## ② Knowledge Base Resolution

→ A Sentence is in conjunctive Normal form (CNF) if it is a conjuction of clause, Each clause being a disjunction literals

$$\underbrace{(A \vee B)}_{\text{clause}} \wedge \underbrace{(C \vee D \vee \neg E)}_{\text{clause}} \wedge \underbrace{(E \vee G)}_{\text{clause}}$$

eg:- CNF conversion

$a \Leftrightarrow B \quad (a \Rightarrow B) \wedge (B \Rightarrow a)$

$a \Rightarrow B \quad \neg a \vee B$

$\neg(a \wedge B) \quad \neg a \vee \neg B$

$\neg(a \vee B) \quad \neg a \wedge \neg B$

$(A \wedge B) \Rightarrow C$

$\neg(A \wedge B) \vee C$

$(\neg A \vee \neg B) \vee C$

$(\neg A \vee \neg B \vee C)$

⇒ If a literal appears in one clause and its Negation in the other one, the two clauses can be merge & that literal can be discarded

⇒ we found an empty clause (KB does entails)

# code

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

def resolve(c1, c2):
    resolve_clause = set(c1) | set(c2)
```

```
for literal in (1):
    if negate_literal (literal) in (2):
        resolved_clause.remove (literal)
        resolved_clause.remove (negate_
                                literal (literal))
    return tuple (resolved_clause)

def resolution (knowledge_base):
while true:
    new clause = 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 knowledge_
                base:
            new clause.add (new_clause)
    if not new clause break.
    Knowledge_base != new_clause
return knowledge_base
if __name__ == "__main__":
    KB = {('p', 'q'), ('~p', 'r'), ('~r', '~r')}
    print ("Resolved KB", result)
```

O/P

Enter statement = ~negation

The statement not entailed by knowledge_base.

```
rules = 'PvQ ~PvR ~QvR' #P=>Q, 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.
```