Lecture 2  9/11/18

Recap

1. What is computation?
2. What is computable?
3. What is computable given our resources

For the first few weeks, we will focus on question one: what is computation?

What we want from a model of computation:

- Simple & easy to reason about mathematically
- Powerful: Must represent a reasonably complex computation

What is a computer?

- Takes in an input → finite strings over finite alphabet
- Does some work (see below)
- (Hopefully) Produces an output → Accept/Reject
  (can express complex output w/ binary)

Work:

Starts at a "blank slate" state, q₀

\[
q₀ \xrightarrow{\text{see/process more inputs}} q₁ \xrightarrow{\text{transition to a "more informed" state}} \cdots \xrightarrow{\text{Accept/Reject}} q₅ \xrightarrow{\text{Accept/Reject}}
\]

Ex:

Want a state diagram for a machine that determines if its input (a binary string) has an even # of Os

Note: \(\square\) indicates an accept state
Def: Let \( \Sigma \) be a finite set (aka alphabet). A language over \( \Sigma \) is a set of finite-length strings over \( \Sigma \).

Ex: Binary alphabet: \( \{0, 1\} \),
    \( \{a, b\} \), \( \{0, 1\} \), ..., \( \{9\} \)

Ex languages:
- \( L_1 = \{\text{anna}3\} \)
- \( L_2 = \{\text{w} \mid \text{w is a string over } \{0, 1\} \text{ with an even \# of } 0s\} \)
- \( L_3 = \{\text{w} \mid \text{w is a string over } \{0, 1\} \text{ with an odd \# of } 0s\} \)
- \( L_4 = \{\text{w} \mid \text{w is a string over } \{0, 1\} \text{ with at least one } 1\} \)
- \( L_5 = \{\text{w} \mid \text{w is a finite binary string s.t. it is a} \ a2 \text{ binary representation of a prime \#}\) \)
- \( L_6 = \{\text{3} = \emptyset \) (the empty set)

How powerful are these "state machines"? (aka "DFAs")

Ex: \( L_3 \) (odd \# of 0s)

Just switch accept/reject states.
A deterministic finite automaton (DFA) is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\) where:

- \(Q\): finite set of states
- \(\Sigma\): finite alphabet
- \(\delta: Q \times \Sigma \rightarrow Q\) is the transition function
- \(q_0 \in Q\) is the start state
- \(F \subseteq Q\) is the set of accepting states

**Example:**

```
\[ Q = \{q_0, q_1, q_2\}, \quad \Sigma = \{\epsilon, a, b\}, \quad F = \{q_0, q_1\} \]
\[ \delta = \begin{array}{c|cc}
q_0 & 0 & q_1 \\
q_0 & 1 & q_0 \\
q_1 & 0 & q_0 \\
q_1 & 1 & q_1 \\
\end{array} \]
```

**Formal definition**

<table>
<thead>
<tr>
<th>State</th>
<th>Symbol</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>(q_0)</td>
<td>0</td>
<td>(q_1)</td>
</tr>
<tr>
<td>(q_0)</td>
<td>1</td>
<td>(q_0)</td>
</tr>
<tr>
<td>(q_1)</td>
<td>0</td>
<td>(q_0)</td>
</tr>
<tr>
<td>(q_1)</td>
<td>1</td>
<td>(q_1)</td>
</tr>
</tbody>
</table>

**Definition**

Let \(M = (Q, \Sigma, \delta, q_0, F)\) be a DFA. Let \(w = w_1 \ldots w_n\) be a string over \(\Sigma\) of length \(n\). Then the computational history of \(M\) on input \(w\) is a sequence \(r_0, r_1, \ldots, r_n\) such that:

- \(r_0 = q_0\)
- \(r_i = \delta(r_{i-1}, w_i)\) for \(1 \leq i \leq n\)

**Definition**

A DFA \(M\) accepts \(w\) if its computational history on \(w\) ends in an accepting state, i.e. \(r_n \in F\)

**Example**

\(w = 010110\)

\(q_0, q_1, q_0, q_0, q_0, q_1\) \[\text{rejects!}\] (using even # Os machine as \(M\))
Def: A DFA $M$ recognizes a language $L$ if for any $w$, $w \in L$ if and only if $M$ accepts $w$.

Def: The language of a DFA $M$, $L(M) = \{w | M$ accepts $w\}$.

How would we prove that the language of our "even # 0s" DFA is correct?

$L_3$ could formally do an induction over # of symbols in input, but ok to hardware / take shortcuts in this class.

Note: The empty string is a string with 0 symbols.

Denoted $\epsilon$.

Ex: $L_1 = \epsilon, ana_3$

Ex: $L_4 = \epsilon$ binary string w/ at least one 1.

Ex: $L_6 = \emptyset$ (empty language).

Def: A language $L$ is regular if it is recognized by some DFA.

Thm: If $L$ is regular, $L^c$ is regular ($L^c = \{w | w \notin L\}$).

Proof:
Let $M^* = (Q, \Sigma, \delta, q_0, \mathcal{F})$ be a DFA recognizing $L$. Construct $M'_1 = (Q, \Sigma, \delta, q_0, Q \setminus \mathcal{F})$. $M'_1$ recognizes $L^c$ because $M'_1$ accepts $w$ if and only if the run of $M^*$ on $w$ ends with $q \in Q \setminus \mathcal{F}$.

**Definition**

Let $A, B$ be languages.

- $A \cup B = \{w \mid w \in A \text{ or } w \in B\}$ (union)
- $A \circ B = \{w_1w_2 \mid w_1 \in A, w_2 \in B\}$ (concatenation)
- $A^* = \{w_1w_2 \ldots w_k \mid k \geq 0, w_i \in A\}$ (Kleene star)

**Theorem**

If $A, B$ are regular languages, then so are

1. $A \cup B$
2. $A \circ B$
3. $A^*$

**Example:** Recall DFA for $L_2$ (even # of 0's) and DFA for $L_4$ (at least one 1)

$L_2$: 

```

```

$L_4$: 

```

How to make machine for $L_2 \cup L_4 = \{w \mid w \text{ has even # zeros and at least one 1}\}$?

When to accept?

$L \iff$ whenever either $L_2$ or $L_4$ contain string (i.e., Cartesian product of accepting states for machines for $L_2$ and $L_4$)

---

General proof (for union)
Let $M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ recognize $A$.
Let $M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$ recognize $B$.
Let $M_C = (Q_C, \Sigma, \delta_C, q_{0C}, F_C)$ be:

- $Q_C = Q_A \times Q_B$
- $\Sigma$ the same
- $\delta_C((q_A, q_B), a) = (\delta_A(q_A, a), \delta_B(q_B, a))$
- $q_{0C} = (q_{0A}, q_{0B})$
- $F_C = \varepsilon(q_A, q_B) | q_A \in F_A \lor q_B \in F_B \lor$