Implement

1- the resolution inference rule,

2-the conjunctive normal form for sentences of propositional logic is also must be implemented as a sub of resolution module

Target: demonstrate how using resolution to derive contradictions suffices to prove a theorems' correctness.

What data type is used to represent a Prepositional Logic sentence or a clause?

#How semantic of sentence is added to the implementation?

How to get CNF?

How to implement resolution?

As a running example,

We prove the tautology $((A \vee B) \wedge (A \rightarrow C) \wedge (B \rightarrow C)) \rightarrow C$ by deriving a contradiction from the clausal list [[A, B], [¬A, C], [¬B, C], [¬C]]

1. The data types used are to record progress are initialized. We keep track of:

- The clause currently being used to generate resolvents (the resolution clause) -initialised to [] (the empty list).
- A list of the resolution clauses which have already been considered – initialized to [].
- The remainder of the clausal list - initialised to the full clausal list.
- A boolean value indicating whether or not a contradiction exists in the (original) KB - initialised to False.

2. The first clause from the remainder is set as the resolution clause - in this case [A, B]. We add the resolution clause to the list of already-considered clauses and delete it from the remainder, leaving [[¬A, C], [¬B, C], [¬C]]. Alternately, if the remainder is the empty list before the resolution clause is set, the algorithm terminates under the exhaustion condition.

3. The first constant of the resolution clause (the resolution constant) is taken as the constant we generate resolvents from - A at this point. This constant is temporarily

deleted from the resolution clause, leaving [B]. (Note this temporary resolution clause can be [].)

4. The resolution constant is negated (if it is already negated, double negation elimination is applied), giving ¬A, and the remainder is searched for all clauses containing this negation. In all clauses where it is found - here the single clause [¬A, C] -the negation is deleted and the temporary resolution clause is appended, leaving the resolvents - here the single resolvent [C, B].

5. The resolvents are filtered against the clauses in the remainder and those resolution clauses already considered - if any of the resolvents are a permutation of any of these clauses, they represent no new information and as such are excluded. Those remaining resolvents (if any) are added to the remainder - [C, B] is not a permutation of any of [¬A, C], [¬B, C], [¬C] or [A, B] and so is added.

6. At this point, if [] was one of the resolvents added to the remainder, we have derived our contradiction. If this is the case, the contradiction boolean is set to True and the algorithm terminates under the contradiction condition. Otherwise, there are two scenarios:

- New resolvents were inferred - the resolution constant is appended to the end of the resolution clause, here giving us [B, A], and the algorithm goes back to step 3. However, if all constants of the resolution clause have been considered, the algorithm moves to the scenario below.
- No new resolvents were inferred - the algorithm goes back to step 2.

The example given runs from start to finish as follows:

(A or B) ^ (¬A or C) ^ (¬B or C) ^ (¬C)

Initially clauses contains [A, B], [¬A, C], [¬B, C], [¬C]

Resolution Constant	Resolution Clause	Clauses with new resolvents
[]	[]	[A, B], [¬A, C], [¬B, C], [¬C]
A	[A, B]	[¬A, C], [¬B, C], [¬C], [B,C]
B	[A, B]	[¬A, C], [¬B, C], [¬C], [B,C], [A,C]
¬A	[¬A, C]	[¬B, C], [¬C], [B,C], [A,C],[C]
C	[¬A, C]	[¬B, C], [¬C], [B,C], [A,C], [C], [¬A]

¬B	[¬B, C]	[¬C], [B,C], [A,C], [C], [¬A]
C	[¬B, C]	[¬C], [B,C], [A,C], [C], [¬A], [¬B]
¬C	[¬C]	[B,C], [A,C], [C], [¬A], [¬B],[B],[A],[]

After inferring [], the algorithm terminates and the proof by contradiction is complete.

Another example

$$(p \vee r) \wedge (q \Leftarrow r) \wedge \neg q \wedge (t \Leftarrow p) \wedge \neg s \wedge (s \Leftarrow t)$$

I CNF

$$(p \vee r) \wedge (q \vee \neg r) \wedge \neg q \wedge (\neg p \vee t) \wedge \neg s \wedge (s \vee \neg t)$$

Contains a contradiction