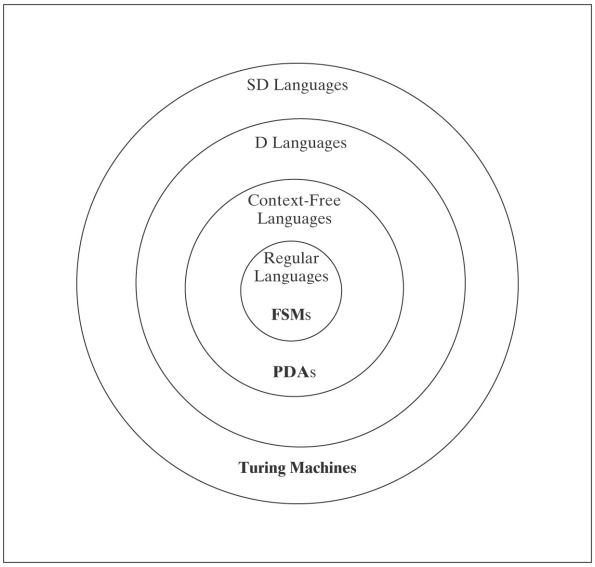
Finite State Machines
Regular Languages

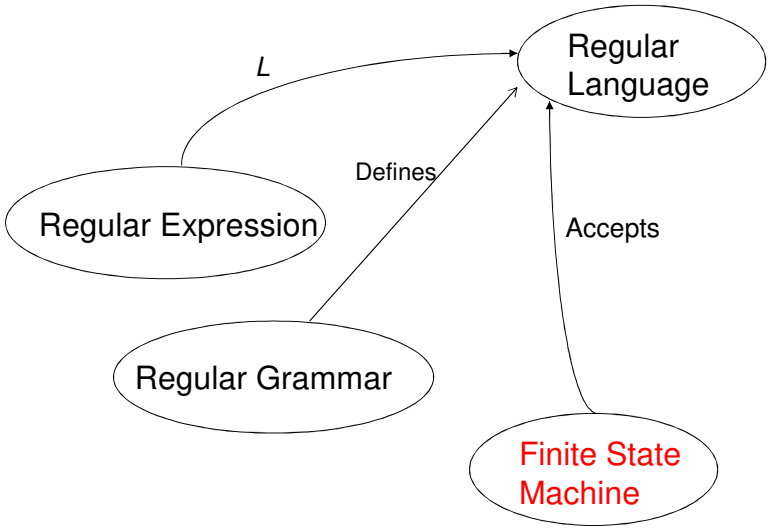
1

Languages and Machines



2

Regular Languages



3

Finite State Machine (FSM)

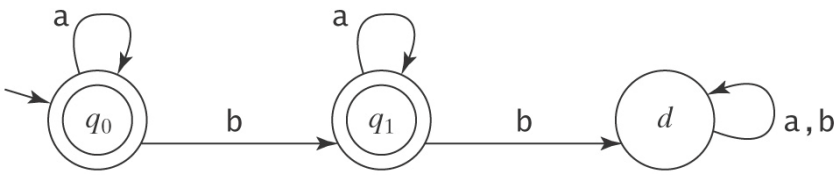
Non-technical definition

Finite State Machine (FSM)

- Computational device having a string as input and having one of 2 possible outputs – *Accept* or *Reject*

4

Example Finite State Machine



Incoming arrow indicates start state
Double Circles → Accepting state
Single Circles → Rejecting state
Note: time required <= |string|

6

Finite State Machine Example

- Drinks in a vending machine cost Rs 25
- Accepts Rs 5 (N), Rs 10(D), Rs 25(Q) coins
- Insert coins, push "DRINK" button, drink is dispensed if "Enough Money"
- LANGUAGE: String of coin sequences adding to Rs. 25 (or more) allowing drink to be dispensed
  - {Q,DDN, NNND, DNNN,DND, etc…}
  - Actually, S can be inserted within any string
  - What about extra money (change)?

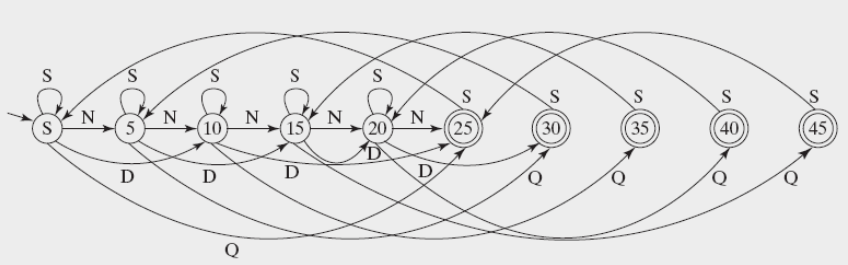
7

FSM for Drink Machine

An FSM to accept Rs 25 in change for a drink
Note that states are "labeled" for clarity
Machine has ERRORS!! Can you find them??

But you get the idea!

Should there be a "50" state? Why or why not?



N=5, D=10, Q=25, S= Push Drink Button

8

Formal Definition of a DFSM

Deterministic Finite State Machine (DFSM) *is M:*

$M = (K, \Sigma, \delta, s, A)$, where:

K is a finite set of states
Σ is an alphabet
$s \in K$ is the initial state
$A \subseteq K$ is the set of accepting states, and
δ is the transition function from $(K \times \Sigma)$ to K

***The set of ALL strings accepted by M form the *Language* defined by the DFSM.**

9

Configurations of DFSMs

A ***Configuration*** of a DFSM *M* is an element of
$K \times \Sigma^*$

Configuration captures the two things that make a difference to *M*'s future behavior:

- its current state
- the input that remains to be read.

The ***Initial Configuration*** of a DFSM *M*, on input *w*, is (s_M, w) , where s_M is start state of M

10

Transition Function - δ

- Defines the DFSM, one state at a time
- δ is set of all pairs of states in M & characters in ∑

(Current State, Current Character) → New State

Note: we often reduce the set δ to those pairs that are "useful" or to only those that can lead to an *Accepting* state

11

Complete vs. Incomplete FSM

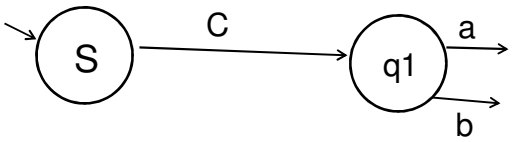
- Complete FSM
  - A transition is defined for every possible state and every possible character in the alphabet
  - Note: Can cause FSM to be larger than necessary, but ALWAYS processes the entire string
- Incomplete FSM
  - One which defines a transition for every possible state & every possible character in the alphabet which can lead to an accepting state
  - Note: If no transition is defined, the string is *Rejected*

12

Complete vs. Incomplete Example

∑ = {a, b, c}

L = {strings beginning with c & followed by only a's & b's, at least 1}



13

The Yields Relations

The *yields-in-one-step* relation $|-_M$
$(q, w) |-_M (q', w')$ iff
- $w = a\ w'$ for some symbol $a \in \Sigma$, and
- $\delta(q, a) = q'$

$|-_M{}^*$ is the reflexive, transitive closure of $|-_M$.

Transitions defined by processing ONE character

M refers to the particular machine, so can leave it off, usually.

14

Computations Using FSMs

A ***Computation*** by M is a finite sequence of configurations $C_0, C_1, \ldots, C_n$ for some $n \geq 0$ such that:

*C_0 is an initial configuration,

*C_n is of the form (q, ε), for some state $q \in K_M$
- ε indicates empty string, entire string is processed & implies a complete DFSM

*$C_0 |-_M C_1 |-_M C_2 |-_M \ldots |-_M C_n$.

However, M **Halts** when the last character has be processed or a next transition is not defined

15

Accepting and Rejecting

A DFSM *M* ***Accepts*** a string *w* iff
$(s, w) \vdash_M^* (q, \varepsilon)$, for some $q \in A_M$.

A DFSM *M* ***Rejects*** a string *w* iff
$(s, w) \vdash_M^* (q, \varepsilon)$, for some $q \notin A_M$.

The ***language accepted by M***, denoted *L*(*M*), is the set of all strings accepted by *M*.

16

Example Language

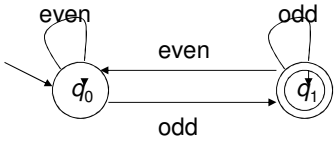
L = {x | x is an odd integer}

- Is 3456 ∈ L?
- How will we develop a machine to answer the question?
- Process characters left to right
- Assume the current character is the LAST character
- What are the necessary states?

17

An Example Computation

An FSM to accept odd integers:



On input 235, the configurations are:

$(q_0, 235)$ |- $(q_0, 35)$
$(q_0, 35)$ |- $(q_1, 5)$
$(q_1, 5)$ |- (q_1, ε) *Accept*

Thus $(q_0, 235)$ |-* (q_1, ε) [* means 0 or more steps]

18

Developing a DFSM

Heuristic – a generality, a recommendation

- List or Label the possible states of a machine based upon the language or problem definition
- Determine the transitions between the states

19

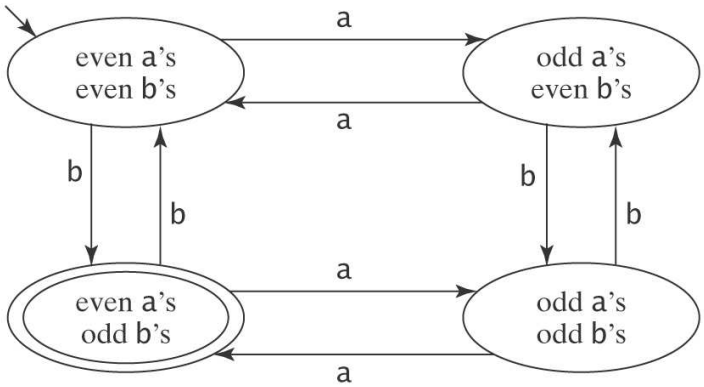
Developing DFSM Examples

- L = {x | x contains an even number of a's}
  - What would the states of the DFSM be?
  - What would the transitions be?
- L = {x | x contains an even number of a's & odd number of b's}
  - What would the states of the DFSM be?
  - What would the transitions be?

20

Even a's Odd b's

Let $L = \{w \in \{a, b\}^* : w$ contains an even number of a's and an odd number of b's$\}$



21

Regular Languages

A language is ***regular*** iff it is accepted by some FSM.

We have defined several regular languages…

22

Regular Examples

L = {w ∈ {a,b}*| every region of a's is of even length}

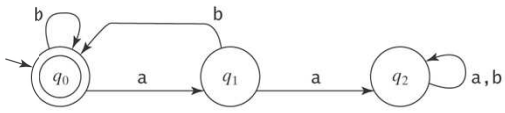
L = {w ∈ {a,b}*| w contains at most 1 b }

L = {w ∈ {a,b}*| no 2 consecutive characters are the same }

23

A Very Simple Example

$L = \{w \in \{a, b\}^* \mid$ every a is immediately followed by b$\}$.



$$\delta = \{((q_0, a), q_1), ((q_0, b), q_0), ((q_1, a), q_2), ((q_1, b), q_0), ((q_2, a), q_2), ((q_2, b), q_2))\}.$$

$$M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_0\})$$

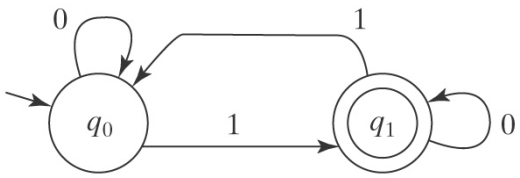
- Is the empty string a member of the language?

24

Parity Checking Example

$L = \{w \in \{0, 1\}^* : w$ has odd parity$\}$.

A binary string has odd parity iff the number of 1's in it is odd.

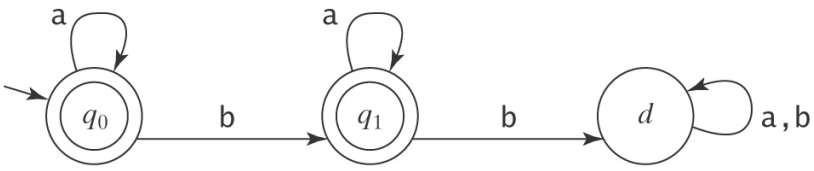


Complete or incomplete DFSM?

25

No More Than One b

$L = \{w \in \{a, b\}^* : w$ contains no more than 1 b$\}$



Complete or incomplete DFSM?

26

Error States or Dead States

An Error State or Dead State is a rejecting state from which the string can never go back to an Accepting State.

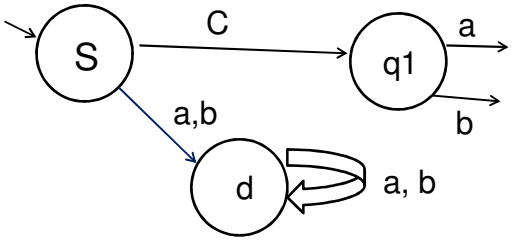
Some DFSM can have, others cannot!

27

Error States or Dead States

∑ = {a, b, c}

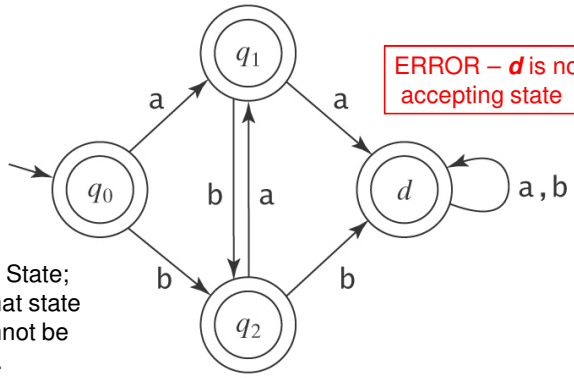
L = {strings beginning with c & followed by only a's & b's, at least 1}



28

Checking Consecutive Characters

$L = \{w \in \{a, b\}^* |$ no two consecutive characters are the same$\}$.



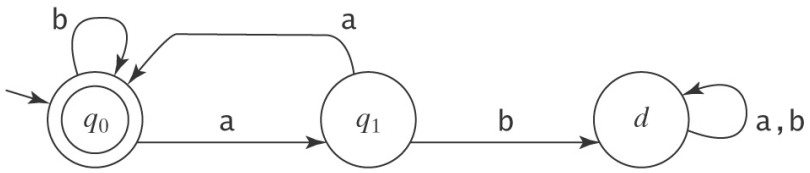
ERROR – ***d*** is not an accepting state

d is Dead State; once in that state string cannot be accepted.

29

DFSM Example

L =
{$w \in$ {a, b}* : every a region in w is of even length}



d is a Dead State

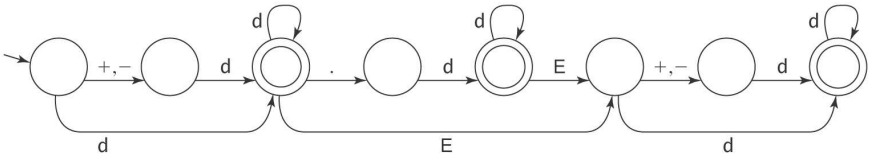
30

The Language of Floating Point Numbers is Regular

Example strings:

+3.0, 3.0, 0.3E1, 0.3E+1, −0.3E+1, −3E8

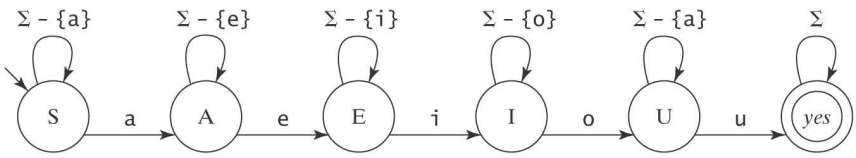
The language is accepted by the DFSM:



31

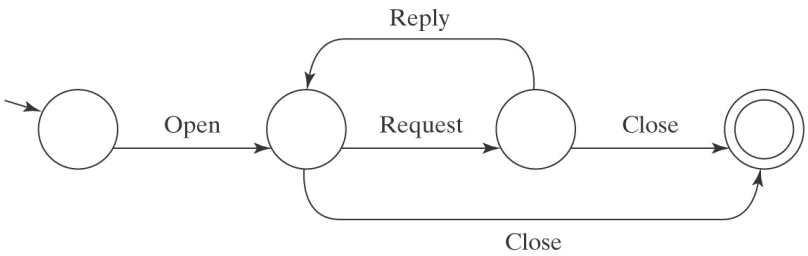
Vowels in Alphabetical Order

$L = \{w \in \{a - z\}^* :$ all five vowels, a, e, i, o, and u, occur in w in alphabetical order$\}$.



32

A Simple Communication Protocol

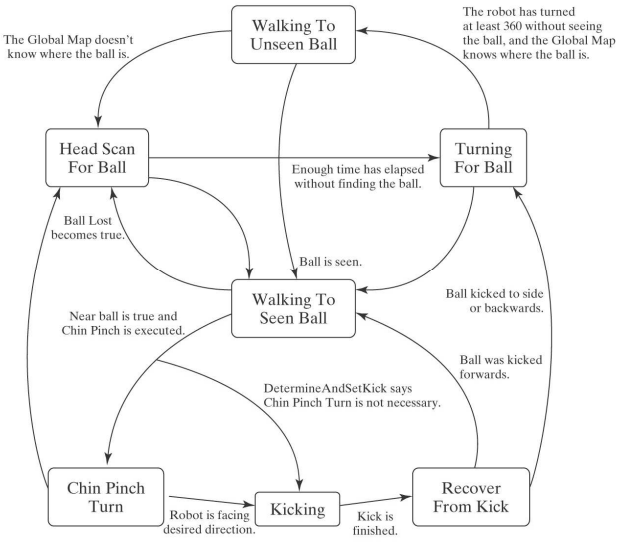


33

Controlling a Soccer-Playing Robot

34

A Simple Controller



35

Developing FSMs

Sometimes, it is "easier" to develop the FSM for the complement of the language, then reverse the *Accept* & *Reject* states

Do you believe this would work???

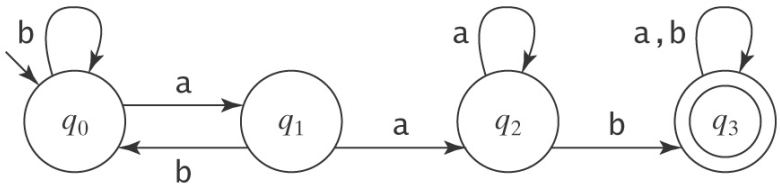
$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ does not contain the substring `aab`$\}$.

36

Complement of FSMs

$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ does not contain the substring `aab`$\}$.

Start with a machine for $\neg L$



How must it be changed?

37

The Missing Letter Language

Let Σ = {a, b, c, d}.

Let $L_{Missing}$ =
{w : there is a symbol $a_i \in \Sigma$ not appearing in w}.

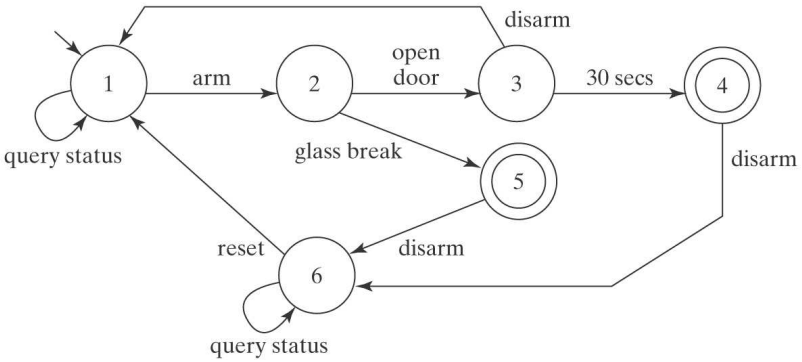
Try to make a DFSM for $L_{missing}$

First develop machine for set of strings containing ALL 4 characters, then reverse.

38

A Building Security System

L = {event sequences such that the alarm should sound}



39

FSMs Predate Computers

The Prague Orloj, originally built in 1410.

40