CMPT 379
Compilers

Anoop Sarkar

http://www.cs.sfu.ca/~anoop
Matching Patterns using Non-deterministic Automata
(conversion from NFA to DFA)
Simulating NFAs

- Simulation == Given a NFA and input string, does the string match the pattern?
- Similar to DFA simulation
- But have to deal with $\varepsilon$ transitions and multiple transitions on the same input
- Instead of one state, we have to consider sets of states
NFA to DFA Conversion

• Simulation implicitly converts NFA -> DFA
• Subset construction
• Idea: subsets of set of all NFA states are equivalent and become one DFA state
• Algorithm simulates movement through NFA
• Key problem: how to treat ε-transitions?
\( \varepsilon \)-Closure

- Start state: \( q_0 \)
- \( \varepsilon \)-closure(\( S \)): \( S \) is a set of states

\[
\begin{align*}
\text{initialize: } & S \leftarrow \{q_0\} \\
T & \leftarrow S \\
\text{repeat } & T' \leftarrow T \\
& T \leftarrow T' \cup \bigcup_{s \in T'} \text{move}(s, \varepsilon) \\
\text{until } & T = T'
\end{align*}
\]
\( \varepsilon \)-Closure (T: set of states)

push all states in T onto stack
initialize \( \varepsilon \)-closure(T) to T
while stack is not empty do begin
  pop t off stack
  for each state u with \( u \in \text{move}(t, \varepsilon) \) do
    if \( u \notin \varepsilon \)-closure(T) do begin
      add u to \( \varepsilon \)-closure(T)
push u onto stack
    end
end
NFA Simulation

• After computing the $\varepsilon$-closure move, we get a set of states

• On some input extend all these states to get a new set of states

\[ \text{DFAedge}(T, c) = \varepsilon\text{-closure} \left( \bigcup_{q \in T} \text{move}(q, c) \right) \]
NFA Simulation

- \( \text{DFAedge}(T, c) = \epsilon\text{-closure}(\bigcup_{q \in T} \text{move}(q, c)) \)
- Start state: \( q_0 \)
- Input: \( c_1, \ldots, c_k \)

\[
T \leftarrow \epsilon\text{-closure}({q_0})
\]

\[
\text{for } i \leftarrow 1 \text{ to } k
\]

\[
T \leftarrow \text{DFAedge}(T, c_i)
\]
Conversion from NFA to DFA

• Conversion method closely follows the NFA simulation algorithm
• Instead of simulating, we can collect those NFA states that behave identically on the same input
• Group this set of states to form one state in the DFA
Example: subset construction
$\varepsilon$-closure($q_0$)
move($\varepsilon$-closure($q_0$), 0)
\[ \varepsilon\text{-closure}(\text{move}(\varepsilon\text{-closure}(q_0), 0)) \]

Diagram with states and transitions labeled with \(\varepsilon\) and 0.
move($\varepsilon$-closure($q_0$), 1)
\( \varepsilon\text{-closure}(\text{move}(\varepsilon\text{-closure}(q_0), 1)) \)
Subset Construction

add $\varepsilon$-closure$(q_0)$ to $Dstates$ unmarked

while $\exists$ unmarked $T \in Dstates$ do begin

mark $T$;

for each symbol $c$ do begin

$U := \varepsilon$-closure$(\text{move}(T, c))$;

if $U \not\in Dstates$ then

add $U$ to $Dstates$ unmarked

$Dtrans[d, c] := U$;

end

end

end
Subset Construction

states[0] = $\varepsilon$-closure($\{q_0\}$)
p = j = 0

while $j \leq p$ do begin
  for each symbol $c$ do begin
    $e = \text{DFAedge}(\text{states}[j], c)$
    if $e = \text{states}[i]$ for some $i \leq p$
    then $\text{Dtrans}[j, c] = i$
    else $p = p+1$
    states[$p$] = $e$
    $\text{Dtrans}[j, c] = p$
  end
  $j = j + 1$
end
DFA (partial)

[1, 2, 3, 4, 6, 9, 12] 0 [3, 4, 5, 6, 8, 9, 10, 13, 14]

1 [3, 4, 6, 7, 8, 9]
DFA for \(((0|1)^*00)|0\)
Minimization of DFAs

- [1, 2, 3, 4, 6, 9, 12]
- [3, 4, 6, 7, 8, 9]
- [3, 4, 5, 6, 8, 9, 10]
- [3, 4, 5, 6, 8, 9, 10, 11, 14]

States with labels [1, 2, 3, 4, 6, 9, 12], [3, 4, 6, 7, 8, 9], and [3, 4, 5, 6, 8, 9, 10] are connected by transitions labeled 0 and 1.
Minimization of DFAs

[3, 4, 5, 6, 8, 9, 10, 11, 14]

[3, 4, 6, 7, 8, 9]

[3, 4, 5, 6, 8, 9, 10]