Finite Automata

N Geetha
AM & CS
PSG Tech

Why Finite Automata and Regular Expressions?

- Regular expressions (REs) are used in many systems.
  - E.g., UNIX, Linux, OS X,… a.*b.
  - E.g., Document Type Definitions describe XML tags with a RE format like person (name, addr, child*).
- Finite automata model protocols, electronic circuits.
  - Theory is used in *model-checking*.

Why Context-Free Grammars?

- Context-free grammars (CFGs) are used to describe the syntax of essentially **every** modern programming language.
- Every modern complier uses CFG concepts to parse programs
  - Not to forget their important role in describing natural languages.
- And Document Type Definitions are really CFG’s.

Why Turing Machines?

- When developing solutions to real problems, we often confront the limitations of what software can do.
  - *Undecidable* things – no program can do it 100% of the time with 100% accuracy.
  - *Intractable* things – there are programs, but no fast programs.
- A course on Automata Theory and Formal Languages gives you the tools.

Other Good Stuff

- Learn how to deal formally with discrete systems.
  - Proofs: You never really prove a program correct, but you need to be thinking of why a tricky technique really works.
- Gain experience with abstract models and constructions.
  - Models layered software architectures.

Finite Automata

Motivation

Examples

Informal Explanation

- Finite automata are finite collections of states with transition rules that take you from one state to another.
- Original application was sequential switching circuits, where the "state" was the settings of internal bits.
- Today, several kinds of software can be modeled by Finite Automata.

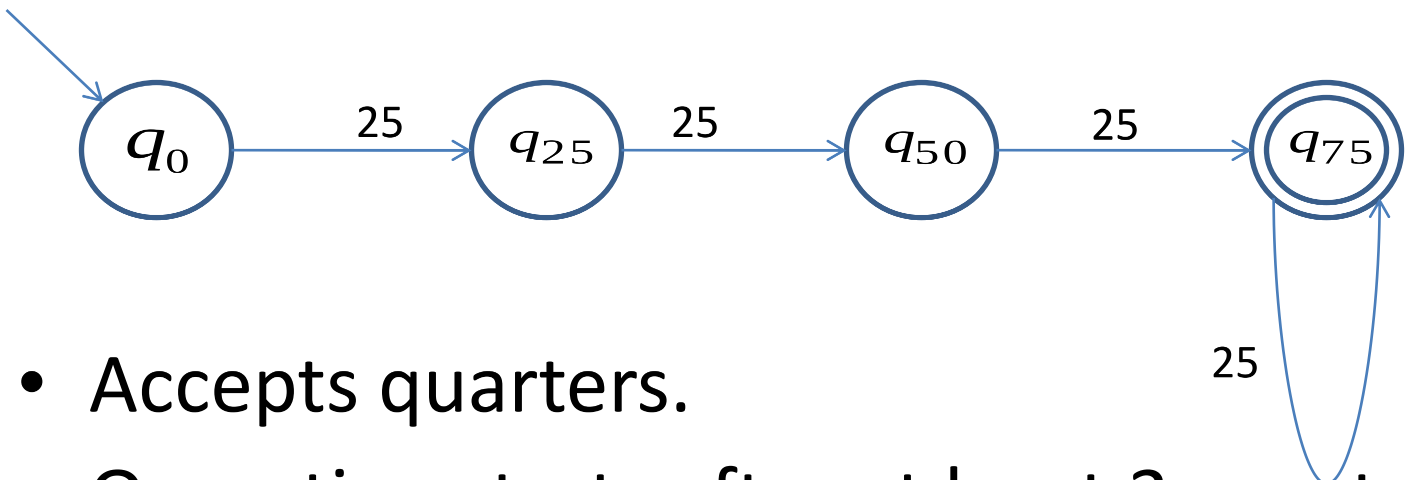
Representing Finite Automata

- Simplest representation is often a graph.
  - Nodes = states.
  - Arcs indicate state transitions.
  - Labels on arcs tell what causes the transition.

Finite Automata - A Short Example

- The control of a washing machine is a very simple example of a finite automaton.
- The most simple washing machine accepts quarters and operation does not start until at least 3 quarters were inserted.

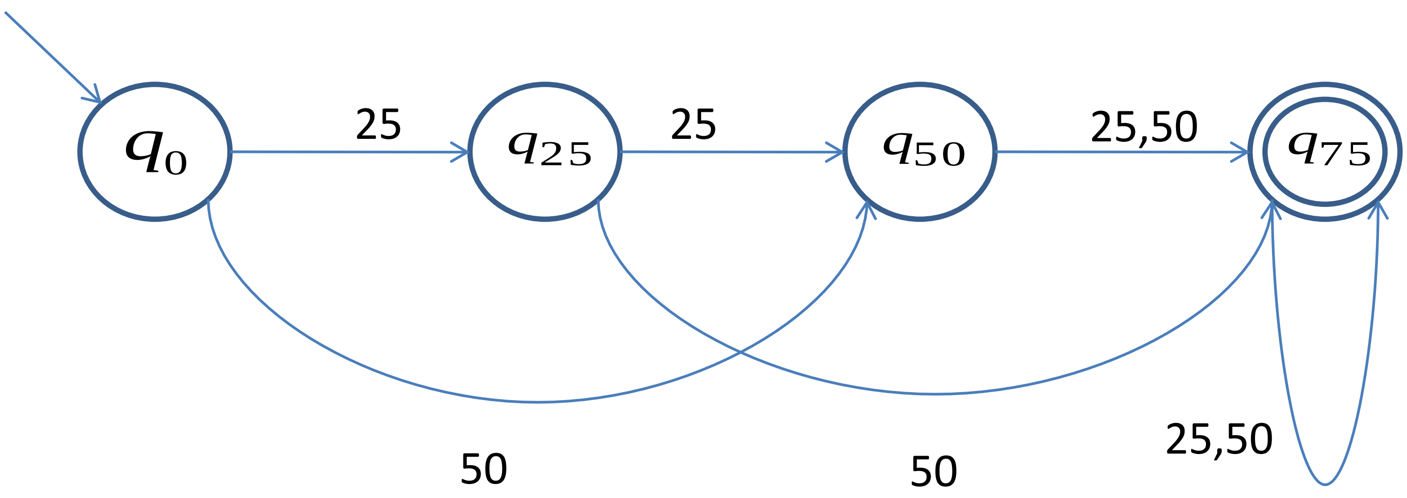
Control of a Simple Washing Machine



- Accepts quarters.
- Operation starts after at least 3 quarters were inserted.
- Accepted words: 25,25,25; 25,25,25,25; …

Finite Automata - A Short Example

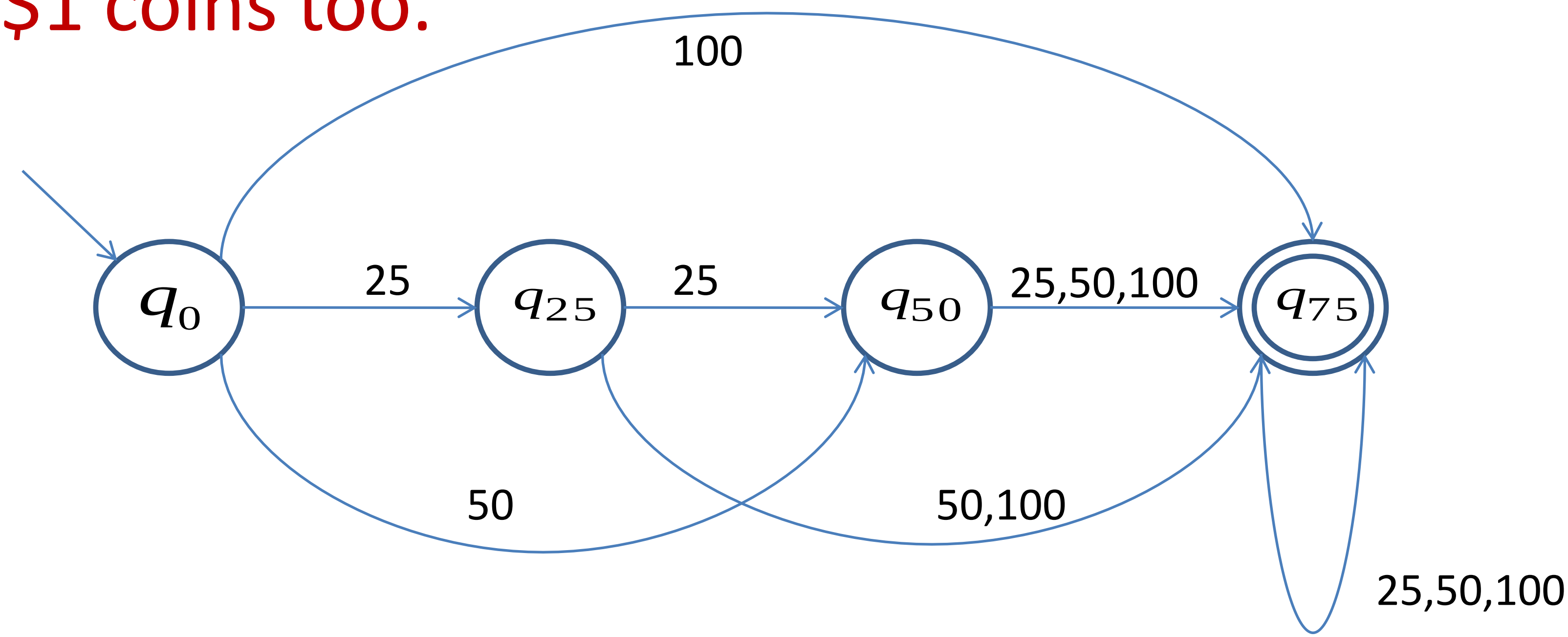
- The second washing machine accepts 50 cents coins as well.



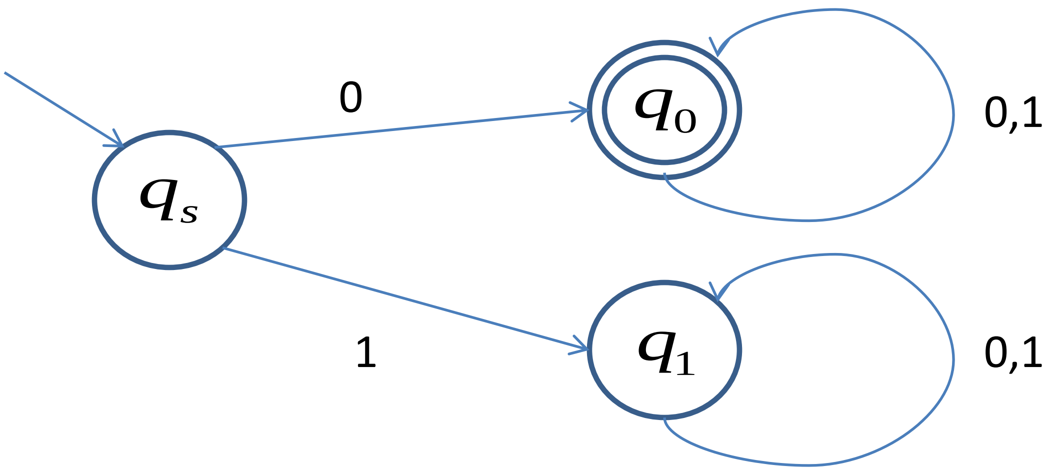
- Accepted words: 25,25,25; 25,50; …

Finite Automata - A Short Example

- The second washing machine accepts 50 cents coins as well.
- The most complex washing machine accepts \$1 coins too.



Finite Automaton - An Example

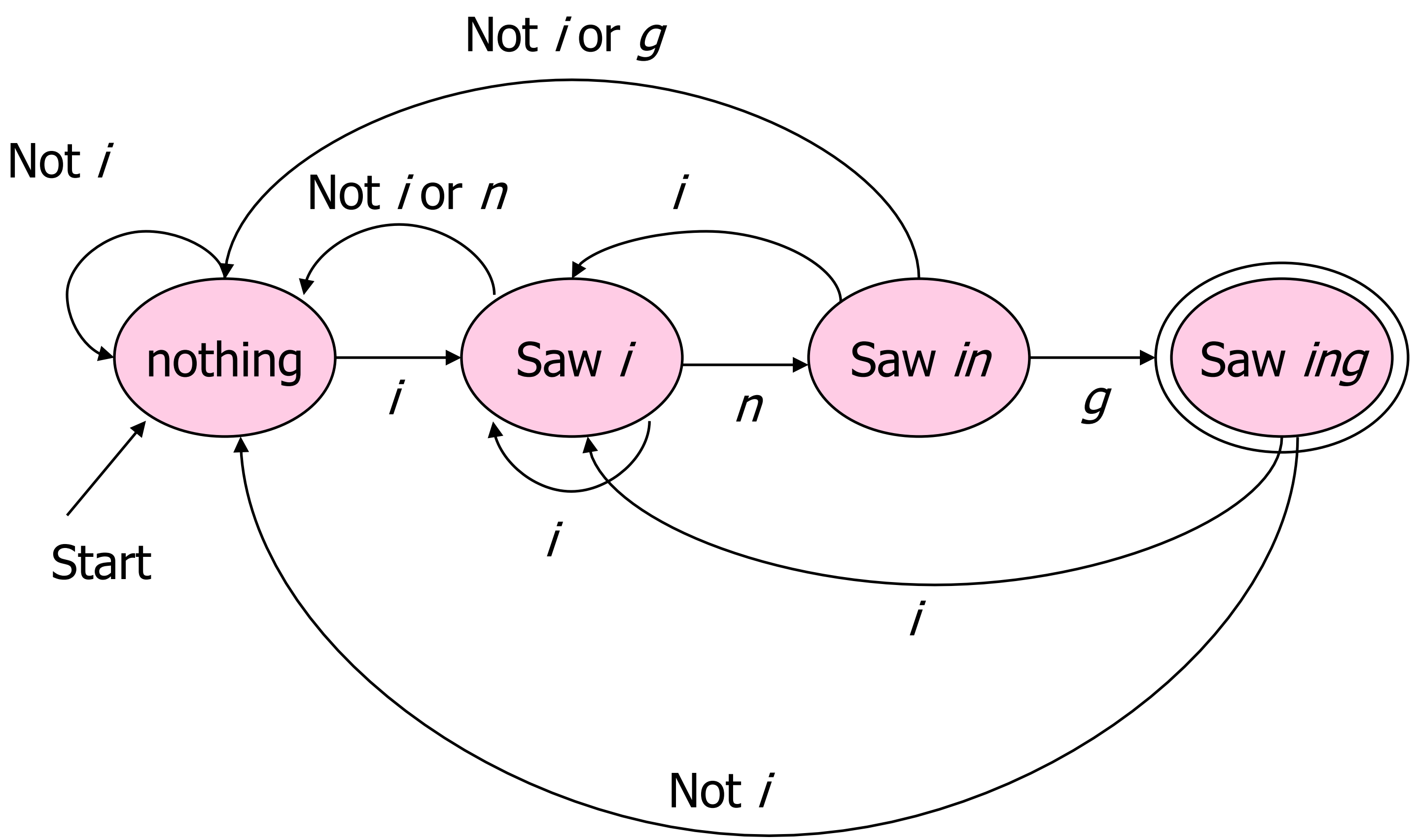


States: $Q = \{q_s, q_0, q_1\}$

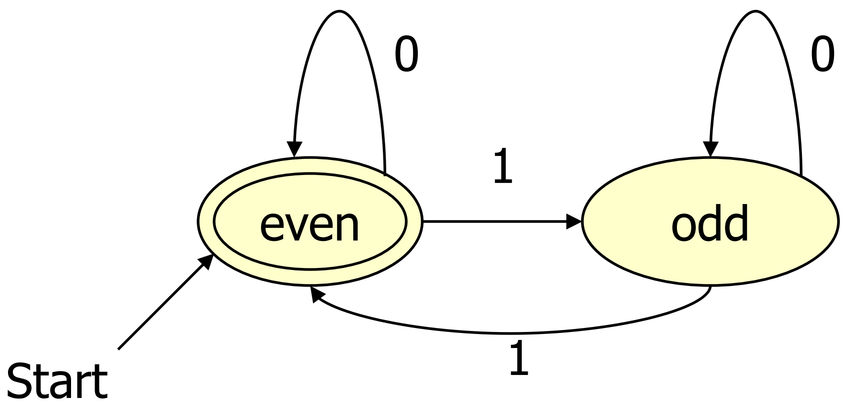
Initial State: q_s

Final State: q_0

Example: Recognizing Strings Ending in "ing"



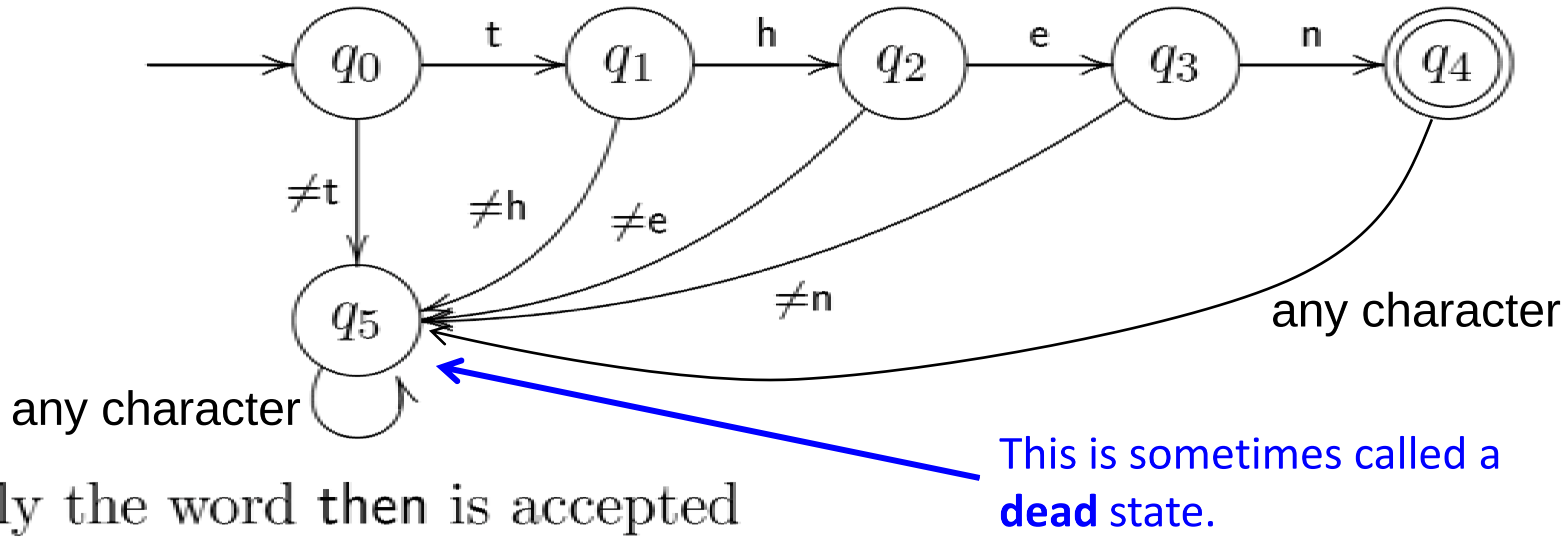
Example: An Even Number of 1's



- How would it look to accept a number of 1's that is a multiple of 3?

Password/Keyword Example

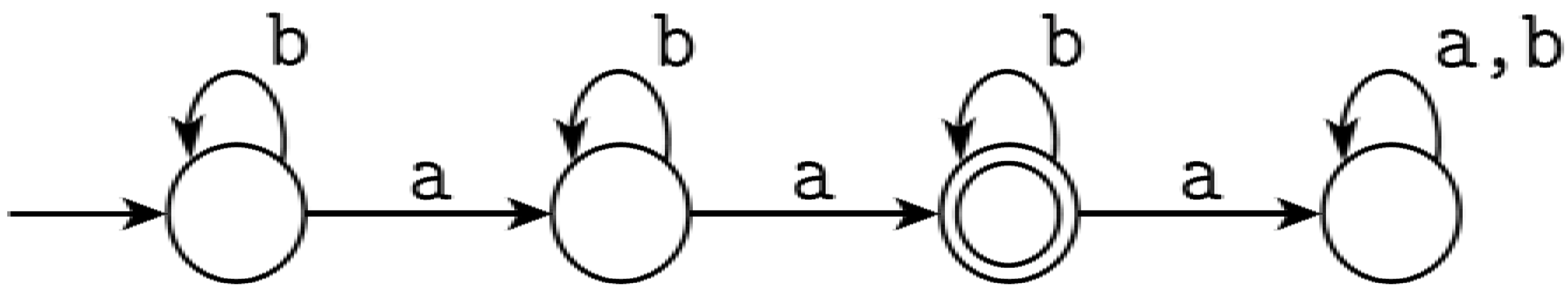
It reads the word and accepts it if it stops in an accepting state



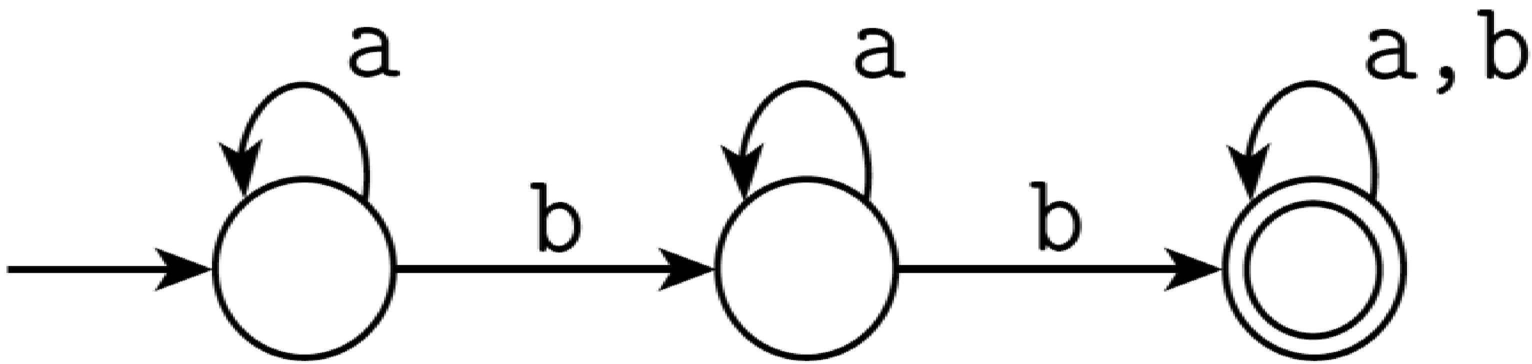
Only the word then is accepted

BTW, there is a potential security risk on the password application if this finite automaton reports failure too quickly.

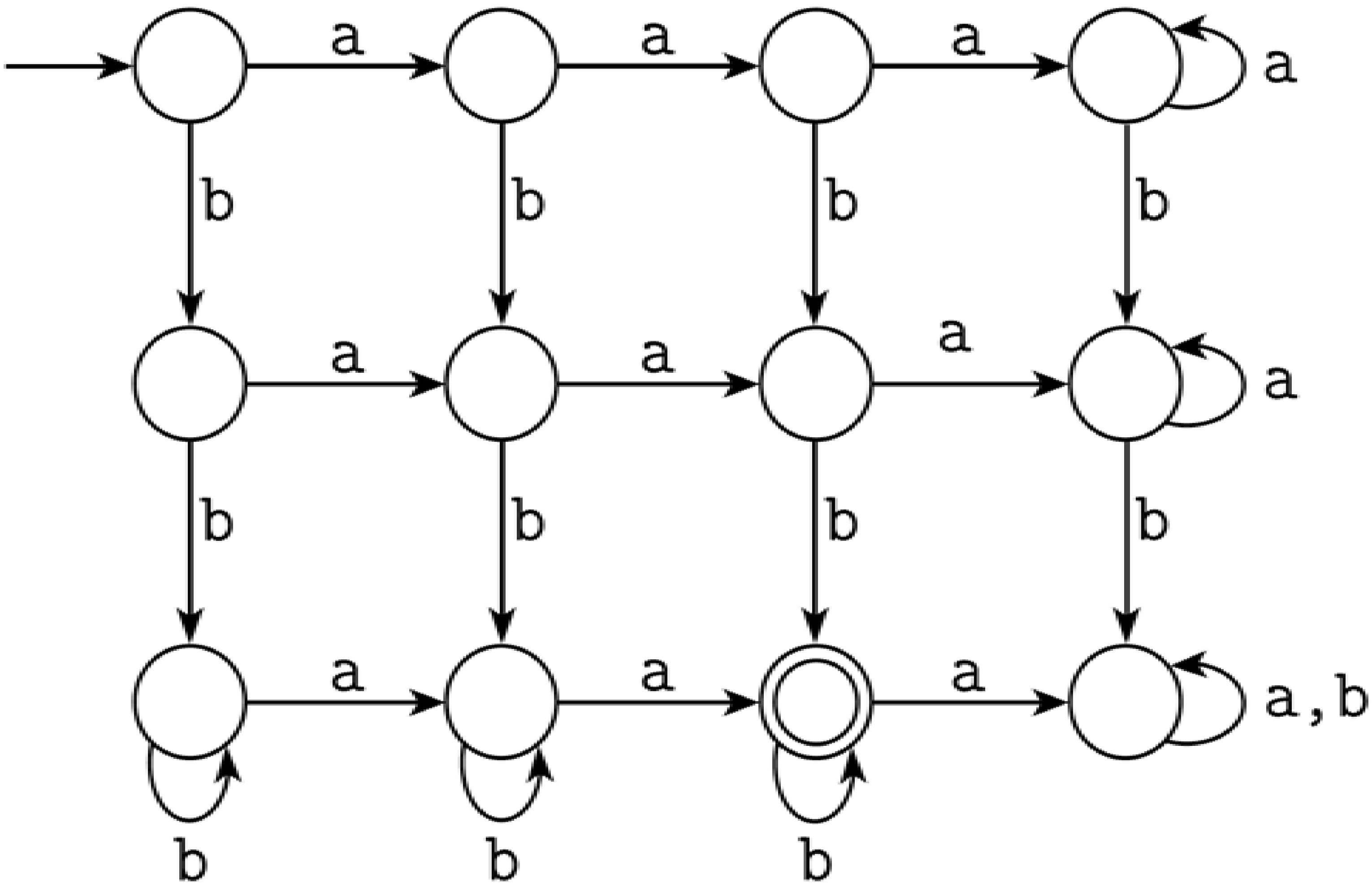
Exactly Two a's



At Least Two b's

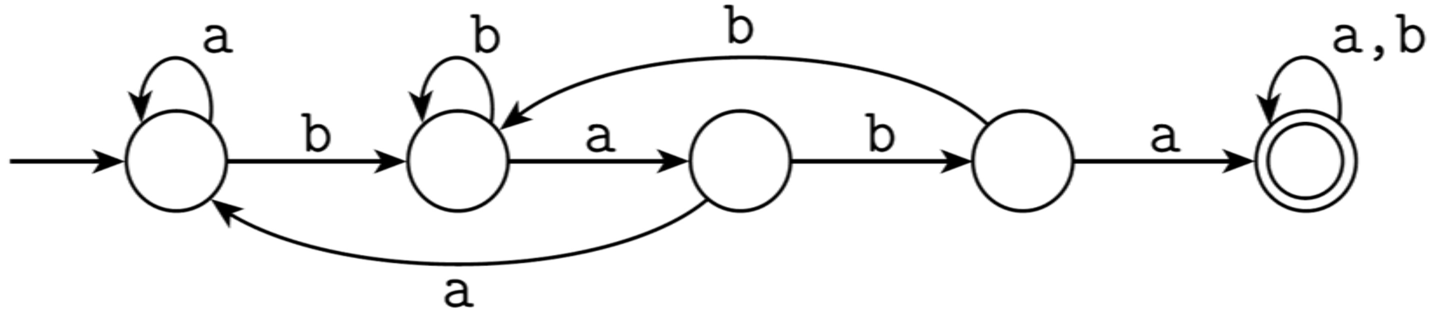


Exactly two a's and at least two b's

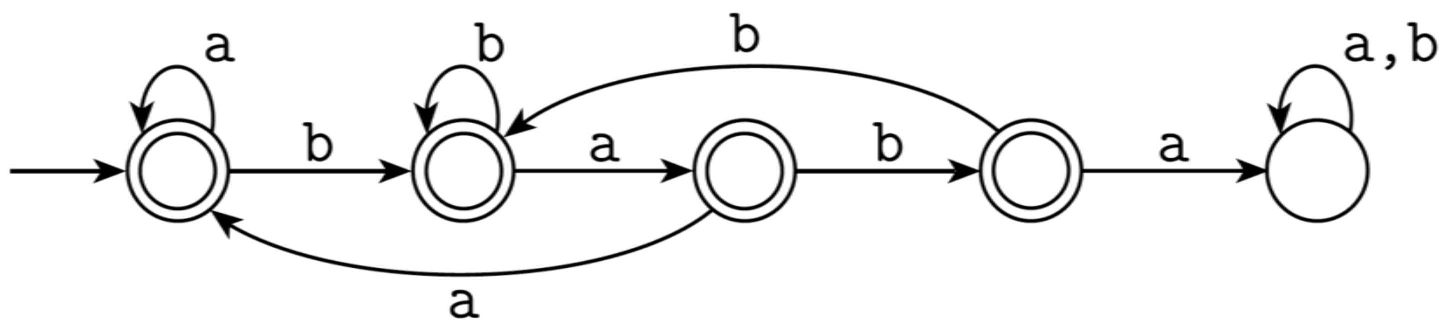


Containing Substrings or Not

- Contains baba:



- Does not contain baba:



Introduction to Finite Automata

Languages

Deterministic Finite Automata

Representations of Automata

Alphabets

- An *alphabet* is any finite set of symbols.
- Examples: ASCII, Unicode, {0,1} (*binary alphabet*), {a,b,c}, {a,b}.

Strings

- The set of *strings* over an alphabet Σ is the set of lists, each element of which is a member of Σ.
  - Strings shown with no commas, e.g., abc.
- Σ* denotes this set of strings.
- ϵ or λ stands for the *empty string* (string of length 0).
- {0,1}* = {λ, 0, 1, 00, 01, 10, 11, 000, 001, . . . }
- Subtlety: 0 as a string, 0 as a symbol look the same.
  - Context determines the type.

Languages

- A *language* is a subset of Σ^* for some alphabet Σ.
- Example: The set of strings of 0's and 1's with no two consecutive 1's.
- L = {λ, 0, 1, 00, 01, 10, 000, 001, 010, 100, 101, 0000, 0001, 0010, 0100, 0101, 1000, 1001, 1010, . . . }

Deterministic Finite Automata (DFA)

- A FA is represented by a 5-tuple $M = (Q,\Sigma,\delta, q_0, F)$ where :

  Q is a finite set of *states,*

  Σ is finite non-empty set of *input alphabet*

  $\delta : Q \times \Sigma \rightarrow Q$ is a *transition function*

  q_0, in Q is a *start state*

  $F \subseteq Q$ is a set of *final states.*

  - "Final" and "accepting" are synonyms.
- Takes two arguments: a state and an input symbol.
- $\delta(q, a) = q'$, the state that the DFA goes to when it is in state *q* and input *a* is received.

Graph Representation of DFA’s

- Nodes = states.
- Arcs represent transition function.
  - Arc from state p to state q labeled by all those input symbols that have transitions from p to q.
- Arrow labeled “Start” to the start state.
- Final states indicated by double circles.

Representations of a FA

1. State Transition Table; 2. State Transition Diagram

TABLE 1

	f	
	Input	
State	**0**	**1**
s_0	s_0	s_1
s_1	s_0	s_2
s_2	s_0	s_0
s_3	s_2	s_1

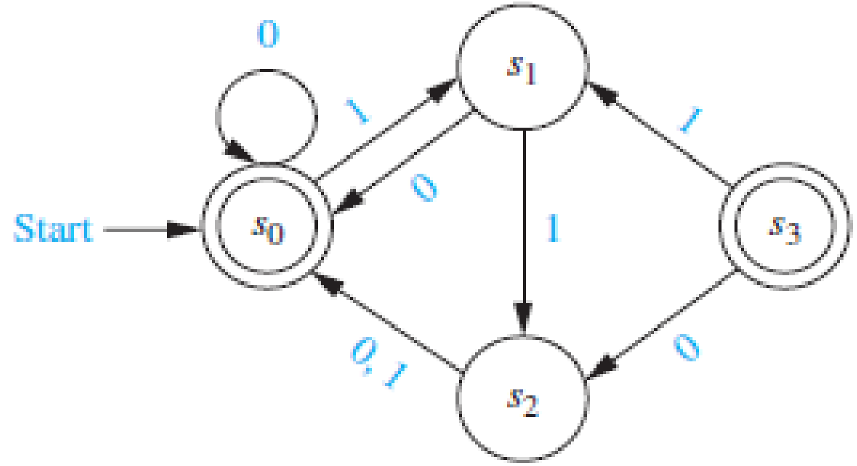


FIGURE 1 **The State Diagram for a Finite-State Automaton.**

3. Representation by Delta rules

δ(s0,0)= s0; δ(s0,1)=s1;

δ(s1,0)=s0; δ(s1,1)=s2;

δ(s2,0)=s0; δ(s2,1)=s0;

δ(s3,0)=s2; δ(s3,1)=s1;

Extended Transition Function

- We describe the effect of a string of inputs on a DFA by extending δ to a state and a string.

- Induction on length of string.
- Basis: $\delta(q, \epsilon) = q$
- Induction: $\delta(q,wa) = \delta(\delta(q,w),a)$
  - w is a string; a is an input symbol.

Extended δ: Intuition

- Convention:
  - … w, x, y, z are strings.
  - a, b, c,… are single symbols.
- Extended δ is computed for state q and inputs $a_1a_2…a_n$ by following a path in the transition graph, starting at q and selecting the arcs with labels a_1, a_2,…,a_n in turn.

Example: Extended Delta

	0	1
A	A	B
B	A	C
C	C	C

δ(B,011) = δ(δ(B,01),1) = δ(δ(δ(B,0),1),1) =

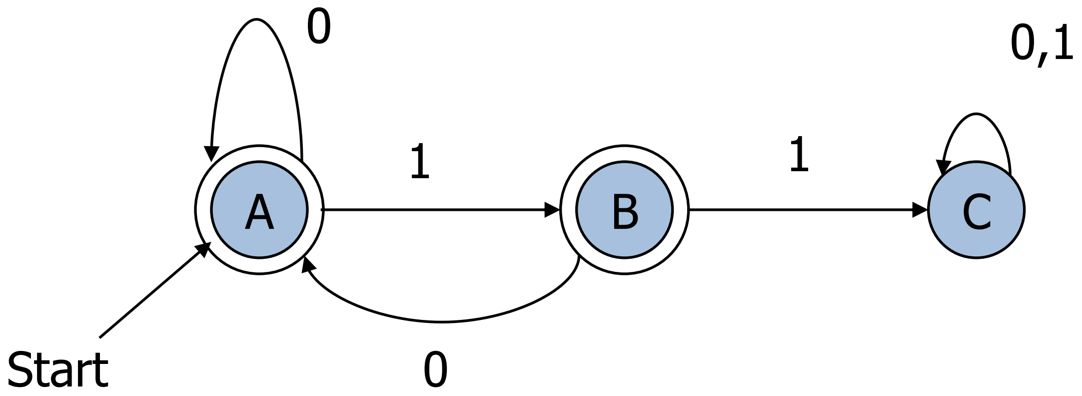
δ(δ(A,1),1) = δ(B,1) = C

Language of a DFA

- Automata of all kinds define languages.
- If M is an automaton, T(M) is its language.
- For a DFA M, T(M) is the set of strings labeling paths from the start state to a final state.
- Formally: $T(M) = \{ w \,/\, \delta(q_0, w) \in F \}$

Example: String in a Language

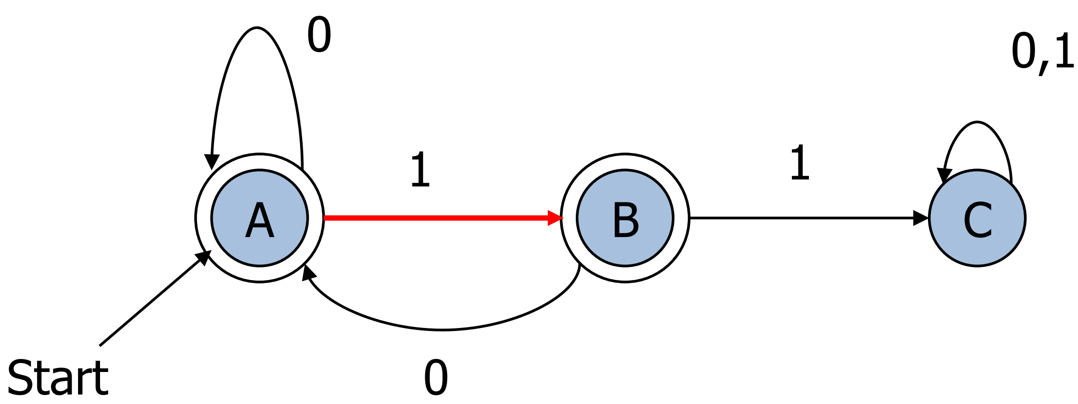
String 101 is in the language of the DFA below.
Start at A.



Example: String in a Language

String 101 is in the language of the DFA below.

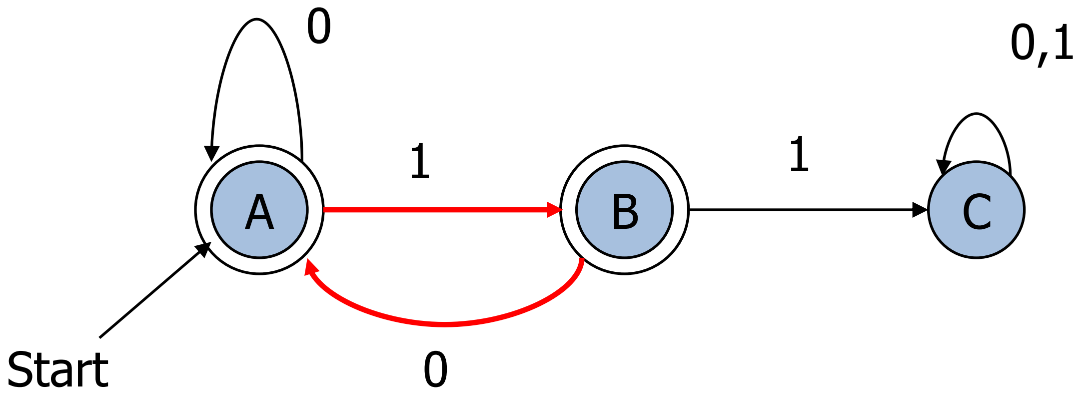
Follow arc labeled 1.



Example: String in a Language

String 101 is in the language of the DFA below.

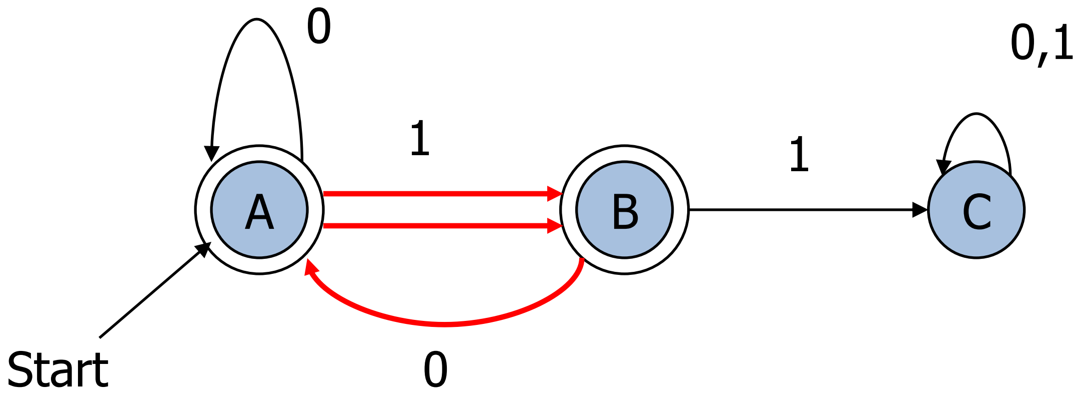
Then arc labeled 0 from current state B.



Example: String in a Language

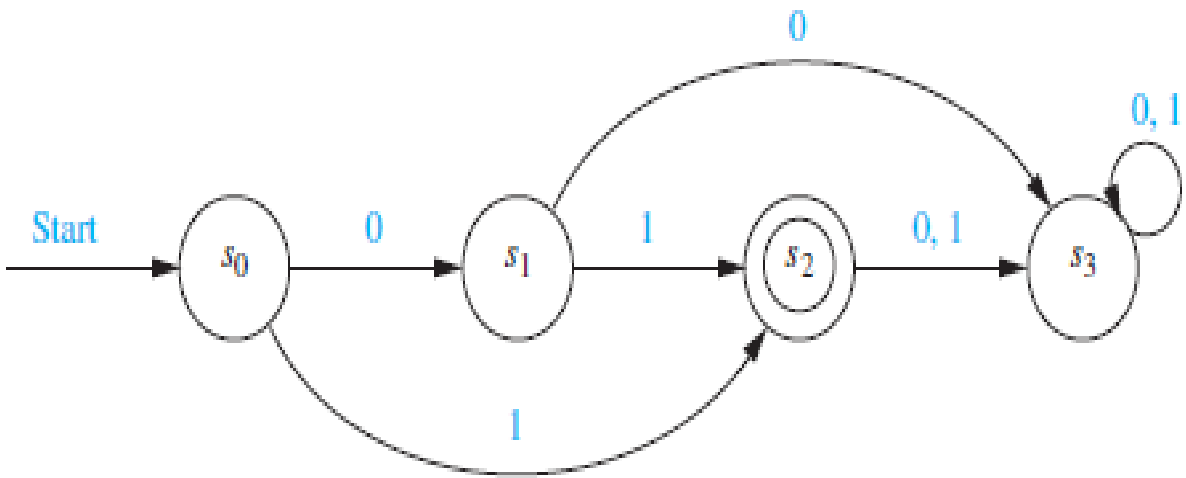
String 101 is in the language of the DFA below.

Finally arc labeled 1 from current state A. Result is an accepting state, so 101 is in the language.

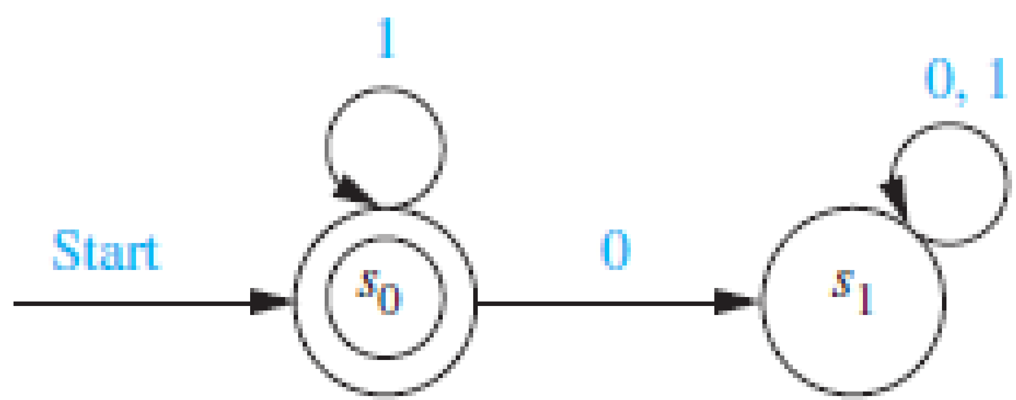


Examples

- $T(M) = \{1, 01\}$

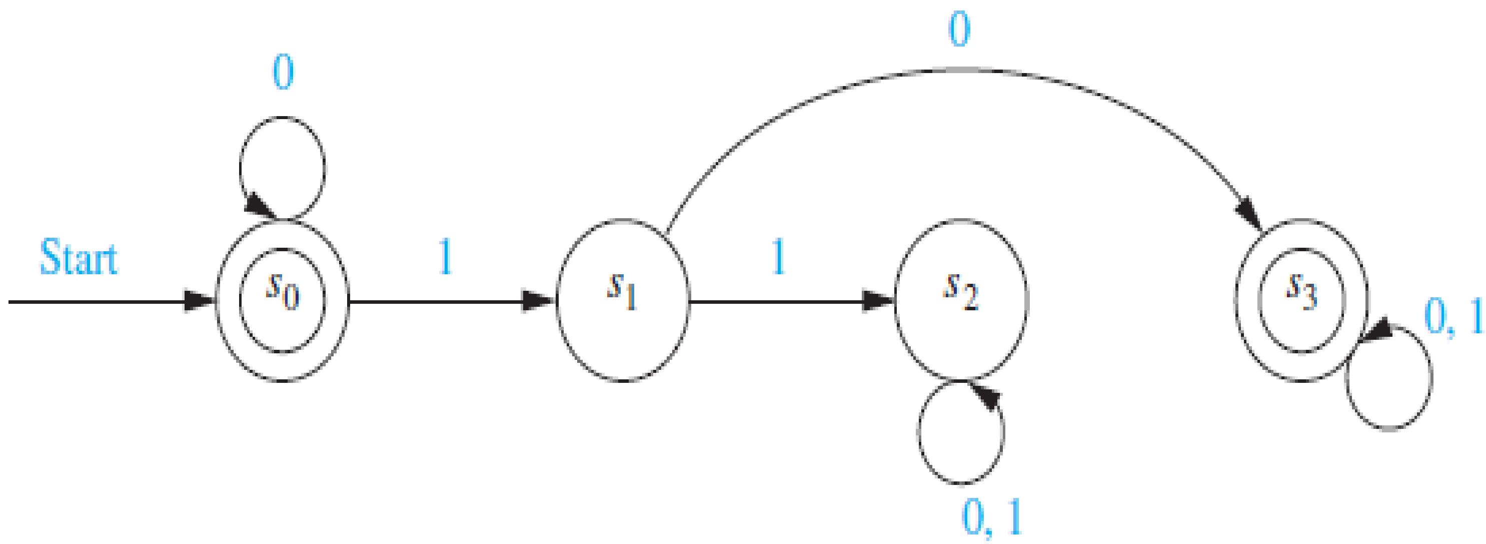


- $T(M)=\{0^n / n>=0\}$



Examples

- $T(M)=\{0^n , 0^n 10x$ / $n>=0$, x is any string of $\{0,1\}\}$



Examples

- T(M)={ do not contain 2 consecutive zeroes}