Minimizing states in a DFA

Assume that there is a transition in the DFA for each state and symbol.

(If not, use the same construction we used earlier — before specifying the algorithm for simulating a DFA.)

So, intuitively, for each state $s$ and input string $x$, we can say that reading $x$ starting in $s$ takes us to a state $t$.

(fig 3.29)

Distinguishing states

We say that string $x$ distinguishes states $s$ and $t$ if reading $x$ starting in $s$ takes us to an accepting state, while reading $x$ starting in $t$ takes us to a non-accepting state, or vice versa.

For example, $\epsilon$ distinguishes every accepting state from every non-accepting state.

(fig 3.29)

For instance, reading $aba$ in $C$ takes us to $B$.

$b$ distinguishes $D$ from every other state.

Is there a string that distinguishes $B$ and $C$? $A$ and $C$?
In constructing a minimal DFA, we simply merge states that are indistinguishable...

The crucial stage of the algorithm partitions the set of states “appropriately.”

(A partition of $S$ is a set of nonempty subsets of $S$ that are pairwise disjoint and together contain all elements of $S$.)

Each element of the partition should consist of states that are indistinguishable from one another. Furthermore, any two indistinguishable states should belong to the same element of the partition.

So, for instance, the appropriate partition of states for (fig 3.29)

is \{ \{A, C\}, \{B\}, \{D\}, \{E\} \}

The algorithm starts with a partition of the set of states into accepting and non-accepting states, and refines this partition step by step.

Here’s essentially how the book describes a refinement step:

\begin{verbatim}
for each element $G$ of the partition $\Pi$ do begin
    partition $G$ so that two states $s$ and $t$ of $G$
    are in the same element of the partition of $G$ iff,
    for all input symbols $a$, reading $a$ in $s$ and reading $a$ in $t$
    take us to states in the same element of $\Pi$;
    replace $G$ in $\Pi$ by the elements of the partition of $G$;
end
\end{verbatim}

You repeat this construction until reaching a fixpoint.

Let me say this a little differently...
What we want is an algorithm to determine which pairs of states are distinguishable:

(If they are not distinguishable, they will be identified in the minimal FA.)

We will construct a set \( D \) of all pairs of distinguishable states, as follows:

let \( D' \) consist of all pairs with one final and one non-final state;
\[ D := \emptyset; \]
while \( D \neq D' \) do begin
\[ D := D'; \]
for each pair \((s, t)\) of states not already in \( D \) do
for each \( a \in \Sigma \) do
if \((\text{move}(s, a), \text{move}(t, a)) \in D\) then add \((s, t)\) to \( D' \);
end

When we’re done, \( D \) consists of all pairs of distinguishable states.

So if a pair \((s, t)\) does not belong to \( D \), then \( s \) and \( t \) are indistinguishable.
Once we have partitioned the states into groups of indistinguishable states (as coarsely as possible), we can construct the minimal FA by

- choosing a single representative for each group of states
- the representative states are the states of the new DFA
- for each representative state \( s \) and input symbol \( a \), there will be a transition from \( s \) to the representative of \( \text{move}(s, a) \)
- a state in the new DFA is final iff it was final in the original DFA

For example, we may choose representatives \( A, B, D, E \) for the partition \{ \{A, C\}, \{B\}, \{D\}, \{E\} \}…

(fig 3.29)

In addition, we can eliminate any state from which no final state is “reachable” and any state that is not “reachable” from the initial state.

Example
Combining FA’s in Lexical Analysis

In general, we are trying to match any number of patterns, looking for the longest possible match.

Depending on the pattern we match, there will also be actions to perform.

One approach would be to build a recognizer for each regular expression, and then try them in sequence, identifying for each the longest prefix of the input (if any) that matches the pattern. (If more than one recognizer accepts the longest prefix of the input, prefer the one with higher priority, as in Lex.)

Another alternative is to reduce each regular expression to an NFA, then combine the NFA’s in a single NFA whose accepting states are also distinguished by the regular expression to which they correspond…

Suppose that \(N(p_1), N(p_2), \ldots, N(p_n)\) are NFA’s for regular expressions \(p_1, \ldots, p_n\). Then the NFA below accepts

\[ p_1 \mid p_2 \mid \cdots \mid p_n \]

(fig 3.34)

We can simulate this NFA as before, except that we do not terminate immediately upon encountering an accepting state. Instead, we always keep track of last accepting state we’ve seen, and the input position when it was seen.

(If a given prefix yields multiple accepting states, remember the one that corresponds to the highest priority pattern.)

We stop when the NFA simulation yields an empty set of successor states.
For example, for a Lex program with translation rules

\[
\text{a} \rightarrow \text{action1; } \\
\text{abb} \rightarrow \text{action2; } \\
\text{a*b+} \rightarrow \text{action3; }
\]

We would build an NFA (similar to)

(fig 3.35b)

On input input \textit{abbba} we pass through sets of states

\[
\{0, 1, 3, 7\}, \{2, 4, 7\}, \{5, 8\}, \{6, 8\}, \emptyset
\]

\{2, 4, 7\}: match \textit{a} with pattern \textit{a}
\{5, 8\}: match \textit{ab} with pattern \textit{a*b+}
\{6, 8\}: match \textit{abb} with pattern \textit{a*b+} (preferred over \textit{a*b+})
\{8\}: match \textit{abbb} with pattern \textit{a*b+}

So we match \textit{abbb} and do \textit{action3}.

Yet another alternative approach:

Reduce the composite NFA to a DFA, again look for longest match...

<table>
<thead>
<tr>
<th>STATE</th>
<th>INPUT SYMBOL</th>
<th>PATTERN ANNOUNCED</th>
</tr>
</thead>
<tbody>
<tr>
<td>0137</td>
<td>247</td>
<td>8</td>
</tr>
<tr>
<td>247</td>
<td>7</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>∅</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>58</td>
<td>∅</td>
<td>68</td>
</tr>
<tr>
<td>68</td>
<td>∅</td>
<td>8</td>
</tr>
</tbody>
</table>

(fig 3.35b)
For next time

Homework 2 is due at beginning of class on Friday.

Exam 1 in class on Monday, October 2nd:

   open book, open notes

For first class after exam:

   Read 4.1–4.3.