CMSC330

Finite Automata

Last lecture

• Languages
  – Sets of strings
  – Operations on languages
• Regular expressions
  – Constants
  – Operators
  – Precedence

This lecture

• Finite automata
  – States
  – Transitions
  – Examples
• Types
  – Deterministic (DFA)
  – Non-deterministic (NFA)

Implementing Regular Expressions

• We can implement a regular expression by turning it into a finite automaton
  – A “machine” for recognizing a regular language

Finite Automata

- Start state
  - State with incoming transition from no other state
  - Can have only 1 start state
- Final state
  - State with double circle
  - Can have 0 or more final states

Finite Automata: States

- Start state
  - State with incoming transition from no other state
  - Can have only 1 start state
- Final state
  - State with double circle
  - Can have 0 or more final states

Finite Automata

- Machine starts in start or initial state
- Repeat until the end of the string is reached
  - Scan the next symbol s of the string
  - Take transition edge labeled with s
- String is accepted if automaton is in final state when end of string reached
Finite Automaton: Example 1

0 0 1 0 1 1
accepted

Finite Automaton: Example 2

0 0 1 0 1 0
not accepted

What Language is This?

• All strings over \{0, 1\} that end in 1
• What is a regular expression for this language?
  \((0|1)^*1\)

Finite Automaton: Example 3

(a,b,c notation shorthand for three self loops)

Last Lecture

• Regular Expression Theory
• Introduction to finite state machines

This lecture

• Finite state machines continued
Finite Automaton: Example 3

(a,b,c notation shorthand for three self loops)

<table>
<thead>
<tr>
<th>string</th>
<th>state at end</th>
<th>accept</th>
</tr>
</thead>
<tbody>
<tr>
<td>aabcc</td>
<td>S2</td>
<td>Y</td>
</tr>
<tr>
<td>acc</td>
<td>S2</td>
<td>Y</td>
</tr>
<tr>
<td>bbc</td>
<td>S2</td>
<td>Y</td>
</tr>
<tr>
<td>aabbb</td>
<td>S1</td>
<td>Y</td>
</tr>
<tr>
<td>aa</td>
<td>S0</td>
<td>Y</td>
</tr>
<tr>
<td>ε</td>
<td>S0</td>
<td>Y</td>
</tr>
<tr>
<td>acba</td>
<td>S3</td>
<td>N</td>
</tr>
</tbody>
</table>

What language does this DFA accept?

- What language does this DFA accept?
  - a*b*c*

S3 is a dead state – a nonfinal state with no transition to another state

Dead State: Shorthand Notation

- If a transition is omitted, assume it goes to a dead state that is not shown.
- Language?
  - Strings over \([0,1,2,3]\) with alternating even and odd digits, beginning with odd digit

Practice

Give the English descriptions and the DFA or regular expression of the following languages:

- All strings with length a multiple of 5
- All alternating binary strings containing the substring "11"

Finite Automaton: Example 4

- Description for each state
  - S0 = ‘Haven't seen anything yet’ OR “seen zero or more b’s” OR “Last symbol seen was a b”
  - S1 = “Last symbol seen was an a”
  - S2 = “Last two symbols seen were ab”
  - S3 = “Last three symbols seen were abb”
Practice

- Give the regular expressions and finite automata for the following languages
  - You and your neighbors’ names
  - All binary strings containing an even length substring of all 1’s
  - All binary strings containing exactly two 1’s
  - All binary strings that start and end with the same number

Types of Finite Automata

- Deterministic Finite Automata (DFA)
  - Exactly one sequence of steps for each string
  - All examples so far

- Nondeterministic Finite Automata (NFA)
  - May have many sequences of steps for each string
  - More compact

Math Alert!

Formal Definition

- A deterministic finite automaton (DFA) is a 5-tuple \((\Sigma, Q, q_0, F, \delta)\) where
  - \(\Sigma\) is an alphabet
  - the strings recognized by the DFA are over this set
  - \(Q\) is a nonempty set of states
  - \(q_0 \in Q\) is the start state
  - \(F \subseteq Q\) is the set of final states
  - How many can there be?
  - \(\delta: Q \times \Sigma \times Q\) specifies the DFA’s transitions

- A DFA accepts \(s\) if it stops at a final state on \(s\) (after consuming all chars in \(s\))

Formal Definition: Example

- 5-tuple \((\Sigma, Q, q_0, F, \delta)\)
  - \(\Sigma = \{0, 1\}\)
  - \(Q = \{S_0, S_1\}\)
  - \(q_0 = S_0\)
  - \(F = \{S_1\}\)

<table>
<thead>
<tr>
<th>(\delta)</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_0</td>
<td>S_0</td>
<td>S_1</td>
</tr>
<tr>
<td>S_1</td>
<td>S_0</td>
<td>S_1</td>
</tr>
</tbody>
</table>

DFA Requirements

- Can not have more than one transition leaving a state on the same symbol
  - I.e., transition function must be a valid function

- Can not have transitions with empty labels
  - Transitions must be labeled by alphabet symbols

- NFAs do not have these requirements!
  - DFA is a special case of NFA
Nondeterministic Finite Automata (NFA)

• An NFA is a 5-tuple \((\Sigma, Q, q_0, F, \delta)\) where
  – \(\Sigma\) is an alphabet
  – \(Q\) is a nonempty set of states
  – \(q_0 \in Q\) is the start state
  – \(F \subseteq Q\) is the set of final states
  – \(\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q\) specifies the NFA’s transitions
    • Transitions on \(\varepsilon\) are allowed – can optionally take these transitions without consuming any input
    • Can have more than one transition for a given state and symbol

• An NFA accepts \(s\) if there is at least one path from its start to final state on \(s\)

Another example DFA

• Language?
  – \((ab|aba)^*\)

NFA for \((a|b)^*abb\)

• \(ba\)
  – Has paths to either \(S0\) or \(S1\)
  – Neither is final, so rejected

• \(babaabb\)
  – Has paths to different states
  – One leads to \(S3\), so accepted

NFA for \((ab|aba)^*\)

• \(aba\)
  – Has paths to states \(S0, S1\)

• \(ababa\)
  – Has paths to \(S0, S1\)
  – Need to use \(\varepsilon\)-transition

Relating REs to DFAs and NFAs

• Regular expressions, NFAs, and DFAs accept the same languages!

Reducing Regular Expressions to NFAs

• Goal: Given regular expression \(e\), construct NFA: \(<e> = (\Sigma, Q, q_0, F, \delta)\)
  – Remember regular expressions are defined recursively from primitive RE languages
  – Invariant: \(|F| = 1\) in our NFAs
    • Recall \(F\) = set of final states

• Base case: \(a\)
  \(<a> = ((a), \{S0, S1\}, S0, \{S1\}, \{(S0, a, S1)\})\)
Reduction (cont.)

- Base case: $\epsilon$
  
  $<\epsilon> = (\epsilon, [\emptyset], \emptyset, [\emptyset], \emptyset)$

- Base case: $\emptyset$

  $<\emptyset> = (\emptyset, [\emptyset, \{5\}], \emptyset, [\emptyset, \{5\}], \emptyset)$

Reduction: Concatenation

- Induction: $AB$

  $<A> \rightarrow_t A \rightarrow_t BA \rightarrow_t AB$

  $<A> = (I_{A}, Q_{A}, Q_{A}, f_{\delta A}, \delta_{A})$

  $<B> = (I_{B}, Q_{B}, Q_{B}, f_{\delta B}, \delta_{B})$

  $<AB> = (I_{A}, I_{B}, Q_{A} U Q_{B}, Q_{A} U Q_{B}, f_{\delta A U \delta B}, (f_{\delta A \delta B}(I_{A}, \epsilon, Q_{A}), (I_{B}, \epsilon, Q_{B}), (f_{\delta A \delta B}(I_{A}, \epsilon, Q_{A}), (f_{\delta A \delta B}(I_{A}, \emptyset, Q_{A})))$}

Reduction: Union (cont.)

- Induction: $(A|B)$

  $<A> \rightarrow_t A \rightarrow_t A|B \rightarrow_t AB$

  $<A> = (I_{A}, Q_{A}, Q_{A}, f_{\delta A}, \delta_{A})$

  $<B> = (I_{B}, Q_{B}, Q_{B}, f_{\delta B}, \delta_{B})$

  $<(A|B)> = (I_{A}, I_{B}, Q_{A} U Q_{B}, Q_{A} U Q_{B}, f_{\delta A U \delta B}, (f_{\delta A \delta B}(I_{A}, \epsilon, Q_{A}), (I_{B}, \epsilon, Q_{B}), (f_{\delta A \delta B}(I_{A}, \epsilon, Q_{A}), (f_{\delta A \delta B}(I_{A}, \emptyset, Q_{A})))$}

Reduction: Closure

- Induction: $A^*$
Reduction: Closure (cont.)

- Induction: \( A^* \)

\[
\begin{align*}
- \langle A \rangle &= (\Sigma_A, Q_A, q_0, \delta_A) \\
- \langle A^* \rangle &= (\Sigma_A, Q_A U \{S0, S1\}, S0, \{S1\}, \\
&\quad \delta_A U (f_A, e, S1), (S0, e, q_0), (S0, e, S1), \\
&\quad (S1, e, S0))
\end{align*}
\]

Reduction Complexity

- Given a regular expression \( A \) of size \( n \)...
  
  Size = # of symbols + # of operations

- How many states does \( \langle A \rangle \) have?
  - 2 added for each |, 2 added for each *
  - \( O(n) \)
  - That’s pretty good!

Practice

- Draw NFAs for the following regular expressions and languages
  - \((0|1)^*110^*\)
  - \(101^*111\)
  - all binary strings ending in 1 (odd numbers)
  - all alphabetic strings which come after “hello” in alphabetic order
  - \((ab^*c|d^*a|ab)d\)

Summary

- Finite automata
  - Deterministic (DFA)
  - Non-deterministic (NFA)

- Questions
  - How are DFAs and NFAs different?
  - When does an NFA accept a string?
  - How to convert regular expression to an NFA?