Invariants for equivalence of finite automata

Mike Spivey

Let $S = (S, \Sigma, \delta_S, s_0, F_S)$ and $\mathcal{T} = (T, \Sigma, \delta_T, t_0, F_T)$ be two deterministic finite automata with the same alphabet Σ. We say S and $\mathcal{T}$ are *equivalent* if, for each word $w \in \Sigma^*$, we have $\delta_S(s_0, w) \in F_S$ if and only if $\delta_T(t_0, w) \in F_T$. In other words, S and $\mathcal{T}$ are equivalent if $R \subseteq E$, where

$$R = \{ (\delta_S(s_0, w), \delta_T(t_0, w)) \mid w \in \Sigma^* \},$$

and

$$E = \{ (s, t) \mid s \in F_S \Leftrightarrow t \in F_T \}.$$

What's more, R is the smallest set that contains (s_0, t_0) and is closed under Δ, where

$$\Delta(s, t) = \{ (\delta_S(s, a), \delta_T(t, a)) \mid a \in \Sigma \}.$$

Let us assume that S and T are disjoint and let $X = S \cup T$ and $\delta = \delta_S \cup \delta_T$ and $F = F_S \cup F_T$. Then the problem of determining whether S and $\mathcal{T}$ are equivalent amounts to determining whether the states s_0 and t_0 are equivalent in a certain sense in this joined automaton. We can extend the definition of Δ to $X \times X$ by writing

$$\Delta(x, y) = \{ (\delta(x, a), \delta(y, a)) \mid a \in \Sigma \},$$

and R remains the smallest relation containing (s_0, t_0) that is closed under this operation.

We'll use the notation $\overline{Q}$ for the *equivalence closure* of a relation Q, that is, the smallest equivalence relation that contains Q. If we take Q as the set of pairs that have been submitted to the *Union* operation in a disjoint-sets data structure, then $\overline{Q}$ is the equivalence relation that is tested with $\mathit{Find}(x) = \mathit{Find}(y)$. This closure operation has the property that if U is an equivalence relation, and Q is any relation, then $Q \subseteq U$ if and only if $\overline{Q} \subseteq U$.

Now observe that we can extend the relation $E \subseteq S \times T$ to an equivalence relation E' on X, with $(x, y) \in E'$ if and only if $x \in F \Leftrightarrow y \in F$. For $s \in S$ and $t \in T$, we have $(s, t) \in E$ if and only if $(s, t) \in E'$. Consequently $R \subseteq E$ if and only if $R \subseteq E'$.

Now consider the program shown in Figure 1. I claim that the following are invariants:

(i) $(s_0, t_0) \in \overline{Q} \cup W$,

```
Q := ∅; W := {(s0, t0)};
while W ≠ ∅ do
  extract (u, v) from W;

  if (u, v) ∉ E then
    return FALSE
  end;

  if (u, v) ∉ Q̄ then
    Q := Q ∪ {(u, v)};
    W := W ∪ Δ(u, v)
  end
end;
return TRUE
```

Figure 1: *Program for testing equivalence*

(ii) $W \subseteq R$,

(iii) $Q \subseteq E$,

(iv) If $(s,t) \in Q$ then $\Delta(s,t) \subseteq \overline{Q} \cup W$.

It's easy to check that most of these relationships are maintained and the program is correct. In particular, if the loop body returns *FALSE*, then (u,v) is in R but not E, so S and $\mathcal{T}$ are not equivalent.

For invariance of (i), observe that the value of $\overline{Q} \cup W$ only increases: if (u,v) is chosen and removed from W, then the loop body ensures that $(u,v) \in \overline{Q}$. In particular, (s_0, t_0) remains in the set. Also for (iv), since $\overline{Q} \cup W$ only increases, we need worry only about a new pair (u,v) added to Q; but the adjustment to W deals with keeping the invariant true.

When the loop terminates, we have $W = \emptyset$, and invariant (iv) tells us that

$$\text{if } (s,t) \in Q \text{ then } \Delta(s,t) \subseteq \overline{Q}.$$

We can write this property as $Q \subseteq U$, where

$$U = \{(x,y) \mid \Delta(x,y) \subseteq \overline{Q}\}.$$

Now U is an equivalence relation, so it follows by the properties of equivalence closure that that $\overline{Q} \subseteq U$ also, and so $\overline{Q}$ is closed under Δ. But also $(s_0, t_0) \in \overline{Q}$, so $\overline{Q}$ includes all of R. Appealing to invariant (iii), we have $Q \subseteq E \subseteq E'$, with E' an equivalence relation, and so $R \subseteq \overline{Q} \subseteq E'$, making S and $\mathcal{T}$ equivalent.

Observing that Q is not used in the program other than via the operations

$$Q := \emptyset$$

$$Q := Q \cup \{(s,t)\}$$

$$(s,t) \in \overline{Q}$$

we can implement Q with a disjoint sets data structure.