1 Two-Way Finite Automata

A two-way deterministic finite automaton (2DFA) is a generalized DFA which can read the input string in two directions. It has a reading hat which can move one character left or right over the input string. The string has two delimiters $e$ and $\$, which are not elements of the input alphabet $\Sigma$, and respectively showing the beginning and the end of the string.

Transition functions: $Q \times \Sigma \rightarrow Q \times \{L,R\}$
System reads a character in a state, then decides to go left or right and switches its state.

State set: There are halting states (accept or reject) in which the reading hat reaches to the right-end and automaton stops, and non-halting states where it does not.

Can it recognize non-regular languages? No, 2DFAs and DFAs are equivalent in the sense of only recognizing regular languages. But they might have some practicalities over DFAs.

Example: Consider the language $L_m = \{a^i \mid i \text{ is a multiple of } m\}$ and take $m = 2 \cdot 3 \cdot 5 \cdot \ldots \cdot 19$.
You can build a 2DFA which traverses the string and for one prime factor of $m$ such as 5, it counts the string length in modulo 5 and checks its divisibility by using only 5 states. By doing this for every factor of $m$, $L_m$ can be recognized with only $2 + 3 \ldots + 19$ states, whereas a DFA needs at least $m$ number of states.

A proof to their equivalence: Let $M$ be a 2DFA with set of states $S$ recognizing $L$.
For every string $w \in \Sigma^*$, define a function $\tau_w: \{s_0\} \cup S \rightarrow \{0\} \cup S$.
This function will serve the purpose of showing how the machine behaves when it crosses the boundary between $w$ and $z$ in an input like $e \ l w \ r z \$.

For any state $s \in S$, $\tau_w(s)$ shows the ultimate result of the motion of $M$ when it starts on the right-most symbol of $w$ in state $s$, i.e. if $M$ ultimately leaves $w$ after some steps, from the right-most symbol into the state $s'$, then $\tau_w(s) = s'$.
If $M$ never leaves $w$ or leaves it from the left, then $\tau_w(s) = 0$.

$\tau_w(s_0)$ has the special meaning of which state the machine lands in when it leaves $w$ for the first time from the right-most symbol i.e. when $M$ is started on the initial state on the left most symbol of $w$, eventually it leaves $w$ from the right-most symbol into the state $s'$ where $\tau_w(s_0) = s'$.
or it never leaves $w$ where $\tau_w(s_0) = 0$.

Now consider two input strings $w_1z$ and $w_2z$. If $M$ accepts a string, the marker ends up at right-end symbol $\$, so it leaves $w$ from its right-most symbol at least once and lands in the left-most symbol of $z$. Let $\tau_{w_1}$ and $\tau_{w_2}$ denote the "table"s of $\tau$ meaning all values of $\tau$ for any element from $\{s_0\} \cup S$. If $\tau_{w_1} = \tau_{w_2}$ for these two strings $w_1$ and $w_2$, then they land in the left-most symbol of $z$ and into the same state, as their tables are the same. So, $M$ either accepts or rejects both, which means for all $z \in L$, $w_1z \equiv_L w_2z \iff w_1 \equiv_L w_2$. 

---

CMPE 598 - Lecture Notes
Asım Gümüş
March 27, 2018
Note also that, if there are \( k \) states in \( S \), there can be only \((k+1)^k\) distinct tables, so there are only a finite number of different \( \tau \) functions!

Using Myhill-Nerode theorem \( \implies \) Only regular languages are recognized by 2DFAs.

# 2 Two-way Probabilistic Finite Automata

Let us build a 2PFA which recognizes \( \{a^nb^n \mid n \geq 0\} \) with error bound \( \varepsilon \) for any desired \( \varepsilon > 0 \).

Let \( x = \frac{\varepsilon^2}{2} \). The machine splits into three paths when it starts.
All paths check whether an "a" appears after a "b" while doing their other jobs, and reject if this happens.

**Path 1** moves on with probability \( x \) and restarts with probability \( 1 - x \) (i.e. goes back to \( \varepsilon \) and switches to the initial state) when reading symbols \( a \) or \( b \).

After reading the right-end marker, it accepts with probability 1.

**Path 2** moves on with probability \( x^2 \), restarts with probability \( 1 - x^2 \) when reading symbol \( a \).

On \( b \)'s, it goes on with probability 1.

At the right-end marker, it rejects with probability \( \frac{\varepsilon}{2} \) and restarts with \( 1 - \frac{\varepsilon}{2} \).

**Path 3** is just like **Path 2**, but transitions between \( a \) and \( b \) are interchanged.

If the input is of the form \( a^mb^n \), then the accept and reject probabilites in the first round -before any restart- are as follows:

\[
P_{\text{acc}} = \frac{1}{3}x^m x^n \quad P_{\text{rej}} = \frac{1}{3} \cdot \frac{\varepsilon}{2} x^{2m} + \frac{1}{3} \cdot \frac{\varepsilon}{2} x^{2n} = \frac{\varepsilon}{6} (x^{2m} + x^{2n})
\]

If \( m = n \),

\[
\frac{P_{\text{rej}}}{P_{\text{acc}}} = \frac{\frac{\varepsilon}{6} (x^{2m} + x^{2m})}{\frac{1}{3}x^{2m}} = \varepsilon
\]

If \( m \neq n \), without loss of generality, \( m = n + d \) for \( d > 0 \). Then,

\[
\frac{P_{\text{acc}}}{P_{\text{rej}}} = \frac{2x^{2n+d}}{\varepsilon \cdot (x^{2n+2d} + x^{2n})} = \frac{2}{\varepsilon} \cdot \frac{x^d}{x^{2d} + 1} < \frac{2}{\varepsilon} \cdot \frac{x^d}{x^{2d}} \leq \frac{2}{\varepsilon} x^d
\]

Substitute \( x = \frac{\varepsilon^2}{2} \):

\[
\implies \frac{P_{\text{acc}}}{P_{\text{rej}}} < \varepsilon
\]

So this 2PFA can recognize this non-regular language with an error bound \( \varepsilon \), but there is a catch: its runtime is bad \( (\text{Frey}) \). For any polynomial \( p \), 2PFA with expected runtime \( \theta(p(n)) \) recognize only the regular languages with bounded error!