WEEK-08

Create a knowledge base consisting of first order logic statements and prove the given query using forward reasoning.

# Define the knowledge base with facts and rules

knowledge\_base = [

    # Rule: Selling weapons to a hostile nation makes one a criminal

    {

        "type": "rule",

        "if": [

            {"type": "sells", "seller": "?X", "item": "?Z", "buyer": "?Y"},

            {"type": "hostile\_nation", "nation": "?Y"},

            {"type": "citizen", "person": "?X", "country": "america"}

        ],

        "then": {"type": "criminal", "person": "?X"}

    },

    # Facts

    {"type": "hostile\_nation", "nation": "CountryA"},

    {"type": "sells", "seller": "Robert", "item": "missiles", "buyer": "CountryA"},

    {"type": "citizen", "person": "Robert", "country": "america"}

]

# Forward chaining function

def forward\_reasoning(kb, query):

    inferred = []  # Track inferred facts

    while True:

        new\_inferences = []

        for rule in [r for r in kb if r["type"] == "rule"]:

            conditions = rule["if"]

            conclusion = rule["then"]

            substitutions = {}

            if match\_conditions(conditions, kb, substitutions):

                inferred\_fact = substitute(conclusion, substitutions)

                if inferred\_fact not in kb and inferred\_fact not in new\_inferences:

                    new\_inferences.append(inferred\_fact)

        if not new\_inferences:

            break

        kb.extend(new\_inferences)

        inferred.extend(new\_inferences)

    return query in kb

# Helper to match conditions

def match\_conditions(conditions, kb, substitutions):

    for condition in conditions:

        if not any(match\_fact(condition, fact, substitutions) for fact in kb):

            return False

    return True

# Helper to match a single fact

def match\_fact(condition, fact, substitutions):

    if condition["type"] != fact["type"]:

        return False

    for key, value in condition.items():

        if key == "type":

            continue

        if isinstance(value, str) and value.startswith("?"):  # Variable

            variable = value

            if variable in substitutions:

                if substitutions[variable] != fact[key]:

                    return False

            else:

                substitutions[variable] = fact[key]

        elif fact[key] != value:  # Constant

            return False

    return True

# Substitute variables with their values

def substitute(conclusion, substitutions):

    result = conclusion.copy()

    for key, value in conclusion.items():

        if isinstance(value, str) and value.startswith("?"):

            result[key] = substitutions[value]

    return result

# Query: Is Robert a criminal?

query = {"type": "criminal", "person": "Robert"}

# Run the reasoning algorithm

if forward\_reasoning(knowledge\_base, query):

    print("Robert is a criminal.")

else:

    print("Could not prove that Robert is a criminal.")

OUTPUT:

