Lecture 8 Minimization

Tuesday, February 2, 2021 11:32 AM

In NFA's the transition relation is

$$\Delta : Q \times \Sigma \to 2^Q$$

$\phi \in 2^Q$ so $\Delta(Q_i, a) = \phi$ is possible

NOTATION REMINDERS : DFA

$$\delta : S \times \Sigma \to S$$

$$\delta^* : S \times \Sigma^* \to S$$

$$\delta^*(s, \varepsilon) = s$$

$$\delta^*(s, aw) = \delta^*(\delta(s,a), w)$$

$$\delta^*(s, wa) = \delta(\delta^*(s,w), a)$$

MINIMIZATION of DFA's:

Lump together states that behave exactly the same way.

Def Given a DFA $M = (S, s_0, \delta, F)$

over alphabet Σ we say, $p, q \in S$ are **equivalent** and write $p \approx q$ if

$$\forall x \in \Sigma^* \quad \delta^*(p,x) \in F \Leftrightarrow \delta^*(q,x) \in F$$

Remark when are p, q not $\approx$?

$$\exists x \in \Sigma^* \ (\delta^*(p,x) \in F \ \& \ \delta^*(q,x) \notin F)$$
$$\text{OR} \ (\delta^*(p,x) \notin F \ \& \ \delta^*(q,x) \in F).$$

we call such an x a **distinguishing string**.

OBSERVATION $\approx$ is an equivalence relation.

We write $[p] = \{q \mid p \approx q\}$

if $p \approx q$ then $[p] = [q]$.

LEMMA A $\quad p \approx q \Rightarrow \forall a \in \Sigma$

$$\delta(p,a) \approx \delta(q,a).$$

PROOF Suppose $\delta^*(\delta(p,a), x) \in F$

$$= \delta^*(p, ax) \in F$$

But we assumed $p \approx q$ so

$$\delta^*(q, ax) \in F$$

i.e. $\delta^*(\delta(q,a), x) \in F$

similarly for the other direction

So the proof is complete ☺

REMARK $p \approx q$ can be written as

$[p] = [q]$. So what we have shown

is $[p] = [q] \Rightarrow [\delta(p,a)] = [\delta(q,a)]$ $\forall a \in \Sigma$.

We can construct a new machine:

$M' = (S', s_0', \delta', F')$

S': equivalence classes of $\approx$

s_0': $[s_0]$

$\delta'([s], a) = [\delta(s,a)]$ [well defined by lemma A]

$F' = \{[s] \mid s \in F\}$

LEMMA B $p \in F$ & $p \approx q \Rightarrow q \in F$ (DIY)

LEMMA C $\forall w \in \Sigma^*$

$$\delta'^*([p], w) = [\delta^*(p, w)]$$

PROOF Induction on w

BASE $\omega = \varepsilon$

$$\delta'^*([p], \varepsilon) = [p] = [\delta^*(p, \varepsilon)] \checkmark$$

Induction step: Assume true for ω

$$\delta'^*([p], \omega) = [\delta^*(p, \omega)]$$

Calculate:

$$\delta'^*([p], \omega a) = \delta'(\delta'^*([p], \omega), a) \quad \text{Def of } \delta'^*$$

$$= \delta'([\delta^*(p, \omega)], a) \quad \text{Ind. hyp.}$$

$$= [\delta(\delta^*(p, \omega), a)] \quad \text{Def of } \delta'$$

$$= [\delta^*(p, \omega a)] \quad \text{Done.}$$

THM $L(M') = L(M)$

Proof $x \in L(M') \Leftrightarrow \delta'^*([s_0], x) \in F'$

$$\Leftrightarrow [\underline{\delta^*(s_0, x)}] \in F'$$

$$\Leftrightarrow \delta^*(s_0, x) \in F \Leftrightarrow x \in L(M) \; \blacksquare$$

The machine with equivalent states lumped together recognizes the same language as the original machine.

Idea : Assume initially all states are equivalent. Then start splitting the states as you examine the transitions.

Def $p \bowtie q$ if $\exists w \in \Sigma^*$ s.t.

$$\delta^*(p, w) \in F \ \& \ \delta^*(q, w) \notin F$$

OR

$$\delta^*(p, w) \notin F \ \& \ \delta^*(q, w) \in F.$$

i.e. $p \bowtie q$ is $\neg(p \approx q)$

If $p \bowtie q$ then p & q are distinguishable

FACT If $\exists a \in \Sigma$ s.t. $\delta(p, a) \bowtie \delta(q, a)$ then $p \bowtie q$. [DIY]

ALGORITHM

Step 0 Get rid of unreachable states.

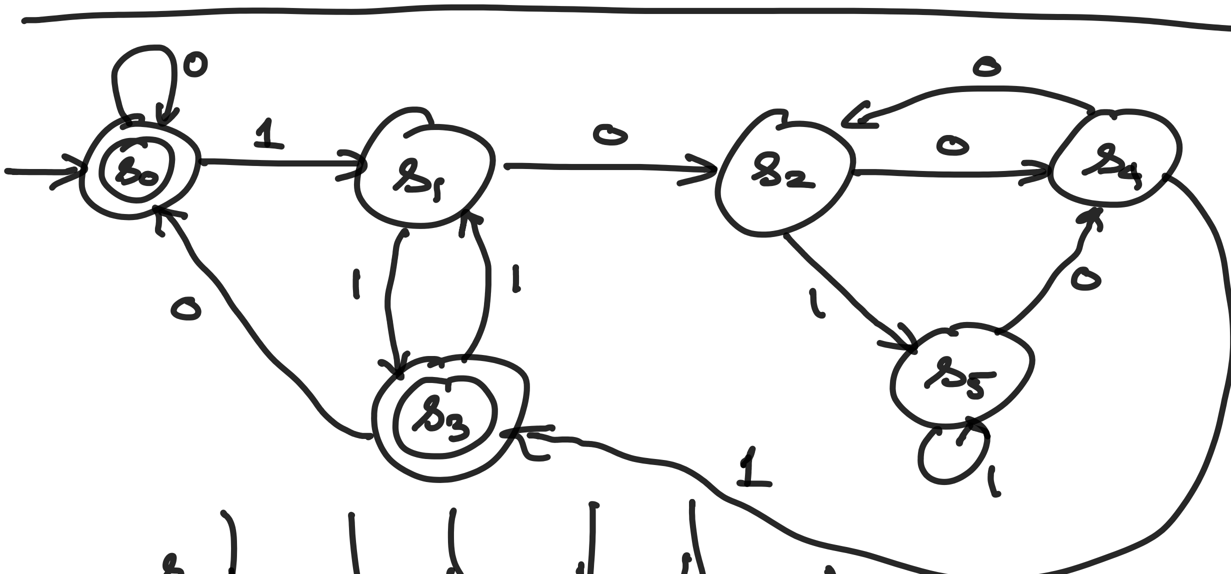
Step 1 Define an $S \times S$ array of booleans

step 2 For every pair $(p,q) \in S \times S$ such that $p \in F$ and $q \notin F$ put a 0 in the (p,q) cell of the array.

step 3 Repeat until there are no more changes:

- For each pair (p,q) that is not marked with a 0 check if $\exists a \in \Sigma$ s.t. $(\delta(p,a), \delta(q,a))$ is marked with a 0, if so mark (p,q) with a 0.

step 4 Mark anything remaining 1



s_0						
s_1	0					

INITIAL

s_2	0	0				
s_3	1	0	0			
s_4	0	1	0	0		
s_5	0	0	1	0	0	
	s_0	s_1	s_2	s_3	s_4	s_5

NEXT

EQUIVALENT

THM If two states are NOT labelled by a 0 then they must be equivalent

PROOF Assume theorem is false. So there are some pairs of states s.t. $p \bowtie q$ but they are not marked. Call such a pair a bad pair. Each such bad pair must have a distinguishing string. Choose a bad pair with the shortest distinguishing string. $x = x_1 \dots x_n$ $x_i \in \Sigma$

Note $x \neq \varepsilon$ [why not?]

Suppose (s,t) is a bad pair and x is the dist. string for this pair

$\delta^*(s,x) \bowtie \delta^*(t,x)$ since

$\delta^*(\delta(s,x_1), x_2 \cdots x_n) \in F$

$\delta^*(\delta(t,x_1), x_2 \cdots x_n) \notin F$

so $\underline{\delta(s,x_1)} \bowtie \underline{\delta(t,x_1)}$

& $x_2 \cdots x_n$ is the distinguishing string.

So $\delta(s,x_1)$ & $\delta(t,x_1)$ cannot be a bad pair so the algorithm did mark them. But then according to the code s,t would be marked in the next step.

This is a contradiction. ∎

So this algorithm finds **all** the equivalent states.

RUNNING TIME: $O(n^4)$ n # of states

each round is $O(n^2)$

$O(n^2)$ rounds

IMPROVED alg $O(n^2 k)$ [illegible]

HOPCROFT'S ALG: $O(n \log n) \leftarrow$

BRZOZOWSKI: $O(2^n)$!