Automata Theory

Finite State Machines
Motivation

- Question: What is a computer?
- Too complex to allow a precise mathematical model
  - We introduce an abstract idealized computer
  - Computational model
    - One such a model is called finite automaton
Key ideas

- Formal definition of a finite automaton
- Deterministic vs. nondeterministic finite automaton
- Regular languages
- Operations on regular languages
- Regular expressions
- Pumping Lemma
Automata Theory

Deterministic Finite Automaton
Example: Automatic door

- Front pad
- Rear pad
- door

Opens when someone approaches
Holds open until person clears
Doesn’t open when someone is standing behind the door
Automatic door Sensors

- The door has 4 sensors
  - FRONT: someone on the front pad
  - BEHIND: someone on the rear pad
  - BOTH: someone on both pads
  - NEITHER: no one on either pad
Automatic door states

- The door has two states
  - OPEN
  - CLOSE
Automatic door: state diagram
Automatic door: state diagram

CLOSED

OPEN

FRONT
REAR
BOTH
Automatic door: state diagram

- REAR
- BOTH
- NEITHER

CLOSED

- FRONT
- REAR
- BOTH

OPEN
Automatic door: state diagram
Automatic door: state diagram

REAR
BOTH
NEITHER

FRONT

CLOSED

OPEN

NEITHER

FRONT
REAR
BOTH
Automatic door: state transition table

<table>
<thead>
<tr>
<th></th>
<th>NEITHER</th>
<th>FRONT</th>
<th>REAR</th>
<th>BOTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOSED</td>
<td>CLOSED</td>
<td>OPEN</td>
<td>CLOSED</td>
<td>CLOSED</td>
</tr>
<tr>
<td>OPEN</td>
<td>CLOSED</td>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
</tbody>
</table>
The machine M has

- States: $q_1, q_2,$ and $q_3$
- Start state: $q_1$ (incoming arrow from nowhere)
- Accept state: $q_2$ (thick circle)
- State transitions: arrows
The machine $M$ on input string
- begins in start state $q_1$
- after reading each symbol, 1 takes transition with matching symbol
Informal definition

- After processing all the input string machine M produces output
  - accept if M is in an accepting state
  - reject otherwise
Informal definition

- What happens on input string?
  - 1101
  - 0010
  - 01100
Informal definition

- What happens on input string?
  - 1101
Informal definition

- What happens on input string?
  - 1101
Informal definition

- What happens on input string?
  - 1101
Informal definition

- What happens on input string?
  - 1101
  - Accept

q1 \rightarrow 0 \rightarrow q1
q1 \rightarrow 1 \rightarrow q2
q2 \rightarrow 1 \rightarrow q2
q2 \rightarrow 0 \rightarrow q3
q3 \rightarrow 0,1 \rightarrow q3
q3 \rightarrow 0,1 \rightarrow q3
Informal definition

What happens on input string?
- $\emptyset 010$
Informal definition

- What happens on input string?
  - $\emptyset\emptyset10$
Informal definition

- What happens on input string?
  - 0010
Informal definition

What happens on input string?
- 0010
- 0,1
- Reject
Can you describe the language consisting of all string that M actually accepts?
Informal definition

- $M$ actually accepts
  - any string that ends with a 1
    - $1,01,11,01010101010101,...$
  - any string that ends with an even number of 0’s after the last 1
    - $100, 101010000,...$
Formal Definitions

- A finite automaton is a 5-tuple \((Q, \Sigma, \xi, q_0, F)\), where
  - \(Q\) is a finite set of states
  - \(\Sigma\) is a finite set of symbols called the alphabet
  - \(\xi : Q \times \Sigma \rightarrow Q\) is the transition function
  - \(q_0 \in Q\) is the start state
  - \(F \subseteq Q\) is the set of accept states or final states
M is a 5-tuple $(Q, \Sigma, \xi, q_1, F)$, where

- $Q$ is a finite set of states $\{q_1, q_2, q_3\}$
- $\Sigma$ is $\{0, 1\}$
- $\xi : Q \times \Sigma \rightarrow Q$ is the transition function
- $q_1 \in Q$ is the start state
- $F \subseteq Q$ is $\{q_2\}$

\[
\begin{array}{c|cc}
\xi & 0 & 1 \\
\hline
q_1 & q_1 & q_2 \\
q_2 & q_3 & q_2 \\
q_3 & q_2 & q_2 \\
\end{array}
\]
M1 is a 5-tuple \((Q, \Sigma, \xi, q_1, F)\), where
- \(Q\) is a finite set of states \(\{q_1, q_2\}\)
- \(\Sigma = \{0, 1\}\)
- \(\xi: Q \times \Sigma \rightarrow Q\) is the transition function
- \(q_1 \in Q\) is the start state
- \(F \subseteq Q\) is \(\{q_2\}\)
Another example: M1

What language does M1 accept?
Another example: M1

What language does M1 accept?

- M1 accepts all the strings that end with a 1
Another example: $M_2$
Another example: M2

- What language does M2 accept?
Another example: M2

- What language does M2 accept?

- M1 accepts all the strings that start and end with the same symbol!
Languages

- The language of machine $M$ is the set of strings $A$ that $M$ accepts
  - $L(M) = A$
- Note that
  - $M$ can accept many strings
  - ... but $M$ accepts only one language
- We say that $M$ recognizes language $A$ if
  - $A = \{w \mid M \text{ accepts } w\}$
A language is called a regular language if some finite automaton recognizes it.
Formal model of computation

- Let $M = (Q, \Sigma, \xi, q_0, F)$ be a Deterministic Finite Automaton (DFA)
- Let $w = w_1w_2\ldots w_n$ be a string over $\Sigma$
- $M$ accepts $w$ if a sequence of states $r_0, \ldots, r_n$ exist such that:
  1. $r_0 = q_0$
  2. $\xi(r_i, w_{i+1}) = r_{i+1}, \ i < n$
  3. $r_n \in F$
Formal model of computation

- Let $M = (Q, \Sigma, \xi, q_0, F)$ be a Deterministic Finite Automaton (DFA).
- Let $w = w_1w_2...w_n$ be a string over $\Sigma$.
- $M$ accepts $w$ if a sequence of states $r_0, ..., r_n$ exist such that:
  1. $r_0 = q_0$: the machine starts in the start state.
  2. $\xi(r_i, w_{i+1}) = r_{i+1}$, $i < n$: the machines goes from state to state according to the transition function.
  3. $r_n \in F$: machine accepts its input if it ends up in an accept state.
The regular operations

- Union operation

- $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

- It takes all strings from $A$ and $B$ and puts them together into one language
The regular operations

- Union operation
  
  \[ A \cup B = \{ x \mid x \in A \text{ or } x \in B \} \]
  
  It takes all string in A and B and puts them together into one language

- Let \( A = \{\text{good, bad}\} \) and \( B = \{\text{boy, girl}\} \)
  
  \[ \Rightarrow A \cup B = \{\text{good, bad, boy, girl}\} \]
The regular operations

- Concatenation operation

- $A \circ B = \{xy| x \in A \text{ and } y \in B\}$

- It attaches a string from $B$ to the end of a string from $A$ in the new language
The regular operations

- **Concatenation operation**

  \[ A \circ B = \{xy \mid x \in A \text{ and } y \in B\} \]

  - It attaches a string from B to the end of a string from A in the new language

  - Let \( A = \{\text{good, bad}\} \) and \( B = \{\text{boy, girl}\} \)

  \[ \Rightarrow A \circ B = \{\text{goodboy, goodgirl, badboy, badgirl}\} \]
The regular operations

- Star operation

- \[ A^* = \{x_1x_2 \ldots x_k | \text{k is positive and each } x_i \in A \} \]

- It applies to a single language

- Attaches any number of string from A to get a string in the new language
The regular operations

- Star operation

- $A^* = \{x_1x_2...x_k | k \text{ is positive and each } x_i \in A}\$

- It applies to a single language
- Attaches any number of string from $A$ to get a string in the new language

- Let $A = \{\text{good, bad}\}$
- $A^* = \{\emptyset, \text{good, bad, goodgood, goodbad, badbad, badgood, goodgoodgood, goodgoodbad, goodbadgood, ...}\}$
The set of natural number \( N = \{1, 2, 3, \ldots \} \) is closed under multiplication:
- For any \( a \) and \( b \) in \( N \), \( a \times b \) is also in \( N \)
  - \( 3 \times 5 = 15 \): 3, 5, and 15 are all naturals

In general, a collection of objects is closed under some operation if applying that operation to members of the collection returns and object still in the collection.

Note; the set of natural numbers is not closed under division:
- \( 3 \div 5 \) is not a natural number
Regular languages closed under union

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

- If $A_1$ and $A_2$ are regular languages, so is $A_1 \cup A_2$
Proof idea

- How to prove such a theorem?
  - Remember that the only result we have seen so far is
    - “A language is called a regular language if some finite automaton recognizes it”
Proof idea

- “A language is called a regular language if some finite automaton recognizes it”
- So, let us look at our theorem from this perspective:
  - If $A_1$ and $A_2$ are regular languages, so is $A_1 \cup A_2$
Proof idea

- “A language is called a regular language if some finite automaton recognizes it”
- So, let us look at our theorem from this perspective:
  - If $A_1$ and $A_2$ are regular languages, so is $A_1 \cup A_2$
Proof idea

- “A language is called a regular language if some finite automaton recognizes it”
- So, let us look at our theorem from this perspective:
  - If A1 and A2 are regular languages, so is A1 U A2
Proof: approach

- Approach: proof by construction
  - Some M1 accepts A1
  - Some M2 accepts A2
  - Construct M that accepts A1 U A2
Proof: first attempt

- Given M1 and M2
  - First simulate M1
  - If M1 doesn’t accept then simulate M2
Proof: first attempt

- Given M1 and M2
  - First simulate M1
  - If M1 doesn’t accept then simulate M2

- What is wrong with this approach?
  - Once the symbols are read by M1, they are lost!
  - \(\rightarrow\) We cannot simulate M2!
  - We need another way
Proof: a second attempt

- We need another way:
  - Simulate both machines simultaneously!
  - That way only one pass through the input is necessary
- But, can we keep track of both simulations with finite memory?
Proof: a second attempt

But, can we keep track of both simulations with finite memory?

- Answer: All you need to remember is the state that each machine would be in if it had read up to this point in the input
  - You need to remember a pair of states!
Proof: a second attempt

M1

\[ q_1 \rightarrow q_2 \rightarrow r_1 \rightarrow r_2 \]

M2

M

q1, r1
Proof: a second attempt

M1

q1 → q2 → r1 → r2

M2

q1, r1 ← q1, r2

M
Proof: a second attempt
Proof: a second attempt
Proof: a second attempt

M1

q1 \rightarrow q2

M2

r1 \rightarrow r2

M1, q1, r1 \rightarrow q1, r1

M1, q1, r2 \rightarrow q1, r2

M2, q2, r1 \rightarrow q2, r1

M2, q2, r2 \rightarrow q2, r2

M
Proof: a second attempt: Accept states

M1

q1  q2

M2

r1  r2

q1,r1  q1,r2

q2,r1  q2,r2

M
Proof: a second attempt: Accept sates

M1

q1 → q2

M2

r1 → r2

q1,r1
q1,r2
q2,r1
q2,r2

M
Proof: formal details

- Let $M_1 = (Q_1, \Sigma, \xi_1, q_1, F_1)$ accept $A_1$
- Let $M_2 = (Q_2, \Sigma, \xi_2, q_2, F_2)$ accept $A_2$
- Define $M = (Q, \Sigma, \xi, q, F)$ as follows

1. $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$

This set is the Cartesian product of sets $Q_1$ and $Q_2$

Denoted by $Q_1 \times Q_2$
Proof: formal details

- Let $M_1 = (Q_1, \Sigma, \xi_1, q_1, F_1)$ accept $A_1$
- Let $M_2 = (Q_2, \Sigma, \xi_2, q_2, F_2)$ accept $A_2$
- Define $M = (Q, \Sigma, \xi, q, F)$ as follows

1. …
2. $\Sigma$ is the same
Proof: formal details

- Let $M_1 = (Q_1, \Sigma, \xi_1, q_1, F_1)$ accept $A_1$
- Let $M_2 = (Q_2, \Sigma, \xi_2, q_2, F_2)$ accept $A_2$
- Define $M = (Q, \Sigma, \xi, q, F)$ as follows

1. ...
2. ...
3. $\xi$, the transition function is defined as follows
   \[
   \xi((r_1, r_2), a) = (\xi_1(r_1, a), \xi_2(r_2, a))
   \]
Proof: formal details

- Let $M_1 = (Q_1, \Sigma, \xi_1, q_1, F_1)$ accept $A_1$
- Let $M_2 = (Q_2, \Sigma, \xi_2, q_2, F_2)$ accept $A_2$
- Define $M = (Q, \Sigma, \xi, q, F)$ as follows

1. ...
2. ...
3. ...
4. $q$ is the pair $(q_1, q_2)$
Proof: formal details

- Let $M_1 = (Q_1, \Sigma, \xi_1, q_1, F_1)$ accept $A_1$
- Let $M_2 = (Q_2, \Sigma, \xi_2, q_2, F_2)$ accept $A_2$
- Define $M = (Q, \Sigma, \xi, q, F)$ as follows
  
  1. ...
  2. ...
  3. ...
  4. ...
  5. $F$ is the set of pairs in which either member is an accept state of $M_1$ or $M_2$
     \[ F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\} \]
Proof: formal details

- Let $M_1 = (Q_1, \Sigma, \xi_1, q_1, F_1)$ accept $A_1$
- Let $M_2 = (Q_2, \Sigma, \xi_2, q_2, F_2)$ accept $A_2$
- Define $M = (Q, \Sigma, \xi, q, F)$ as follows

1. ...
2. ...
3. ...
4. ...
5. Why not choose $F$ as follows
   
   \[ F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ and } r_2 \in F_2\} = F_1 \times F_2 \]
What about concatenation?

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

  - If $A_1$ and $A_2$ are regular languages, so is $A_1 \circ A_2$.
Proof Idea

- Simulate M1 for a while, then switch to M2
- BUT: when to switch?
- … leads us into nondeterminism
Conclusions

Deterministic Finite State Machines

Regular Languages

Regular Operations
  Union
  Concatenation
  Star

Next time: Nondeterministic FA