

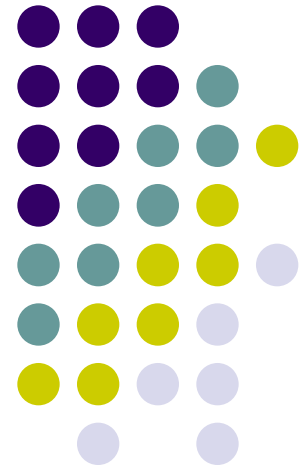
# DFA Minimization

---

Jeremy Mange

CS 6800

Summer 2009

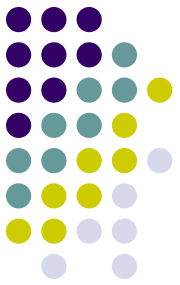




# DFA

- Deterministic Finite Automata (DFSA)
  - $(Q, \Sigma, \delta, q_0, F)$ 
    - $Q$  – (finite) set of states
    - $\Sigma$  – alphabet – (finite) set of input symbols
    - $\delta$  – transition function
    - $q_0$  – start state
    - $F$  – set of final / accepting states

# DFA



- Often representing as a diagram:



# DFA Minimization

- Some states can be redundant:
  - The following DFA accepts  $(a|b)^+$
  - State  $s_1$  is not necessary

# DFA Minimization



- So these two DFAs are *equivalent*:



# DFA Minimization

- This is a *state-minimized* (or just *minimized*) DFA
  - Every remaining state is necessary



# DFA Minimization

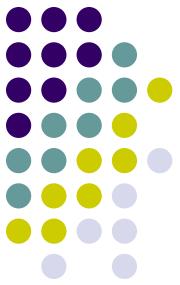
- The task of *DFA minimization*, then, is to automatically transform a given DFA into a state-minimized DFA
  - Several algorithms and variants are known
  - Note that this also in effect can minimize an NFA (since we know algorithm to convert NFA to DFA)



# DFA Minimization Algorithm

- Recall that a DFA  $M=(Q, \Sigma, \delta, q_0, F)$
- Two states  $p$  and  $q$  are distinct if
  - $p$  in  $F$  and  $q$  not in  $F$  or vice versa, or
  - for some  $\alpha$  in  $\Sigma$ ,  $\delta(p, \alpha)$  and  $\delta(q, \alpha)$  are distinct
- Using this inductive definition, we can calculate which states are distinct

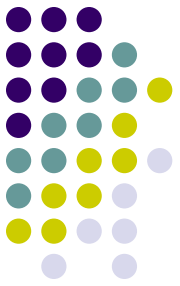




# DFA Minimization Algorithm

- Create lower-triangular table DISTINCT, initially blank
- For every pair of states  $(p, q)$ :
  - If  $p$  is final and  $q$  is not, or vice versa
    - $\text{DISTINCT}(p, q) = \varepsilon$
- Loop until no change for an iteration:
  - For every pair of states  $(p, q)$  and each symbol  $\alpha$ 
    - If  $\text{DISTINCT}(p, q)$  is blank and  $\text{DISTINCT}(\delta(p, \alpha), \delta(q, \alpha))$  is not blank
      - $\text{DISTINCT}(p, q) = \alpha$
- Combine all states that are not distinct

# Very Simple Example



s0			
s1			
s2			
	s0	s1	s2

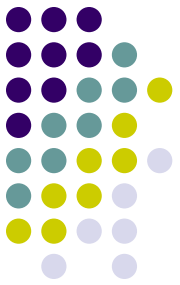


# Very Simple Example

s0			
s1	$\epsilon$		
s2	$\epsilon$		
	s0	s1	s2

Label pairs with  $\epsilon$  where one is a final state and the other is not

# Very Simple Example



s0			
s1	$\epsilon$		
s2	$\epsilon$		
	s0	s1	s2

Main loop (no changes occur)



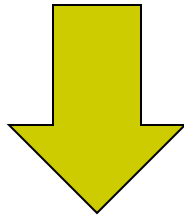
# Very Simple Example

s0			
s1	$\epsilon$		
s2	$\epsilon$		
	s0	s1	s2

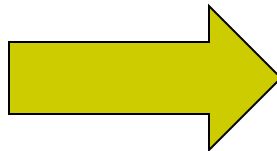
$\text{DISTINCT}(s1, s2)$  is empty, so s1 and s2 are equivalent states



# Very Simple Example



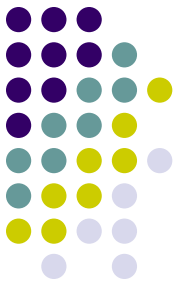
**Merge s1 and s2**



# More Complex Example



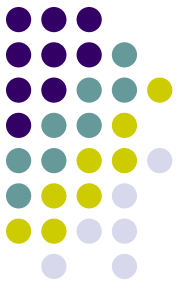
# More Complex Example



- Check for pairs with one state final and one not:

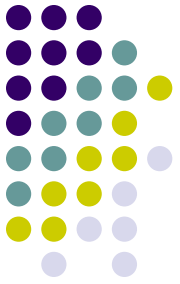


# More Complex Example

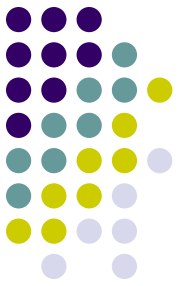


- First iteration of main loop:

# More Complex Example

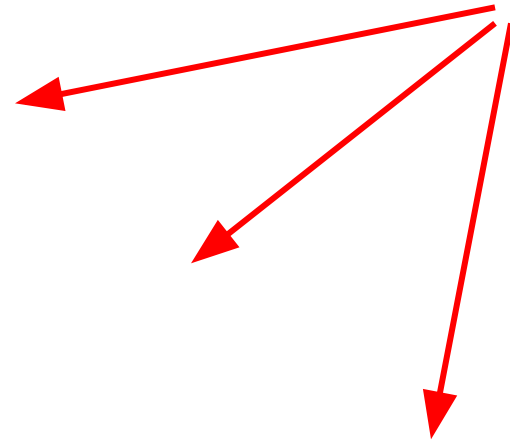


- Second iteration of main loop:

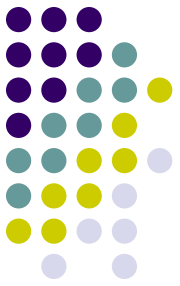


# More Complex Example

- Third iteration makes no changes
  - Blank cells are equivalent pairs of states



# More Complex Example



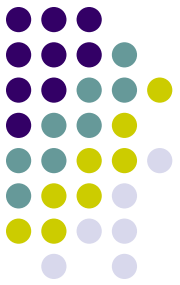
- Combine equivalent states for minimized DFA:



# Conclusion

- DFA Minimization is a fairly understandable process, and is useful in several areas
  - Regular expression matching implementation
  - Very similar algorithm is used for compiler optimization to eliminate duplicate computations
- The algorithm described is  $O(kn^2)$ 
  - John Hopcraft describes another more complex algorithm that is  $O(k (n \log n) )$

# Possible Exam Question



- *Question:* Inductively define when two states in a DFA are distinct.
- *Answer:*
  - Two states  $p$  and  $q$  are distinct if
    - $p \in F$  and  $q \notin F$  or vice versa, or
    - for some  $\alpha \in \Sigma^*$ ,  $\delta(p, \alpha)$  and  $\delta(q, \alpha)$  are distinct



# References

- Ullman, A. V., Hopcroft, J. E. and Ullman, J. D. (1974) *The Design and Analysis of Computer Algorithms*. Addison-Wesley.
- Hopcroft, J. (1971) *An  $N \log N$  Algorithm for Minimizing States in a Finite Automaton*. Stanford University.
- Parthasarathy, M. and Fleck, M. (2007) *DFA Minimization*. University of Illinois at Urbana-Champaign.  
<http://www.cs.uiuc.edu/class/fa07/cs273/Handouts/minimization/minimization.pdf>