19-11-24

Lab - 08

⇒ First order logic ⇒ Unification

⇒ Implement unification in first order logic

→ Finding a common substitution for variables in different terms to make them match.

⇒ Statements:

1) Every cat loves to chase mice
2) Tom is a cat
3) Jerry is a mouse
4) If someone loves to chase Jerry, they will try to catch Jerry
5) Jerry can run fast
6) If someone tries to catch a mouse that runs fast, they will fail to catch it

⇒ 1) $\forall x\,(Cat(x) \rightarrow \exists y(Mouse(y) \wedge LovesToChase(x, y))$
2) $Cat(Tom)$
3) $Mouse(Jerry)$
4) $\forall x(LovesToChase(x, Jerry) \rightarrow TriesToCatch(x, Jerry))$
5) $RunsFast(Jerry)$
6) $\forall x \forall y((Mouse(y) \wedge RunsFast(y) \wedge TriesToCatch(x, y)) \rightarrow FailsToCatch$

⇒ 7) Does Tom fail to catch Jerry?

FOL: $FailsToCatch(Tom, Jerry)$

⇒ (i) From statement 2 and 1, we can infer
→ Since Tom is a cat, there exists some y(mouse) such that
$Mouse(y) \wedge LovesToChase(Tom, y)$

(ii) Using statement 3 ⇒ $y = Jerry$

Conclusion : $LovesToChase(Tom, Jerry)$

(iii) From statement 4 and the fact that LovesToChase(Tom, Jerry)
Conclusion: TriesToCatch(Tom, Jerry)

(iv) From statement 5 and 3, Jerry satisfies the conditions of statement
Since, Mouse(Jerry), RunsFast(Jerry) and TriesToCatch(Tom, Jerry),
we conclude ⇒ FailsToCatch(Tom, Jerry)

→ ∴ The unification is successful, and FailsToCatch(Tom, Jerry)
is true based on the knowledge base.

Lab-08

```
knowledge_base = [
  { "type": "rule", "rule": "∀x ∀y (Programmer(x) ∧ Project(y) ⟶ WritesCodeFor(x, y)" },
  { "type": "fact", "fact": "Programmer(Alice)" },
  { "type": "fact", "fact": "Project(Project x)" }
  { "type": "rule", "rule": "∀x ∀y (WritesCodeFor(x,y) = AssignedTo(x, y))" }
  { "type": "rule", "rule": "∀x ∀y (AssignedTo(x, y) ⟶ CanAccess(x,y)" }
  { "type": "fact", "fact": "AssignedTo(Bob, Projectx)" }
]

query = { "predicate": "canAccess", "arguments": [ "?", "Projectx" ] }

def unify(kb, query):
    predicate = query ["Predicate"]
    target_project = query("arguments"][1]
    result = []

    for item in kb:
        if item["type"] == "rule" and predicate
        in item("rule"]:
        rule = item["rule"]

    if "AssignedTo(x,y)" is rule and "canAccess(x,y)" is rule:

    for fact in kb:
        if fact["type"] == "fact" and "AssignedTo"
            in fact["fact"]
            fact_parts = fact("fact"). split("(")[1]. strip(")"). spli
            person. project = fact_parts
            if project == target_project:
                result. append(person)
```

```
if result:
    return f"The query '{query['predicate']}
      ({query['arguments'][0]}, {target_project})'
      is unified: {', '.join(result)} can access {target_project}"
else:
    return f"The query '{query['predicate']}
      ({query['arguments'][0]}, {target_project})'
      could not be unified with the knowledge base."

result = unify(knowledge_base, query)
print(result)
```

Output:-

The query '(Can Access ?, Project X)' is unified - Bob can access Project X

RM
26/4/24