4b

Lexical analysis
Finite Automata
Finite Automata (FA)

• FA also called Finite State Machine (FSM)
  – Abstract model of a computing entity.
  – Decides whether to accept or reject a string.
  – Every regular expression can be represented as a FA and vice versa

• Two types of FAs:
  – Non-deterministic (NFA): Has more than one alternative action for the same input symbol.
  – Deterministic (DFA): Has at most one action for a given input symbol.

• Example: how do we write a program to recognize the Java keyword “int”?

```
q0 -> i : q1 -> n : q2 -> t : q3
```

CMSC 331, Some material © 1998 by Addison Wesley Longman, Inc.
RE and Finite State Automaton (FA)

- Regular expressions are a declarative way to describe the tokens
  - Describes *what* is a token, but not *how* to recognize the token
- FAs are used to describe *how* the token is recognized
  - FAs are easy to simulate in programs
- There is a 1-1 correspondence between FAs & regular expressions
  - A scanner generator (e.g., lex) bridges the gap between regular expressions and FAs.
Main components of scanner generation (e.g., Lex)

- Convert a regular expression to a non-deterministic finite automaton (NFA)
- Convert the NFA to a deterministic finite automaton (DFA)
- Improve the DFA to minimize the number of states
- Generate a program in C or some other language to "simulate" the DFA
Non-deterministic Finite Automata (FA)

• NFA (Non-deterministic Finite Automaton) is a 5-tuple 
  \((S, \Sigma, \delta, S_0, F)\):
  
  – \(S\): a set of states;
  
  – \(\Sigma\): the symbols of the input alphabet;
  
  – \(\delta\): a set of transition functions;
    
    » move(state, symbol) \(\rightarrow\) a set of states
  
  – \(S_0\): \(s_0 \in S\), the start state;
  
  – \(F\): \(F \subseteq S\), a set of final or accepting states.

• Non-deterministic -- a state and symbol pair can be mapped to a set of states.

• Finite—the number of states is finite.
Transition Diagram

- FA can be represented using transition diagram.
- Corresponding to FA definition, a transition diagram has:
  - **States** represented by circles;
  - An **Alphabet** (Σ) represented by labels on edges;
  - **Transitions** represented by labeled directed edges between states. The label is the input symbol;
  - One **Start State** shown as having an arrow head;
  - One or more **Final State(s)** represented by double circles.

- Example transition diagram to recognize (a|b)*abb
Simple examples of FA

- $a$
  - Transition: Start $\rightarrow$ 0 $\rightarrow$ 1

- $a^*$
  - Transition: Start $\rightarrow$ 0 $\rightarrow$ 0

- $a^+$
  - Transition: Start $\rightarrow$ 0 $\rightarrow$ 1 $\rightarrow$ 0

- $(a|b)^*$
  - Transition: Start $\rightarrow$ 0 $\rightarrow$ 0 $\rightarrow$ (a, b)
Procedures of defining a DFA/NFA

• Defining input alphabet and initial state
• Draw the transition diagram
• Check
  – Do all states have out-going arcs labeled with all the input symbols (DFA)
  – Any missing final states?
  – Any duplicate states?
  – Can all strings in the language can be accepted?
  – Are any strings not in the language accepted?
• Naming all the states
• Defining \((S, \Sigma, \delta, q_0, F)\)
Example of constructing a FA

• Construct a DFA that accepts a language L over the alphabet \{0, 1\} such that L is the set of all strings with any number of “0”s followed by any number of “1”s.

• Regular expression: 0*1*

• \( \Sigma = \{0, 1\} \)

• Draw initial state of the transition diagram

```
Start ————>
```
Example of constructing a FA

- Draft the transition diagram

- Is “111” accepted?
- The leftmost state has missed an arc with input “1”
Example of constructing a FA

- Is “00” accepted?
- The leftmost two states are also final states
  - First state from the left: $\varepsilon$ is also accepted
  - Second state from the left: strings with “0”s only are also accepted
Example of constructing a FA

• The leftmost two states are duplicate
  – their arcs point to the same states with the same symbols

• Check that they are correct
  – All strings in the language can be accepted
    » ε, the empty string, is accepted
    » strings with “0”s / “1”s only are accepted
  – No strings not in language are accepted

• Naming all the states

\[
\begin{array}{c}
\text{Start} \\
\quad \rightarrow 1 \\
\quad \rightarrow 0 \\
\end{array}
\]
How does a FA work

• NFA definition for \((a|b)^*abb\)
  – \(S = \{ q_0, q_1, q_2, q_3 \} \)
  – \(\Sigma = \{ a, b \} \)
  – Transitions: move(q0,a)={q0, q1}, move(q0,b)={q0}, ....
  – \(s_0 = q_0\)
  – \(F = \{ q_3 \} \)

• Transition diagram representation
  – Non-determinism:
    » exiting from one state there are multiple edges labeled with same symbol, or
    » There are epsilon edges.
  – How does FA work? Input: ababb

\[
\begin{align*}
move(0, a) &= 1 \\
move(1, b) &= 2 \\
move(2, a) &= ? \text{ (undefined)} \\
move(0, a) &= 0 \\
move(0, b) &= 0 \\
move(0, a) &= 1 \\
move(1, b) &= 2 \\
move(2, b) &= 3 \\
\text{REJECT !} \\
\text{ACCEPT !}
\end{align*}
\]
FA for \((a|b)^*abb\)

- What does it mean that a string is accepted by a FA?
  
  An FA accepts an input string \(x\) iff there is a path from the start state to a final state, such that the edge labels along this path spell out \(x\);

- A path for “aabb”: \(Q0\rightarrow^a q0\rightarrow^a q1\rightarrow^b q2\rightarrow^b q3\)

- Is “aab” acceptable?
  
  \(Q0\rightarrow^a q0\rightarrow^a q1\rightarrow^b q2\)
  
  \(Q0\rightarrow^a q0\rightarrow^a q0\rightarrow^b q0\)

  »Final state must be reached;
  
  »In general, there could be several paths.

- Is “aabbb” acceptable?
  
  \(Q0\rightarrow^a q0\rightarrow^a q1\rightarrow^b q2\rightarrow^b q3\)

  »Labels on the path must spell out the entire string.
Transition table

• A transition table is a good way to implement a FSA
  – One row for each state, S
  – One column for each symbol, A
  – Entry in cell (S, A) gives the state or set of states can be reached from state S on input A.

• A Nondeterministic Finite Automaton (NFA) has at least one cell with more than one state.

• A Deterministic Finite Automaton (DFA) has a single state in every cell

\[(a|b)^*abb\]

<table>
<thead>
<tr>
<th>STATES</th>
<th>INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>&gt;Q0</td>
<td>{q0, q1}</td>
</tr>
<tr>
<td>Q1</td>
<td></td>
</tr>
<tr>
<td>Q2</td>
<td></td>
</tr>
<tr>
<td>*Q3</td>
<td></td>
</tr>
</tbody>
</table>
DFA (Deterministic Finite Automaton)

• A special case of NFA where the transition function maps the pair (state, symbol) to one state.
  – When represented by transition diagram, for each state S and symbol a, there is at most one edge labeled a leaving S;
  – When represented transition table, each entry in the table is a single state.
  – There are no ε-transition

• Example: DFA for \((a|b)^*abb\)

<table>
<thead>
<tr>
<th>STATES</th>
<th>INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
</tr>
<tr>
<td>q0</td>
<td>q1</td>
</tr>
<tr>
<td>q1</td>
<td>q1</td>
</tr>
<tr>
<td>q2</td>
<td>q1</td>
</tr>
<tr>
<td>q3</td>
<td>q1</td>
</tr>
</tbody>
</table>

• Recall the NFA:
DFA to program

• NFA is more concise, but not as easy to implement;
• In DFA, since transition tables don’t have any alternative options, DFAs are easily simulated via an algorithm.
• Every NFA can be converted to an equivalent DFA
  – What does equivalent mean?
• There are general algorithms that can take a DFA and produce a “minimal DFA.
  – Minimal in what sense?
• There are programs that take a regular expression and produce a program based on a minimal DFA to recognize strings defined by the RE.
• You can find out more in 451 (automata theory) and/or 431 (Compiler design)