CS477 Formal Software Development Methods

Elsa L Gunter
2112 SC, UIUC
egunter@illinois.edu
http://courses.engr.illinois.edu/cs477

How to build a Reduced Ordered BDD

- Done by recursion on the structure of the formula
- Order propositional variables
- For each variable, create a three node tree corresponding to (if Var then true else false)
- For formula not P, build ROBDD from ROBDD for P by flipping values in the leaves

How to build a Reduced Ordered BDD

- For P <op> Q
  - Fill in each branch of BDDs for P and Q with all nodes appearing in either BDD, not already on branch
  - For each ordered sequence of values (done in order of variables, greatest to least, false before true) build the branch in partial BDD for (P <op> Q)
  - Value in end leaf is v1 <op> v2 where v1 is value at end of branch in BDD for P, v2 for branch in Q

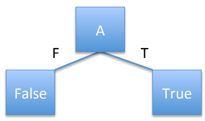
How to build a Reduced Ordered BDD

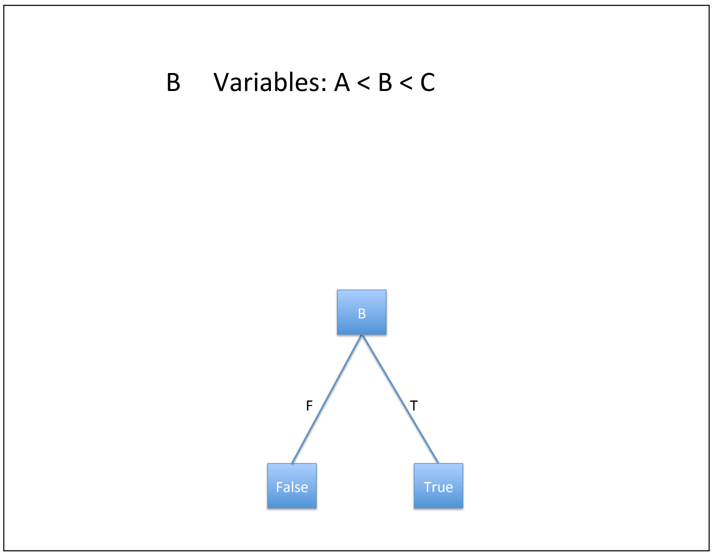
- When you add a new node
  - If the outedges point to same node, ghost it.
  - If there exists another node with the same label whose outedges of the a given label point the same nodes as its outedges of the given label, remove the new node and move all edges pointing to it to the other node
- Final step when returning final result
  - For each ghost node, move all edges pointing to it to point to what it points to, and remove it

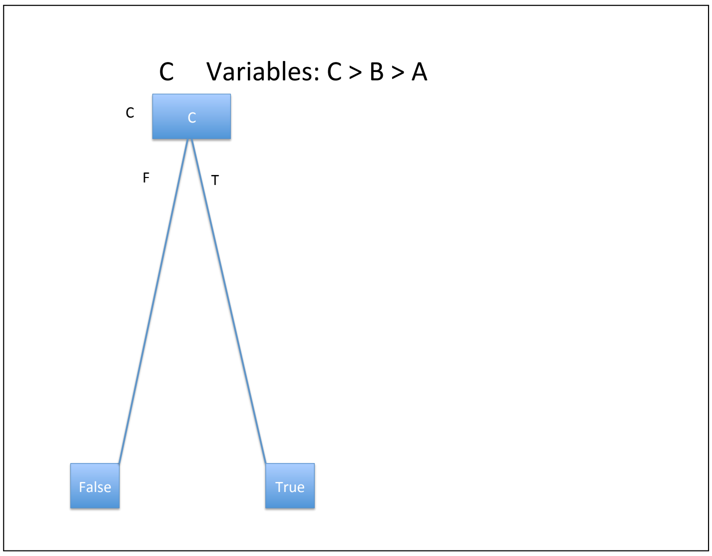
Example

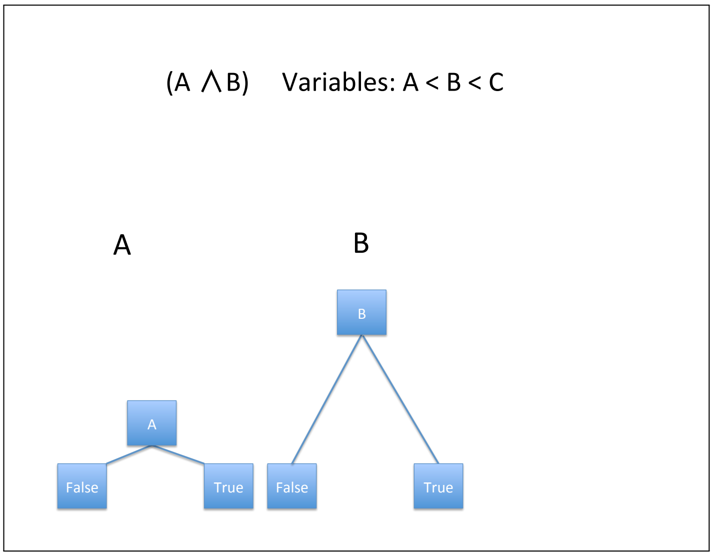
- Find ROBDD for $(A \wedge B) \vee (\text{not } C)$

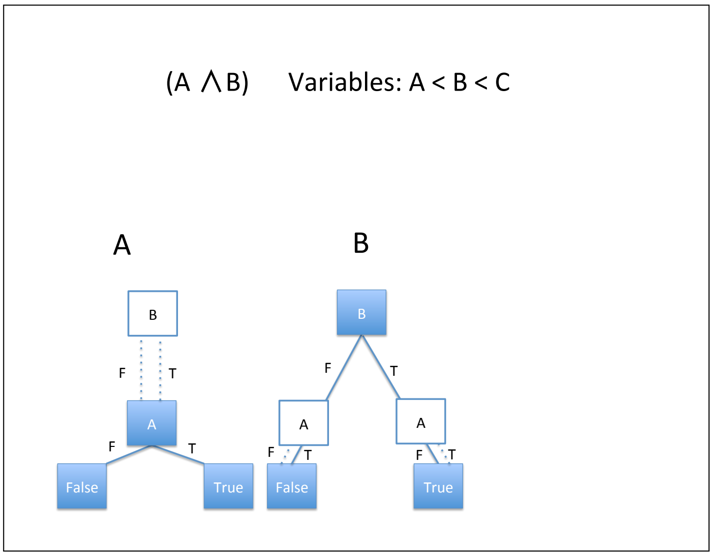
A   Variables: A < B < C

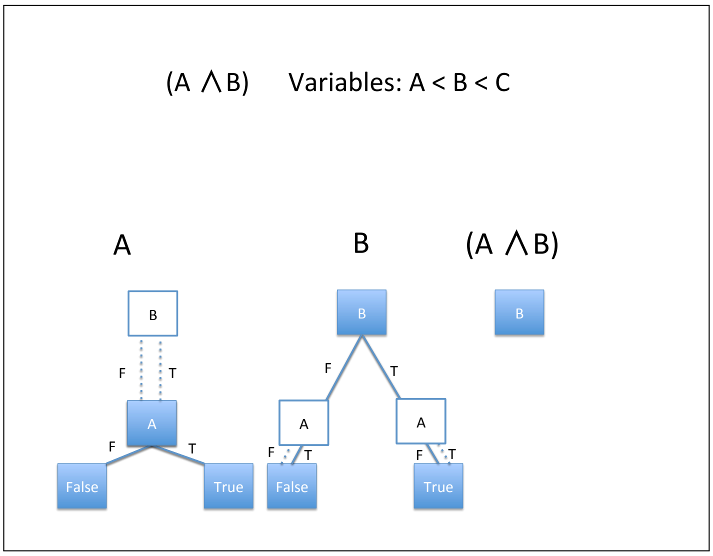


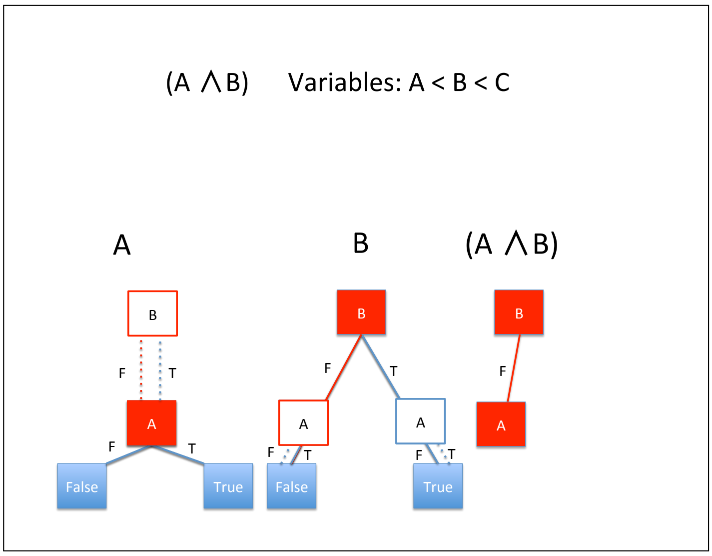

B
Variables: A < B < C
B
F
T
False
True

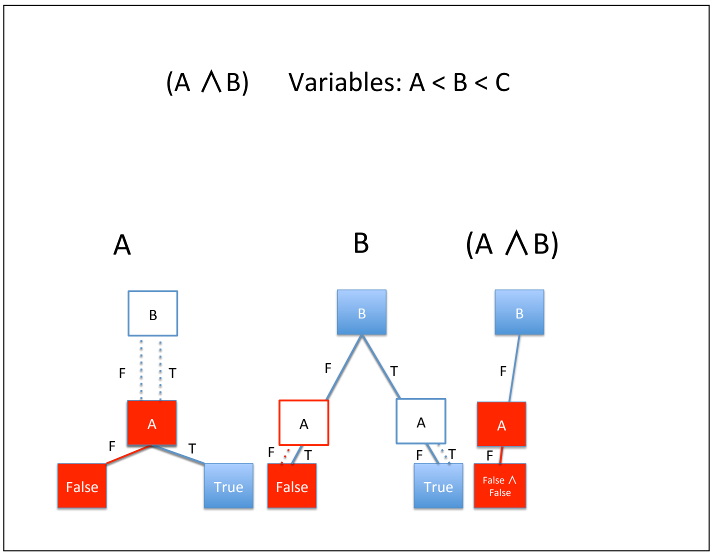

C
Variables: C > B > A
C
C
F
T
False
True

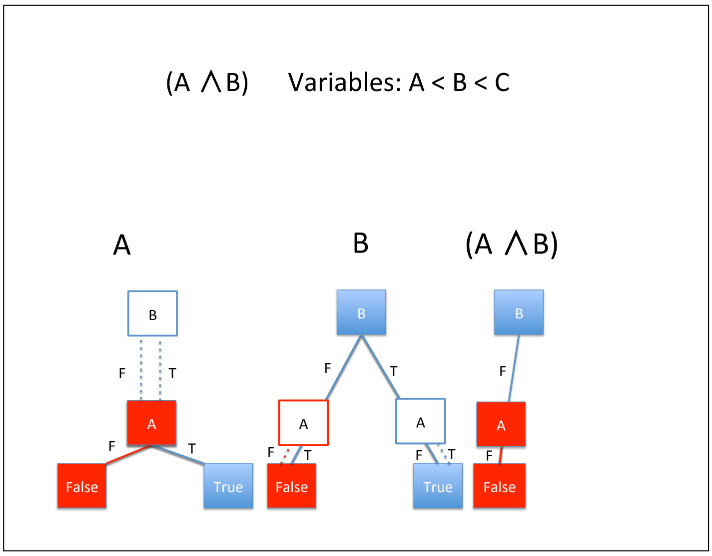

(A ⋀B)
Variables: A < B < C
A
B
A
False
True
B
False
True

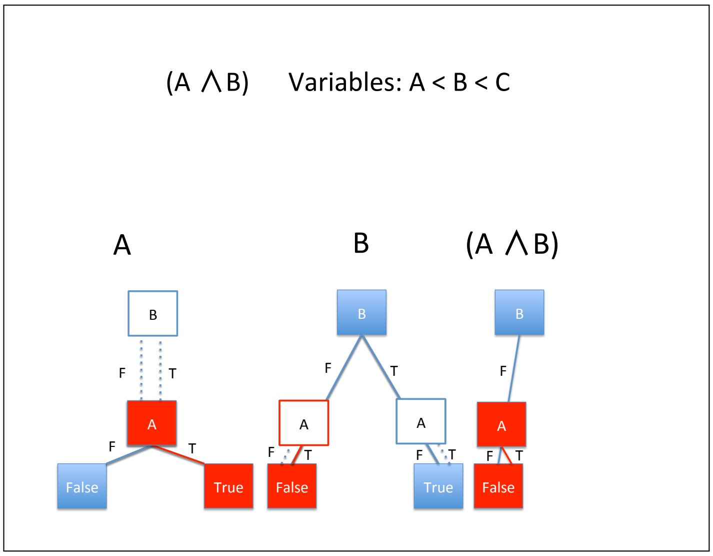

(A ⋀B)
Variables: A < B < C
A
B
B
F
T
A
F
T
False
True
B
F
T
A
A
F
T
F
T
False
True

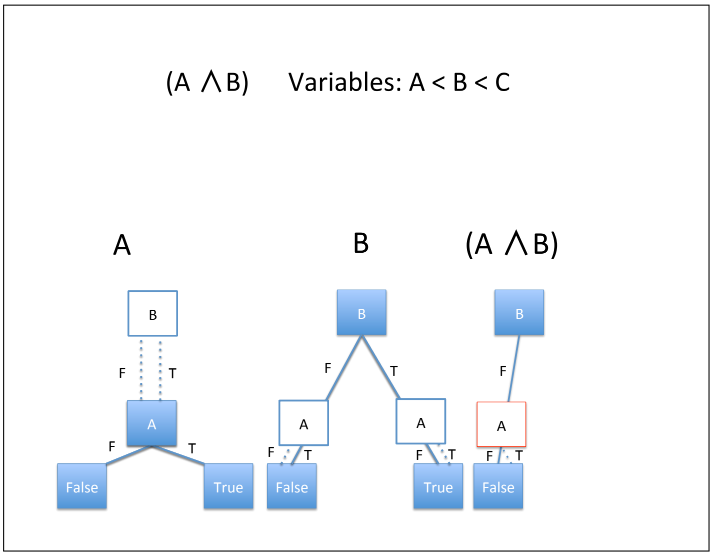

(A ⋀B)
Variables: A < B < C
A
B
(A ⋀B)
B
F
T
A
F
T
False
True
B
F
T
A
A
F
T
F
T
False
True
B

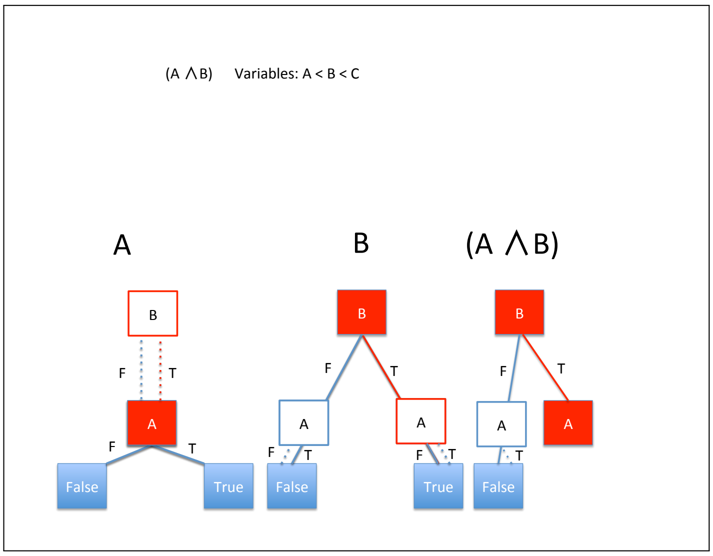

(A ⋀B)
Variables: A < B < C
A
B
(A ⋀B)
B
F
T
A
F
T
False
True
B
F
T
A
A
F
T
F
T
False
True
B
F
A

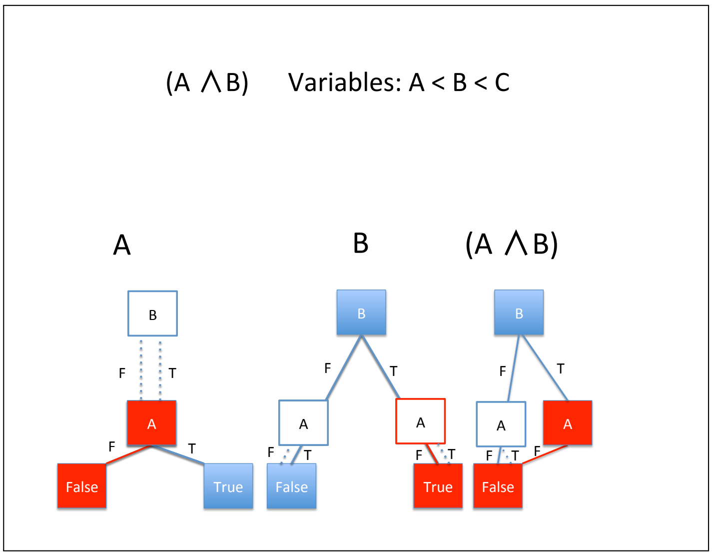

(A ∧B) Variables: A < B < C
A
B
(A ∧B)
B
F
T
A
False
True
False ∧ False

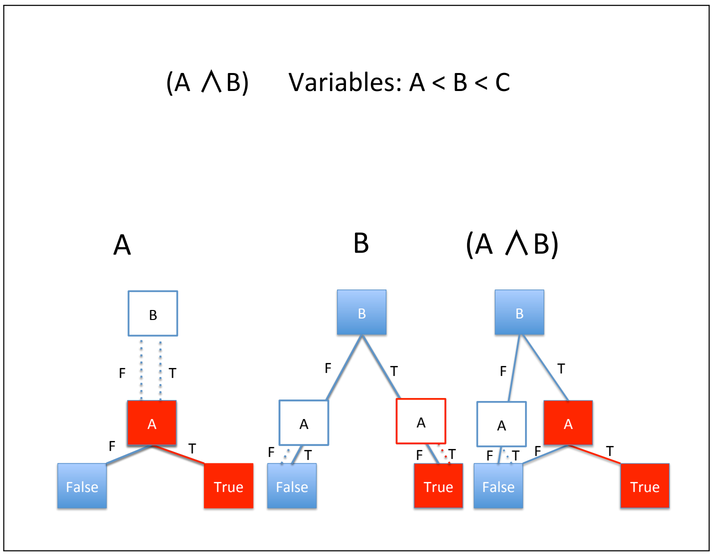

(A ∧B) Variables: A < B < C
A
B
(A ∧B)
B
F
T
A
False
True

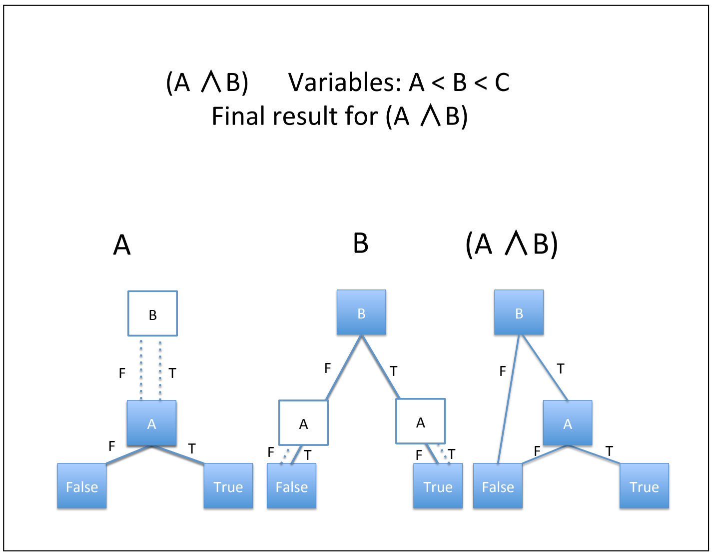

(A ∧B) Variables: A < B < C
A
B
(A ∧B)
B
F
T
A
False
True

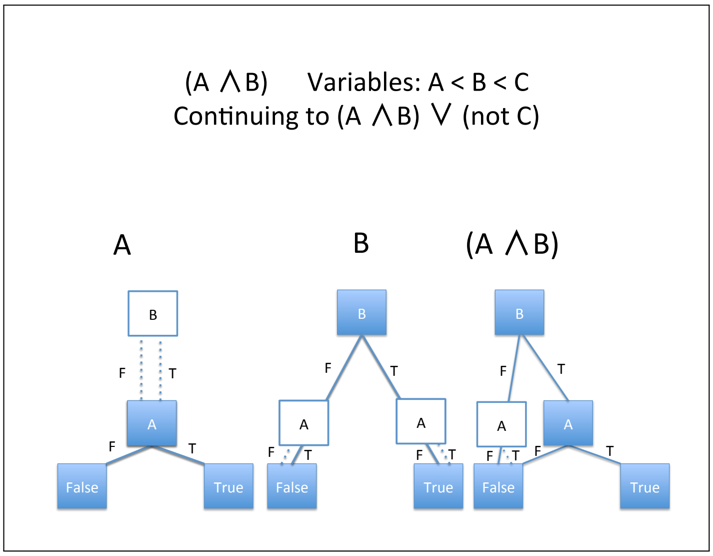

(A ∧B) Variables: A < B < C
A
B
(A ∧B)
B
F
T
A
False
True

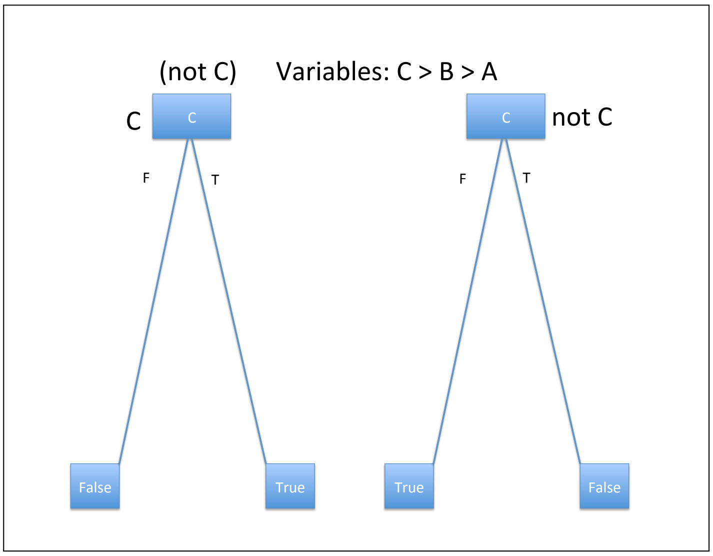

(A ∧B) Variables: A < B < C
A
B
(A ∧B)
B
F
T
A
False
True

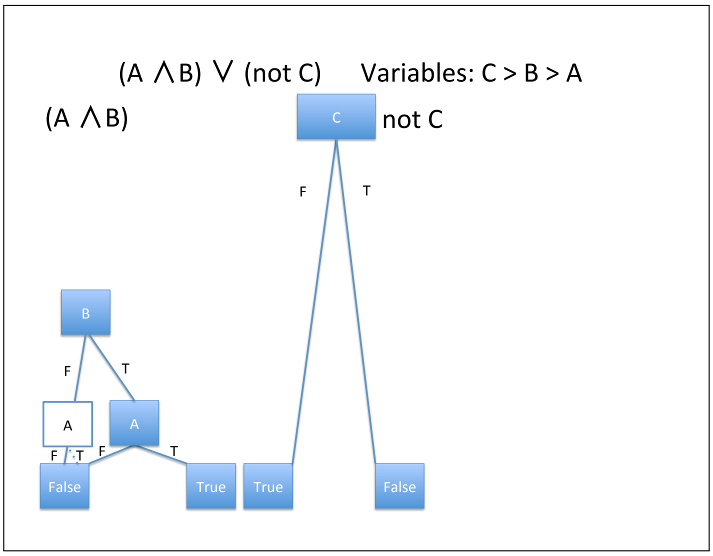

(A ∧B) Variables: A < B < C
A
B
(A ∧B)
B
F
T
A
False
True

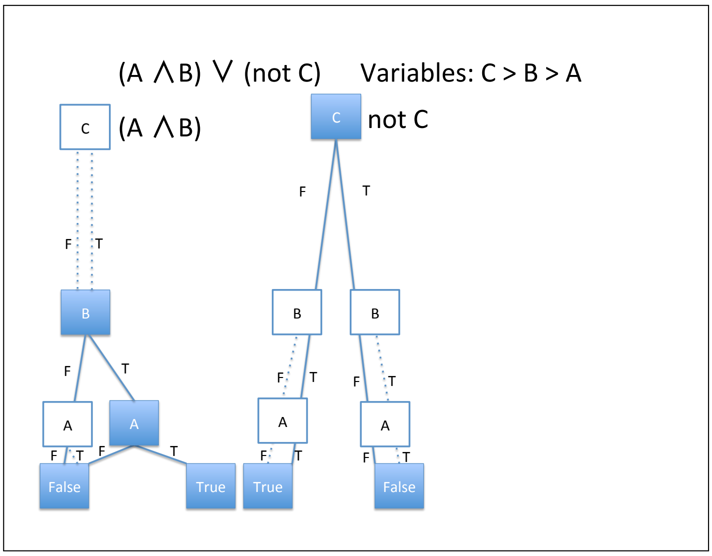

(A ∧B) Variables: A < B < C
A
B
(A ∧B)

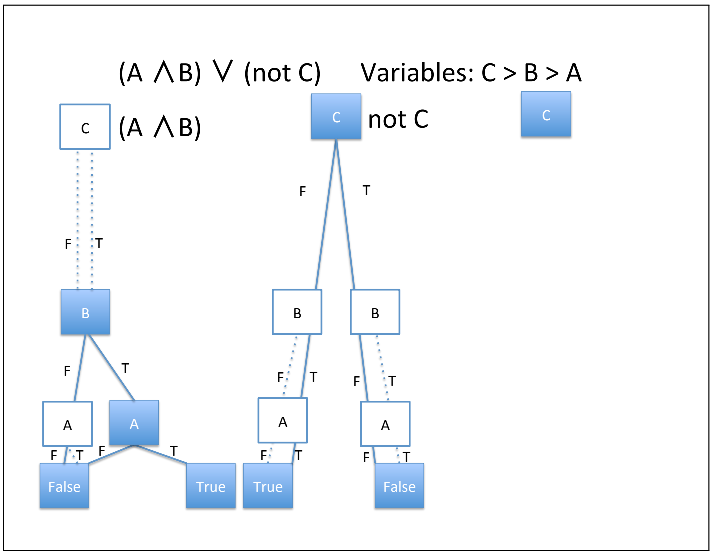

(A ∧B) Variables: A < B < C
Final result for (A ∧B)
A
B
(A ∧B)

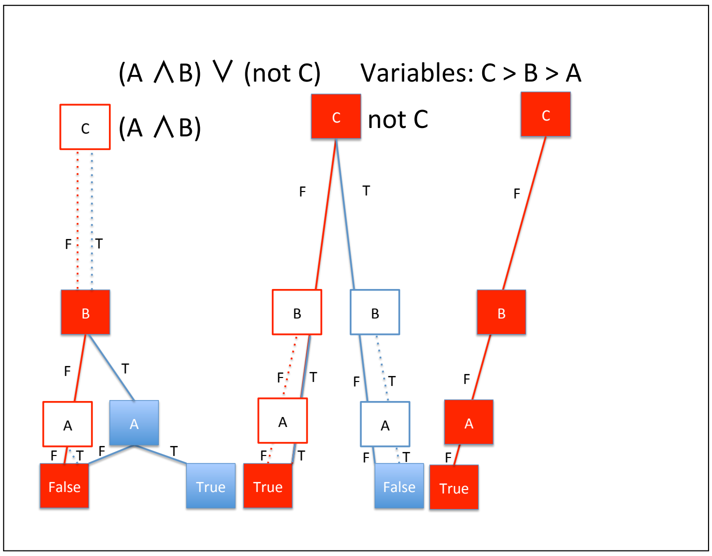

(A ∧B) Variables: A < B < C
Continuing to (A ∧B) ∨ (not C)
A
B
(A ∧B)

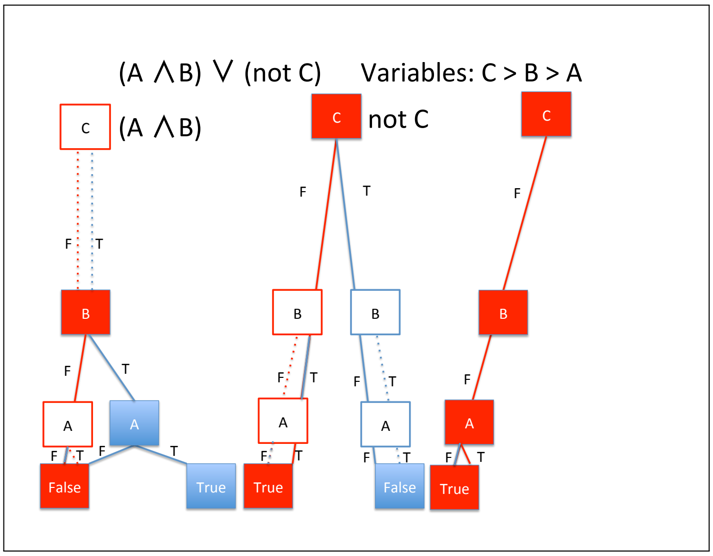

(not C) Variables: C > B > A
C
not C

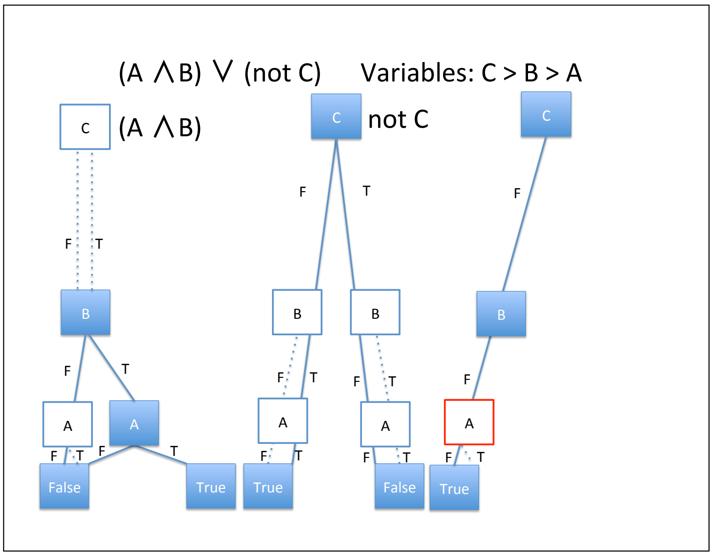

(A ∧B) ∨ (not C) Variables: C > B > A
(A ∧B)
not C

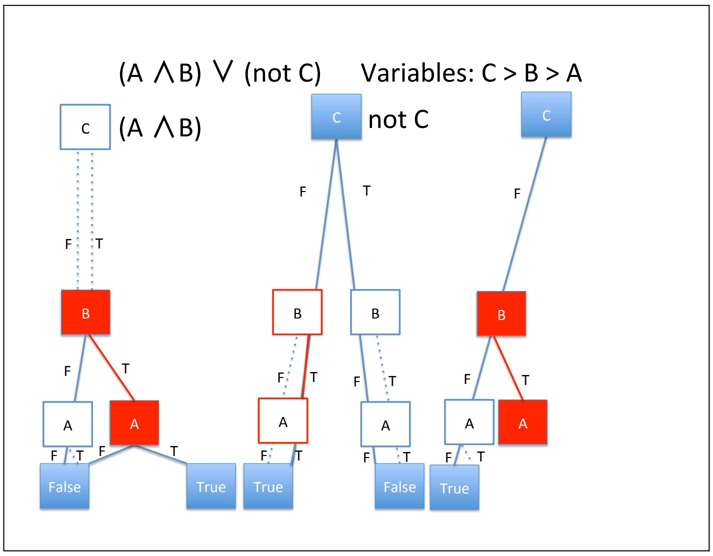

(A ∧B) ∨ (not C) Variables: C > B > A
(A ∧B)
not C

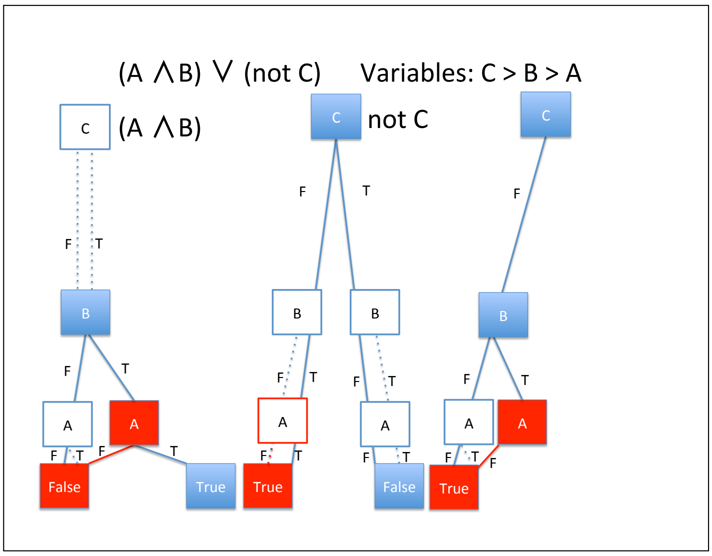

(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C

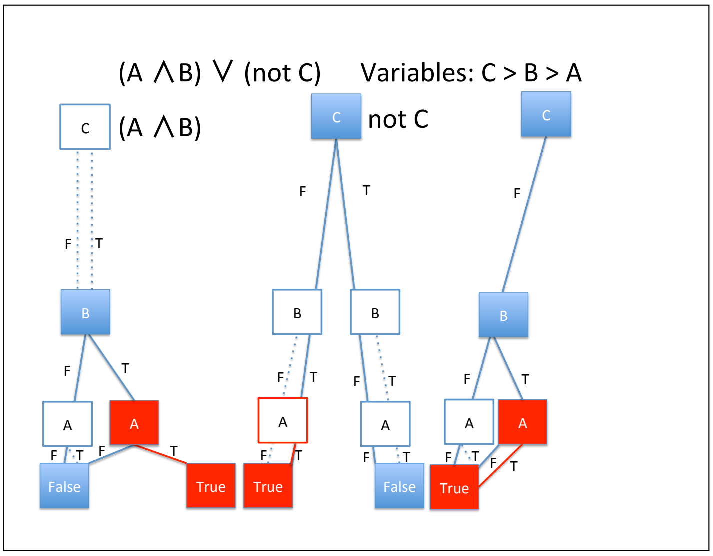

(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C

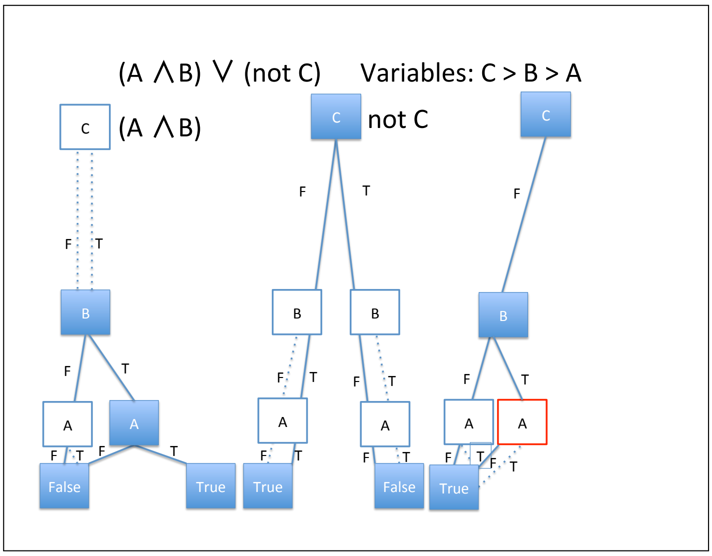

(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C

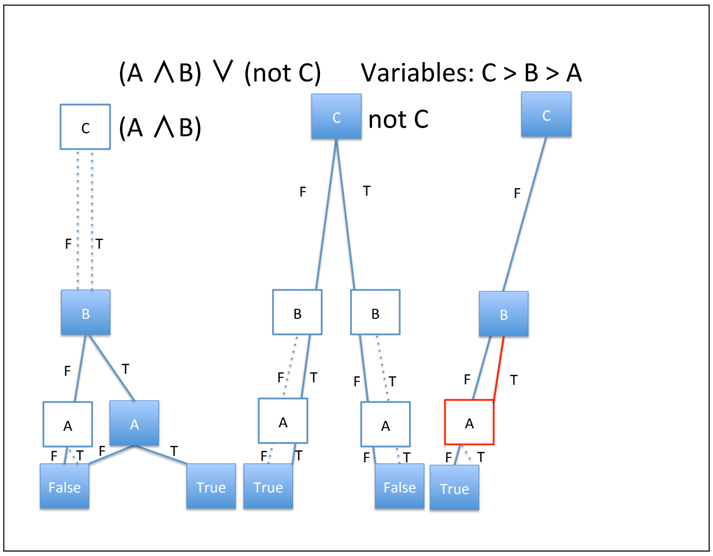

(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C

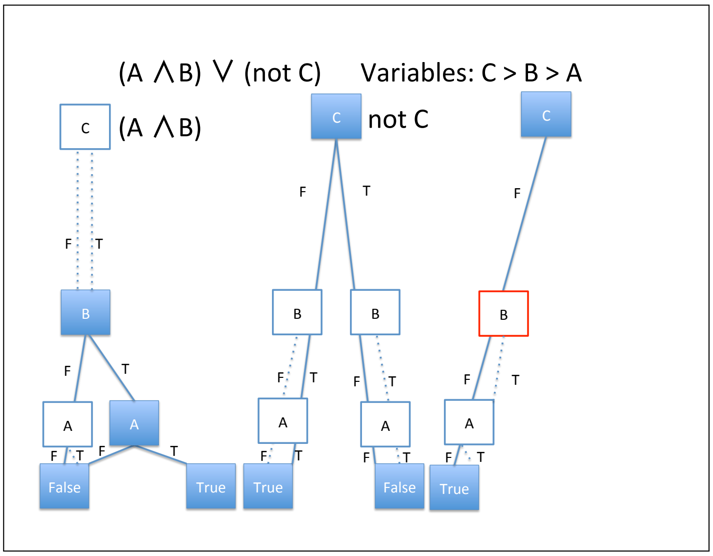

(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C

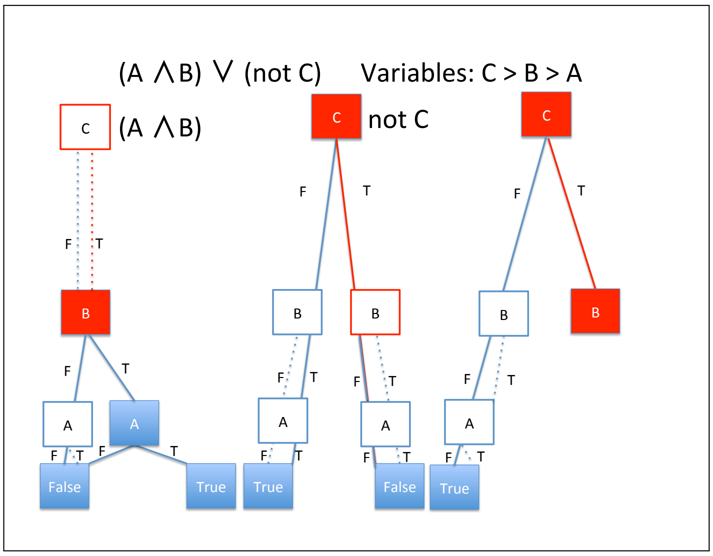

(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C

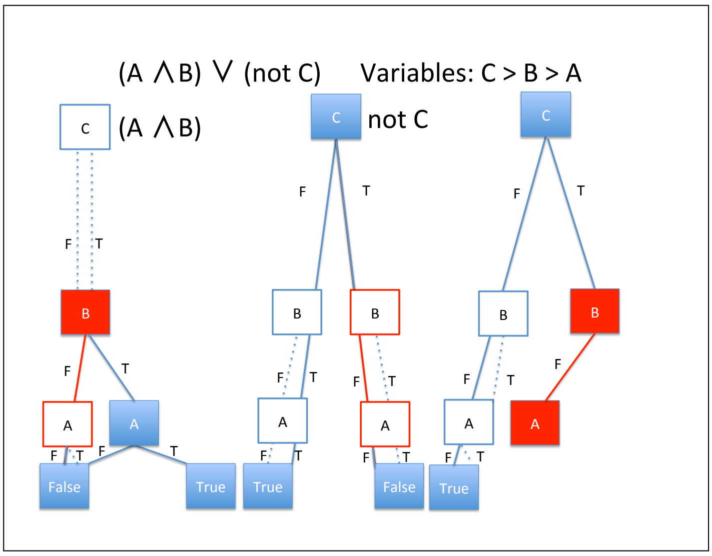

(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C

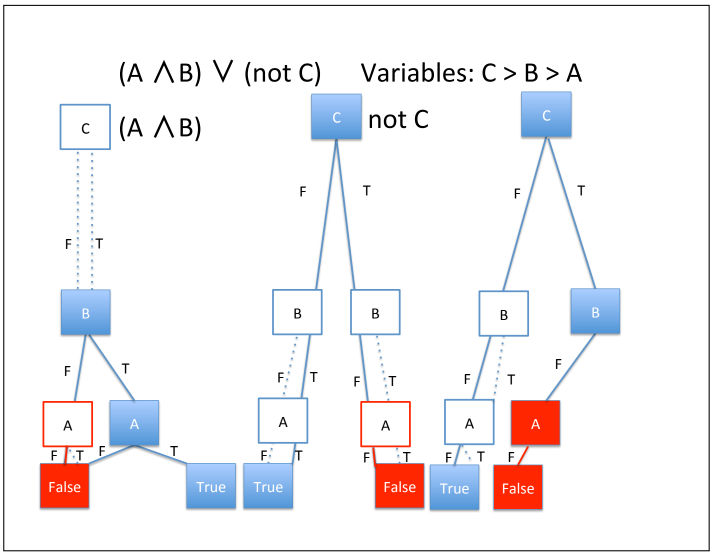

(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C

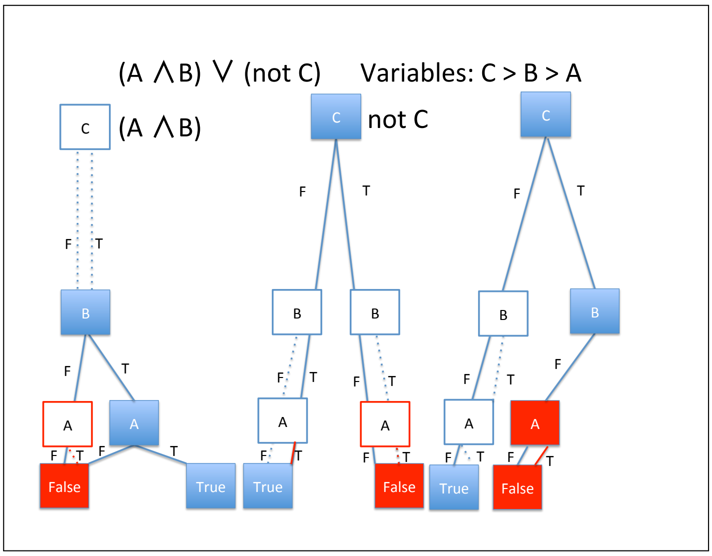

(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C

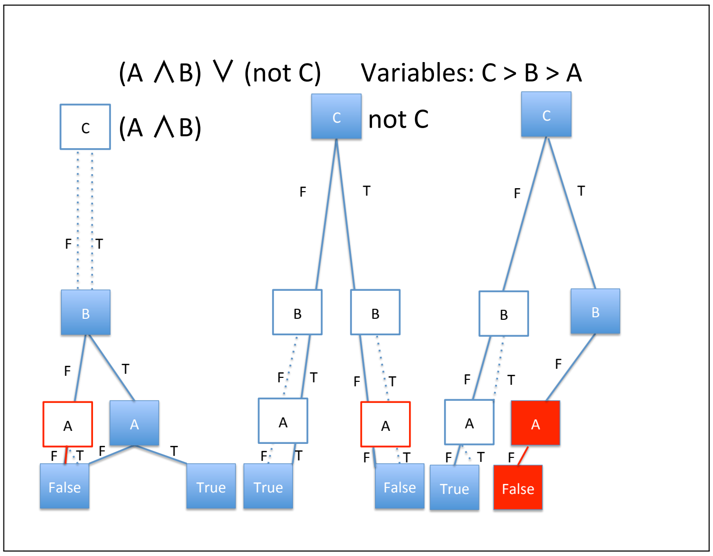

(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C

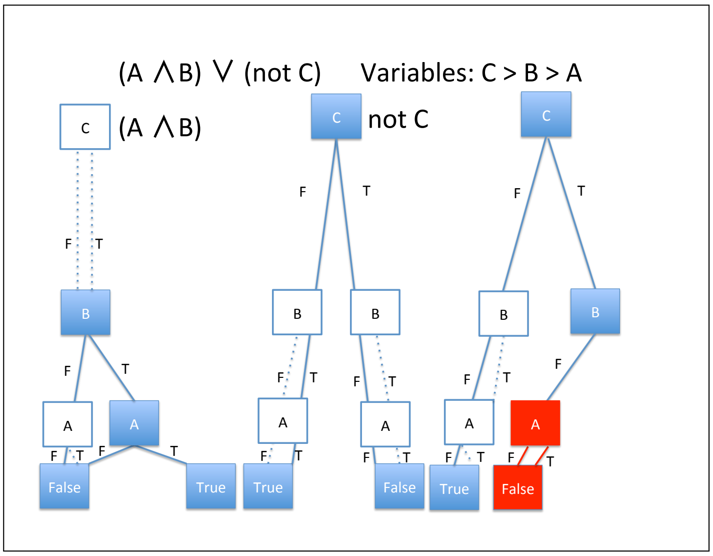

(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C

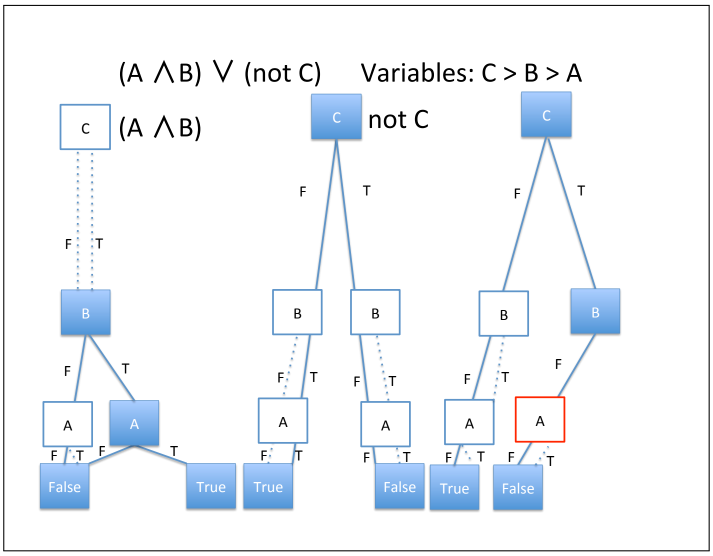

(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C

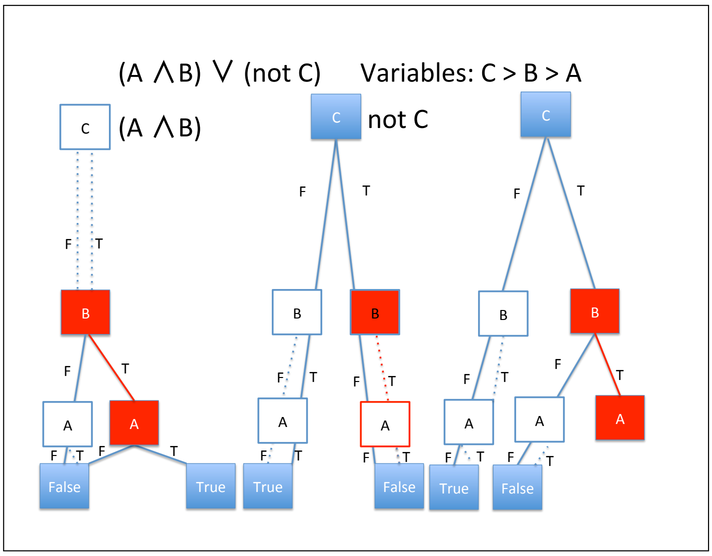

(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C


(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C

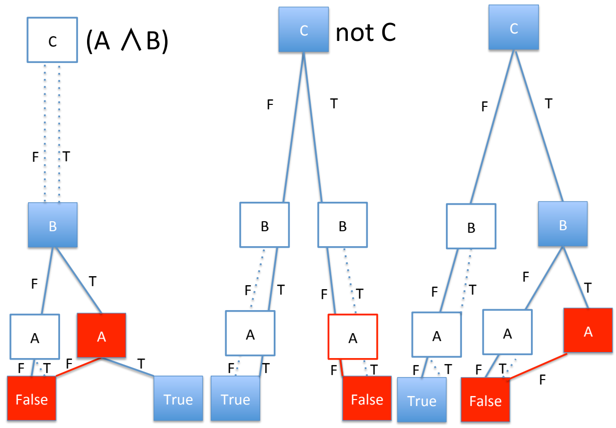

(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C


(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C

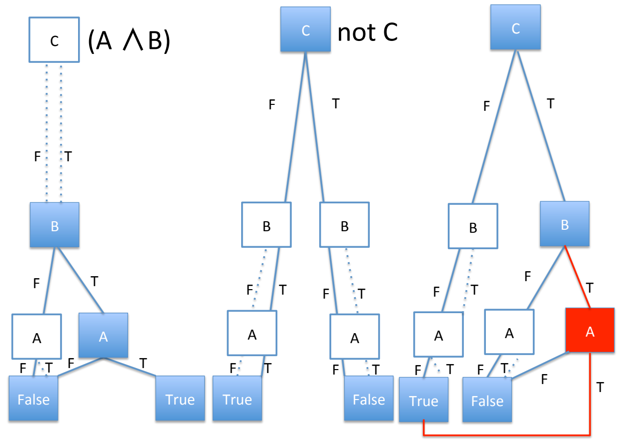

(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C


(A ∧B) ∨ (not C)
Variables: C > B > A
(A ∧B)
not C

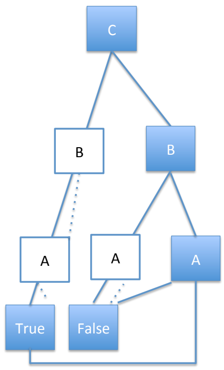
(A ⋀B) ⋁ (not C)   Variables: C > B > A



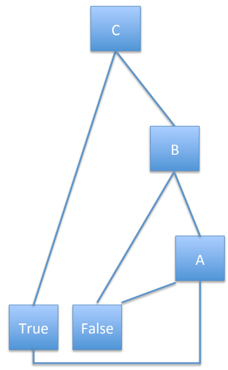
(A ⋀B) ⋁ (not C)   Variables: C > B > A



(A ⋀B) ⋁ (not C)   Variables: C > B > A



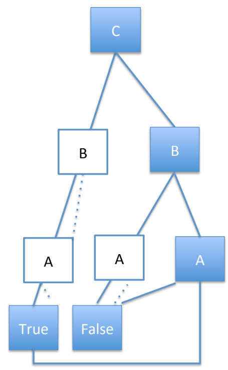
(A ⋀B) ⋁ (not C)   Variables: C > B > A

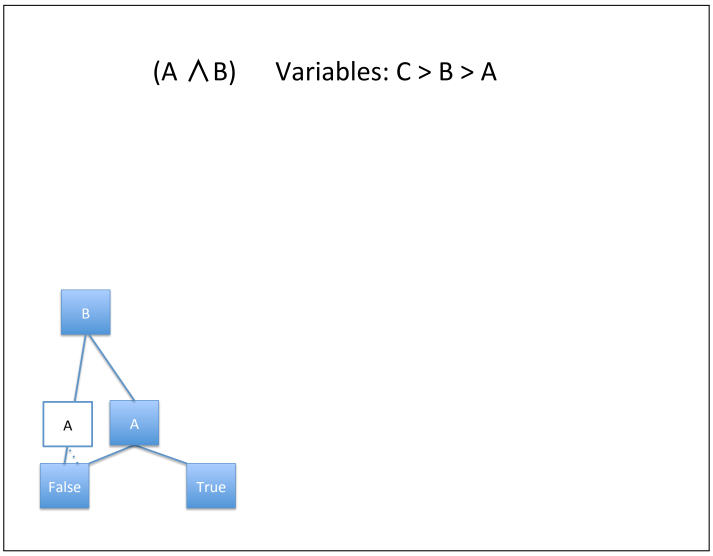


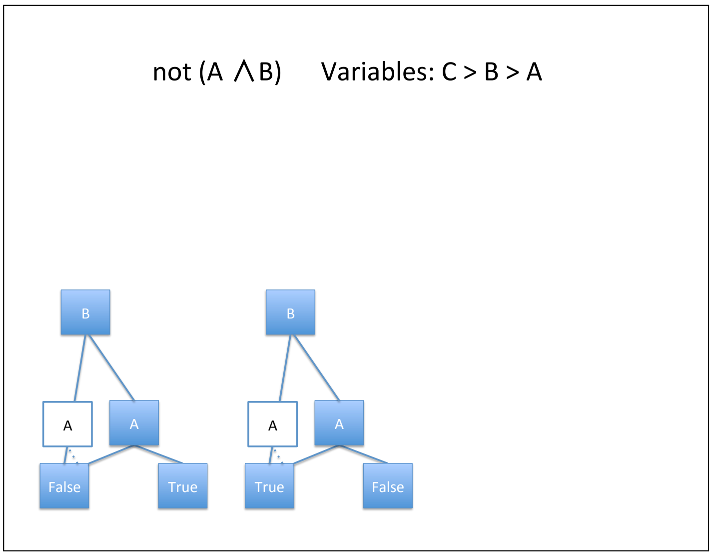
Example

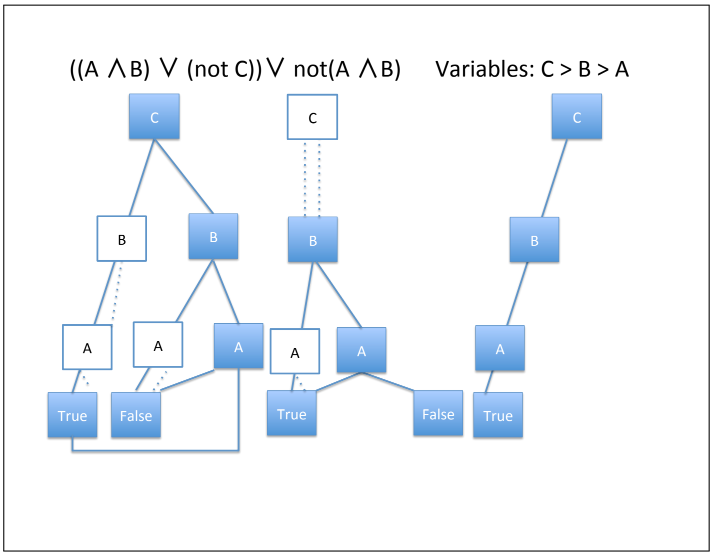
- Find ROBDD for ((A ∧B) ∨ (not C)) ∨ not (A ∧B)

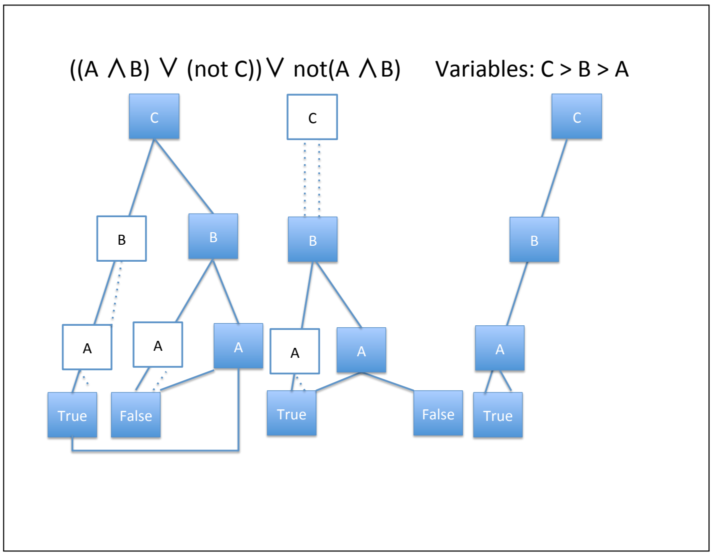
((A ⋀B) ⋁ (not C))   Variables: C > B > A

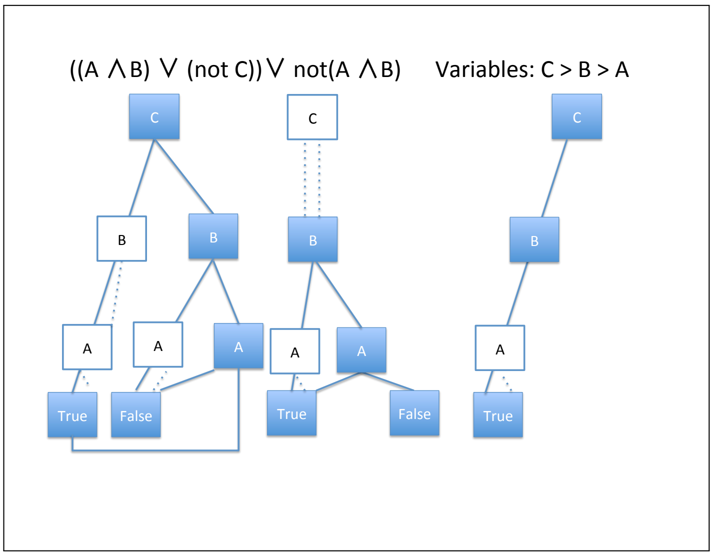


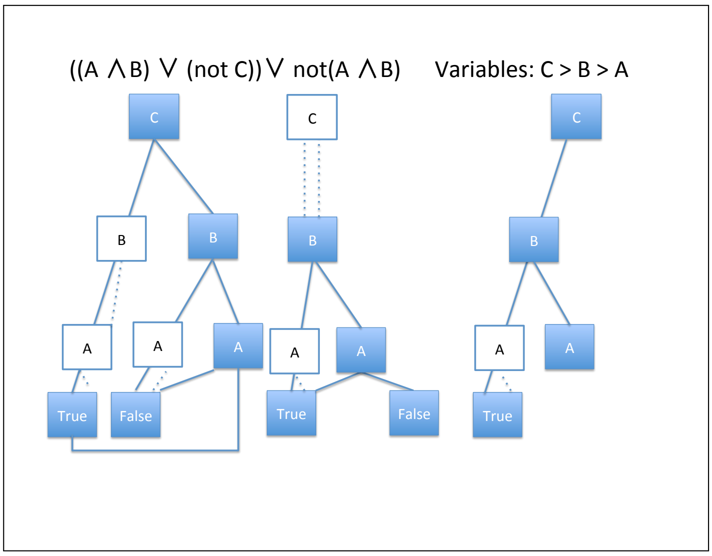

(A ∧ B) Variables: C > B > A

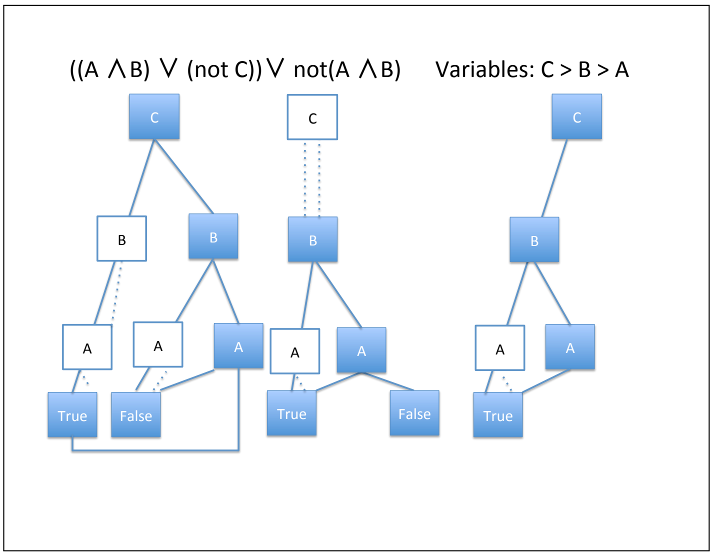

not (A ∧ B) Variables: C > B > A

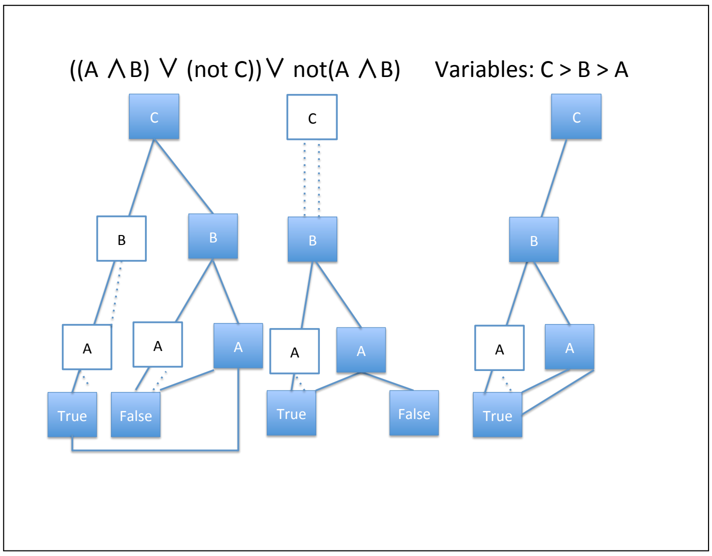

((A ∧ B) ∨ (not C)) ∨ not(A ∧ B) Variables: C > B > A

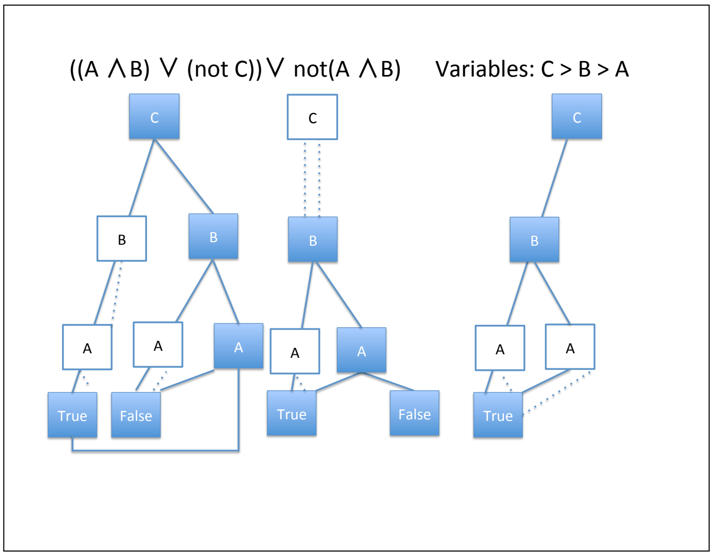

((A ∧ B) ∨ (not C)) ∨ not(A ∧ B) Variables: C > B > A

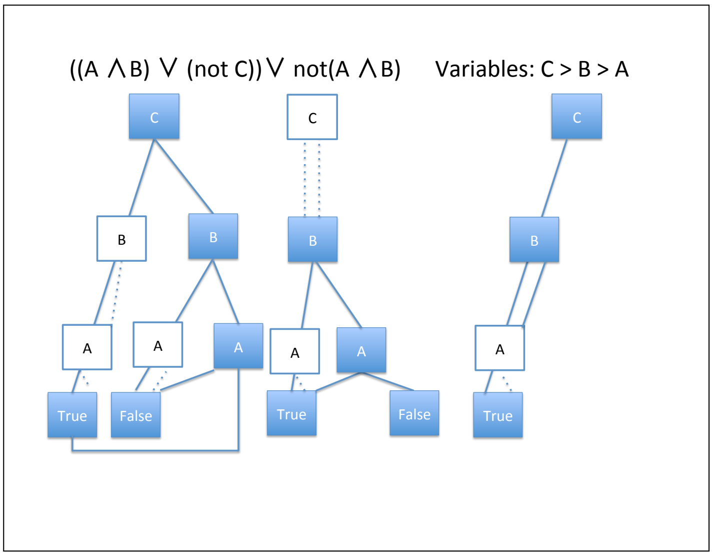

((A ∧ B) ∨ (not C)) ∨ not(A ∧ B) Variables: C > B > A

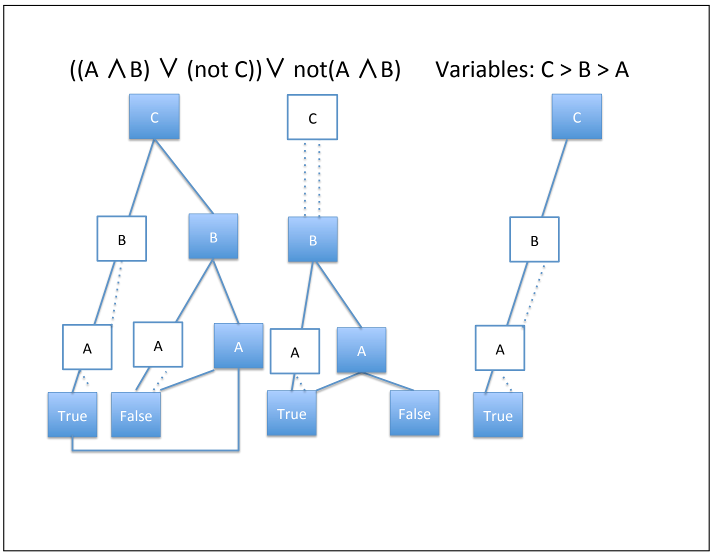

((A ∧ B) ∨ (not C)) ∨ not(A ∧ B) Variables: C > B > A

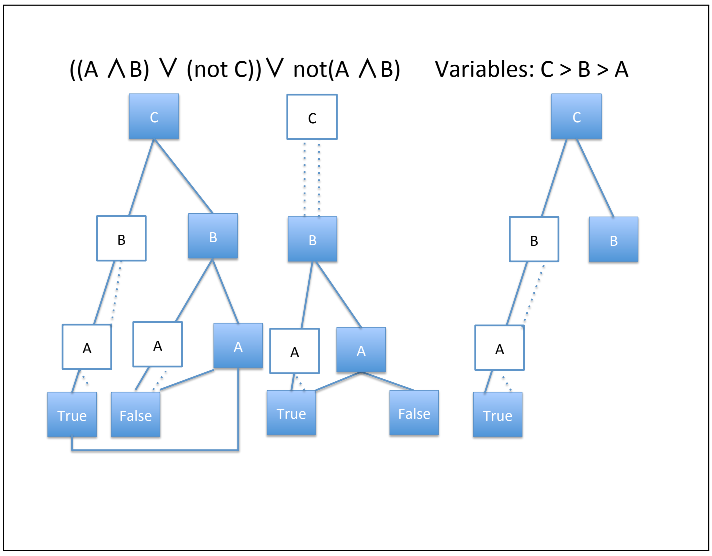

((A ∧B) ∨ (not C))∨ not(A ∧B)
Variables: C > B > A
C
B
A
True
False

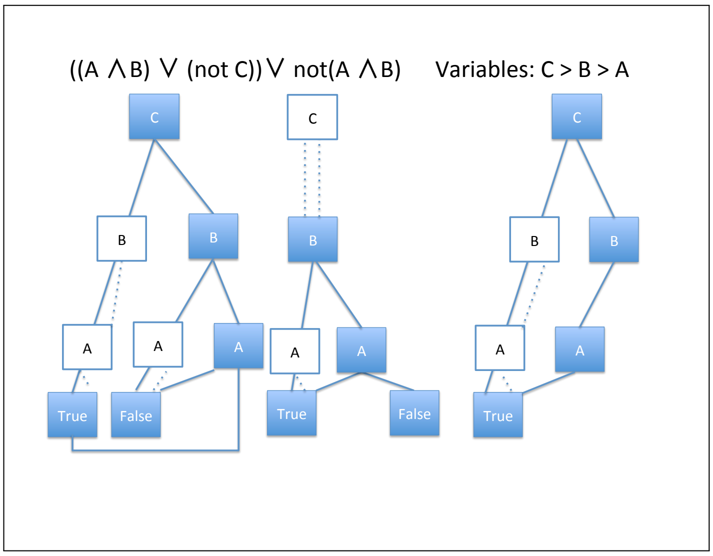

((A ∧B) ∨ (not C))∨ not(A ∧B)
Variables: C > B > A
C
B
A
True
False

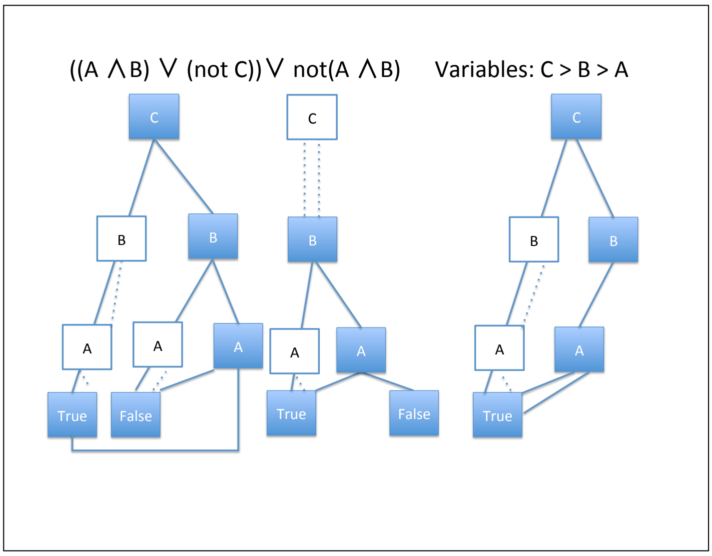

((A ∧B) ∨ (not C))∨ not(A ∧B)
Variables: C > B > A
C
B
A
True
False

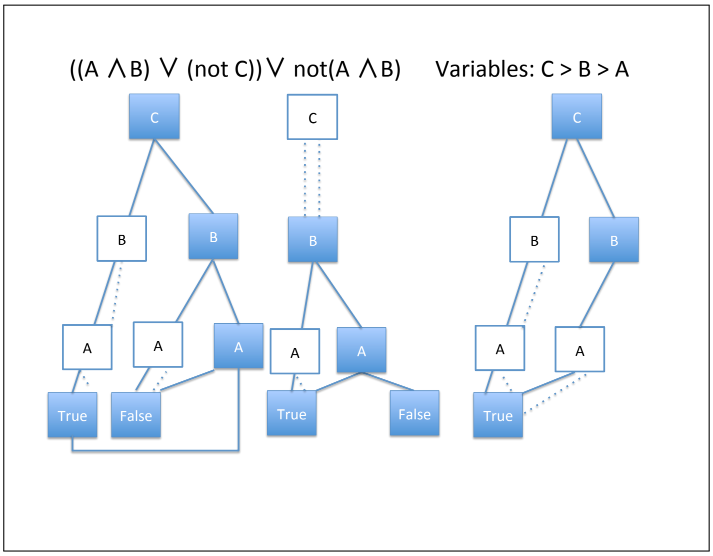

((A ∧B) ∨ (not C))∨ not(A ∧B)
Variables: C > B > A
C
B
A
True
False

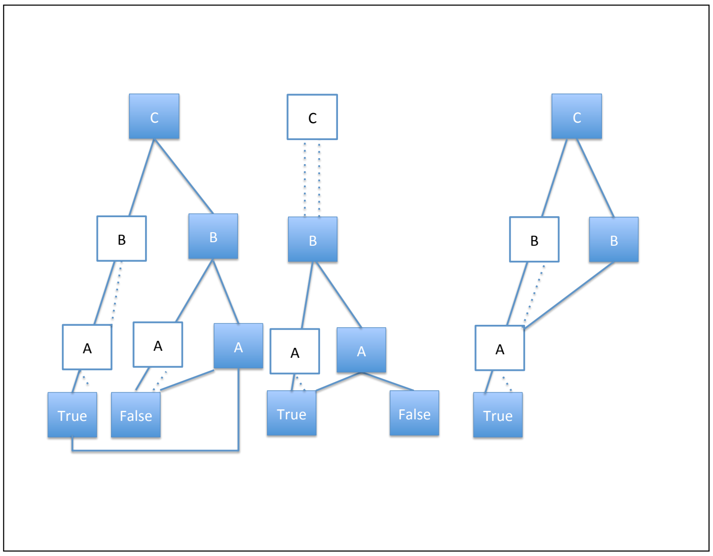

((A ∧B) ∨ (not C))∨ not(A ∧B)
Variables: C > B > A
C
B
A
True
False

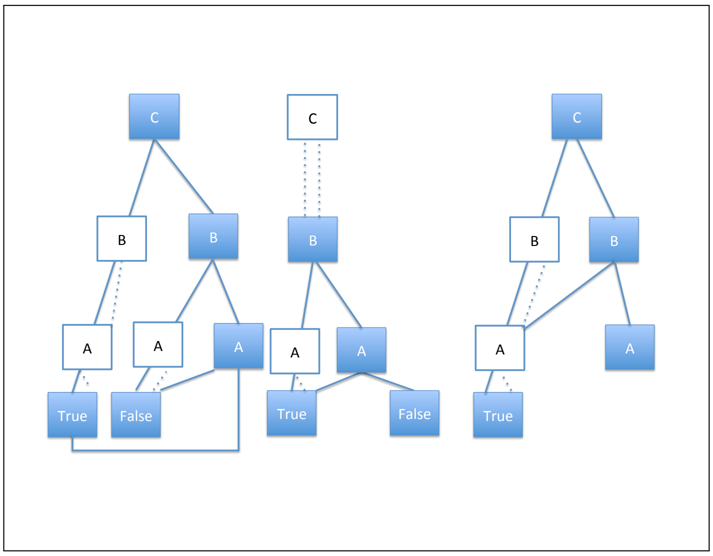

((A ∧B) ∨ (not C))∨ not(A ∧B)
Variables: C > B > A
C
B
A
True
False

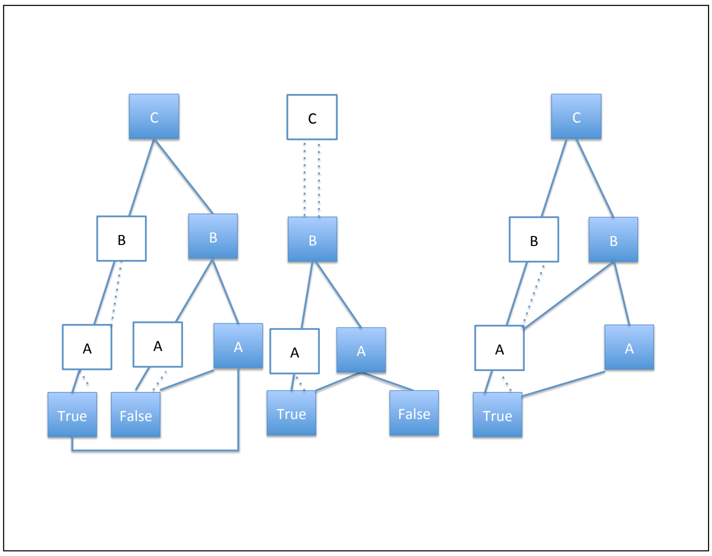

((A ∧B) ∨ (not C))∨ not(A ∧B)
Variables: C > B > A

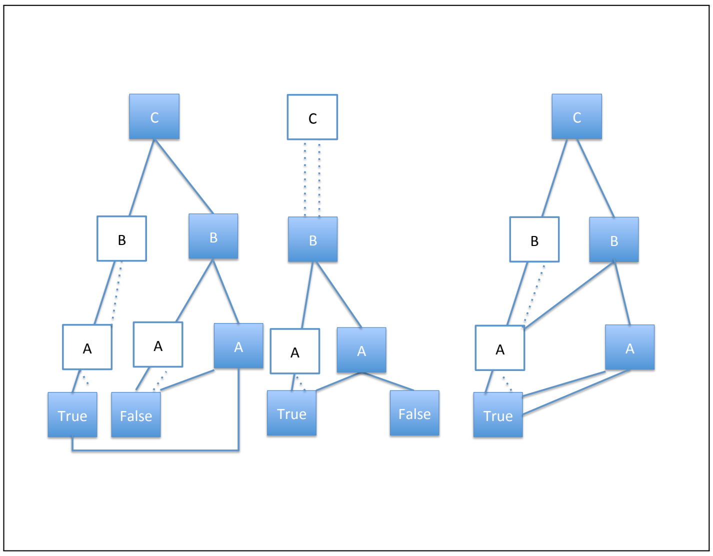

((A ∧B) ∨ (not C))∨ not(A ∧B)
Variables: C > B > A

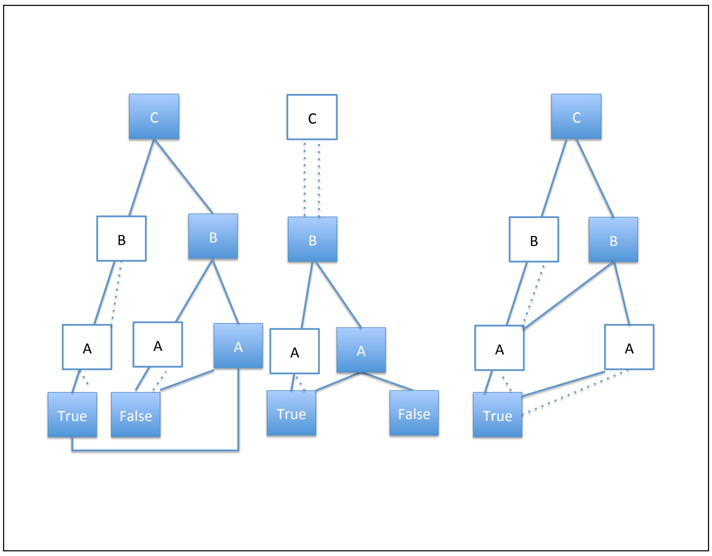

((A ∧B) ∨ (not C))∨ not(A ∧B)
Variables: C > B > A

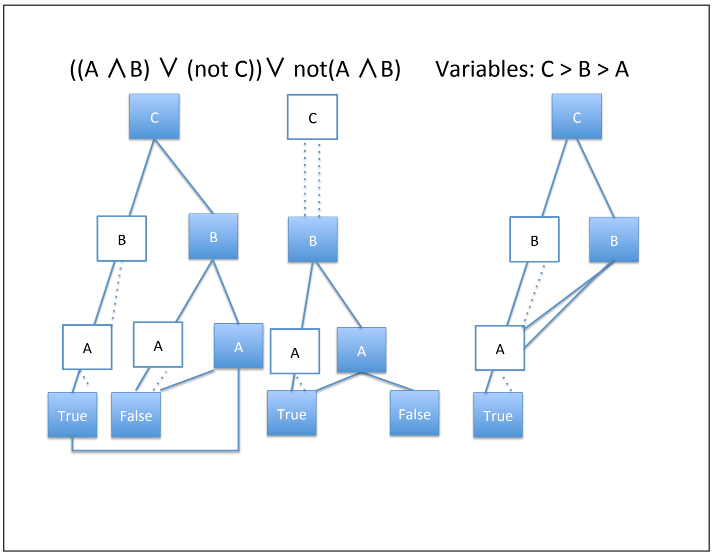


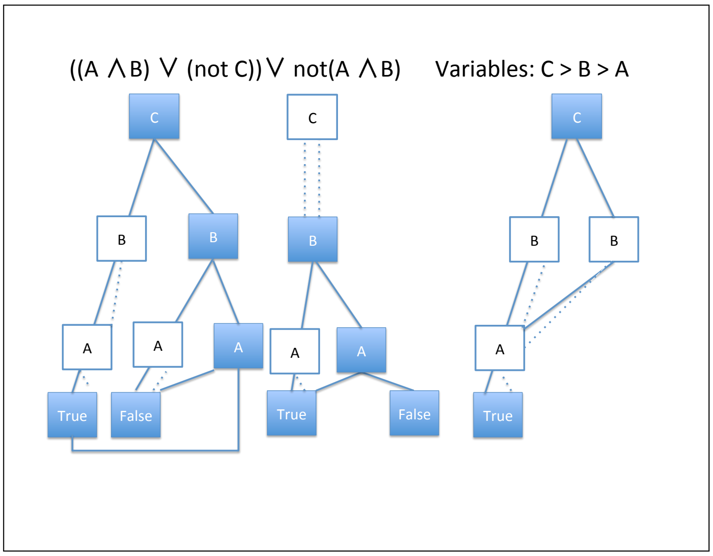


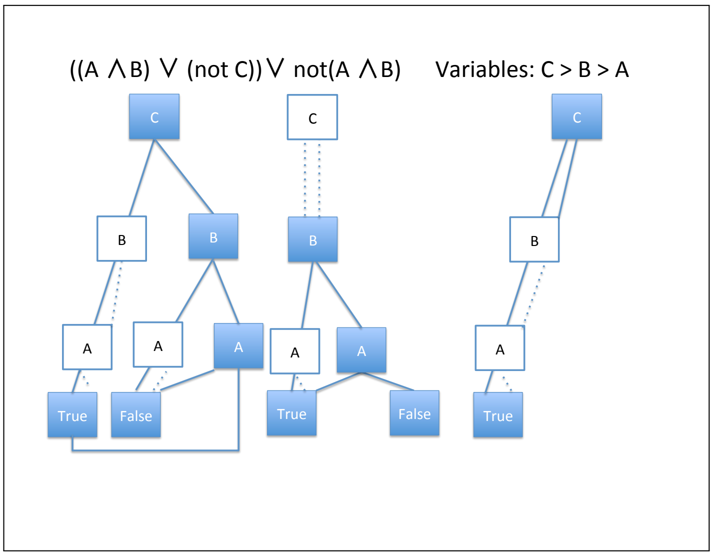


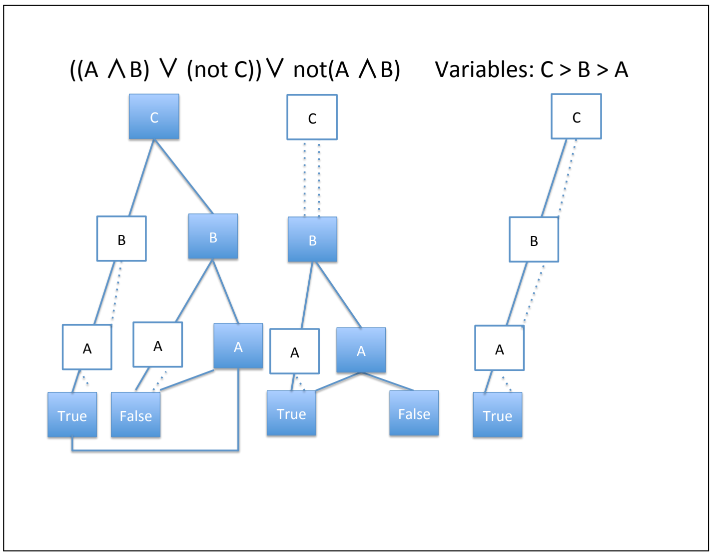

C
B
A
True
False

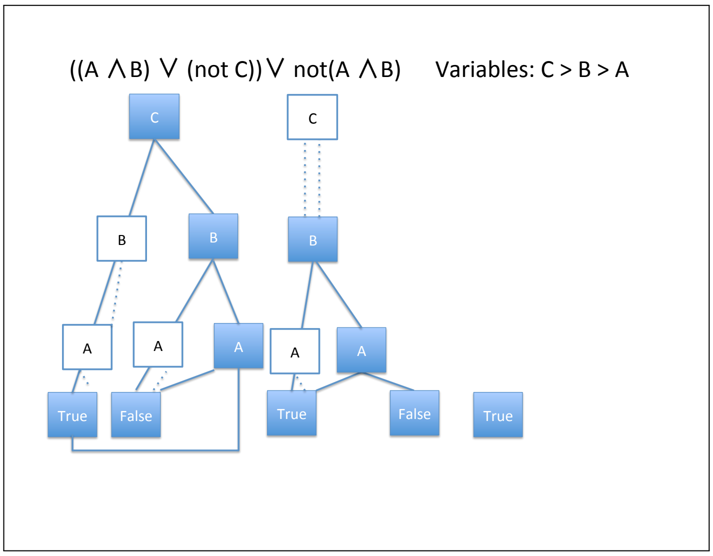

C
B
A
True
False


((A ⋀B) ⋁ (not C))⋁ not(A ⋀B)
Variables: C > B > A
C
B
A
True
False


((A ⋀B) ⋁ (not C))⋁ not(A ⋀B)
Variables: C > B > A
C
B
A
True
False


((A ⋀B) ⋁ (not C))⋁ not(A ⋀B)
Variables: C > B > A
C
B
A
True
False


((A ⋀B) ⋁ (not C))⋁ not(A ⋀B)
Variables: C > B > A
C
B
A
True
False



Uses of ROBDDs

- Reduced Order BDD for a proposition is unique for a fixed variable ordering
- Proposition is valid iff its ROBDD is just True
- Proposition is satisfiable its ROBDD is not just False
- Can check if a given valuation satisfies proposition in time linear to number of variables by walking the corresponding branch