Octavio Almanzana
UIN: 628005492

Homework 3

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: "$\exists x$ tomato(x)∧red(x)∧taste(x,delicious)". See the lecture slides for more examples and guidance.

- bowling balls are sporting equipment
  - $\forall x$ $\textbf{bowlingBall}(x) \rightarrow \textbf{sportEquipment}(x)$
- 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")
  - $\forall x, y\{\textbf{horse}(x) \wedge \textbf{frog}(y) \rightarrow \textbf{greaterThan}(\textbf{speed}(x), \textbf{speed}(y))\}$
- all domesticated horses have an owner
  - $\forall x$ $\{\textbf{horse}(x) \wedge \textbf{domesticated}(x) \rightarrow \exists y$ $\{\textbf{person}(y) \wedge \textbf{owns}(y, x)\}\}$
- the rider of a horse can be different than the owner
  - $\forall x, y$ $\textbf{horse}(x) \wedge \textbf{person}(y) \wedge \textbf{rides}(y, x) \rightarrow \textbf{owns}(y, x) \vee [\exists z$ $\textbf{person}(z) \wedge (z \neq y) \wedge \textbf{owns}(z, x)]$
- a finger is any digit on a hand other than the thumb
  - $\forall x$ $\textbf{finger}(x) \wedge \neg \textbf{thumb}(x) \rightarrow \textbf{digit}(x)$
- 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
  - $\forall_{x,y,z}$ $\textbf{unique}(x, y, z) \Leftrightarrow (x \neq y) \wedge (y \neq z) \wedge (x \neq z)$
  - $\forall_x$ $\textbf{threeVerticesOf}(v1, v2, v3, x) \Leftrightarrow \textbf{unique}(v1, v2, v3) \wedge \textbf{vertexOf}(v1, x) \wedge \textbf{vertexOf}(v2, x) \wedge \textbf{vertexOf}(v3, x) \wedge [\forall_v$ $\textbf{vertexOf}(v, x) \rightarrow (v = v1) \vee (v = v2) \vee (v = v3)]$
  - $\forall_x$ $\textbf{threeEdgesOf}(e1, e2, e3, x) \Leftrightarrow \textbf{unique}(e1, e2, e3) \wedge \textbf{edgeOf}(e1, x) \wedge \textbf{edgeOf}(e2, x) \wedge \textbf{edgeOf}(e3, x) \wedge [\forall_e$ $\textbf{edgeOf}(e, x) \rightarrow (e = e1) \vee (e = e2) \vee (e = e3)]$
  - $\forall_x$ $\textbf{triangle}(\textbf{x}) \Leftrightarrow \textbf{polygon}(x) \wedge \exists_{v1,v2,v3,e1,e2,e3}$ $\textbf{threeVerticesOf}(v1, v2, v3, x) \wedge \textbf{three-EdgesOf}(e1, e2, e3, x) \wedge \textbf{connects}(v1, e1, e2) \wedge \textbf{connects}(v2, e2, e3) \wedge \textbf{connects}(v3, e3, e1)$
  - $\forall_x$ $\textbf{isoscelesTriangle}(\textbf{x}) \Leftrightarrow \textbf{triangle}(x) \wedge \exists_{e1,e2,e3}$ $\textbf{threeEdgesOf}(e1, e2, e3, x) \wedge (\textbf{length}(e1) = \textbf{length}(e2)) \wedge (\textbf{length}(e1) \neq \textbf{length}(e3))$

2. Convert the following first-order logic sentence into CNF:
$\forall x$ **person(x)** $\wedge$ [$\exists z$ **petOf(x, z)** $\wedge$ $\forall y$ **petOf(x, y)$\rightarrow$dog(y)**] $\rightarrow$ **doglover(x)**

i. Implication Elimination

- $\forall x$ $\neg$(person(x) $\wedge$ [$\exists z$ petOf(x, z) $\wedge$ $\forall y$ petOf(x, y)$\rightarrow$dog(y)]) $\vee$ dogLover(x)

ii. DeMorgan's Rule

- $\forall x$ $\neg$person(x) $\vee$ $\neg$[$\exists z$ petOf(x, z) $\wedge$ $\forall y$ petOf(x, y)$\rightarrow$dog(y)] $\vee$ dogLover(x)

iii. Implication Elimination

- $\forall x$ $\neg$person(x) $\vee$ $\neg$[$\exists z$ petOf(x, z) $\wedge$ $\forall y$ $\neg$petOf(x, y) $\vee$ dog(y)] $\vee$ dogLover(x)

iv. DeMorgan's Rule

- $\forall x$ $\neg$person(x) $\vee$ $\forall z$ $\neg$petOf(x, z) $\vee$ $\exists y$ petOf(x, y) $\wedge$ $\neg$dog(y) $\vee$ dogLover(x)

v. Variable standardization is unecessary since variables do not repeat in unrelated predicates.

vi. Skolemization. Since y is an existentially quantified variable whose scope is within the universal quantifier x, we replace y with the Skolem function $Y(x)$ and remove y's existential quantifier.

- $\forall x$ $\neg$person(x) $\vee$ $\forall z$ $\neg$petOf(x, z) $\vee$ petOf($x, Y(x)$) $\wedge$ $\neg$dog($Y(x)$) $\vee$ dogLover(x)

vii. Drop universal quantifiers

- $\neg$person(x) $\vee$ $\neg$petOf(x, z) $\vee$ petOf($x, Y(x)$) $\wedge$ $\neg$dog($Y(x)$) $\vee$ dogLover(x)

viii. Apply the commutative property

- $\neg$person(x) $\vee$ $\neg$petOf(x, z) $\vee$ dogLover(x) $\vee$ [petOf($x, Y(x)$) $\wedge$ $\neg$dog($Y(x)$)]

ix. Distribute $\vee$ over $\wedge$

- [$\neg$person(x) $\vee$ $\neg$petOf(x, z) $\vee$ dogLover(x) $\vee$ petOf($x, Y(x)$)] $\wedge$ [$\neg$person(x) $\vee$ $\neg$petOf(x, z) $\vee$ dogLover(x) $\vee$ $\neg$dog($Y(x)$)]

x. Apply and elimination. We finish with two clauses in CNF:

- **$\neg$person(x)** $\vee$ **$\neg$petOf(x, z)** $\vee$ **dogLover(x)** $\vee$ **petOf($x, Y(x)$)**
- **$\neg$person(x)** $\vee$ **$\neg$petOf(x, z)** $\vee$ **dogLover(x)** $\vee$ **$\neg$dog($Y(x)$)**

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

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

- Yes, the pair of predicates above are unifiable.
- We can use the unifier $u =$**{**X**/ferrari,** Z**/Citibank,** Y**/ferrari}**
- The unified expression is: **owes(owner(ferrari), citibank, cost(ferrari))**

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

- The pair of predicates above are not unifiable. This is because we cannot bind jerry to brother(X) or vice-versa since they are both constants.

opened(X**,result(open(**X**),** $s0$**))** and **opened(toolbox,**Z**)**

- Yes, the pair of predicates above are unifiable.
- We can use the unifier 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.
Caesar 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. $\forall X$ Pompeian(X) $\rightarrow$ Roman(X)
3. Ruler(caesar)
4. $\forall X$ Roman(X) $\rightarrow$ (Loyal(X, caesar) $\vee$ Hate(X, caesar)) $\wedge$ ($\neg$Loyal(X, caesar) $\vee$ $\neg$Hate(X, caesar))
5. $\forall X \forall Y$ Ruler(Y) $\wedge$ triesToAssasinate(X, Y) $\rightarrow$ $\neg$Loyal(X,Y)
6. triesToAssasinate(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.

7. **Roman(marcus)** *by generalized Modus Ponens on sentences 1 and 2.* $\theta = \{X/\text{marcus}\}$
8. **Ruler(caesar)** $\wedge$ **triesToAssasinate(marcus, caesar)** *by applying And Introduction on sentences 3 and 6*
9. **Ruler(caesar)** $\wedge$ **triesToAssasinate(marcus, caesar)** $\rightarrow$ $\neg$**Loyal(marcus,caesar)** *by applying universal instantiation on sentence 5.* $\theta = \{\text{X/marcus, Y/caesar}\}$
10. $\neg$**Loyal(marcus,caesar)** *by applying modus ponens on 8 and 9.*
11. **Roman(marcus)** $\rightarrow$ **(Loyal(marcus, caesar)** $\vee$ **Hate(marcus, caesar))** $\wedge$ **(**$\neg$**Loyal(marcus, caesar)** $\vee$ $\neg$**Hate(marcus, caesar))** *by applying universal instantiation on sentence 4.* $\theta = \{\text{X/marcus}\}$
12. **(Loyal(marcus, caesar)** $\vee$ **Hate(marcus, caesar))** $\wedge$ **(**$\neg$**Loyal(marcus, caesar)** $\vee$ $\neg$**Hate(marcus, caesar))** *by applying Modus Ponens on sentences 7 and 11.*
13. **Loyal(marcus, caesar)** $\vee$ **Hate(marcus, caesar)** *by applying And Elimination on sentence 12.*
14. $\neg$**Loyal(marcus, caesar)** $\rightarrow$ **Hate(marcus, caesar)** *by applying implication introduction on sentence 13.*
15. **Hate(marcus, caesar)** *by applying modus ponens on sentence 10 and 14.*

c.) Convert all the sentences into CNF

1. Pompeian(marcus)
2. ¬Pompeian(X) ∨ Roman(X)
3. Ruler(caesar)
4. ¬Roman(X) ∨ Loyal(X, caesar) ∨ Hate(X, caesar)
5. ¬Roman(X) ∨ ¬Loyal(X, caesar) ∨ ¬Hate(X, caesar)
6. ¬Ruler(Y) ∨ ¬triesToAssasinate(X, Y) ∨ ¬Loyal(X,Y) *after dropping universal quantifiers*
7. triesToAssasinate(marcus, caesar)

Note: Rules 4 and 5 are a result of applying And Elimination during the conversion of the original sentence 4 to CNF.

d.) Prove that Marcus hates Caesar using Resolution Refutation.
Note: For the following proof, I used the numbered rules obtained from 4c.
Negate the query:

- ¬Hate(marcus, caesar)

Apply resolution with clause 4 **and with substitution** θ =**{**X**/marcus}**

- ¬Roman(marcus) ∨ Loyal(marcus,caesar)

Apply resolution with clause 2 **and with substitution** θ =**{**X**/marcus}**

- ¬Pompeian(marcus) ∨ Loyal(marcus,caesar)

Apply resolution with clause 1

- Loyal(marcus,caesar)

Apply resolution with rule 6 **with substitution** θ =**{**X**/marcus,** Y**/caesar}**

- ¬Ruler(caesar) ∨ ¬triesToAssasinate(marcus, caesar)

Apply resolution with rule 3

- ¬triesToAssasinate(marcus, caesar)

Apply resolution with rule 8

- $\emptyset$

Since we were able to derive the empty clause $\emptyset$ from the premise **¬Hate(marcus, caesar)**, we have proven that the clause **¬Hate(marcus, caesar)** causes an inconsistency in the knowledge base, and consequently proved **Hate(marcus, caesar)** is true

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)'.

- color(Red), color(Green), color(Blue)
- state(WA), state(NT), state(SA), state(Q), state(NSW), state(V), state(T)
- neighbor(WA, NT), neighbor(WA, SA)
- neighbor(NT, WA), neighbor(NT, SA), neighbor(NT,Q)
- neighbor(SA,WA), neighbor(SA,NT), neighbor(SA,Q), neighbor(SA,NSW), neighbor(SA,V)
- neighbor(Q,NT), neighbor(Q,SA), neighbor(Q,NSW)
- neighbor(NSW,Q), neighbor(NSW,SA), neighbor(NSW,V)
- neighbor(V,SA), neighbor(V,NSW)
- $\forall_s$ state(s) $\rightarrow \exists_c$ color(c) $\wedge$ isColor(s, c)
- $\forall_{s1,s2,c}$ state$(s1)$ $\wedge$ state$(s2)$ $\wedge$ neighbor(s1, s2) $\wedge$ color(c) $\wedge$ isColor$(s1, c) \rightarrow \neg$ isColor$(s2, c)$
- $\forall_{s,c1,c2}$ state(s) $\wedge$ isColor$(s, c1)$ $\wedge$ isColor$(s, c2) \rightarrow c1 = c2$
- $\forall_{s,c1,c2,c3}$ state(x) $\wedge$ color$(c1)$ $\wedge$ color$(c2)$ $\wedge$ color$(c3)$ $\wedge \neg$isColor$(s, c1)$ $\wedge \neg$isColor$(s, c2) \rightarrow$ isColor$(s, c3)$

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).

- color(W), color(Y), color(B)
- box(1), box(2), box(3)
- $\forall_b$ box$(b) \rightarrow \exists_c$ color(c) $\wedge$ cont(b, c)
- $\forall_{b,c}$ box(b) $\wedge$ color(c) $\wedge$ obs(b,c) $\Leftrightarrow$ cont(b,c) $\vee$ cont(b,B)
- $\forall_{b,c}$ box(b) $\wedge$ color(c) $\wedge$ label(b, c) $\rightarrow \neg$cont(b,c)
- $\forall_{b1,b2,c}$ box(b1) $\wedge$ box(b2) $\wedge$ color(c) $\wedge$ cont(b1,c) $\rightarrow \neg$cont(b2,c)
- $\forall_{b,c1}$ box(b) $\wedge$ color(c1) $\wedge$ cont(b,c1) $\rightarrow \forall_{c2}$ c2 != c1 $\neg$cont(b,c2)
- $\forall_b$ box(b) $\rightarrow \exists_c$ color(c) $\wedge$ (c$\neq$B) $\wedge$ obs(b,c)
- $\forall_b$ box(b) $\rightarrow \exists_c$ color(c) $\wedge$ label(b,c)

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'.

Note: For the following problem, I assummed that the wumpus, the pits, and the gold would all be in different coordinates.

- $\forall_{x,y,p,q}$ adjacent$(x, y, p, q) \Leftrightarrow (x = p \wedge (y = q - 1 \vee y = q + 1)) \vee (y = q \wedge (x = p - 1 \vee x = p + 1))$
- $\forall_{x,y}$ wumpus$(x, y) \Leftrightarrow \forall_{p,q}$ adjacent$(x, y, p, q) \wedge$ stench$(p, q$
- $\forall_{x,y}$ pit$(x, y) \Leftrightarrow \forall_{p,q}$ adjacent$(x, y, p, q) \wedge$ breezy(p, q)
- $\forall_{x,y}$ gold$(x, y) \Leftrightarrow \forall_{p,q}$ adjacent$(x, y, p, q) \wedge$ glitter(p, q)
- $\forall_{x,y}$ stench$(x, y) \Rightarrow \exists_{p,q}$ adjacent$(x, y, p, q) \wedge$ wumpus(p, q)
- $\forall_{x,y}$ breezy$(x, y) \Rightarrow \exists_{p,q}$ adjacent$(x, y, p, q) \wedge$ pit(p, q)
- $\forall_{x,y}$ glitter$(x, y) \Rightarrow \exists_{p,q}$ adjacent$(x, y, p, q) \wedge$ gold(p, q)
- $\forall_{x,y}$ safe$(x, y) \Leftrightarrow \neg$wumpus(x,y) $\wedge$ $\neg$pit(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. Don't forget to quantify all your variables.

- $\forall_r$ row$(r) \rightarrow \exists_c$ col(c) $\wedge$ queen(r, c)
- $\forall_c$ col$(c) \rightarrow \exists_r$ row(r) $\wedge$ queen(r, c)
- $\forall_{r1,c1}$ queen$(r1, c1) \rightarrow \forall_{r2,c2}$ row$(r2) \wedge$ col$(c2) \wedge (r1 < r2) \wedge (c1 < c2) \wedge (r2 - r1 = c2 - c1)$ $\wedge \neg$queen$(r2, c2)$
- $\forall_{r1,c1}$ queen$(r1, c1) \rightarrow \forall_{r2,c2}$ row$(r2) \wedge$ col$(c2) \wedge (r1 > r2) \wedge (c1 > c2) \wedge (r1 - r2 = c1 - c2)$ $\wedge \neg$queen$(r2, c2)$
- $\forall_{r1,c1}$ queen$(r1, c1) \rightarrow \forall_{r2,c2}$ row$(r2) \wedge$ col$(c2) \wedge (r1 < r2) \wedge (c1 > c2) \wedge (r2 - r1 = c1 - c2)$ $\wedge \neg$queen$(r2, c2)$
- $\forall_{r1,c1}$ queen$(r1, c1) \rightarrow \forall_{r2,c2}$ row$(r2) \wedge$ col$(c2) \wedge (r1 > r2) \wedge (c1 < c2) \wedge (r1 - r2 = c2 - c1)$ $\wedge \neg$queen$(r2, c2)$
- $\forall_{r1,c1,r2,c2}$ queen$(r1, c1) \wedge$ queen$(r2, c2) \wedge (r1 = r2) \rightarrow (c1 = c2)$
- $\forall_{r1,c1,r2,c2}$ queen$(r1, c1) \wedge$ queen$(r2, c2) \wedge (c1 = c2) \rightarrow (r1 = r2)$