7 ⇒ (2) • Knowledged Based ~~entailment~~ resolution:

Algorithm:-

Initialization:

⇒ Create an instance of knowledge Base class with an empty list of clauses:-

self.clauses = []

⇒ Adding a clause:-

append a new clause to the list of clauses in the knowledge Base.

⇒ Resolving clause:-

combine 2 clauses by ~~Resolving literals~~ common literals
(Eliminating complementary literals)

Negate the query & adds it to the knowledge base

→ Repeatedly resolve the pairs of clauses in the knowledge base until a contradiction found or No new resolution are possible.

def resolve (self, clause_a, clause_b)

return [literal for literal in clause_a + clause_b if ('not' + literal) not in clause_a or literal not in clause_b]

Bafna Gold
Date: Page:

code (2)

```
def negate-literal (literal)
    if literal[0] == '~':
        return literal[1:]
    else:
        return '~' + literal

def resolve (c1, c2):
    resolved-clause = set(c1) | set(c2)

    for literal in c1:
        if negate-literal (literal) in c2:
            resolved-clause.remove (literal)
            resolved-clause.remove (negate-
                literal (literal))
    return tuple (resolved-clause)

def resolution (knowledge-base):

    while True:
        new clauses = set()
        for i, c1 in enumerate (knowledge-base)
            for j, c2 in enumerate (knowledge
                base):
                if i != j:
                    new-clause = resolve (c1, c2)
                    if len (new-clause) > 0 & new-
                        clause not in
                        knowledge-base:
                        new-clause.add (new-clause)

        if not new-clause
            break
```

```
        knowledge_base != new_clause
    return knowledge_base

if __name__ == "__main__":
    kb = set([('p', 'q'), ('~p', 'r'), ('~q', '~r')])
    result = resolution(kb)
    print("Original KB", kb)
    print("Resolved KB", result)
```

Output:-

Enter statement : a negation.

The statement is not entailed by knowledge-based

```
rules = 'PvQ ~PvR ~QvR' #P=vQ, P=>Q : ~PvQ, Q=>R, ~QvR
goal = 'R'
main(rules, goal)
```

```
Step    |Clause |Derivation
-----------------------------
 1.     | PvQ   | Given.
 2.     | ~PvR  | Given.
 3.     | ~QvR  | Given.
 4.     | ~R    | Negated conclusion.
 5.     | QvR   | Resolved from PvQ and ~PvR.
 6.     | PvR   | Resolved from PvQ and ~QvR.
 7.     | ~P    | Resolved from ~PvR and ~R.
 8.     | ~Q    | Resolved from ~QvR and ~R.
 9.     | Q     | Resolved from ~R and QvR.
 10.    | P     | Resolved from ~R and PvR.
 11.    | R     | Resolved from QvR and ~Q.
 12.    |       | Resolved R and ~R to Rv~R, which is in turn null.
A contradiction is found when ~R is assumed as true. Hence, R is true.
```