11.1 Finite Automata

Motivation:

- TMs without a tape: maybe we can at least fully understand such a simple model?
- Algorithms (e.g. string matching)
- Computing with very limited memory
- Formal verification of distributed protocols,
- Hardware and circuit design

Example: Home Stereo

- \( P = \) power button (ON/OFF)
- \( S = \) source button (CD/Radio/TV), only works when stereo is ON, but source remembered when stereo is OFF.
- Starts OFF, in CD mode.
- A computational problem: does a given a sequence of button presses \( w \in \{P,S\}^* \) leave the system with the radio on?

The Home Stereo DFA
**Formal Definition of a DFA**

- A DFA $M$ is a 5-Tuple $(Q, \Sigma, \delta, q_0, F)$
  
  $Q$: Finite set of states
  
  $\Sigma$: Alphabet
  
  $\delta$: “Transition function”, $Q \times \Sigma \rightarrow Q$
  
  $q_0$: Start state, $q_0 \in Q$
  
  $F$: Accept (or final) states, $F \subseteq Q$

- If $\delta(p, \sigma) = q$,
  
  then if $M$ is in state $p$ and reads symbol $\sigma \in \Sigma$

  then $M$ enters state $q$ (while moving to next input symbol)

**Another Visualization**

$M$ accepts string $x$ if

- After starting $M$ in the start[initial] state with head on first square,

- when all of $x$ has been read,

- $M$ winds up in a final state.
Example

Bounded Counting: A DFA that recognizes \( \{ x : x \text{ has an even # of } a\text{'s and an odd # of } b\text{'s} \} \)

Transition function \( \delta \):  

\[
\begin{array}{c|cc}
 & a & b \\
\hline
q_0 & q_1 & q_2 \\
q_1 & q_0 & q_3 \\
q_2 & q_3 & q_0 \\
q_3 & q_2 & q_1 \\
\end{array}
\]

i.e. \( \delta(q_0, a) = q_1 \), etc.

\( \bigcirc \) = start state  \( \bigcirc \) = final state

\[ Q = \{ q_0, q_1, q_2, q_3 \} \quad \Sigma = \{ a, b \} \quad F = \{ q_2 \} \]

Formal Definition of Computation

\[ M = (Q, \Sigma, \delta, q_0, F) \text{ accepts } w = w_1w_2 \cdots w_n \in \Sigma^* \text{ (where each } w_i \in \Sigma) \text{ if there exist } r_0, \ldots, r_n \in Q \text{ such that} \]

1. \( r_0 = q_0 \).
2. \( \delta(r_i, w_{i+1}) = r_{i+1} \) for each \( i = 0, \ldots, n - 1 \), and
3. \( r_n \in F \).

The language recognized (or accepted) by \( M \), denoted \( L(M) \), is the set of all strings accepted by \( M \).
Another Example, To Do On Your Own

- **Pattern Recognition**: A DFA that accepts \( \{ x : x \text{ has } aab \text{ as a substring} \} \).

Another Example, To Do On Your Own

- **Pattern Recognition**: A DFA that accepts \( \{ x : x \text{ has } ababa \text{ as a substring} \} \).

**Using DFAs for Pattern Recognition**

**Problem**: given a pattern \( w \in \Sigma^* \) of length \( m \) and a string \( x \in \Sigma^* \) of length \( n \), decide whether \( w \) is a substring of \( x \).

**Algorithm**:

1. Construct a DFA \( M \) that accepts \( L_w = \{ x \in \Sigma^* : w \text{ is a substring of } x \} \).
   - States are \( Q = \{0, 1, \ldots, m\} \). State \( q \) represents:
   - Transitions: \( \delta(q, \sigma) = \)
   - Time to construct \( M \) (naively): \( O(m^3 \cdot |\Sigma|) \).

2. Run \( M \) on \( x \).
   - Time: \( O(n) \)

The running time can be improved to \( O(m + n) \), using an appropriate implicit representation of the DFA. Widely used in practice! (Look up the Knuth-Morris-Pratt algorithm.)
Characterizing the Power of Finite Automata

**Def:** A language $L \subseteq \Sigma^*$ is *regular* iff there is a DFA $M$ such that $L(M) = L$. REG denotes the class of regular languages.

The terminology “regular” comes from an equivalent characterization in terms of *regular expressions* (which we won’t cover in lecture, but possibly will on a problem set). Note that $\text{REG} \subseteq \text{TIME}_{\text{TM}}(n)$; it also can be shown that $\text{REG} \subseteq \text{CF}$. Unlike classes associated with universal models (like TMs and Word-RAMs), we have a fairly complete understanding of the class of regular languages. In particular,

**Myhill-Nerode Theorem:** A language $L \subseteq \Sigma^*$ is regular iff there are only finitely many equivalence classes under the following equivalence relation $\sim_L$ on $\Sigma^*$: $x \sim_L y$ iff for all strings $z \in \Sigma^*$, we have $xz \in L \iff yz \in L$. Moreover, the minimum number of states in a DFA for $L$ is exactly the number of equivalence classes under $\sim_L$.

(Exercises: refresh your memory on the definition of equivalence relations and equivalence classes.)

**Proof:** $\Rightarrow$. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA such that $L(M) = L$. Note that if $x, y \in \Sigma^*$ drive $M$ to the same state (starting from $q_0$), then for all $z \in \Sigma^*$, $xz$ and $yz$ drive $M$ to the same state and hence both are in $L(M) = L$ or neither are in $L(M)$. Thus $x \sim_L y$. Hence the number of equivalence classes under $\sim_L$ is at most $|Q|$. 

$\Leftarrow$. Suppose $\sim_L$ has finitely many equivalence classes, where we write $[x]_L$ for the equivalence class containing $x$. We construct a DFA $M = (Q, \Sigma, \delta, q_0, F)$ as follows:

- $Q$ is the set of equivalence classes under $\sim_L$.
- $q_0 = [\epsilon]_L$.
- $F = \{[x]_L : x \in L\}$.
- $\delta([x]_L, \sigma) = [x\sigma]_L$. (Note that this is well-defined: if $x \sim_L y$, then $x\sigma \sim_L y\sigma$, so the choice of the representative $x$ of the equivalence class does not affect the result.)

By induction on $|x|$, it can be shown that running $M$ on $x$ leads to state $[x]_L$, and hence we accept exactly the strings in $L$. $\blacksquare$
**Proving that languages are nonregular.** To show that $L$ is nonregular, we only need to exhibit an infinite set of strings that are all inequivalent under $\sim_L$. Some examples follow:

- $L = \{a^n b^n : n \geq 0\}$. Exercise: prove that $\epsilon, a, a^2, a^3, a^4, \ldots$ are all pairwise inequivalent under $\sim_L$.

- $L = \{w \in \Sigma^* : |w| = 2^n \text{ for some } n \geq 0\}$. Claim: $\epsilon, a, a^2, a^3, a^4, \ldots$ are all inequivalent under $\sim_L$. Suppose $a^i \sim_L a^j$ for some $i > j$. Let $k$ be any power of 2 larger than $i$ and $j$. Then $a^i \cdot a^{k-j} \in L$, so $a^i \cdot a^{k-j} \in L$ and hence $k + i - j$ is a power of 2. But $2k$ is the next larger power of 2 after $k$. $\Rightarrow \Leftarrow$.

- $L = \{w \in \Sigma^* : w = w^R\}$ (palindromes). Exercise: prove that $a, a^2 b, a^3 b, \ldots$ are pairwise inequivalent.