DATE: PAGE:

Sumanth.K.V.
1BM18CS112
5 C
AI lab test-2
23/12/2020

```
import re

def isVariable(x):
    return len(x)==1 and x.islower() and
    x.isalpha()

def getAttributes(string):
    expr = '\([^)]+\)'
    matches = re.findall(expr, string)
    return matches

def getPredicates(string):
    expr = '([a-z~]+)\([^&|]+\)'
    return re.findall(expr, string)

class Fact:
    def __init__(self, expression):
        self.expression = expression
        predicate, params = self.splitExpress-
        -ion(expression)
        self.predicate = predicate
        self.params = params
        self.result = any[self.getConstants())
```

Sumanth.K.V

```
def splitExpression(self, expression):
    predicate = getPredicate(expression)
    params = getAttributes(expression)
    .strip(')').split(',')

    return [predicate, params]

def getResult(self):
    return self.result

def getConstants(self):
    return [None if isVariable(c)
        else c for c in self.params]

def getVariables(self):
    return [v if isVariable(v) else
        None for v in self.params]

def substitute(self, constants):
    c = constants.copy()
    f = f"{self.predicate}({','.
        join([constants.pop(0) if

    isVariable(p) else p for p in
    self.params])})"

    return Fact(f)

class Implication:
    def __init__(self, expression):
        self.expression = expression
        l = expression.split("=>")
        self.lhs = [Fact(f) for f in
```

```
l[0].split('&')]
self.rhs = Fact(l[1])

def evaluate(self, facts):
    constants = {}
    new_lhs = []
    for fact in facts:
        for val in self.lhs:
            if val.predicate == fact.predicate:
                for i, v in enumerate(val.
                        getVariables()):
                    if v:
                        constants[v] = fact.
                            get(constants()[i]

                new_lhs.append(fact)

    predicate, attributes = getPredicate(self.
        rhs.expression)[0], str(getAttributes(
        self.rhs.expression)[0])

    for key in constants:
        if constants[key]:
            attributes = attributes.replace
            (key, constants[key])

    expr = f'{predicate}{attributes}'

    return Fact(expr) if len(new_lhs) and
    all([f.getResult() for f in new_lhs])
    else None
```

DATE: PAGE:

```
class KB:
    def __init__(self):
        self.facts = set()
        self.implications = set()

    def tell(self, e):
        if "=>" in e:
            self.implications.add(Implication(e))
        else:
            self.facts.add(Fact(e))

        for i in self.implications:
            res = i.evaluate(self.facts)
            if res:
                self.facts.add(res)

    def query(self, e):
        facts = set([f.expression for f in
                     self.facts])

        i = 1

        for f in facts:

            if Fact(f).predicate == Fact(e).predica
                print(f'\t{i}. {f}')

                i += 1
```

```
def display(self):
    for i,f in enumerate(set([f.
    expression for f in self.facts])):
        print(f'\t{i+1}.{f}')
```

(5) ~~Sumanth.kb~~

```
kb = KB()
kb.tell('dog(x) => animal(x)')
kb.tell('animal(y) => die(y)')
kb.tell('dog(spoogy)')
kb.query('die(dog)')
```

Output:

1. die(spoogy)

(5) Sumanth.kb