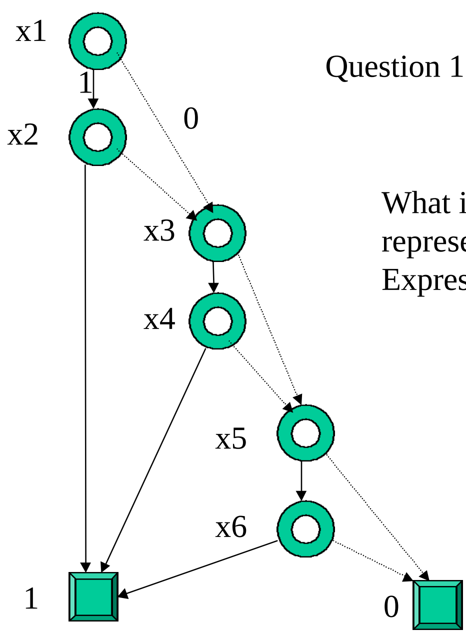
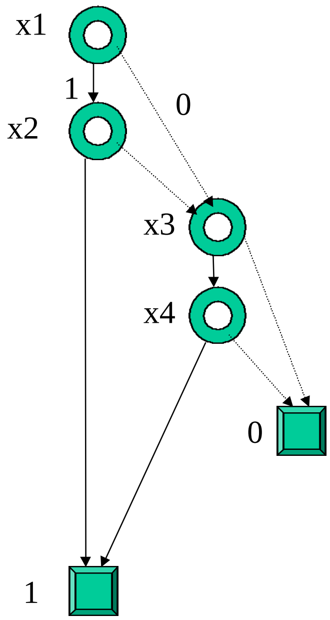
Midterm COM3220

- Open book/open notes
- Tuesday, April 28, 6pm - 7.30 pm



Question 1: 5 points

What is the Boolean function represented by this BDD?
Express it in terms of **and** and **or** only.

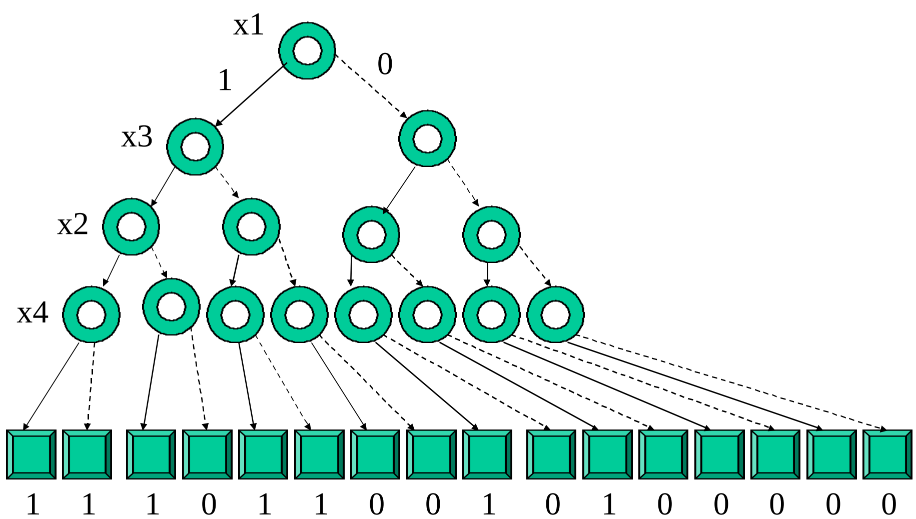


Question 2: 15 points

Consider the BDD on the left.
It uses the variable ordering
x1, x2, x3, x4.
Construct the BDD for the variable ordering x1, x3, x2, x4.

5/19/98 Midterm/COM 3220 3

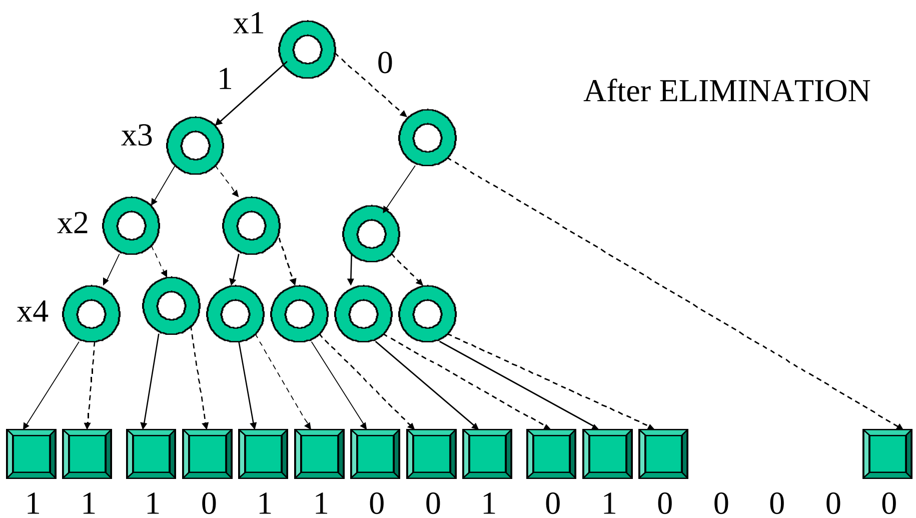
(x1 and x2) or (x3 and x4)



5/19/98 Midterm/COM 3220 4

(x1 and x2) or (x3 and x4)

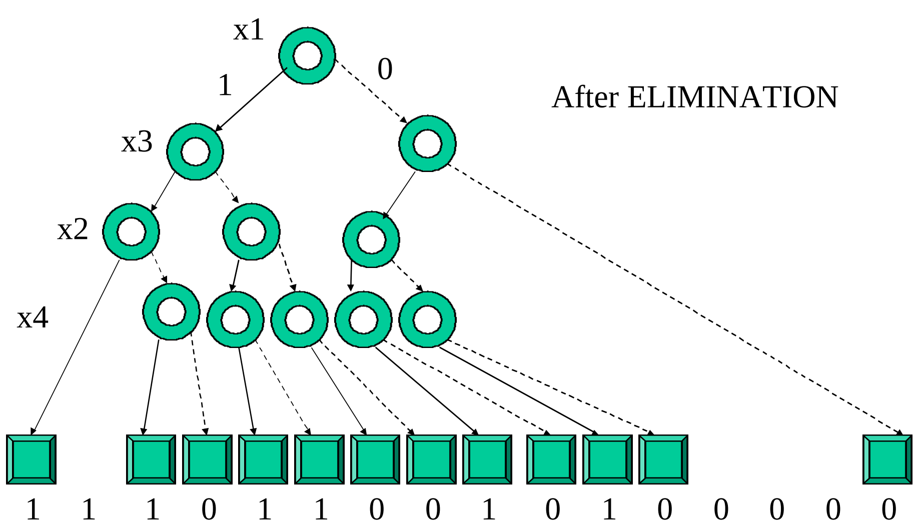
After ELIMINATION



5/19/98 Midterm/COM 3220 5

(x1 and x2) or (x3 and x4)

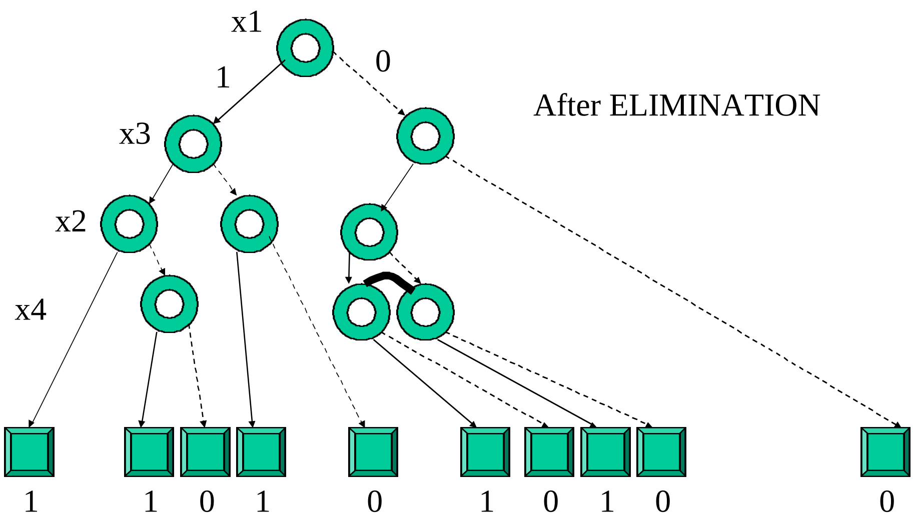
After ELIMINATION



5/19/98 Midterm/COM 3220 6

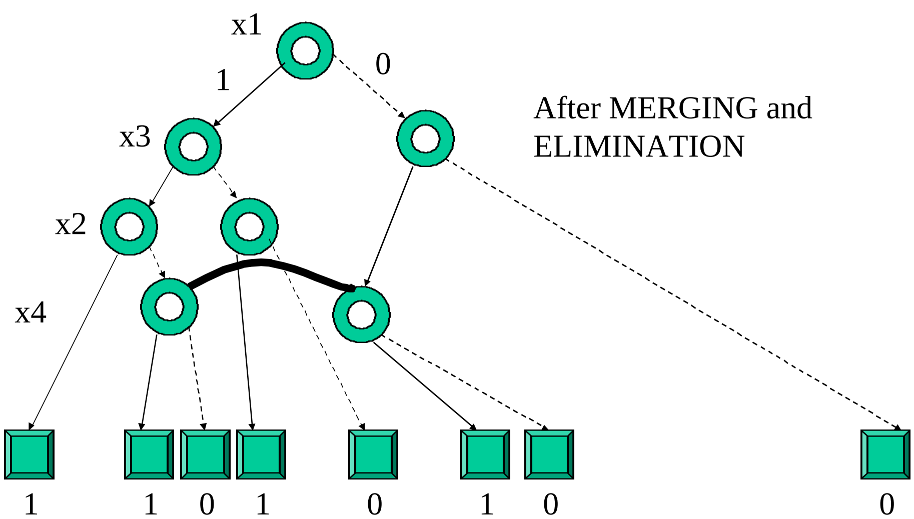
(x1 and x2) or (x3 and x4)

After ELIMINATION



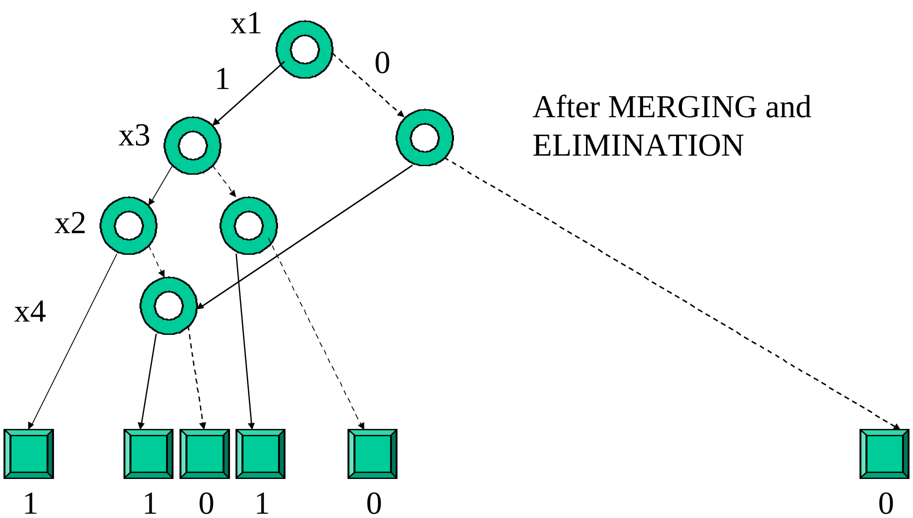
(x1 and x2) or (x3 and x4)

After MERGING and ELIMINATION



(x1 and x2) or (x3 and x4)

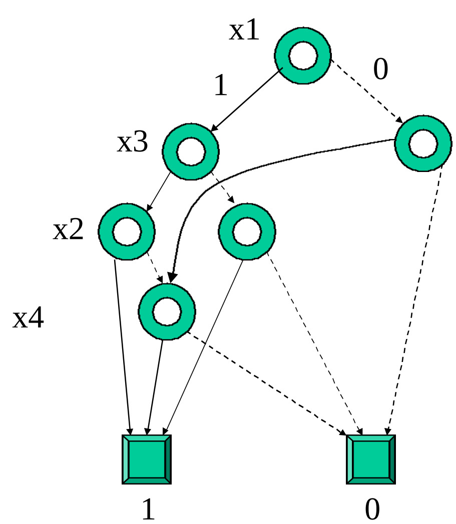
After MERGING and ELIMINATION



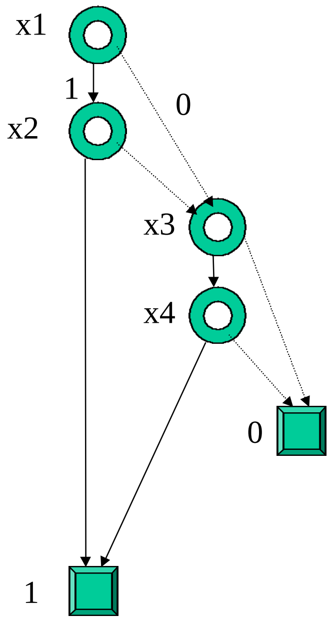
5/19/98 Midterm/COM 3220 9

(x1 and x2) or (x3 and x4)

After TERMINAL



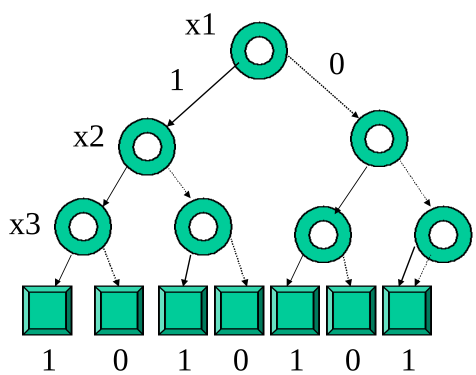
5/19/98 Midterm/COM 3220 10



Question 2: 15 points

Consider the BDD on the left.
It uses the variable ordering
x1, x2, x3, x4.
Construct the BDD for the variable ordering x1, x3, x2, x4.

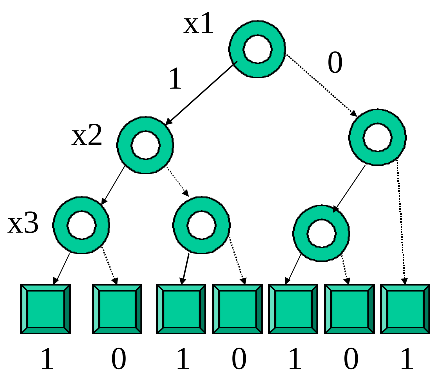
Compare with x1,x2,x3,x4 ordering: 4 versus 6 interior nodes



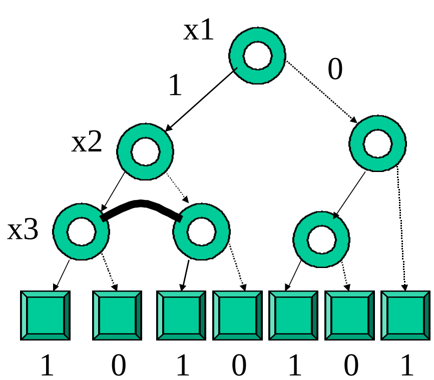
Question 3: 10 points

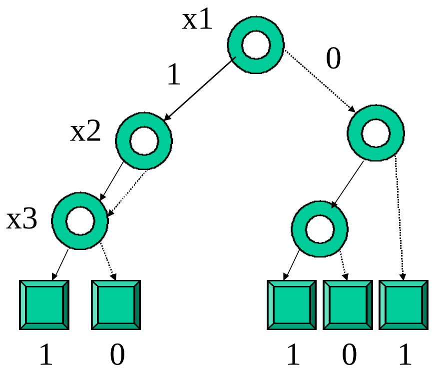
The BDD to the left is not reduced. Bring it to reduced form.

After ELIMINATION

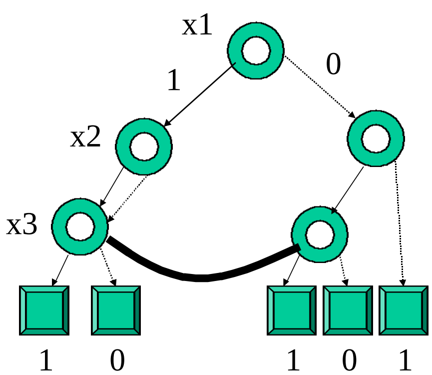


Prepare for MERGING

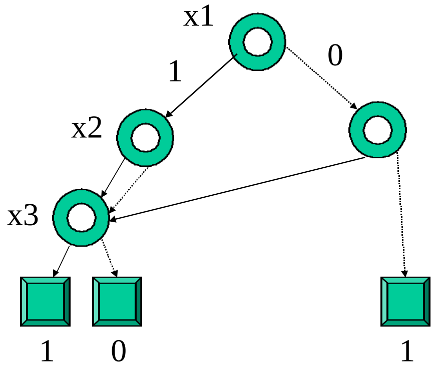




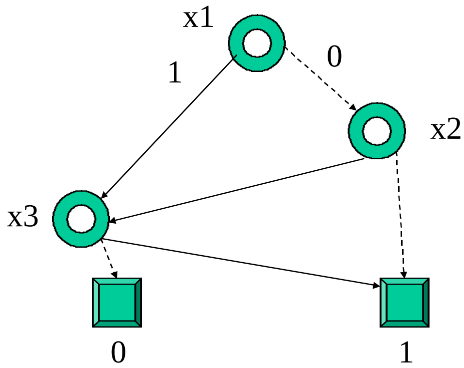
After MERGING



Prepare for MERGING

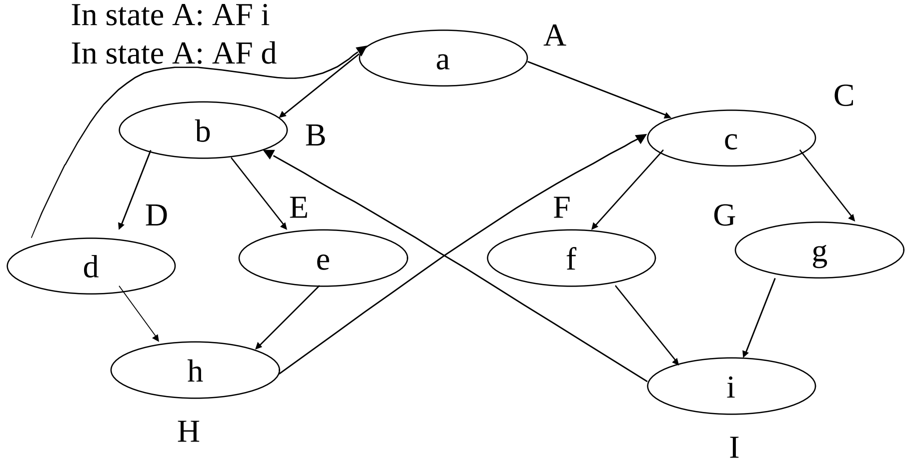


After MERGING



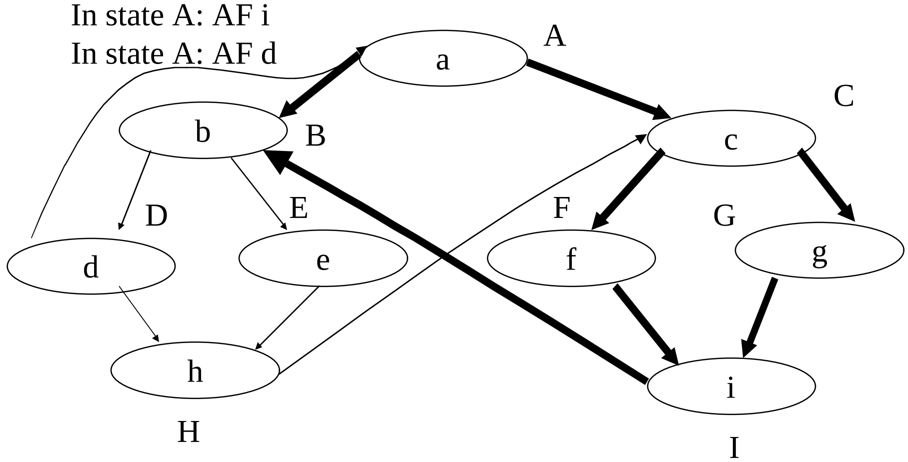
After ELIMINATION
and TERMINAL

Question 4: 15 points
Which of the following formulas hold? Explain!
In state A: AF b
In state A: AF i
In state A: AF d



5/19/98 Midterm/COM 3220 19

Question 4: 15 points
Which of the following formulas hold? Explain!
In state A: AF b : TRUE
In state A: AF i
In state A: AF d



5/19/98 Midterm/COM 3220 20

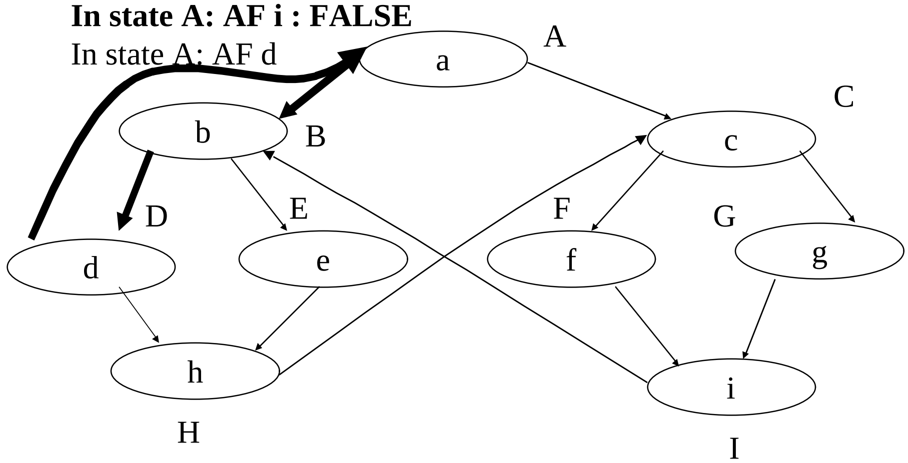
Question 4: 15 points

Which of the following formulas holds Explain!

In state A: AF b

In state A: AF i : FALSE

In state A: AF d



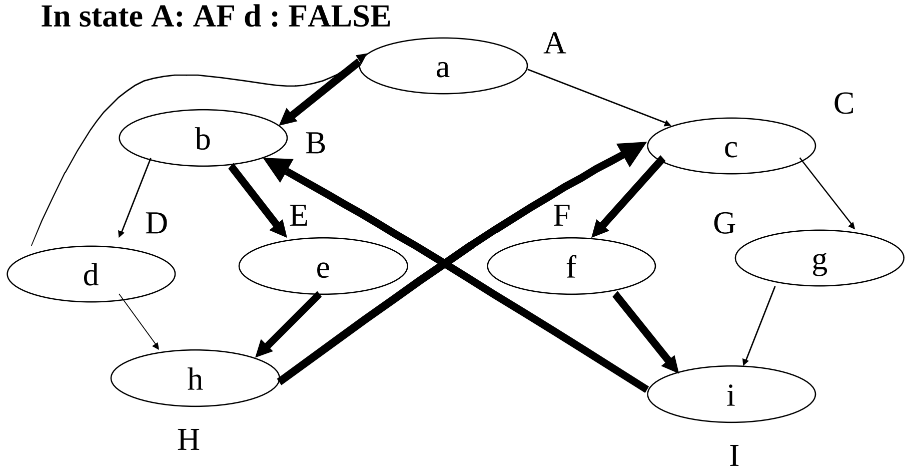
Question 4: 15 points

Which of the following formulas hold? Explain!

In state A: AF b

In state A: AF i

In state A: AF d : FALSE



Some background

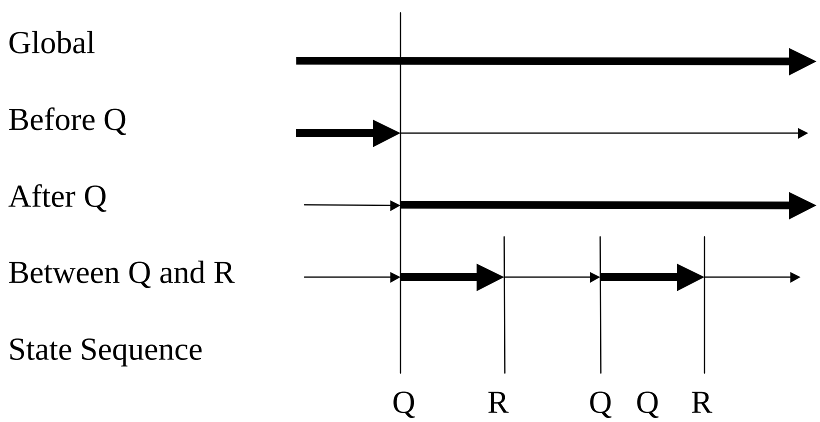
- The following questions involve a *scope* which is the extent of a program's execution over which a formula must hold. There are five basic kinds of scopes: global, before, after, between, after-until.

Some background

- *scope*
  - global (the entire program execution),
  - before (the execution up to a given state),
  - after (the execution after a given state)
  - between (any part of the execution from one given state to another given state)
  - after-until (like between even if the second state does not occur)

Some background

- A scope itself should be interpreted as optional; if the scope delimiters are not present in an execution then the specification will be true.



Four Formula Scopes

Question 5: Absence
2 points per unknown

- The purpose of the following CTL formulas is to describe a portion of a system's execution that is free of certain states.
- In the following you will have to find unknowns Y1, Y2, … Those unknowns you should replace by identifiers and/or symbols to make the formula correct. Example:
  Y1 + 3 = 8. Solution: Y1 = 5

CTL formulas for Absence

- P is false
  - Globally: Y1 Y2(!P)
  - Before R: A[!Y3 **U** (Y4 or AG(!R))]
  - After Q: Y5 G(Q Y6 AG(!P))
  - Between Q and R: Y7 G(Q=>A[!P **U** (Y8 or Y9 Y10 (!R))])
  - The next intentionally does not contain unknowns
  - After Q until R: AG(Q=>!E[!R **U** (P and !R)])

CTL formulas for Absence

- P is false
  - **Globally: Y1 Y2(!P)   AG(!P)**
  - Before R: A[!Y3 **U** (Y4 or AG(!R))]
  - After Q: Y5 G(Q Y6 AG(!P))
  - Between Q and R: Y7 G(Q=>A[!P **U** (Y8 or Y9 Y10 (!R))])
  - The next intentionally does not contain unknowns
  - After Q until R: AG(Q=>!E[!R **U** (P and !R)])

CTL formulas for Absence

- P is false
  - **Before R: A[!Y3 U (Y4 or AG(!R))]**
  - **Before R: A[!P   U (R or AG(!R))]**
  - P is false until R holds or until R will never hold

CTL formulas for Absence

- P is false
  - **After Q: Y5 G(Q Y6 AG(!P))**
  - **After Q: A G(Q => AG(!P))**
  - **For all paths the following condition holds at every state: If Q holds at a state then for all paths from that state !P holds globally.**

CTL formulas for Absence

- P is false
  - **Between Q and R: Y7 G(Q=>A[!P U (Y8 or Y9 Y10 (!R))])**
  - **Between Q and R: A G(Q=>A[!P U (R or A G (!R))])**
  - **Globally, if Q holds at a state s then P is false until R holds or R is false globally from s.**

Question 6: Response
5 points per unknown

- The purpose of the following CTL formulas is to describe cause-effect relationships between a pair of states. An occurrence of the first, the cause, must be followed by an occurrence of the second, the effect, within a defined portion of a system's execution.
- Find the three UNKNOWNs

CTL formulas for Response

- S responds to P: (P is the cause, S the effect)
  - UNKNOWN2 Q: AG(Q=>AG(P=>AF(S)))
  - UNKNOWN1: AG(P=>AF(S))
  - UNKNOWN3 R: A[(P=>A[!R **U** ((S and !R) or AG(!R))]) **U** (R or AG(!R))]
- Note: the three UNKNOWN are part of the explanation of the CTL formula. Each unknown is one word. Explain the formula for UNKNOWN3.

CTL formulas for Response

- S responds to P: (P is the cause, S the effect)
  - **UNKNOWN2 Q: AG(Q=>AG(P=>AF(S)))**
  - **AFTER Q: AG(Q=>AG(P=>AF(S))) : Globally, if Q holds, then if P holds, eventually S will hold.**
  - UNKNOWN1: AG(P=>AF(S))
  - UNKNOWN3 R: A[(P=>A[!R **U** ((S and !R) or AG(!R))]) **U** (R or AG(!R))]

CTL formulas for Response

- S responds to P: (P is the cause, S the effect)
  - UNKNOWN2 Q: AG(Q=>AG(P=>AF(S)))
  - **UNKNOWN1 : AG(P=>AF(S))**
  - **GLOBALLY : AG(P=>AF(S)): Globally, if P holds then S will eventually hold.**

CTL formulas for Response

- S responds to P: (P is the cause, S the effect)
  - **UNKNOWN3 R: A[(P=>A[!R U ((S and !R) or AG(!R))]) U (R or AG(!R))]**
  - **BEFORE R: …**
  - **Amazing how complex it is to express BEFORE.**
  - **Until R holds or R never holds, if P holds then for all paths until (S and !R) holds or R never holds not R holds.**

Question 7: Properties of assignment /10 points

- Assume that the property {q*y+x=a} holds before we execute the two assignment statements: x:=x-y; q:=q+1; Does the property still hold after execution of the two assignment statements? Explain your answer.
- **Solution: Substitute: (q+1)y+x-y=qy+x. Therefore, qy+x=a still holds.**

Question 8: Blackbox Testing: Topological Sorting

- Assume you have to test a program written for the following specification: Given a directed acyclic graph G=(V,E) with n vertices, label the vertices from 1 to n such that, if v is labeled k, then all vertices that can be reached from v by a directed path are labeled with labels >k.

What to do.

- Write test requirements and test specifications for this testing task. 30 points
- Outline an algorithm for implementing the specification. Any implications on your test requirements? 10 points

Test requirements

- Graph has zero vertices
- Graph has one vertex
- Graph has >1 vertices
- Graph has zero edges
- Graph has one edge
- Graph has >1 edges

Test requirements

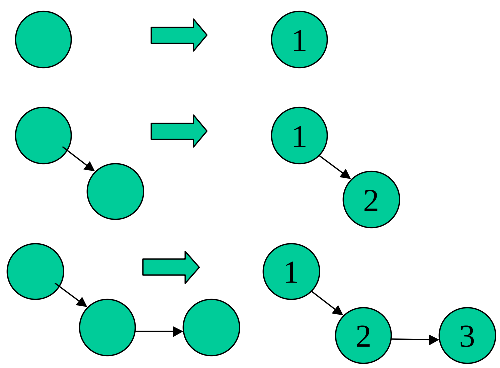
- Graph has a cycle: expect error
- Graph is not connected

Algorithm

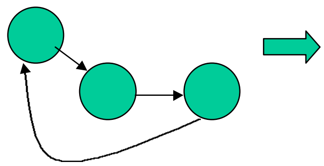
- Compute number of predecessors of each vertex.
- While there is a node with 0 predecessors
  - put such a node into topological order and delete node and all outgoing edges. Update predecessor counts.
- If there are nodes left in the graph, there must be a cycle.

Test specifications

- Empty graph => output: empty graph



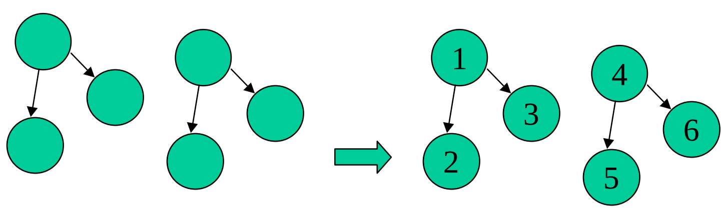
Test specifications



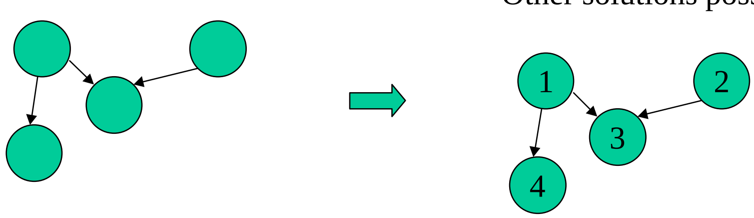
Error: graph cyclic

5/19/98 Midterm/COM 3220 45

Test specifications



Other solutions possible



5/19/98 Midterm/COM 3220 46

Testing whether topological

- i < j: test whether there is a path from i to j. Use Warshall's algorithm with adjacency matrix representation:
- for y:=1 to V do for x:=1 to V do
- if a[x,y] then
- for j:=1 to V do
- if a[y,j] then a[x,j]:= true;