UNIVERSITY OF PATRAS
DEPT. OF COMPUTER ENGINEERING & INFORMATICS
ARTIFICIAL INTELLIGENCE

SOLUTIONS of **3d Assignment**

1. Translate the following natural language sentences into first-order logic (FOL) formulas.

a. "Pluto loves its master"
b. "Every dog has a master"
c. "John either hates George or is ambitious"
d. "Apples are a kind of food"
e. "Every carnivorous animal eats all animals smaller than itself"
f. "No man likes a woman who is vegeterian"

Answer

a. loves(Pluto, master_of(Pluto))
b. (∀x) dog(x) ⇒ (∃y) master(y, x)
c. hates(John. George) ∨ ambitious(John)
d. food(apples) **or**
(∀x) apple(x) ⇒ food(x)
(it depends on the way of use of the sentence)
e. (∀x) (∀y) ((carnivorous(x) ∧ animal(y) ∧ smallerthan(y,χ)) ⇒ eats(x, y)) **or**
(∀x) (carnivorous (x) ⇒ ((∀y) (animal (y) ∧ smallerthan(y,χ)) ⇒ eats(χ,y)))
(they are equivalent)
f. (∀x)(∀y)((man(x) ∧ woman(y) ∧ vegeterian(y)) ⇒ ¬likes(x,y)) **or**
¬((∃x)(∃y)(man(x) ∧ woman(y) ∧ vegeterian(y) ∧likes(x,y)))
(they are equivalent)

2. Convert the following FOL formulas in their Clause Normal Form.

a. (∀x) (∀y) (∀z) ((pet(x) ∧ master(x, y) ∧ lives(y, z)) ⇒ lives(x, z))
b. ((∀x) ((∃y) a(y) ⇒ b(x, y))) ∨ ((∀x) c(x))

Answer

a. (∀x) (∀y) (∀z) ((pet(x) ∧ master(x, y) ∧ lives(y, z)) ⇒ lives(x, z))

Present the conversión steps analyticaly.

1. Implication removal (based on F => G ≡ ¬F∨G):
(∀x) (∀y) (∀z) (¬(pet(x) ∧ master(x,y) ∧lives(y,z)) ∨ lives(x,z))

2. Moving negation at atom level (making required changes):
(∀x) (∀y) (∀z) ((¬pet(x) ∨ ¬master(x,y) ∨ ¬lives(y,z)) ∨ lives(x,z))

3. Renaming variables:
Not applicable

4. Transform to PNF (Prenex Normal Form)
Not applicable

5. Remove existential quantifiers
Not applicable

6. Remove universal quantifiers
$((\neg pet(x) \vee \neg master(x,y)$ $(\neg lives(y,z))$ $(lives(x,z))$

7. Transform to CNF (Conjunctive Normal Form)
Not applicable

8. Remove connectives and write down produced clauses
$\varphi \equiv (\neg pet(x), \neg master(x,y), \neg lives(y,z), lives(x,z))$

9. Rename variables
Not applicable (only one clause)

b. $((\forall x)\ ((\exists y)\ a(y) \Rightarrow b(x, y))) \vee ((\forall x)\ c(x))$

1. Implication removal (based on F => G ≡ ¬F∨G):
$((\forall x)\ ((\exists y)\ (\neg a(y) \vee b(x, y)))) \vee ((\forall x)\ c(x))$

2. Moving negation at atom level (making required changes):
Not applicable

3. Renaming variables:
$((\forall x)\ ((\exists y)\ (\neg a(y) \vee b(x, y)))) \vee ((\forall z)\ c(z))$

4. Transform to PNF (Prenex Normal Form)
$(\forall x)\ (\exists y)\ (\forall z)\ (\neg a(y) \vee b(x,y) \vee c(z))$

5. Remove existential quantifiers {y = f(x)}
$(\forall x)(\ \forall z)\ (\neg a(f(x)) \vee b(x,f(x)) \vee c(z))$

6. Remove universal quantifiers
$(\neg a(f(x)) \vee b(x,f(x))$ $(\ c(z))$

7. Transform to CNF (Conjunctive Normal Form)
Not applicable

8. Remove connectives and write down produced clauses
φ ≡ (¬a(f(x)) , b(x, f(x)) , c(z))

9. Rename variables
Not applicable (only one clause)

3. Check whether theliterals in the following couples can be unified. In case they can, find the most general unifier. Otherwise, explain why they cannot.

a. p(x, y) , p(a, z)
b. p(x, x) , p(a, b)
c. descendant(x, father-of(x)) , descendant(john, bill)
d. descendant (x, y) , descendant(bill, father-of(bill))
e. q(x, a, y) , q(z, z, b)
f. q(x) , ¬q(a)

Answer

a. p(x, y) , p(a, z)
They unify with mgu (most general unifier): σ = {a/x, y/z}

b. p(x, x) , p(a, b)
They are not unifiable because there is no valid mgu. If there was, it would be σ = {a/x, b/x}, which however is not valid, because the variable "x" has two different constants as bindings, which creates a conflict.

c. descendant(x, father-of(x)) , descendant(john, bill)
They are not unifiable, because while «x» unifiew with constant «john», the functional term «father-of(x)» cannot unify with a constant («bill»).

d. descendant (x, y) , descendant(bill, father-of(bill))
They unify with σ = {bill/x, father-of(bill)/y}

e. q(x, a, y) , q(z, z, b)
They unify with mgu σ = {z/x, a/z, b/y}, which results in σ = {a/x, a/z, b/y}

f. q(x) , ¬q(a)
They do not unify, they have different polarity.

4. The following FOL formulas are given:

(1) works-in (george, patras)
(2) works-in (paul, rio)
(3) master (george, pluto)
(4) master (paul, boby)
(5) (∀x) (∀y) (works-in (x, y) ⇒ lives-in (x, y))
(6) (∀x) (∀y) (∀z) ((master (x, y) ∧ lives-in (x, z)) ⇒ lives-in (y, z))

where x, y, z are variables.

(α) Using resolution refutation, prove that “Pluto lives in Patra”.
(β) Using resolution refutation, answer the question “Who lives in Rio?”.

Answer

(α) Using resolution refutation, prove that “Pluto lives in Patra”.

We first convert the given FOL formulas into clausal form (following the steps in Q2) and rename variables so that any two clauses have no common variables.

(1) (works-in(george, patras))
(2) (works-in(paul, rio))
(3) (master(george, pluto))
(4) (master(paul, boby))
(5) (¬works-in(x1, y1), lives-in(x1, y1))
(6) (¬master(x2, y2), ¬lives-in(x2, z), lives-in(y2, z))

Then, we tranfer “Pluto lives in Patra” into FOL formula: “lives-in(Pluto, Patra)”. According to resolution refutation, we negate the formula and get “¬lives-in(Pluto, Patra)”, and afterwards we convert it to a clause: (¬lives-in(Pluto, Patra)), which is the 7th clause of our knowledge base:

(7) (¬lives-in(Pluto, Patra))

Now, we apply the inference rule of Resolution between couples of resolvable clauses, trying to produce the ‘empty’ clause:

(8) **(¬lives-in(george, z), lives-in(pluto, z)**) is produced by resolving (3) and (6) with σ1={george/x2 , pluto/y2}

(9) **(lives-in(george, patras))** is produced by resolving (1) and (5) with σ2={george/x1 , patras/y1}

(10) **(lives-in(pluto, patras))** is produced by resolving (8) and (9) with σ3={patras/z}

(11) **()** Empty clause is produced by resolving (7) and (10).

Given that the empty claused is produced, the “lives-in(Pluto, Patra)” is true.

(β) Using resolution refutation, answer the question “Who lives in Rio?”.

First, we transfer “Who lives in Rio?” into a FOL formula. Questions denote existential quantification. So, the corresponding FOL formula is: “(∃x) lives-in(x,Rio)”. According to resolution refutation, we negate the formula and get “¬(∃x) lives-in(x,Rio)” and then we convert it into a clause: (∀x) ¬lives-in(x,Rio) → (¬lives-in(x,Rio)), which is now the 7th clause in our knowledge base:

(1) (works-in(george, patras))
(2) (works-in(paul, rio))
(3) (master(george, pluto))
(4) (master(paul, boby))
(5) (¬works-in(x1, y1), lives-in(x1, y1))
(6) (¬master(x2, y2), ¬lives-in(x2, z), lives-in(y2, z))
(7) (¬lives-in(x,Rio))

Now, we are looking not only for the empty clause, but simultaneously for a value for variable 'x', which will come from the resulted substitutions.

(8) **(lives-in(paul, rio))** is produced by resolving (2) and (5) with σ1={paul/x1 , rio/y1}

(9) **()** Empty clause is produced by resolving (7) and (8). with **σ2={paul/x}**

As we can see variable 'x' has taken the value **'paul'** in σ2, which is an answer to the initial question.

To find all possible answers (values of 'x'), we should continue resolution refutation using other options for resolution parent clauses:

(8) **(¬lives-in(paul, z), lives-in(boby, z))** is produced by resolving (4) and (6) with σ1={paul/x2 , boby/y2}

(9) **(¬works-in(paul, z)**, lives-in(boby, z)) is produced by resolving (5) and (8) with σ2={paul/x1 , z/y1}

(10) **(lives-in(boby, rio))** is produced by resolving (2) and (9) with σ3={rio/z}

(11) **()** Empty clause is produced by resolving (7) and (8)with **σ4={boby/x}**

So, a second answer (value), from σ4, is **'boby'**. Those are the only answers to the question.