**CSCE 420 - Fall 2023 Homework 2 (HW2)**

**due: Tues, Oct 24, 11:59 pm**

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

|  |  |  |
| --- | --- | --- |
| When pushing the final version of your HW1, also run the command: | |  |
| **git tag "HW1" && git push origin "HW1"** |  |
| This will be used to record time of submission for late penalty when applicable. | | |

All homeworks must be typed, *not* hand-written and scanned/photo-shot.

1a. Prove that “Implication Introduction” (the opposite of Implication Elimination) is a **sound rule of inference** (ROI) using a **truth table**. If you have a Horn clause, with 1 positive literal and *n-1* negative literals, like (¬XvZv¬Y), you can transform it into a conjunctive rule by collecting the negative literals as positive antecedents, e.g. X^Y→Z. It is sufficient to prove this for *n-1*=2 antecedents. (In fact, this is a truth-preserving operation, hence sound.)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| X | Y | Z | -X | -Y | -X v Z v -Y | X ^ Y | X ^ Y -> Z |
| T | T | T | F | F | T | T | T |
| T | T | F | F | F | F | T | F |
| T | F | T | F | T | T | F | T |
| T | F | F | F | T | T | F | T |
| F | T | T | T | F | T | F | T |
| F | T | F | T | F | T | F | T |
| F | F | T | T | T | T | F | T |
| F | F | F | T | T | T | F | T |

The truth values for these sentences are the same so they are truth preserving meaning it is a sound rule of inference

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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | A^B | C^D | A^B->C^D | A^B->C |
| T | T | T | T | T | T | T | T |
| T | T | T | F | T | F | F | T |
| T | T | F | T | T | F | F | F |
| T | T | F | F | T | F | F | F |
| T | F | T | T | F | T | T | T |
| T | F | T | F | F | F | T | T |
| T | F | F | T | F | F | T | T |
| T | F | F | F | F | F | T | T |
| F | T | T | T | F | T | T | T |
| F | T | T | F | F | F | T | T |
| F | T | F | T | F | F | T | T |
| F | T | F | F | F | F | T | T |
| F | F | T | T | F | T | T | T |
| F | F | T | F | F | F | T | T |
| F | F | F | F | F | F | T | T |

This is rule is sound because all the models that satisfy the premise also satisfy the derived sentence.

1c. Also prove (A^B→C^D) |= (A^B→C) using **Natural Deduction.** (hint: use 1a above)

1 -(A^B) v C^D implication elimination

2 -Av-B v C^D demorgans law (1)

3 -A v -B v C distribution (2)

4 -A v -B v D distribution (2)

5 A ^ B -> C implication introduction (4)

1d. Also prove (A^B→C^D) |= (A^B→C) using **Resolution**.

1 -(A^B) v C rewrote derived sentence

2 -(-(A^B) v C) negate derived sentence (1)

3 A^B^-C Demorgans law (2)

4 A And elimination (3)

5 B And elimination (3)

6 -C And elimination (3)

7 A^B -> C^D from premise

8 -(A^B) v C^D implication elimination (7)

9 -Av-B v C^D Demorgans law (8)

10 -A v -B v C distribution (9)

11 -A v -B v D distribution (9)

12 -B v C Resolution (10 and 4)

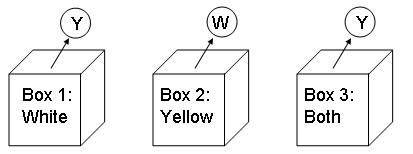
13 C Resolution (12 and 6)

14 {} Resolution (13 and 6)

By proving the negated derived sentence results in an empty sentence though resolution, the premise is proved to entail the derived sentence.

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).

// labels are incorrect

1 L1Y -> -C1Y

2 L1W -> -C1W

3 L1B -> -C1B

4 L2Y -> -C2Y

5 L2W -> -C2W

6 L2B -> -C2B

7 L3Y -> -C3Y

8 L3W -> -C3W

9 L3B -> -C3B

// all contents different

10 C1W -> -C2W ^ -C3W

11 C1Y -> -C2Y ^ -C3Y

12 C1B -> -C2B ^ -C3B

13 C2W -> -C1W ^ -C3W

14 C2Y -> -C1Y ^ -C3Y

15 C2B -> -C1B ^ -C3B

16 C3W -> -C1W ^ -C2W

17 C3Y -> -C1Y ^ -C2Y

18 C3B -> -C1B ^ -C2B

// observed color implies either box is that color or contains both

19 O1Y -> C1Y v C1B

20 O1W -> C1W v C1B

21 O2Y -> C2Y v C2B

22 O2W -> C2W v C2B

23 O3Y -> C3Y v C3B

24 O3W -> C3W v C3B

// labels are incorrect (another way to label)

25 L1Y -> C2Y v C3Y

26 L1W -> C2W v C3W

27 L1B -> C2B v C3B

28 L2Y -> C1Y v C3Y

29 L2W -> C1W v C3W

30 L2B -> C1B v C3B

31 L3Y -> C1Y v C2Y

32 L3W -> C1W v C2W

33 L3B -> C1B v C2B

// cant have multiple labels

34 C1Y -> -C1W

35 C1Y -> -C1B

36 C1W -> -C1Y

37 C1W -> -C1B

38 C1B -> -C1Y

39 C1B -> -C1W

34 C2Y -> -C2W

35 C2Y -> -C2B

36 C2W -> -C2Y

37 C2W -> -C2B

38 C2B -> -C2Y

39 C2B -> -C2W

34 C3Y -> -C3W

35 C3Y -> -C3B

36 C3W -> -C3Y

37 C3W -> -C3B

38 C3B -> -C3Y

39 C3B -> -C3W

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

a L2Y observed

b O2W observed

c L1W observed

d O1Y observed

e L3B observed

f O3Y observed

g -C2Y (a and 4)

h -C1W (c and 2)

i -C3B (e and 9)

j C2W v C2B (b and 22)

k C1Y v C1B (d and 19)

l C3Y v C3B (f and 23)

m C3Y resolution (i and l)

n -C1Y ^ -C2Y (m and 17)

o -C1Y and elimination (n)

p -C2Y and elimination (n)

q C1B resolution (k and o)

r -C2B ^ -C3B (q and 12)

s -C2B and elimination (r)

t -C3B and elimination (r)

u C2W resolution (j and s)

2c. Convert your KB to CNF.

// labels are incorrect

1 -L1Y v -C1Y

2 -L1W v -C1W

3 -L1B v -C1B

4 -L2Y v -C2Y

5 -L2W v -C2W

6 -L2B v -C2B

7 -L3Y v -C3Y

8 -L3W v -C3W

9 -L3B v -C3B

// all contents different

10 -C1W v -C2W

11 -C1W v -C3W

12 -C1Y v -C2Y

13 -C1Y v -C3Y

14 -C1B v -C2B

15 -C1B v -C3B

16 -C2W v -C1W

17 -C2W v -C3W

18 -C2Y v -C1Y

19 -C2Y v -C3Y

20 -C2B v -C1B

21 -C2B v -C3B

22 -C3W v -C1W

23 -C3W v -C2W

24 -C3Y v -C1Y

25 -C3Y v -C2Y

26 -C3B v -C1B

27 -C3B v -C2B

// observed color implies either box is that color or contains both

28 -O1Y v C1Y v C1B

29 -O1W v C1W v C1B

30 -O2Y v C2Y v C2B

31 -O2W v C2W v C2B

32 -O3Y v C3Y v C3B

33 -O3W v C3W v C3B

// labels are incorrect (another way to label)

34 -L1Y v C2Y v C3Y

35 -L1W v C2W v C3W

36 -L1B v C2B v C3B

37 -L2Y v C1Y v C3Y

38 -L2W v C1W v C3W

39 -L2B v C1B v C3B

40 -L3Y v C1Y v C2Y

41 -L3W v C1W v C2W

42 -L3B v C1B v C2B

// cant have multiple labels

43 -C1Y v -C1W

44 -C1Y v -C1B

45 -C1W v -C1Y

46 -C1W v -C1B

47 -C1B v -C1Y

48 -C1B v -C1W

49 -C2Y v -C2W

50 -C2Y v -C2B

51 -C2W v -C2Y

52 -C2W v -C2B

53 -C2B v -C2Y

54 -C2B v -C2W

55 -C3Y v -C3W

56 -C3Y v -C3B

57 -C3W v -C3Y

58 -C3W v -C3B

59 -C3B v -C3Y

60 -C3B v -C3W

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

Facts = { i O1Y,

ii L1W,

iii O2W,

iv L2Y,

v O3Y,

vi L3B }

a -C2W negate derived sentence

b -L1W v C3W resolution (a and 35)

c -L3W v C1W resolution (a and 41)

d -O2W v C2B resolution (a and 31)

e C3W resolution (b and iii)

f C3Y v C3B 32 and v

g -C3Y e and 55

h C3B f and g

i -C3W h and 60

j { } e and i

Because the negated derived sentence resulted in an empty sentence, the derived sentence is true through resolution refutation.

1. 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

* 1. CanDriveToWork → CanGetToWork
  2. CanWalkToWork → CanGetToWork
  3. HaveBike ^ WorkCloseToHome ^ Sunny → CanBikeToWork
  4. HaveMountainBike → HaveBike
  5. HaveTenSpeed → HaveBike
  6. OwnCar → CanDriveToWork
  7. OwnCar → MustGetAnnualInspection
  8. OwnCar → MustHaveValidLicense
  9. CanRentCar → CanDriveToWork
  10. HaveMoney ^ CarRentalOpen → CanRentCar
  11. HertzOpen→ CarRentalOpen
  12. AvisOpen→ CarRentalOpen
  13. EnterpriseOpen→ CarRentalOpen
  14. CarRentalOpen → IsNotAHoliday
  15. HaveMoney ^ TaxiAvailable → CanDriveToWork
  16. Sunny ^ WorkCloseToHome → CanWalkToWork
  17. HaveUmbrella ^ WorkCloseToHome → CanWalkToWork
  18. Sunny → StreetsDry }

Facts: { Rainy, HaveMoutainBike, EnjoyPlayingSoccer, WorkForUniversity,

WorkCloseToHome, HaveMoney, HertzClosed, AvisOpen, McDonaldsOpen }

Facts (agenda):

1) Rainy, HaveMoutainBike, EnjoyPlayingSoccer, WorkForUniversity,

WorkCloseToHome, HaveMoney, HertzClosed, AvisOpen, McDonaldsOpen

2) Rainy, HaveMoutainBike, EnjoyPlayingSoccer, WorkForUniversity,

WorkCloseToHome, HaveMoney, HertzClosed, AvisOpen, McDonaldsOpen, HaveBike

3) Rainy, HaveMoutainBike, EnjoyPlayingSoccer, WorkForUniversity,

WorkCloseToHome, HaveMoney, HertzClosed, AvisOpen, McDonaldsOpen, HaveBike, CarRentalOpen

4) Rainy, HaveMoutainBike, EnjoyPlayingSoccer, WorkForUniversity,

WorkCloseToHome, HaveMoney, HertzClosed, AvisOpen, McDonaldsOpen, HaveBike, CarRentalOpen, CanRentCar, IsNotAHoliday

5) Rainy, HaveMoutainBike, EnjoyPlayingSoccer, WorkForUniversity,

WorkCloseToHome, HaveMoney, HertzClosed, AvisOpen, McDonaldsOpen, HaveBike, CarRentalOpen, CanRentCar, IsNotAHoliday, CanDriveToWork

6) Rainy, HaveMoutainBike, EnjoyPlayingSoccer, WorkForUniversity,

WorkCloseToHome, HaveMoney, HertzClosed, AvisOpen, McDonaldsOpen, HaveBike, CarRentalOpen, CanRentCar, IsNotAHoliday, CanDriveToWork, CanGetToWork

7) Rainy, HaveMoutainBike, EnjoyPlayingSoccer, WorkForUniversity,

WorkCloseToHome, HaveMoney, HertzClosed, AvisOpen, McDonaldsOpen, HaveBike, CarRentalOpen, CanRentCar, IsNotAHoliday, CanDriveToWork, CanGetToWork

CanGetToWork is generated from the given facts

1. 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.

CanGetToWork goal

CanBikeToWork pop CanGetToWork

HaveBike, WorkCloseToHome, Sunny pop CanBikeToWork

HaveMountainBike, WorkCloseToHome, Sunny pop HaveBike

HaveTenSpeed, WorkCloseToHome, Sunny backtrack; not provable

CanDriveToWork backtrack; not provable

OwnCar pop CanDriveToWork

CanRentCar backtrack; not provable

HaveMoney, CarRentalOpen pop CanRentCar

CarRentalOpen pop HaveMoney (fact)

HertzOpen pop CarRentalOpen

AvisOpen backtrack; not provable

{} pop AvisOpen

CanGetToWork is true since it resulted in an empty set through back chaining.