CSCE 625: ARTIFICIAL INTELLIGENCE: HOMEWORK 2

- ANIKET SANJIV BONDE

Q1) Translate the following sentences to FOL:

- Tomatoes are either a fruit or vegetable.

$$\forall x\ tomatoes(x) \Rightarrow fruit(x) \lor vegetable(x)$$

- Some mushrooms are poisonous.

$$\exists x\ mushrooms(x) \land poisonous(x)$$

- Define 'triangle'

$$\forall a\ \forall b\ \forall c\ \ point(a) \land point(b) \land point(c) \land \neg equals(a,b) \land \neg equals(a,c) \land \neg equals(b,c) \land line\_connected(a,b) \land line\_connected(a,c) \land line\_connected(b,c) \land \neg colinear(a,b,c) \Rightarrow triangle(a,b,c)$$

- A plant can only produce seeds after it has been pollinated.

$$\forall p\ \forall s\ plant(p) \land seed(s) \land part\_of(s,p) \Rightarrow after(polination(p), produce(p,s))$$

- John's favourite movies are any movie by Stephen King except Cujo.

$$\forall x\ movies(x) \land bystephenking(x) \land \neg equals(x, Cujo) \Rightarrow favourite(John, x)$$

- The winner of a football game is the team that has the most points at the end.

$$\forall g\ \exists x\ \forall y\ team(x) \land football(g) \land team(y) \land plays(x,g) \land plays(y,g) \land (points(x,end) > points(y,end)) \land winner(x)$$

- The warning light of a Ford Exporer will be on when its gas tank is more than 90% empty.

$$\forall x\ \ fordexporer(x) \land (gas\_in(gastank(x)) < 0.1 * capacity(gastank(x))) \Rightarrow on(warninglight(x))$$

- Al and Bob bought their computers from the same manufacturer.

$$\exists m\ bought(A1, computer, m) \land bought(Bob, computer, m)$$

- All laptops sold by Dell in 2012 have at least 4 gigabytes of memory

$$\forall x\ laptop(x) \land soldby(x, Dell) \land sell\_date(x, in(2012)) \Rightarrow part\_of(x, (memory(x) > 4GB))$$

Q2) Convert the following sentence to CNF:

- $\forall x\, P(x) \implies [\forall y\, P(y) \implies P(f(x, y))] \land [\neg\forall y\, Q(x, y) \implies P(y)]$

By Implication Elimination rule inside square brackets:

- $\forall x\, P(x) \implies [\forall y\, \neg P(y) \lor P(f(x, y))] \land [\neg\forall y\, \neg Q(x, y) \lor P(y)]$

By Implication Elimination rule:

- $\forall x\, \neg P(x) \lor [\forall y\, \neg P(y) \lor P(f(x, y))] \land [\neg\forall y\, \neg Q(x, y) \lor P(y)]$

By moving $\neg$ inwards or by ($\neg\forall x\ \ p\ \equiv\ \exists x\ \ \neg p$) rule:

- $\forall x\, \neg P(x) \lor [\forall y\, \neg P(y) \lor P(f(x, y))] \land [\exists y\, Q(x, y) \land \neg P(y)]$

By changing variable name in second square bracket to avoid confusion (standardizing variables):

- $\forall x\, \neg P(x) \lor [\forall y\, \neg P(y) \lor P(f(x, y))] \land [\exists z\, Q(x, z) \land \neg P(z)]$

By Skolemization, where $F(x)$ is a Skolem function:

- $\forall x\, \neg P(x) \lor [\forall y\, \neg P(y) \lor P(f(x, y))] \land [Q(x, F(x)) \land \neg P(F(x))]$

Moving all universal quantifiers to left side:

- $\forall x\, \forall y\, \neg P(x) \lor [\neg P(y) \lor P(f(x, y))] \land [Q\big(x, F(x)\big) \land \neg P(F(x))]$

As all the universal quantifiers are to the left side, we can drop them:

- $\neg P(x) \lor [\neg P(y) \lor P(f(x, y))] \land [Q\big(x, F(x)\big) \land \neg P(F(x))]$

Expanding:

- $[\neg P(x) \lor \neg P(y) \lor P(f(x, y))] \land [\neg P(x) \lor [Q\big(x, F(x)\big) \land \neg P(F(x))]]$
- $[\neg P(x) \lor \neg P(y) \lor P(f(x, y))] \land [\neg P(x) \lor Q\big(x, F(x)\big)] \land [\neg P(x) \lor \neg P(F(x))]$

Thus, final CNF form, where $F(x)$ is a Skolem function:

- $[\neg P(x) \lor \neg P(y) \lor P(f(x, y))] \land [\neg P(x) \lor Q\big(x, F(x)\big)] \land [\neg P(x) \lor \neg P(F(x))]$

Q3) 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.

3a) Translate these sentences to First-Order Logic.

Marcus is a Pompeian:

1) $pompeian(marcus)$

All Pompeians are Romans:

2) $\forall x\ pompeian(x) \Longrightarrow roman(x)$

Caesar is a ruler:

3) $ruler(caeser)$

All Romans are either loyal to Caesar or hate Caesar (but not both)

4) $\forall x\ [roman(x) \land loyal(caeser, x)] \Longrightarrow\ \neg hates(x, caeser)$

5) $\forall x\ [roman(x) \land \neg loyal(caeser, x)] \Longrightarrow\ hates(x, caeser)$

Everyone is loyal to someone.

6) $\forall x\ person(x) \Longrightarrow\ \exists y\ loyal(x, y) \land person(y)$

People only try to assassinate rulers they are not loyal to.

7) $\forall x\ \forall y\ assassinate(x, y) \land person(x) \land person(y) \land ruler(y)\ \Longrightarrow \neg loyal(y, x)$

Marcus tries to assassinate Caesar.

8) $assassinate(marcus, caeser)$

3b & 3c) Prove that **Marcus hates Caesar** using Natural Deduction. Label all derived sentences with the prior sentences and unifier used:

We need to prove that Knowledge Base entails $\boldsymbol{hates(marcus, caeser)}$

By applying Generalized Modus Ponens on rule no. 1 and 2, we get

Unifier: $\theta = \{x/marcus\}$

9) $roman(marcus)$

By applying Generalized Modus Ponens on rule no. 3, 7 and 8, we get

Unifier: $\theta = \{x/marcus, y/caeser\}$

10) $\neg loyal(caeser, marcus)$

By applying Generalized Modus Ponens on rule no. 5, 9 and 10, we get

Unifier: $\theta = \{x/marcus\}$

11) $hates(marcus, caeser)$

Hence, proved that Knowledge Base entails $\boldsymbol{hates(marcus, caeser)}$ by Natural Deduction with labelling done for all derived sentences and unifiers used.

Q4)

4a)

Predicates: obs (b, c), lab (b, l), cont (b, l)

Unary Predicates: box (b), label (l), color (c)

Knowledge base:

These constraints ensure that each label (viz. W, Y, B) are only given to single distinct box:

1) ∀ l, b1 label (l) ^ box (b1) ^ lab (b1, l) -> ∀ b2 box (b2) ^ ¬Equal (b1, b2) ^ ¬lab (b2, l)

These constraints ensure that each correct label (viz. W, Y, B) are only given to single distinct box:

2) ∀ l, b1 label (l) ^ box (b1) ^ cont (b1, l) -> ∀ b2 box (b2) ^ ¬Equal (b1, b2) ^ ¬cont (b2, l)

These constraints ensure that boxes are incorrectly labelled:

3) ∀ l, b label (l) ^ box (b) ^ lab (b, l) <-> ¬cont (b, l)

These constraints ensure that correct colored balls are picked from correct labelled boxes:

4) ∀ b, c color (c) ^ box (b) ^ obs (b, c) -> cont (b, c) ˅ cont (b, B)

These (trivial) constraints ensure that each picked ball is of a color:

5) ∀ b, c1 color (c1) ^ box (b) ^ obs (b, c1) -> ∀ c2 color (c2) ^ ¬Equal (c1, c2) ^ ¬obs (b, c2)

These constraints ensure that each label (viz. W, Y, B) must be present amongst 3 boxes:

6) ∀ l label (l) -> ∃ b box (b) ^ lab (b, l)

These constraints ensure that each label (viz. W, Y, B) must be present amongst 3 boxes:

7) ∀ l label (l) -> ∃ b box (b) ^ cont (b, l)

These constraints ensure that each box must be labelled:

8) ∀ b box (b) -> ∃ l label (l) ^ lab (b, l)

9) ∀ b box (b) -> ∃ l label (l) ^ cont (b, l)

These constraints ensure that balls picked from boxes must have at least have 'W' or 'Y' color:

10) ∀ b color (b) -> ∃ b box (b) ^ obs (b, c)

These are the unary predicates:

11) box (1)

12) box (2)

13) box (3)

14) color (W)

15) color (Y)

16) label (W)

17) label (Y)

18) label (B)

19) ∀ b, l1,l2, l3 label (l1) ^ label (l2) ^ label (l3) ^ box (b) ^ ¬Equal (l1, l2 ^ ¬Equal (l1, l3) ^ ¬Equal (l2, l3) ^ ¬cont(b, l1) ¬cont(b, l2) -> cont(b, l3)

4b)

Initial facts:

20) obs (1, Y)

21) obs (2, W)

22) obs (3, Y)

23) obs (1, W)

24) obs (2, Y)

25) lab (3, B)

To prove: Knowledge Base entails **cont (2, W)**

Proof:

Applying Generalized Modus Ponens to 3, 13, 18 and 25 with

Unifier: θ = {b/3, c/B}

26) ¬cont (3, B)

Applying Generalized Modus Ponens to 4, 13, 15 and 22 with

Unifier: θ = {b/3, c/Y}

27) cont (3, Y) ˅ cont (3, B)

From 26 and 27

28) cont (3, Y)

Applying Generalized Modus Ponens to 4, 11, 15 and 20 with

Unifier: θ = {b/1, c/Y}

29) cont (1, Y) ˅ cont (1, B)

Applying Generalized Modus Ponens to 2, 13, 17 and 28 with

Unifier: θ = {b1/3, l/Y}

30) ∀ b2 box (b2) ^ ¬Equal (3, b2) ^ ¬cont (b2, Y)

Using 11 and 30

31) box (1) ^ ¬Equal (3, 1) ^ ¬cont (1, Y)

From 31

32) ¬cont (1, Y)

From 29 and 32

33) cont (1, B)

Applying Generalized Modus Ponens on 2, 13, 17 and 28 with

Unifier: θ = {b1/3, l/Y}

34) ∀ b2 box (b2) ^ ¬Equal (3, b2) ^ ¬cont (b2, Y)

From 34 and 12

35) ¬cont (2, Y)

Applying Generalized Modus Ponens on 2, 11, 18 and 33 with

Unifier: θ = {b1/1, l/B}

36) ∀ b2 box (b2) ^ ¬Equal (1, b2) ^ ¬cont (b2, B)

From 36 and 12

37) ¬cont (2, B)

Applying Generalized Modus Ponens to 19, 35, 37, 12, 16, 17, 18 with

Unifier: θ = {b/2, l1/Y, l2/B, l3/W}

38) cont (2, W)

Hence, proved that Knowledge Base entails **cont(2,W)** by **Natural Deduction** with labelling done for all derived sentences and unifiers used.

Q5)

Predicates:

Symbol (s), Row (r), Col (c), p (s, r, c), b (r, c),

canWin (s, r, c), forcedMove (s, r, c), TwoInARow (s, r), TwoInACol (s, c), TwoInADiag (s, d), move (s, r, c)

Knowledge Base:

Definition of **TwoInARow:**

1) ∀ s, r Symbol (s) ^ Row (r) ^ [∃ c1, c2 Col (c1) ^ Col (c2) ^ p (s, r, c1) ^ p (s, r, c2)]

-> TwoInARow (s, r)

Definition of **TwoInARow:**

2) ∀ s, c Symbol (s) ^ Col (c) ^ [∃ r1, r2 Row (r1) ^ Row (r2) ^ p (s, r1, c) ^ p (s, r2, c)]

-> TwoInACol (s, c)

These constraints identify the CanWin situations to take actions upon later:

3) ∀ s, r Symbol (s) ^ Row (r) ^ TwoInARow (s, r) ^ [∃ c b (r, c)] -> canWin (s, r, c)

4) ∀ s, c Symbol (s) ^ Col (r) ^ TwoInACol (s, r) ^ [∃ r b (r, c)] -> canWin (s, r, c)

Definition of **TwoInADiag:**

5) ∀ s Symbol (s) ^ [∃ r1, r2, Row (r1) ^ Row (r2) ^ p (s, r1, r1) ^ p (s, r2, r2)] -> TwoInADiag (s, left)

6) ∀ s Symbol (s) ^ [∃ r1, r2, Row (r1) ^ Row (r2) ^ p (s, r1, 4 - r1) ^ p (s, r2, 4 - r2)] -> TwoInADiag (s, right)

These constraints identify the CanWin situations to take actions upon later:

7) ∀ s Symbol (s) ^ TwoInADiag (s, left) ^ [∃ r Row (r) b (r, r)] -> canWin (s, r, r)

8) ∀ s Symbol (s) ^ TwoInADiag (s, right) ^ [∃ r Row (r) b (r,4 - r)] -> canWin (s, r, 4 - r)

These constraints ensure that when there is a CanWin of 'O', 'X' should place it at that place forcefully:

9) ∀ s1, r, c Symbol (s1) ^ Row (r) ^ Col (c) ^ canWin (s1, r, c) -> ∃ s2 Symbol (s2) ^ ¬Equal (s1, s2) ^ forcedMove (s2, r, c)

These constraints ensure that when there is a situation of winning, ‘X’ plays that move to win:

10) ∀ s, r, c Symbol (s) ^ Row (r) ^ Col (c) ^ canWin (s, r, c) -> move (s, r, c)

These constraints ensure that CanWin is given more priority than ForcedMove:

11) ∀ s, r, c Symbol (s) ^ Row (r) ^ Col (c) ^ [∀ a, b row (a) ^ col (b) ^ ¬canWin (s, a, b)] ^ forcedMove (s, r, c) -> move (s, r, c)