## Resolution Theorem Proving with Exhaustive Search

```
RTP(Γ: Set of FOL sentences; φ; FOL sentence) return boolean {
1.   Δ = Γ ∪ {¬φ}                                          % Add the negated goal
2.   Δ = Convert Δ to CNF;
3.   Set<Clause> OLD = ∅;
4.   FIFOQueue<Clause> NEW = Δ;
5.   repeat {
6.        α = NEW.pop();

% Apply the rule of factoring to α
7.        OLD = OLD ∪ { α };
8.        γ = all possible factorings of α;
9.        for (each Clause C ∈ γ)
10.           if (C ∉ OLD ∪ NEW)
11.               NEW.push(C);

% Apply the rule of resolution to α with each clause β ∈OLD
12.       for (β ∈ OLD) {
13.           γ = all possible resolutions of α with β;
14.           for (each Clause C ∈ γ) {
15.               if (C == null clause) return TRUE;          % the null clause is reached.
16.               if (C ∉ OLD ∪ NEW)
17.                   NEW.push(C);
18.               }                                           % end for C
18.           }                                               % end for β
19.       } until (NEW is empty)                              % end repeat.
20.  return FALSE; % Every possible sequence of rule applications has been tried without success
21. }
```

This satisfies the following properties:

A. If $\phi$ is a consequence of $\Gamma$, then the algorithm returns TRUE.

B. If $\phi$ is not a consequence of $\Gamma$, then the algorithm either returns FALSE or goes into an infinite loop.

Now, the above is the simplest way to achieve these properties using resolution theorem proving. There are more sophisticated methods that are often more efficient in finding solutions, or that less often go into infinite loops. For instance, if you initialize OLD to be the clauses corresponding to $\Gamma$ and NEW to be the clauses corresponding to $\neg\phi$, that is often more efficient and avoids many infinite loops. But there is no algorithm that always returns FALSE if and only if $\phi$ is not a consequence of $\Gamma$.