State Reduction:

What is state Reduction?	The reduction of the number of flip-flops in a sequential circuit is referred to as the state reduction problem. When designing complex state machines, it often happens that there are equivalent states that can be grouped together to obtain a more efficient implementation
State Reduction Techniques	State-reduction algorithms are concerned with procedures for reducing the number of states in a state table, while keeping the external input-output requirements unchanged.
	Since (N) flip-flops produce (2^N) states, a reduction in the number of states may (or may not) result in a reduction in the number of flip-flops.
Equivalent Machines	An unpredictable outcome in reducing the number of flip-flops is the possibility that the equivalent circuit (with fewer flip-flops) may require more combinational gates.
Equivalent Machines	Assume that we have found a sequential circuit whose state diagram with a reduced number of states. We wish to compare it with the unreduced circuit state diagram. If identical input sequences are applied to the two circuits and identical outputs occur for all input sequences, then the two circuits are said to be equivalent (as far as the input-output is concerned) and one may be replaced by the other.

The problem of state reduction is to find ways of reducing the number of states in a sequential circuit without altering the input-output relationships.

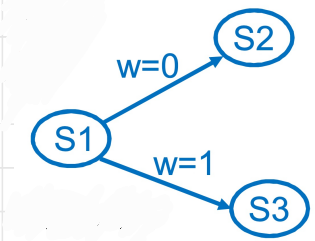
<

Equivalent States

- Two states S1 and S2 are called equivalent if and only if, for every possible input sequence, the same output will be produced regardless of whether S1 or S2 is the initial state.

State Reduction Techniques

- State Partitioning Technique
- Implication Table Technique



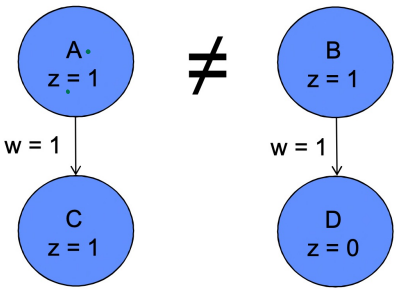
0- and 1-successors

- If input w=0 is applied to a state machine in state S1 and the result is that the machine moves to state S2, we say that S2 is a 0-successor of S1
- If input w=1 is applied to a state machine in state S1 and the result is that the machine moves to state S3, we say that S3 is a 1-successor of S1
- We will refer to successorsas k-successors, where k can be 0 or 1.

State minimization Basic idea:

Two states are not equivalent if they have different output values

Two states are not equivalent if at least one of their k-successors are not equivalent

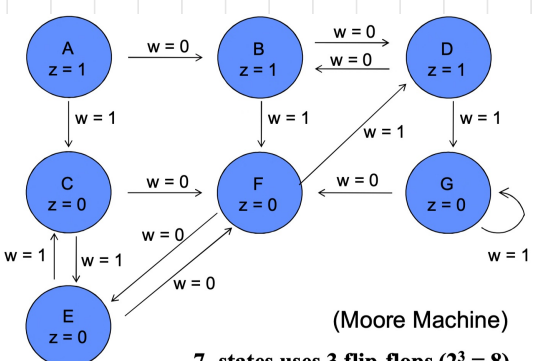


State Reduction by Partitioning

The minimization procedure first considers the states of a machine as a set and then breaks this set into partitions that are not equivalent.

- A partition consists of one or more blocks
  - each block contains states that may be equivalent
  - different blocks contain states that are definitely not equivalent

Example State Minimization:



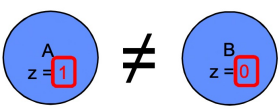
Present state	Next state		Output z
	w = 0	w = 1	
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

We start with one block containing all states

- P_1= (ABCDEFG)
- Stage 1:
  - Which states have different outputs?
  - ABD has output z = 1
  - CEFG have output z = 0
  - => P2= (ABD) (CEFG)

Present state	Next state $w=0$	Next state $w=1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

States A, B, D can therefore never be equivalent to any of the conditions C, E, F, G so they form different groups



P2= (ABD) (CEFG)

Stage 2:

– Which states have different k-successors?

Present state	Next state $w=0$	Next state $w=1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

- Block ABD

- 0-successor: A -> B , B-> D , D -> B (all transitions go to the same block)

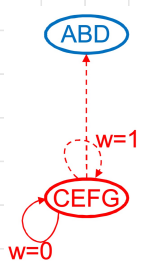
– 1-successor: A -> C , B -> F , D -> G (all transitions go to the same block)

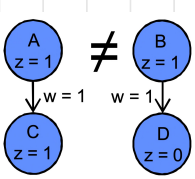
- Block CEFG

– 0-successor: C -> F , E -> F , F -> E , G -> F (all transitions goto . the same block)

– 1-successor: C -> E , E -> C , F -> D , G -> G (F -> D goes to another block)

=> P_3= (ABD) (CEG) (F)





P3= (ABD) (CEG) (F)

Step 3

– What states have different k-successors?

- Block ABD

– 0-successor: A-> B, B -> D, D -> B (all transitions go to the same block)

– 1-successor: A -> C, B -> F, D -> G (B -> F goes to another block)

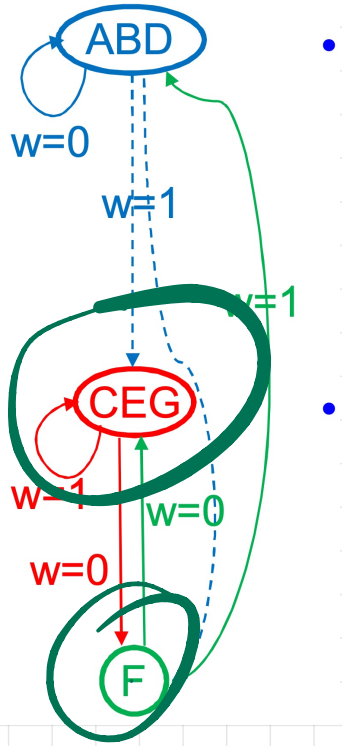
=> P4= (AD) (B) (CEG) (F)

- Block (CEG)

– 0-successor: C -> F-> E -> F, G -> F (all transitions go to the same block)

– 1-successor: C -> E, E -> C, G -> G (all transitions go to the same block)

=> P4= (AD) (B) (CEG) (F)



P4= (AD) (B) (CEG) (F)

Next partition P_5 becomes the same as P_4. Thus the procedure is finished.

– States in each block are equivalent

if they were not, their k-successors would have to be in different blocks

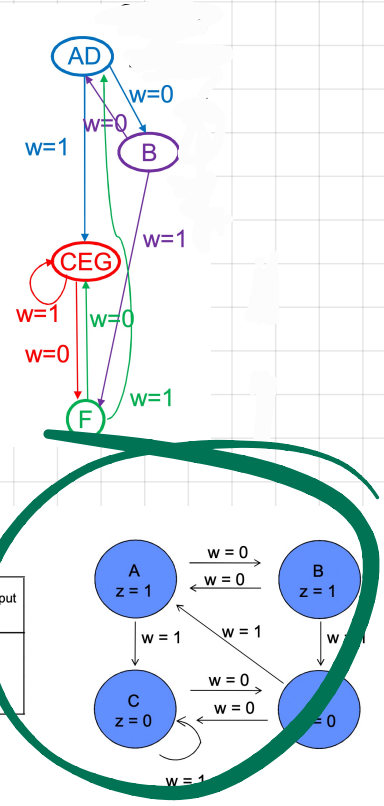
A becomes the representive of AD and C represents CEG.

P_4 = (AD)(B)(CEG)(F) = (A)(B)(C)(F)

Present state	Next state $w=0$	Next state $w=1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Present state	Next State $w=0$	Next State $w=1$	Output z
A	B	C	1
B	A	F	1
C	F	C	0
F	C	A	0

Final State Table



4 states needs 2 flip-flops ($2^2 = 4$).

Present state	Next state $w=0$	Next state $w=1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

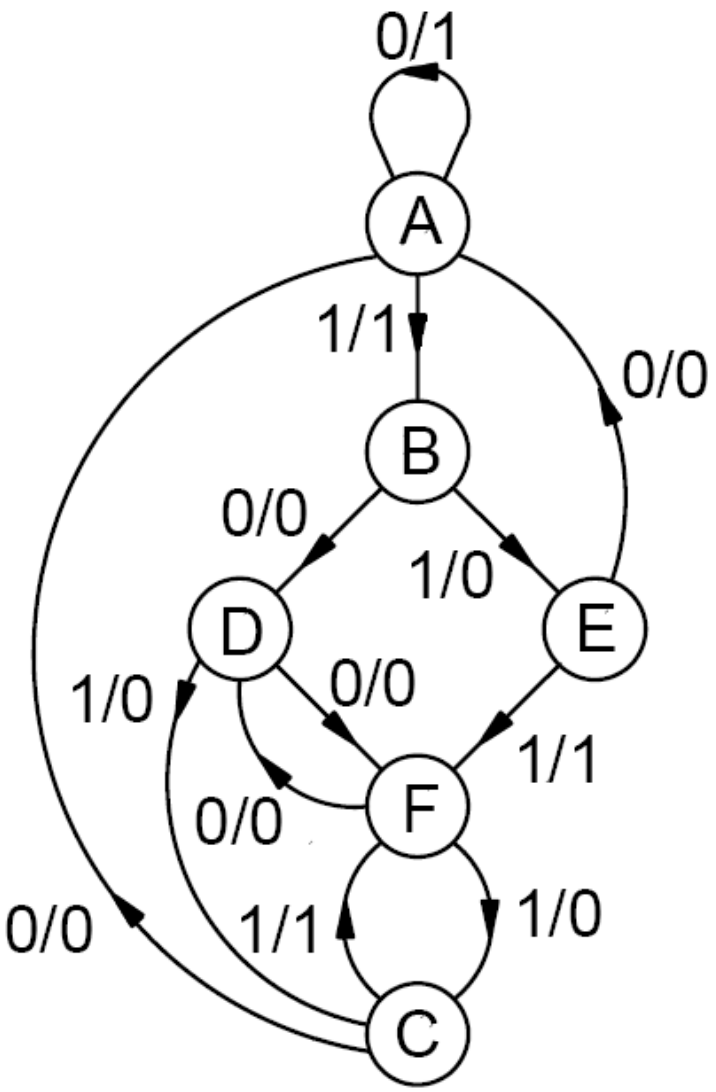
Present state	Next State $w=0$	Next State $w=1$	Output z
A	B	C	1
B	A	F	1
C	F	C	0
F	C	A	0

Final State Table

State Reduction IMPLICATION CHARTS

FSM with one input *X*, and one output *Z*:
(transitions are labelled *X/Z*)

State diagrams should never be used for state reduction. In more complex systems it is very easy to miss possible state reductions due to the complexity of the diagrams involved.



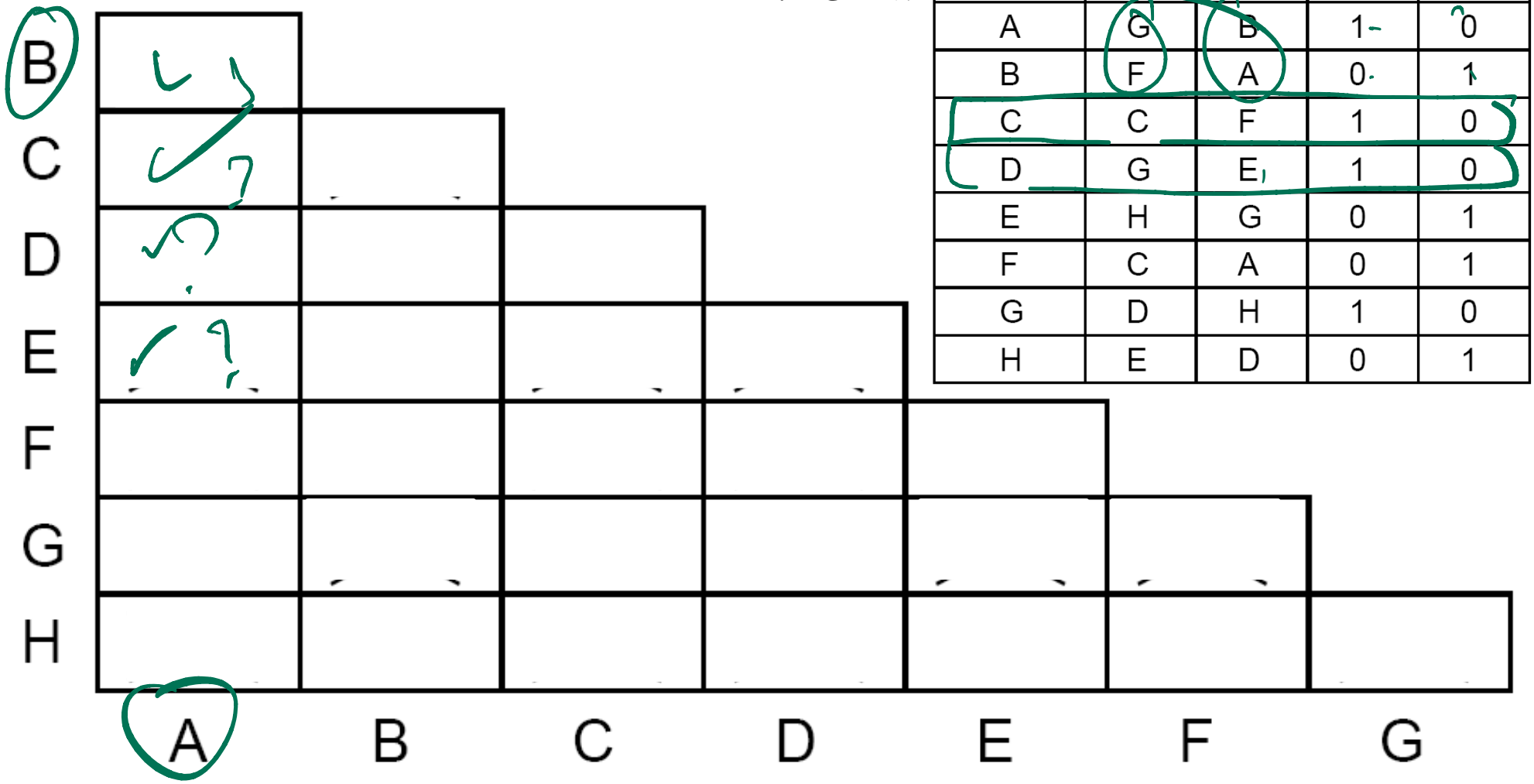
Implication Charts

Present state	Next state		Output Z	
	X=0	X=1	X=0	X=1
A	G	B	1	0
B	F	A	0	1
C	C	F	1	0
D	G	E	1	0
E	H	G	0	1
F	C	A	0	1
G	D	H	1	0
H	E	D	0	1

Implication Charts

1. Draw

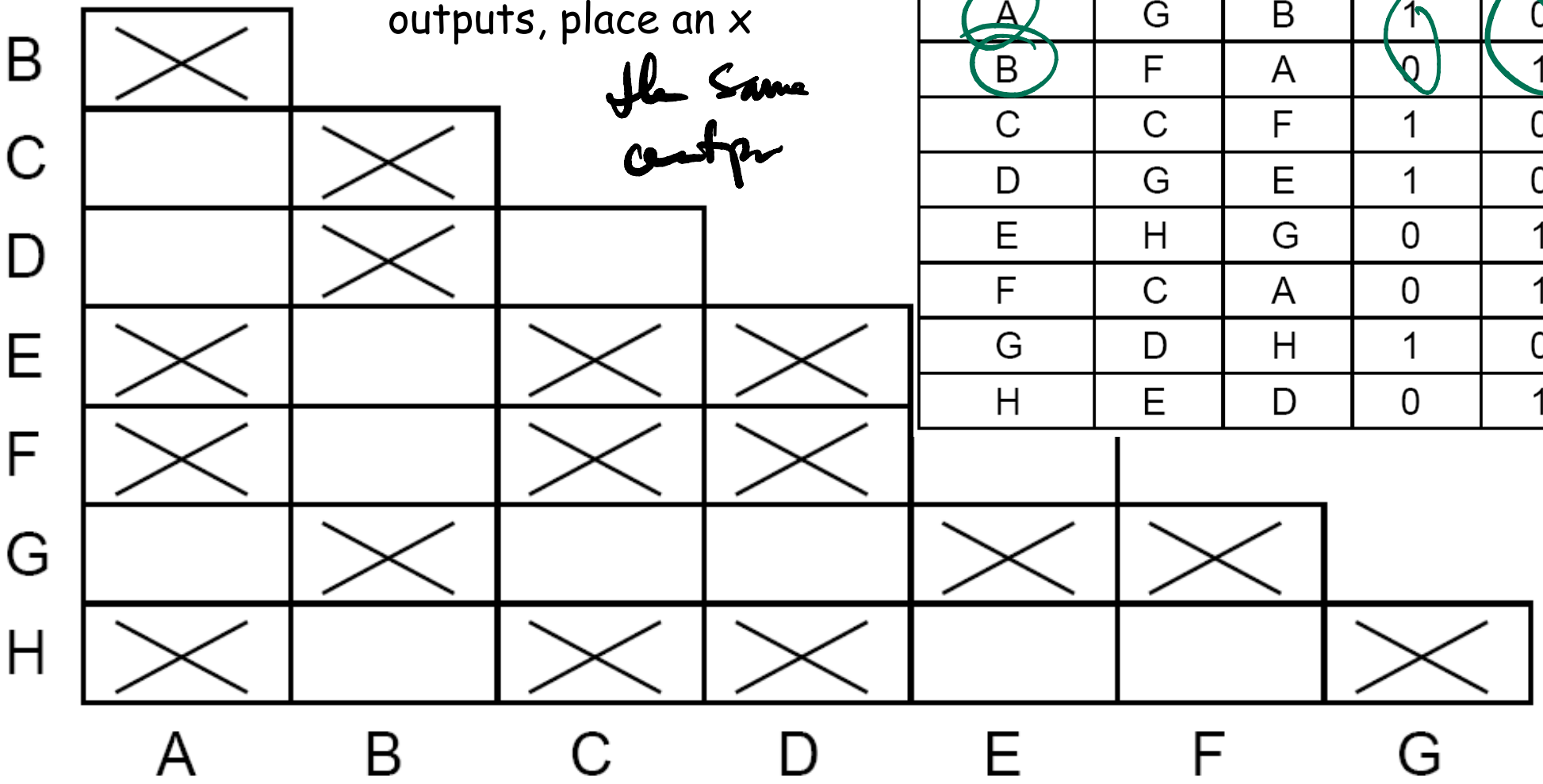
Present state	Next state		Output Z	
	X=0	X=1	X=0	X=1
A	G	B	1	0
B	F	A	0	1
C	C	F	1	0
D	G	E	1	0
E	H	G	0	1
F	C	A	0	1
G	D	H	1	0
H	E	D	0	1



Implication Charts

2. Where states do not have the same outputs, place an x

Present state	Next state		Output Z	
	X=0	X=1	X=0	X=1
A	G	B	1	0
B	F	A	0	1
C	C	F	1	0
D	G	E	1	0
E	H	G	0	1
F	C	A	0	1
G	D	H	1	0
H	E	D	0	1



Implication Charts

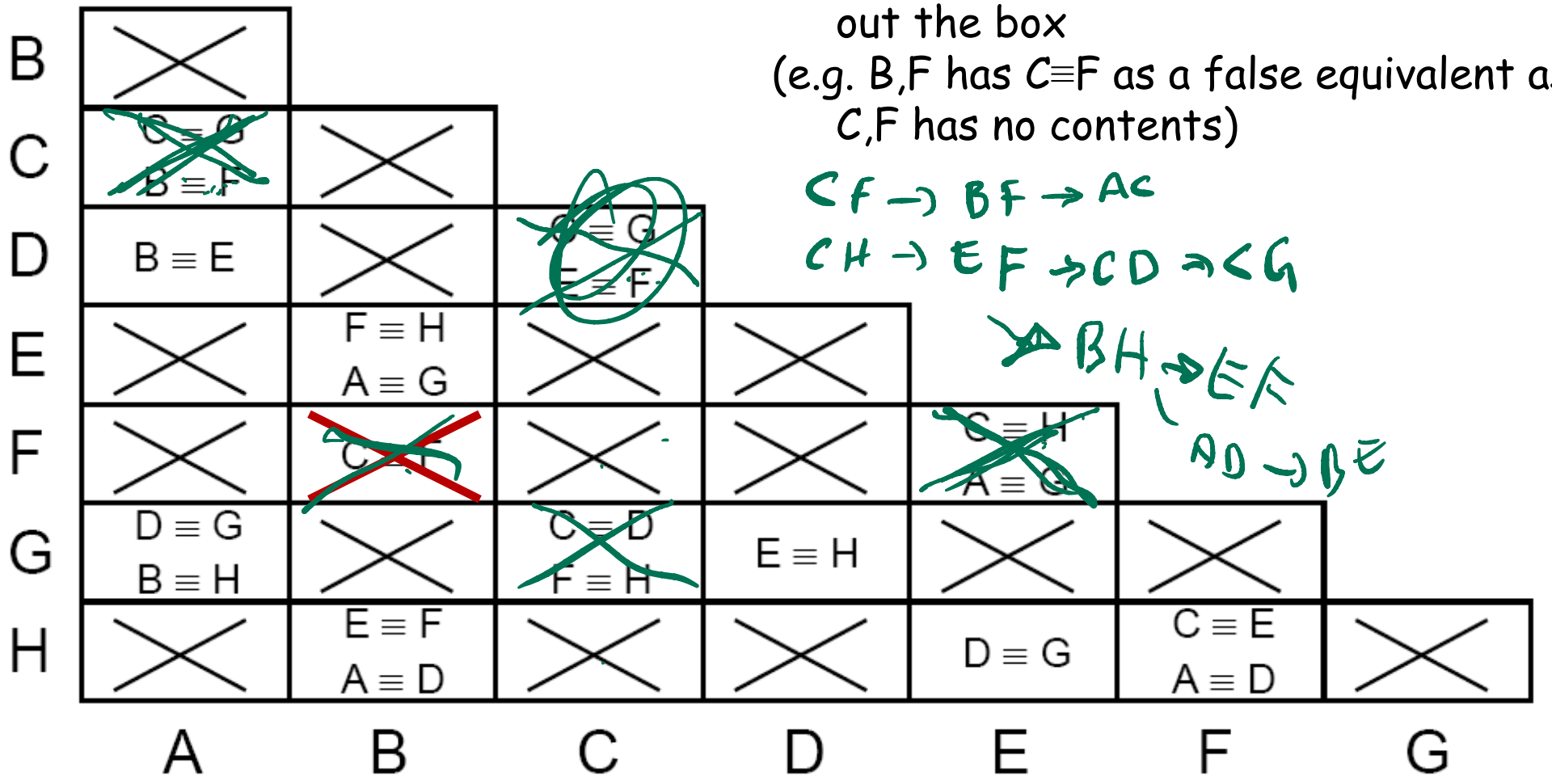
Present state	Next state		Output Z	
	X=0	X=1	X=0	X=1
A	G	B	1	0
B	F	A	0	1
C	G	F	1	0
D	G	E	1	0
E	H	G	0	1
F	C	A	0	1
G	D	H	1	0
H	E	D	0	1

3. Where states have the same output, put equalities in the box

A	G	B	1	0
C	C	F	1	0

	A	B	C	D	E	F	G
B	×						
C	C ≡ G B ≡ F	×					
D	B ≡ E	×	C ≡ G E ≡ F				
E	×	F ≡ H A ≡ G	×	×			
F	×	C ≡ F	×	×	C ≡ H A ≡ G		
G	D ≡ G B ≡ H	×	C ≡ D F ≡ H	E ≡ H	×	×	
H	×	E ≡ F A ≡ D	×	×	D ≡ G	C ≡ E A ≡ D	×

Implication Charts



4. Where equivalents are false, cross out the box

(e.g. B,F has C≡F as a false equivalent as C,F has no contents)

CF → BF → AC

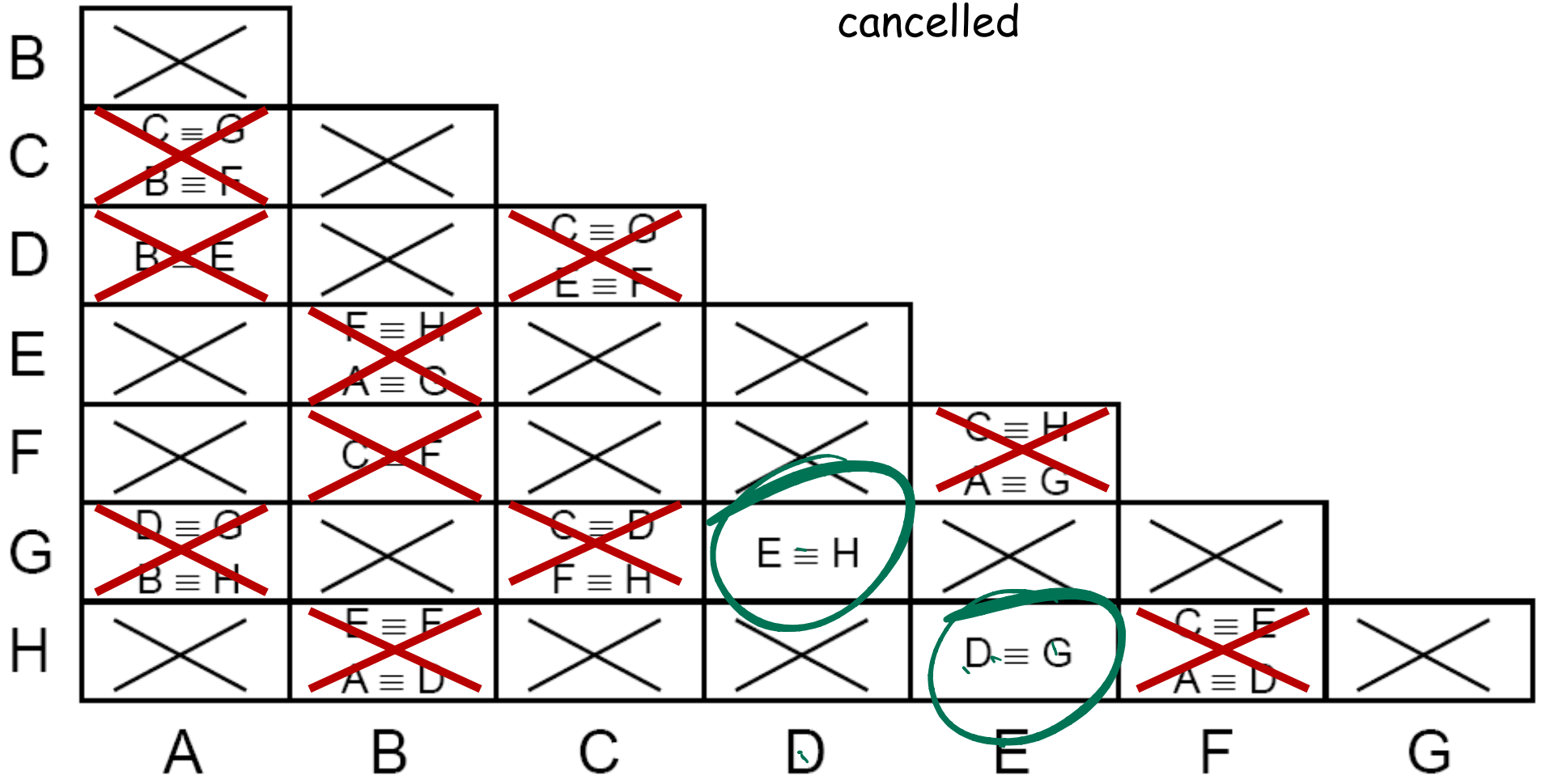
CH → EF → CD → CG

→ BH → EF

AD → BE

Implication Charts

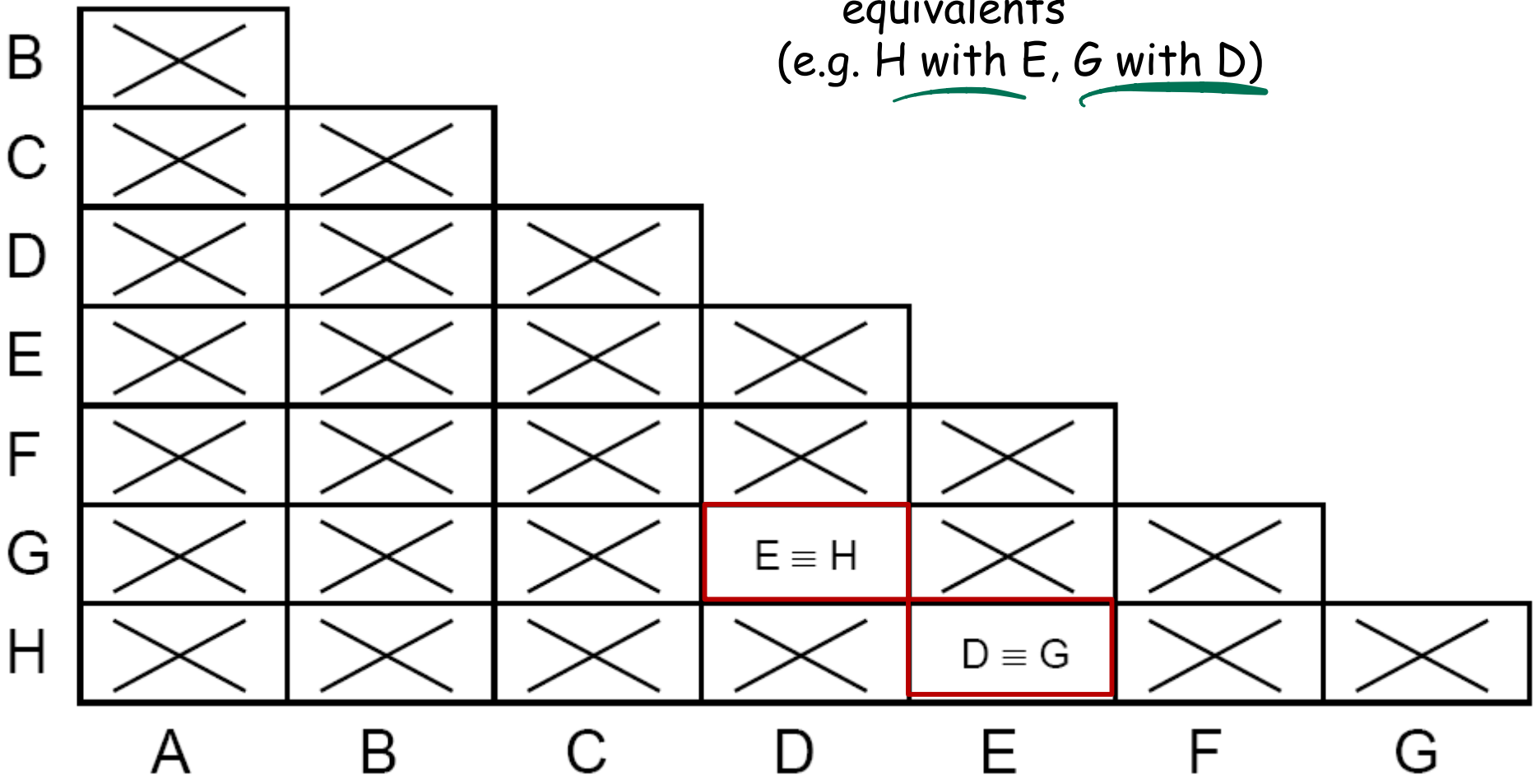
5. Repeat until no more can be cancelled



Implication Charts

6. What is left can be replaced with equivalents

(e.g. H with E, G with D)



Implication Charts

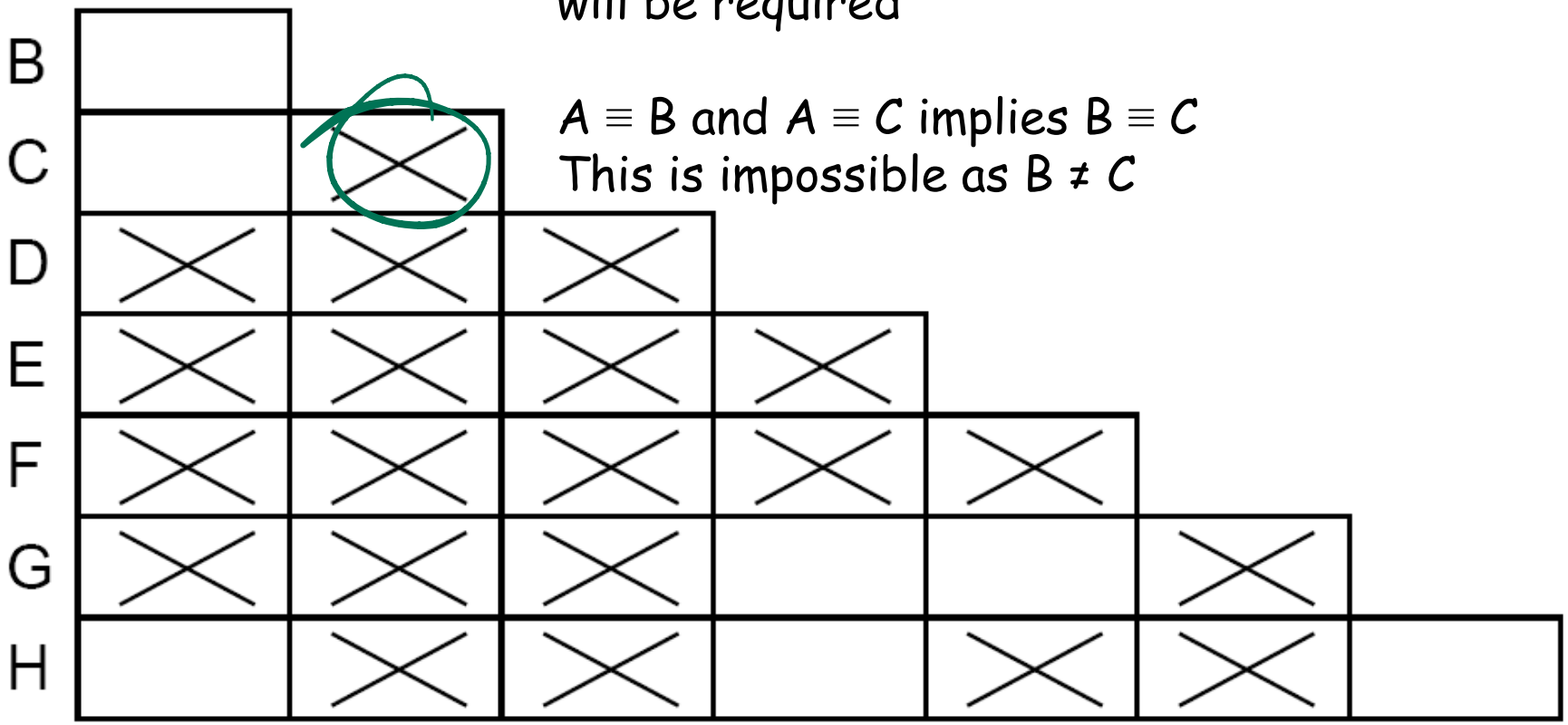
7. Draw reduced state table

Present state	Next state		Output Z	
	X=0	X=1	X=0	X=1
A	D	B	1	0
B	F	A	0	1
C	C	F	1	0
D	D	E	1	0
E	E	D	0	1
F	C	A	0	1

Merger Diagrams

Sometimes implication charts show many possible equivalences. If this occurs a Merger diagram will be required

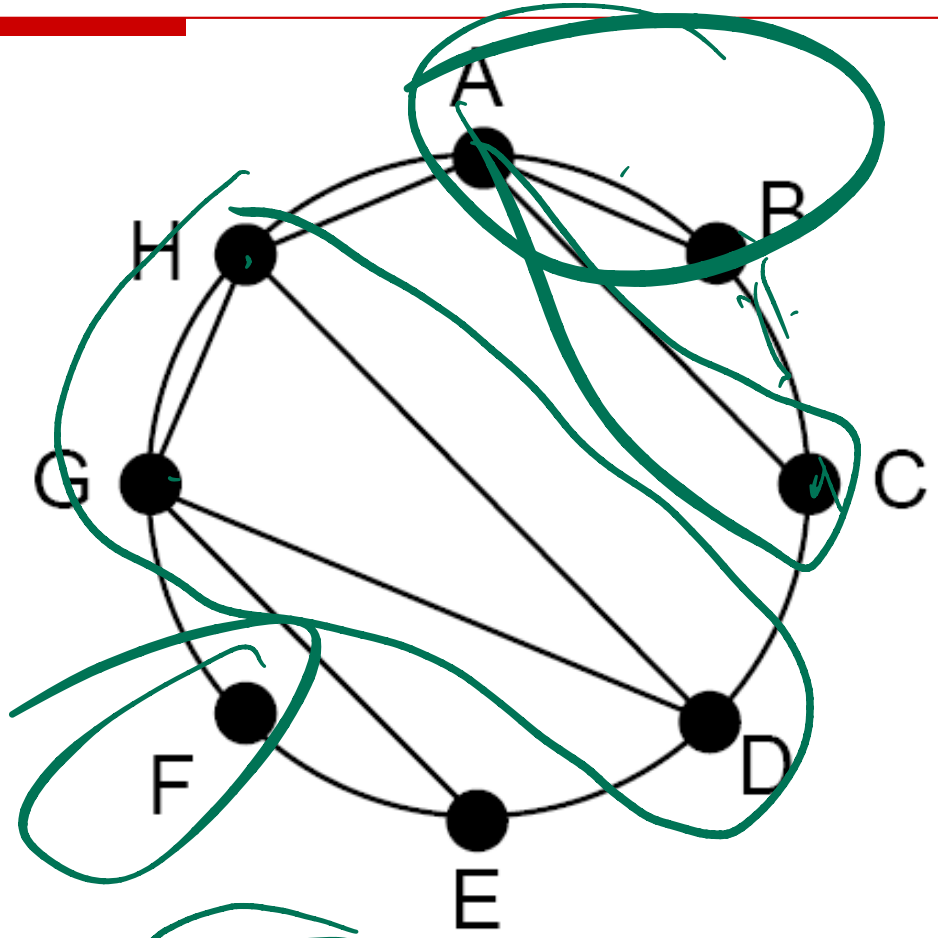
$A \equiv B$ and $A \equiv C$ implies $B \equiv C$
This is impossible as $B \neq C$



A≡B A≡C A≡H D≡G D≡H E≡G G≡H

Merger Diagrams

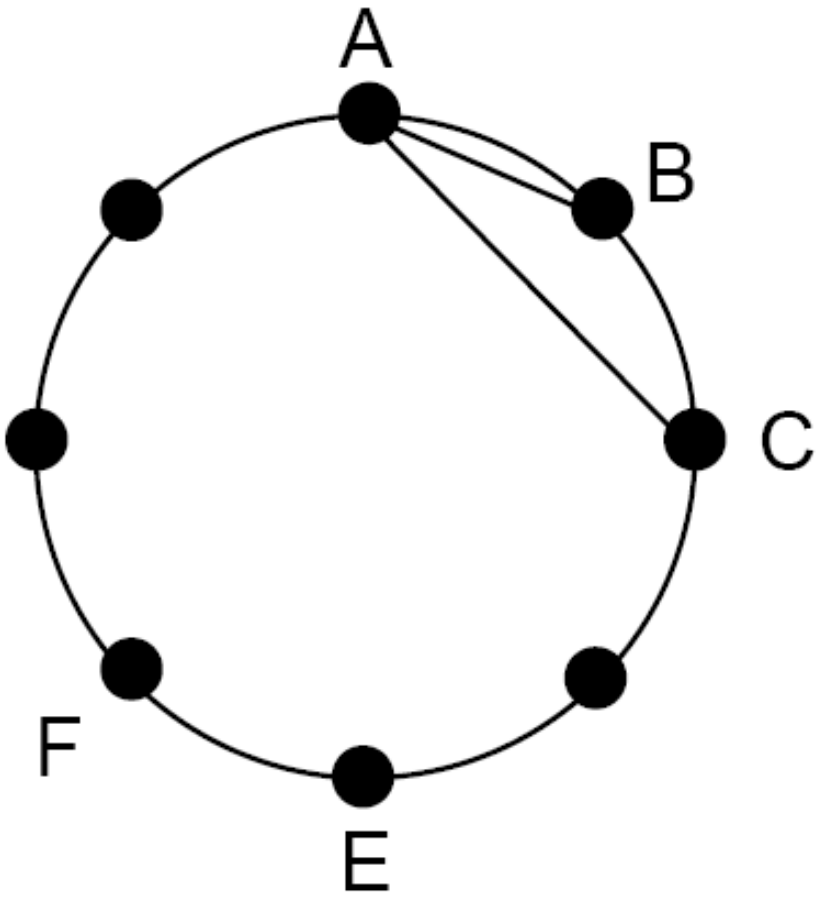
- Lines are placed on the merger diagram with regards to all possible equivalences
- Polygons formed by these lines with all their sides displayed are to be found.
- The triangle GDH determines that G≡D≡H must be true



G≡H≡D
A ≡ B
A ≡ C
B ≠ C

Merger Diagrams

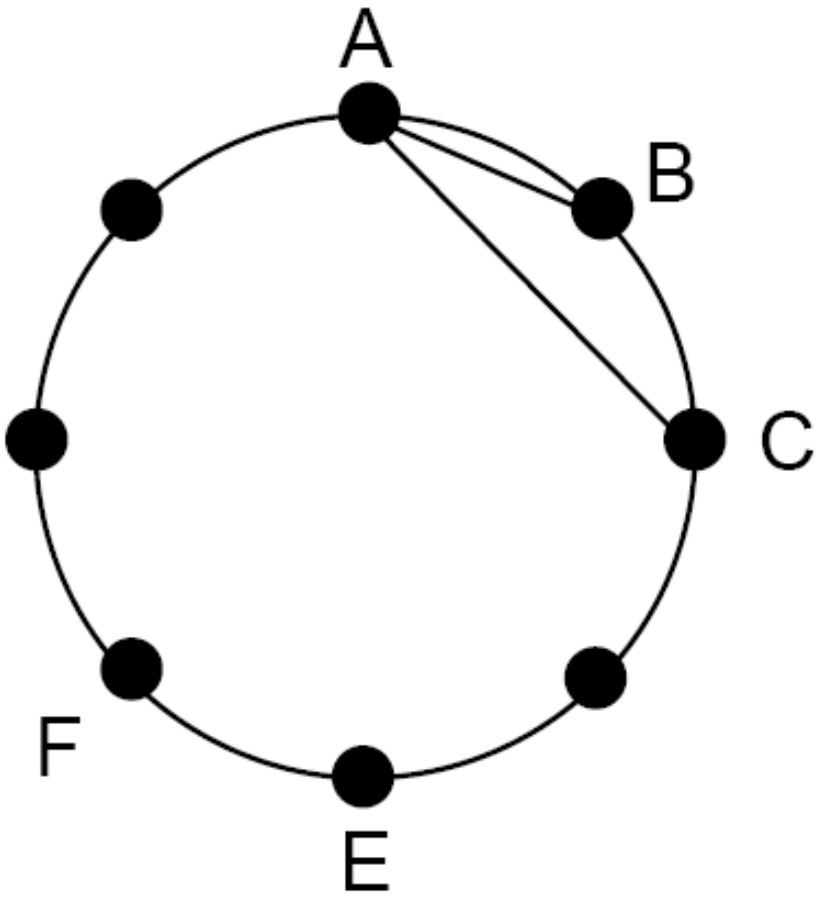
- We are left with an arbitrary decision between A≡B and A≡C



$A \equiv B$
$A \equiv C$
$B \neq C$

Merger Diagrams

- We are left with an arbitrary decision between A≡B and A≡C



$A \equiv B$
$A \equiv C$
$B \neq C$