01/01/2021

AI Lab Test 2

Sahana L
1BM18CS089

Write up

(5) Forward Reasoning Program ("Someone who are intelligent cannot read")

```
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.splitExpression(expression)
        self.predicate = predicate
        self.params = params
        self.result = any(self.getConstants())
```

01/01/2021

AI Lab Test 2

Sahana L
1BM18CS 089

```
def splitExpression (self, expression):
    predicate = getPredicates (expression) [0]
    params = getAttributes (expression) [0].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~~ Fact (f)
```

10/1/2021

A I Lab Test 2

Sahana L
1BM18CS089

```
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.getAttributes~~
                  for i, v in enumerate(val.getVariables()):
                      if v:
                          constants[v] = fact.getConstants()[i]
                  new_lhs.append(fact)
      predicate, attributes = getPredicates(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~~
```

01/01/2021

AI Lab Test 2

Sahana L
1BM18CS089

```
all([f.getResult() for f in new_lhs]) else None

~~cto~~

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.
      print(f'Querying {e}:')
      for f in facts:
         if Fact(f).predicate == Fact(e).predicate:
            print(f'\t{i}.{f}')
            i+=1
```

[illegible]

01/01/2021

AI Lab Test 2

Sahana L
1BM18CS089

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


def main():
    kb = KB()
    print("Enter KB: (enter e to exit)")
    while True:
        t= input()
        if (t == 'e'):
            break
        kb.tell (t)
    print ("Enter Query:")
    q= input()
    kb.query(q)
    kb.display ()


main ()
```

[illegible]

01/01/2021

AI Lab Test 2

Sahana L
1BM18CS089

$D(x)$: x is Dolphin

$L(x)$: x is literate

$I(x)$: x is Intelligent

$R(x)$: x can read.

$\forall x \; R(x) \Rightarrow L(x)$

$\forall x \; D(x) \Rightarrow \neg L(x)$.

$\exists x \; D(x) \cancel{\Rightarrow} \land I(x)$

$\exists x \; I(x) \land \neg R(x)$

[illegible]