Lecture 2

Last time:
• Finite Automata

Today:
• Finite Automata
• Operations on languages
• Nondeterminism

Tomorrow:
Homework 0 due
Homework 1 out

Sofya Raskhodnikova

Sofya Raskhodnikova; based on slides by Nick Hopper
Review: FAs

Diagram:

- States: $q_0$, $q_1$
- Transitions:
  - $q_0 \xrightarrow{0} q_0$
  - $q_0 \xrightarrow{1} q_1$
  - $q_1 \xrightarrow{1} q_1$
  - $q_1 \xrightarrow{0} q_0$

Sofya Raskhodnikova; based on slides by Nick Hopper
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 $M$ 

$= \text{set of all strings machine } M \text{ accepts}$

$M \text{ recognizes the language } L(M)$
Formal definition of FA

\[ M = (Q, \Sigma, \delta, q_0, F) \]

where

- \( Q = \{q_0, q_1, q_2, q_3\} \)
- \( \Sigma = \{0, 1\} \)
- \( \delta : Q \times \Sigma \rightarrow Q \) is the transition function
- \( q_0 \in Q \) is the start state
- \( F = \{q_1, q_2\} \subseteq Q \) are the accept states

\[ \delta \]

<table>
<thead>
<tr>
<th>( \delta )</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( q_3 )</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Regular languages

A language is **regular** if it is recognized by a finite automaton

\[ L = \{ w \mid w \text{ contains 001} \} \text{ is regular} \]
\[ L = \{ w \mid w \text{ has an even number of 1s} \} \text{ is regular} \]

Many interesting programs recognize regular languages

NETWORK PROTOCOLS

COMPILERS

GENETIC TESTING

ARITHMETIC
Let $\text{TCPS} = \{ w \mid w \text{ is a complete TCP Session} \}$

**Theorem.** $\text{TCPS}$ is regular.
Arithmetic

\[
\text{LET } \Sigma_3 = \{ [0], [0], [0], [0], [0], [0], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1], [1] \}

- A string over \( \Sigma_3 \) has three ROWS
- Each ROW \( b_0b_1b_2...b_N \) represents the integer \( b_0 + 2b_1 + ... + 2^Nb_N \).
- Let \( \text{ADD} = \{ S | \text{ROW}_1 + \text{ROW}_2 = \text{ROW}_3 \} \)

**Theorem.** \( \text{ADD} \) is regular.
COMMENTS:
Are delimited by /* */
Cannot have NESTED /* */
Must be CLOSED by */
*/ is ILLEGAL outside a comment

COMMENTS = \{ \text{strings over \{0,1, /, *\} with legal comments} \}

Theorem. COMMENTS is regular.
DNA SEQUENCES are strings over the alphabet \{A,C,G,T\}.

A GENE $g$ is a special substring.

A GENETIC TEST searches a DNA SEQUENCE for a gene.

$\text{GeneticTest}_g = \{\text{strings over } \{A,C,G,T\} \text{ containing a copy of } g\}$

**Theorem.** GeneticTest$_g$ is regular for every gene $g$. 

*Sofya Raskhodnikova; based on slides by Nick Hopper*
Regular Operations on languages

Complement: $\overline{A} = \{ w \mid w \notin A \}$

Union:

Intersection:

Reverse:

Concatenation:

Star:

$= \{ \epsilon \} \cup A \cup AA \cup AAA \cup AAAA \cup ...$
THEOREM. The class of regular languages is **closed** under all 6 operations.

If A and B are regular, applying any of these operation yields a regular language.
Closure properties

Complement: \( \bar{A} = \{ w \mid w \notin A \} \)

**Theorem.** The complement of a regular language is also a regular language.

**Proof:**
Closure properties

Complement: $\overline{A} = \{ w \mid w \notin A \}$

Union: $A \cup B = \{ w \mid w \in A \text{ or } w \in B \}$
Theorem. The union of two regular languages is also a regular language.

Proof: Consider two regular languages $A_1$ and $A_2$. Prove that $A_1 \cup A_2$ is regular.

Let $M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$ be finite automaton for $A_1$ and $M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$ be finite automaton for $A_2$.

Construct a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes $A = A_1 \cup A_2$.
Example

\[ M = ? \]
Idea: Run both $M_1$ and $M_2$ at the same time!

$Q = \text{pairs of states, one from } M_1 \text{ and one from } M_2$

$= \{ (q_1, q_2) \mid q_1 \in Q_1 \text{ and } q_2 \in Q_2 \}$

$= Q_1 \times Q_2$

$q_0 = (q_0^1, q_0^2)$

$F = \{ (q_1, q_2) \mid q_1 \in F_1 \text{ or } q_2 \in F_2 \}$

$\delta( (q_1, q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$
Example (continued)

\[ M_1 \]

\[ M_2 \]

\[ M \text{ intersection} \]
Closure properties

Complement: \( -A = \{ w \mid w \notin A \} \)

Union: \( A \cup B = \{ w \mid w \in A \text{ or } w \in B \} \)

Intersection: \( A \cap B = \{ w \mid w \in A \text{ and } w \in B \} \)

Reverse: \( A^R = \{ w_1 \ldots w_k \mid w_k \ldots w_1 \in A \} \)
Closure under reverse

**Theorem.** The reverse of a regular language is also a regular language.

**Proof:** Let $A$ be a regular language and $M$ be a finite automaton that recognizes it.

Construct a finite automaton $M'$ recognizing $A^R$.

**Idea:** Define $M'$ as $M$ with the arrows reversed. Swap start and accept states.
Closure under reverse

$M'$ IS NOT ALWAYS A DFA!

It may have many start states.
Some states may have too many outgoing edges, or none.
Example
What happens with 100?

Nondeterministic Finite Automaton (NFA) accepts if there is a way to make it reach an accept state.