CMSC 330: Organization of Programming Languages

Theory of Regular Expressions
DFA and NFAs

Reminders
- Project 1 due Sep. 24
- Homework 1 posted
- Exam 1 on Sep. 25
- Exam topics list posted
- Practice homework (and solutions) posted

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 and for all symbols
- Etc...

Review
- Basic parts of a regular expression?
  - Concatenation, \( \cdot \), \( * \), \( \epsilon \), \( \emptyset \), \( (a) \)
- What does a DFA do?
- Basic parts of a DFA?
  - Alphabet, set of states, start state, final states, transition function \( (X, Q, q_0, F, \delta) \)

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?
  - \( (ab)^*abb \)

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
  - \(\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q\) specifies the NFA's transitions
    - Transitions on \(\epsilon\) 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

```
DFA

<table>
<thead>
<tr>
<th>State</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

- Language?
- \((ab|aba)*\)

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

```
NFA

<table>
<thead>
<tr>
<th>State</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a</td>
</tr>
</tbody>
</table>
```

- \(aba\) – Has paths to states \(S0, S1\)
- \(ababa\) – Has paths to \(S0, S1\)
  - Need to use \(\epsilon\)-transition

Relating R.E.'s 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 r.e. defined recursively from primitive r.e. languages
  - Invariant: |\(F| = 1\) in our NFAs
    - Recall \(F\) = set of final states
  - Base case: \(a\)

```
NFA

<table>
<thead>
<tr>
<th>State</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td></td>
</tr>
</tbody>
</table>
```

\(<e> = ((a), (S0, S1), S0, (S1), ((S0, a, S1)))\)
Reduction (cont’d)

- Base case: $\varepsilon$

$$\langle \varepsilon \rangle = (\varepsilon, \{S0\}, \{S0\}, \emptyset)$$

- Base case: $\emptyset$

$$\langle \emptyset \rangle = (\emptyset, \{S0, S1\}, \emptyset, \{\emptyset\}, \emptyset)$$

Induction: $AB$

$$\langle A \rangle = (S0, QA, qA, \{fA\}, S0)$$

$$\langle B \rangle = (S0, QB, qB, \{fB\}, S0)$$

$$\langle AB \rangle = (S0, QA \cup QB, qA, \{fB\}, S0, \{fA, \varepsilon, qB\})$$

Practice

- Draw the NFA for these regular expressions using exactly the reduction method:
  - $ab$
  - $hello$

- Write the formal (5-tuple) NFA for the same regular expressions

Induction: $(A|B)$

$$\langle A \rangle = (S0, QA, qA, \{fA\}, S0)$$

$$\langle B \rangle = (S0, QB, qB, \{fB\}, S0)$$

$$\langle (A|B) \rangle = (S0, QA \cup QB, qA, \{fB\}, S0, \{fA, \varepsilon, S1\}, \{fB, \varepsilon, S1\})$$
Practice

- Draw the NFA for these regular expressions using exactly the reduction method:
  - \( a b \mid bc \)
  - \( \text{hello} \mid \text{hi} \)

- Write the formal NFA for the same regular expressions

Reduction (cont’d)

- Induction: \( A^* \)

\[
\begin{align*}
\text{Reduction Complexity} & \quad \text{Practice} \\
\text{Reduction (cont’d)} & \quad \text{Practice}
\end{align*}
\]

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 ε-transitions**

What if we want to remove all those unneeded ε-transitions?

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

**ε-closure**

- For any state \( p \), the ε-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 ε-closure of \( S_2 \) in this NFA?


  ![NFA Diagram](image)

  - \( \{S_2, S_0\} \)

**Example**

- Find the ε-closure for each of the states in this NFA:


  ![NFA Diagram](image)

- Make the NFA for the regular expression
  - \( (0|1^*)111(0^*1) \)

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