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

\[ z = 0, 1 \wedge \exists m \exists z = \not \exists \forall \exists 1 \wedge w \text{ is even} \]
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

\[ M = (Q, \Sigma, s, s_0, F) \]

where

- \( Q \) is a set (states)
- \( s \in Q \) start state
- \( \Sigma \) is a finite set (alphabet)
- \( F \subseteq Q \) Final states
- Accepting states
- \( s : Q \times \Sigma \to Q \) transition function
Formal version of parity, I

\[ M_{\text{parity}} = (Q, \Sigma, \delta, q_0, F) \]

where

\[ Q = \{ \text{even, odd} \} \]

\[ \Sigma = \{ 0, 1 \} \]

\[ q_0 = \text{even} \quad (\text{one element}) \]

\[ F = \{ \text{even} \} \quad (\text{a set containing one element}) \]

\[ \delta(q, a) \]

\[
\begin{array}{c|c|c|c}
q & 0 & 1 \\
\hline
\text{even} & \text{even} & \text{odd} \\
\text{odd} & \text{odd} & \text{even}
\end{array}
\]
Even more succinctly, if we let $Q = \{0, 1\}$ also
then $\delta(q, a) = (q + a) \mod 2$

for all $q$ in $Q$ and all $a$ in $\Sigma$
Example

\[ \Sigma = \{ a, b \} \]

\[ L = \{ w \mid \text{2nd letter of } w \text{ is “a”} \} \]
Example

\[ \Sigma = \{ a, b \} \]

\[ L = \{ w | \text{3rd letter of } w \text{ is } \text{“a”} \} \]
\[ \Sigma = \{a, b, \epsilon\} \]
\[ L = \{ \omega \mid 3^{rd} \text{ letter from the right end of } \omega \text{ is } a \} \]

<table>
<thead>
<tr>
<th></th>
<th>( \epsilon )</th>
<th>( a )</th>
<th>( b )</th>
<th>( \text{aa, ab, ba, bb} )</th>
<th>( \text{aaa} )</th>
<th>( \text{aab} )</th>
<th>( baa )</th>
<th>( \text{bbb} )</th>
<th>( \ldots )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>\ldots</td>
</tr>
</tbody>
</table>
L = { w in {a,b}* | 3rd letter from the right end of w is "a" }
L = \{ w \in \{a,b\}^* \mid 3rd letter from the right end of w is "a" \}

M = (\emptyset, \Sigma, S, \delta, \epsilon, F)

\Sigma = \{a, b\}

\emptyset = \{ w \in \Sigma^+ \mid |w| \leq 3 \}

\delta_0 = \varepsilon

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

\forall a, b, c \in \Sigma \delta(c, w, c) = \text{last 3 letters of } w$
DEF

"M ends in state \( q \) after reading \( w \in \Sigma^* \) if:

1. \( w = w_1, w_2, \ldots, w_n \)
   where \( w_i \in \Sigma \)

2. \( \exists \) state \( r_0, r_1, r_2, \ldots, r_n \in Q \)
   s.t.
   a) \( r_0 = q_0 \)
   b) \( \forall 1 \leq i \leq n \)
   \( s(r_{i-1}, w_i) = r_i \)

3. \( r_n = q \)

Fact: \( q \) is unique

because \( s \) is a function, basically.

Exercise: what state is \( M \) in after reading \( \varepsilon \)?
**Defn**

\[ M \text{ accepts } w \in \Sigma^* \iff \text{the state } q, \text{ reached by } M \text{ after reading } w \text{ is an accepting state, i.e., } q \in F. \]

And \( M \) rejects \( w \) iff \( q \notin F \)

**Defn**

The language recognized by \( M \),

\[ L(M) = \{ w \in \Sigma^+ \mid \text{M accepts } w \} \]

*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: 0.0

0.1
Example

\[ M : \emptyset \quad L(M) = \Sigma^* \]
Example

\[ M : \begin{array}{c}
0,1 \\
0,0
\end{array} \]

\[ L(\alpha) = \Sigma^* \]

\[ L_{\text{pal}} = \{ \omega \in \{0,1,3\}^* \mid \omega = \omega^R \} \]

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

\[ M \text{ above accepts every palindrome} \]

\[ \therefore \ L_{\text{pal}} \subseteq L(\alpha) \]

but \( M \) also accepts some (in fact, all) non palindromes

\[ \therefore \ L_{\text{pal}} \neq L(\alpha) \]
An example

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

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

$M = (\{0,1,2,3\}, \{0,1\}, \delta, 0, \{1,3\})$ where

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

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

[Isn’t this just the defn of \( \delta \)? No; \( w \in \Sigma^* \), not \( \Sigma \)]

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-\( \varepsilon \) string has a last letter, \( w = xa \) for some \( a \) in \( \Sigma \), and some string \( x \) of length \(< n \). Let \( i = (\#_1(x)) \mod 4 \). I.H. applies to \( x \), so we may assume \( M \) is in state \( i \) after reading \( x \). By def of \( \delta \) and “state reached after reading a string,” after reading \( w = xa \), \( M \) is in state \( \delta(i,a) \). Two cases, depending on \( a \) (and \( \delta \)):

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

Which establishes the claim.
• Corollary: the language recognized by $M$ is $\{w \in \{0,1\}^* \mid \#_1(w) \mod 4 = 1 \text{ or } 3 \}$. 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:

```c
int i = 0;

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

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

claim: i == 0
claim: i == 1read so far
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

\[ x \text{ is a prefix of } w \]

if \( \exists y \text{ s.t. } w = x y \)  \((w, x, y \text{ in } \Sigma^*)\)

Eq.

prefixes of \( abb \) are \( \varepsilon, a, ab, abb \)

Facts

\( \varepsilon \) is always a prefix

every \( w \) is a prefix of itself

if \( |w| = n \) then \( w \) has \( n+1 \) prefixes
Another Induction Example
\[ \Sigma = \{a, b, \lambda\} \]
\[ f(\lambda) = \#_a(\lambda) - \#_b(\lambda) \]
\[ \text{leg} = \exists w \mid f(w) = 0 \]

\[ L = \{ w \mid f(w) = 0 \land \forall x \leq |w|, \exists y \leq |x| \quad f(x) \leq 4 \} \]

\[ g(w) = \begin{cases} f(w) & \text{if } f(x) \leq 4 \text{ for all prefixes } x \leq w \\ 0 & \text{o.w.} \end{cases} \]

\[ Q = \{ -42, -41, \ldots, 41, 42, 43 \} \]
\[ q(0) = \begin{cases} 8 + 1 & \text{if } c = a, 8 < q_2 \\ \frac{c}{b-1} & \text{if } c = b, 8 \geq q_2, \text{o.w.} \end{cases} \]
Claim: \( \forall w \in \Sigma^* \) the state reached by \( M \) after reading \( w \) is \( q = g(w) \)

Correct: \( M \) accepts \( L \) (but not \( L_{eq} \))

Proof:
- \( M \) accepts \( w \) if and only if \( M \) ends in \( F \) by definition and construction.
- \( M \) ends in \( F \) if and only if \( 0 = g(w) \) by claim.
- \( 0 = g(w) \) if and only if \( w \in L \) by definition.
Claim \( \forall w \in \Sigma^* \), state reached by \( M \) after reading \( w \) is \( g(w) \)

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

To prove \( \forall n \geq 0 \ P(n) \)

**Basis** \( n = 0 \), \( w = \varepsilon \)

- \( M \) reaches state \( 0 \) on \( \varepsilon \)
  - By construction, \( g(\varepsilon) = 0 \) by construction
  - Say more

**Induction** \( P(n) \Rightarrow P(n+1) \)

Let \( w \) be of length \( n+1 \)

- \( w = x \varepsilon \) for some \( x \in \Sigma^n \)

**Case 1**: \( c = a \)

- \( g(x) = 99 \)
  - \( M \) is in \( 99 \) after reading \( x \) \( \) by \( IH \)
  - By work

  - \( g(x, a) = 99 \)
  - \( g(x, a) = 99 \)
  - \( P(n+1) \)

  - Argue based on \( g(x, a) = 99 \)
(b) \( g(x) = 42 \)

\[ \text{similar} \]

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

\[ \text{Min} g(x) \text{ after } x \]

\[ s(g(x), a) = g(x) + 1 \]

\[ g(x+1) = g(x)+1 \]

\[ \therefore P(x+1) \]

\[ g(x) \leq 42 \]

\[ f(x) \leq 42 \]

\[ f(x+1) = f(x)+1 \leq 42 \]
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
"\Sigma^*" is regular
Closure Properties
Are there general ways to prove languages are regular, other than making more and 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 $w$ $\iff$

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

$\iff M' \ldots \ldots \ldots$

$\iff M'$ rejects $w$ (since $q \in F \iff q \not\in (Q - F)$)

$\therefore w \in L(M) \iff w \not\in L(M')$

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 yields a set member.

Examples

\[ \mathbb{N} \text{ is closed under} +, \times \quad (e.g. 1 + 2 \in \mathbb{N}) \]

but not under \(-, \) (e.g. 1 - 2 \notin \mathbb{N})

\[ \mathbb{Z} \text{ is closed under} +, -, \times \quad (1 - 2 \in \mathbb{Z}) \]

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

The set of regular language is closed under complementation.

Unary ops, too; e.g.:

\[ \mathbb{N} \text{ is closed under squaring} \]

but not sqrt.
Suppose

1. Program 1 recognizes $L_1$
2. Program 2 recognizes $L_2$

Is there a program recognizing $L_1 \cup L_2$?

$L_1 \cap L_2$?
• 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 \( \cup \)/both \( \cap \) 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

- $\Sigma = \{0, 1, a, b\}$

- $L_1 = \{ w \in \{0,1\}^* | w \text{ has even parity} \}$

- $L_2 = \{ w \in \{a,b\}^* | w \text{ has exactly 5 a's} \}$

- $L_1 \cup L_2$  
  Easy-ish: 1st letter tells which case

- $L_3 = \{ w \in \{0,1\}^* | w \text{ has exactly 5 1's} \}$

- $L_1 \cup L_3$  
  Not so easy: both cases use just 0/1
Closure under Union

\[ M_i = (Q_i, \Sigma, S_i, q_0, \delta_i, F_i) \]

\[ M = (Q_1 \times Q_2, \Sigma, \delta, (q_0, q_0), F) \]

\[ \forall q_1 \in Q_1, \forall q_2 \in Q_2: a \in \Sigma \]

\[ \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)) \]

\[ F = (F_1 \times Q_2) \cup (Q_1 \times F_2) \]

\[ F = \{ (a, b) \mid \text{either } a \in F_1 \text{ or } b \in F_2 \} \]
Claim:
\[ \forall q_1 \in Q_1, \forall q_2 \in Q_2, \forall w \in \Sigma^* \]
\[ M \text{ is in state } (q_1, q_2) \text{ after reading } w \iff M_1 \text{ is in } q_1 \text{ after reading } w \]
\[ \text{and } M_2 \text{ is in } q_2 \]

Proof:
Homework (induction on \(|w|\))

Corollary:
\[ L(M) = L(M_1) \cup L(M_2) \]

Note:
Claim looks a lot like def of $\delta$

But $\delta(-,a)$ for finite set $Q$ or

claim "$\ldots\,$" for infinite set $w \in \Sigma^*$