CSCE 420 – Fall 2023 Homework 3 (HW3)

due: Thurs, Nov 16, 2023, 11:59 pm

Turn-in answers as a Word document (HW3.docx or .pdf) and commit/push it to your class github repo (in your homework_3/ directory).

When pushing the final version of your HW1, also run the command:

git tag "HW3" && git push origin "HW3"

This will be used to record time of submission for late penalty when applicable.

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

I was not able to type universal and existential quantifier symbols so I used A and E instead

1. **Translate** the following sentences into First-Order Logic. Remember to break things down to simple concepts (with short predicate and function names), and make use of quantifiers. For example, don't say "tasteDelicious(someRedTomatos)", but rather: "Ex tomato(x)^red(x)^ taste(x,delicious)". See the lecture slides for more examples and guidance.

   - **bowling balls are sporting equipment**

Ax bowlingBall(x) -> sportingEquipment(x)

   - **horses are faster than frogs (there are many ways to say this in FOL; try expressing it this way: "all horses have a higher speed than any frog")**

Ax, y horse(x) ^ frog(y) -> speed(x) > speed(y)

   - **all domesticated horses have an owner**

Ax horse(x) ^ domesticated(x) -> Ey owner(y, x)

   - **the rider of a horse can be different than the owner**

Ex, y horse(x) ^ rides(y, x) ^ -owner(y, x)

   - **a finger is any digit on a hand other than the thumb**

Ax finger(x) -> Ey partOf(x, hand(y)) ^ -thumb(x)

- **an isosceles triangle is defined as a polygon with 3 edges connected at 3 vertices, where 2 (but not 3) edges have the same length**

Ax isoscelesTriangle(x) -> Ea, b, c, d, e, f edge(a) ^ edge(b) ^ edge(c) ^ vertex(d) ^ vertex(e) ^ vertex(f) ^ connected(a, b, d) ^ connected(b, c, e) ^ connected(c, a, f) ^ length(a) = length(b) ^ length(b) = length(c) ^ length(a) != length(c)

2. Convert the following first-order logic sentence into CNF:

A x person(x) ^ [Ez petOf(x,z) ^ Ay petOf(x,y)→dog(y)] → doglover(x)

[Ax person(x) ^ -[Ez petOf(x, z) ^ Ay petOf(x, y) -> dog(y)]] v doglover(x)
[Ax person(x) ^ -[-(Ez petOf(x, z) ^ Ay petOf(x, y)) v dog(y)]] v doglover(x)
[Ax person(x) ^ [Ez petOf(x, z) ^ Ay petOf(x, y) ^ -dog(y)]] v doglover(x)
[Ax person(x) ^ [petOf(x, F(x)) ^ Ay petOf(x, y) ^ -dog(y)]] v doglover(x)
[person(x) ^ petOf(x, F(x)) ^ petOf(x, y) ^ -dog(y)] v doglover(x)
[person(x) v doglover(x)] ^ [petOf(x, F(x)) v doglover(x)] ^ [petOf(x, y) v doglover(x)] ^ [-dog(y) v doglover(x)]

3. Determine whether or not the following pairs of predicates are **unifiable**. If they are, give the most-general unifier and show the result of applying the substitution to each predicate. If they are not unifiable, indicate why. Capital letters represent variables; constants and function names are lowercase. For example, 'loves(A,hay)' and 'loves(horse,hay)' are unifiable, the unifier is u={A/horse}, and the unified expression is 'loves(horse,hay)' for both.

- **owes(owner(X),citibank,cost(X)) owes(owner(ferrari),Z,cost(Y))**

unifier u = {X / ferrari, Y / ferrari, Z / citibank}

owes(owner(ferrari),citibank,cost(ferrari)) owes(owner(ferrari),citibank,cost(ferrari))

- **gives(bill, jerry, book21) gives(X,brother(X),Z)**

Not unifiable, we don't know if jerry is the brother of bill

- **opened(X,result(open(X),s0))) opened(toolbox,Z)**

{X / toolbox, Z / result(open(toolbox), s0)}

Opened(toolbox, result(open(toolbox), s0)) opened(toolbox, result(open(toolbox), s0))

4. Consider the following situation:

Marcus is a Pompeian.
All Pompeians are Romans.
Ceasar is a ruler.
All Romans are either loyal to Caesar or hate Caesar (but not both).
Everyone is loyal to someone.
People only try to assassinate rulers they are not loyal to.
Marcus tries to assassinate Caesar.

a) Translate these sentences to First-Order Logic.

1 Pompeian(Marcus)

2 Aa Pompenians(a) -> Roman(a)

3 Ruler(Ceasar)

4 Ab Roman(b) -> (Loyal(b, Caesar) ^ -Hate(b, Caesar)) v (-Loyal(b, Caesar) ^ Hate(b, Caesar))

5 Ac Ed Loyal(c, d)

6 Ae, f Ruler(f) ^ Attempts(Assassinate(e, f)) -> -Loyal(e, f)

7 Attempts(Assassinate(Marcus, Caesar))

b) Prove that ***Marcus hates Caesar*** using Natural Deduction. Label all derived sentences with the ROI and which prior sentences and unifier were used.

8 -Loyal(Marcus, Caesar) [MP 3, 7, 6] u={ f/Caesar, e/Marcus }

9 Roman(Marcus) [MP 1, 2] u={ a/Marcus }

10 (Loyal(Marcus, Caesar) ^ -Hates(Marcus, Caesar)) v (-Loyal(Marucs, Caesar) ^ Hate(Marucs, Caesar)) [MP 9, 4] u={b/Marcus}

11 (-Loyal(Marucs, Caesar) ^ Hate(Marucs, Caesar)) [DS 8, 11]

12 Hate(Marcus, Caesar) [AE 11]

c) <u>Convert all the sentences into CNF</u>

1 Pompeian(Marcus)

2 -Pompenians(a) v Roman(a)

3 Ruler(Ceasar)

4 -Roman(b) v Loyal(b, Caesar) v Hate(b, Caesar)

5 -Roman(g) v -Loyal(g, Caesar) v -Hate(g, Caesar)

6 -Roman(h) v -Hate(h, Caesar) v -Loyal(h, Caesar)

7 Loyal(c, F(x))

8 -Ruler(f) v -Attempts(Assassinate(e, f)) v -Loyal(e, f)

9 Attempts(Assassinate(Marcus, Caesar))

d) Prove that ***Marcus hates Caesar*** using <u>Resolution Refutation</u>.

10 -Hates(Marcus, Caesar)
11 -Roman(Marcus) v Loyal(Marcus, Caesar) [10, 4] u={b/Marcus}
12 Roman(Marcus) [1, 2] u={a/Marcus}
13 Loyal(Marcus, Caesar) [11, 12]
14 -Attempts(Assassinate(Marcus, Caesar)) v -Loyal(Marcus, Caesar) [8, 3] u={f/Caesar, e/Marcus}
15 -Loyal(Marcus, Caesar) [14, 9]
16 {} [13, 15]

5. Write a KB in First-Order Logic with rules/axioms for…

a. **Map-coloring** – every state must be exactly 1 color, and adjacent states must be different colors. Assume possible colors are states are defined using unary predicate like color(red) or state(WA). To say a state has a color, use a binary predicate, e.g. ‘color(WA,red)’.

Ax State(x) -> Ey Color(y) ^ Color(x, y) ^ (Az Color(z) ^ Color(x, z) -> y=z)

Ax State(x) -> Ey Color(y) ^ Color(x, y) ^ (Az State(z) ^ Adjacent(x, z) -> Ea Color(a) ^ Color(z, a) ^ a != y)

b. **Sammy’s Sport Shop** – include implications of facts like obs(1,W) or label(2,B), as well as constraints about the boxes and colors. Use predicate ‘cont(x,q)’ to represent that box x contains tennis balls of color q (where q could be W, Y, or B).

Ax, y Box(x) ^ Color(y) ^ Label(x, y) -> Ez Color(z) ^ Cont(x, z) ^ z != y // label incorrect

Ax, y Box(x) ^ Color(y) ^ Obs(x, y) -> Cont(x, y) v Cont(x, both) // content observed or both

Aa, b Box(a) ^ Color(b) ^ Label(a, b) -> Ax, y Box(x) ^ Color(y) ^ Label(x, y) ^ a != x ^ b != y // labels different

Aa, b Box(a) ^ Color(b) ^ Cont(a, b) -> Ax, y Box(x) ^ Color(y) ^ Cont(x, y) ^ a != x ^ b != y // contents different

Ax, y Box(x) ^ Color(y) ^ Label(x, y) -> Az Color(z) ^ (y = z v -Label(x, z)) // 1 label

Ax, y Box(x) ^ Color(y) ^ Cont(x, y) -> Az Color(z) ^ (y = z v -Cont(x, z)) // 1 content

Ax, y Box(x) ^ Color(y) ^ Obs(x, y) -> Az Color(z) ^ (y = z v -Obs(x, z)) // 1 observation

Ax Color(x) -> W v Y v B // either white, yellow, both

c. **Wumpus World** - (hint start by defining a helper concept ‘adjacent(x,y,p,q)’ which defines when a room at coordinates (x,y) is adjacent to another room at (p,q). Don’t forget rules for ‘stench’, ‘breezy’, and ‘safe’.

Ax, y Room(x, y) ^ Breezy(x, y) -> Ep, q Room(p, q) ^ Adjacent(x, y, p, q) ^ Pit(p, q)

Ax, y Room(x, y) ^ Stenchy(x, y) -> Ep, q Room(p, q) ^ Adjacent(x, y, p, q) ^ Wumpus(p, q)

Ax, y Room(x, y) ^ -Pit(x, y) ^ -Wumpus(x, y) <-> Safe(x, y)

d. **4-Queens** – assume row(1)…row(4) and col(1)…col(4) are facts; write rules that describe configurations of 4 queens such that none can attack each other, using ‘queen(r,c)’ to represent that there is a queen in row r and col c.

Ar, c Row(r) ^ Col(c) ^ Queen(r, c) -> Ax Col(x) ^ (-Queen(r, x) v x=c) // each row has exactly 1 queen

Ar, c Row(r) ^ Col(c) ^ Queen(r, c) -> Ax Row(x) ^ (-Queen(x, c) v x=r) // each col has exactly 1 queen

Ar, c Row(r) ^ Col(c) ^ Queen(r, q) -> An ^ int(n) ^ n>0 ^ -Queen(r+n, c+n) ^ -Queen(r+n, c-n) ^ -Queen(r-n, c+n) ^ -Queen(r-n, c-n) // no queens in diagonal

Ar Row(r) -> Ec Q(r, c) // each row has a queen

Ac Col(c) -> Er Row(r) ^ Q(r, c) // each col has a queen

Don’t forget to quantify all your variables.