Outline

• What is a Finite State Automata
• DFA definition
• Example DFA construction
• The language of a DFA
• Regular Operations
• Closure under union
• Closure under concatenation

From Sipser Chapter 1.1
What is a Computer?

• We start with a computational model
  – Different models will have different features and may match a real computer better in some ways and worse in others
  – Trade off between generality and precision
• Our first model is the finite state machine or finite automata
  – Model for machines with finite memory
Finite Automata

Models of computers with extremely limited memory

– Many simple computers have extremely limited memories and are in fact finite state machines
– Can you name any? Sare in this building but have nothing specifically to do with our department
  • Vending machine
  • Elevator
  • Thermostat
  • Automatic door at supermarket
Automatic Door

• What is the desired behavior? Describe the actions and then list the states.
  – Person approaches, door should open
  – Door **should** stay open while person going through
  – Door should shut if no one near the doorway
  – States are open and closed

• More details about automatic door
  – Front pad   Door   Rear Pad
  – Describe behavior now
    • Hint: action depends not just on what happens, but what **state** you are currently in
    • If you walk through, the door should stay open while you are on rear pad
    • But if door is closed and someone steps on rear pad, door does not open
# Automatic Door

<table>
<thead>
<tr>
<th></th>
<th>NEITHER</th>
<th>FRONT</th>
<th>REAR</th>
<th>BOTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOSED CLOSED</td>
<td>CLOSED</td>
<td>OPEN</td>
<td>CLOSED</td>
<td>CLOSED</td>
</tr>
<tr>
<td>OPEN CLOSED</td>
<td>CLOSED</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
</tbody>
</table>

Diagram:

- REAR, BOTH, NEITHER
- FRONT, REAR, BOTH
- FRONT
- NEITHER

9/10/19
More on Finite Automata

- How many bits of data does this FSM store?
  - 1 bit: open or closed
- What about state information for elevators, thermostats, vending machines, etc?
- FSM used in speech processing, optical character recognition, etc.
A finite automata

A finite automata M1 with 3 states

- State diagram
  - Start state $q_0$ (entering arrow from no state),
  - accept state $q_1$ (double circle),
  - and several transitions (arrows)

- M1 will accept (or recognize) a string like “1101” if it ends in accept state. Otherwise the string is rejected! What will it do?

- Can you describe all string that this model will accept?
  - It will accept all strings ending in a 1 and any string with an even number of 0’s following the last 1
Formal Definition of Finite Automata

• A finite automata 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
Formal Definition

- **M1 = (Q, Σ, δ, q₀, F)**
  - Q = \{q₀, q₁, q₂\}
  - Σ = \{0,1\}
  - q₀ is the start state
  - F = \{q₁\}

Transition function δ

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>q₀</td>
<td>q₀</td>
<td>q₁</td>
</tr>
<tr>
<td>q₁</td>
<td>q₂</td>
<td>q₁</td>
</tr>
<tr>
<td>q₂</td>
<td>q₁</td>
<td>q₁</td>
</tr>
</tbody>
</table>
The Language of M1

• The language of DFA M is the set A of all strings accepted by the DFA M
  – \( L(M) = A \)
  – We also say that M recognizes A or M accepts A

• A machine may accept many strings, but only one language

• Convention: M accepts string and recognizes a language
What is the Language of M1?

• $L(M1) = A$ or M1 recognizes A

• What is A?
  – $A = \{ w \mid \ldots \}$
  – $A = \{ w \mid w$ contains at least one 1 and an even number of 0’s follows the last 1}
What is the Language of M2?

- M2 = \{q_0, q_1\}, \{0,1\}, \delta, q_0, \{q_1\}
  - I leave \delta as an exercise
  - What is the language of M2?
    - L(M2) = \{w \mid \text{w ends in a 1}\}
What is the Language of M3?

- M3 is M2 with different accept state
- What is the language of M3?
  - $L(M3) = \{w| ? \}$
  - $L(M3) = \{w| w \text{ ends in } 0\}$ [Not quite right! Why?]
  - $L(M3) = \{w| w \text{ is the empty string } \varepsilon \text{ or ends in } 0\}$
What is the Language of M4?

What language does M4 accept?

Figure 1.12 on page 38
What is the Language of M4?

• What does M4 accept?
  – All strings that start and end with a or start and end with b
  – More simply, language is all string starting and ending with the same symbol
    • Note that length of 1 is okay
Construct M5 to do Modulo Arithmetic

- Let $\Sigma = \{\text{RESET}, 0, 1, 2\}$
- Construct M5 to accept a string only if the sum of each input symbol modulo 3 is 0 and RESET sets the sum back to 0 (1.13, page 39)
Now Generalize M5

• Generalize M5 to accept if sum of symbols is a multiple of $i$ instead of 3

  - ($\{q_0, q_1, q_2, q_3, \ldots, q_{i-1}\}, \{0,1,2,\text{RESET}\}, \delta_i, q_0, \{q_0\})$
    - $\delta_i(q_j, 0) = q_j$
    - $\delta_i(q_j, 1) = q_k$ where $k=j+1$ modulo $i$
    - $\delta_i(q_j, 2) = q_k$ where $k=j+2$ modulo $i$
    - $\delta_i(q_j, \text{RESET}) = q_o$

• Note: as long as $i$ is finite, we are okay and only need finite memory (log # of states)

• Could you generalize on $\Sigma = \{1, 2, 3, \ldots k\}$?
Formal Definition of Accept

• Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automata and let $w = w_1w_2 \ldots w_n$ be a string where $w_i \in \Sigma$.

• 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 automata recognizes it.

- That is, if there exists a finite automata that recognizes all and only the strings in the language.
Designing Finite Automata

• You will need to design FAs to accept a language

• Strategies
  – Determine what you need to remember (the states)
    • How many states to determine even/odd number of 1’s in an input?
    • What does each state represent
  – Set the start and accept states based on what each state represents
  – Assign the transitions
  – Check your solution: it should accept $w \in L$ and not accept $w$ not in $L$
  – Be careful about the empty string
Designing FAs

ALWAYS start by asking: “What do I need to remember?”

• Design a FA to accept the language of binary strings where the number of 1’s is odd, zero counts as even (page 43)
• Design a FA to accept all string with 001 as a substring (page 44)
• Design a FA to accept a string with substring \textit{abab}
Regular Operations

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

- **Union**: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
- **Concatenation**: $A \cdot B$ where $\{xy \mid x \in A \text{ and } y \in B\}$
- **Star**: $A^* = \{x_1x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

  – Star is a unary operator on a single language
  – Star repeats a string 0 or more times
Examples of Regular Operations

• Let $A = \{0, 1\}$ and $B = \{c, d\}$

• Then what is:
  
  – $A \cup B = \{0, 1, c, d\}$
  
  – $A \cdot B = \{0c, 0d, 1c, 1d\}$
  
  – $A^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$
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.

• The set of natural numbers is closed under addition and multiplication (but not division and subtraction).
Closure for Regular Languages

• Regular languages are closed under the 3 regular operators we just introduced

• Why we care?
  – If these operators are closed, then if we can implement each operator using a FA, then we can build a FA to recognize a regular expression
Closure of Union

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

- If $A_1$ and $A_2$ are regular languages then so is $A_1 \cup A_2$
- How can we prove this? Use proof by construction.
  - Assume $M_1$ accepts $A_1$ and $M_2$ accepts $A_2$
  - Construct $M_3$ using $M_1$ and $M_2$ to accept $A_1 \cup A_2$
  - We need to simulate $M_1$ and $M_2$ running in parallel and stop if either reaches an accept state
    - This last part is feasible since we can have multiple accept states
    - You need to remember where you would be in both machines
Closure of Union

• You need to generate a state to represent the state you would be in for both M1 and M2
• Let M1=(Q₁, Σ, δ₁, q₁, F₁) and M2=(Q₂, Σ, δ₂, q₂, F₂)
• Build M3=(Q, Σ, δ, q₀, F) as follows:
  – Q={(r₁,r₂)|r₁ ∈ Q₁ and r₂ ∈ Q₂} (Cartesian product)
    • Careful: order of (r₁,r₂) does not matter!
    • More correct: Q is the set of unordered pairs such that one state is in Q₁ and the other is in Q₂
  – Σ = Σ₁ ∪ Σ₂
  – q₀ is the pair (q₁, q₂)
  – F = {(r₁, r₂)|r₁ ∈ F₁ or r₂ ∈ F₂}
  – δ((r₁,r₂),a) = (δ(r₁, a), δ₂(r₂, a))
Theorem 1.26: The class of regular languages is closed under the concatenation operator

- If $A_1$ and $A_2$ are regular languages then so is $A_1 \cdot A_2$
- Can you see how to do this simply?
  - Not trivial since cannot just concatenate $M_1$ and $M_2$, where start states of $M_2$ become the finish states of $M_1$
    - Because we do not accept a string as soon as it enters the finish state (wait until string is done) it can leave and come back
    - Thus we do not know when to start using $M_2$; if we make the wrong choice will not accept a string that can be accepted
    - This proof is easy if we have nondeterministic FA
Concatenation: Simple Example

• Concatenation of the following:
  – \( L(M_1) = A \), where \( \Sigma = \{0, 1\} \) and \( A \) = binary string with exactly 2 1’s
  – \( L(M_2) = B \), where \( \Sigma = \{0, 1\} \) and \( B \) = binary string with exactly 3 1’s

• \( M_1 \) will enter accept state as soon as sees 2 1’s. It can then loop back on any 0’s or move to \( M_2 \) without issue. It can move immediately to \( M_2 \) on a 1, and not have an issue since it cannot loop back, since \( A \) accepts only exactly 2 1’s. Once in \( M_2 \) everything will work okay.
• \( L(M1) = A \), where \( \Sigma = \{0, 1\} \) and \( A = \) binary string with \textit{at least} 2 1’s
• \( L(M2) = B \), where \( \Sigma = \{0, 1\} \) and \( B = \) binary string with exactly 2 1’s
• This does not work (but easy with NFA or more complicated DFA)
  – If in M1 and see 2 1’s, enter accept state. When see another 1, have choice to loop back into accept state in M1, or start moving into M2, to the state that represents saw first 1 for string in B.
    • If the concatenated string has exactly 4 1’s total, then will only accept if move into M2 as early as possible (after seeing the first 2 1’s)
    • If the concatenated string has more than 4 1’s, then will only accept if loop in M1 accept state until only 2 1’s left.
• Note that the general procedure for putting M1 and M2 together involves superimposing the start state for M2 onto accept state of M1 and removing the original arcs in M1 for that state.