CMSC 330: Organization of Programming Languages

Theory of Regular Expressions
Finite Automata
Previous Course Review

• \( \{s \mid s \text{ defined}\} \) means the set of string \( s \) such that \( s \) is chosen or defined as given
• \( s \in A \) means \( s \) is an element of the set \( A \)
• De Morgan’s Laws:
  \[
  (A \cap B)^C = A^C \cup B^C \\
  (A \cup B)^C = A^C \cap B^C 
  \]
• There exists (\( \exists \)) and for all (\( \forall \)) symbols
Regular Expression Review

• Terms
  – Alphabet
  – String
  – Language
  – Regular expression (“regex”)

• Operations
  – Concatentation
  – Union
  – Kleene closure

• Ruby vs. theoretical
Implementing Regular Expressions

• We can implement regular expressions by turning them into a finite automaton
  – A “machine” for recognizing a regular language
Example

- 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 $s$
- The string is *accepted* if the automaton is in a *final* or *accepting* state when the end of the string is reached
Example

Accepted
Example

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? \((0|1)^*1\)
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?
  – \(\delta : Q \times \Sigma \rightarrow Q\) specifies the DFA's transitions
    • What's this definition saying that \(\delta\) is?
More on DFAs

• A finite state automata can have more than one final state:

• A string is accepted as long as there is at least one path to a final state
Our Example, Formally

- \( \Sigma = \{0, 1\} \)
- \( Q = \{S0, S1\} \)
- \( q_0 = S0 \)
- \( F = \{S1\} \)

\[
\begin{array}{c|cc}
\delta & 0 & 1 \\
\hline
S0 & S0 & S1 \\
S1 & S0 & S1 \\
\end{array}
\]
Another Example

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

<table>
<thead>
<tr>
<th>string</th>
<th>state at end</th>
<th>accepts</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>
Another Example (cont’d)

What language does this DFA accept? $a^*b^*c^*$

S3 is a dead state – a nonfinal state with no transition to another 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
What Lang. Does This DFA Accept?

\[a^*b^*c^*\] again, so DFAs are not unique
Practice

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

• \(((0|1)(0|1)(0|1))\)^*  
  all strings with length a multiple of 3

• \((01)^*|(10)^*|(01)^*0|(10)^*1\)  
  all alternating binary strings

• \((0|1)(0|1)(0|1))\)  
  all binary strings containing the substring “11”
Practice

Give the DFAs for the following languages:

• All valid strings including only “A”, “C”, “G”, “T” and appearing in multiples of 3
• All binary strings containing an even, non-zero 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
Review

• Basic parts of a regular expression?
  \textit{concatenation, |, *, \epsilon, \emptyset, \{a\}}

• What does a DFA do?

• Basic parts of a DFA?
  \textit{alphabet, set of states, start state, final states, transition function (\Sigma, Q, q_0, F, \delta)}
Example DFA

- $S_0$ = “Haven’t seen anything yet” OR “seen zero or more b’s” OR “Last symbol seen was a b”
- $S_1$ = “Last symbol seen was an $a$”
- $S_2$ = “Last two symbols seen were $ab$”
- $S_3$ = “Last three symbols seen were $abb$”

- Language?
- $(a|b)^*abb$
Notes about the DFA definition

• Can not have more than one transition leaving a state on the same symbol
  – the transition function must be a valid function

• Can not have transitions with no or empty labels
  – the transitions must be labeled by alphabet symbols
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
    - There may be 0, 1, or many
  - \(\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q\) 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 \(s\)
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
Another example DFA

- Language?
- \((ab|aba)^*\)
NFA for (ab|aba)*

• aba
  – Has paths to states S0, S1

• ababa
  – Has paths to S0, S1
  – Need to use $\varepsilon$-transition
Relating R.E.'s to DFAs and NFAs

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

(we’ll discuss this next)
Reducing Regular Expressions to NFAs

• Goal: Given regular expression e, construct NFA: \(<e> = (\Sigma, Q, q_0, F, \delta)\)
  – Remember r.e. defined recursively from primitive r.e. 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’d)

- Base case: $\varepsilon$

  \[<\varepsilon> = (\varepsilon, \{S0\}, S0, \{S0\}, \emptyset)\]

- Base case: $\emptyset$

  \[<\emptyset> = (\emptyset, \{S0, S1\}, S0, \{S1\}, \emptyset)\]
Reduction (cont’d)

- Induction: AB
Reduction (cont’d)

• Induction: \( AB \)

\[
\begin{align*}
\langle A \rangle &= (\Sigma_A, Q_A, q_A, \{f_A\}, \delta_A) \\
\langle B \rangle &= (\Sigma_B, Q_B, q_B, \{f_B\}, \delta_B) \\
\langle AB \rangle &= (\Sigma_A \cup \Sigma_B, Q_A \cup Q_B, q_A, \{f_B\}, \delta_A \cup \delta_B \cup \{(f_A, \varepsilon, q_B)\})
\end{align*}
\]
Practice

• Draw the NFA for these regular expressions using the reduction method:
  – ab
  – hello

• Write the formal (5-tuple) NFA for the same regular expressions
Reduction (cont’d)

• Induction: (A|B)
Reduction (cont’d)

• Induction: \((A|B)\)

– \(\langle A \rangle = (\Sigma_A, Q_A, q_A, \{f_A\}, \delta_A)\)
– \(\langle B \rangle = (\Sigma_B, Q_B, q_B, \{f_B\}, \delta_B)\)
– \(\langle (A|B) \rangle = (\Sigma_A \cup \Sigma_B, Q_A \cup Q_B \cup \{S0,S1\}, S0, \{S1\}, \delta_A \cup \delta_B \cup \{(S0,\epsilon,q_A), (S0,\epsilon,q_B), (f_A,\epsilon,S1), (f_B,\epsilon,S1)\})\)
Practice

• Draw the NFA for these regular expressions using exactly the reduction method:
  – ab|bc
  – hello|hi

• Write the formal NFA for the same regular expressions
Reduction (cont’d)

• Induction: $A^*$
Reduction (cont’d)

• Induction: $A^*$

- $<A> = (\Sigma_A, Q_A, q_A, \{f_A\}, \delta_A)$
- $<A^*> = (\Sigma_A, Q_A \cup \{S0,S1\}, S0, \{S1\},$
  $\delta_A \cup \{(f_A, \epsilon, S1), (S0, \epsilon, q_A), (S0, \epsilon, S1), (S1, \epsilon, S0)\})$
Practice

• Draw the NFA for these regular expressions using exactly the reduction method:
  – (ab|bc*)*
  – hello|(hi)*

• Write the formal NFA for the same regular expressions
Reduction Complexity

• Given a regular expression $A$ of size $n$...
  Size = # of symbols + # of operations

• How many states does $<A>$ 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
• $(ab^*c|d^*a|ab)d$
Handling $\varepsilon$-transitions

What if we want to remove all those unneeded $\varepsilon$-transitions?

First, some definitions:

• We say: $p \xrightarrow{\varepsilon} q$
  
  – if it is possible to transition from state $p$ to state $q$ taking only $\varepsilon$-transitions
  
  – if $\exists p, p_1, p_2, \ldots p_n, q \in Q (p \neq q)$ such that
    $\{p, \varepsilon, p_1\} \in \delta, \{p_1, \varepsilon, p_2\} \in \delta, \ldots, \{p_n, \varepsilon, q\} \in \delta$
\( \varepsilon \)-closure

- For any state \( p \), the \( \varepsilon \)-closure of \( p \) is defined as the set of states \( q \) such that \( p \xrightarrow{\varepsilon} q \)

- \( \{ q \mid p \xrightarrow{\varepsilon} q \} \)
Example

• What’s the $\varepsilon$-closure of S2 in this NFA?

• $\{S2, S0\}$
Example

- Find the $\epsilon$-closure for each of the states in this NFA:
Example

• Make the NFA for the regular expression
  – (0|1*)111(0*|1)

• Find the epsilon closure for each of the states of your NFA