Lab 6:

Propositional Logic:-

P → Q (if P is true, then Q is true)
(we know P is true)

Knowledge based:

1) Alice is mother of bob
2) Bob is father of Charlie
3) A father is a parent
4) A mother is a parent
5) All parents have children
6) If someone is a parent, their children are sibling
7) Alice is married to David

Hypothesis
"Charlie is a sibling of Bob

Propositional Logic

1) M(A, B) : Alice is mother of Bob
2) F(B, C) : Bob is father of Charlie
3) Parent(x) : x is a parent
4) Child(y, x) : y is child of x
5) Siblings(x, y) : x & y are siblings
6) Married(A, D) : Alice is married to David
7) Parent(x) ∧ a children(y) ∧ who are siblings (x & y)

Date __/__/__
Page

Logical reasoning

1) From statement 1 & 4
   M(A, B) & (y, x) -> Alice is parent
2) From 2 & 4:
   F(B, C) & (y, x) -> Bob is parent
3) from 1 & 3 & 7:
   M(A, B) & F(B, C) & (x ≠ c & (x, y)) -> Bob & Charlie are not siblings

Code:

```
class Knowledge:
    def __init__(self):
        self.rules = []
        self.facts = set()
    def add_fact(self, fact):
        self.facts.add(fact)
    def add_rule(self, premises, conclusion):
        self.rules.append((premises, conclusion))
    def infer(self):
        new_inferences = True
        while new_inferences:
            new_inferences = False
            for premises, conclusion in self.rules:
                if all(fact in self.facts for fact in
                    premises):
                    if conclusion not in self.facts:
                        self.facts.add(conclusion)
                        new_inferences = True
    def entails(self, hypothesis):
        return hypothesis in self.facts
```

Kb = Knowledge Base()
Kb.add_fact("Alice is mother of bob")
Kb.add_fact("")
Kb.add_fact("")
Kb.add_fact("")
Kb.add_fact("")
Kb.add_fact("")
Kb.add_rule([("bob is father of charlie", "A father is a parent"), "bob is a parent")
Kb.add_rule([("Alice is mother of Bob", "A mother is parent"], "Alice is a parent")
Kb.add_rule([("Bob is a parent", "A parent have children")
Kb.infer()
hypothesis = "Charlie & Bob are Siblings"
if Kb.entails(hypothesis):
    printf("{hypothesis}" is entailed by Knowledge base)
else:
printf("{hypothesis}" is not entailed by Kb")

Lab 7

Date / /
Page

Unification in First order Logic

Key conditions

i) Same predicate symbol : the predicate symbols in the expression must match
ii) Same no of arguments : the expression must have an equal no of arguments
iii) Variable conflict resolution : variables cannot take multiple conflicting values
iv) No conflicting function symbols : Different function symbols cannot unify

Examples :-

i) Expression A : Knows (f(x, y) , g(x))
Expression B : Knows (f(Alice, Bob) , g(z))

Sol :-

i) By comparing the predicate, we arrive that Both are Knows
ii) Consider f(x, y) => f (Alice, Bob)
x = Alice , y = Bob
g(x) -> g(z)
z = Alice (Since x = Alice)

iii) x = Alice
y = Bob
z = Alice

iv) Unified Expression
Knows (f (Alice, Bob) , g (Alice))

Date __/__/__
Page _____

Code:-

```
Knowledge base = [
{"type": "rule", "rule": "∀x ∃y (Doctor(x) ∧ sickly(y) → Treats(x,y))"}
{"type": "fact", "fact": "Doctor(Jephter)"}
{"type": "fact", "fact": "∃x (Doctor(x) → ∃h (Hospital(h) ∧ works At(x, h)))"}
{"type": "fact", "fact": "Hospital(General Hospital)"},
{"type": "fact", "fact": "WorksAt(John, General Hospital)"}]

query = {"Predicate": "Treats", "arguments": ["?", "Mary"]}
def unify(kb, query):
    predicate = query["Predicate"]
    target_arg = query["arguments"][1]
    result = None

for item in kb:
if item["type"] == "rule" and predicate
in item["rule"]: rule = item["rule"]
if Doctor(x)" in rule and "sickly" in rule:
    doctor = None
    sick_person = None

for fact in kb:
if fact["type"] == "fact" and "Doctor" in
fact["fact"]:
doctor = fact["fact"].split("(")[1][:-1]
if fact["type"] == "fact" and "sick" in
fact("fact"), sick_person = fact
["fact"].split("(")[1][:-1]
```

if result:
return f"The query '{query['Predicate']}({result}, {target_args})' is unified: {result} treats {target_args}"
else:
return "The query '{query}' query['Predicate']}({query['args'][0]}, {target_args})' could not required with knowledge base."
result = unify(knowledge_base, query)
print(result)

Output :-
The query can't access? project 'X':
unified: John Treats Mary

Implemented
3/12/24