ASSIGNMENT 3

1. FOL

a. Define your own lexicon for the text and use it to represent the sentences of FOL.

Here Alpha, Beta, Gamma are people who watch movies.
There are two categories of movies, they are Thriller and Scifi.
There are two types of pets, Cats and Frogs.
Constants: Alpha, Beta, Gamma, Thriller, Scifi, Cat, Frog.
Predicates: person(), movie(), pet(), likes(person, movie), has(person, pet).
Variables: X, Y, Z
Logic Symbols: ⇨ , ∧ , ∨, ⇔
Universal Quantification: ∀
Existential Quantification: ∃
FOL:
R1: person(Alpha)
R2: person(Beta)
R3: person(Gamma)
R4: movie(Thriller)
R5: movie(Scifi)
R6: pet(Cat)
R7: pet(Frog)
R8: ∀X [likes(person(X), movie(Thriller)) ∧ ¬ likes(person(X), movie(Scifi))] ⇨ has (person(X), pet(Cat))
R9: ∀Y has(person(Alpha), pet(Y)) ⇔ has(person(Gamma), pet(Y))
R10: ¬ likes(person(Gamma), movie(Scifi))
R11: has(person(Beta), pet(Frog))

b. Convert the sentences to CNF.

CNF:
R1: person(Alpha)
R2: person(Beta)
R3: person(Gamma)
R4: movie(Thriller)
R5: movie(Scifi)
R6: pet(Cat)
R7: pet(Frog)
R8: ∀X [likes(person(X), movie(Thriller)) ∧ ¬ likes(person(X), movie(Scifi))] ⇨ has (person(X), pet(Cat))
R8: --> ¬ (likes(person(X), movie(Thriller)) V likes(person(X), movie(Scifi))) V (has (person(X), pet(Cat)))
R9: ∀Y has(person(Alpha), pet(Y)) ⇔ has(person(Gamma), pet(Y))

R9: --> [has(person(Alpha), pet(Y)) ⇨ has(person(Gamma), pet(Y))] ∧ [has(person(Gamma), pet(Y)) ⇨ has(person(Alpha), pet(Y))]
R9: --> [¬ has(person(Alpha), pet(Y)) V has(person(Gamma), pet(Y))] ∧ [¬ has(person(Gamma), pet(Y)) V has(person(Alpha), pet(Y))]
R9a: [¬ has(person(Alpha), pet(Y)) V has(person(Gamma), pet(Y))]
R9b: [¬ has(person(Gamma), pet(Y)) V has(person(Alpha), pet(Y))]
R10: ¬ likes(person(Gamma), movie(Scifi))
R11: has(person(Beta), pet(Frog))

c. Using the results of subpart 2. Convert the following statements to FOL then CNF showing your work and use the resolution to prove it. Show your work using line numbering as shown in class.

Somebody likes to watch movies and has a cat.
FOL : ∃X ∃Z likes(person(X), movie(Z)) ∧ has (person(X), pet(Cat))
CNF : likes(person(X), movie(Z)) ∧ has(person(X), pet(Cat))
Resolution:
R1 : person(Alpha)
R2 : person(Beta)
R3 : person(Gamma)
R4 : movie(Thriller)
R5 : movie(Scifi)
R6 : pet(Cat)
R7 : pet(Frog)
R8 : ¬ likes(person(X), movie(Thriller)) V likes(person(X), movie(Scifi)) V has (person(X), pet(Cat))
R9a : ¬ has(person(Alpha), pet(Y)) V has(person(Gamma), pet(Y))
R9b : ¬ has(person(Gamma), pet(Y)) V has(person(Alpha), pet(Y))
R10a : ¬ likes(person(Gamma), movie(Scifi))
R10b : likes(person(Gamma), movie(Thriller))
R11 : has(person(Beta), pet(Frog))
R12 : ¬ [likes(person(X), movie(Z)) ∧ has(person(X), pet(Cat))]
R12 : ¬ likes(person(X), movie(Z)) V ¬ has(person(X), pet(Cat))

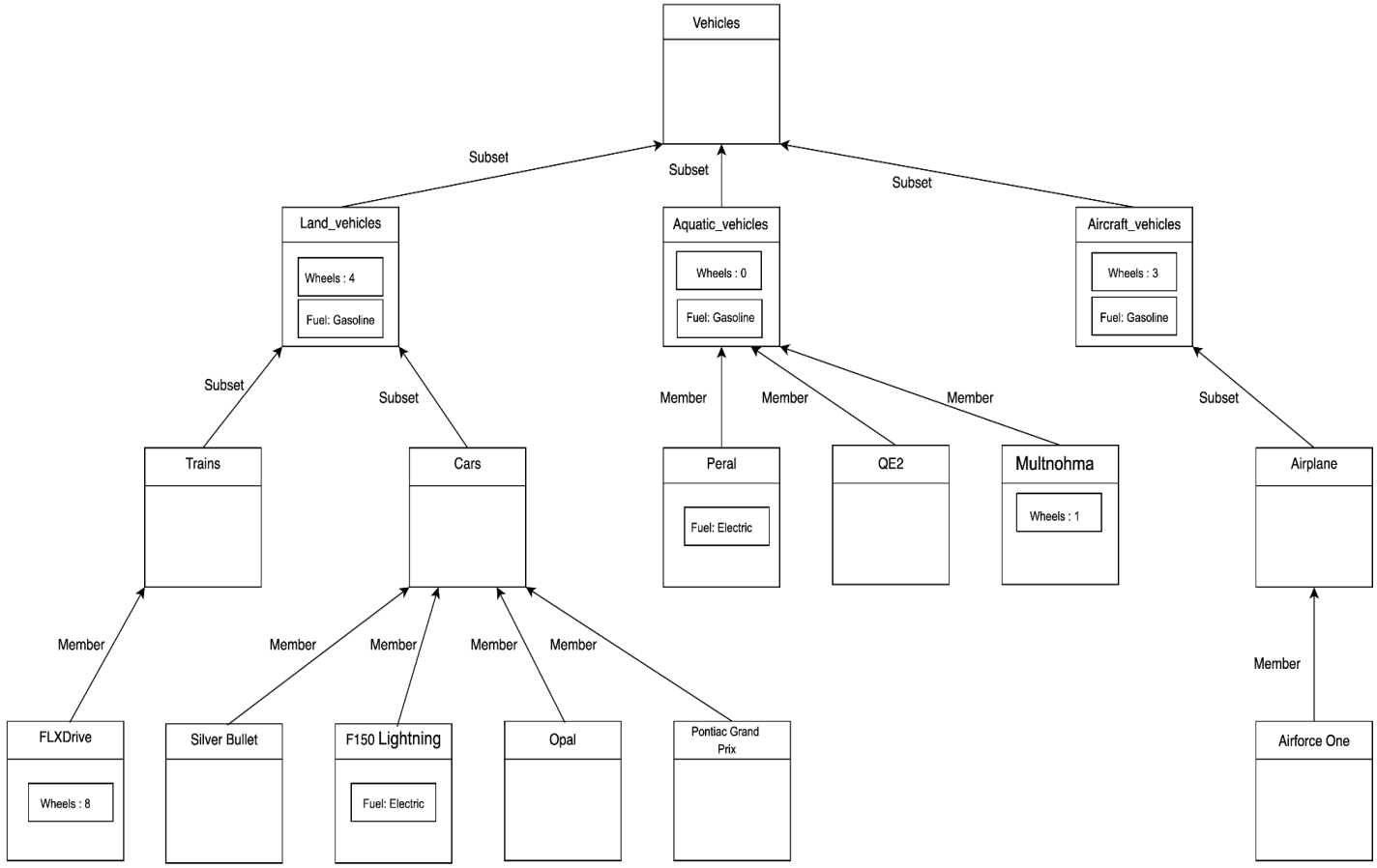
R10b + R8 (X : {Gamma}) :
likes(person(Gamma), movie(Scifi)) V has (person(Gamma), pet(Cat)) ------> R13

R12 + R13 (X : {Gamma} , Z : {Scifi}):
¬ likes(person(Gamma), movie(Scifi)) V ¬ has(person(Gamma), pet(Cat)) V likes(person(Gamma), movie(Scifi)) V has (person(Gamma), pet(Cat))
Φ

Therefore, Since the negated proof is false our resolution is True. Hence proved.

2. Knowledge Engineering

a. Draw this taxonomy as a frame representation graph, labeling the nodes, and edges as appropriate. Use both concept nodes and fields.



b. Represent the facts as FOPL facts in Prolog.

```
rel(opal, isa, car).
rel(pontiac_grand_prix, isa, car).
rel(flx_drive, isa, train).
rel(silver_bullet, isa, car).
rel(f150_lightning, isa, car).
rel(peral, isa, aquatic_vehicles).
rel(qe2, isa, aquatic_vehicles).
rel(multnohma, isa, aquatic_vehicles).
rel(airforce_one, isa, airplane).
rel(train, subset, land_vehicles).
rel(car, subset, land_vehicles).
rel(airplane, subset, aricraft_vehicles).
```

rel(land_vehicles, subset, vehicle).
rel(aquatic_vehicles, subset, vehicle).
rel(aricraft_vehicles, subset, vehicle).
property(wheels, flx_drive, 8).
property(fuel, f150_lightning, electric).
property(fuel, peral, electric).
property(wheels, multnohma, 1).
property(wheels, land_vehicles, 4).
property(fuel, land_vehicles, gasoline).
property(wheels, aquatic_vehicles, 0).
property(fuel, aquatic_vehicles, gasoline).
property(wheels, aircraft_vehicles, 3).
property(fuel, aircraft_vehicles, gasoline).
property(fuel, vehicle, gasoline).