CISC 4090
Theory of Computation
Finite state machines &
Regular languages
Professor Daniel Leeds
dleeds@fordham.edu
JMH 332

Stereotypical computer
Central processing unit (CPU)
– performs all the
instructions
Memory – stores data and
instructions for CPU
Input – collects information
from the world
Output – provides
information to the world

Super-simple computers
Small number of potential inputs
Small number of potential outputs/actions
• Thermostat
• Elevator
• Vending machine
• Automatic door

Automatic door
Desired behavior
• Person approaches entryway, door opens
• Person goes through entryway, door stays open
• Person is no longer near entryway, door closes
• Nobody near entryway, door stays closed
Two states: Open, Closed
Two inputs: Front-sensor, Back-sensor

Finite state machine
Graph and table representations

<table>
<thead>
<tr>
<th></th>
<th>Front</th>
<th>Back</th>
<th>Neither</th>
<th>Both</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closed</td>
<td>Open</td>
<td>Open</td>
<td>Closed</td>
<td>Open</td>
</tr>
<tr>
<td>Open</td>
<td>Open</td>
<td>Open</td>
<td>Closed</td>
<td>Open</td>
</tr>
</tbody>
</table>

More finite state machine applications

- Text parsing
- Traffic light
- Pac-Man
- Electronic locks

Coding a combination lock

- A finite automaton $M_1$ with 3 states
- Start state $q_1$: accept state $q_2$ (double circle)
- Example accepted string: 1101

Formal definition of Finite State Automaton

A finite state automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

- $Q$ is a finite set called states
- $\Sigma$ is a finite set called 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

String ending with 1 or string end with 1 followed by even number of 0's
Describe M1 using formal definition

\[ M_1 = (Q, \Sigma, \delta, q_0, F) \]

- \( Q = \{q_1, q_2, q_3\} \)
- \( \Sigma = \{0, 1\} \)
- Start state: \( q_1 \)
- \( F = \{q_2\} \)

\[ \delta = \begin{array}{ll}
0 & 1 \\
q_1 & q_1 \\
q_2 & q_4 \\
q_3 & q_2 \\
\end{array} \]

Language of M1

If \( A \) is set of all strings accepted by \( M \), \( A \) is language of \( M \)

- \( L(M) = A \)

A machine may accept many strings, but only one language

- \( M \) accepts a string
- \( M \) recognizes a language

Describe \( L(M_1) = A \)

- \( A = \{w \mid w \text{ ends with } 1 \text{ or } w \text{ end with one } 1 \text{ followed by even number of } 0s\} \)

Describe M2 using formal definition

\[ M_2 = (Q, \{0, 1\}, \delta, q_1, \{q_2\}) \]

- \( Q = \{q_1, q_2\} \)
- Start state: \( q_1 \)

\[ \delta = \begin{array}{ll}
0 & 1 \\
q_1 & q_1 \\
q_2 & q_2 \\
\end{array} \]
What is the language of M2?
\[ L(M2) = \{ w | \text{w ends with at least one 1} \} \]

What is the language of M4?
\[ L(M4) = \{ w | \} \]

CFG practice

\[ \begin{align*}
G_1 \rightarrow & \ AB \\
A \rightarrow & \ xxA \mid y \\
B \rightarrow & \ wBw \mid z \\
G_2 \rightarrow & \ D \\
D \rightarrow & \ nD \mid o
\end{align*} \]

What is the language of M4?
(page 38, Ex. 1.11)
\[ L(M4) = \{ w | \} \]
Perform modulo arithmetic

Let \( \Sigma = \{ \text{RESET}, 0, 1, 2 \} \)

Construct M5 to accept a string only if the sum of each input symbol is multiple of 3, and RESET sets the sum back to 0
(1.13, page 39)

More modulo arithmetic

Generalize M5 to accept if sum of symbols is a multiple of \( i \) instead of 3

\( \{ q_0, q_1, q_2, q_3, \ldots, q_{i-1} \}, \{ 0,1,2, \text{RESET} \}, \delta, q_0, F \)  

Definition of M accepting a string

Let \( M = (Q, \Sigma, \delta, q_0, F) \) be a finite automaton and let \( w = w_1w_2 \cdots w_n \)

Then M accepts w if a sequence of states \( r_0, r_1, \ldots, r_n \) in \( Q \) exists with 3 conditions

- \( r_0 = q_0 \)
- \( \delta(r_i, w_{i+1}) = r_{i+1} \) for \( i = 0, 1, \ldots, n - 1 \)
- \( r_n \in F \)

Regular languages

Definition: a language is called a regular language if some finite automaton recognizes it

\[ \text{equivalently} \]

All of the strings in a regular language are accepted by some finite automaton
Designing finite automata (FAs)

- Determine what you need to remember
  - How many states needed for your task?
- Set start and finish states
- Assign transitions
- Check your solution
  - Should accept $w \in L$
  - Should reject $w \notin L$
  - Be careful about $\varepsilon$!

FA design practice!

- FA to accept language where number of 1’s is odd (page 43)
  - Should accept $w \in L$
  - Should reject $w \notin L$
  - Be careful about $\varepsilon$!

- FA to accept string with 001 as substring (page 44)
- FA to accept string with substring abab

Regular operations

Let $A$ and $B$ be languages. We define 3 regular operations:

- Union: $A \cup B = \{x|x \in A \text{ or } x \in B\}$
- Concatenation: $A \cdot B = \{xy|x \in A \text{ and } y \in B\}$
- Star: $A^* = \{x_1x_2 \cdots x_k|k \geq 0 \text{ and each } x_i \in A\}$
  - Repeat a string 0 or more times

Examples of regular operations

Let $A = \{\text{good, bad}\}$ and $B = \{\text{boy, girl}\}$

What is:

- $A \cup B = 
- A \cdot B = 
- A^* = $
Express CFG as RegEx?

$$G_2 \rightarrow D$$
$$D \rightarrow nD \mid o$$

$$G_1 \rightarrow AB$$
$$A \rightarrow xxA \mid y$$
$$B \rightarrow wBw \mid z$$

Closure

A collection of objects is closed under an operation if applying that operation to members of the collection returns an object in the collection.

Regular languages are closed under $\cup$, $\cdot$, $\ast$.

Closure of Concatenation

Theorem 1.26: The class of regular languages is closed under the concatenation operation.

• If $A_1$ and $A_2$ are regular languages, then so is $A_1 \cdot A_2$
• Challenge: How do we know when $M_1$ ends and $M_2$ begins?

Determinism vs. non-determinism

Determinism: Single transition allowed given current state and given input.

Non-determinism:
• multiple transitions allowed for current state and given input
• transition permitted for null input $\varepsilon$
NFA in action

- When there is a choice, follow all paths – like cloning
- If there is no forward arrow, path terminates and clone dies (no accept)
- NFA will “accept” if at least one path terminates at accept

Alternative thought:
- Magically pick best path from the set of options

The language of M10

- List some accepted strings
- What is $L(M10)$?

NFA construction practice

Build an NFA that accepts all strings over \{0,1\} with 1 in the third position from the end

NFA -> DFA

Build an NFA that accepts all strings over \{0,1\} with 1 in the third position from the end

Can we construct a DFA for this?
Formal definition of Nondeterministic Finite Automaton

Similar to DFA: a 5-tuple $(Q, \Sigma, \delta, q_0, F)$
- $Q$ is a finite set called states
- $\Sigma$ is a finite set called the alphabet
- $\delta: Q \times \Sigma \rightarrow P(\Sigma) \cup \epsilon$ is the transition function
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states

Describe M10 using formal definition

$M_1 = (Q, \Sigma, \delta, q_0, F)$

- $Q = \ldots$
- $\delta = \ldots$
- $\Sigma = \ldots$
- $q_0 \in Q$ is the start state
- $F \subseteq Q$ is the set of accept states

Consider NFA $N_1$

Language: $L(N_1) = \{w \mid w \text{ begins with } 0, \text{ ends with } 01, \text{ every } 1 \text{ in } w \text{ is preceded by a } 0\}$

Convert NFA $N_1$ to DFA $M_1$
Equivalence of NFAs and DFAs

NFAs and DFAs recognize the same class of languages

Two machines are equivalent if they recognize the same language

Every NFA has an equivalent DFA

Union Closure with NFAs

- Proofs by construction – fewer states!
- Any NFA proof applies to DFA

Given two regular languages \( A_1 \) and \( A_2 \) recognized by \( N_1 \) and \( N_2 \) respectively, construct \( N \) to recognize \( A_1 \cup A_2 \)

Let’s consider two languages

\( L_1 \): start with 0, end with 1
\( L_2 \): start with 1, end with 0

Construct machines for each languages
Construct machines \( N_3 \) to recognize \( L_1 \cup L_2 \)

Let’s consider two languages

\( L_1 \): start with 0, end with 1
\( L_2 \): start with 1, end with 0

\( N_1 \)

\( N_2 \)
Closure under concatenation

Given two regular languages $A_1$ and $A_2$ recognized by $N_1$ and $N_2$ respectively, construct $N$ to recognize $A_1 \cdot A_2$.

Closure of Union

Theorem 1.25: The class of regular languages is closed under the union operation.

Proof by construction.