Minimizing DFAs

CSC320

Fall 2015

DFA State Minimization

- Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, for a language L, there is a procedure for constructing a *minimal* DFA with as few states as possible which is unique up to isomorphism (i.e., renumbering of the states).

- The process has two stages:
  - Get rid of inaccessible states.
  - Collapse "equivalent" states.

Collapsing states

$\delta(q, w) =$ state reached starting in state q and processing every symbol in w.

If $M = (Q, \Sigma, \delta, q_0, F)$, and $q \in Q$, let M_q be identical to M, but with start state q instead of q_0

Idea: p and q can be collapsed if $L(M_p) = L(M_q)$

Collapsing states – why does it work?

- Consider M with p and q collapsed – call it M' and call the collapsed state pq
- Suppose $w \in L(M)$. There will be a *unique* path in M from q_0 to a final state.
  - If the path does not pass through p or q, then nothing changes when we collapse them
  - Otherwise, suppose $w = uv$ and $\delta(q_0, u) = p$. Then $v \in L(M_p) = L(M_q)$, so in M', $\delta'(q_0, u) = pq$, and $v \in L({M'}_{pq})$, so $w = uv \in L(M')$
- Similar argument for $w \notin L(M)$.

Collapsibility is an equivalence relation

If p and q can be collapsed we say they are *equivalent*, and we write $p \equiv q$.
Otherwise we say they are *distinguishable* and there is some string which *distinguishes* p from q.

Claim: The relation $\equiv$ is an equivalence relation, that is, it is

- reflexive: $p \equiv p$ for all p.
- symmetric: if $p \equiv q$, then $q \equiv p$.
- transitive: if $p \equiv r$ and $r \equiv q$ then $p \equiv q$.

Equivalence class of $q = [q] = \{p \mid p \equiv q\}$

Defining M'

$M' = (Q', \Sigma, \delta', [q_0], F')$, where

- $Q' = \{[p] \mid p \in Q\}$
- $\delta([p], a) = [\delta(p, a)]$
- $F' = \{[p] | p \in F\}$

Correctness of the definition – defining δ'

Claim: δ' is well-defined

Proof: It must be the case that for every $q \in [p]$, and every $a \in \Sigma$ if $\delta(q, a) = r$ then $r \in [\delta(p, a)]$. Assume not, then there is a string w which distinguishes r from $\delta(p, a)$. But then aw distinguishes q from p, contradicting our assumption that $q \in [p]$.

Correctness of the definition – equivalence

Theorem: $L(M') = L(M)$

Proof: We begin by observing that $\delta'([q], w) = [\delta(q, w)]$ (why?)

But then:

$$\delta'([q_0], w) \in F' \text{ iff } [\delta(q_0, w)] \in F' \text{ iff } \delta(q_0, w) \in F.$$

I.e. $w \in L(M')$ iff $w \in L(M)$

Correctness of the definition – minimality

- Suppose there is a DFA M'' such that $L(M'') = L(M')$ and M'' has fewer states. Let q_0'' be the start state of M''

- There must be states $p \neq q$ in M' and a state r in M'' for which there are strings u, v such that $\delta'([q_0], u) = p$, $\delta'([q_0], v) = q$, and $\delta''(q_0'', u) = \delta''(q_0'', v) = r$ (why?)

- So for any x, $ux \in L(M'')$ iff $vx \in L(M'')$
- But since p and q were not collapsed in M', $ux \in L(M')$ iff $v \notin L(M')$

Constructing M'

- Basically comes down to constructing Q'
- We construct Q' by *successively refining* a partition of Q
- We rely on the following
- **Lemma:** if p and q are not equivalent, then one of the following must hold (why?)
  1. $p \in F$ and $q \notin F$ (or vice versa)
  2. There are states p' and q' which are not equivalent, and a symbol a such that $\delta(p,a) = p'$ and $\delta(q,a) = q'$

Constructing M'

Once we have defined Q' we are pretty much done. So we just need a procedure that does the following:

Input: $M = (Q, \Sigma, \delta, q_0, F)$

Output: A partition $\Pi = \{C_1, C_2, \dots, C_k\}$ of Q such that each C_i is an $\equiv$-equivalence class

The procedure

$\Pi \leftarrow \{F, Q - F\}$; refined $\leftarrow true;$

While refined **do**

refined $\leftarrow false;$

For $B \in \Pi$ **do**

If $\exists\, a, \exists\, p, q \in B, \exists\, B' \in \Pi$, **s.t.** $\delta(p, a) \in B'$ **and** $\delta(q, a) \notin B'$ **then**

$B_1 \leftarrow \{p \in B \mid \delta(p, a) \in B'\}; B_2 \leftarrow B - B_1;$

$\Pi \leftarrow (\Pi - \mathrm{B}) \cup B_1 \cup B_2$; refined $\leftarrow true;$

Correctness of the procedure

- This follows directly from the lemma – as long as there are states which are not equivalent, refined will be *true*