Deterministic Finite Automata

Logistics

- E-mail
  - Should have received mail from me.
  - If not:
    - Check LDAP entry
    - Indicate on attendance sheet

Announcements

- It’s Co-op time
  - Orientation
    - Wednesday, Sept 17th, 4:00 pm – 5:30 pm, 77-A190
    - Thursday, Sept 18th, 12:00 pm – 1:30 pm, 70-1455
    - Friday, Sept 19th, 1:00 pm – 2:30 pm, 76-1125
    - Friday, Sept 26th, 1:00 pm – 2:30 pm, 76-1125
    - Tuesday, Oct 7th, 1:00 pm – 2:30 pm, 76-1125
    - Wednesday, Oct 22nd, 4:00 pm – 5:30 pm, 77-A190
- Job Fair
  - Wednesday, Oct 15, 1-6pm, Clark Gym

First things

- Homework #1
  - From textbook
    - Exercise 2.2.4b and c (pg 54)
      - Given formal n-tuple definition with transitions given in a graph
    - Exercise 2.2.7 (pg 54)
    - Exercise 2.3.2 (pg 66)
    - Exercise 2.5.2 (pg 80)
      - Given formal n-tuple definition with transitions given in a graph

Questions

- Any questions before we start?

Languages

- Recall
  - What is a language?
String Recognition machine

- Given a string and a definition of a language (set of strings), is the string a member of the language?

Input string → Language recognition machine → YES, string is in Language → NO, string is not in Language

Language Classes

- Recall, we will be looking at classes of languages:
  - Each class will have its own means for describing the language
  - Each class will have its own machine model for string recognition
  - Languages and machines get more complex as we move forward in the course.

Languages

- A language is a set of strings.
- A class of languages is nothing more than a set of languages

Regular Languages

- Today we start looking at our first class of languages: Regular languages
  - Machine for accepting: Finite Automata
  - Means of defining: Regular Expressions

- We’ll be studying different aspects of regular languages until Exam 1.

Deterministic Finite Automata

- A deterministic finite automata (DFA) consists of
  - A read tape
  - A machine that can be in one or more “states”
    - Start state – The state the machine is in at the beginning of execution
    - Accepting states – The state(s) the machine has to be in after execution in order for a string to be “accepted”
Deterministic Finite Automata

• How the automaton works
  – Read a character on the tape
  – Based on the character read and the current “state” of the machine, put your machine into another “state”
  – Move the read head to the right
  – Repeat the above until all characters have been read.

• Testing a string for membership
  – Place the string to be tested on the read tape
  – Place the machine into the start “state”
  – Let the machine run to completion
  – If, upon completion, the machine is in an accepting “state”, the string is accepted, otherwise it is not.

• Transition function
  – Defines what state the machine will move into given:
    • The current machine state
    • The character read off the tape
  – Sometimes illustrated as directed graph where nodes represent states and edges represent transitions.

Deterministic Finite Automata

Example: Tic Tac Toe

\[
\begin{array}{cccc}
0 & 1 & 2 \\
3 & 4 & 5 \\
6 & 7 & 8 \\
\end{array}
\]

\[
\begin{array}{cccc}
a & b & c \\
d & e & f \\
g & h & i \\
\end{array}
\]

\[
\begin{array}{cccc}
A & B & C \\
D & E & F \\
G & H & I \\
\end{array}
\]

Deterministic Finite Automata

\[
\begin{array}{c|c|c|c}
0 & 1 & \text{#0s} & \text{#1s} \\
0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 \\
2 & 0 & 1 & 0 \\
3 & 0 & 0 & 1 \\
\end{array}
\]

Deterministic Finite Automata

• This transition can also be given by a table.

\[
\begin{array}{c|c|c}
\text{state} & 0 & 1 \\
\hline
0 & 0 & 3 \\
1 & 0 & 2 \\
2 & 3 & 1 \\
3 & 2 & 0 \\
\end{array}
\]
Finite Automata Visualization

- JFLAP
  - Java Formal Language and Automata Package
  - By Susan Rodger at Duke University

Deterministic Finite Automata

- Demo using FLAP

Deterministic Finite Automata

- Another example
  - Let’s see that in action.

Deterministic Finite Automata

- Consists of
  - A set of states
  - A start state
  - A set of accepting states
  - Read symbols
  - Transition function

- Let’s define an automata formally

Deterministic Finite Automata

- A deterministic finite automaton (finite-state machine) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where
  - $Q$ is a finite set (of states)
  - $\Sigma$ is a finite alphabet of symbols
  - $q_0 \in Q$ is the start state
  - $F \subseteq Q$ is the set of final states
  - $\delta$ is a function from $Q \times \Sigma$ to $Q$ (transition function)
Transition Function

- The transition function
  - \( \delta \) is a function from \( Q \times \Sigma \) to \( Q \)
  - \( \delta(q, a) = q' \) where
    - \( q, q' \in Q \)
    - \( a \in \Sigma \)
  - \( \delta \) defines, given a current state \( q \) and reading character \( a \), to which state the DFA will move.

Transition Function on Strings

- Applying the transition function to a string.
  - \( \delta \) is a function from \( Q \times \Sigma^* \) to \( Q \)
  - \( \delta(q, x) = q' \) where
    - \( q, q' \in Q \)
    - \( x \in \Sigma^* \)
  - \( \delta \) defines, given a current state \( q \) and reading a string \( x \), to which state the DFA will move once all characters of \( x \) are read.

Transition Function on Strings

- Recursive definition of \( \delta \)
  - Let \( A = (Q, \Sigma, \delta, q_0, F) \) be an FA.
  - Define \( \delta : Q \times \Sigma^* \rightarrow Q \)
    1. For any \( q \in Q \), \( \delta(q, \epsilon) = q \)
    2. For any \( y \in \Sigma^* \), \( a \in \Sigma \), and \( q \in Q \)
       \( \delta(q, ya) = \delta(\delta(q, y), a) \)

Example:
\[
\delta(q, abc) = \delta(\delta(q, ab), c) = \delta(\delta(\delta(q, a), b), c) = \delta(\delta(q_1, b), c) = \delta(q_2, c) = q_3
\]

Language accepted by a DFA

- Let \( A = (Q, \Sigma, \delta, q_0, F) \) be a DFA.
- A string \( x \in \Sigma^* \) is accepted by \( A \) if
  - \( \delta(q_0, x) \in F \)
  - In other words,
    - Starting in your start state \( q_0 \)
    - Run the machine with input \( x \)
    - The machine ends up in a final state
  - If a string \( x \) is not accepted by \( A \) it is said to be rejected by \( A \).

Language accepted by a DFA

- The language accepted or recognized by \( A \) is:
  - \( L(A) = \{ x \in \Sigma^* \mid x \text{ is accepted by } A \} \)
- If \( L \) is a language over \( \Sigma \), \( L \) is accepted by \( A \) iff \( L = L(A) \).
  - For all \( x \in L \), \( x \) is accepted by \( A \).
  - For all \( x \not\in L \), \( x \) is rejected by \( A \).
Reality Check

• What we’ve learned so far
  – An DFA can be expressed by \((Q, \Sigma, \delta, q_0, F)\)
  – Transition function: \(\delta: Q \times \Sigma \rightarrow Q\)
  – Transition function on strings
    \(\delta: Q \times \Sigma^* \rightarrow Q\)
    • Defined recursively
  – DFA accepting a string
  – The language accepted by an DFA

Questions?

Let’s take a break.