CSE 322, Fall 2010

(Deterministic)
Finite State Machines

Finite State Automaton (FSA)

5 pieces

- States
- Alphabet
- Transitions
- Start
- Final or Accept

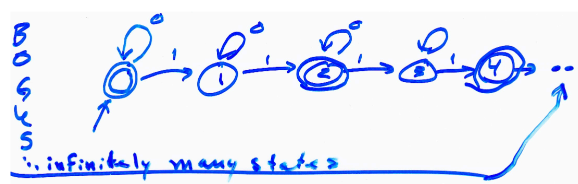
An Example: Even Parity

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

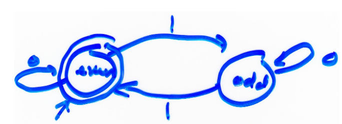
$L = \{ w \in \Sigma^* \mid \#$ of 1's in w is even $\}$

An Example: Even Parity

- The "obvious" algorithm: first count the 1's, then decide whether the count is even:



- It works, but is not a finite state machine. This is:



Formal definition

A finite state machine

$M = (Q, \Sigma, \delta, q_0, F)$

where

Q is a finite set (states)

$q_0 \in Q$ start state

Σ is a finite set (alphabet)

$F \subseteq Q$ Final states / Accepting states

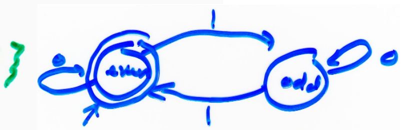
$\delta : Q \times \Sigma \to Q$ transition function

function

Formal version of parity, I

$M_{parity} = (Q, \Sigma, \delta, q_0, F)$

where



$Q = \{\text{even}, \text{odd}\}$

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

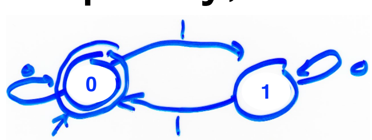
$q_0 = \text{even}$ (one element)

$F = \{\text{even}\}$ (a set containing one element)

$\delta(q, a)$:

q \ a	0	1
even	even	odd
odd	odd	even

Formal version of parity, II



Even more succinctly

if we let $Q = \{0, 1\}$ also

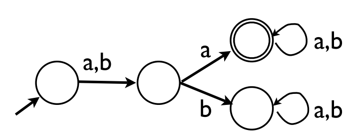
then $\delta(q, a) = (q + a) \bmod 2$

for all q in Q and all a in Σ

Example

$\Sigma = \{ a, b \}$

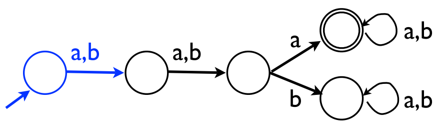
$L = \{ w \mid \text{2nd letter of } w \text{ is "a"} \}$



Example

$\Sigma = \{ a, b \}$

$L = \{ w \mid$ 3rd letter of w is "a" $\}$

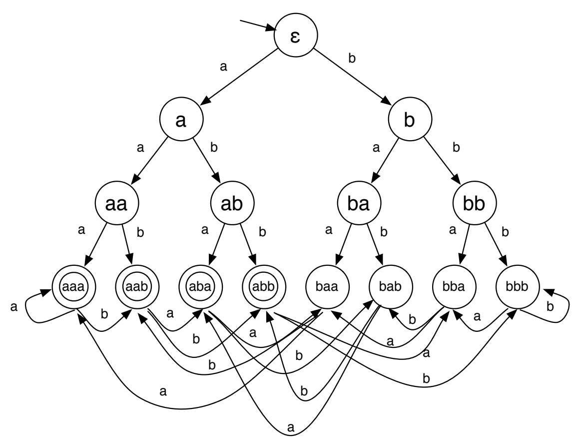


$\Sigma = \{ a, b \}$

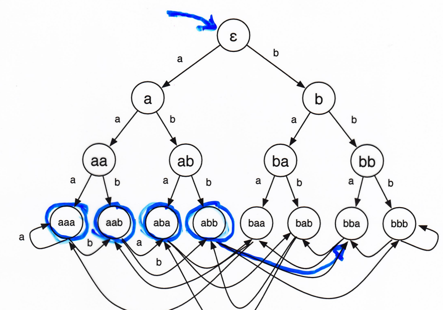
$L = \{ w \mid 3^{rd}$ letter from the right end of w is "a" $\}$

epsilon	N
a	N
b	N
aa, ab, ba, bb	N
aaa	Y
aab	Y
baa	N
bbb	N
...	...

L = { w in {a,b}* I 3rd letter from the right end of w is "a" }



L = { w in {a,b}* I 3rd letter from the right end of w is "a" }



← a "shift register"

$M = (Q, \Sigma, \delta, q_0, F)$

$\Sigma = \{a, b\}$

$Q = \{ w \in \Sigma^* \mid |w| \le 3 \}$

$q_0 = \varepsilon$

$F = \{ w \in \Sigma^* \mid w = ax, |x| = 2 \}$

$\forall w \in Q$ $\forall c \in \Sigma$ $\delta(w, c) =$ Last 3 letters of w.c

DEFN ("is in state q")

M ends in state q after reading $w \in \Sigma^*$ if

(1) $w = w_1 w_2 \cdots w_n$ where $w_i \in \Sigma$

(2) $\exists$ states $r_0, r_1, r_2 \cdots r_n \in Q$ s.t.

(a) $r_0 = q_0$

(b) $\forall\, 1 \le i \le n$ $\delta(r_{i-1}, w_i) = r_i$

(c) $r_n = q$

Fact: q is unique
because δ is a function, basically

Exercise: what state is M in after reading ε?

Defn

M accepts $w \in \Sigma^*$ $\Leftrightarrow$ the state, q, reached by M after reading w is an accepting state, i.e., $q \in F$.

And M rejects w iff $q \notin F$

Defn

The language recognized by M, $L(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$.

Strings are accepted/rejected

Languages are recognized (or not)

Note

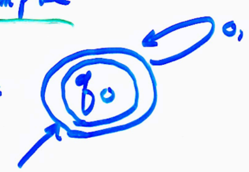
Every M recognizes exactly one language. Implicitly, it "recognizes" both strings it must accept and those it must reject.

Example

M:

Example

M:



$L(M) = \Sigma^*$

Example

M:

$L(M) = \Sigma^*$

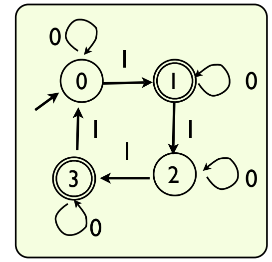
$L_{pal} = \{ w \in \{0,1\}^* \mid w = w^R \}$

e.g. 101 and 001100 are palindromes
110 is not

M above accepts every palindrome
$\therefore L_{pal} \subseteq L(M)$

but M also accepts some (in fact, all) non palindromes
$\therefore L_{pal} \neq L(M)$

An example



Defn for any a in Σ, w in Σ^*
$\#_a(w)$ is the number of instances of the symbol a in the string w

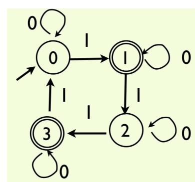
E.g. $\#_1(1011) = 3$

M = ({0,1,2,3}, {0,1}, δ, 0, {1,3}) where

$\delta(i,0) = i$
$\delta(i,1) = (i+1) \bmod 4$

What does M do?

Claim: $\forall w \in \Sigma^*$, the state M is in after reading w ("$\delta(0,w)$") is $(\#_1(w)) \bmod 4$



[Isn't this just the defn of δ? No; $w \in \Sigma^*$, not Σ]

Proof: By induction on |w|

Basis (|w| = 0): then $w=\varepsilon$, and $\#_1(\varepsilon)=0$, and by definition of "state M is in...", M is in its start state, namely state 0.

Ind hyp: For some n > 0, assume the statement in the claim is true for all strings w of length < n.

Ind: Let w be a string of length n. Since every non-ε string has a last letter, w=xa for some a in Σ, and some string x of length <n. Let $i=(\#_1(x)) \bmod 4$. I.H. applies to x, so we may assume M is in state i after reading x. By def of δ and "state reached after reading a string," after reading w=xa, M is in state $\delta(i,a)$. Two cases, depending on a (and δ):

case 1: a=0. Then $\delta(i,a)=i$, and $\#_1(xa) = \#_1(x) \equiv i \bmod 4$
case 2: a=1. Then $\delta(i,a)=(i+1) \bmod 4$, and
$\#_1(xa) = \#_1(x)+1 \equiv i+1 \pmod 4$

Which establishes the claim.

- Corollary: the *language recognized* by M is {w in $\{0,1\}^*$ | $\#_1(w) \bmod 4$ = 1 or 3 }. Equivalently, $\#_1(w)$ is odd.

  Proof: by claim, exactly these strings cause M to end in state 1 or 3, which are its only final states

- Note: it's important that the claim above *ignored* final states. E.g., if we changed the set of final states to, say, {1,2} then the claim is still valid (tho the corollaries above would need to be adjusted accordingly).

Compare above to:

```
int i = 0;
while(! end_of_file){
  char a = get_char_from_file;
  if( a == '1') { i = i+1;}
}
print i;
```

Compare above to:

```
int i = 0;
```

claim: i == 0

```
while(! end_of_file){
  char a = get_char_from_file;
  if( a == '1') { i = i+1;}
```

claim: i == $\#_1$ read so far

```
}
print i;
```

claim: i == $\#_1$ in file

The message

- A program is a finite, static thing
- But to understand it, you need to reason about its dynamic behavior in *infinitely* many situations
- Like it or not, you do induction on loops (and recursions) all the time

Prefix

x is a prefix of w

if $\exists$ y s.t. w = x y (w, x, y in Σ*)

E.g.

prefixes of abb are

ε, a, ab, abb

Facts

ε is always a prefix

every w is a prefix of itself

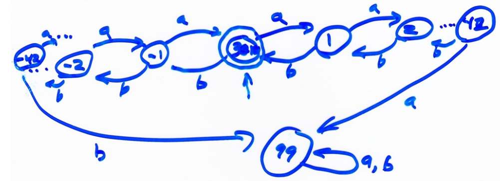
if |w| = n then w has n+1 prefixes

Another Induction Example

$\Sigma = \{a, b\}$

$f(w) = \#_a(w) - \#_b(w)$

$L_{eq} = \{ w \mid f(w) = 0 \}$



$L = \{ w \mid f(w) = 0 \ \& \ \forall \text{ prefix } x \text{ of } w \ |f(x)| \le 42 \}$

$$g(w) = \begin{cases} f(w) & \text{if } |f(x)| \le 42 \text{ for all prefixes } x \text{ of } w \\ 99 & \text{o.w.} \end{cases}$$

$Q = \{-42, -41, \dots, 41, 42, 99\}$

$q_0 = 0$, $F = \{0\}$

$$\delta(q, c) = \begin{cases} q+1 & \text{if } c = a,\ q < 42 \\ q-1 & \text{if } c = b,\ q > -42,\ \neq 99 \\ 99 & \text{o.w.} \end{cases}$$

5-

Claim $\forall w \in \Sigma^*$ the state reached by M after reading w is $q = g(w)$

corr. M accepts L (but *not* L_{eq})

pf M accepts w $\Leftrightarrow$ M ends in F (by defn + constr.)

$\Leftrightarrow$ M ends in 0

$\Leftrightarrow$ $0 = g(w)$ (by claim)

$\Leftrightarrow$ $w \in L$ (by defn.)

Claim $\forall w \in \Sigma^*$, state reached by M after reading w is $g(w)$

$P(n)$: $\forall w \in \Sigma^n$ state .... is $g(w)$

To prove $\forall n \ge 0\ P(n)$

Basis $n = 0$ $w = \varepsilon$

M reaches state 0 on ε by construction

$g(\varepsilon) = 0$ by inspection

say more

Ind $P(n) \Rightarrow P(n+1)$

let w be of length n+1

$w = xc$ for some $c \in \Sigma$, $x \in \Sigma^n$

Case 1, $c = a$

(a) $g(x) = 99$

M is in 99 after reading x (by I.H.)

$\delta(99, a) = 99$ (by const.)

$g(x \cdot a) = 99$ ← argue based on $g(x) = 99$

$\therefore P(n+1)$

5-

(b) $g(x) = 42$

.... similar

(c) $-42 \leq g(x) < 42$

Min $g(x)$ after x IH

$\delta(g(x), a) = g(x) + 1$ constr.

$g(xa) = g(x)+1$

$\therefore P(x+1)$

$g(x) < 42$

$\therefore f(x) < 42$

$f(xa) = f(x) + 1 \leq 42$

Case 2, c = b: similar

QED

(end of induction example; Suggest you work through it yourself, to see that you can fill in the missing steps and write justifications for other steps.)

Regular Languages

$L \subseteq \Sigma^*$ is regular iff

$L = L(M)$ for some F.A. M

Examples

"even parity" is regular

"3rd from right" is regular

"odd length" is regular

"Σ^*" is regular

Closure Properties

Are there general ways to prove languages are regular, other than making more & more example M's?

Theorem

If L is regular then so is $\Sigma^* - L$

Theorem

If L is regular then so is $\Sigma^* - L$

Proof

L regular, so $L = L(M)$ for some fa $M = (Q, \Sigma, \delta, q_0, F)$

Let $M' = (Q, \Sigma, \delta, q_0, Q - F)$

For all $w \in \Sigma^*$:

M accepts $w \Leftrightarrow$

M is in a state $q \in F$ after reading w

$\Leftrightarrow M'$ .. ..

$\Leftrightarrow M'$ rejects w (since $q \in F \Leftrightarrow q \notin Q - F$)

$\therefore\ w \in L(M) \Leftrightarrow w \notin L(M')$

i.e. $L(M') = \Sigma^* - L$ is regular.

Closure Properties

A set is "closed" under some operation if applying the op to set members always yeilds a set member

Examples

$\mathbb{N}$ is closed under $+\ \times$ (eg $1+2 \in \mathbb{N}$)

but not under $-\ /$ (eg $1-2 \notin \mathbb{N}$)

$\mathbb{Z}$ is closed under $+\ -\ \times$ ($1-2 \in \mathbb{Z}$)

but not under $/$ ($1/2 \notin \mathbb{Z}$)

The set of regular languages is closed under complementation

Unary ops, too; e.g.: N is closed under squaring but not sqrt

Suppose

Program 1 recognizes L_1

& Program 2 recognizes L_2

Is there a program recognizing

$L_1 \cup L_2$?

$L_1 \cap L_2$?

$\vdots$

- Need to define carefully "language recognized by a Java program," etc., but the results suggested above are fairly intuitive
- Run prog 1 on input, then run prog 2 on same input; accept if either (∪)/both (∩) do.
- A really important difficulty: what if P1 doesn't halt?
- Fix for this problem: run both *in parallel*: 1st step of P2 then 1st step of P2 then next step of P1, then...
- Bottom Line: "yes, the set of languages recognized by Java programs *is* closed under union and intersection."

Example for FAs

- $\Sigma = \{0, 1, a, b\}$
- $L_1 = \{ w \in \{0,1\}^* \mid w \text{ has even parity } \}$
- $L_2 = \{ w \in \{a,b\}^* \mid w \text{ has exactly 5 a's } \}$
- $L_1 \cup L_2$ ← Easy-ish: 1st letter tells which case
- $L_3 = \{ w \in \{0,1\}^* \mid w \text{ has exactly 5 1's } \}$
- $L_1 \cup L_3$? ← Not so easy: both cases use just 0/1

Closure under Union

$M_i = (Q_i, \Sigma, \delta_i, q_{0i}, F_i)$

$M = (Q_1 \times Q_2, \Sigma, \delta, (q_{01}, q_{02}), F)$

$\forall\, q_1 \in Q_1,\ q_2 \in Q_2\quad a \in \Sigma$

$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$

Equivalent

$F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$

$F = \{ (a, b) \mid \text{either } a \in F_1 \text{ or } b \in F_2 \}$

Claim:

$\forall q_1 \in Q_1, \forall q_2 \in Q_2, \forall w \in \Sigma^*$
M is in state (q_1, q_2) after reading
w $\Leftrightarrow$ M_1 is in q_1 after reading w
and M_2 is in q_2 .. .. ..

Proof:

Homework (induction on $|w|$)

Corollary:

$$L(M) = L(M_1) \cup L(M_2)$$

Note:

Claim looks a lot like defn of δ.
BUT $\delta(-, a)$ for finite set $a \in \Sigma$
claim "... w" for infinite set $w \in \Sigma^*$