CSCI 3434: Theory of Computation

Lecture 02: Regular Languages and Finite Automata

Ashutosh Trivedi (ashutosh.trivedi@colorado.edu)
Department of Computer Science, University of Colorado Boulder
Alphabet, Strings, and Languages

• An alphabet $\Sigma = \{a, b, c\}$ is a finite set of letters/symbols.

• A string over an alphabet $\Sigma$ is finite sequence of symbols, e.g.
  - sequences $cab$, $baa$, and $aaa$ are some strings over $\Sigma = \{a, b, c\}$
  - sequences $\epsilon$, 0, 1, 00, and 01 are some strings over $\Sigma = \{0, 1\}$

• $\Sigma^*$ is the set of all strings over $\Sigma$, e.g. $aabbaa \in \Sigma^*$,

• Naturally, A language $L$ is a collection/set of strings over some alphabet, i.e. $L \subseteq \Sigma^*$ e.g.,
  - $L_{\text{even}} = \{w \in \Sigma^* : w$ is of even length$\}$
  - $L_{\{a^n b^n\}} = \{w \in \Sigma^* : w$ is of the form $a^n b^n$ for $n \geq 0\}$
Programs to Accept Languages

• We say that a program accepts a language, if for every string input, it returns yes, if the string is in the language and no otherwise.

• Let’s write programs to accept languages $L_{\text{even}}$ and $L_{\{a^n b^n\}}$.

• Let’s consider more fancy languages:
  • Language of all three colorable graphs
  • Language of all prime numbers
  • Language of all sorted arrays
  • Language of all true theorems in a given logic

• Can we write programs to accept all the languages?
  • Decidable and Undecidable Languages
Existence of undecidable languages.

Proof.

1. Set of all programs over some instruction:
   - \( P = \{\epsilon, 0, 1, 00, 01, 10, 11, \ldots\} \) over binary alphabet \( \Sigma = \{0, 1\} \).

2. Set of all strings over alphabet \( \Sigma = \{0, 1\} \):
   - \( S = \{\epsilon, 0, 1, 00, 01, 10, 11, \ldots\} \)

3. Set of all languages over alphabet \( \Sigma = \{0, 1\} \) are \( 2^S \) e.g,
   - \( L_1 = \{0, 1\}, \ L_2 = \{00, 11, 11, 10\}, \ L_3 = \emptyset \)

4. From Cantor’s theorem, we know that \( |2^S| > |S| \).

5. There must be some languages for which one can not write acceptor programs, i.e. undecidable languages.
Programming Exercise-I

String Matching Problem $MATCH(A, B)$

**Input:** Lists of strings $A = \langle s_1, s_2, \ldots, s_n \rangle$ and $B = \langle t_1, t_2, \ldots, t_n \rangle$ of equal length.

**Output:** YES if there is a sequence of combining elements that produces same string for both lists. NO, otherwise.

Formally, decide whether there exists a finite sequence (of any length)

\[ 1 \leq i_1, i_2, \ldots, i_m \leq n \]

such that

\[ s_{i_1}s_{i_2}\ldots s_{i_n} = t_{i_1}t_{i_2}\ldots t_{i_n}. \]
Example 1:

**Input:** \(A = \langle 110, 0011, 0110 \rangle\) and \(B = \langle 110110, 00, 110 \rangle\).

**Output:** YES.

Since sequence 2, 3, 1 gives the same strings

\[ s_2 s_3 s_1 = 00110110110 \quad \text{and} \quad t_2 t_3 t_1 = 00110110110 \]

Example 2:

\(A = \langle 0011, 11, 1101 \rangle\) and \(B = \langle 101, 011, 110 \rangle\).

Example 3:

\(A = \langle 100, 0, 1 \rangle\) and \(B = \langle 1, 100, 0 \rangle\).
Finite State Automata
Finite State Automata

- Introduced first by two neuro-psychologists Warren S. McCullough and Walter Pitts in 1943 as a model for human brain.
- Finite automata can naturally model microprocessors and even software programs working on variables with bounded domain.
- Capture so-called regular languages that occur in many different fields (regular expression, monadic second-order logic, algebra).
- Nice theoretical properties.
- Applications in digital circuit/protocol verification, compilers, pattern recognition, etc.
Calculemus!

Let us observe our mental process while we compute the following:

• Recognize language of strings of an even length.
• Recognize language of binary strings with an even number of 0’s.
• Recognize language of binary strings with an odd number of 0’s.
• Recognize language of strings containing your identikey.
• Recognize language of binary (decimal) strings multiple of 2.
• Recognize language of binary (decimal) strings multiple of 3.
• Recognize language of binary strings with equal number of 0’s and 1’s.
• Recognize language of binary strings of the form $0^n1^n$
• Recognize language of binary strings with a prime number of 1’s
Finite State Automata: Examples

1. Automaton accepting strings of even length:

![Diagram of an FSM accepting strings of even length]
Finite State Automata: Examples

2. Automaton accepting strings with an even number of 1’s:
Finite State Automata: Examples

3. Automaton accepting binary strings characterizing an even number:
A finite state automaton is a tuple $(S, \Sigma, \delta, s_0, F)$, where:

- $S$ is a finite set called the states,
- $\Sigma$ is a finite set called the alphabet,
- $\delta : S \times \Sigma \to S$ is the transition function,
- $s_0 \in S$ is the start state, and
- $F \subseteq S$ is the set of accept states.

**Example:** The automaton in the figure above can be represented as $(S, \Sigma, \delta, s_0, F)$, where

- $S = \{E, O\}$, $\Sigma = \{0, 1\}$, $s_0 = E$, $F = \{E\}$,
- and transition function $\delta$ is such that
  - $\delta(E, 0) = E$,
  - $\delta(E, 1) = O$, and
  - $\delta(O, 0) = O$,
  - $\delta(O, 1) = E$. 

![Finite State Automata Diagram](image-url)
State Diagram

Let’s draw the state diagram of the following automaton \((S, \Sigma, \delta, s_1, F)\):

- \(S = \{s_1, s_2, s_3\}\)
- \(\Sigma = \{0,1\}\)
- \(\delta\) is given in a tabular form below:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s_1)</td>
<td>(s_1)</td>
<td>(s_2)</td>
</tr>
<tr>
<td>(s_2)</td>
<td>(s_3)</td>
<td>(s_2)</td>
</tr>
<tr>
<td>(s_3)</td>
<td>(s_2)</td>
<td>(s_2)</td>
</tr>
</tbody>
</table>

- \(s_1\) is the initial state, and
- \(F = \{s_2\}\).

Which language does it accept?
Semantics (Meaning) of Finite State Automata

A finite state automaton (DFA) is a tuple $(S, \Sigma, \delta, s_0, F)$, where:

- $S$ is a finite set called the states,
- $\Sigma$ is a finite set called the alphabet,
- $\delta : S \times \Sigma \rightarrow S$ is the transition function,
- $s_0 \in S$ is the start state, and
- $F \subseteq S$ is the set of accept states.

- A computation or a run of a DFA on a string $w = a_0 a_1 ... a_{n-1}$ is the finite sequence
  
  $s_0, a_1, s_1, a_2, ..., a_{n-1}, s_n$

  where $s_0$ is the starting state, and $\delta(s_{i-1}, a_i) = s_{i+1}$.

- A run is accepting if the last state of the unique computation is an accept state, i.e. $s_n \in F$.

- The Language of a DFA $A$
  
  $L(A) = \{w :$ the unique run of $A$ on $w$ is accepting$\}$.

- A language is called regular if it is accepted by a finite state automata.
Examples

- Recognize language of **strings of an even length**.
- Recognize language of binary strings with an **even number of 0’s**.
- Recognize language of binary strings with an **odd number of 0’s**.
- Recognize language of strings containing your **identikey**.
- Recognize language of binary (decimal) strings **multiple of 2**.
- Recognize language of binary (decimal) strings **multiple of 3**.
- Recognize language of binary strings with **equal number of 0’s and 1’s**. ❌
- Recognize language of binary strings of the form $0^n1^n$ ❌
- Recognize language of binary strings with a **prime number of 1’s** ❌
- Recognize language of binary strings that **end with a 0**.
- Recognize language of binary strings that **begin with a 1**.
Properties of Regular Languages

- Let $A$ and $B$ be languages (remember they are sets). We define the following operations on them:
  - **Union**: $A \cup B = \{w : w \in A \text{ or } w \in B\}$
  - **Concatenation**: $AB = \{wv : w \in A \text{ and } v \in B\}$
  - **Closure (Kleene Closure, or Star)**:
    \[ A^* = \{w_1w_2...w_k : k \geq 0 \text{ and } w_i \in A\}. \]
    Or, $A^* = \bigcup_{i \geq 0} A_i$ where $A_0 = \emptyset, A_1 = A, A_2 = AA$, and so on.

- Define the notion of a set being **closed under an operation** (say, $\mathbb{N}$ and $\times$).

**Theorem.** Regular languages are closed under union, intersection, concatenation, and Kleene star.