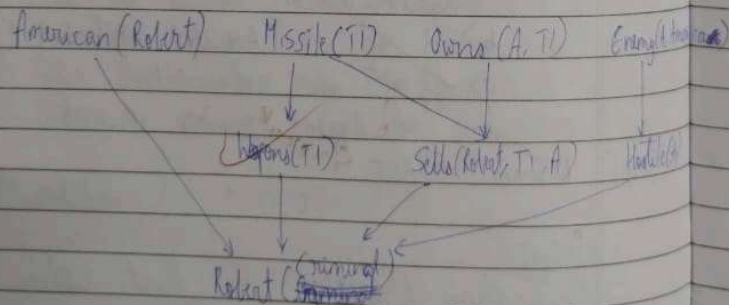


Forward Reasoning Algorithm:

8) Forward Reasoning Algorithm.

function FOL-FC-ASK(KB, α) returns a substitution or \perp
 repeat until new is empty
 new $\leftarrow \{\}$
 for each rule in KB do
 $(p \wedge \wedge p_i \Rightarrow q) \leftarrow \text{STANDARDIZE-VARIABLES}$
 for each θ such that $\text{SUBST}(\theta, p \wedge \wedge p_i)$
 $= \text{SUBST}(\theta, p_i \wedge \wedge p_i)$
 for some $p_i \dots p_n$ in KB
 $q' \leftarrow \text{SUBST}(\theta, q)$
 if q' does not unify with any sentence
 already in KB or new then
 add q' to new
 $\phi \leftarrow \text{UNIFY}(q', \alpha)$
 if ϕ is not fail then return ϕ
 add new to KB
 return False

Proof Tree



Output.

Inferred Facts
 - Country A
 - It is a
 - The enemy
 - Country A
 - All missiles
 - Colored
 - Missile
 - A has

Nearest Inferred
 The Gun

Q
29

Output:

Inferred Facts:

- Country A is an enemy of America
- It is a crime for Americans to sell a weapon to the enemy of America
- Crime has been committed
- All missiles were sold to A by Colonel
- Colonel is American
- Missile is a weapon
- A has some missiles

Newly Inferred Facts:

The crime has been committed.

29/11/2021

from 07/11/21

AKIA/EL/2021

1. A (A)

2. Intercom

turn 0

Enemy (A, America)

Hostile (A)

Code:

```
class KnowledgeBase:
    def __init__(self):
        self.facts = set()    # Set of known facts
        self.rules = []      # List of rules

    def add_fact(self, fact):
        self.facts.add(fact)

    def add_rule(self, rule):
        self.rules.append(rule)

    def infer(self):
        inferred = True
        while inferred:
            inferred = False
            for rule in self.rules:
                if rule.apply(self.facts):
                    inferred = True

# Define the Rule class
class Rule:
    def __init__(self, premises, conclusion):
        self.premises = premises    # List of conditions
        self.conclusion = conclusion    # Conclusion to add if premises are met

    def apply(self, facts):
        if all(premise in facts for premise in self.premises):
            if self.conclusion not in facts:
                facts.add(self.conclusion)
```

```

        print(f"Inferred: {self.conclusion}")
        return True
    return False

# Initialize the knowledge base
kb = KnowledgeBase()

# Facts in the problem
kb.add_fact("American(Robert)")
kb.add_fact("Missile(T1)")
kb.add_fact("Owns(A, T1)")
kb.add_fact("Enemy(A, America)")

# Rules based on the problem
# 1. Missile(x) implies Weapon(x)
kb.add_rule(Rule(["Missile(T1)"], "Weapon(T1)"))

# 2. Enemy(x, America) implies Hostile(x)
kb.add_rule(Rule(["Enemy(A, America)"], "Hostile(A)"))

# 3. Missile(x) and Owns(A, x) imply Sells(Robert, x, A)
kb.add_rule(Rule(["Missile(T1)", "Owns(A, T1)"], "Sells(Robert, T1, A)"))

# 4. American(p) and Weapon(q) and Sells(p, q, r) and Hostile(r) imply
Criminal(p)
kb.add_rule(Rule(["American(Robert)", "Weapon(T1)", "Sells(Robert, T1, A)",
"Hostile(A)"], "Criminal(Robert)"))

# Infer new facts based on the rules
kb.infer()

```

```
# Check if Robert is a criminal
if "Criminal(Robert)" in kb.facts:
    print("Conclusion: Robert is a criminal.")
else:
    print("Conclusion: Unable to prove Robert is a criminal.")
```

Output:

Inferred: Weapon(T1)

Inferred: Hostile(A)

Inferred: Sells(Robert, T1, A)

Inferred: Criminal(Robert)

Conclusion: Robert is a criminal.