AI - lab test - 2

Ankitha
1BM18CS016
V 'A'
29/12/20.

3. Given $P \Rightarrow Q$. and $R \Rightarrow S$,
Prove $P \vee R \Rightarrow Q \vee S$ using resolution

→

```
import re

def negation(term):
    return f'-{term}' if term[0] != '~' else term[1]

def reverse(clause):
    if len(clause) > 2:
        t = split(clause)
        return f'{t[1]}v{t[0]}'
    return

def split(rule):
    exp = '(~*[PQRS])'
    terms = re.findall(exp, rule)
    return terms

def contradiction(query, clause):
    contradictions = [f'{query}v negation(query)},
                      f'{negation(query)}v{query}']
    return clause in contraction or reverse(clause)
           is contradiction

def resolve(kb, query)
    temp = kb.query copy()
    tempt = [negation(query)]
    steps = dict()
    for rule in temp()
        steps[rule] = 'Given'
    steps {negation(query)} = 'Negated conclusion'
```

Antitha

```
i = 0
while i < len(temp):
    n = len(temp)
    j = (i+1) % n
    clauses = []
    while j != i:
        terms1 = split([temp[i]])
        terms2 = split(temp[j])
        for c in terms1:
            if negation(c) in terms2:
                t1 = [t for t in terms1 if t != c]
                t2 = [t for t in terms2 if t != negation(c)]
                gen = t1 + t2
                if (len(gen) == 2):
                    if gen[0] != negation(gen[1]):
                        clauses += [f'{gen[0]} V {gen[1]}']
                    else:
                        if contradiction(query, f'{gen[0]} V {gen[1]}'):
                            temp.append(f'{gen[0]} V {gen[1]}')
                            steps[' '] = f"Resolved {temp[i]} and {temp[j]}
                            to {temp[-1]}, which is null. In
                            a empty set is found which {negation
                            (query)} is assumed as true, hence
                            {query} entails the KB"
                            return steps
                elif:
                    if contradiction(query, f'{terms1[0]} V
                                                terms2[0]'):
                        temp.append(f'{terms1[0]} V {terms2[0]}')
                        steps[' '] = f"Resolved ({temp[i]} and
                        {temp[j]} to {temp[-1]} which is null
```

In A empty set is found {negation(query)} is assumed as true. Hence it entails.

return steps.

elif len(gen) == 1:
    clause += [f'{gen[0]}']

else:
    if contradiction(query, f'{terms1[0]} v {terms2[0]}'):
        temp.append(f'{terms1[0]} v {terms2[0]}')
        steps = [''] = f"Resolved {temp[i]} and temp[j]

for clause in clauses:
    if if clause not in temp and clause != reverse(clause) and reverse(clause) not in Temp:
        temp.append(clause)
        steps[clause] = f'Resolved from {temp[i]} and {temp[j]}'.

j = (j+1) % n

i += 1

return steps

```
def resolution(kb, query):
    kb = kb.split('.')
    steps = resolve(kb, query)
    print('\n No. \t|| clause \t|| Derivation \t')
    print('-' * 30)
    i = 1
    for no in steps:
        print(f'{i}. \t || {no} \n || {steps[no]} \t')
        i += 1
```

print ("Enter the knowledge base")

kb = '~P V Q ~R V S'

query = '~P ∧ ~R V Q V S'

resolution (kb, query)

$\alpha \Rightarrow \beta$

$= \neg\alpha \vee \beta$

$P \vee R \Rightarrow Q \vee S.$

$\neg(P \vee R) \vee (Q \vee S)$

$(\neg P \wedge \neg R \vee Q) \vee$

$\neg P \wedge \neg R \vee S)$