CMPT 379
Compilers

Anoop Sarkar

http://www.cs.sfu.ca/~anoop
NFA to DFA Complexity Analysis
NFA to DFA

• Subset construction converts NFA to DFA
• Complexity:
  – For FSAs, we measure complexity in terms of initial cost (creating the automaton) and per string cost
  – Let $r$ be the length of the regexp and $n$ be the length of the input string
  – NFA, Initial cost: $O(r)$; Per string: $O(rn)$
  – DFA, Initial cost: $O(r^2s)$; Per string: $O(n)$
  – DFA, common case, $s = r$, but worst case $s = 2^r$
NFA to DFA

• A regexp of size $r$ can become a $2^r$ state DFA, an exponential increase in complexity
  – Try the subset construction on NFA built for the regexp $A^*aA^{n-1}$ where $A$ is the regexp $(a|b)$

• Note that the NFA for regexp of size $r$ will have $r$ states

• Minimization can reduce the number of states

• But minimization requires determinization
NFA to DFA
NFA to DFA
NFA to DFA

\[ 2^5 = 32 \text{ states} \]
## NFA vs. DFA in the wild

<table>
<thead>
<tr>
<th>Engine Type</th>
<th>Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFA</td>
<td><em>awk</em> (most versions), <em>egrep</em> (most versions), <em>flex</em>, <em>lex</em>, MySQL, Procmail</td>
</tr>
<tr>
<td>Traditional NFA</td>
<td>GNU <em>Emacs</em>, Java, <em>grep</em> (most versions), <em>less</em>, <em>more</em>, .NET languages, PCRE library, Perl, PHP (pcre routines), Python, Ruby, <em>sed</em> (most versions), <em>vi</em></td>
</tr>
<tr>
<td>POSIX NFA</td>
<td><em>mawk</em>, MKS utilities, GNU <em>Emacs</em> (when requested)</td>
</tr>
<tr>
<td>Hybrid NFA/DFA</td>
<td>GNU <em>awk</em>, GNU <em>grep/egrep</em>, Tcl</td>
</tr>
</tbody>
</table>
Extensions to Regular Expressions

• Most modern regexp implementations provide extensions:
  – matching groups; \1 refers to the string matched by the first grouping (\), \2 to the second match, etc.,
    • e.g. ([a-z]+)\1 which matches abab where \1=ab
  – match and replace operations,
    • e.g. s/([a-z]+)/\1\1/g which changes ab into abab where \1=ab

• These extensions are no longer “regular”. In fact, extended regexp matching is NP-hard
  – Extended regular expressions (including POSIX and Perl) are called REGEX to distinguish from regexp (which are regular)

• In order to capture these difficult cases, the algorithms used even for simple regexp matching run in time exponential in the length of the input