CSCE 420 – Fall 2025

Homework 4 (HW4)

due: Tues, Nov 11, 2025, 11:59 pm

Turn-in answers as a PDF document (HW4.pdf) and commit/push it to your class github repo (in your HW4/ directory).

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

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: "∃x tomato(x)^red(x)^ taste(x,delicious)". See the lecture slides for more examples and guidance.

- **bowling balls are sporting equipment**

  ∀x (bowlingBall(x) →sportingEquipment(x))

  Explanation: For all bowling balls x, we can imply that x is a sporting equipment

- **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")**

  ∀x (Horse(x) → ∀y (Frog(y) → (speed(x)>speed(y))))

  Explanation: For ALL horses x, we can imply that for ALL frogs y, the speed of x will be greater than the speed of y

- **all domesticated horses have an owner**

  ∀x(domesticatedHorse(x) → ∃y(Owner(y,x)))

  Explanation: For ALL domesticated horses x, we can assume there exists an owner y

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

  ∀x (Horse(x) ^ ∃y,z (Rider (y,x,) ^ Owner(y,x) ^ y≠z))

  Explanation: For all horses, there exists a rider y and owner z that do not have to be the same person

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

  ∀x(Digit(x) ^ ¬Thumb(x) ^ onHand(x) → Finger(x))

Explanation: For every x that is a digit on a hand that is NOT a thumb, we can imply that x is a finger

- **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**

$\forall$x(Triangle (x) ^ Edges (x, 3) ^ Vertices(x,3) ^ $\exists$y1,y2,y3 (Edge(y1,x) ^ Edge(y2,x) ^ Edge(y3, x) ^ ((Length(y1) = Length(y2)) $\neq$ Length(y3)) v (Length(y2)= Length(y3) $\neq$ Length(y1)) v (Length(y1) = Length(y3) $\neq$ Length(y2)) $\rightarrow$ Isosceles(x)

Explanation: For all triangles that have 3 edges and 3 vertices we can imply a set of conditions that must be met for the triangle to be isosceles. This includes an existence of edges y1, y2, and y3 in the triangle x where at least 2 of the edges have the same length and don't equal the third edge.

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

∀ x (person(x) ^ [∃z petOf(x,z) ^ [∀y petOf(x,y)→dog(y)]]) → doglover(x)

Steps:

1. **Get rid of the first implication**
∀ x (person(x) ^ [∃z petOf(x,z) ^ [∀y ¬petOf(x,y) v dog(y)]]) → doglover(x)
2. **Get rid of the last implication**
∀x(¬(person(x) ^ [∃z petOf(x,z) ^ [∀y ¬petOf(x,y) v dog(y)]]) v doglover(x)
3. **Move ¬ negation inwards**
∀**x(**¬person(x) v [∀z ¬petOf(x,z)] v [∃y petOf(x,y) ^ ¬dog(y)] v doglover(x)
4. **Skollemization?? E(y) turns into f(x,z) by using the variables that came before**
∀**x(**¬person(x)] v [∀z ¬petOf(x,z)] v [petOf(x,f(x,z)) ^ ¬dog(f(x,z))] v doglover(x)
5. **Factor out the quantifiers**
∀**x**∀**z[**¬person(x) v ¬petOf(x,z) v petOf(x,f(x,z)) ^ ¬dog(f(x,z))] v doglover(x)
6. **Drop the quantifiers because they are universal**
¬person(x) v ¬petOf(x,z) v (petOf(x,f(x,z)) ^ ¬dog(f(x,z)) v doglover(x)
7. **Distribute ^ over v**
(¬person(x) v ¬petOf(x,z) v petOf(x,f(x,z)v doglover(x)) ^ (¬person(x) v ¬petOf(x,z) v ¬dog(f(x,z)) v doglover(x))
8. **Finished, this clause is now in CNF form and is a conjunction of disjunctive clauses**

Keep in mind that the quantifiers have *lower* precedence than all the other operators in FOL sentences.

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))**

  Yes this sentence can be unified by using u={X/ferrari, Z/citibank, Y/ferrari}

  The unified expression is **owes(owner(ferrari),citibank,cost(ferrari))**

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

  No, we cannot unify this sentence. We would have to substitute brother(X) with jerry in order to match the second constant in the predicate. However, we cannot directly substitute a constant for a function. Without knowing if jerry is truly the brother of bill, this sentence is not unifiable.

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

Yes this sentence can be unified by using u= {X/toolbox, Z/result(open(X),s0)} or u={X/toolbox, Z/result(open(toolbox),s0)}

The unified expression is **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. Note*:These translations and proofs are based on our class notes First-Order Logic PowerPoint regarding Colonel West.
   1. Marcus is a Pompeian translates to…
      Pompeian(Marcus)
   2. All Pompeians are Romans translates to…
      ∀x(Pompeian(x) →Roman(x) ^ People(x))   * added in People so the proof works because Pompeians are both Romans AND people
   3. Caesar is a ruler translates to…
      Ruler(Caesar)
   4. All Romans are either loyal to Caesar or hate Caesar (but not both) translates to
      ∀x(Roman(x) -> (Loyal(x, Caesar) ^ ¬Hate(x, Caesar) v (Hate(x,Caesar) ^ ¬Loyal(x,Caesar))
   5. Everyone is loyal to someone translates to …
      ∀x ∃y Loyal(x,y)
   6. People only try to assassinate rulers they are not loyal to translates to …
      ∀x∀y(People(x) ^ Ruler(y) ^ Assassinates (x, y) → ¬Loyal(x,y))
   7. Marcus tries to assassinate Caesar
      Assassinates(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.
   Steps:
   8. Roman(Marcus): **Modus ponens using rules 1 and 2**
   9. People(Marcus) : **Modus Ponens rule 1 and 2**
   10. (Loyal(Marcus, Caesar) ^ ¬Hate(Marcus, Caesar)) v (Hate(Marcus,Caesar) ^ ¬Loyal(Marcus,Caesar)): **Modus Ponens rule 4 and 8**
   11. ¬Loyal(Marcus,Caesar): **Modus Ponens rules 3, 9, 7, 6**

12. Hate(Marcus,Caesar) **: Resolve on rule 11 and 10, if Marcus is not loyal to Caesar than the clause Hate(Marcus, Caesar) must be true.**

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

1. Marcus is a Pompeian translates to… (already in CNF)
   Pompeian(Marcus)
2. All Pompeians are Romans translates to… using implication elimination
   ¬Pompeian(x) v (Roman(x) ^ People(x))
3. Caesar is a ruler translates to… (already in CNF)
   Ruler(Caesar)
4. All Romans are either loyal to Caesar or hate Caesar (but not both) translates to… using implication elimination
   ¬Roman(x) v ((Loyal(x, Caesar) ^ ¬Hate(x, Caesar) v (Hate(x,Caesar) ^ ¬Loyal(x,Caesar)))
5. Everyone is loyal to someone translates to … using Skolemization
   Loyal(x,f(x))
6. People only try to assassinate rulers they are not loyal to translates to …

   ¬People(x) ^ ¬Ruler(y) ^ ¬Assassinates (x, y) v ¬Loyal(x,y)
7. Marcus tries to assassinate Caesar translates to… (already in CNF)

   Assassinates(Marcus, Caesar)

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

Steps (assuming x is Marcus and y is Caesar):
8. Negate the Conclusion : **¬Hate(Marcus, Caesar)**
9. Resolve on rule 1 and 2 to get **Roman(Marcus)**
10. Resolve on rule 1 and 2 to get **People(Marcus)**
11. Resolve rule 9 and 4 to get **(Loyal(x, Caesar) ^ ¬Hate(x, Caesar) v (Hate(x,Caesar) ^ ¬Loyal(x,Caesar)**
12. Resolve rule 10, 3, 7, and 6 to get **¬Loyal(x,y)**
13. Split rule 4 into 2 clauses using distribution. Clause 1 is **(Loyal(Marcus, Caesar)vHate(Marcus,Caesar))**
14. Split rule 4 into 2 clauses. Clause 2 is (**¬Hate(x, Caesar) v ¬Loyal(x,Caesar))**
15. Resolve rule 12 and rule 13 to get **Hate(Marcus,Caesar)**
16. If we resolved the negated conclusion rule 8 **¬Hate(Marcus, Caesar)** with rule 15 **Hate(Marcus,Caesar)** we will get **Ø which is a contradiction, we have proved our conclusion!**

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)'.

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).

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'.

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.

Don't forget to quantify all your variables.

A) Map-coloring FOL Knowledge base for Australia:

***Note this KB is directly given in our class notes so I have directly referenced that**

Define the neighbors:

neighbor(WA, NT)

neighbor(WA,SA)

neighbor(NT,SA)

neighbor(NT,Q)

neighbor(SA,NSW)

neighbor(SA,Q)

neighbor(SA,V)

neighbor(Q, NSW)

neighbor(NSW,V)

Define the allowed colors:

color(R) color(G) color(B)

Define the existing states:

state(WA) state(NT) state(SA) state(Q) state(NSW) state(V) state(T)

Each state has a color:

$\forall s\ state(s) \rightarrow \exists c\ color(c) \wedge hasColor(s,c)$

Each state can have only 1 color assigned:

$\forall$s,c,d state(s)^hasColor(s,c)^hasColor(s,d)$\rightarrow$c=d

If 2 states are neighbors then they cannot be the same color:

$\forall$s,t,c state(s)^state(t)^neigh(s,t)^hasColor(s,c) $\rightarrow\neg$hasColor(t,c)

B) Sammy's Sport Shop KB using the example from HW 3:

Define the colors:

color(Y) color(W) color(B)

Direct Observations:

Observe(1,Y) Observe(2,W) Observe(3, Y)

Label(1,W) Label(2,Y) Label(3,B)

Each box has a color:

$\forall$x (Box(x) $\rightarrow\exists$q (color(q)^ Cont(x,q)))

What is on the label cannot be contained in the box:

$\forall$x,q(Label(x,q) $\rightarrow$ $\neg$Cont(x,q))

2 boxes cannot contain the same color:

$\forall$x,y,q1, q2 (Cont(x,q1) ^ Cont(y,q2) $\rightarrow$ q1 $\neq$ q2

If observed ball is Y then box can contain either Y or B:

$\forall$x(Observed(x,Y) $\rightarrow$ Cont(x, Y) v Cont(x, B))

If observed ball is W then box can contain either W or B:

$\forall$x(Observed(x,W) $\rightarrow$ Cont(x, W) v Cont(x, B))

C) Wumpus World Knowledge Base:

***added references per Professor Ioeger's request**

Referencing John Hopkins University FOL lecture:
https://www.cs.jhu.edu/~phi/ai/slides/lecture-first-order-logic.pdf

Define if (x,y) is adjacent to (p,q):

adjacent(x,y,p,q)

Define the state of a room:

Stench(x,y) Breezy(x,y) Safe(x,y)

Define what can be in a room:

Pit(x,y) Wumpus(x,y) Gold(x,y)

If a room is breezy, then there exists a room adjacent to it that has a pit:

$\forall x,y$ Breezy(x,y) → ∃ p,q P it(p,q) ^ Adjacent(x, y,p,q)

If a room is stenchy, then there exists a room adjacent to it that has a Wumpus:

$\forall x,y$ Stench(x,y) → ∃ p,q Wumpus(p,q) ^ Adjacent(x, y,p,q)

If a room is stenchy and breezy, then the room is safe:

$\forall x,y$ Stench(x,y) ^ Breezy(x,y) → Safe(x,y)

D) 4 Queens Knowledge Base:

***added references per Professor Ioeger's request**

Referencing University of Waterloo FOL lecture:
https://cs.uwaterloo.ca/~ppoupart/teaching/cs486-fall08/slides/Lecture6notes.pdf

Define the state of a queen at coordinates row x, col y:

Queen(x,y)

2 queens cannot be in the same row (if the row x is the same, then it doesn't matter what col a queen can be placed in because it will still be in the same row, violating rules:

$\forall x,y1,y2$ Queen(x,y1) ^ $y1 \neq y2$ → ¬Queen(x,y2)

2 queens cannot be in the same col (if the col y is the same, then it doesn't matter what row a queen can be placed in because it will still be in the same col, violating rules:

$\forall x1,x2,y$ Queen(x1,y) ^ $x1 \neq x2$ → ¬Queen(x2,y)

There exists a queen in every row and every column

$\forall x\ \exists y$ Queen(x,y)

$\forall y\ \exists x$ Queen(x,y)