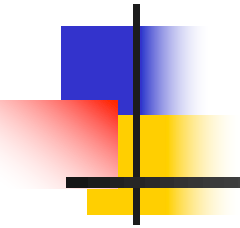
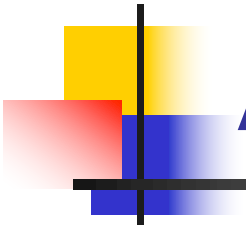


Course 8
Equivalence and Minimization of DFAs

The structure and the content of the lecture is based on http://www.eecs.wsu.edu/~ananth/CptS317/Lectures/index.htm



Applications of interest

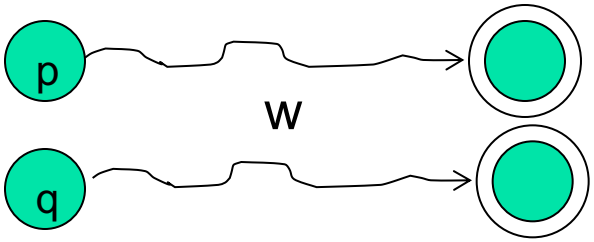
- Comparing two DFAs:
  - $L(DFA_1) == L(DFA_2)$?
- How to minimize a DFA?
  1. Remove unreachable states
  2. Identify & condense equivalent states into one

When to call two states in a DFA "equivalent"?

Past doesn't matter - only future does!

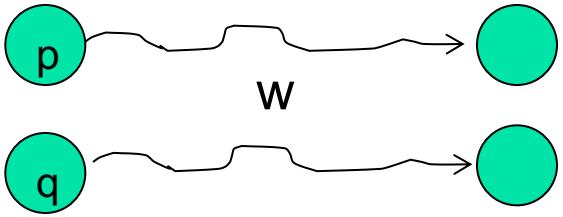
Two states p and q are said to be *equivalent* iff:

i) Any string w accepted by starting at p is also accepted by starting at q;



AND

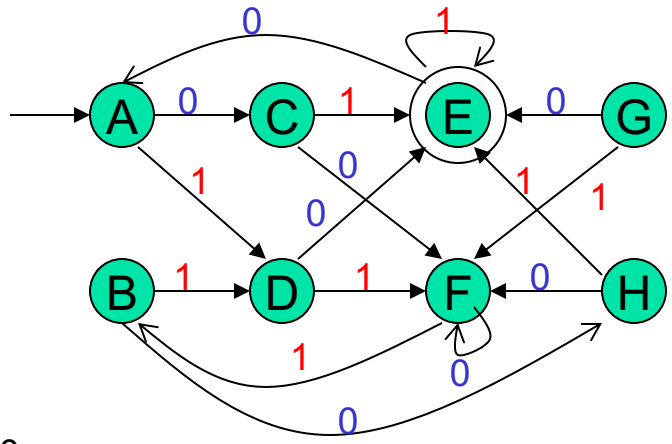
i) Any string w rejected by starting at p is also rejected by starting at q.



➔ p≡q

Computing equivalent states in a DFA

Table Filling Algorithm



	A	B	C	D	E	F	G	H
A	=							
B	=	=						
C	x	x	=					
D	x	x	x	=				
E	x	x	x	x	=			
F	x	x	x	x	x	=		
G	x	x	x	=	x	x	=	
H	x	x	=	x	x	x	x	=

Pass #0

1. Mark accepting states ≠ non-accepting states

Pass #1

1. Compare every pair of states
2. Distinguish by one symbol transition
3. Mark = or ≠ or blank (i.e. can not distinguish)

Pass #2

1. Compare every pair of states
2. Distinguish by up to two symbol transitions (until different or same or tbd)

….

(keep repeating until table complete) *How the table on the right was obtained?* **Table Filling Algorithm**

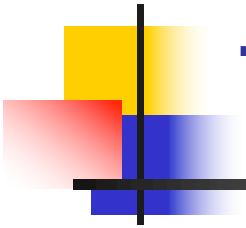
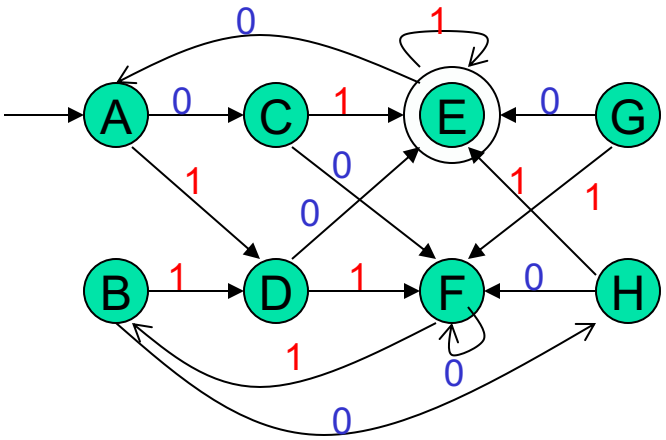


Table Filling Algorithm

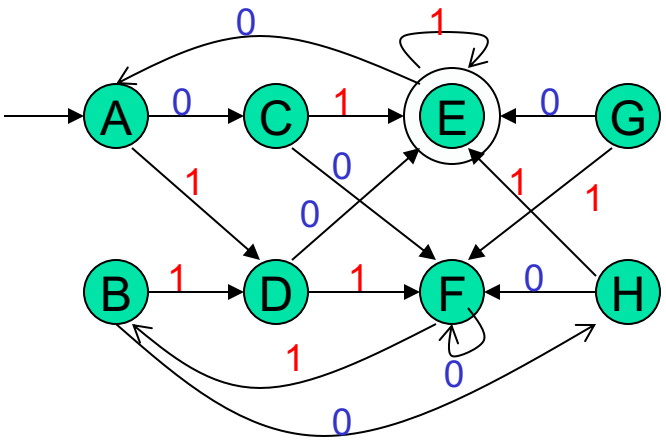
- Recursive discovery of distinguishable states in a DFA
  - **Base case**: If p is an accepting state and q is not accepting then the pair {p,q} is distinguishable.
  - **Induction**: Let p, q be states s.t. for some input symbol a, r = δ (p, a) and s = δ (q, a) are known to be distinguishable. Then the pair {p,q} is distinguishable.

Table Filling Algorithm - step by step



A	=							
B		=						
C			=					
D				=				
E					=			
F						=		
G							=	
H								=
	A	B	C	D	E	F	G	H

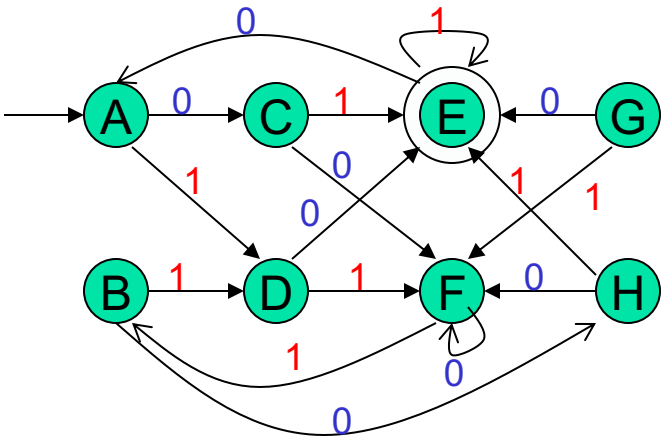
Table Filling Algorithm - step by step



	A	B	C	D	E	F	G	H
A	=							
B		=						
C			=					
D				=				
E	X	X	X	X	=			
F					X	=		
G					X		=	
H					X			=

1. Mark X between accepting vs. non-accepting state

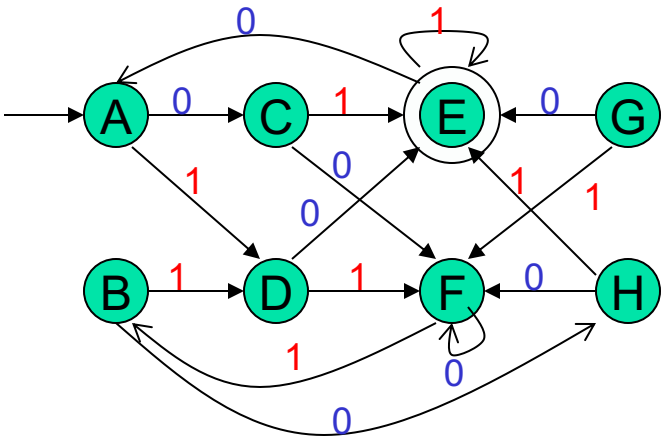
Table Filling Algorithm - step by step



1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings

	A	B	C	D	E	F	G	H
A	=							
B		=						
C	X		=					
D	X			=				
E	X	X	X	X	=			
F					X	=		
G	X				X		=	
H	X				X			=

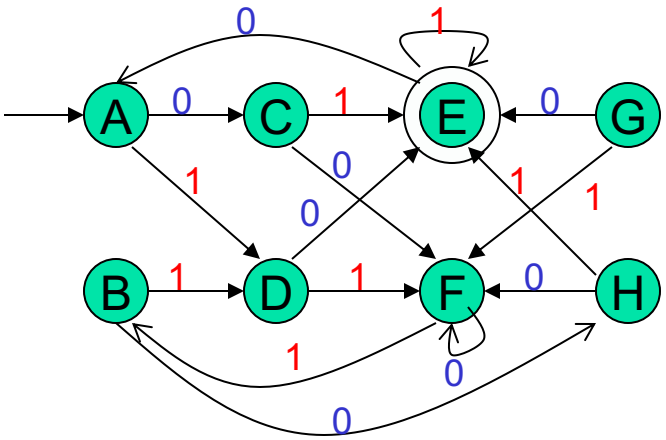
Table Filling Algorithm - step by step



1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings

	A	B	C	D	E	F	G	H
A	=							
B		=						
C	X	X	=					
D	X	X		=				
E	X	X	X	X	=			
F					X	=		
G	X	X			X		=	
H	X	X			X			=

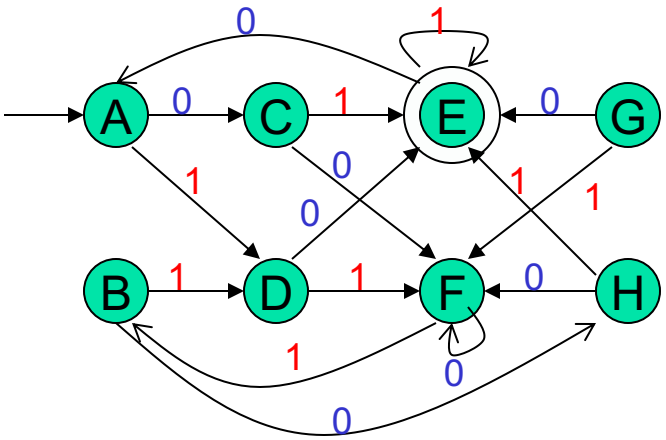
Table Filling Algorithm - step by step



1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings

	A	B	C	D	E	F	G	H
A	=							
B		=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F			X		X	=		
G	X	X	X		X		=	
H	X	X	=		X			=

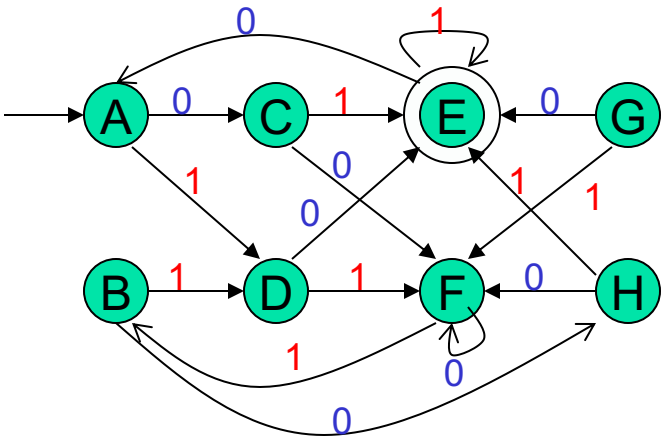
Table Filling Algorithm - step by step



1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings

	A	B	C	D	E	F	G	H
A	=							
B		=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F			X	X	X	=		
G	X	X	X	=	X		=	
H	X	X	=	X	X			=

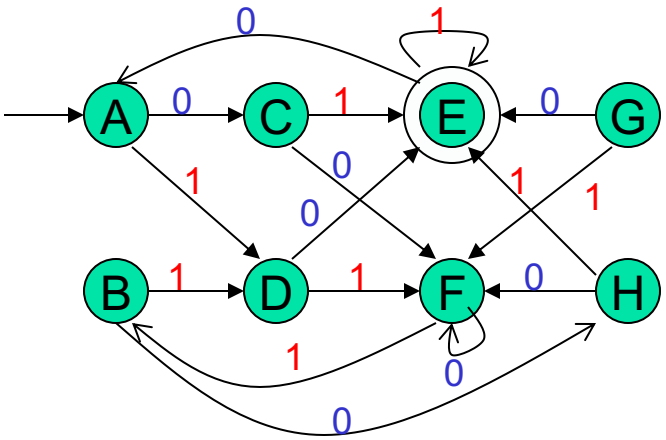
Table Filling Algorithm - step by step



1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings

A	=							
B		=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F			X	X	X	=		
G	X	X	X	=	X	X	=	
H	X	X	=	X	X	X		=
	A	B	C	D	E	F	G	H

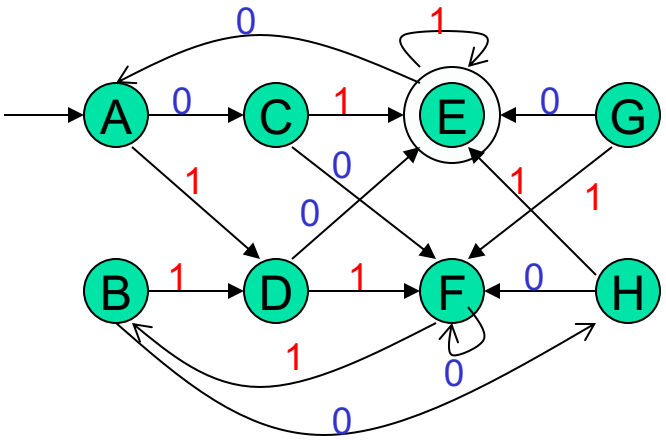
Table Filling Algorithm - step by step



1. Mark X between accepting vs. non-accepting state
2. Look 1- hop away for distinguishing states or strings

A	=							
B		=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F			X	X	X	=		
G	X	X	X	=	X	X	=	
H	X	X	=	X	X	X	X	=
	A	B	C	D	E	F	G	H

Table Filling Algorithm - step by step

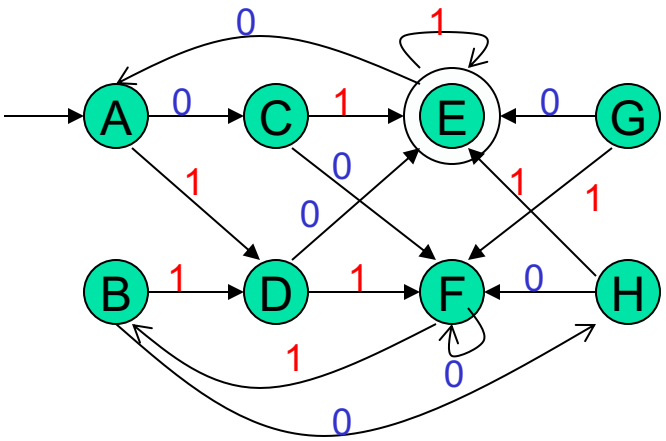


A	=							
B	**=**	=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F	**X**	**X**	X	X	X	=		
G	X	X	X	**=**	X	X	=	
H	X	X	**=**	X	X	X	X	=
	A	B	C	D	E	F	G	H

1. Mark X between accepting vs. non-accepting state
2. Pass 1:
   Look 1- hop away for distinguishing states or strings
3. Pass 2:
   Look 1-hop away again for distinguishing states or strings

continue….

Table Filling Algorithm - step by step



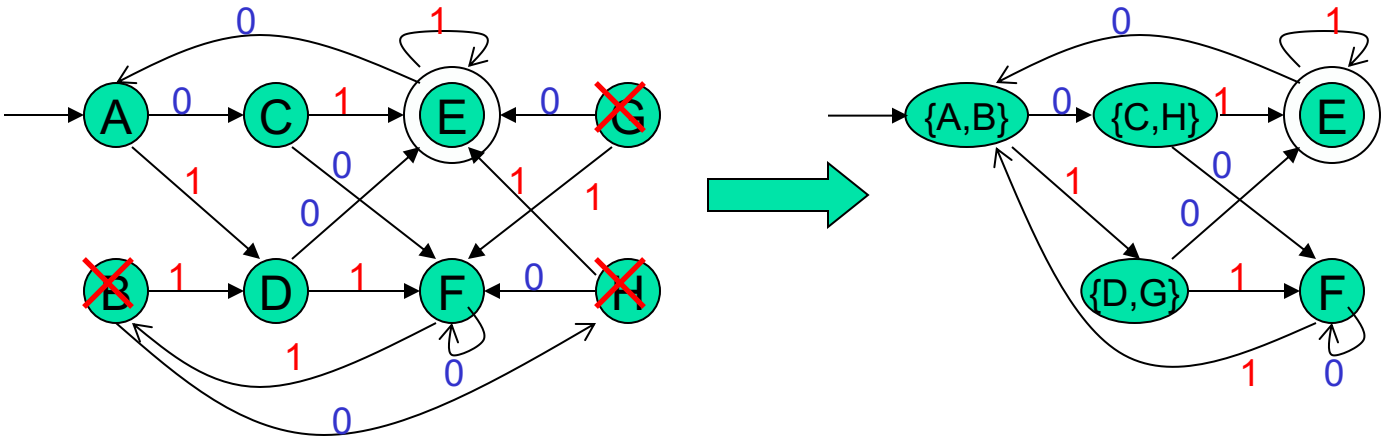
	A	B	C	D	E	F	G	H
A	=							
B	=	=						
C	X	X	=					
D	X	X	X	=				
E	X	X	X	X	=			
F	X	X	X	X	X	=		
G	X	X	X	=	X	X	=	
H	X	X	=	X	X	X	X	=

1. Mark X between accepting vs. non-accepting state
2. Pass 1:
   Look 1- hop away for distinguishing states or strings
3. Pass 2:
   Look 1-hop away again for distinguishing states or strings

continue….

Equivalences:
- A=B
- C=H
- D=G

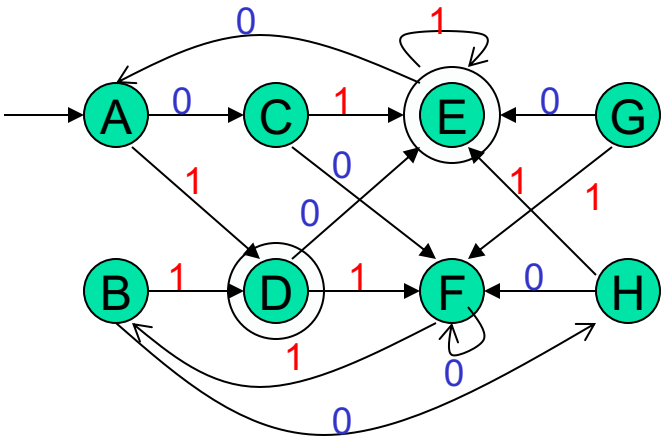
Table Filling Algorithm - step by step



Retrain only one copy for each equivalence set of states

Equivalences:
- A=B
- C=H
- D=G

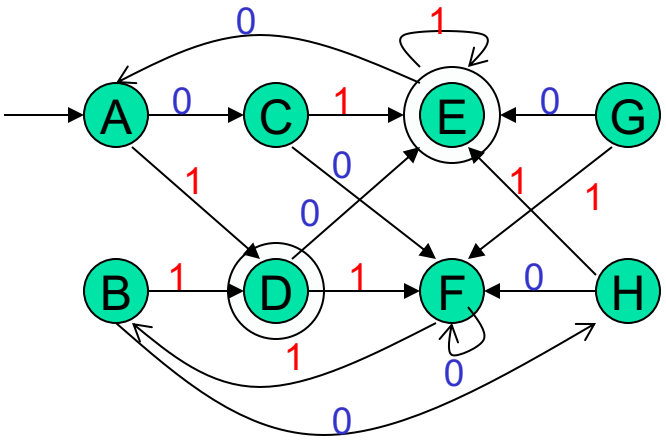
Table Filling Algorithm – special case



	A	B	C	D	E	F	G	H
A	=							
B		=						
C			=					
D				=				
E				?	=			
F						=		
G							=	
H								=

Q) What happens if the input DFA has more than one final state?
Can all final states initially be treated as equivalent to one another?

DFA Minimization by state equivalence method



0-Equivalence: {A,B,C,F,G,H} {D,E}

1-Equivalence {A,B,C,H} {F} {G} {D} {E}

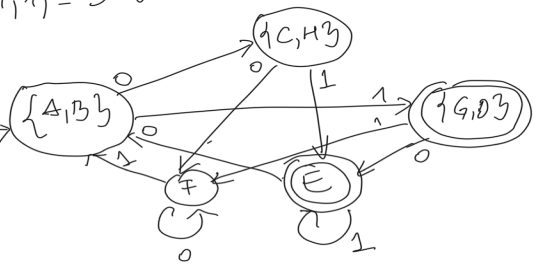
δ(A,0) = C
δ(B,0) = H
δ(A,1) = D
δ(B,1) = D
δ(C,0) = F
δ(C,1) = E
δ(G,0) = E
δ(F,0) = F
δ(F,1) = B
δ(G,1) = F
δ(H,0) = F = δ(C,0)
δ(H,1) = E = δ(C,1)
δ(D,0) = E
δ(E,0) = A

2-Equivalence

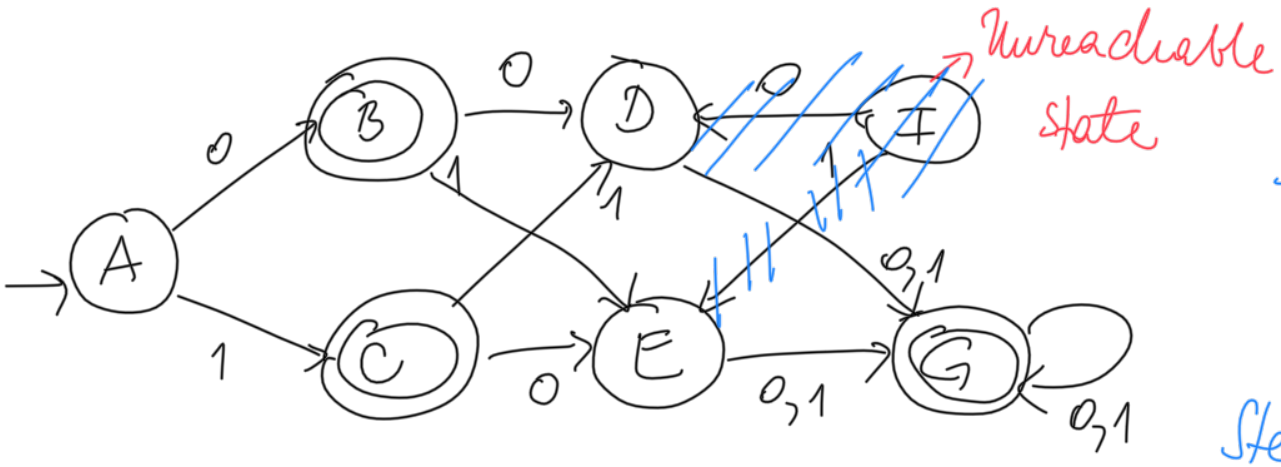
{A,B} {C,H} {F} {G,D} {E}

δ(A,0) = C
δ(H,0) = F
δ(C,0) = F
δ(F,0) = F
δ(G,0) = E
δ(H,1) = E
δ(C,1) = E
δ(F,1) = B
δ(D,1) = F
δ(G,1) = F

3-Equivalence
sets don't change!



DFA Minimization with unreacheable states



Step 1. Eliminate the unreach. State

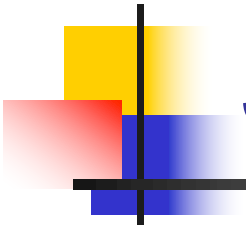
Step 2. Proceed with State equivalence or table filling methods.

How to minimize a DFA?

- Goal: Minimize the number of states in a DFA
- Algorithm:
  1. Eliminate states unreachable from the start state
  2. Identify and remove equivalent states
  3. Output the resultant DFA

Depth-first traversal from the start state

Table filling algorithm



Summary

- Simplification of DFAs
  - How to remove unreachable states?
  - How to identify and collapse equivalent states?
  - How to minimize a DFA?