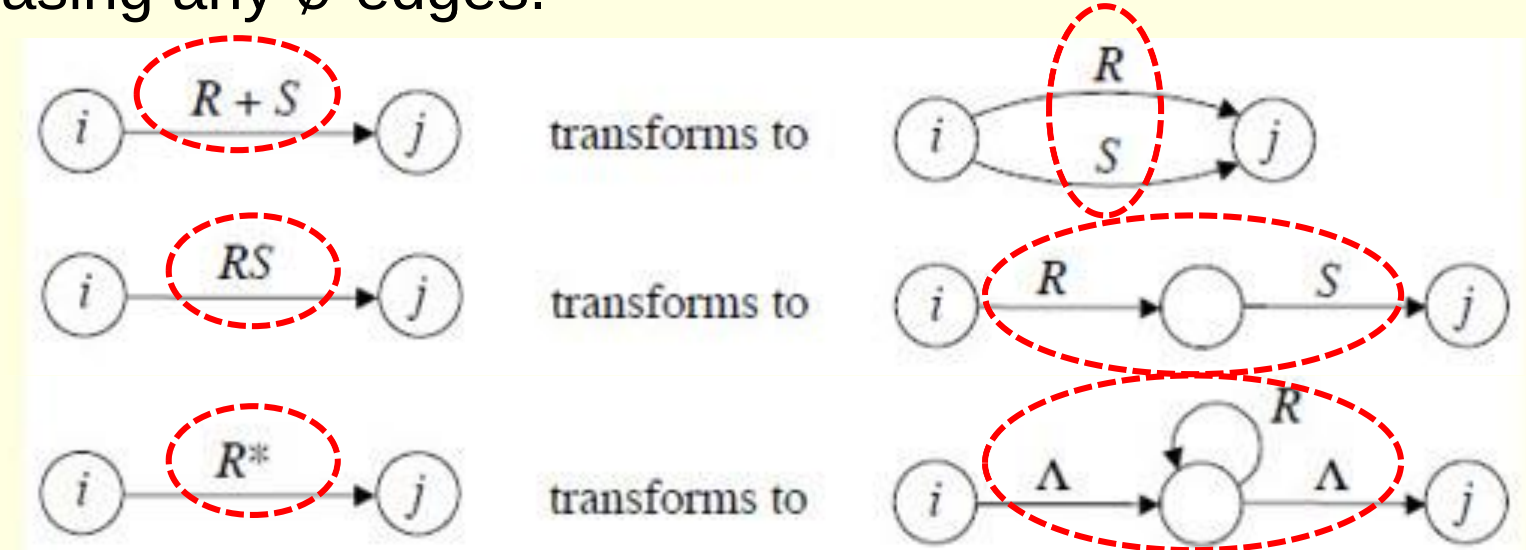
(DFA or NFA)

Algorithm: Transform a *Regular Expression (RE)* into a *Finite Automaton*

(1) Placing the RE on the edge between a start and final state:



(2) Apply the following rules to obtain a finite automaton after erasing any ∅-edges.



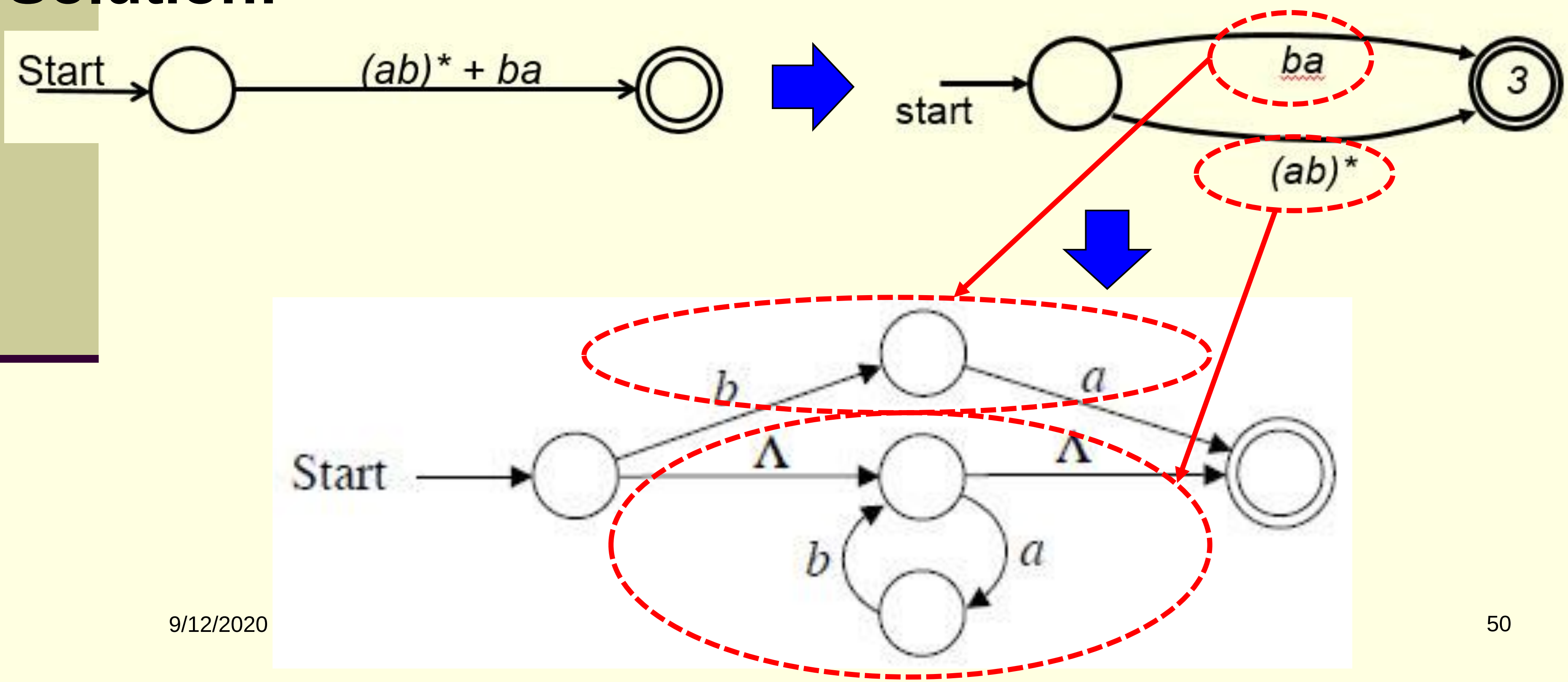
6. Regular Languages & Finite Automata

- Finite Automata

Example. Use the algorithm to construct a finite automaton for

$(ab)^* + ba.$

Solution:

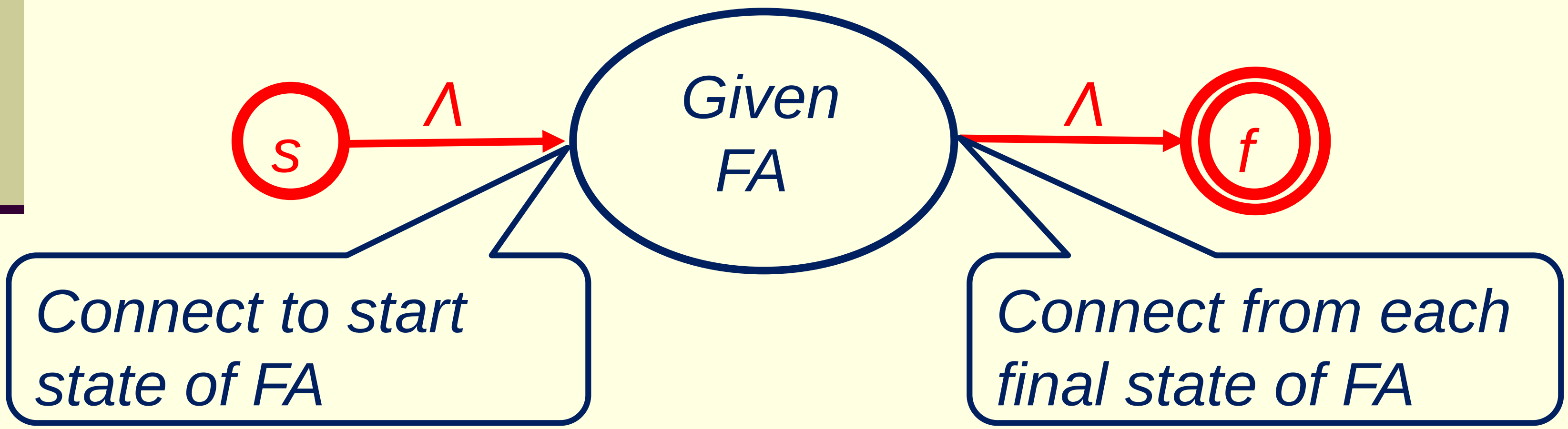


6. Regular Languages & Finite Automata

- Finite Automata

Algorithm: Transform a Finite Automaton to a Regular Expression

Connect a new start state *s* to the start state of the FA and connect each final state of the FA to a new final state *f* as shown in the figure.

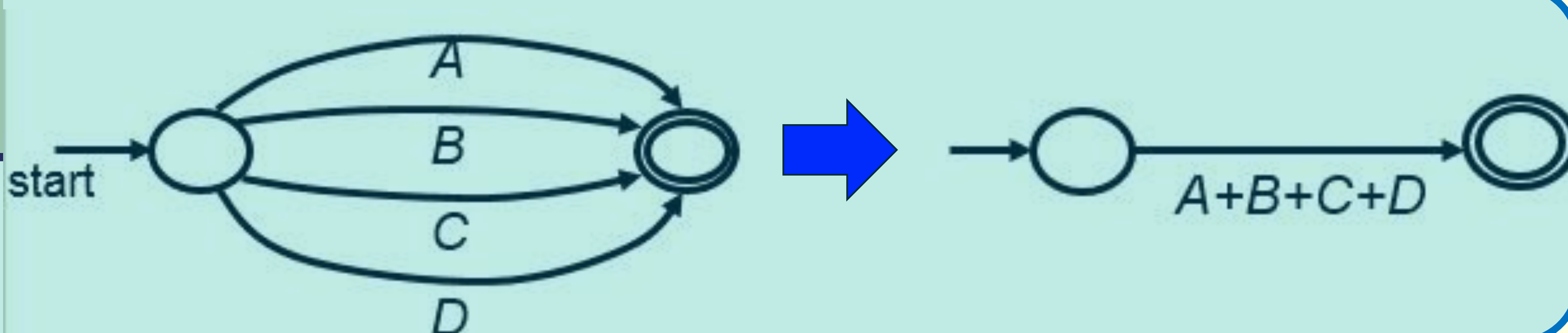


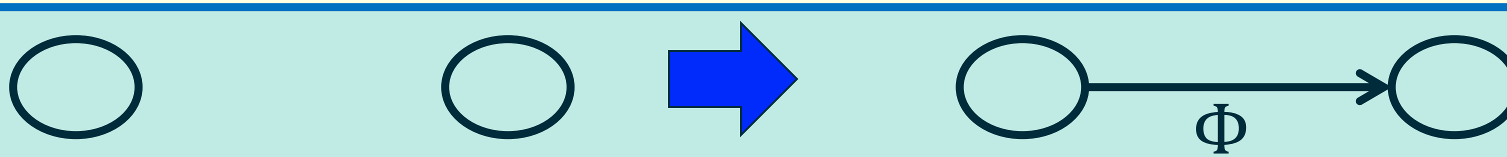
6. Regular Languages & Finite Automata

- Finite Automata

If needed, combine all multiple edges between the same two nodes into one edge with label the sum of the labels on the multiple edges.

If there is no edge between two states, assume there is an ∅-edge.





6. Regular Languages & Finite Automata

- Finite Automata

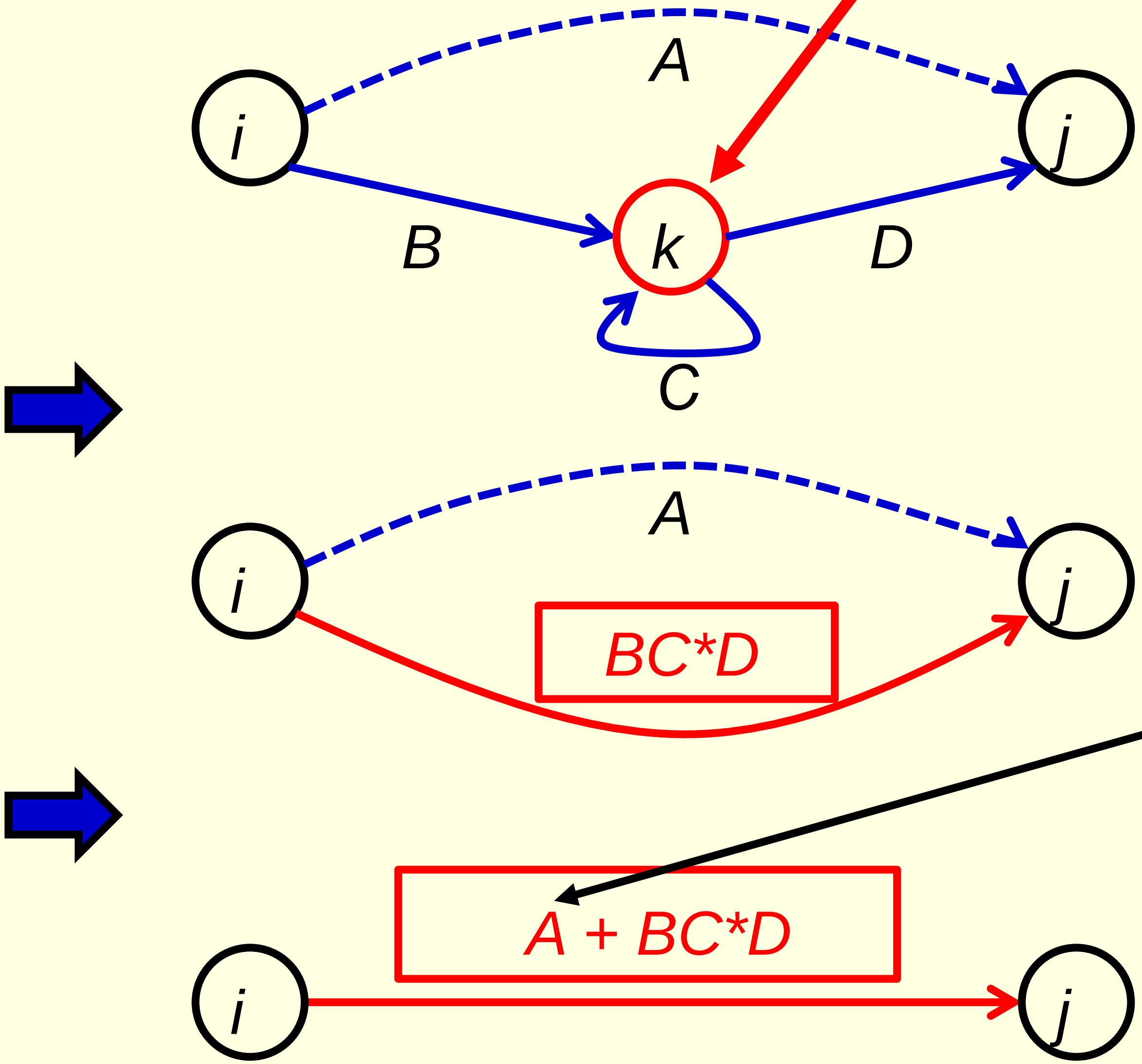
Now eliminate each state k of the FA by constructing a new edge (i, j) for each pair of edges (i, k) and (k, j) where $i \neq k$ and $j \neq k$.

New label $new(i, j)$ is defined as follows:

$$\boxed{new(i, j)} = \boxed{old(i, j)} + \boxed{old(i, k)\ old(k, k)^*\ old(k, j)}$$

Example:

Think of the process of eliminating state k as a two-step procedure:

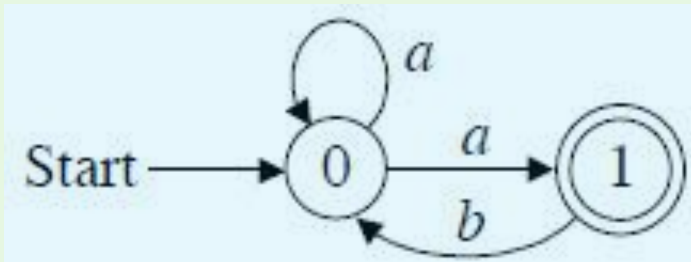


Put Φ here if there was not a direct edge between i and j originally

6. Regular Languages & Finite Automata

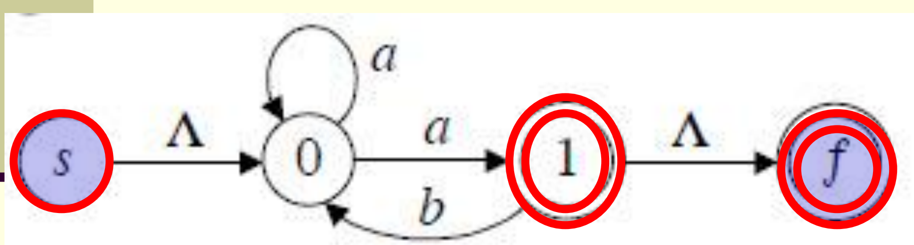
- Finite Automata

Example. Transform the following NFA into a regular expression.



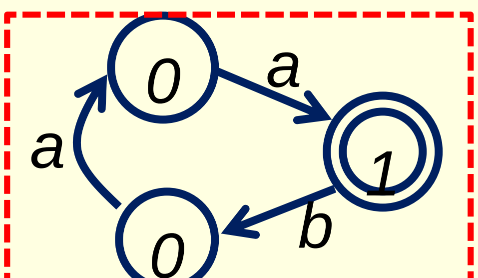
$a^*a(ba^*a)^*$

Solution I (eliminate state 1 first):



State 1 is considered a state between state 0 and state f

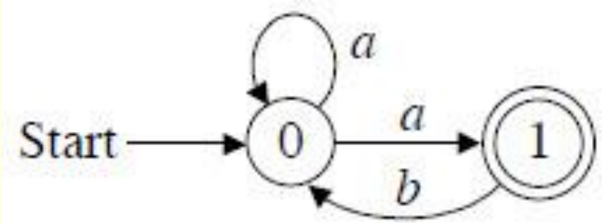
State 1 is also considered a state between state 0 and state 0



6. Regular Languages & Finite Automata

- Finite Automata

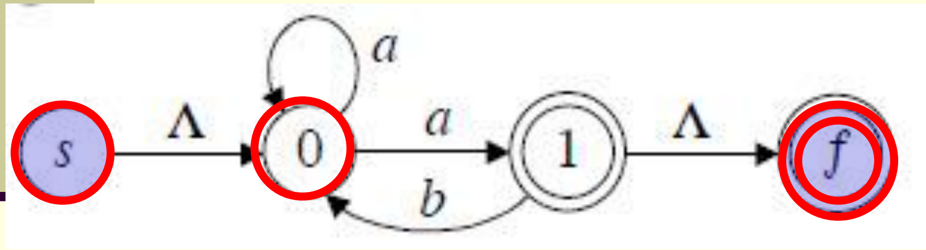
Example. Transform the following NFA into a regular expression.

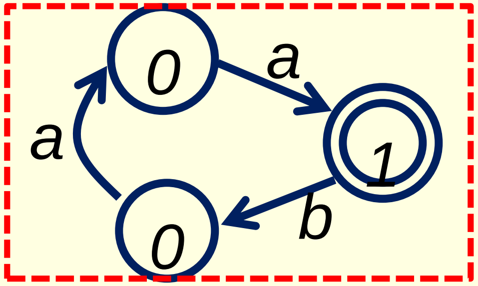


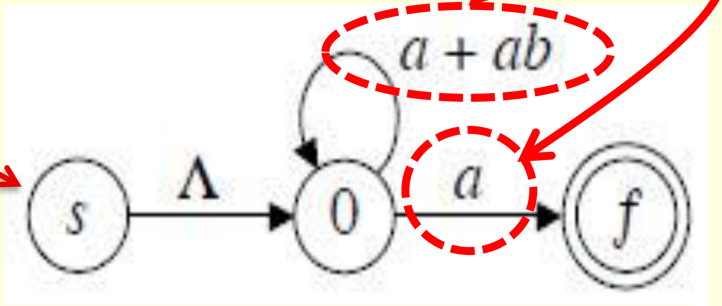
Solution I (eliminate state 1 first):

$$\text{new}(0, f) = \varnothing + a\varnothing^{*}\Lambda = a$$

$$\text{new}(0,0) = a + a\varnothing^{*}b = a + ab$$



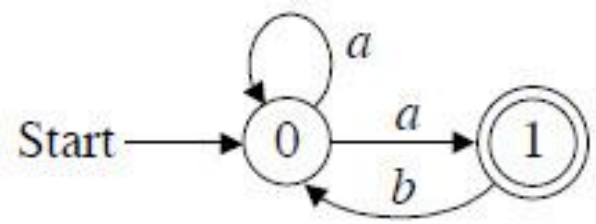




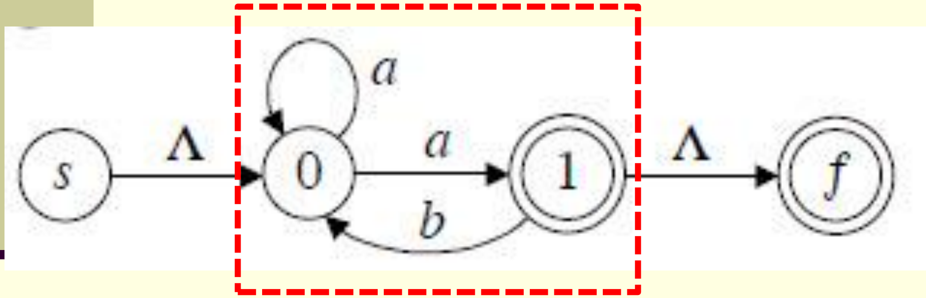
6. Regular Languages & Finite Automata

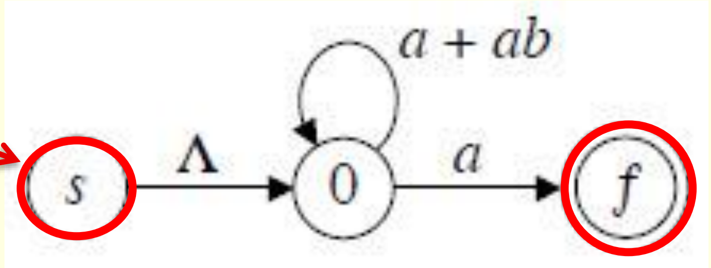
- Finite Automata

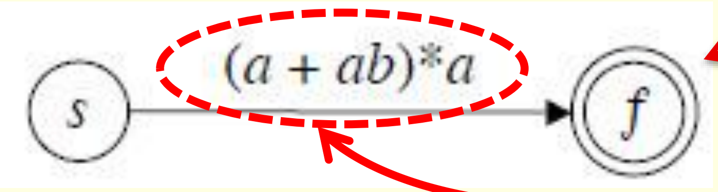
Example. Transform the following NFA into a regular expression.



Solution I (eliminate state 1 first):

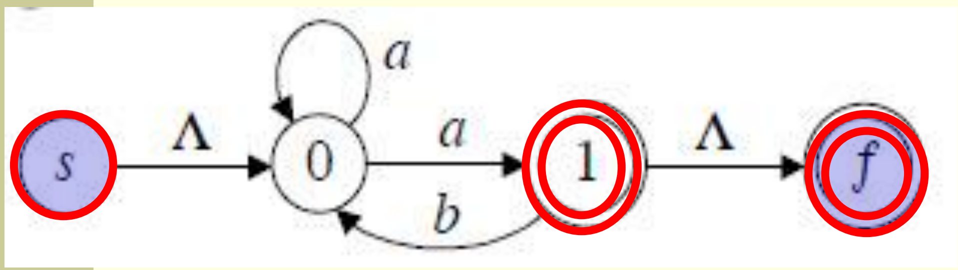




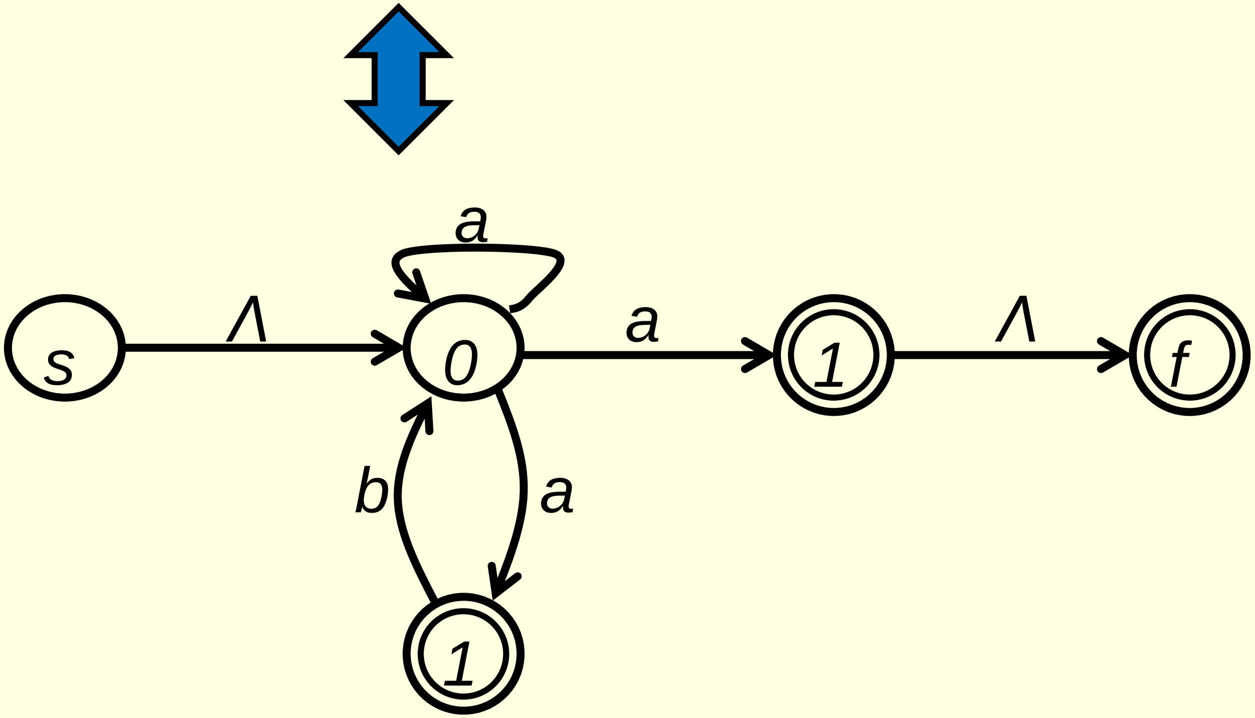


$$\text{new}(s, f) = \varnothing + \Lambda(a+ab)^{*}a = (a+ab)^{*}a$$

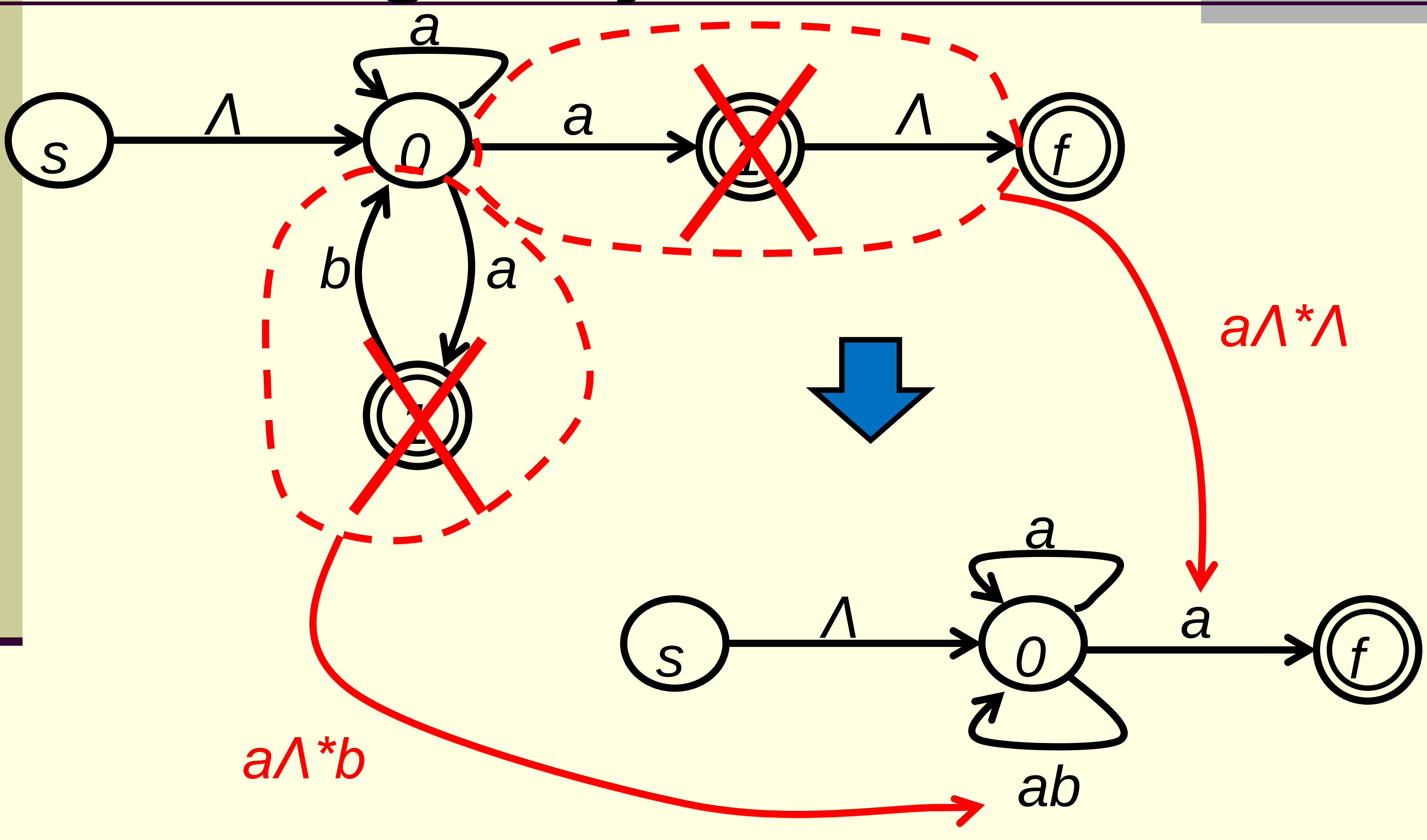
Or, eliminate state 1 first, the following way:



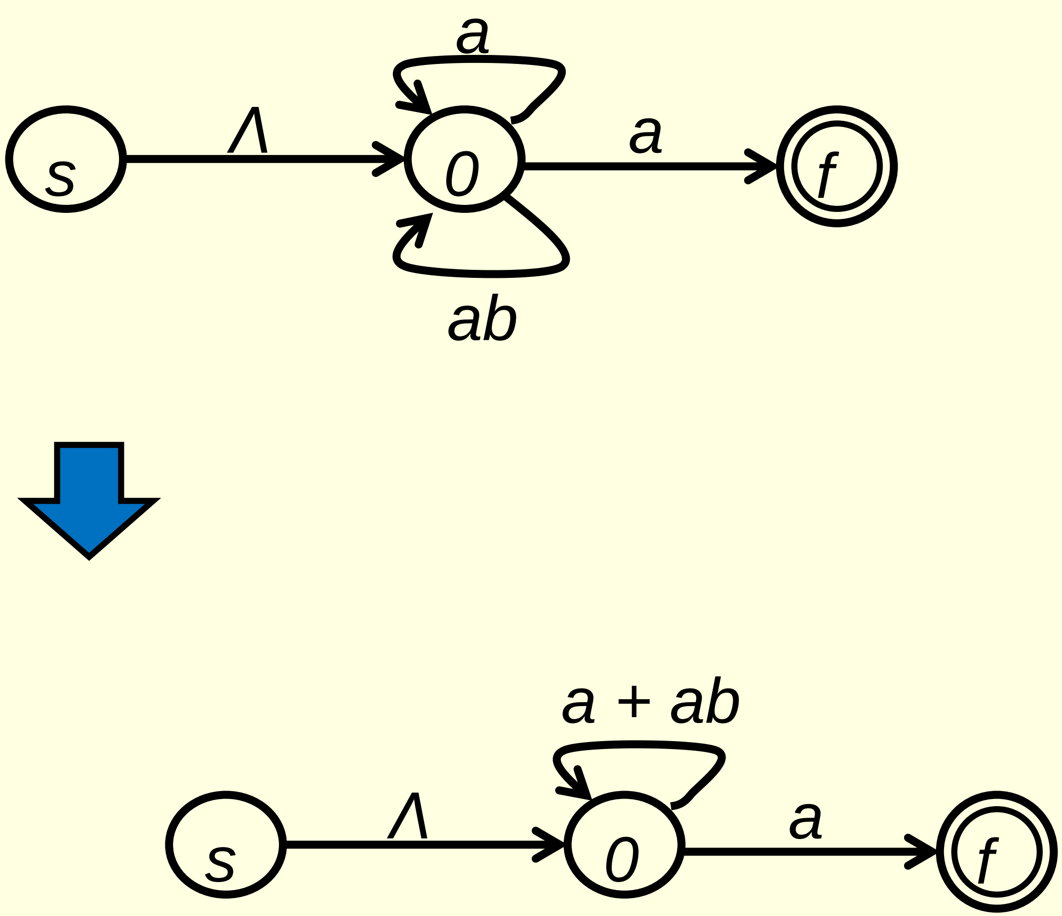
You can think of the given NFA as an NFA of the following form



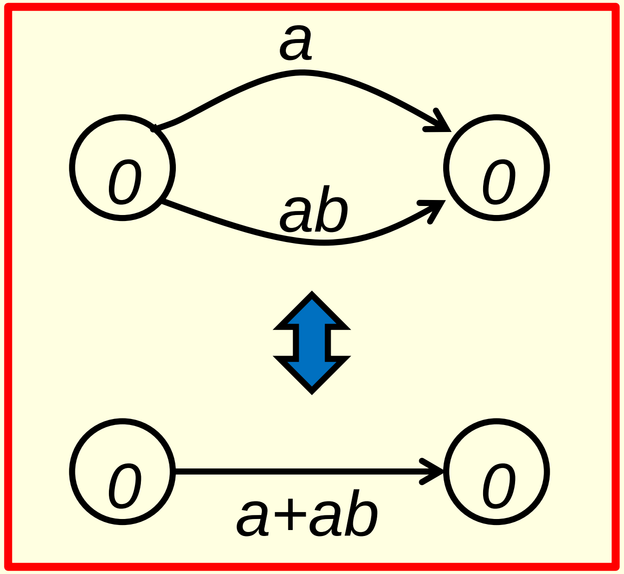
Or, eliminate state 1 first, the following way:



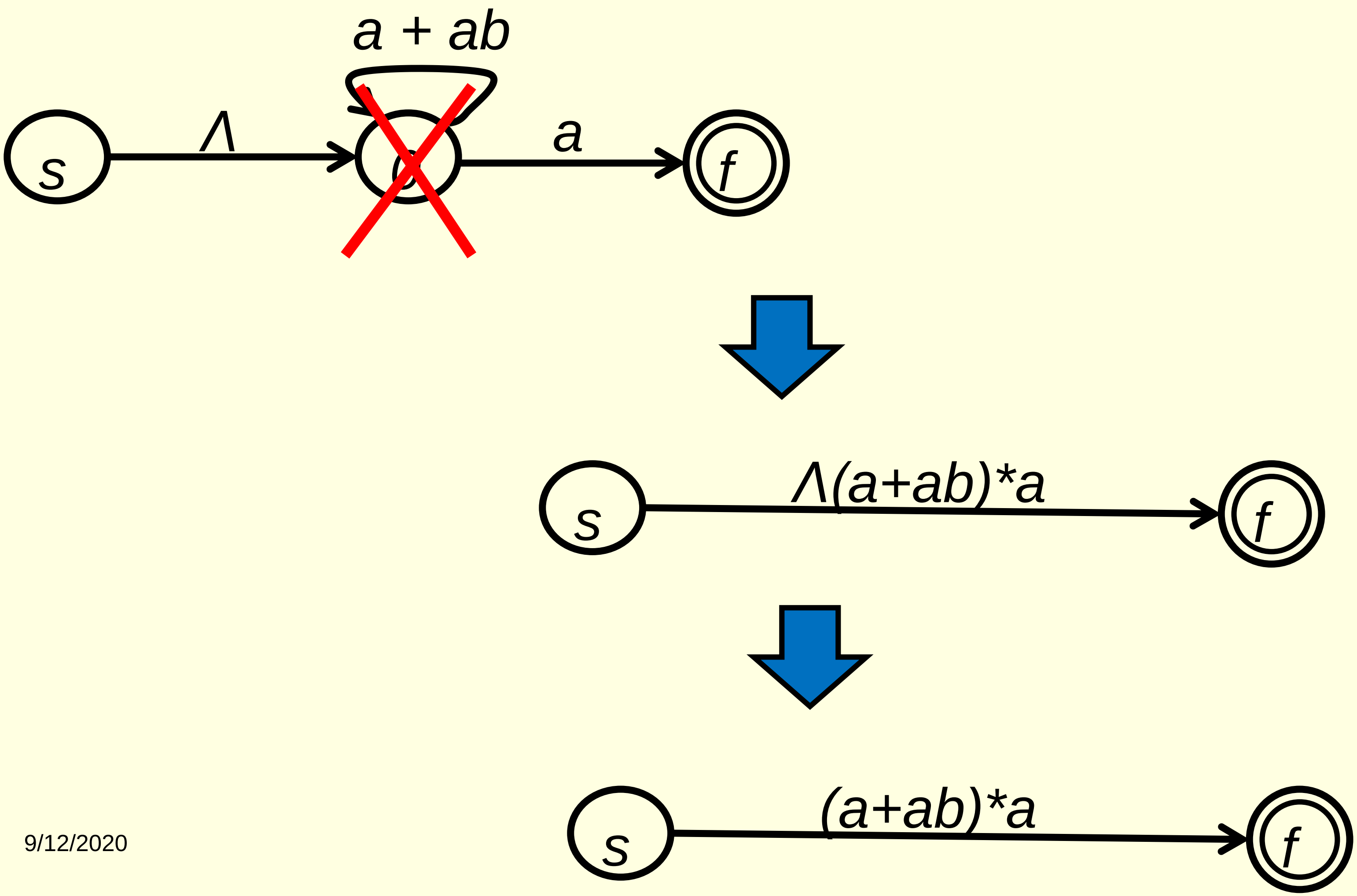
Or, eliminate state 1 first, the following way:



Why?



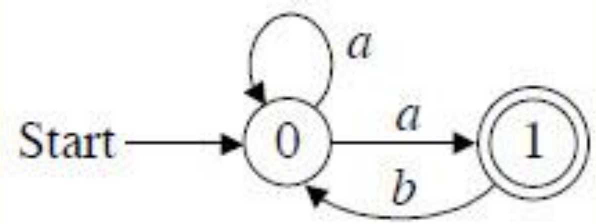
Or, eliminate state 1 first, the following way:



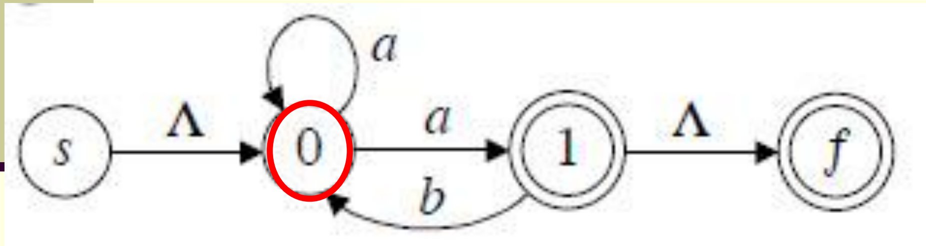
6. Regular Languages & Finite Automata

- Finite Automata

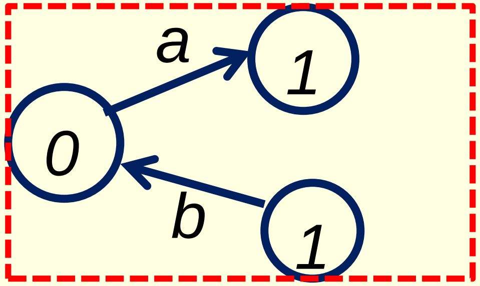
Example. Transform the following NFA into a regular expression.



Solution II (eliminate state 0 first):



State 0 is considered a state between state s and state 1

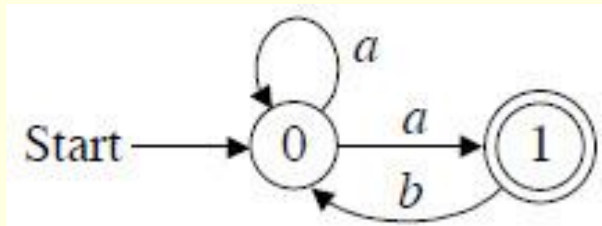


State 0 is also considered a state between state 1 and state 1

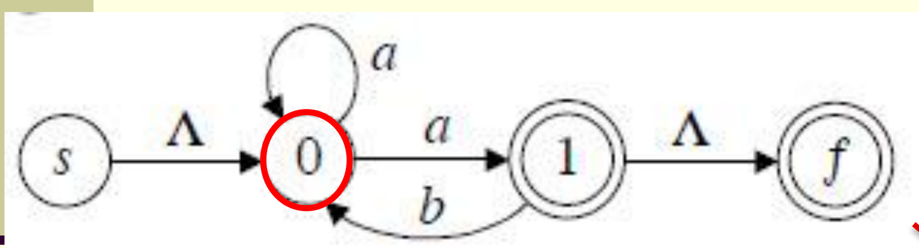
6. Regular Languages & Finite Automata

- Finite Automata

Example. Transform the following NFA into a regular expression.

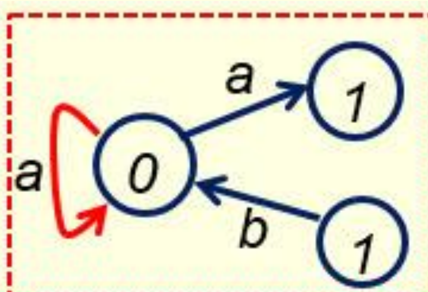


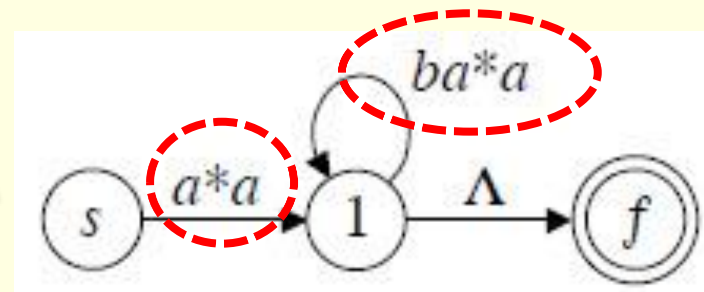
Solution II (eliminate state 0 first):



$$\boxed{\text{new}(s,1) = \underline{\varnothing} + \underline{\Lambda}\,\underline{a}^*\underline{a} = a^*a}$$

$$\boxed{\text{new}(1,1) = \underline{\varnothing} + \underline{b}\,\underline{a}^*\underline{a} = ba^*a}$$

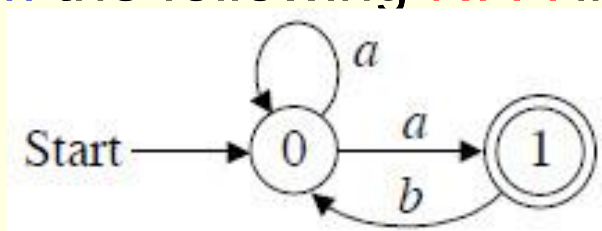




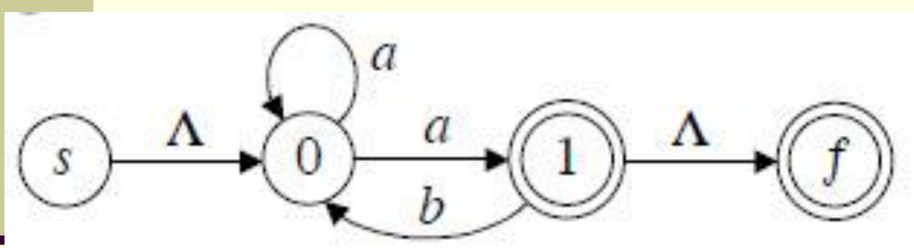
6. Regular Languages & Finite Automata

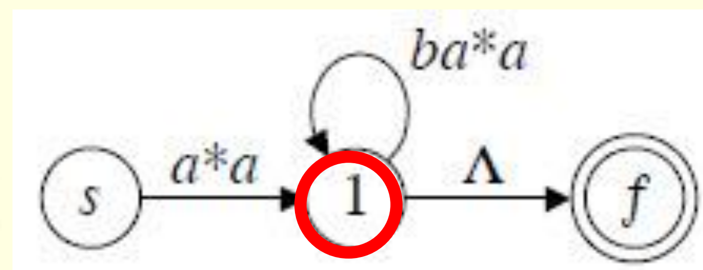
- Finite Automata

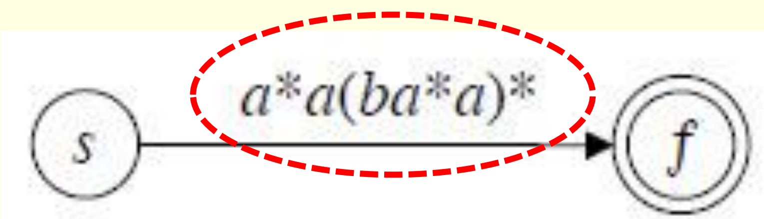
Example. Transform the following NFA into a regular expression.



Solution II (eliminate state 0 first):

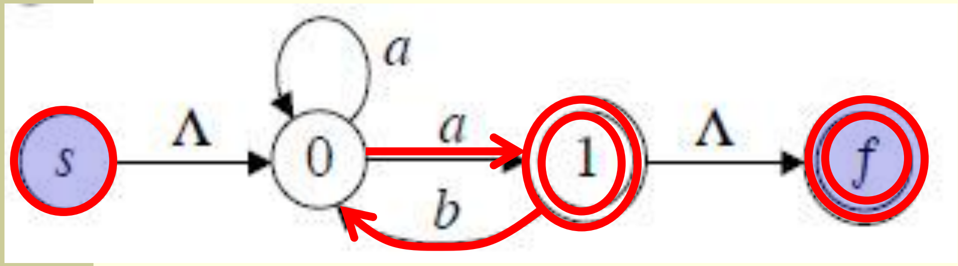




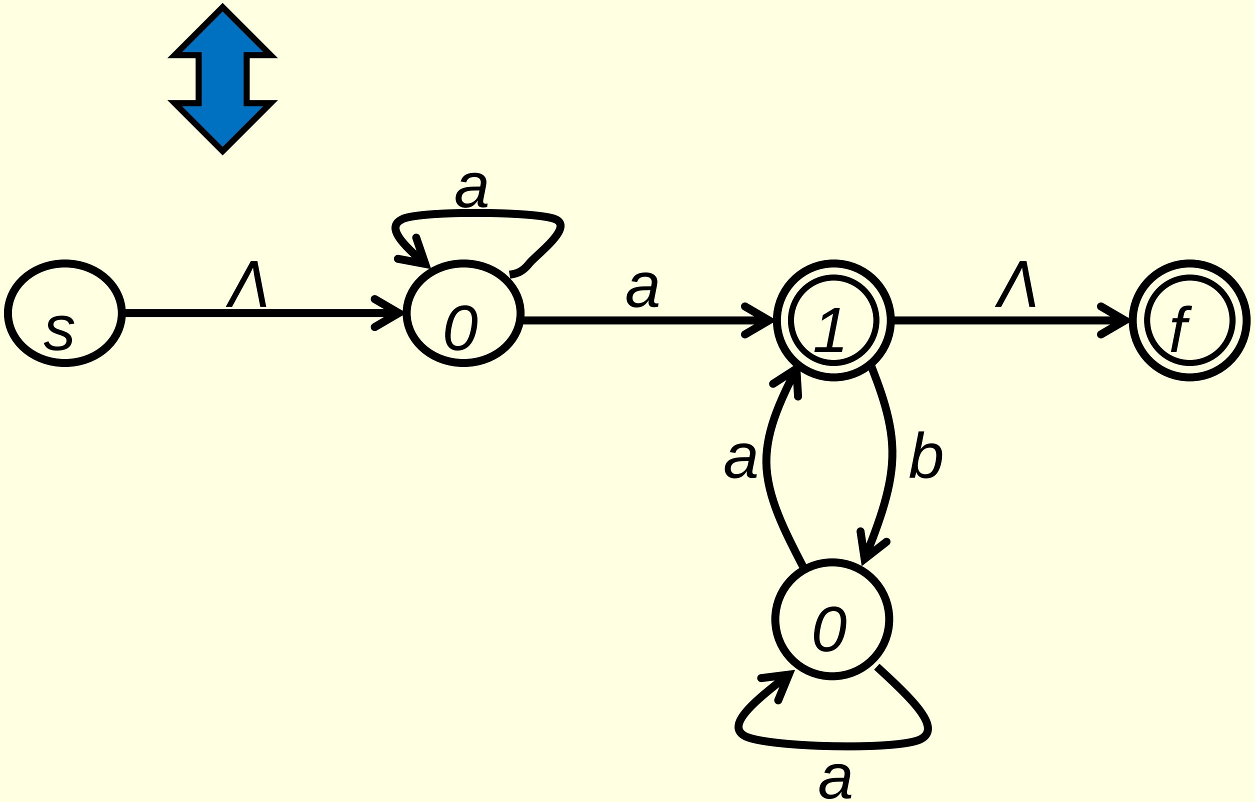


$$\boxed{\text{new}(s, f) = \varnothing + a^* a(ba^* a)^* \Lambda = a^* a(ba^* a)^*}$$

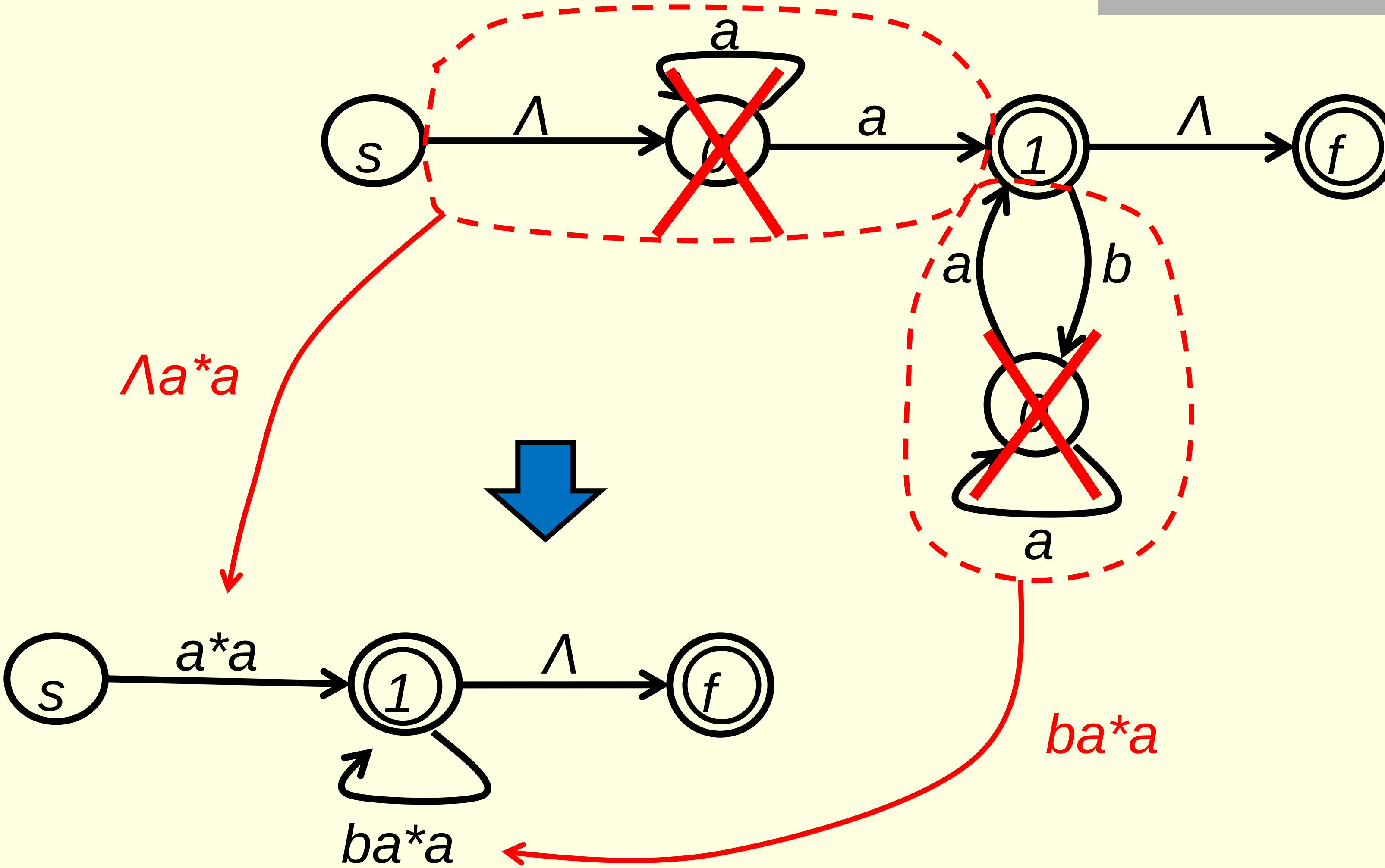
Or, eliminate state 0 first, the following way:



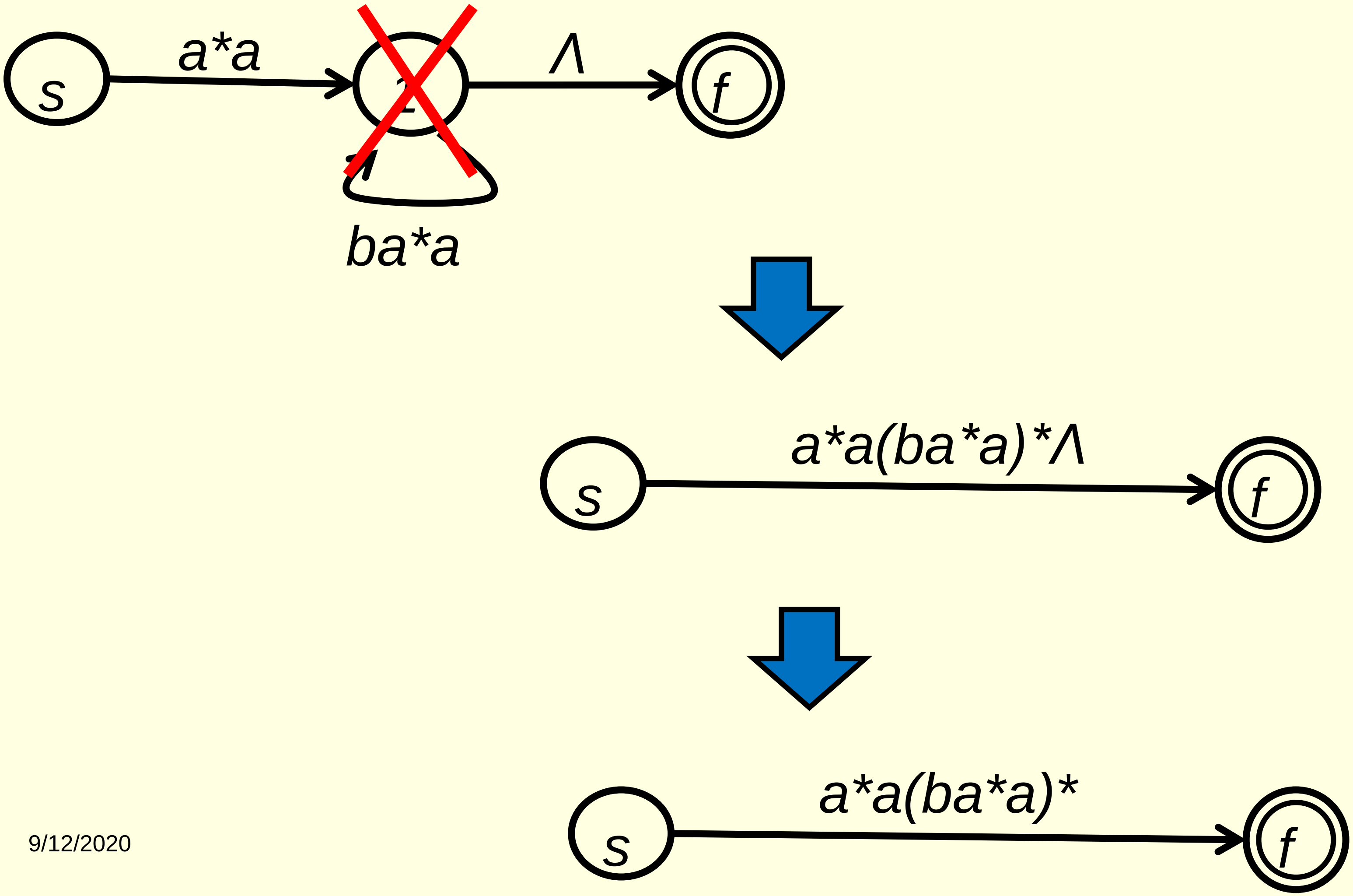
You can think of the given NFA as an NFA of the following form



Or, eliminate state 0 first, the following way:



Or, eliminate state 0 first, the following way:



6. Regular Languages & Finite Automata

- Finite Automata

Note. The two regular expressions obtained in the previous example are equal, i.e., $a^*a(\boxed{ba^*}a)^* = (a + ab)^*a.$

Proof I.

$$(a + ab)^* a$$

$$= [\, a^* (\, (ab)\, a^*)^* \,]\, a$$

$$= a^* [\, (\, \boxed{(a\, b)\, a^*})^*\, a \,]$$

$$= a^* [\, (\, \boxed{a\, (\, b\, a^*)})^*\, a \,]$$

$$= a^* [\, a\, ((\, ba^*)\, a\,)^* \,]$$

$$= a^*\, a\, (ba^*\, a)^*$$

$$(R + S)^* = R^* (S R^*)^*$$

$\cdot$ is associative

$\cdot$ is associative

$$(R S)^* R = R (S R)^*$$

$\cdot$ is associative

6. Regular Languages & Finite Automata

- Finite Automata

Note. The two regular expressions obtained in the previous example are equal, i.e., $a^*a(ba^*a)^* = (a + ab)^*a.$

Proof II.

$a^*\, a\, (\underline{ba^*a})^* = a^*[a((\underline{ba^*})\, a)^*]$ · is associative

$= a^*[(\underline{a}\, (\underline{ba^*}))^*\underline{a}\,]$ $\boxed{R(SR)^* = (RS)^*R}$

$= a^*[\, (\, (\underline{ab})a^*\,)^*\, a]$ · is associative

$= \boxed{[\underline{a^*}\, ((\underline{ab})\, \underline{a^*})^*\,]}\, a$ · is associative

$= (\underline{a} + \underline{ab})^*a$ $\boxed{R^*(SR^*)^* = (R + S)^*}$

QED.

6. Regular Languages & R

- Finite Automata

any combinations of
a^* and $(ab)^*$

Note. The two regular expressions obtained in the previous example are equal, i.e., $a^*a(ba^*a)^* = \underline{(a + ab)^*}a.$

Intuitive Proof.

$LHS = a^*\,a\,(\,ba^*a\,)^* = a^*a\underline{(ba^*a)(ba^*a)(ba^*a) \cdots (ba^*a)}$

$RHS = (a + ab)^*a = \underline{a^*(ab)^*a^*(ab)^* \cdots a^*(ab)^*a}$

LHS ⊆ RHS   Why?

$a^*a\underline{(ba^*a)(ba^*a)(ba^*a)}\ \ \epsilon\ \ LHS$

$= a^*(ab)a^*(ab)a^*(ab)a^*a$

$= \underline{a^*(ab)a^*(ab)a^*(ab)a^*(ab)^0a}\ \ \epsilon\ \ RHS$

Hence, LHS ⊆ RHS

6. Regular Languages & Finite Automata

- Finite Automata

Note. The two regular expressions obtained in the previous example are equal, i.e., $a^*a(ba^*a)^* = (a + ab)^*a$.

Intuitive Proof (conti).

$LHS = a^* a (ba^*a)^* = \underline{a^*a(ba^*a)(ba^*a)(ba^*a) \cdots (ba^*a)}$

$RHS = (a + ab)^*a = a^*(ab)^*a^*(ab)^* \cdots a^*(ab)^*a$

$RHS \subseteq LHS$ *Why?*

$a^*(ab)^2a^*(ab)^2a^*(ab)^2a \; \epsilon \; RHS$

$= \underline{a^*}\,\underline{(ab)}\,\underline{(ab)a^*}\,\underline{(ab)}\,\underline{(ab)a^*}\,\underline{(ab)}\,\underline{(ab)a}$

$= a^*a(ba)(ba^*a)(ba)(ba^*a)(ba)(ba)$

$= \underline{a^*a(ba^0a)(ba^*a)(ba^0a)(ba^*a)(ba^0a)(ba^0a)} \; \epsilon \; LHS$

Hence, $RHS \subseteq LHS$

End of Regular Languages and Finite Automata II

Regular expression of this NFA:

*a*a(ba*a)**