CMSC452 Elementary Theory of Computation

Michael Li

Contents

1 Deterministic Finite Automata (DFA)

1.1 Alphabet and Strings

Σ is defined as the alphabet (e.g. $\{0,1,2\}$) and a **string** is a sequence of symbols of Σ

$\Sigma^i = \{\sigma_1 \cdots \sigma_i \mid \sigma_1, \cdots, \sigma_i \in \Sigma\}$

$\Sigma^0 = \{e\}$ the **empty string**. Useful for $w \cdot e = w$ (concatenation on string w)

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \cdots$ set of all strings including e

1.2 Concatenation, Number of a's, Prefix

Let $x, y \in \Sigma^*$. **Concatenation** of x and y is denoted xy or $x \cdot y$.

If $A, B \subseteq \Sigma^*$ then $A \cdot B = \{x \cdot y \mid x \in A \wedge y \in B\}$

if $x \in \{a,b\}^*$ then $\#_a(x)$ is the number of a's in x

if $x \in \{a,b\}^*$ then $x \preceq y$ means that x is a prefix of y

1.3 Modular Arithmetic

$x \equiv y \pmod{n}$ iff n divides $x - y$

- Addition: wrap around n
- Multiplication: wrap around n
- Division: find a number y such that $\frac{1}{x} * y \equiv 1 \pmod{n}$

  Example: $\frac{1}{3} \equiv 9 \pmod{26}$ since $9 * 3 \equiv 1 \pmod{26}$

  Note: a number x only has an inverse $\pmod{n}$ if x and n have no common factors

1.4 DFA Formal Definition

DFA is a tuple $(Q, \Sigma, \delta, s, f)$ where

- Q is a finite set of **state**
- Σ is a finite alphabet
- $\delta \colon Q \times \Sigma \to Q$ is the **transition function**
- $s \in Q$ is the **start state**
- $F \subseteq Q$ is the set of **final states**

If M is a DFA and $x \in \Sigma^*$, $M(x)$ **accepts** x if when running x through m ends up in a final state.

The Language $L(M) = \{x \mid M(x) \text{ accepts}\}$

Let $L \subseteq \Sigma^*$. If there exists a DFA M such that $L(M) = L$ then L is **regular**

DFA Limitations:

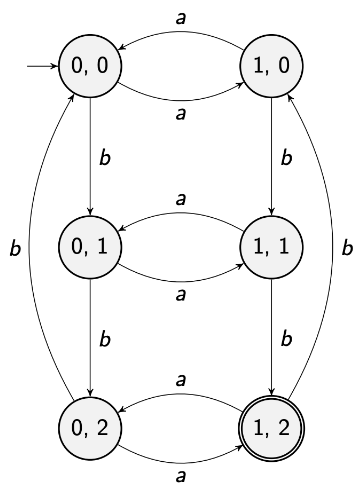
- DFA can only read 1 letter at a time and can never look at it again (one scan)

- DFA only has a finite number of states, $O(1)$ memory
- DFA CAN keep track of $\#_a(w) \pmod{17}$
- DFA CANNOT keep track of $\#_a(w)$

1.5 DFA Representations

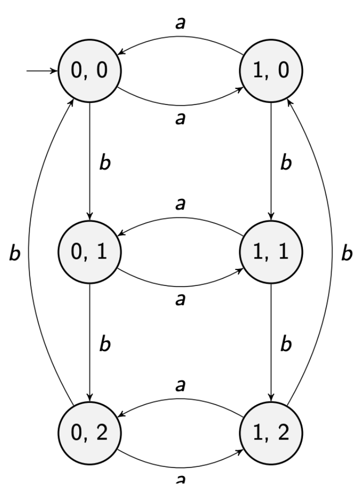
Normal DFA that **accepts** and **rejecs** strings

$$\{w : \#_a(w) \equiv 1 \pmod 2 \wedge \#_b(w) \equiv 2 \pmod 3\}$$



DFA Classifer that DOES NOT **accept** or **reject** strings. It **classifies** possible states.

$$\{w : \#_a(w) \pmod 2 \wedge \#_b(w) \pmod 3\}$$



Proof the above DFA has at least 6 states.

Proof by contradiction: assume DFA M has ≤ 5 states.

- on input e, the empty string, goes to state q_e
- on input a, goes to state q_a
- on input b, goes to state q_b

- on input bb, goes to state q_{bb}
- on input ab, goes to state q_{ab}
- on input abb, goes to state q_{abb}
- Since ≤ 5 states, 2 of these go to the same state (e.g. q_{aa} and a_{bb}). However,
  - $aa \cdot abb$ goes to some state q_i which must accept since $aaabb \in L$
  - $bb \cdot abb$ goes to some state q_i which also must be accepted but $bbabb \notin L$. Thus, contradiction is reached.
  - Would need to repeat the above contradiction for all pairs of q

Transition Table Representation

$$\{w : \#_a(w) \equiv 0 \pmod{n} \wedge \#_b \equiv 0 \pmod{m}\}$$

$$\begin{aligned}
&Q = \{0, \ldots, n-1\} \times \{0, \ldots, m-1\} \\
&s = (0,0) \\
&F = \{(0,0)\} \\
&\delta((i,j), a) = (i+1 \pmod{n}, j). \\
&\delta((i,j), b) = (i, j+1 \pmod{m}).
\end{aligned}$$

Transition table will have nm states

Further Notes:

- For an alphabet $\Sigma = \{a, b\}$, $\Sigma^* a \Sigma^i$ can be done with 2^{i+1} states.
- Any finite set can be recognized by a DFA. Just draw a different state for each string in the set. This will take up around 2^n states, but typically can do this in fewer states.