AI LAB

LAB 9

Convert a given first order logic statement into Conjunctive Normal Form (CNF)

Algorithm:

- **Eliminate biconditionals and implications**
  - Replace:
  - A↔B with (A→B)∧(B→A)

- **Move NOT (¬) inwards (Negation Normal Form)**
  - Use De Morgan’s laws:
    - ¬(A ∧ B) → (¬A ∨ ¬B)
    - ¬(A ∨ B) → (¬A ∧ ¬B)
  - For quantifiers:
    - ¬∀x P(x) → ∃x ¬P(x)
    - ¬∃x P(x) → ∀x ¬P(x)

- **Standardize variables**
  - Rename bound variables so that each quantifier has a unique variable.

- **Skolemization (remove existential quantifiers)**
  - Replace ∃x P(x) with $P(f(y_1,...,y_n))$ where $y_1...y_n$ are universally quantified variables in scope.
  - This introduces **Skolem functions**.

- **Drop universal quantifiers**
  - After Skolemization, all remaining quantifiers are universal — can be dropped implicitly.

- **Distribute ∨ over ∧**
  - Apply distributive laws to obtain a conjunction of disjunctions:
    - (A∨(B∧C))(A ∨ (B ∧ C))(A∨(B∧C)) → (A∨B)∧(A∨C)(A ∨ B) ∧ (A ∨ C)(A∨B)∧(A∨C)

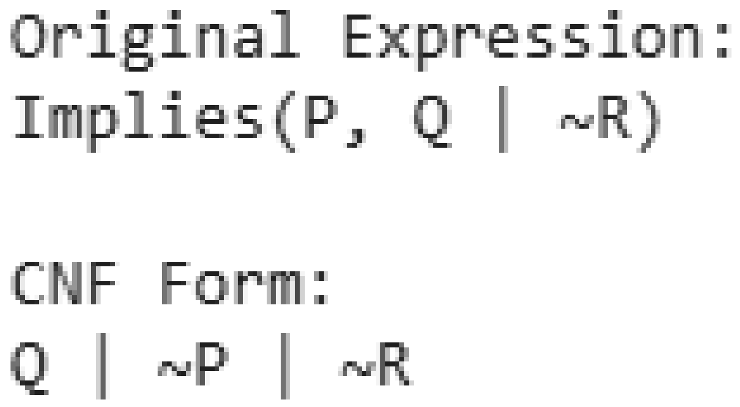
- **Simplify**

- Remove duplicate literals, tautologies, etc.

CODE:

```
from sympy import symbols
from sympy.logic.boolalg import to_cnf
from sympy.abc import x, y

# Define propositional variables (for simplicity)
P, Q, R = symbols('P Q R')

# Example: (P >> (Q | ~R))
expr = P >> (Q | ~R)

# Convert to CNF
cnf_expr = to_cnf(expr, simplify=True)

print("Original Expression:")
print(expr)
print("\nCNF Form:")
print(cnf_expr)
```

Output:



```
Original Expression:
Implies(P, Q | ~R)

CNF Form:
Q | ~P | ~R
```