```
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

            for condition in if_conditions:
                predicate, args = condition.split("(")
                args = args[:-1].split(",")

                matched = False
                for fact in facts:
                    fact_predicate, fact_args = fact.split("(")
                    fact_args = fact_args[:-1].split(",")

                    if predicate == fact_predicate and len(args) == len(fact_args):
                        temp_subs = {}
                        for var, val in zip(args, fact_args):
                            if var.islower():
                                if var in temp_subs and temp_subs[var] != val:
                                    break
                                temp_subs[var] = val
                            elif var != val:
                                break
                        else:
                            matched = True
                            substitutions.update(temp_subs)
                            break

                if not matched:
                    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

result = forward_chaining(knowledge_base)
print('SWAPNIL SAHIL(1BM22CS300):')

if "Criminal(Robert)" in result:
    print("Proved: Robert is a criminal.")
else:
    print("Could not prove that Robert is a criminal.")
```

```
SWAPNIL SAHIL(1BM22CS300):
Proved: Robert is a criminal.
```