Finite Automata (Section 11.2)

Finite Automata

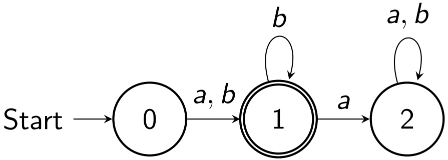
- Can a machine (i.e., algorithm) recognize a regular language?

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:* where final states are indicated by double circles



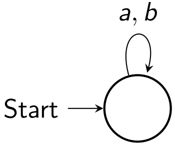
Execution of a DFA

- The *execution* of a 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 the DFA is the set of accepted strings.
- *Example:* The previous 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 $(a + b)b^*$.

Regular Languages

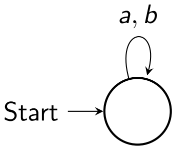
- *Theorem (Kleene):* The class of regular languages is exactly the same as the class of languages accepted by DFAs.
- Find a DFA for each of the following languages over the alphabet $\{a, b\}$
    - $\emptyset$

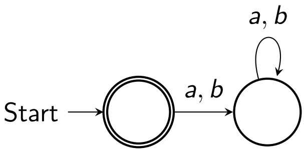
Regular Languages

- *Theorem (Kleene):* The class of regular languages is exactly the same as the class of languages accepted by DFAs.
- Find a DFA for each of the following languages over the alphabet $\{a, b\}$
  - $\varnothing$
  - 

  - $\{\Lambda\}$

Regular Languages

- *Theorem (Kleene):* The class of regular languages is exactly the same as the class of languages accepted by DFAs.
- Find a DFA for each of the following languages over the alphabet $\{a, b\}$
  - $\varnothing$
  - 

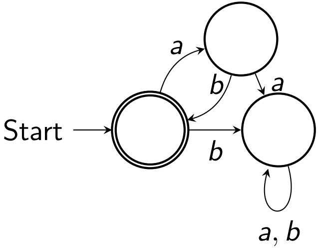
  - $\{\Lambda\}$
  - 

Another example

Find a DFA for the language $\{(ab)^n | n \in \mathbb{N}\}$

Another example

Find a DFA for the language $\{(ab)^n | n \in \mathbb{N}\}$



Finding DFAs

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

Finding DFAs

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

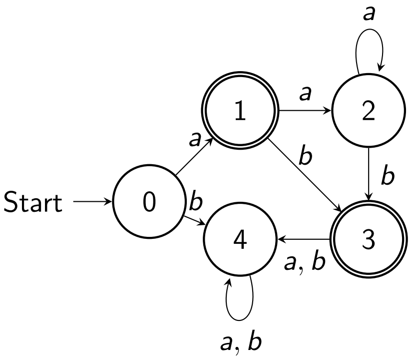
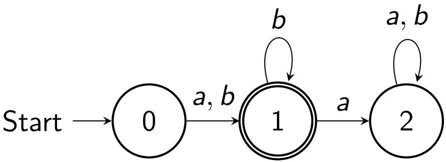


Table Representation of a DFA

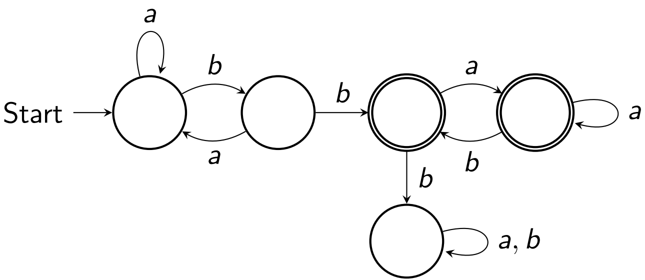
A DFA over A can be represented by a transition function T : States $\times A \rightarrow$ States, where $T(i, a)$ is the state reached from state i along the edge labeled a, and we mark the start and final states. For example, we show a DFA and its *transition table*.



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

Original Example

Back to the problem of describing input strings over $\{a, b\}$ that contain exactly one substring bb. These strings can be described by the regular expression $(a + ba)^*bb(a + ab)^*$. Here is a DFA:



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

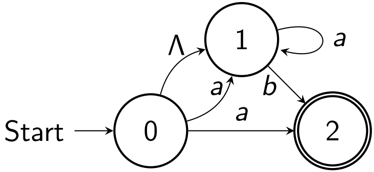


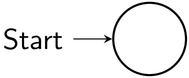
Table representation of NFA

An NFA over A can be represented by a function T : States $\times A \cup \{\Lambda\} \rightarrow$ 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. Here is the table for the preceding NFA:

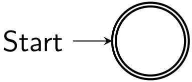
	T	a	b	Λ
start	0	$\{1, 2\}$	$\varnothing$	$\{1\}$
	1	$\{1\}$	$\{2\}$	$\varnothing$
final	2	$\varnothing$	$\varnothing$	$\varnothing$

Theorem: (Rabin and Scott)

- The class of regular languages is exactly the same as the class of languages accepted by NFAs.
- Find an NFA for the language $\varnothing$:

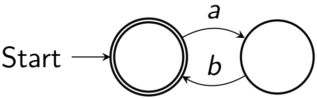


- Find an NFA for the language $\{\Lambda\}$:



Continued

- Find an NFA for the language $\{(ab)^n | n \in \mathbb{N}\}$:

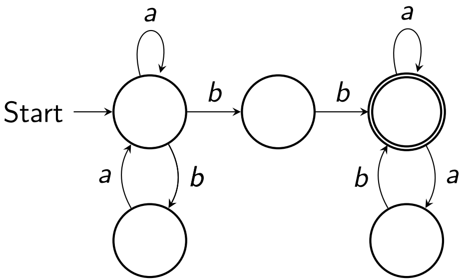


Original Problem yet again

Back to the problem of describing input strings over $\{a, b\}$ that contain exactly on string 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:

Original Problem yet again

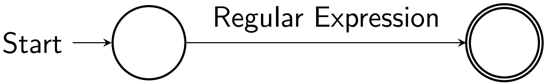
Back to the problem of describing input strings over $\{a, b\}$ that contain exactly on string bb. We observed that the strings could be described by the regular expression $(a + ba)^* bb(a + ab)^*$. Find an NFA to recognize the language:



Regular Expression to Finite Automaton

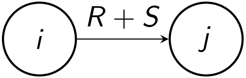
How to transform any regular expression into a finite automaton:

- Start by placing the regular expression on the edge between a start and final state:

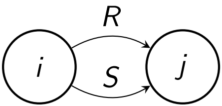


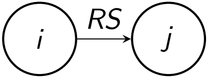
Reg Exp to FA continued

Apply the follow rules to obtain a finite automaton after erasing any $\varnothing$-edges.

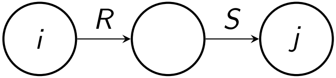


transforms to

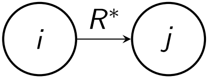




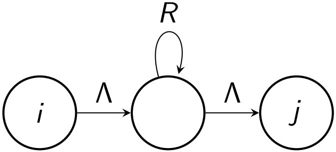
transforms to



Reg Exp to FA continued



transforms to

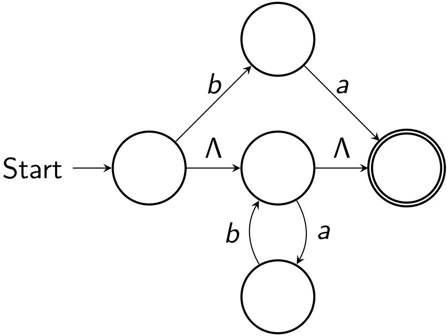


Example

Use the algorithm to construct an FA for $(ab)^* + ba$

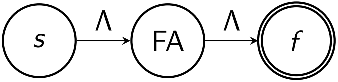
Example

Use the algorithm to construct an FA for $(ab)^* + ba$



Transform an FA into a Reg Exp

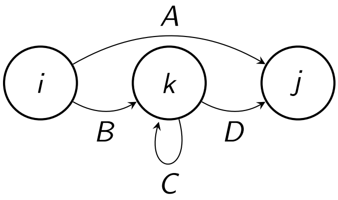
Algorithm: Connect a new start state s to the start state of the FA and connect each final state of the FA to a new final state f as shown:



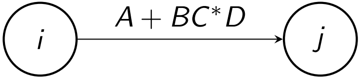
Continued

- Then, 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 are no edges between two states, assume there is $\emptyset$-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:
- new$(i, j) =$old$(i, j)+$old(i, k)old$(k, k)^*$old(k, j)

Example

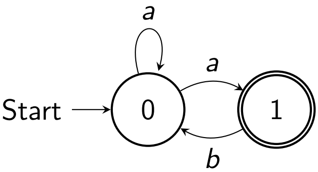


becomes

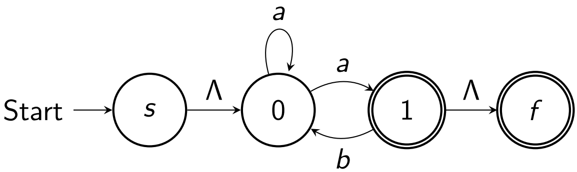


Complete Example

Use the algorithm to transform the following NFA into a regular expression. We will do this twice, changing the order that we eliminate the states.



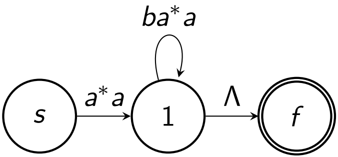
First connect the NFA to a new start state s and new final state f:



First Solution

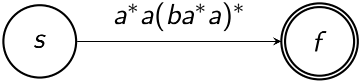
Eliminate state 0 to obtain:

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



Eliminate state 1 to obtain:

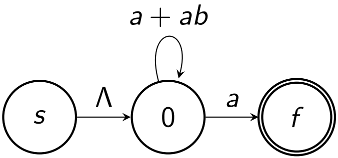
- $\text{new}(s, f) = \varnothing + a^* a(ba^* a)^* \Lambda = a^* a(ba^* a)^*.$



Second Solution

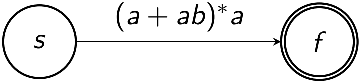
Eliminate state 1 to obtain:

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



Eliminate state 0 to obtain:

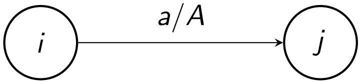
- $\text{new}(s, f) = \varnothing + \Lambda(a + ab)^* a = (a + ab)^* a$



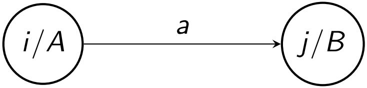
Automata with Output

There are two kinds:

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



- Moore machine: Associate an output action with each state.



Example

Output the complement of a binary number: