CSCE 420 - Fall 2025

Homework 3 (HW3)

due: Tues, Oct 28, 11:59 pm

Turn-in answers as a PDF (HW3.pdf) and commit/push it to your class github repo.

All homeworks must be typed, ***not*** hand-written and scanned as photos.

1a. 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	$A \land B$	$C \land D$	$A \land B \to C \land D$	$A \land B \to C$	$(A \land B \to C \land D) \Rightarrow (A \land B \to C)$?
F	F	F	F	F	F	T	T	T
F	F	F	T	F	F	T	T	T
F	F	T	F	F	F	T	T	T
F	F	T	T	F	F	T	T	T
F	T	F	F	F	F	T	T	T
F	T	F	T	F	F	T	T	T
F	T	T	F	F	F	T	T	T
F	T	T	T	F	F	T	T	T
T	F	F	F	F	F	T	T	T
T	F	F	T	F	F	T	T	T

T	F	T	F	F	F	T	T	T
T	F	T	T	F	F	T	T	T
T	T	F	F	T	F	F	F	T
T	T	F	T	T	F	F	F	T
T	T	T	F	T	F	F	T	T
T	T	T	T	T	T	T	T	T

1b. Also prove (A^B®C^D) |= (A^B®C) using **Natural Deduction.**

> (Hint: it might help to use a ROI for "Implication Introduction". If you have a Horn clause, with 1 positive literal and *n-1* negative literals, like (¬XÚZÚ¬Y), you can transform it into a conjunctive rule by collecting the negative literals as positive antecedents, e.g. X^Y®Z. This is a truth-preserving operation (hence sound), which you could prove to yourself using a truth table.)

Premise:
A ^ B -> C ^ D if A and B are both true then C and D are both true
Assume:
A ^ B assume let hand side is true to derive right hand side (implication intro)
Modus ponens on C ^ D as if A ^ B -> C ^ D if A ^ B is True then C ^ D can be concluded as True. For C ^ D to be true, C must be True and D must be True using Modus Ponens. Hence, by And Elimination if P ^ Q is True then you can conclude that just P or just Q. Hence, we can conclude that P is True. So if A ^ B is true then C is True too.

1c. Also prove (A^B->C^D) |= (A^B->C) using **Resolution**.

To prove KB |= α
Assume the negation of the conclusion (¬α)
Add it to the KB
Convert to CNF
And use resolution to derive it

Assuming (A^B->C^D) wanting to prove (A^B->C)

Converting to CNF
A ^ B -> C ^ D
= ¬ (A ^ B) v (C ^ D) implication elimination
= (¬A v ¬B) v (C v D) demoregan law
= (¬A v ¬B v C) ^ (¬A v ¬B v D) distribution
Clauses:
¬A v ¬B v C - 1
¬A v ¬B v D - 2

Want to prove (A^B->C) so
¬ (A^B->C) = A ^ B ^ ¬C
Clauses:
A - 3
B - 4
¬C - 5

Resolution Proof
From 1 and 3
¬A v ¬B v C
A=> ¬B v C - 6
Resolution on A and ¬A

From 6 and 4
¬B v C, B=> C - 7
Resolution on B and ¬B

From 7 and 5
Resolution on C and ¬C

Hence, as drive empty clause (unsatisfiable clauses) statement of (A^B->C^D) |= (A^B->C) holds

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, <u>write a propositional knowledge base</u> (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

Box parameters

(C1Y v C1W v C1B) must have some content
¬(C1Y ^ C1W) can't be yellow and white
¬(C1Y ^ C1B) can't be yellow and both
¬(C1W ^ C1B) can't be white and both

Same for box 2 and 3

(C2Y v C2W v C2B) must have some content
¬(C2Y ^ C2W) can't be yellow and white
¬(C2Y ^ C2B) can't be yellow and both
¬(C2W ^ C2B) can't be white and both

(C3Y v C3W v C3B) must have some content
¬(C3Y ^ C3W) can't be yellow and white
¬(C3Y ^ C3B) can't be yellow and both
¬(C3W ^ C3B) can't be white and both

Each box is labelled incorrectly

L1Y → ¬C1Y
L1W → ¬C1W
L1B → ¬C1B
L2Y → ¬C2Y
L2W → ¬C2W
L2B → ¬C2B

L3Y → ¬C3Y
L3W → ¬C3W
L3B → ¬C3B

When u draw a ball the box should be based on whats drawn so if u draw yellow for example the box can either be yellow or both (observation rules)

O1Y → (C1Y v C1B)
O1W → (C1W v C1B)

O2Y → (C2Y v C2B)
O2W → (C2W v C2B)

O3Y → (C3Y v C3B)
O3W → (C3W v C3B)

All boxes have different actual content

¬(C1Y ^ C2Y)
¬(C1Y ^ C3Y)
¬(C2Y ^ C3Y)

¬(C1W ^ C2W)
¬(C1W ^ C3W)
¬(C2W ^ C3W)

¬(C1B ^ C2B)
¬(C1B ^ C3B)
¬(C2B ^ C3B)

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

O1Y -> yellow ball draw from box 1
L1W -> box 1 labelled white
O2W -> drew white call from box 2
L2Y -> labelled yellow
O3Y -> drew yellow ball from box 3
L3B -> labelled both

From observation rules
O1Y
O1Y → (C1Y v C1B)
C1Y v C1B

O2W

O2W → (C2W ∨ C2B)

C2W ∨ C2B

O3Y

O3Y → (C3Y ∨ C3B)

C3Y ∨ C3B

All labelled wrong

L1W → ¬C1W
L2Y → ¬C2Y
L3B → ¬C3B

To prove C2W we know C2W ^ C2B

From above

1. O2W
2. O2W → (C2W ∨ C2B) observation rule
3. C2W ∨ C2B
4. L2Y
5. L2Y → ¬C2Y wrong label rule
6. ¬C2Y
7. Assume ¬C2W
8. From 3 and 7 must be C2B - disjunction elimination
9. O1Y
10. O1Y → (C1Y v C1B)
11. C1Y v C1B

12. L1W
13. L1W → ¬C1W
14. ¬C1W
15. O3Y
16. O3Y → (C3Y ∨ C3B)
17. C3Y ∨ C3B
18. L3B
19. L3B → ¬C3B
20. ¬C3B
21. From 17 and 19 must be C3Y disjunction elimination
22. From 8 C2B from 21 C3Y
23. From 11 C1Y V C1B
24. Try and assign C1B but theres contradiction only 1 box can be both
25. So ¬C1B from 23 and its C1Y
26. But now theres contradiction as there cannot be two Y's
27. Hence, assumption C2W leads to contradiction
28. Hence, C2W proven true

2c. Convert your KB to CNF.

Box parameters

(C1Y v C1W v C1B) must have some content
¬(C1Y ^ C1W) can't be yellow and white
¬(C1Y ^ C1B) can't be yellow and both
¬(C1W ^ C1B) can't be white and both

(C2Y v C2W v C2B) must have some content
¬(C2Y ^ C2W) can't be yellow and white
¬(C2Y ^ C2B) can't be yellow and both
¬(C2W ^ C2B) can't be white and both

(C3Y v C3W v C3B) must have some content
¬(C3Y ^ C3W) can't be yellow and white
¬(C3Y ^ C3B) can't be yellow and both
¬(C3W ^ C3B) can't be white and both

Changed to CNF using DeMorgan's Law

$(C1Y \lor C1W \lor C1B)$

$(\neg C1Y \lor \neg C1W)$

$(\neg C1Y \lor \neg C1B)$

$(\neg C1W \lor \neg C1B)$

$(C2Y \lor C2W \lor C2B)$

$(\neg C2Y \lor \neg C2W)$

$(\neg C2Y \lor \neg C2B)$

$(\neg C2W \lor \neg C2B)$

$(C3Y \lor C3W \lor C3B)$

$(\neg C3Y \lor \neg C3W)$

$(\neg C3Y \lor \neg C3B)$

$(\neg C3W \lor \neg C3B)$

Each box is labelled incorrectly

$L1Y \rightarrow \neg C1Y$
$L1W \rightarrow \neg C1W$
$L1B \rightarrow \neg C1B$
$L2Y \rightarrow \neg C2Y$
$L2W \rightarrow \neg C2W$
$L2B \rightarrow \neg C2B$

$L3Y \rightarrow \neg C3Y$
$L3W \rightarrow \neg C3W$
$L3B \rightarrow \neg C3B$

Changed to CNF using implication elimination

$(\neg L1Y \lor \neg C1Y)$
$(\neg L1W \lor \neg C1W)$
$(\neg L1B \lor \neg C1B)$

$(\neg L2Y \lor \neg C2Y)$
$(\neg L2W \lor \neg C2W)$

(¬L2B ∨ ¬C2B)

(¬L3Y ∨ ¬C3Y)
(¬L3W ∨ ¬C3W)
(¬L3B ∨ ¬C3B)

When u draw a ball the box should be based on whats drawn so if u draw yellow for example the box can either be yellow or both (observation rules)

O1Y → (C1Y v C1B)
O1W → (C1W v C1B)

O2Y → (C2Y v C2B)
O2W → (C2W v C2B)

O3Y → (C3Y v C3B)
O3W → (C3W v C3B)

Change to CNF using Implication elimination

(¬O1Y ∨ C1Y ∨ C1B)
(¬O1W ∨ C1W ∨ C1B)

(¬O2Y ∨ C2Y ∨ C2B)
(¬O2W ∨ C2W ∨ C2B)

(¬O3Y ∨ C3Y ∨ C3B)
(¬O3W ∨ C3W ∨ C3B)

All boxes have different actual content

¬(C1Y ^ C2Y)
¬(C1Y ^ C3Y)
¬(C2Y ^ C3Y)

¬(C1W ^ C2W)
¬(C1W ^ C3W)
¬(C2W ^ C3W)

¬(C1B ^ C2B)
¬(C1B ^ C3B)

¬(C2B ^ C3B)

simplified using DeMorgan's Law

(¬C1Y ∨ ¬C2Y)
(¬C1Y ∨ ¬C3Y)
(¬C2Y ∨ ¬C3Y)

(¬C1W ∨ ¬C2W)
(¬C1W ∨ ¬C3W)
(¬C2W ∨ ¬C3W)

(¬C1B ∨ ¬C2B)
(¬C1B ∨ ¬C3B)
(¬C2B ∨ ¬C3B)

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

Assume ¬C2W

1. O2W
2. ¬O2W ∨ C2W ∨ C2B
3. Resolution on O2W is C2W v C2B

1. Using ¬C2W and previous resolution of C2W v C2B
2. Resolution on ¬C2W and C2W is C2B

1. Trying C2B
2. From (¬O1Y ∨ C1Y ∨ C1B)
3. Gives us O1Y v C1B
4. L1W
5. (¬L1W ∨ ¬C1W)
6. ¬C1W
7. (¬O3Y ∨ C3Y ∨ C3B)
8. O3Y
9. O3Y v C3B
10. L3B
11. (¬L3B ∨ ¬C3B)
12. ¬C3B

Therefore
C2 is both from assumption
C1 is Y or B

C3 is Y from above

Assuming C1 = B
Error as there cannot be two both boxes (using unique rule stated in KB)
Assuming C1 = Y
Error as there cannot be two Y boxes (using unique rule stated in KB)

Hence, derived empty clauses from assuming ¬C2W
This means assumption is False Therefore we can concluse 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 Ù WorkCloseToHome ^ Sunny ® CanBikeToWork
       e. HaveMountainBike ® HaveBike
       f. HaveTenSpeed ® HaveBike
       g. OwnCar ® CanDriveToWork
       h. OwnCar ® MustGetAnnualInspection
       i. OwnCar ® MustHaveValidLicense
       j. CanRentCar ® CanDriveToWork
       k. HaveMoney Ù CarRentalOpen ® CanRentCar
       l. HertzOpen® CarRentalOpen
       m. AvisOpen® CarRentalOpen
       n. EnterpriseOpen® CarRentalOpen
       o. CarRentalOpen ® IsNotAHoliday
       p. HaveMoney Ù TaxiAvailable ® CanDriveToWork
       q. Sunny ^ WorkCloseToHome ® CanWalkToWork
       r. HaveUmbrella ^ WorkCloseToHome ® CanWalkToWork
       s. Sunny ® StreetsDry }
```

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

1. HaveMoutainBike -> HaveBike infer: HaveBike
2. HaveMoney ^ CarRentalOpen -> CanRentCar (we have money but need car rental to be open) Infer: CanRentCar

3. HertzOpen → CarRentalOpen (HertzClosed)

4. AvisOpen → CarRentalOpen (AvisOpen implies CarRentalOpen implies CanRentCar) Infer: CanRentalOpen
5. CanRentCar → CanDriveToWork infer: CanDriveToWork
6. CanBikeToWork → CanGetToWork
7. CanDriveToWork → CanGetToWork infer: CanGetToWork
8. HaveBike ^ WorkCloseToHome ^ Sunny → CanBikeToWork (its close to hope and havebike but does not know if its sunny or not so cannot infer)

9. HaveMoney ^ TaxiAvailable → CanDriveToWork (no taxi mention cannot)
10. OwnCar → CanDriveToWork (nope)

11. Sunny ^ WorkCloseToHome → CanWalkToWork (dont know if suny)

12. HaveUmbrella ^ WorkCloseToHome → CanWalkToWork (do not have umbrella)

13. Sunny → StreetsDry (dont know if suny)

Hence, CanGetToWork is inferred

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.

Pop CanGetToWork

Rules that CanGetToWork

a. CanBikeToWork → CanGetToWork
b. CanDriveToWork → CanGetToWork
c. CanWalkToWork → CanGetToWork

Goal Stack = [CanBikeToWork]

Pop CanBikeToWork

Rules:

HaveBike ^ WorkCloseToHome ^ Sunny → CanBikeToWork

Goal Stack = [Sunny, WorkCloseToHome, HaveBike]

Pop Sunny

Sunny is not a fact and no rules concludes it so Fail BackTrack

Try Rules B

Goal Stack = [CanDriveToWork]

Rules:

OwnCar → CanDriveToWork
CanRentCar → CanDriveToWork
HaveMoney ^ TaxiAvailable → CanDriveToWork

Goal Stack = [OwnCar]

OwnCar not a fact and no rules on it fail and backtrack

Goal Stack = [CanRentCar]

Pop CanRentCar

Rules: HaveMoney ^ CarRentalOpen → CanRentCar

Goal Stack = [CarRentalOpen, HaveMoney]

Pop CarRentalOpen

Rules:

HertzOpen → CarRentalOpen
AvisOpen → CarRentalOpen
EnterpriseOpen → CarRentalOpen
Try HertzOpen but its closed so fail and backtrack

Try AvisOpen

AvisOpen is known fact

Goal Stack = [HaveMoney]

Pop HaveMoney

HaveMoney is known fact therefore, success

All subhoals for Rule K are satisfied and hence canRentCar and CanDriveToWork and CanGetToWork