Lab-7

Date 29/11/2024
Page

Forward reasoning Algorithm.

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

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') 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', α)
if φ is not fail then
return φ
add new to KB
return false

Date __/__/__
Page

Q 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 where sold to it by Robert, who is an American citizen.

Prove Robert is criminal.

Knowledge-based

```
"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)"],
  },
],
}
```

output:

after running forward chaining algo.

Proved: Robert is criminal.

Date __/__/__
Page _____

Q Write the FOL for following.

→ Emily is either a surgeon or a lawyer
Occupation(Emily, surgeon) ∨
Occupation(Emily, Lawyer)

→ Joe is an actor, but he also holds another job
Occupation(Joe, Actor) ∧ ∃ X : Occupation(Joe, X)
∧ X ≠ Actor

→ All surgeons are doctors
∀ p : Occupation(p, surgeon) ⇒ Occupation
(p, Doctor)

→ Joe does not have a lawyer
∀ p : ¬Occupation(p, Lawyer) ∧
Customer(Joe, p)

→ Emily has a boss who is a lawyer
∃ p : Boss(p, Emily) ∧ Occupation(p, Lawyer)

→ There exists a lawyer all of whose customers are doctors
∃p : Occupation(p, Lawyer) ∧ ∀ c :
Customer(c, p) ⇒ Occupation(c, Doctor)

→ Every surgeon has a lawyer
∀ p : Occupation(p, surgeon) ⇒ ∃ l :
Occupation(l, Lawyer) ∧ Customer
(p, l)

fmem3ehyc

December 21, 2024

[1]:
```python
# Facts in the Knowledge Base
print("Name:Sudarshan Komar","USN:1BM22CS291",sep="\n")
knowledge_base = {
    "facts": {
        "American(Robert)": True,
        "Enemy(A, America)": True,
        "Owns(A, T1)": True,
        "Missile(T1)": True,
    },
    "rules": [
        # Rule 1: Missiles are weapons
        {"if": ["Missile(x)"], "then": ["Weapon(x)"]},

        # Rule 2: Enemy of America is hostile
        {"if": ["Enemy(x, America)"], "then": ["Hostile(x)"]},

        # Rule 3: If a missile is owned by A, Robert sold it
        {"if": ["Missile(x)", "Owns(A, x)"], "then": ["Sells(Robert, x, A)"]},

        # Rule 4: Law: Selling weapons to hostile nations is a crime
        {
            "if": ["American(p)", "Weapon(q)", "Sells(p, q, r)", "Hostile(r)"],
            "then": ["Criminal(p)"],
        },
    ],
}


def forward_chaining(kb):
    # Extract facts and rules
    facts = kb["facts"].copy()
    rules = kb["rules"]

    # Keep track of inferred facts
    inferred = set()

    while True:
```

```
        new_inferences = set()

        for rule in rules:
            # Check if all conditions of the rule are satisfied
            if_conditions = rule["if"]
            then_conditions = rule["then"]

            # Create a substitution for the rule variables
            substitutions = {}
            all_conditions_met = True

            for condition in if_conditions:
                # Extract the predicate and arguments
                predicate, args = condition.split("(")
                args = args[:-1].split(",")  # Remove closing parenthesis and␣
↪split

                # Check for matching facts
                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):
                        # Match variables or constants
                        temp_subs = {}
                        for var, val in zip(args, fact_args):
                            if var.islower():  # It's a variable
                                if var in temp_subs and temp_subs[var] != val:
                                    break  # Conflict in substitution
                                temp_subs[var] = val
                            elif var != val:  # Constants must match
                                break
                        else:
                            # Valid match
                            matched = True
                            substitutions.update(temp_subs)
                            break

                if not matched:
                    all_conditions_met = False
                    break

            # If all conditions are met, infer the "then" facts
            if all_conditions_met:
                for condition in then_conditions:
```

```
                    # Apply substitutions to the "then" condition
                    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)

        # Add new inferences to the knowledge base
        if new_inferences - inferred:
            inferred.update(new_inferences)
            facts.update({fact: True for fact in new_inferences})
        else:
            break

    return inferred


# Run forward chaining
result = forward_chaining(knowledge_base)

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

```
Name:Sudarshan Komar
USN:1BM22CS291
Proved: Robert is a criminal.
```