An Example: Even Parity

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

  \[ S = \{ 0, 1 \} \]
  \[ L = \{ \exists w \in \Sigma^* | W \text{ with even } 3 \} \]

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

A finite state machine

\[ M = (Q, \Sigma, s, f_0, F) \]

where

- \( Q \) is a set (states)
- \( s_0 \in Q \) is the start state
- \( \Sigma \) is a finite set (alphabet)
- \( F \subseteq Q \) is the set of accepting states
- \( \delta : Q \times \Sigma \rightarrow Q \) is the transition function

\[ \delta(q, a) = (q + a) \mod 2 \]

for all \( q \) in \( Q \) and all \( a \) in \( \Sigma \)

**Formal version of parity, I**

\[ M_{\text{parity}} = (Q, \Sigma, s_0, F) \]

where

- \( Q = \{ \text{even}, \text{odd} \} \)
- \( \Sigma = \{ 0, 1 \} \)
- \( s_0 = \text{even} \) (one element)
- \( F = \{ \text{even} \} \) (a set containing one element)

\[ \delta(q, a) = \begin{cases} a & \text{if } q \text{ is even} \\ \text{even} & \text{if } q \text{ is odd} \\ \text{odd} & \text{if } q \text{ is even} \\ \text{even} & \text{if } q \text{ is odd} \end{cases} \]

**Example**

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

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

**Formal version of parity, II**

Even more succinctly if we let \( Q = \{ 0, 1 \} \) also then \( \delta(q, a) = (q + a) \mod 2 \)
Example

\( \Sigma = \{ a, b \} \)

\( L = \{ w \mid \text{3rd letter of } w \text{ is } "a" \} \)

\( \Sigma = \{ \epsilon, a, b \} \)

\( L = \{ w \mid \text{3rd letter from the right end of } w \text{ is } "a" \} \)

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

\( \Sigma = \{ a, b \} \)

\( q_0 = \{ \epsilon \} \)

\( F = \{ a^n b^m \mid n, m \geq 3 \} \)

\( \delta( q_0, \epsilon) = q_0 \)

\( \delta( q_0, a) = \{ a, b \} \)

\( \delta( q_0, b) = \{ b, a \} \)

\( \delta( \{ a, b \}, a) = \{ a, b \} \)

\( \delta( \{ a, b \}, b) = \{ b, a \} \)

\( v \in S \times \{ \epsilon, a, b \} \Rightarrow \text{Last 3 letters of } v \)
Exercise: what state is $M$ in after reading $\varepsilon$?

Fact: $q_0$ is unique because $\varepsilon$ is a string.

Example

$M: \begin{array}{c}
0 \rightarrow 0,
\end{array}$

$L(M) = \varepsilon^*$
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
δ₁(0) = i
δ₁(1) = (i+1) mod 4
What does M do?

• Corollary: the language recognized by M is {w in {0,1}* | #₁(w) mod 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).
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
Another Induction Example

Claim: \( \forall w \in \Sigma^* \text{ state reached by } M \text{ after reading } w \) is \( q = g(w) \)

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

Proof: \( M \) accepts \( w \) \( \Rightarrow \) \( M \text{ moves on } P \) by definition

\( \Rightarrow \) \( M \text{ moves on } 0 \) by definition

\( \Rightarrow \) \( 0 = g(w) \) by claim

\( \Rightarrow \) \( w \in L \) by definition.
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.)

Closure Properties

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

Theorem

If \( L \) is regular then so is \( \Sigma^* - L \)

Proof

\( L \) regular, so \( L = L(M) \) for some \( F \)

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

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

- \( M \) accepts \( w \) if \( w \) is in a state \( q \in F \) after reading \( w \)
- \( M' \) also accepts \( w \)
- \( M' \) rejects \( w \) (since \( (q, \sigma, q') \notin \delta \))

\( w \in L(M) \) \( \iff \) \( w \notin L(M') \)

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 yields a set member.

Examples

- \( N \) is closed under + \( x \) (\( y \rightarrow x \rightarrow y \))
  but not under - \( y \) (\( 1 \rightarrow 1 \rightarrow 1 \))

- \( \mathbb{Z} \) is closed under + \( x \) (\( 1 \rightarrow 2 \rightarrow 3 \))
  but not under - \( y \) (\( 1 \rightarrow \frac{1}{2} \))

The set of regular languages is closed under complementation.
Suppose
Program 1 recognizes $L_1$
& Program 2 recognizes $L_2$
Is there a program recognizing $L_1 \cup L_2$?
$L_1 \cap L_2$?

• 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

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

Closure under Union

$M_1 = (Q_1, I, S_1, \delta, \gamma_1, F_1)$
$M_2 = (Q_1 \times Q_2, I, S_1 \times S_2, \delta \times \delta, (\gamma_1 \times \gamma_2), F_1 \times F_2)$
$Q = Q_1 \times Q_2, \delta_{Q1 \times Q2}(s_1, s_2, a) = \delta_1(q_1, s_1) \times \delta_2(q_2, s_2)$
$F = \{ (q_1, q_2) | \text{either } q_1 \in F_1 \text{ or } q_2 \in F_2 \}$
Claim:
\[ \forall q_1, q_2, \forall w \in \Sigma^* \]
Missenses \((q_1, w)\) after reading \(w \in M_1 \) is in \( q_1 \) after reading \( w \) and \( M_2 \) starts \( q_2 \) \[ \vdots \]

Prof:
Homework (Induction on \( \Sigma^* \))

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

Note:
Claim looks a lot like definition,
BUT \( b(\ldots, \epsilon) \) for finite set \( a \in \Sigma^* \)
chain \( \ldots \epsilon \) for infinite set \( \omega \)