CS 172: Computability and Complexity

DFAs and NFAs

Sanjit A. Seshia
EECS, UC Berkeley

Acknowledgments: L. von Ahn, L. Blum, M. Blum
What we’ll do today

• Introduction to
  – DFA : Deterministic Finite Automata
  – NFA : Non-deterministic Finite Automata
• DFAs to/from regular languages
• Operations on regular languages
• Non-determinism and NFAs
• HW1 is out today
DFA Terminology

- **Start state** $q_0$
- **States** $Q$
- **Transitions**
- **Accept/final states** $F$

Diagrams showing states and transitions.
A finite automaton is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$

- $Q$ is the set of states
- $\Sigma$ is the 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 accept states

$L(M) =$ the language of machine $M$

$= $ set of all strings machine $M$ accepts
Acceptance (Formal Def.)

• M is a DFA with alphabet \( \Sigma \)
• An input string (word) \( w \) is a sequence \( w_1 \ w_2 \ w_3 \ldots \ w_n \) where each \( w_i \in \Sigma \)
  – \( \epsilon \) denotes the empty string
• M accepts \( w \) if “after fully reading \( w \) it ends in an accept state”
  – If there is a sequence of states \( s_0, s_1, \ldots s_n \) such that
    • \( s_0 = q_0 \)
    • \( s_i = \delta(s_{i-1}, w_i) \)
    • \( s_n \in F \)
Regular Language

• $M$ is a DFA with alphabet $\Sigma$
• $L(M)$ is the set of strings accepted by $M$
  – $L(M)$ is the language of $M$
  – $M$ recognizes $L(M)$

• Regular language: set of strings that is the language of some DFA

• The set of all possible strings is denoted $\Sigma^*$
• Is $\Sigma^*$ regular?
L(M) = ?
\[ L(M) = ? \]
Design a DFA

$L(M) = \{ w \mid w \text{ has an even number of } 1s \}$
$L(M) = \{ \ w | \ w \text{ has an even number of 1s} \}$
Sample Regular Languages

- $\Sigma^*$
- $\emptyset$
- $\{ w \mid w \text{ ends in a } 1 \}$
- $\{ w \mid w \text{ has an even number of } 1\text{s} \}$
Points to Ponder

For a DFA $M = (Q, \Sigma, \delta, q_0, F)$

- Number of states, $|Q|$, is finite
- Is $\Sigma$ a finite set?
- Is $L(M)$ a finite set?
  - Does it have a finite representation?
- Suppose you are given the 5-tuple in advance, so you know $|Q|$, etc. Do you know how long you will have to run $M$ on any input? Input length is finite, but unbounded by parameters of $M$
Operations on Regular Languages

Given: Two regular languages A and B

- **Union**: $A \cup B = \{w \mid w \in A \text{ or } w \in B\}$
- **Intersection**: $A \cap B = ?$
- **Complementation**: $\overline{A} = \{w \mid w \not\in A\}$
- **Reverse**: $A^R = \{w_1w_2\ldots w_k \mid w_kw_{k-1}\ldots w_1 \in A\}$
- **Concatenation**:
  $$A \cdot B = \{vw \mid v \in A \text{ and } w \in B\}$$
- **Star**:
  $$A^* = \{w_1w_2\ldots w_k \mid k \geq 0 \text{ and each } w_i \in A\}$$
Closure

• If you perform an operation on one/more regular languages, is the result also a regular language?

• Example:
  Is $A \cup B$ regular if $A$ and $B$ are regular? 
  YES
Union Theorem

The union of two regular languages is also a regular language

Proof (Hints)?
An Example
Closure under Reverse

• Reverse: \( A^R = \{w_1w_2...w_k \mid w_kw_{k-1}...w_1 \in A\} \)

• Regular languages are closed under reverse. Here’s an attempt to prove it:
  – Given M that recognizes A
  – What if you could “run it backwards”?
  – Construct \( M^R \) as M with all arrows reversed & accept state becomes start state
  – Is \( M^R \) guaranteed to be a DFA?
What happens with 100?

We will say that the NFA accepts if there is a way to make it reach an accept state.
A non-deterministic finite automaton (NFA) is also a 5-tuple \( M = (Q, \Sigma, \delta, q_0, F) \)

- \( Q \) is the set of states
- \( \Sigma \) is the alphabet
- \( \delta : Q \times \Sigma_\varepsilon \to 2^Q \) is the transition function
- \( q_0 \in Q \) is the start state
- \( F \subseteq Q \) is the set of accept states

\( 2^Q \) is the set of subsets of \( Q \) and \( \Sigma_\varepsilon = \Sigma \cup \{\varepsilon\} \)

\( L(M) \) defined in the same way as for DFAs
NFA Example

How does this differ from the DFAs we’ve seen?

- Epsilon transitions
- Missing transitions from some states
- More than 1 transition from a state on same symbol

What’s its language?
NFAs and DFAs

• Is every DFA also an NFA?
• Is every NFA a DFA?
Tree of Computations

Deterministic Computation

accept or reject

Non-Deterministic Computation

accept

reject

stuck
Where does non-determinism arise?

- In practice: From lack of information
  - A precise model might use randomness (probabilistic transitions between states), but we might not know the probabilities
  - Important! Non-determinism is *not* the same as probabilistic computation.

- NFAs and DFAs are “equivalent”!
  - Their languages are the same
    - More on this next time
  - But NFAs are simpler
NFAs are simpler than DFAs

An NFA that recognizes the language \{1\}:

A DFA that recognizes the language \{1\}:
Union Theorem for NFAs

• If you have two NFAs $N_1$ and $N_2$, with languages $L_1$ and $L_2$, how can we get the NFA for $L_1 \cup L_2$?
Next Steps

• Read Sipser 1.2 in preparation for next lecture