Deterministic Finite Automata

Logistics

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

First things

- Homework #1
  - From textbook
    - Exercise 2.2.5 a,b,c (use JFLAP)
    - Exercise 2.2.7
    - Exercise 2.3.2
    - Exercise 2.5.2
    - Exercise 2.5.3 a, b, c (use JFLAP)
  - Due 9/21

Finite Automata Visualization

- JFLAP
  - Java Formal Language and Automata Package
  - By Susan Rodger at Duke University
  - http://www.cs.duke.edu/~rodger/tools/jflap
  - Available on mycourses FILES section.
- Running JFLAP
  - UNIX
    - java –jar JFLAP.jar
    - java –jar ~jmg/JFLAP.jar
  - Windows: double click on JFLAP.jar

Submitting JFLAP files

- Via e-mail (jmg@cs.rit.edu)
  - Use obvious file names
    - Eg. 225a.DFA, 225b.DFA, etc.
    - Need DFA extension!
  - UNIX: tar file
  - Windows: zip file
  - Include a README with name(s)

Questions

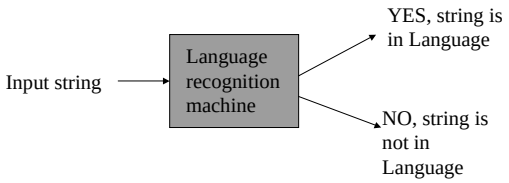
- Any questions before we start?

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?



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

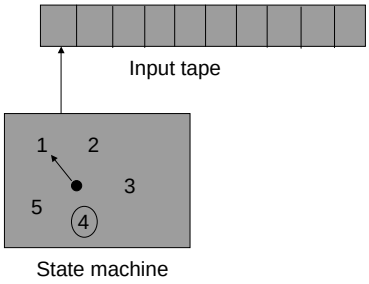
Regular Languages

- 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 for the next 3-4 weeks

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



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.

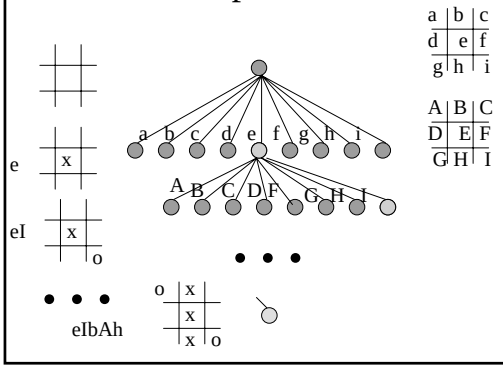
Deterministic Finite Automata

- 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.

Deterministic Finite Automata

- 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.

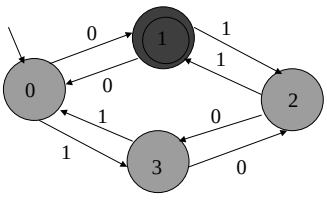
Example: Tic Tac Toe



Deterministic Finite Automata

- L = {x ∈ {0,1}* | x contains an odd number of 0s and an even number of 1s}

	# 0s	#1s
0	even	even
1	odd	even
2	odd	odd
3	even	odd



Deterministic Finite Automata

- This transition can also be given by a table.

state \ character	0	1
→ 0	1	3
*1	0	2
2	3	1
3	2	0

Finite Automata Visualization

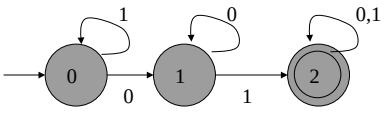
- JFLAP
  - Java Formal Language and Automata Package
  - By Susan Rodger at Duke University
  - http://www.cs.duke.edu/~rodger/tools/jflap

Deterministic Finite Automata

- Demo using FLAP

Deterministic Finite Automata

- Another example
  - L = $\{x \in \{0,1\}^* \mid x \text{ contains 01 as a substring}\}$



Deterministic Finite Automata

- Another example
  - L = $\{x \in \{0,1\}^* \mid x \text{ contains 01 as a substring}\}$
- 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 $(Q, \Sigma, \delta, q_o, F)$ where
  - Q is a finite set (of states)
  - Σ is a finite alphabet of symbols
  - $q_o \in Q$ is the start state
  - $F \subseteq Q$ is the set of final states
  - δ is a function from $Q \times \Sigma$ to Q (transition function)

Transition Function

- The transition function
  - δ is a function from $Q \times \Sigma$ to Q
  - $\delta(q, a) = q'$ where
    - $q, q' \in Q$
    - $a \in \Sigma$
  - δ 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.
  - $\hat{\delta}$ is a function from $Q \times \Sigma^*$ to Q
  - $\hat{\delta}(q, x) = q'$ where
    - $q, q' \in Q$
    - $x \in \Sigma^*$
  - $\hat{\delta}$ 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

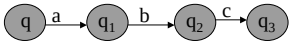
- Recursive definition of $\hat{\delta}$
  - Let $A = (Q, \Sigma, \delta, q_o, F)$ be an FA.
  - Define $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$
    1. For any $q \in Q$, $\hat{\delta}(q, \varepsilon) = q$
    2. For any $y \in \Sigma^*$, $a \in \Sigma$, and $q \in Q$
       $\hat{\delta}(q,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

Transition Function on Strings

- Example

$\hat{\delta}(q, \varepsilon) = q$

$\hat{\delta}(q, ya) = \delta(\hat{\delta}(q, y), a)$



$\hat{\delta}(q, \underline{abc}) = \delta(\hat{\delta}(q, \underline{ab}), c)$
$= \delta(\delta(\hat{\delta}(q, \underline{a}), b), c)$
$= \delta(\delta(\hat{\delta}(q, \underline{\varepsilon a}), b), c)$
$= \delta(\delta(\delta(\hat{\delta}(q, \varepsilon), a), b), c)$
$= \delta(\delta(\underline{\delta(q, a)}, b), c)$
$= \delta(\underline{\delta(q_1, b)}, c)$
$= \underline{\delta(q_2, c)}$
$= q_3$

Language accepted by a DFA

- Let $A = (Q, \Sigma, \delta, q_o, F)$ be a DFA.
- A string $x \in \Sigma^*$ is <u>accepted</u> by A if
  - $\hat{\delta}(q_0, x) \in F$
  - 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 <u>rejected</u> 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 Σ, L is accepted by A iff L = L(A).
  - For all $x \in L$, x is accepted by A.
  - For all $x \notin L$, x is rejected by A.

Reality Check

- What we've learned so far
  - An DFA can be expressed by $(Q, \Sigma, \delta, q_o, F)$
  - Transition function: $\delta: Q \times \Sigma \rightarrow Q$
  - Transition function on strings
    - $\hat{\delta}: Q \times \Sigma^* \rightarrow Q$
    - Defined recursively
  - DFA accepting a string
  - The language accepted by an DFA

Reality Check

- Questions?
- Let's take a break.