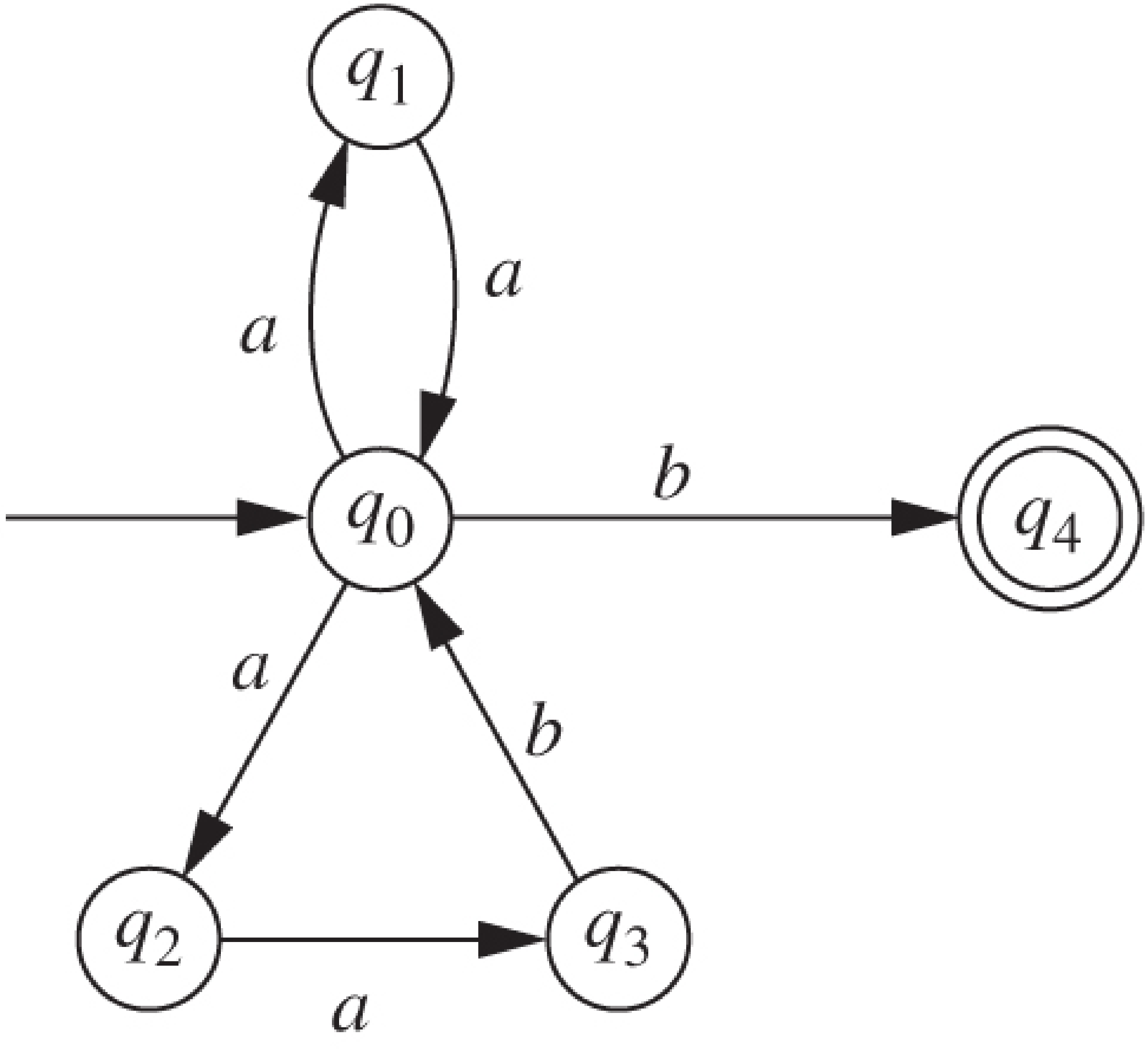
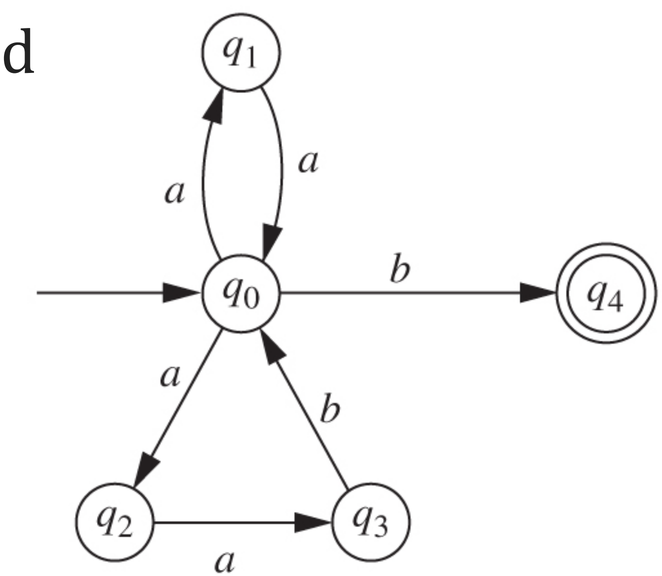
Nondeterministic Finite Automata


q_1
a
a
q_0
b
q_4
a
b
q_2
q_3
a

Nondeterministic Finite Automata

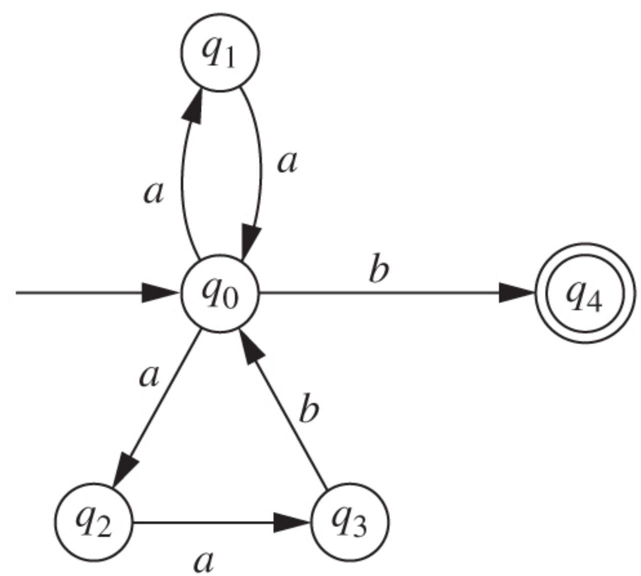
- This NFA closely resembles the regular expression **(*aa* + *aab*)****b***
  - The top loop is *aa*
  - The bottom loop is *aab*
  - By following the links we can generate any string in the language
- This is not the transition diagram for an FA; some nodes have more than one *a*-arc, some have none
- Example: *aaaabaab* can be either accepted (top-bottom-top-b) or

not accepted (top-bottom-bottom loops).



Nondeterministic Finite Automata

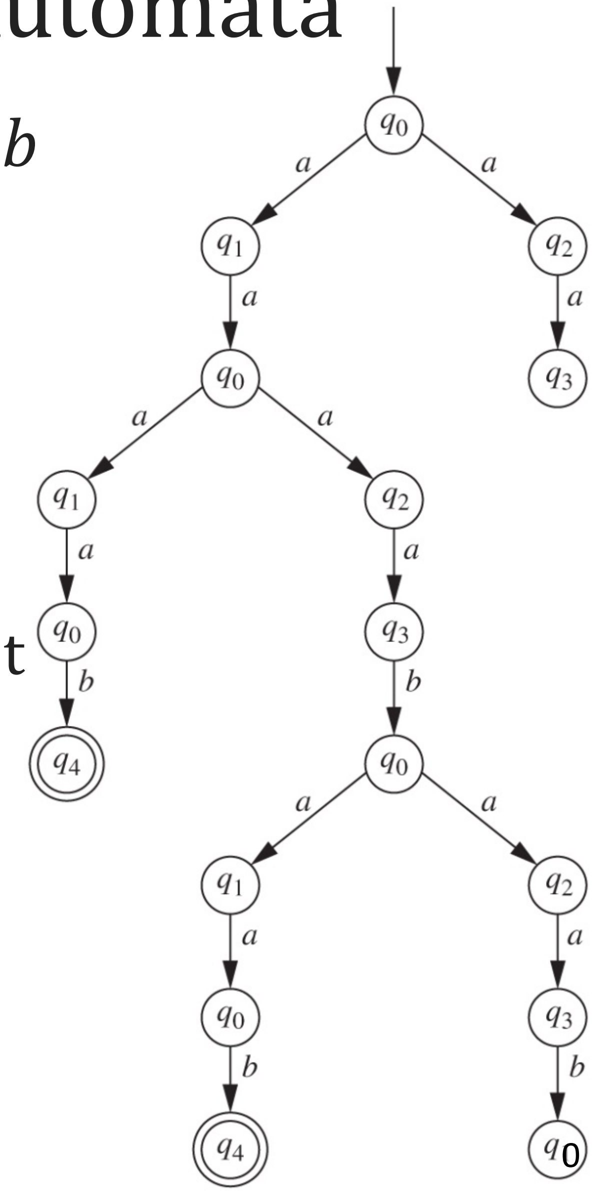
- For this reason, we should **not** think of an NFA as describing an algorithm for recognizing a language
- Instead, consider it as describing a number of different sequences of steps that *might* be followed

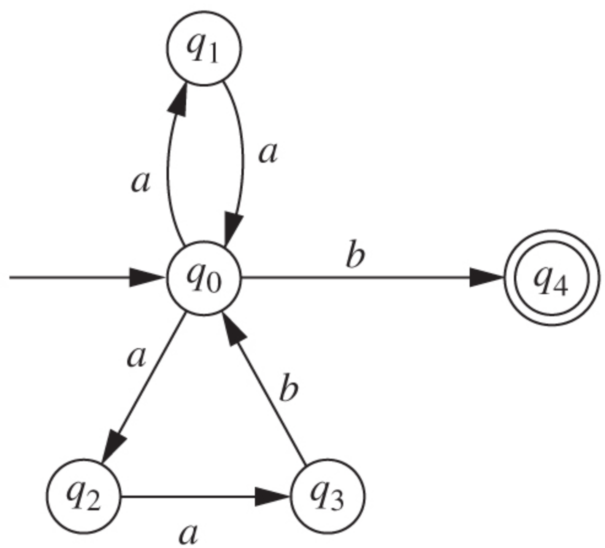


Nondeterministic Finite Automata

This is the "computation tree" for *aaaabaab*

- Each level corresponds to a prefix of the input string
- Each state on a level is one the machine *could* be in after processing that prefix
- There is an accepting path for the input string (as well as other paths that are not accepting)

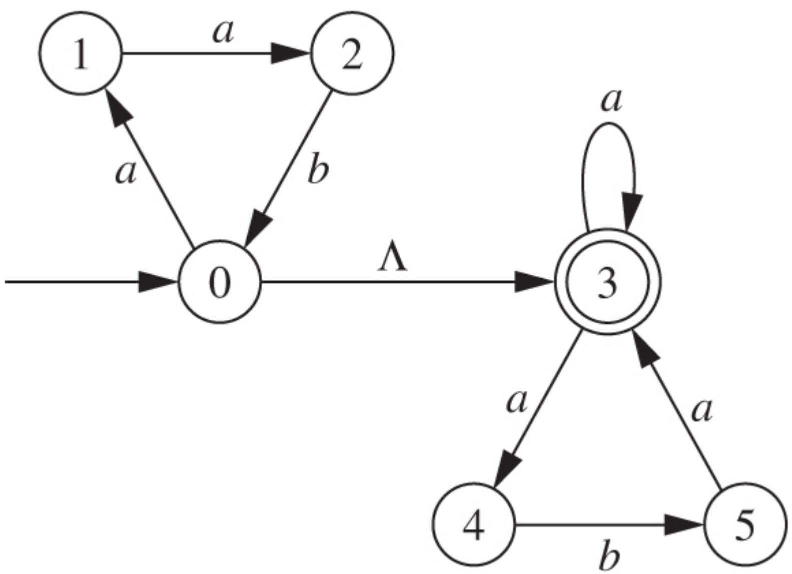


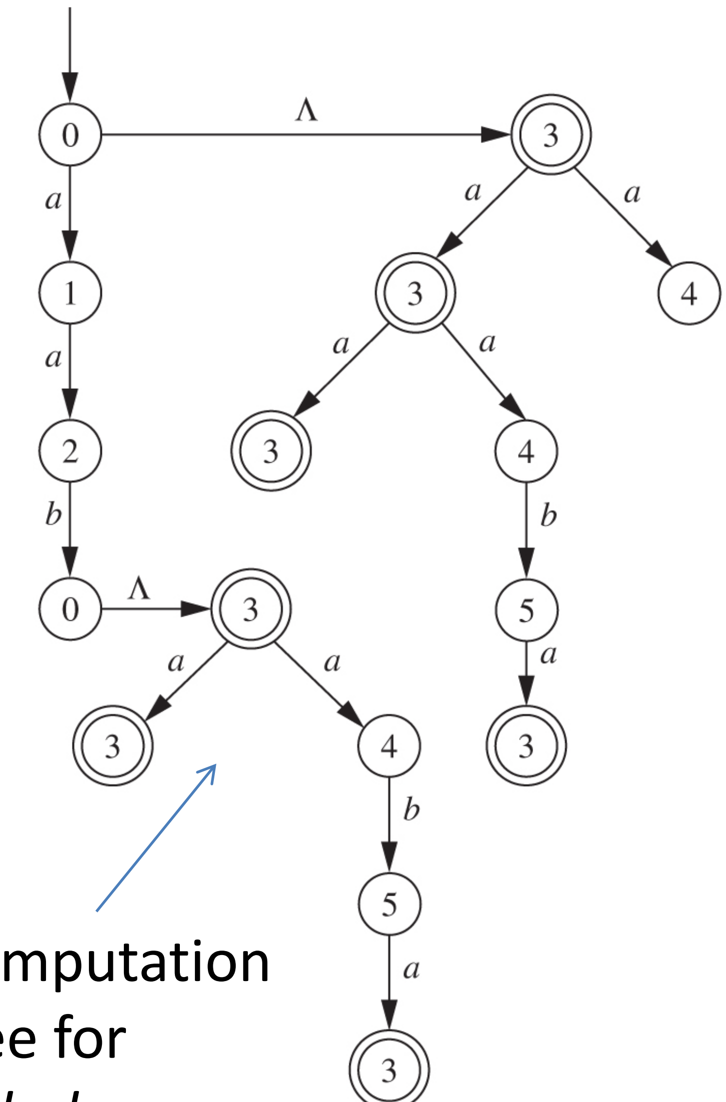


NFA: **Λ-transitions**

The technique in previous example does not provide a simple way to draw a transition diagram for *(aab)*(a+aba)**

- We introduce a new feature called **Λ-transition.**
- **It allows the device to change state without reading the next symbol.**





Computation tree for *aababa*

Nondeterministic Finite Automata

- Definition: A *nondeterministic finite automaton* (NFA) is a 5-tuple $(Q, \Sigma, q_0, A, \delta)$, where:
  - Q is a finite set of states,
  - Σ is a finite input alphabet
  - $q_0 \in Q$ is the initial state
  - $A \subseteq Q$ is the set of accepting states
  - $\delta : Q \times (\Sigma \cup \{\Lambda\}) \to 2^Q$ is the transition function.

  (The values of δ are not single states, but *sets* of states)
- For every $q \in Q$ and every $\sigma \in \Sigma \cup \{\Lambda\}$, we interpret $\delta(q, \sigma)$ as the set of states to which the NFA can move from state q on input σ

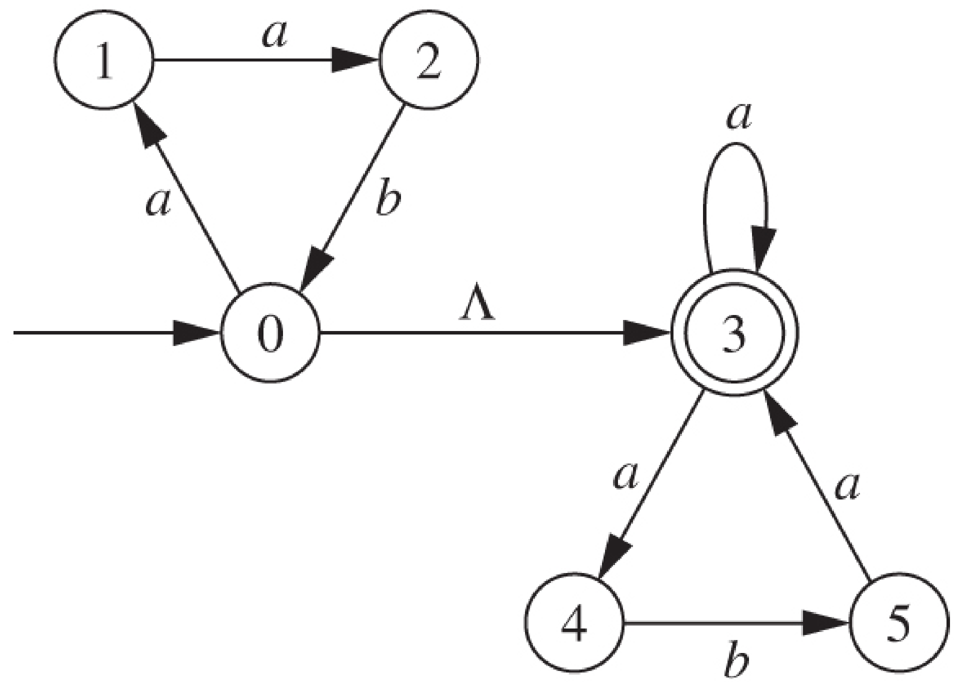
- Example:

$\delta(0,a) = \{1\}$

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

$\delta(0,b) = \varnothing$

$\delta(3,a) = \{3, 4\}$



Nondeterministic Finite Automata

How to define $\delta^*(q, x\sigma)$?

Defining δ^* is a little harder than for an FA, since $\delta^*(q, x)$ is a set, as is $\delta(p, \sigma)$ for any p in the first set:

$\cup \{ \delta(p, \sigma) \mid p \in \delta^*(q, x)\}$ is a first step towards δ^*

We must also consider Λ-transitions, which could potentially occur at any stage

- Definition: Suppose $M = (Q, \Sigma, q_0, A, \delta)$ is an NFA, and $S \subseteq Q$ is a set of states
  - **The Λ-*closure* of S is the set $\Lambda(S)$** that can be defined recursively as follows:
    - $S \subseteq \Lambda(S)$
    - For every $q \in \Lambda(S)$, $\delta(q, \Lambda) \subseteq \Lambda(S)$

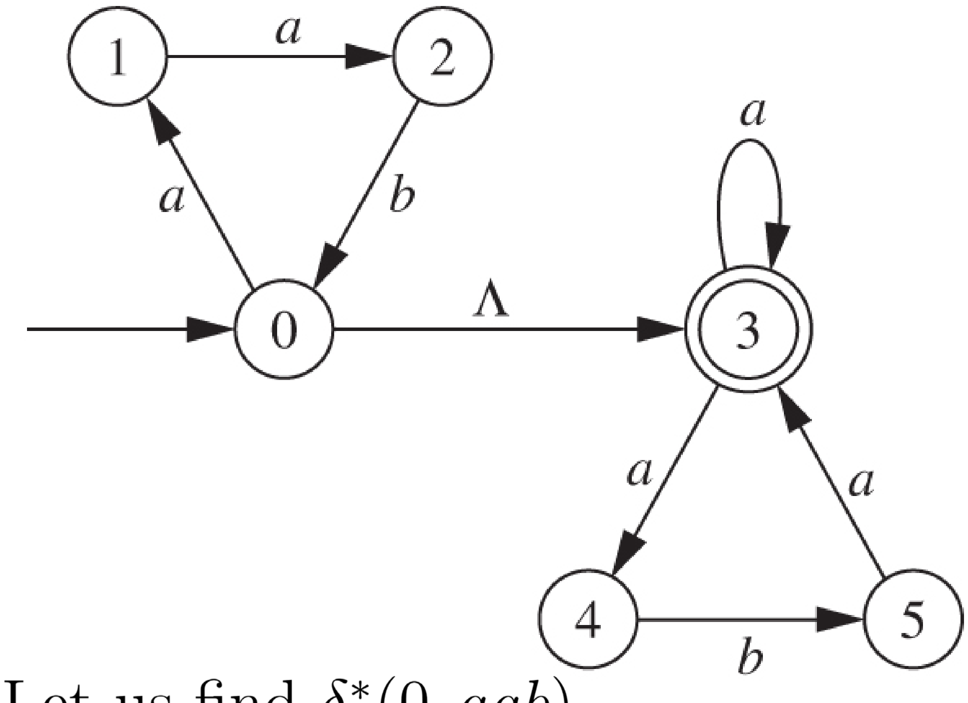
- Definition: Suppose $M = (Q, \Sigma, q_0, A, \delta)$ is an NFA, and $S \subseteq Q$ is a set of states
  - **The Λ-*closure* of S is the set $\Lambda(S)$** that can be defined recursively as follows:
    - $S \subseteq \Lambda(S)$
    - For every $q \in \Lambda(S)$, $\delta(q, \Lambda) \subseteq \Lambda(S)$

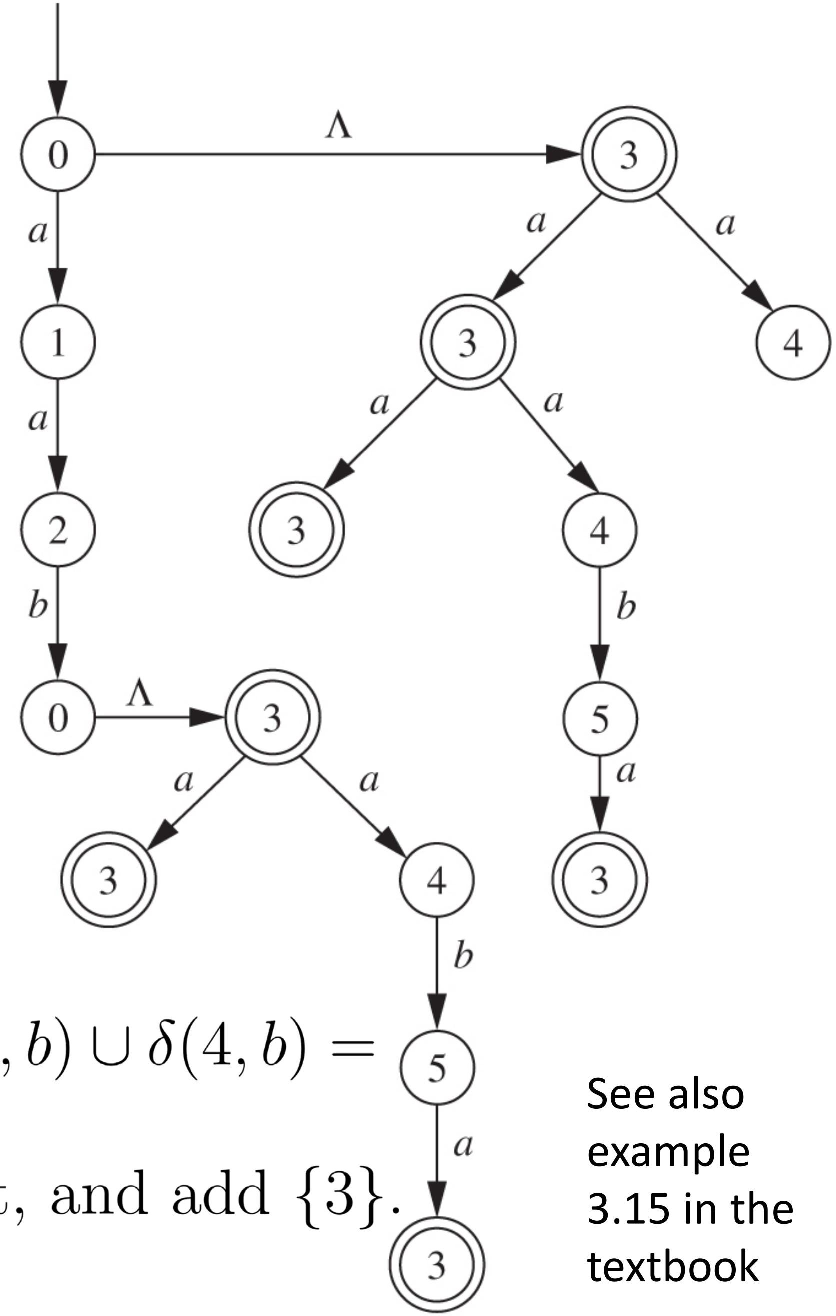
- As for any finite set that is defined recursively, we can easily formulate **an algorithm to calculate $\Lambda(S)$**:
  - Initialize T to be S, as in the basis part of the definition
  - Make a sequence of passes, in each pass considering every $q \in T$ and adding every state in $\delta(q, \Lambda)$ not already there
  - Stop after the first pass in which T does not change
  - The final value of T is $\Lambda(S)$

Nondeterministic Finite Automata

- Definition: Let $M=(Q, \Sigma, q_0, A, \delta)$ be an NFA
- Define the **extended transition function** $\delta^* : Q \times \Sigma^* \to 2^Q$ as follows:
  - For every $q \in Q$, $\delta^*(q,\Lambda) = \Lambda(\{q\})$
  - For every $q \in Q$, every $y \in \Sigma^*$, and every $\sigma \in \Sigma$
    - $\boldsymbol{\delta^*(q, y\sigma) = \Lambda(\cup \{\delta(p, \sigma) \mid p \in \delta^*(q, y)\})}$
  - A string $x \in \Sigma^*$ is accepted by M if $\delta^*(q_0, x) \cap A \neq \emptyset$

  (i.e., some sequence of transitions **involving the symbols of *x* and Λ's** leads from q_0 to an accepting state)
- <u>The language $L(M)$ accepted by M is the set of all strings accepted by M</u>





Let us find $\delta^*(0, aab)$

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

Then, $\delta^*(0, aa) = \{2, 3, 4\}$;

Now we add b to aa

$\cup\{\delta(p, b)|p \in \{2, 3, 4\}\} = \delta(2, b) \cup \delta(3, b) \cup \delta(4, b) =$

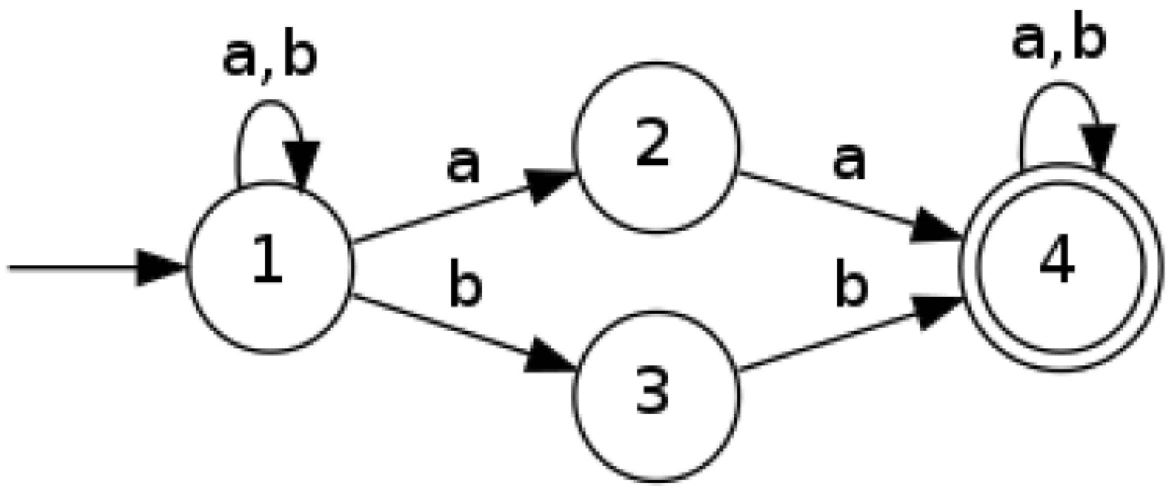
$= \{0\} \cup \emptyset \cup \{5\} = \{0, 5\}$;

Now we compute Λ-closure of this set, and add $\{3\}$.

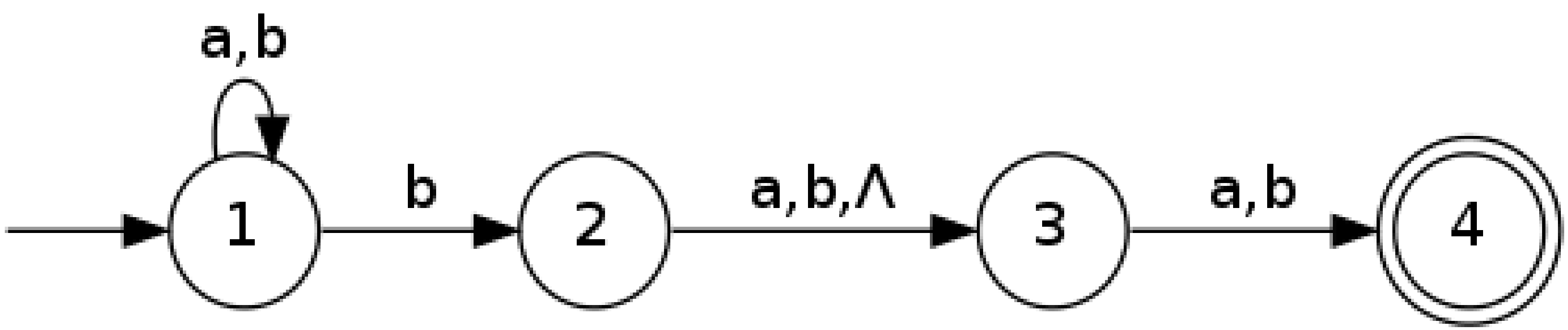
The answer is $\delta^*(0, aab) = \{0, 5, 3\}$.

See also example 3.15 in the textbook

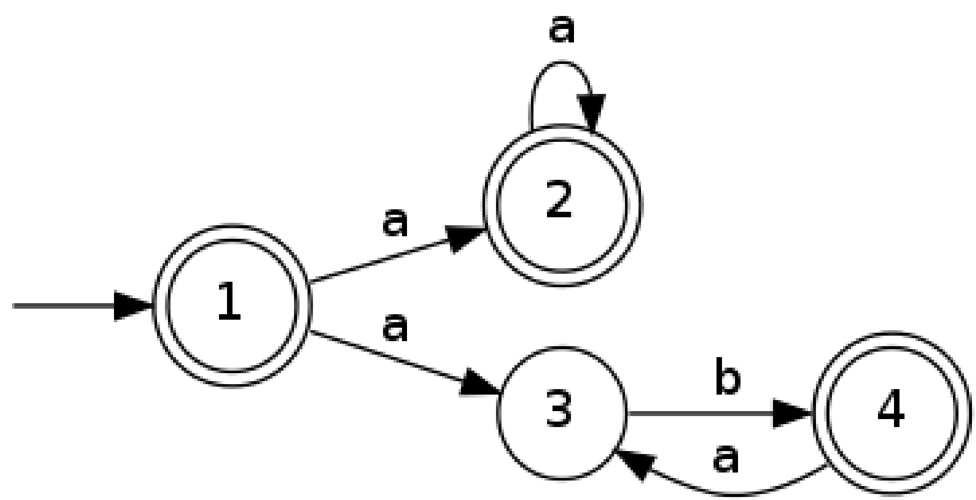
An NFA that accepts strings that contain aa or bb as a substring.

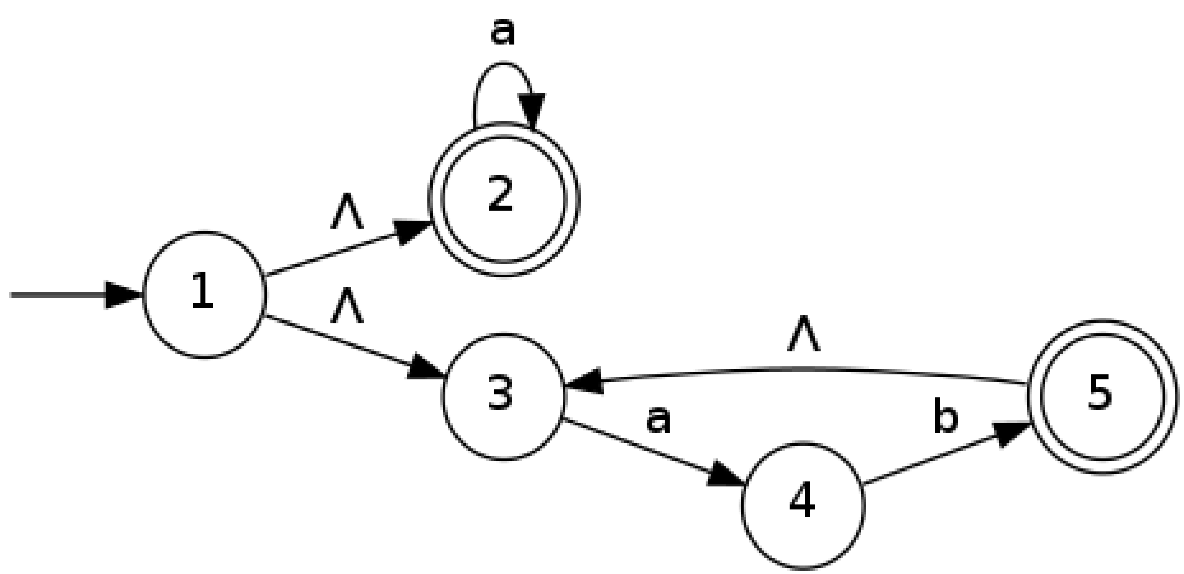


An NFA that accepts strings over {a,b} that contain b either at the third position from the right or at the second position from the right.

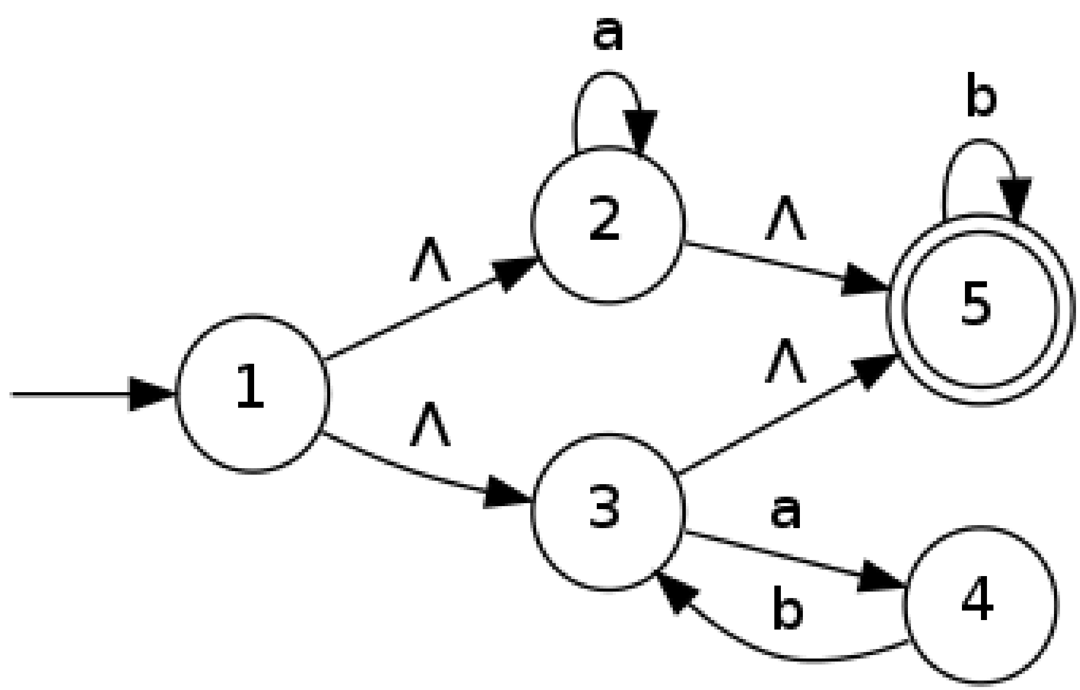


Simultaneous Pattern: NFA for a*+(ab)*

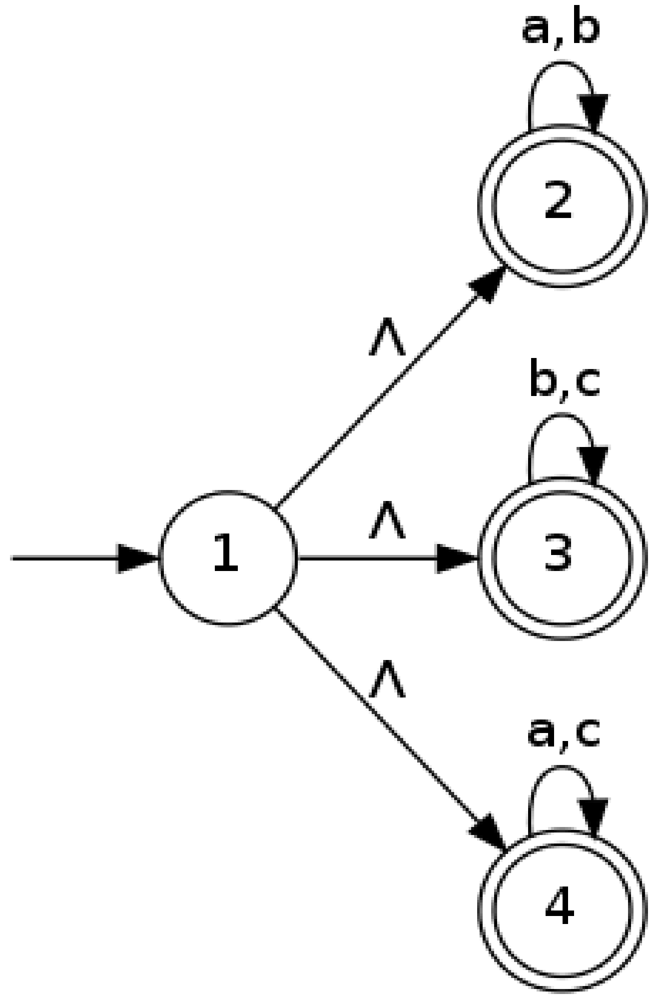




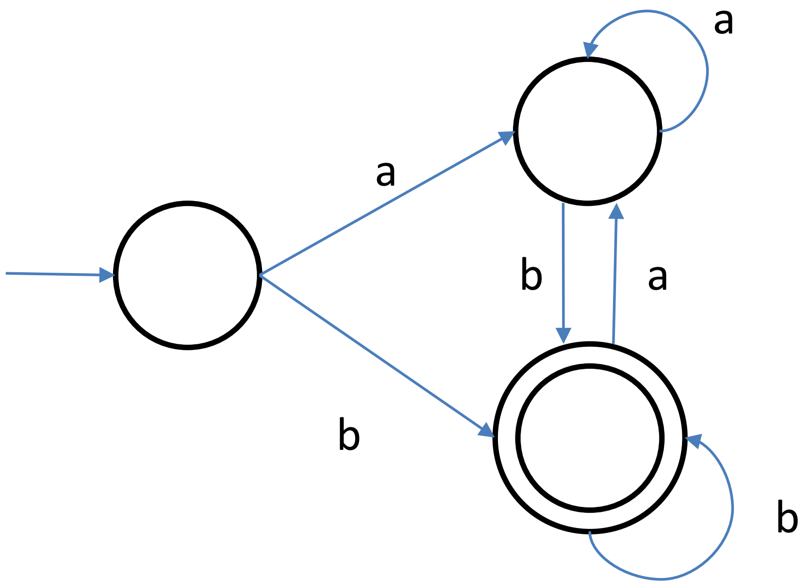
Simultaneous Pattern: NFA for (a*+(ab)*)b*



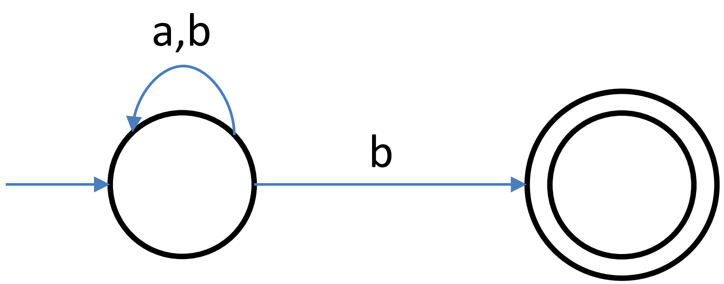
Simultaneous Pattern: NFA for all strings over {a,b,c} that are missing at least one letter. For example: ab,ccccc, bcbcbb, cacaaa



L = (a+b)*b



FA



NFA