Date
Page

Lab - 9 First Order Logic Forward Chaining

Considering Statement

As per law, it is a crime for an American to sell weapons to hostile nations. Country A, an enemy of America, has some missiles, and all missiles were sold to it by Robert, who is American

Prove Robert is a criminal.

Deduced Statements:

$American(x) \wedge Weapon(y) \wedge Sold(z, y, x) \wedge Hostile(z) \Rightarrow Criminal(x)$

$Enemy(A, America)$

~~Has (mis~~

~~$\forall x\ Missile(x) \rightarrow W$~~

$\exists x\ Missiles(x) \wedge Owns(A, x)$

$\forall x\ Missiles(x) \wedge Owns(A, x) \Rightarrow Sold(A, x, Robert)$

$American(Robert)$

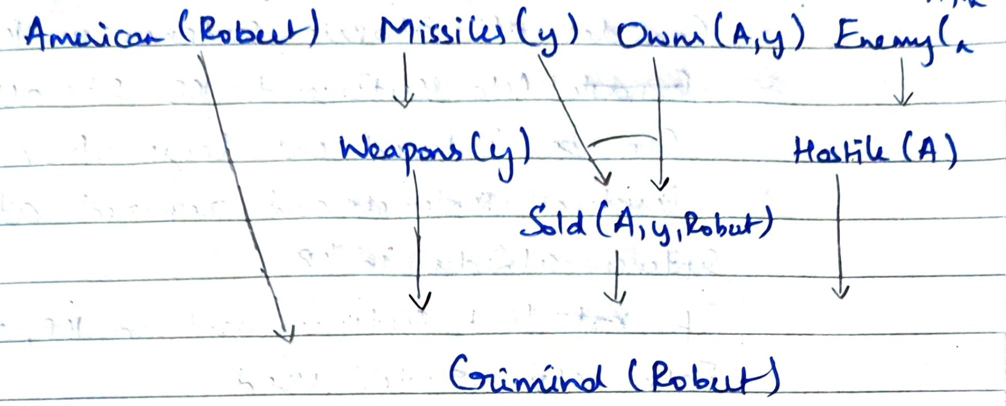
We can deduce that missiles are weapons

$\forall x\ Missile(x) \rightarrow Weapon(x)$

We can also deduce that an enemy of America is hostile

$\forall x\ Enemy(x, America) \Rightarrow Hostile(x)$

∴ ~~From~~ First level of atomic statements



<u>Algorithm:</u>

~~forward-chain (KB, α)~~

Assuming KB represents all current knowledge
α represents atomic query to be validated

```
while new != {}
    for rule in KB do
        (p1 ∧ p2 ... ∧ pn → q) ⇐ Standardize-variables (rule)
        for θ ~~in new~~ s.t
            (θ, p1 ∧ p2 ... ∧ pn) = (θ, p1' ∧ p2' ... pn')
            for p1', p2' ... in KB
        q' = subst(θ, q)
        if q' does not unify some sentence in
        KB or new
            new += q'
            φ = Unify(q', α)
            if not φ return False
            else return φ
    ~~new +=~~ KB += new
return False
```

Date
Page

General Algorithm:

- Consider all atomic sentences in KB
- ~~Generate~~ Add sentences to KB s.t all can be inferred one layer from current level of sentences and required atomic sentence already in KB.
- If ~~not~~ find sentence not in KB, return False else rules used.

→ **Resolution using FoL**

Consider the following sentences:

1) food (x) → likes(John, x)
2) food (Apple)
3) food (vegetables)
4) ~~∀~~ eats (y~~q~~, z) ∧ $\overline{\text{killed}}$ (~~z~~y) → food (z)
5) eats (Anil, Peanuts)
6) alive (Anil)
8) killed (x) ∨ alive(x)
7) ¬ killed (x) ∨ ¬alive(x)

To prove 4: likes (John, Peanuts)

Adding negation (¬ likes (John, Peanuts)) to KB and resolving

¬ likes (John, Peanuts)   ¬food(x) ∨ likes (John, x)

{x/peanuts}

¬food (Peanuts)   ¬eats(y, z) ∨ killed(y) ∨ food(z)

{z/Peanuts}

¬eats (y, Peanuts) ∨ killed (y) ∨ food (Peanuts)

eats (Anil, Peanuts)

{y/Anil}

killed (y) Anil ~~∨ food (Peanuts)~~   ¬alive(x) ∨ ¬killed(x)

{x/anil}

¬alive (Anil)   alive (Anil)

{}

Results in empty set, therefore contradicts

∴ likes (John, Peanuts) is true.