

01/01/21

2<sup>nd</sup> Programme.Given (KB):  $A \Rightarrow B$  and  $C \Rightarrow D$ Query :  $A \vee C \Rightarrow B \vee D$ 

Program:-

import re

def negate(term):

return '~(' + term + ')' if term[0] != '~' else term[1:]

def reverse(clause):

if len(clause) &gt; 2:

t = split\_terms(clause)

return '{' + t[1] + '}' + '{' + t[0] + '}'

return ""

def split\_terms(rule):

exp = '(~\* [PQ R S])'

terms = re.findall(exp, rule)

return terms

def Contradiction(query, clause):

contradictions = ['{ ' + query + '}' + '{ negate(' + query + ')}' ,

'{ negate(' + query + ')}' + '{ ' + query + '}' ]

(1)

return clause in Contradictions or remove (clause) in Contradictions

```
def resolve(kb, query):  
    temp = kb.copy()  
    temp += [negate(query)]  
    steps = dict()  
    for rule in temp:  
        steps[rule] = 'Given.'  
    steps[negate(query)] = 'Negated Conclusion'  
    i = 0  
    while i < len(temp):  
        n = len(temp)  
        j = (i+1) % n  
        clauses = []  
        while j != i:  
            terms1 = split-terms(temp[i])  
            terms2 = split-terms(temp[j])  
            for c in terms1:  
                if negate(c) in terms2:  
                    temp = [t for t in temp if t != c]
```



$t2 = [t \text{ for } t \text{ in terms2 if } t! \text{ negate}(c)]$

$gen = t1 + t2$

if  $\text{len}(gen) == 2$ :

if  $(gen[0]) != \text{negate}(gen[1])$ :

$\text{clauses} += [r'(gen[0]) \vee gen[1]]$

else:

if  $\text{contradiction}(query, r'(gen[0]) \vee gen[1])$ :

$\text{temp.append}(r'\{gen[0]\} \vee \{gen[1]\})$

$\text{step}[i] = \{r''$

Removed from  $\text{step}[i]$   
and  $\{temp[j]\}$

return steps.

else if  $\text{len}(gen) == 1$ :

$\text{clauses} += [r'\{gen[0]\}]$

else:

if  $\text{contradiction}(query, r'\{terms[0]\} \vee$

(3)

$\{terms\ 2[i]\}$  }  
temp.append(f' {terms1[i]} {terms2[i]} ' )  
steps[i] = f"Resolved {temp1[i]} and  
{temp2[i]} to {tempE[i]} .

return steps .

for clause in clauses:

if clause not in clauses and clause !=  
reverse(~~clause~~ clause):

temp.append(clause)

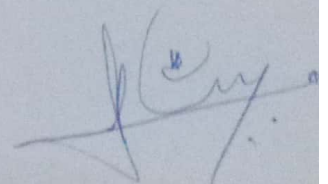
steps[clause] = f"Resolved from  
{temp1[i]} and {temp2[i]} .

$$j = (j+1) \% n.$$

~~i = i + 1~~  
 $i = i + 1$

return steps .

(6)





```
def resolution(Kb, query):
```

```
    Kb = Kb.split(" ")
```

```
    steps = resolve(Kb, query)
```

```
    print("In step t | clause | t | Denotation | t")
```

```
    print("-" * 30)
```

```
    i = 1
```

```
    for step in steps:
```

```
        print(f"{{i}}, t | step | t | {{steps[step]}}")
```

```
        i = i + 1
```

```
def main():
```

```
    print("Enter Kb:")
```

```
    Kb = input()
```

```
    print("Enter the query:")
```

```
    query = input()
```

```
    resolution(Kb, query)
```

```
if __name__ == "__main__":
```

```
    main()
```

5

Signature