classmate
Date
Page

29/12/22

Lab - 6 - Knowledge Base

Aim: Create a knowledge base using propositional logic and show that the given query entails the knowledge base or not.

Algorithm -

function TT-ENTAILS? (KB, a) returns true or false:
inputs: KB, the knowledge base, a sentence in propositional logic a, the query, a sentence in propositional logic.

symbols a list of the proposition symbol in KB and a.
return TT-CHECK-ALL(KB, a, symbols, {})

function TT-CHECK-ALL (KB, a, symbols, model)
returns true or false

if Empty?(symbols) then
if PL-TRUE?(KB, model) then return PL-TRUE?(a, model)
else return true // if when KB is false.
// always return true.
else do
P · FIRST(symbols) then.
rest · 4 - REST(symbols).
return (TT-CHECK-ALL(KB, a, rest, model
∨ {P = TRUE})
and.
TT-CHECK-ALL(KB, a, rest, model ∪
{P = false}))

Code:

```
combinations = [(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}
priority = {'v': 1, '^': 2, '~': 3}.

kb = ''
q = ''

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

def isLeftParenthesis(c):
    return c == "("

def isRightParenthesis(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

```
def toPostfix(infix):
    stack = []
    postfix = ''
    for c in infix:
        if isOperand(c):
            postfix += c
        else:
            if isLeftParenthesis(c):
                stack.append(c)
            elif isRightParenthesis(c):
                operator = stack.pop()
                while not isLeftParenthesis(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 eval(i, val1, val2):
    if i == '^': return val2 and val1.
    return val2 or val1

def evaluatePostfix(exp, canb):
    stack = []
    for i in exp:
        if isOperand(i):
```

```
            stack.append(comb[variable[i]])
        elif i == '~':
            val1 = stack.pop()
            val2 = stack.pop()
            stack.append(not val1)
        else:
            val1 = stack.pop()
            val2 = stack.pop()
            stack.append(-eval(i, val2, val1))
    return stack.pop()

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

def entailment():
    global kb, q
    print('*'*10 + "Truth Table Reference" + '*'*10)
    print("kb", "alpha")
    print('*'*10)
    for comb in combinations:
        s = evaluate PostFix(toPostFix(kb), comb)
        f = evaluate PostFix(toPostFix(q), comb)
        print(s, f)
        print('-'*10)
        if s and not f:
            return False
    return True

input(rules)
ans = entailment()
if ans: print("The knowledge Base entails query")
```

else: print("The knowledge base doesnot entail query")

Output:

Enter Rule: (~q v p v r) ^ (~q ^ p) ^ q
Enter Query: r
* * * * * * Truth Table Reference * * * * * * *
kb alpha
* * * * * * *
False. True.
- - - - - - - -
False False
- - - - - - -
False True
- - - - - - -
False False
- - - - - - -
False True
- - - - - - -
False False
- - - - - - -
False True
- - - - - - -
False False
- - - - - - -

28-12-22

The knowledge base Entails the Query.