

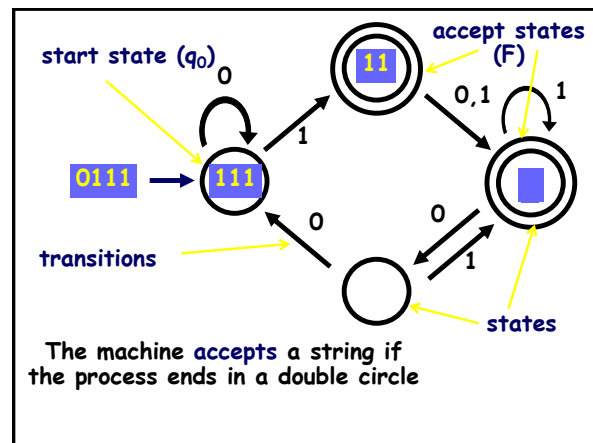
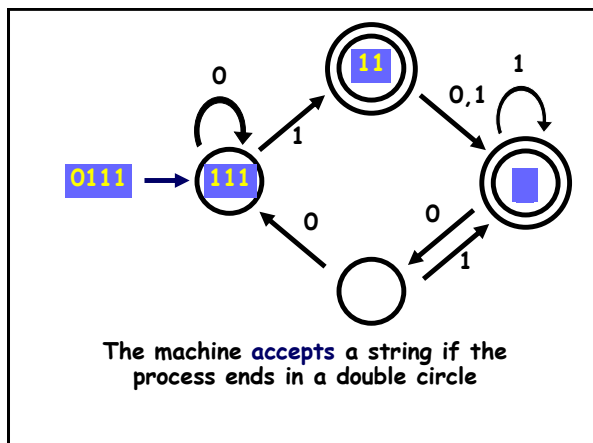
Great Theoretical Ideas In Computer Science
 Victor Adamchik CS 15-251 Spring 2010
 Danny Sleator
 Lecture 20 Mar 30, 2010 Carnegie Mellon University

Finite Automata

Deterministic Finite Automata

A machine so simple that you can understand it in less than one minute

Wishful thinking...



Anatomy of a Deterministic Finite Automaton

The singular of automata is automaton.

The **alphabet** of a finite automaton is the set where the symbols come from, for example $\{0,1\}$

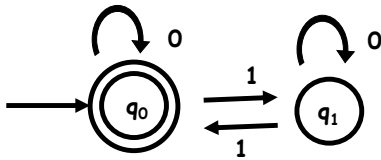
The **language** of a finite automaton is the set of strings that it accepts

The Language $L(M)$ of Machine M

$L(M) = \text{All strings of 0s and 1s}$

The **language** of a finite automaton is the set of strings that it accepts

The **Language** $L(M)$ of Machine M



$L(M) = \{ w \mid w \text{ has an even number of } 1\text{s} \}$

Notation

An alphabet Σ is a finite set (e.g., $\Sigma = \{0,1\}$)

A string over Σ is a finite-length sequence of elements of Σ

For x a string, $|x|$ is the length of x

The unique string of length 0 will be denoted by ϵ and will be called the empty or null string

A **language** over Σ is a set of strings over Σ

A finite automaton is $M = (Q, \Sigma, \delta, q_0, F)$

Q is the finite set of states

Σ is the alphabet

$\delta : Q \times \Sigma \rightarrow Q$ is the transition function

$q_0 \in Q$ is the start state

$F \subseteq Q$ is the set of accept states

$L(M)$ = the language of machine M
= set of all strings machine M accepts

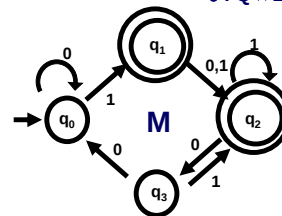
$M = (Q, \Sigma, \delta, q_0, F)$ where $Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0,1\}$

$q_0 \in Q$ is start state

$F = \{q_1, q_2\} \subseteq Q$ accept states

$\delta : Q \times \Sigma \rightarrow Q$ transition function



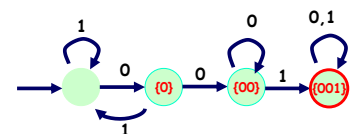
δ	0	1
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_3	q_2
q_3	q_0	q_2

The finite-state automata are **deterministic**, if for each pair $Q \times \Sigma$ of state and input value there is a unique next state given by the transition function.

There is another type machine in which there may be several possible next states. Such machines called **nondeterministic**.

EXAMPLE

Build an automaton that accepts all and only those strings that contain 001



Build an automaton that accepts all binary numbers that are divisible by 3, i.e., $L = 0, 11, 110, 1001, 1100, 1111, 10010, 10101, \dots$

A **language** over Σ is a set of strings over Σ

A language is **regular** if it is recognized by a deterministic finite automaton

$L = \{ w \mid w \text{ contains } 001 \}$ is regular

$L = \{ w \mid w \text{ has an even number of 1s} \}$ is regular

Determine the language recognized by

$L(M) = \{1^n \mid n = 0, 1, 2, \dots\}$

Determine the language recognized by

$L(M) = \{1, 01\}$

Determine the language recognized by

$L(M) = \{0^n, 0^n 10x \mid n=0,1,2,\dots, \text{ and } x \text{ is any string}\}$

DFA Membership problem

Determine whether some word belongs to the language.

Theorem: The DFA Membership Problem is solvable in linear time.

Let $M = (Q, \Sigma, \delta, q_0, F)$ and $w = w_1 \dots w_m$.
Algorithm for DFA M :

$p := q_0$;

for $i := 1$ to m do $p := \delta(p, w_i)$;

if $p \in F$ then return Yes else return No.

Equivalence of two DFAs

Definition: Two DFAs M_1 and M_2 over the same alphabet are equivalent if they accept the same language: $L(M_1) = L(M_2)$.

Given a few equivalent machines, we are naturally interested in the **smallest** one with the least number of states.

Union Theorem

Given two languages, L_1 and L_2 , define the union of L_1 and L_2 as

$$L_1 \cup L_2 = \{ w \mid w \in L_1 \text{ or } w \in L_2 \}$$

Theorem: The union of two regular languages is also a regular language.

Theorem: The union of two regular languages is also a regular language

Proof (Sketch): Let

$M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$ be finite automaton for L_1
and

$M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$ be finite automaton for L_2

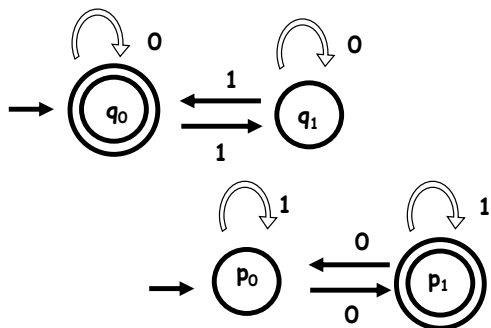
We want to construct a finite automaton

$M = (Q, \Sigma, \delta, q_0, F)$ that recognizes $L = L_1 \cup L_2$

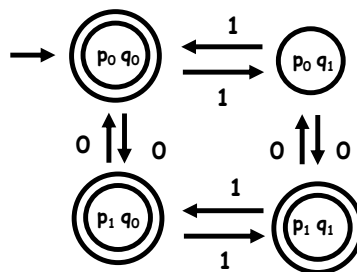
Idea: Run both M_1 and M_2 at the same

Q = pairs of states, one from M_1 and one from M_2
 $= \{ (q_1, q_2) \mid q_1 \in Q_1 \text{ and } q_2 \in Q_2 \}$
 $= Q_1 \times Q_2$

Theorem: The union of two regular languages is also a regular language



Automaton for Union



The Regular Operations

Union: $A \cup B = \{ w \mid w \in A \text{ or } w \in B \}$

Intersection: $A \cap B = \{ w \mid w \in A \text{ and } w \in B \}$

Negation: $\neg A = \{ w \mid w \notin A \}$

Reverse: $A^R = \{ w_1 \dots w_k \mid w_k \dots w_1 \in A \}$

Concatenation: $A \cdot B = \{ vw \mid v \in A \text{ and } w \in B \}$

Star: $A^* = \{ w_1 \dots w_k \mid k \geq 0 \text{ and each } w_i \in A \}$

Reverse

Reverse: $A^R = \{ w_1 \dots w_k \mid w_k \dots w_1 \in A \}$

How to construct a DFA for the reversal of a language?

The direction in which we read a string should be irrelevant. If we flip transitions around we might not get a DFA.

The Kleene closure: A^*

Star: $A^* = \{ w_1 \dots w_k \mid k \geq 0 \text{ and each } w_i \in A \}$

From the definition of the concatenation, we define A^n , $n = 0, 1, 2, \dots$ recursively

$$A^0 = \{\epsilon\}$$

$$A^{n+1} = A^n \cdot A$$

A^* is a set consisting of concatenations of arbitrary many strings from A .

$$A^* = \bigcup_{k=0}^{\infty} A^k$$

The Kleene closure: A^*

What is A^* of $A=\{0,1\}$?

All binary strings

What is A^* of $A=\{11\}$?

All binary strings of an even number of 1s

Regular Languages Are Closed Under The Regular Operations

We have seen part of the proof for Union. The proof for intersection is very similar. The proof for negation is easy.

Theorem: Any finite language is regular

Claim 1: Let w be a string over an alphabet. Then $\{w\}$ is a regular language.

Proof: By induction on the number of characters. If $\{a\}$ and $\{b\}$ are regular then $\{ab\}$ is regular

Claim 2: A language consisting of n strings is regular

Proof: By induction on the number of strings. If $\{a\}$ then $LU\{a\}$ is regular

Pattern Matching

Input: Text T of length t , string S of length n

Problem: Does string S appear inside text T ?

Naïve method:

$a_1, a_2, a_3, a_4, a_5, \dots, a_t$

Cost: Roughly nt comparisons

Automata Solution

Build a machine M that accepts any string with S as a consecutive substring

Feed the text to M

Cost: t comparisons + time to build M

As luck would have it, the Knuth, Morris, Pratt algorithm builds M quickly

Real-life Uses of DFAs

Grep

Coke Machines

Thermostats (fridge)

Elevators

Train Track Switches

Lexical Analyzers for Parsers



Are **all** languages regular?

Consider the language $L = \{ a^n b^n \mid n > 0 \}$

i.e., a bunch of **a**'s followed by an equal number of **b**'s

No finite automaton accepts this language

Can you prove this?



$a^n b^n$ is not regular.
No machine has enough states to keep track of the number of **a**'s it might encounter

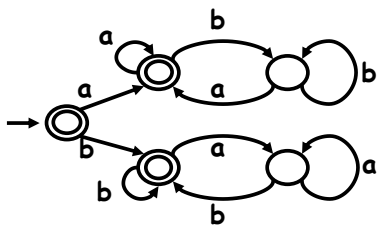
That is a fairly weak argument

Consider the following example...



L = strings where the # of occurrences of the pattern ab is equal to the number of occurrences of the pattern ba

Can't be regular. No machine has enough states to keep track of the number of occurrences of ab



M accepts only the strings with an equal number of ab 's and ba 's!

L = strings where the # of occurrences of the pattern ab is equal to the number of occurrences of the pattern ba

Can't be regular. No machine has enough states to keep track of the number of occurrences of ab



Let me show you a professional strength proof that $a^n b^n$ is not regular...



How to prove a language is not regular...

Assume it is regular, hence is accepted by a DFA M with n states.

Show that there are two strings s_1 and s_2 which both reach some state in M (usually by pigeonhole principle)

Then show there is some string t such that string $s_1 t$ is in the language, but $s_2 t$ is not. However, M accepts either both or neither.

Pigeonhole principle:



If we put n objects into m pigeonholes and if $n > m$, then at least one pigeonhole must have more than one item in it.

Theorem: $L = \{a^n b^n \mid n > 0\}$ is not regular

Proof (by contradiction):

Assume that L is regular, $M = (Q, \{a, b\}, \delta, q_0, F)$

Consider $\delta(q_0, a^i)$ for $i = 1, 2, 3, \dots$

There are infinitely many i 's but a finite number of states.

$\delta(q_0, a^n) = q$ and $\delta(q_0, a^m) = q$, and $n \neq m$

Since M accepts $a^n b^n$ $\delta(q, b^n) = q_f$

$\delta(q_0, a^m b^n) = \delta(\delta(q_0, a^m), b^n) = \delta(q, b^n) = q_f$

It follows that M accepts $a^m b^n$, and $n \neq m$

The finite-state automata are **deterministic**, if for each pair of state and input value there is a unique next state given by the transition function.

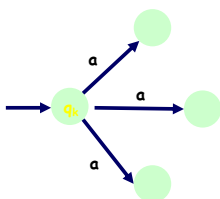
There is another type machine in which there may be several possible next states. Such machines called **nondeterministic**.

Nondeterministic finite automaton (NFA)

A NFA is defined using the same notations $M = (Q, \Sigma, \delta, q_0, F)$ as DFA except the transition function δ assigns a set of states to each pair $Q \times \Sigma$ of state and input.

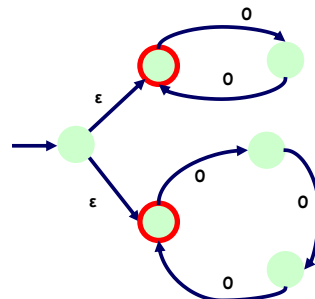
Note, every DFA is automatically also a NFA.

Nondeterministic finite automaton

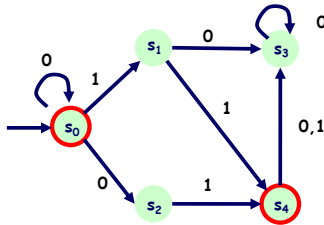


Allows transitions from q_k on the same symbol to many states

NFA for $\{0^k \mid k \text{ is a multiple of 2 or 3}\}$



What does it mean that for a NFA to recognize a string $x = x_1x_2...x_k$?



Since each input symbol x_j (for $j > 1$) takes the previous state to a set of states, we shall use a **union** of these states.

What does it mean that for a NFA to recognize a string?

Here we are going formally define this.

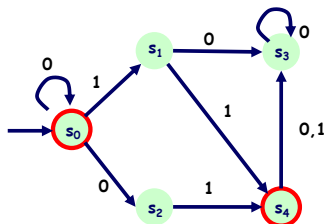
For a state q and string w , $\delta^*(q, w)$ is the set of states that the NFA can reach when it reads the string w starting at the state q .

Thus for NFA = $(Q, \Sigma, \delta, q_0, F)$, the function

$$\delta^*: Q \times \Sigma^* \rightarrow 2^Q$$

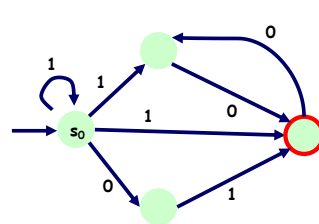
is defined by $\delta^*(q, \gamma x_k) = \bigcup_{p \in \delta^*(q, \gamma)} \delta(p, x_k)$

Find the language recognized by this NFA



$$L = \{0^n, 0^n01, 0^n11 \mid n = 0, 1, 2, \dots\}$$

Find the language recognized by this NFA



$$L = 1^*(01, 1, 10)(00)^*$$

Nondeterministic finite automaton

Theorem.

If the language L is recognized by a NFA M_0 , then L is also recognized by a DFA M_1 .

In other words,

if we ask if there is a NFA that is not equivalent to any DFA. The answer is No.

NFA vs. DFA

Advantages.

Easier to construct and manipulate.
Sometimes exponentially smaller.
Sometimes algorithms much easier.

Drawbacks

Acceptance testing slower.
Sometimes algorithms more complicated.



- DFA
- NFA

Study Bee