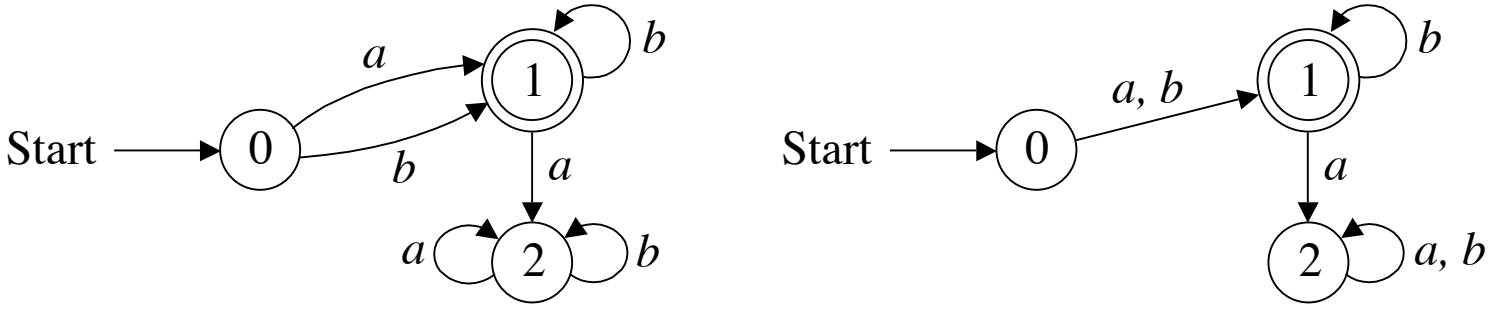
Section 11.2 Finite Automata

Can a machine(i.e., algorithm) recognize a regular language? Yes!

Deterministic Finite Automata

A *deterministic finite automaton* (DFA) over an alphabet A is a finite digraph (where vertices or nodes are called *states*) for which each state emits one labeled edge for each letter of A. One state is designated as the *start state* and a set of states may be *final states*.

Example. Either of the following alternatives is acceptable for representing a DFA, where final states are indicated by double circles.



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.

Example. The example DFA accepts the strings

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

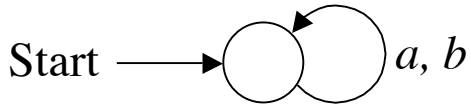
So the language of the DFA is given by the regular expression $(a + b)b^*$.

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

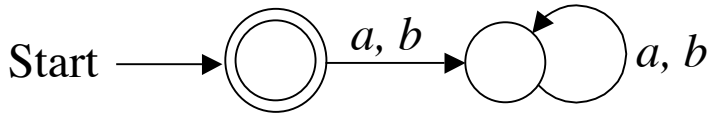
Quiz. Find an DFA for each of the following languages over the alphabet $\{a, b\}$.

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

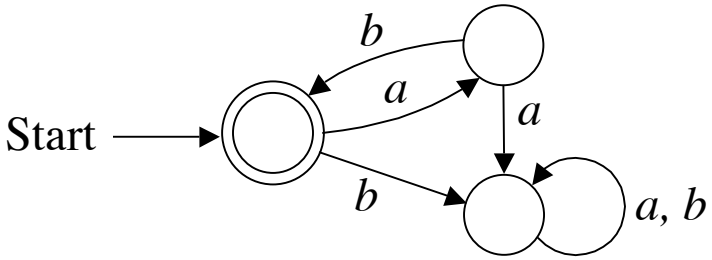
Solutions: (a):



(b):



(c):



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

Solution:

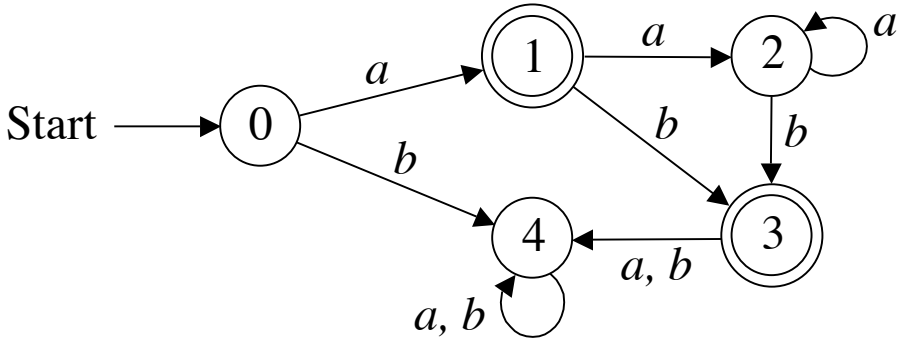
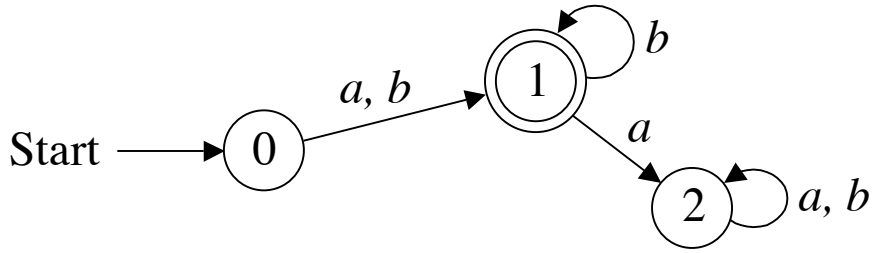


Table Representation of a DFA

A DFA over A can be represented by a *transition function* T : States × A → 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. For example, the following figures show a DFA and its *transition table*.



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

Note: T can be extended to T : States × A^* → States by

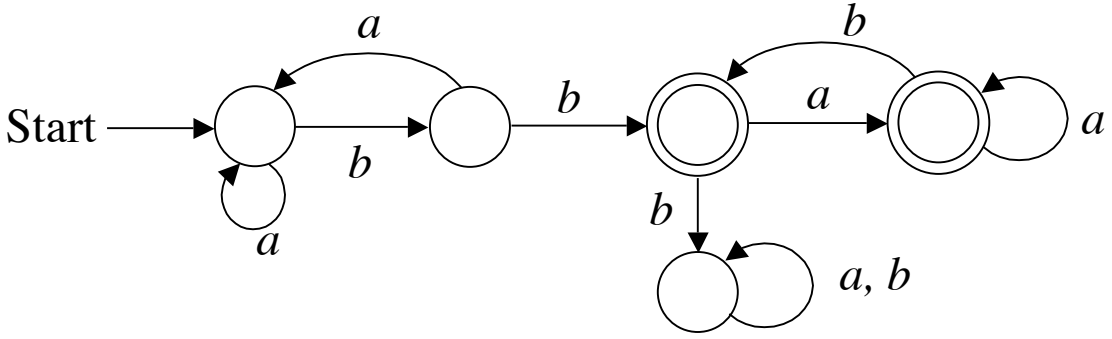
$$T(i, \Lambda) = i \quad \textit{and} \quad T(i, aw) = T(T(i, a), w) \qquad \text{for } a \in A \text{ and } w \in A^*.$$

Quiz: Calculate $T(0, bba)$.

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

Example/Quiz. Back to the problem of describing input strings over $\{a, b\}$ that contain exactly one substring bb. We observed that the strings could be described by the regular expression $(a + ba)^*bb(a + ab)^*$. Find a DFA to recognize the language.

A solution:



Nondeterministic Finite Automata

A *nondeterministic finite automaton* (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 $a + aa^*b + a^*b$.

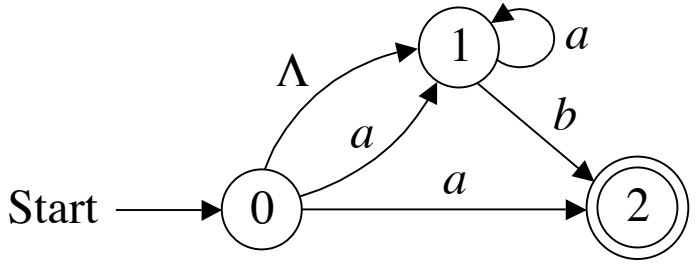


Table representation of NFA

An NFA over A can be represented by a function T : States × $A \cup \{\Lambda\} \rightarrow$ power(States), where $T(i, a)$ is the set of states reached from state i along the edge labeled a, and we mark the start and final states. The following figure shows the table for the preceding NFA.

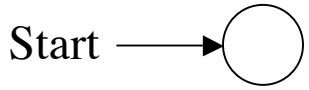
	T	a	b	Λ
start	0	{1, 2}	∅	{1}
	1	{1}	{2}	∅
final	2	∅	∅	∅

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

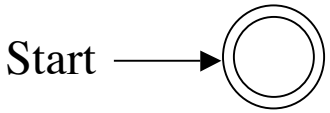
Quizzes. Find an NFA for each of the following languages over $\{a, b\}$.

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

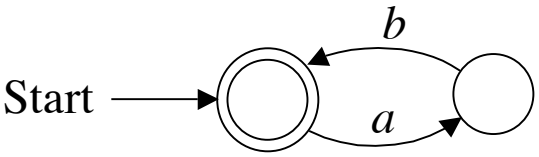
Solutions: (a):



(b):

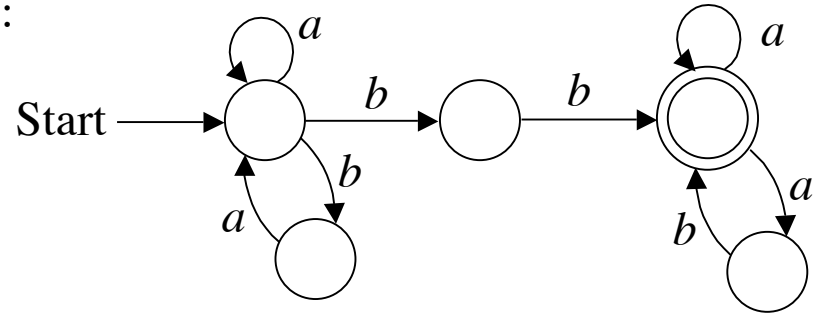


(c):

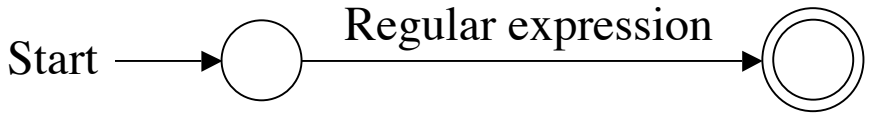


ExampleQuiz. Back to the problem of describing input strings over $\{a, b\}$ that contain exactly one substring *bb*. We observed that the strings could be described by the regular expression $(a + ba)^*bb(a + ab)^*$. Find an NFA to recognize the language.

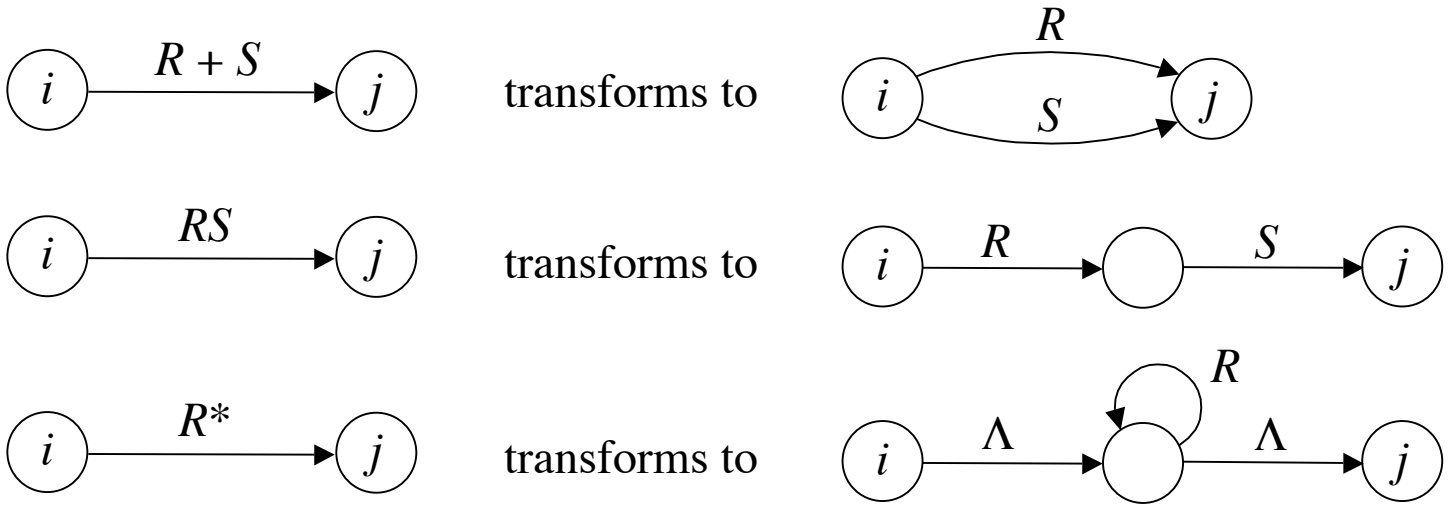
A solution:



Algorithm: *Transform a Regular Expression into a Finite Automaton*
Start by placing the regular expression on the edge between a start and final state:

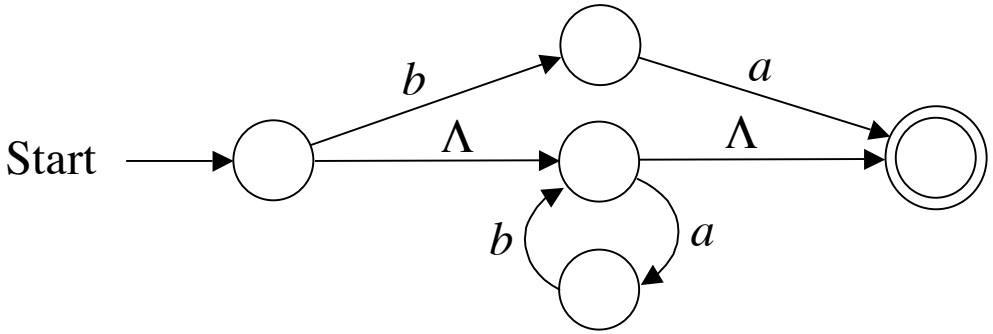


Apply the following rules to obtain a finite automaton after erasing any ∅-edges.

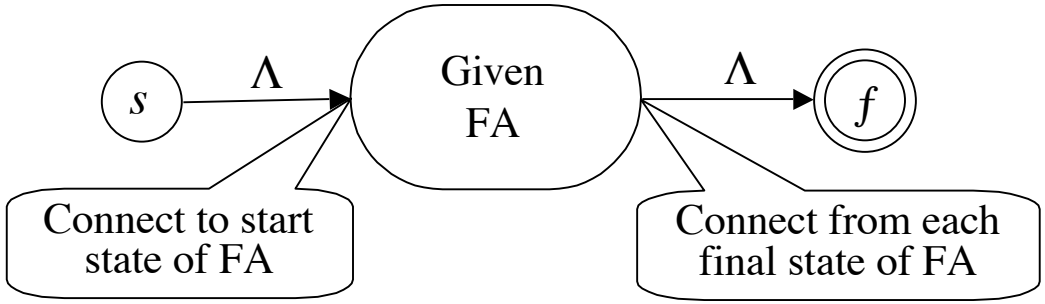


Quiz. Use the algorithm to construct a finite automaton for (*ab*)* + *ba*.

Answer:



Algorithm: *Transform a Finite Automaton into 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.

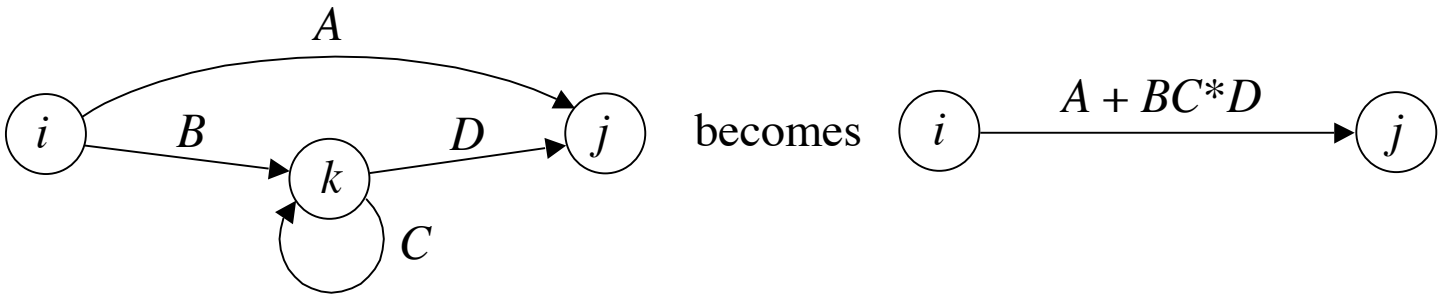


If needed, combine all multiple edges between the same two nodes into one edge with label the sum of the labels on the multiple edges. If there is no edge between two states, assume there is an ∅-edge.

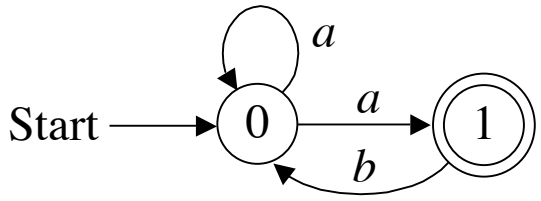
Now eliminate each state k of the FA by constructing a new edge (i, j) for each pair of edges (i, k) and (k, j) where $i \neq k$ and $j \neq k$. The new label new(i, j) is defined in terms of the old labels by the formula

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

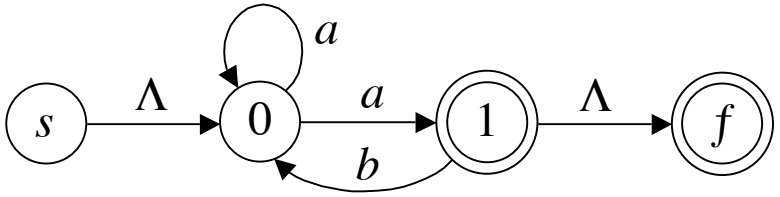
Example.



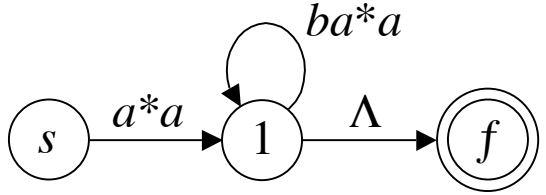
Quiz. Use the algorithm to transform the following NFA into a regular expression. (Half the class eliminate state 0 then state 1 and half the class eliminate state 1 then state 0.)



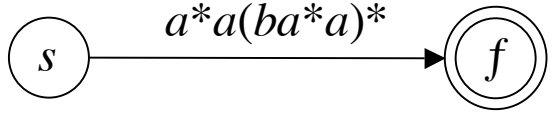
Solution: Connect the NFA to new a start state s and a new final state f as pictured.



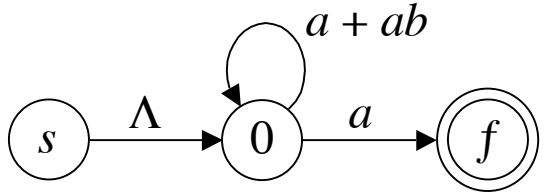
First Solution: Eliminate state 0 to obtain:
new(s, 1) = $\varnothing + \Lambda a^*a = a^*a$.
new(1, 1) = $\varnothing + ba^*a = ba^*a$.



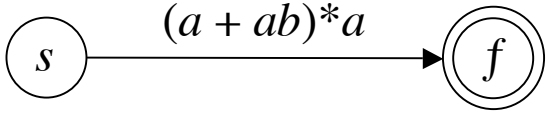
Eliminate state 1 to obtain:
new(s, f) = $\varnothing + a^*a(ba^*a)^*\Lambda = a^*a(ba^*a)^*$.



Second Solution: Eliminate state 1 to obtain:
new(0, f) = $\varnothing + a\varnothing^*\Lambda = a$.
new(0, 0) = $a + a\varnothing^*b = a + ab$.



Eliminate state 0 to obtain:
new(s, f) = $\varnothing + \Lambda(a + ab)^*a = (a + ab)^*a$.



Quiz. Use regular algebra to show the following equality from the previous quiz.

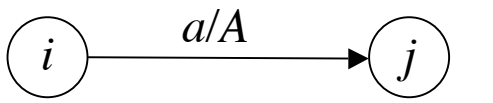
$a^*a(ba^*a)^* = (a + ab)^*a$.

Proof:

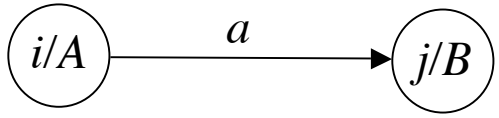
$a^*a(ba^*a)^*$	$= a^*[a((ba^*)a)^*]$	· is associative
	$= a^*[(a(ba^*))^*a]$	$R(SR)^* = (RS)^*R$
	$= a^*[((ab)a^*)^*a]$	· is associative
	$= [a^*((ab)a^*)^*]a$	· is associative
	$= (a + ab)^*a$	$R^*(SR^*)^* = (R + S)^*$. QED.

Automata with Output (Mealy and Moore)

Mealy machine: Associate an output action with each transition between states.



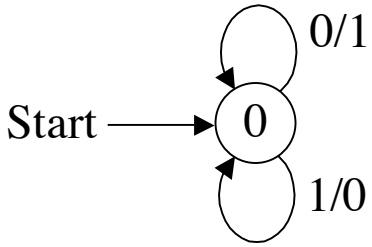
Moore machine: Associate an output action with each state.

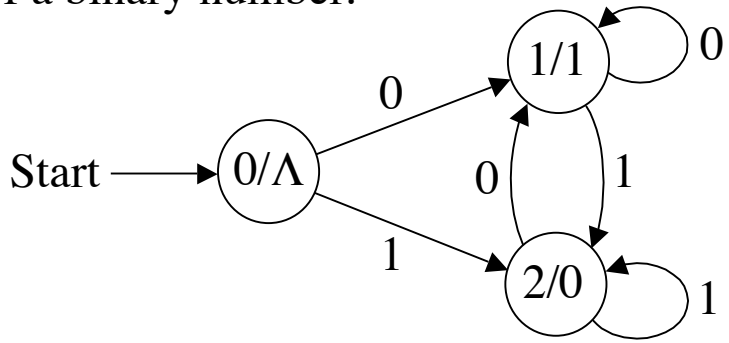


Theorem: Mealy and Moore machines are equivalent.

Example/Quiz. Output the complement of a binary number.

Solutions:





Challenge: Describe a two-floor elevator system with a Mealy or Moore machine.