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”?

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 a 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.

Transition Diagram

• FA can be represented using transition diagram.
• Corresponding to FA definition, a transition diagram has:
  – States represented by circles;
  – An Alphabet ($) 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.

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 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

\[
\text{Start} \quad \circ \quad 0 \quad 1 \quad 1
\]

Example of constructing a FA

• Is “00” accepted?
• The leftmost two states are also final states
  – First state from the left: \( g \) is also accepted
  – Second state from the left:
    strings with “0”s only are also accepted

\[
\text{Start} \quad \circ \quad 0 \quad 1 \quad 1
\]

Example of constructing a FA

• Is “111” accepted?
• The leftmost state has missed an arc with input “1”

\[
\text{Start} \quad 0 \quad 1 \quad 1
\]

Example of constructing a FA

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

\[
\text{Start} \quad 0 \quad 1 \quad 1
\]

How does a FA work

• NFA definition for \((a|b)*abb\)
  – \( S = \{ q_0, q_1, q_2, q_3 \} \)
  – \( \Sigma = \{a, b\} \)
  – Transitions: \( \text{move}(q_0, a) = \{q_0, q_1\}, \text{move}(q_1, b) = \{q_2\}, \ldots \)
  – \( q_2 = q_3 \)
  – \( F = \{q_3\} \)
• 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: “aab”

\[
\text{Start} \quad q_0 \quad q_1 \quad q_3
\]

FA for \((a|b)*abb\)

• What does it mean that a string is accepted by a FA?
  – An FA accepts an input string \( x \) if there is a path from start to a final state, such that the edge labels along this path spell out \( x \);
  – A path for “aabbb”:
    \( Q_0 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \)
  – Is “aab” acceptable?
    \( Q_0 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \)
  – Is “aabbb” acceptable?
    \( Q_0 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \)
  – 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 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>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>q0</td>
<td>q0</td>
<td>q1</td>
</tr>
<tr>
<td>q1</td>
<td>q3</td>
<td>q3</td>
</tr>
<tr>
<td>q2</td>
<td>q3</td>
<td>q3</td>
</tr>
<tr>
<td>q3</td>
<td>q3</td>
<td>q3</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 by transition table, each entry in the table is a single state.
  - There are no ε-transitions
- Example: DFA for (a|b)*abb

<table>
<thead>
<tr>
<th>STATES</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>q0</td>
<td>q0</td>
<td>q0</td>
</tr>
<tr>
<td>q1</td>
<td>q1</td>
<td>q1</td>
</tr>
<tr>
<td>q2</td>
<td>q2</td>
<td>q2</td>
</tr>
<tr>
<td>q3</td>
<td>q3</td>
<td>q3</td>
</tr>
<tr>
<td>q4</td>
<td>q4</td>
<td>q4</td>
</tr>
</tbody>
</table>

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)