Announcements

• Homework 10 is due today
  – Turn it in now

• Quiz 10 is on Tuesday

• Exam 3 is next Thursday (No Final Exam)
  • Last Day of Class

• Allowed to bring in one 8.5 x 11in sheet of paper
  – Hand written notes of anything you would like

Non-Deterministic Finite-State Automata (NFA)

• Similar to DFA but many contain multiple state transitions for a particular input symbol

• A language is accepted if one of the possible states the NFA could be in accepts the language

• Any NFA can be converted into a DFA
Nondeterministic Finite Automaton (NFA)

Alphabet = \{a\}

Two choices
Alphabet = \{ a \}

Two choices

First Choice
First Choice

All input is consumed

Second Choice
Second Choice

No transition: the automaton hangs
An NFA accepts a string:
if there is a computation of the NFA
that accepts the string

i.e., all the input is consumed and the automaton
is in an accepting state
Example

\[ aa \] is accepted by the NFA:

because this computation accepts

Rejection example

\[ a \]
First Choice

\[ a \]

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \]

\[ q_0 \xrightarrow{a} q_3 \]

"reject"

First Choice

\[ a \]

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \]

\[ q_0 \xrightarrow{a} q_3 \]
An NFA rejects a string:
if there is no computation of the NFA that accepts the string.

For each computation:

- All the input is consumed and the automaton is in a non final state
- OR
- The input cannot be consumed
Example

\(a\) is rejected by the NFA:

Rejection example

\[ a \ a \ a \]
First Choice

No transition: the automaton hangs
First Choice

Input cannot be consumed

Second Choice
The automaton hangs
Second Choice

Input cannot be consumed

aaa is rejected by the NFA:

All possible computations lead to rejection
13.4 Language Recognition

- Previously discussed using finite-state automata as language recognizers

- What sets can be recognized by languages?
  - Regular Sets
    - Introduced by Stephen Kleene and his Kleene Theorem

- Regular Sets
  - Sets built up from the null set {}, the empty string λ, and singleton strings by taking concatenation, unions, and Kleene closures in arbitrary order
Regular Expressions

- To further define regular sets we need to define *regular expressions*

- The regular expressions over a set $S$ are defined recursively by
  - The symbol $\{\}$ is a regular expression
  - The symbol $\lambda$ is a regular expression (empty string)
  - The symbol $x$ is a regular expression whenever $x$ is an element of $S$
  - The symbols $(AB)$, $(A \cup B)$ and $A^*$ are regular expressions whenever $A$ and $B$ are regular expressions

- Sets represented by regular expressions are called *regular sets*

Regular Expression Examples

- What are the strings in the regular sets specified by the following:
  - $10^*$: A 1 followed by any number of 0s including no zeros
  - $(10)^*$: Any number of copies of 10
  - $0 \cup 01$: the string 0 or 01
  - $0 (0 \cup 1)^*$: Any string beginning with 0
  - $(0*1)^*$: Any string not ending in 0
Kleene’s Theorem

- A set is regular if and only if it is recognized by a finite-state automaton (FSA)

- Prove the only if part
  - Every regular set is recognized by a finite-state automaton

- To prove the only if part we need to show the following:
  - Show that \{ \} is recognized by an FSA (empty set)
  - Show that \{ \lambda \} is recognized by an FSA (set containing empty string)
  - Show that \{a\} is recognized by an FSA whenever a is a symbol in input language
  - Show that \{AB\} is recognized by an FSA whenever both A and B are
  - Show that \{A U B\} is recognized by an FSA whenever A and B are
  - Show that \{A*\} is recognized by an FSA whenever A is

Kleene’s Theorem

- Show that \{ \} is recognized by FSA

- Show that \{ \lambda \} is recognized by FSA

- Show that \{a\} is recognized by FSA
**Concatenation – Kleene Theorem**

- Show that \( AB \) is recognized by an FSA whenever \( A \) and \( B \) are

  Transition to final state in \( M_A \) produces a transition to \( s_B \)

  Transition from \( s_B \) in \( M_B \) produces a transition from \( s_{AB} = s_A \)

  Start state is \( s_{AB} = s_A \), which is final if \( s_A \) and \( s_B \) are final

**Union – Kleene Theorem**

- Show that \( A \cup B \) is recognized by an FSA whenever both \( A \) and \( B \) are

  Final states are the final states of \( M_A \) or \( M_B \)

  \( s_{AUB} \) is the new start state, which is final if \( s_A \) or \( s_B \) is final
### Kleene Closure – Kleene Theorem

- **Show that** $A^*$ **is recognized by an FSA whenever** $A$ **is**

Transitions from $s_A$ produce $A$ transitions from $s_A^*$ and all final states of $M_A$.

### The Turing Machine (TM)

- **Turing Machine** is a model of computing agent.
  - A model that captures the essential features of a computing agent.

- **A Turing machine consists of**
  - a tape that extends arbitrarily far in both directions
  - the tape is divided into cells, each cell can carry a symbol.
  - the symbols comes from a finite set of symbols called the **alphabet**
  - the **alphabet** consists of symbols for example: $b$ (blank), 0, 1
    
    $\begin{array}{cccccc}
    b & 0 & 1 & 0 & 1 & 0 \\
    1 & & & & & \\
    \end{array}$

  - the tape serve as memory
  - has a finite number of $k$ states, $1, \ldots, k$
Turning Machine Behavior

- Turing Machine actions depend on two inputs:
  - The current state of the machine
  - content of cell currently being read (input)
- A Turing machine can do only one operation at a time. Each time an operation is done, three actions may take place:
  - write a symbol to cell
  - go into a new state
  - move one cell left or right

The Turing Machine (TM)

Example: Assume a Turing machine (TM) instruction:

<table>
<thead>
<tr>
<th>Current State</th>
<th>Symbol Read</th>
<th>Action</th>
<th>New State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>W</td>
<td>2</td>
</tr>
</tbody>
</table>

Written as: \((1, 0, 1, R, 2)\)
**The Turing Machine (TM)**

**Example:** Design a Turing Machine which will invert the string of binary digits
- if the input string is 10110 then the output string should be 01001
- let us draw a state diagram

---

**Unary Representation for TM**

Unary representation is used in order to do any arithmetic operations on a TM.
- Unary representation looks as follows
  0 → 1
  1 → 11
  2 → 111
  3 → 1111
  4 → 11111

**Example:** Design a Turing Machine to add 1 to any number
- start in state 1
- if the state is 1 and current input is 1, write 1 and move right and stay in state 1
- if the current state is 1 and current input is b, write 1 and move to state 2 and move right and HALT
- TM instructions:
  - (1,1,1,R,1)
  - (1,b,1,R,2)

state 2 does not exist, so halt
Let the initial configuration be: ... b 1 1 1 1 b ...

s1... b 1 1 1 1 b ...

s1... b 1 1 1 1 b ...

s1... b 1 1 1 1 b ...

s2... b 1 1 1 1 b ...

1/1

b/1

Halt

2 in unary representation

3 in unary representation
Example: Adding of two non-zero numbers (unary representation)

Initial Setup: . . b 1 1 1 b 1 1 b . .
Answer should be: . . b b b 1 1 1 1 b . .

(1,1,b,R,2): Erase leftmost 1 and move right
(2,1,b,R,3): Erase second 1 and move right
(3,1,1,R,3): pass over any 1’s until a blank is found
(3,b,1,R,4): write 1 over the blank
(4,1,1,R,4): pass over remaining 1’s
(4,b,1,R,5): halt

Review for Exam 3

• Covers Chapters 7,9,10 and 13

• Topics Include
  – Probability and Counting
  – Closures and Relations
  – Graphs
    • Adjacency Matrix
  – Euler and Hamilton Circuits and Paths
  – DFAs and FSMs
  – Kleene Closure
  – Turing Machines

• Understand the following notation
  – $K_n$ (Complete)
  – $Q_n$ (Cube)
  – $W_n$ (Wheel)
  – $C_n$ (Cycle)
  – $K_{n,n}$ (Bipartite)