```python
knowledge_base = {
    "facts": {
        "American(Robert)": True,
        "Enemy(A, America)": True,
        "Owns(A, T1)": True,
        "Missile(T1)": True,
    },
    "rules": [
        {"if": ["Missile(x)"], "then": ["Weapon(x)"]},
        {"if": ["Enemy(x, America)"], "then": ["Hostile(x)"]},
        {"if": ["Missile(x)", "Owns(A, x)"], "then": ["Sells(Robert, x, A)"]},
        {
            "if": ["American(p)", "Weapon(q)", "Sells(p, q, r)", "Hostile(r)"],
            "then": ["Criminal(p)"],
        },
    ],
}

def forward_chaining(kb):
    facts = kb["facts"].copy()
    rules = kb["rules"]
    inferred = set()

    while True:
        new_inferences = set()
        for rule in rules:
            if_conditions = rule["if"]
            then_conditions = rule["then"]
            substitutions = {}
            all_conditions_met = True
```

```python
                    all_conditions_met = False
                    break

            if all_conditions_met:
                for condition in then_conditions:
                    predicate, args = condition.split("(")
                    args = args[:-1].split(",")
                    new_fact = predicate + "(" + ",".join(
                        substitutions.get(arg, arg) for arg in args
                    ) + ")"
                    new_inferences.add(new_fact)

        if new_inferences - inferred:
            inferred.update(new_inferences)
            facts.update({fact: True for fact in new_inferences})
        else:
            break

    return inferred

# Run the forward chaining algorithm
result = forward_chaining(knowledge_base)

# Output the result
print("Vyom Gupta (1BM22CS333):")
if "Criminal(Robert)" in result:
    print("Proved: Robert is a criminal.")
else:
    print("Could not prove that Robert is a criminal.")
```

```
Vyom Gupta (1BM22CS333):
Proved: Robert is a criminal.
```

Date: 29/11/24 Lab-7

Forward Reasoning Algorithm

function FOL-FC-ASK(KB, α) returns a substitution or false

inputs: KB, the knowledge base, a set of first-order definite clauses
α, the query, an atomic sentence

local variables: new, the new sentences inferred on each iteration

repeat until new is empty
new ← { }
for each rule in KB do
(P1 ∧ ... ∧ Pn ⇒ q) ← STANDARDIZE-VARIABLES(rule)
for each θ such that SUBST(θ, P1 ∧ ... ∧ Pn) = SUBST(θ, P1' ∧ ... ∧ Pn')
for some P1', ..., Pn' in KB
q' ← SUBST(θ, q)
if q' does not unify with some sentence already in KB or new then
add q' to new
φ ← UNIFY(q', ...)
if φ is not fail

add new to KB
return false

Output:

Robert is a criminal for selling weapons to a hostile nation.

Updated Facts for Robert:
Name: Robert
is_american: True
is_hostile_nation: True
selling_weapons: True
is_criminal: True

Updated Facts for John:
Name: John
is_american: True
is_hostile_nation: False
selling_weapons: True
is_criminal: False

2/11/24.

Question: Write FOL:

a.) ⇒ Occupation(Emily, Surgeon) ∨ Occupation(Emi...

b.) ⇒ Occupation(Joe, Actor) ∧ ∃o(o ≠ Actor ∧ Occupation...

c.) ⇒ ∀p(Occupation(p, Surgeon) → Occupa...

d.) ⇒ ∀p(Occupation(p, Lawyer) → ∀ → Customer(Joe, ...

e.) ⇒ ∃p(Boss(p, Emily) ∧ Occupation(p, Lawyer))

f.) ∃p(Occupation(p, Lawyer) ∧ ∀q(Customer(...
→ Occupation(q, Doctor)))

g.) ∀p(Occupation(p, Surgeon) → ∃q(Customer(p, ...
∧ Occupation(q, Lawyer)))