**CSCE 420 - Spring 2023  
Homework 2 (HW2)  
due: Tues, Mar 29, 5:00pm - Late written homeworks will not receive credit.**

**Turn-in answers as a Word document (HW2.docx or .pdf) and commit/push it to your class github repo.**

1. Prove that (A^B→C^D) |- (A^B→C) ("conjunctive rule splitting") is a **sound rule-of- inference** using a **truth table**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | (A^B->C^D) | (A^B->C) |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

As shown in the truth table, all models that satisfy the premise (A^B→C^D) also satisfy the derived sentence (A^B->C).

Also prove (A^B→C^D) |= (A^B→C) using **Natural Deduction.**

-(A^B) v (C^D) <- Used Implication Elimination

(-A v -B) v (C^D) <- Used DeMorgan’s Law

((-A v -B) v C ) ^ ((-A v -B) v D) <- Used Distributivity

(-A v -B) v C <- Used And Elimination

(A ^ B -> C) <- Used Implication Introduction

Also prove (A^B→C^D) |= (A^B→C) using **Resolution**.

-(A^B) v (C^D) <- Used Implication Elimination

(-A v -B) v (C^D) <- Used DeMorgan’s Law

((-A v -B) v C ) ^ ((-A v -B) v D) <- Used Distributivity

((-A v -B) v C ), ((-A v -B) v D) <- Used And Elimination

KB = { ((-A v -B) v C ) , ((-A v -B) v D) , -(A^B→C) } <- Negate query

KB = { ((-A v -B) v C ) , ((-A v -B) v D) , -(-(A^B) v C) } <- Used Implication Elimination

KB = { ((-A v -B) v C ) , ((-A v -B) v D) , - -(A^B) ^ -C } <- Used DeMorgan’s Law

KB = { ((-A v -B) v C ) , ((-A v -B) v D) , (A^B) ^ -C } <- Used Double Negation Elimination

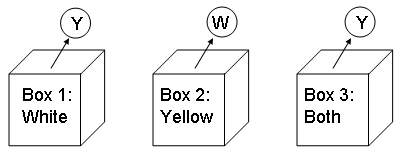
KB = { ((-A v -B) v C ) , ((-A v -B) v D) , A, B, -C } <- Used And Elimination

KB = { -A v -B, ((-A v -B) v D) , A, B } <- Resolve premises 1 and 5

KB = { -A, ((-A v -B) v D) , A} <- Resolve premises 1 and 4

KB = { empty clause, ((-A v -B) v D) } <- Resolve premises 1 and 3

2. **Sammy’s Sport Shop**You are the proprietor of *Sammy’s Sport Shop*. You have just received a shipment of three boxes filled with tennis balls. One box contains only yellow tennis balls, one box contains only white tennis balls, and one contains both yellow and white tennis balls. You would like to stock the tennis balls in appropriate places on your shelves. Unfortunately, the boxes have been labeled incorrectly; the manufacturer tells you that you have exactly one box of each, but that **each box is definitely labeled wrong**. You draw one ball from each box and observe its color. Given the initial (incorrect) labeling of the boxes above, and the three observations, use Propositional Logic to infer the correct contents of the middle box.



Use propositional symbols in the following form: O1Y means a yellow ball was drawn (observed) from box 1, L1W means box 1 was initially labeled white, C1W means box 1 contains (only) white balls, and C1B means box 1 actually contains both types of tennis balls. Note, there is no ‘O1B’, etc, because you can't directly "observe both". When you draw a tennis ball, it will either be white or yellow.

The initial facts describing this particular situation are: {O1Y, L1W, O2W, L2Y, O3Y, L3B}

2a. Using these propositional symbols, write a propositional knowledge base (sammy.kb) that captures the knowledge in this domain (i.e. implications of what different observations or labels mean, as well as constraints inherent in this problem, such as that all boxes have different contents). *Do it in a complete and general way*, writing down *all* the rules and constraints, not just the ones needed to make the specific inference about the middle box. *Do not include* *derived knowledge* that depends on the particular labeling of this instance shown above; stick to what is stated in the problem description above. Your KB should be general enough to reason about any alternative scenario, not just the one given above (e.g. with different observations and labels and box contents).

KB = {O1W -> C1W v C1B, O1W -> -C1Y, O1Y -> C1Y v C1B, O1Y -> -C1W,

O2W -> C2W v C2B, O2W -> -C2Y, O2Y -> C2Y v C2B, O2Y -> -C2W,

O3W -> C3W v C3B, O3W -> -C3Y, O3Y -> C3Y v C3B, O3Y -> -C3W,

L1W -> -C1W, L1Y -> -C1Y, L1B -> -C1B,

L2W -> -C2W, L2Y -> -C2Y, L2B -> -C2B,

L3W -> -C3W, L3Y -> -C3Y, L3B -> -C3B,

C1W -> -C2W ^ -C3W, C1Y -> -C2Y ^ -C3Y, C1B -> -C2B ^ -C3B

C2W -> -C1W ^ -C3W, C2Y -> -C1Y ^ -C3Y, C2B -> -C1B ^ -C3B

C3W -> -C2W ^ -C1W, C3Y -> -C2Y ^ -C1Y, C3B -> -C2B ^ -C1B}

2b. Prove that box 2 must contain white balls (**C2W**) using **Natural Deduction**.

Premises

1. L1W -> -C1W
2. L2Y -> -C2Y
3. L3B -> -C3B
4. O1Y -> C1Y v C1B
5. O2W -> C2W v C2B
6. O3Y -> C3Y v C3B
7. L1W, L2Y, L3B
8. O1Y, O2W, O3Y
9. C3Y -> -C2Y ^ -C1Y
10. C1B -> -C2B ^ -C3B

Derivations

1. -C3B (Modus Ponens, 3, 7)
2. C3Y v C3B (Modus Ponens, 6, 8)
3. C3Y (Resolution, 10, 11)
4. -C2Y ^ -C1Y (Modus Ponens, 9, 12)
5. -C1Y (And Elimination, 13)
6. C1Y v C1B (Modus Ponens, 4, 8)
7. C1B (Resolution, 14, 15)
8. -C2Y (Modus Ponens, 2, 7)
9. C2W v C2B (Modus Ponens, 5, 8)
10. -C2B ^ -C3B (Modus Ponens, 16, a)
11. -C2B (And Elimination, 19)
12. C2W (Resolution, 18, 20)

2c. Convert your KB to CNF.

KB = {-O1W v C1W v C1B, -O1W v -C1Y, - O1Y v C1Y v C1B, -O1Y v -C1W,

-O2W v C2W v C2B, -O2W v -C2Y, -O2Y v C2Y v C2B, -O2Y v -C2W,

-O3W v C3W v C3B, -O3W v -C3Y, -O3Y v C3Y v C3B, -O3Y v -C3W,

-L1W v -C1W, -L1Y v -C1Y, -L1B v -C1B,

-L2W v -C2W, -L2Y v -C2Y, -L2B v -C2B,

-L3W v -C3W, -L3Y v -C3Y, -L3B v -C3B,

L1W, L2Y, L3B,

O1Y, O2W, O3Y,

-C1W v (-C2W ^ -C3W), -C1Y v (-C2Y ^ -C3Y), -C1B v (-C2B ^ -C3B)

-C2W v (-C1W ^ -C3W), -C2Y v (-C1Y ^ -C3Y), -C2B v (-C1B ^ -C3B)

-C3W v (-C2W ^ -C1W), -C3Y v (-C2Y ^ -C1Y), -C3B v (-C2B ^ -C1B)}

The last three lines are simplified with Distributivity:

KB = {….,

(-C1W v -C2W) ^ (-C1W v -C3W), (-C1Y v -C2Y) ^ (-C1Y v -C3Y), (-C1B v -C2B) ^ (-C1B v -C3B),

(-C2W v -C1W) ^ (-C2W v -C3W), (-C2Y v -C1Y) ^ (-C2Y v -C3Y), (-C2B v -C1B) ^ (-C2B v -C3B),

(-C3W v -C2W) ^ (-C3W v -C1W), (-C3Y v -C2Y) ^ (-C3Y v -C1Y), (-C3B v -C2B) ^ (-C3B v -C1B)}

The last three lines are simplified with And Elimination:

KB = {….,

(-C1W v -C2W), (-C1W v -C3W), (-C1Y v -C2Y), (-C1Y v -C3Y), (-C1B v -C2B), (-C1B v -C3B),

(-C2W v -C3W), (-C2Y v -C3Y), (-C2B v -C3B)}

2d. Prove C2W using **Resolution.**

1. -C3B (Resolution on L3B and -L3B v -C3B)
2. C3Y v C3B (Resolution on O3Y and -O3Y v C3Y v C3B)
3. C3Y (Resolution on 1 and 3)
4. -C1Y (Resolution on 3 and -C1Y v -C3Y)
5. C1Y v C1B (Resolution on O1Y and - O1Y v C1Y v C1B)
6. C1B (Resolution on 4 and 5)
7. -C2Y (Resolution on L2Y and -L2Y v -C2Y)
8. C2W v C2B (Resolution on O2W and -O2W v C2W v C2B)
9. -C2B (Resolution on 6 and -C1B v -C2B)
10. C2W (Resolution on 8 and 9)
11. Null (Resolution on 10 and -C2W)

3. Do **Forward Chaining** for the *CanGetToWork* KB below.

You don’t need to follow the formal FC algorithm (with agenda/queue and counts array). Just indicate which rules are triggered (in any order), and keep going until all consequences are generated.  
Show the final list of all inferred propositions at the end. *Is CanGetToWork among them?*

KB = { a. CanBikeToWork → CanGetToWork  
b. CanDriveToWork → CanGetToWork  
c. CanWalkToWork → CanGetToWork  
d. HaveBike ∧ WorkCloseToHome ^ Sunny → CanBikeToWork

e. HaveMountainBike → HaveBike

f. HaveTenSpeed → HaveBike  
g. OwnCar → CanDriveToWork  
h. OwnCar → MustGetAnnualInspection  
i. OwnCar → MustHaveValidLicense  
j. CanRentCar → CanDriveToWork  
k. HaveMoney ∧ CarRentalOpen → CanRentCar  
l. HertzOpen→ CarRentalOpen  
m. AvisOpen→ CarRentalOpen  
n. EnterpriseOpen→ CarRentalOpen  
o. CarRentalOpen → IsNotAHoliday  
p. HaveMoney ∧ TaxiAvailable → CanDriveToWork  
q. Sunny ^ WorkCloseToHome → CanWalkToWork  
r. HaveUmbrella ^ WorkCloseToHome → CanWalkToWork

s. Sunny → StreetsDry }

Facts: { Rainy, HaveMoutainBike, EnjoyPlayingSoccer, WorkForUniversity, WorkCloseToHome, HaveMoney, HertzClosed, AvisOpen, McDonaldsOpen }

Rule e. is triggered and infers HaveBike

Rule m. is triggered and infers CarRentalOpen

Rule o. is triggered and infers IsNotAHoliday

Rule k. is triggered and infers CanRentCar

Rule j. is triggered and infers CanDriveToWork

Rule b. is triggered and infers CanGetToWork

=> Conclusion: Yes, CanGetToWork is among the inferred propositions

4. Do **Backward Chaining** for the *CanGetToWork* KB.  
In this case, you should follow the BC algorithm closely (the pseudocode for the propositional

version of Back-chaining is given in the lecture slides).

Important: when you pop a subgoal (proposition) from the goal stack, you should systematically go through all rules that can be used to prove it IN THE ORDER THEY APPEAR IN THE KB. In some cases, this will lead to *back-tracking*, which you should show.

Also, the sequence of results depends on order in which antecedents are pushed onto the stack. If you have a rule like A^B→C, and you pop C off the stack, push the antecedents in reverse order, so B goes in first, then A; in the next iteration, A would be the next subgoal popped off the stack.

Goal stack: Explanation:

CanGetToWork Push query

CanBikeToWork Rule a

HaveBike, WorkCloseToHome, Sunny Rule d

HaveMountainBike, WorkCloseToHome, Sunny Rule e

WorkCloseToHome, Sunny Known fact

Sunny Known fact

Backtrack

CanDriveToWork Rule b

OwnCar Rule g

Backtrack

CanRentCar Rule j

HaveMoney, CarRentalOpen Rule k

CarRentalOpen Known fact

HertzOpen Rule l

Backtrack

AvisOpen Rule m

Known fact, and stack is empty

5. In what kinds of problems would it be better to use forward-chaining? When would it be better to use backward-chaining?

Forward-chaining is better to use in problems with a very large number of rules without a specific query in mind. Backward chaining is better to use in problems where there is a specific query and it is not clear how the rule set can be used to arrive to the goal.