Automata Theory  
Optional Homework 2  
Key

November 1, 2003

1. Apply the state minimization algorithm to the following DFA’s. In the not_equiv matrix, if state i is not equivalent to state j show the number of the pass through the main loop in which not_equiv[i][j] is assigned the value FALSE.

(a) The automaton in Figure 2-20 on page 94.
(b) The automaton in Figure 2-2 on page 59.

Recall the modified algorithm:

```c
int not_equiv[n][n];

for (i = 0; i < n-1; i++)
    for (j = i+1; j < n; j++)
        if (Is_final(i) != Is_final(j))
            not_equiv[i][j] = -1; /* TRUE */
        else
            not_equiv[i][j] = INFINITY; /* FALSE */

int counter = 0;
int changes_made = FALSE;
do {
    for (i = 0; i < n-1; i++)
        for (j = i+1; j < n; j++)
            if (not_equiv[i][j] == INFINITY) /* FALSE */
                for each a in Sigma
                    /* Use function in case delta(i,a) >= delta(j,a) */
                    if (!Equiv(not_equiv, delta(i,a), delta(j,a))) {
                        not_equiv[i][j] = counter; /* TRUE */
                        changes_made = TRUE;
                        break;
                    }
}
```
```cpp
counter++;
}
while(changes_made);
```

(a) In the row and column headers, the first value is the number used for the state in the algorithm, the second value is the number in figure 2-20.

<table>
<thead>
<tr>
<th></th>
<th>0=q_1</th>
<th>1=q_2</th>
<th>2=q_3</th>
<th>3=q_4</th>
<th>4=q_5</th>
<th>5=q_6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0=q_1</td>
<td>x</td>
<td>-1</td>
<td>∞</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1=q_2</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2=q_3</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>3=q_4</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>4=q_5</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>5=q_6</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

(b) Once again, in the row and column headers, the first value is the number used for the state in the algorithms, the second value is the number in figure 2-2.

<table>
<thead>
<tr>
<th></th>
<th>0=q_0</th>
<th>1=q_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0=q_0</td>
<td>x</td>
<td>-1</td>
</tr>
<tr>
<td>1=q_1</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

2. Suppose that \( L \) is a language. If \( u, v \in \Sigma^* \), define \( u \sim_L v \) to mean that for each \( w \in \Sigma^* \), \( uw \in L \) iff \( vw \in L \). Show that \( \sim_L \) is an equivalence relation on \( \Sigma^* \).

We need to see that \( \sim_L \) is reflexive, symmetric and transitive:

(a) Reflexive. Suppose \( u \in \Sigma^* \). We need to see that \( u \sim_L u \). This is equivalent to seeing that for each \( w \in \Sigma^* \), \( uw \in L \) iff \( uw \in L \). Since this is clearly the case, \( \sim_L \) is reflexive.

(b) Symmetric. Suppose \( u, v \in \Sigma^* \), and \( u \sim_L v \). We need to see that \( v \sim_L u \), or, equivalently, that for each \( w \in \Sigma^* \), \( vw \in L \) iff \( uw \in L \). But since by assumption, \( u \sim_L v \), we know that for each \( w \in \Sigma^* \), \( vw \in L \) iff \( uw \in L \). So for each \( w \in L \), \( vw \in L \) iff \( uw \in L \), and \( v \sim_L u \).

(c) Transitive. Suppose \( t, u, v \in \Sigma^* \), \( t \sim_L u \), and \( u \sim_L v \). We need to see that \( t \sim_L v \), or, equivalently, that for each \( w \in \Sigma^* \), \( tw \in L \) iff \( vw \in L \). Since \( t \sim_L u \) and \( u \sim_L v \), we know that for each \( w \in \Sigma^* \), \( tw \in L \) iff \( uw \in L \), and \( uw \in L \) iff \( vw \in L \). So for each \( w \in L \), \( tw \in L \) iff \( vw \in L \), and \( t \sim_L v \).

3. Use the principle of mathematical induction to show that \( \hat{\delta}(\hat{\delta}(q, u), v) = \hat{\delta}(q, uv) \) for all states \( q \) and for all strings \( u, v \in \Sigma^* \).

We proceed by induction on \(|u|\). If \(|u| = 0\), then \( u = e \), and

\[
\hat{\delta}(\hat{\delta}(q, u), v) = \hat{\delta}(\hat{\delta}(q, e), v) = \hat{\delta}(q, v) \quad \text{Definition of } \hat{\delta} = \hat{\delta}(q, uv)
\]
So suppose that \( n \geq 0, \) and if \( |u| = n \) and \( q \) is a state, then
\[
\hat{\delta}(\hat{\delta}(q, u), v) = \hat{\delta}(q, uv).
\]

Consider the case \( |u| = n + 1 \geq 1. \) Then \( u = aw, \) for some \( a \in \Sigma \) and for some \( w \in \Sigma^* \) with length \( n. \) We have
\[
\hat{\delta}(\hat{\delta}(q, u), v) = \hat{\delta}(\hat{\delta}(q, aw), v)
= \hat{\delta}(\hat{\delta}(\delta(q, a), w), v) \quad \text{Definition of } \hat{\delta}
= \hat{\delta}(\delta(q, a), \hat{\delta}(w, v)) \quad \text{Induction Hypothesis}
= \hat{\delta}(q, awv) \quad \text{Definition of } \hat{\delta}
\]

4. If \( M \) is a DFA with no unreachable states, we can use \( \simeq \) to define an equivalence relation \( \simeq_M \) on \( \Sigma^*. \) If \( u, v \in \Sigma^* \), define \( u \simeq_M v, \) if \( \hat{\delta}(s, u) \simeq \hat{\delta}(s, v). \)

(a) Show that \( \simeq_M \) is an equivalence relation.

(b) Is there any relation between the equivalence classes on \( \Sigma^* \) defined by \( \simeq_M \) and \( \sim_{L(M)}? \)

For example, are the equivalence classes of \( \simeq_M \) subsets of the equivalence classes of \( \sim_{L(M)}? \) If so, what is the relation?

(a) We need to see that \( \simeq_M \) is reflexive, symmetric and transitive.

i. Reflexive. Suppose that \( u \in \Sigma^*. \) We need to see that \( u \simeq_M u. \) This is equivalent to seeing that \( \hat{\delta}(s, u) \simeq \hat{\delta}(s, u). \) But we’ve already shown that \( \simeq \) is an equivalence relation. So \( \hat{\delta}(s, u) \simeq \hat{\delta}(s, u) \) and \( u \simeq_M u. \)

ii. Symmetric. Suppose that \( u, v \in \Sigma^*, \) and \( u \simeq_M v. \) We need to see that \( v \simeq_M u. \) This also follows from the fact that \( \simeq \) is an equivalence relation. Since \( u \simeq_M v, \) we know that \( \hat{\delta}(s, u) \simeq \hat{\delta}(s, v). \) Since \( \simeq \) is symmetric, we also know that \( \hat{\delta}(s, v) \simeq \hat{\delta}(s, u). \) This is precisely the condition for \( v \simeq_M u. \)

iii. Transitive. Suppose that \( t, u, v \in \Sigma^*, \) \( t \simeq_M u, \) and \( u \simeq_M v. \) Then \( \hat{\delta}(s, t) \simeq \hat{\delta}(s, u) \) and \( \hat{\delta}(s, u) \simeq \hat{\delta}(s, v). \) Since \( \simeq \) is an equivalence relation, we have that \( \hat{\delta}(s, t) \simeq \hat{\delta}(s, v), \) and hence \( t \simeq_M v. \)

(b) Yes. The equivalence classes of \( \simeq_M \) are subsets of the equivalence classes of \( \sim_{L(M)} \).

If \( u \in \Sigma^*, \) denote its equivalence class under \( \simeq_M \) by \( [u]_M. \) That is,
\[
[u]_M = \{v \in \Sigma^*: v \simeq_M u\}.
\]

Also denote the equivalence class of \( u \) with respect to \( \sim_{L(M)} \) by
\[
[u]_L = \{t \in \Sigma^*: t \sim_{L(M)} u\}.
\]

Finally, in order to simplify notation, we’ll write \( L = L(M) \).

We want to see that if \( u \in \Sigma^*, \) then \( [u]_M \subseteq [u]_L. \) So let \( v \in [u]_M. \) Then \( v \simeq_M u, \) and \( \hat{\delta}(s, v) \simeq \hat{\delta}(s, u). \) Let \( p = \hat{\delta}(s, v) \) and \( q = \hat{\delta}(s, u). \) Then, by definition of \( \simeq, \) we have that
for each $w \in \Sigma^*$, $\hat{\delta}(p, w) \in F$ iff $\hat{\delta}(q, w) \in F$. But from the preceding problem, we know that

$$\hat{\delta}(p, w) = \hat{\delta}(\hat{\delta}(s, v), w) = \hat{\delta}(s, vw),$$

and

$$\hat{\delta}(q, w) = \hat{\delta}(\hat{\delta}(s, u), w) = \hat{\delta}(s, uw).$$

So for each $w \in \Sigma^*$, $\hat{\delta}(s, vw) \in F$ iff $\hat{\delta}(s, uw) \in F$. Now $\hat{\delta}(s, vw) \in F$ is exactly the condition for $vw \in L$, and $\hat{\delta}(s, uw) \in F$ is exactly the condition for $uw \in L$. Thus, if $v \simeq_M u$, we have that for each $w \in \Sigma^*$, $vw \in L$ iff $uw \in L$. This is the condition for $v \sim_L u$, or, equivalently, for $v \in [u]_L$.

Thus, if $v \in [u]_M$, then $v \in [u]_L$, and hence $[u]_M \subseteq [u]_L$. 

4