Week 8: Forward Reasoning
Consider the following problem:

As per the 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 the missiles were sold to it by Robert, who is an American citizen."

Prove that "Robert is criminal."

Code:

```
# 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:

```
Robert is a criminal.
```