29/12/2020

AI Lab test - 2

AYAN DAS
1BM18CS022

Writeup

```
Combination = [(True, True, True), (True, True, False), (True, False, True), (True, False, False),
               (False, True, True), (False, True, False), (False, False, True), (False, False, False)].
variable = {'p':0, 'q':1, 'r':2}

    kb = ''
    q = ''
    priority = {'~':3, 'v':1, '^':2}

    def input rules ():
            global kb, q
            kb = (input ("Enter rule: "))
            q = (input ("Enter query: "))

    def eval (i, val1, val2):
            if i == '^':
                    return val2 and val1
            return val2 or val1

    def evalPostfix (exp, comb):
            stack = []
            for i in exp:
                    if isOperand (i):
                            stack.append (comb [variable[i]])
                    elif i == '~':
                            val1 = stack.pop ()
                            val2 = stack.pop ()
                            stack.append (eval (i, val1, val2))
            return stack.pop()

    def toPostfix (infix):
            stack = []
            postfix = ''
            for c in infix:
                    if isOperand (c):
                            postfix += c
```

```
        else:
            if isLeftParanthesis(c):
                stack.append(c)
            elif isRightParanthesis(c):
                operator = stack.pop()
                while not isLeftParanthesis(operator):
                    postfix += operator
                    operator = stack.pop()
            else:
                while (not isEmpty(stack)) and hasLessOrEqualPriority
                                                (c, peek(stack)):
                    postfix += stack.pop()
                stack.append(c)
    while (not isEmpty(stack)):
        postfix += stack.pop()
    return postfix

def entailment():
    global kb, q
    print(' '*10 + "Truth Table reference"+ ' ' * 10)
    print("kb", 'alpha')
    print('*' * 10)
    for comb in combination:
        s = evaluatePostfix(toPostfix(kb), comb)
        f = evaluatePostfix(toPostfix(q), comb)
        print(s, f)
        print('-' * 10)
        if s and not f:
            return ~~False~~ False
    return True

def isOperand(c):
    return c.isalpha() and c != 'v'
```

```
def isleftParan(c):
        return c=='('
def isrightParan(c):
        return c == ')'

def isEmpty(stack):
        return len(stack)==0
def peek(stack):
        return stack[-1]

def hasLessorEqualPriority(c1, c2)
        try: return priority[c1] <= priority[c2]
        except KeyError: return False

input rules()
ans = entailment()
if ans:
        print("Knowledge base entails query")
else:
        print("Knowledge base does not entail query")
```