Problem 1. Convert the following set of sentences to clausal form.

S1 : \( A \iff (B \lor E) \).

**Solution** . \( \neg A \lor B \lor E \land \neg B \lor A \land \neg E \lor A \).

S2 : \( E \Rightarrow D \).

**Solution** . \( \neg E \lor D \).

S3 : \( C \land F \Rightarrow \neg B \).

**Solution** . \( \neg C \lor \neg F \lor \neg B \).

S4 : \( E \Rightarrow B \).

**Solution** . \( \neg E \lor B \).

S5 : \( B \Rightarrow F \).

**Solution** . \( \neg B \lor F \).

S6 : \( B \Rightarrow C \).

**Solution** . \( \neg B \lor C \).

Give a trace of the execution of DPLL on the conjunction of these clauses.

Problem 2. Use resolution to prove the sentence \( \neg A \land \neg B \) from the clauses in Problem 1.

**Solution** . To prove the conjunction, it suffices to prove each literal separately. To prove \( B \), add the negated goal \( S7 : B \).

- Resolve \( S7 \) with \( S5 \), giving \( S8 : F \).
- Resolve \( S7 \) with \( S6 \), giving \( S9 : C \).
- Resolve \( S8 \) with \( S3 \), giving \( S10 : (\neg C \lor \neg B) \).
• Resolve S9 with S10, giving S11: \( \neg B \).
• Resolve S7 with S11 giving the empty clause.

To prove \( \neg A \), add the negated goal S7: A.
• Resolve S7 with the first clause of S1, giving S8: \( B \lor E \).
• Resolve S8 with S4, giving S9: B.
• Proceed as above to derive the empty clause.

**Problem 3 (Optional).** Is a randomly generated 4-CNF sentence with \( n \) symbols and \( m \) clauses more or less likely to be solvable than a randomly generated 3-CNF sentence with \( n \) symbols and \( m \) clauses. Explain.

**Solution.** It is more likely to be solvable: adding literals to disjunctive clauses makes them easier to satisfy.

**Problem 4 (Optional).** Any propositional logic sentence is logically equivalent to the assertion that each possible world in which it would be false is not the case. From this observation, prove that any sentence can be written in CNF.

**Solution.** Each possible world can be written as a conjunction of literals, e.g. \( (A \lor B \lor \neg C) \). Asserting that a possible world is not the case can be written by negating that, e.g. \( \neg(A \land B \land \neg C) \), which can be rewritten as \( \neg A \lor \neg B \lor C \). This is the form of a clause; a conjunction of these clauses is a CNF sentence, and can list the negations of all the possible worlds that would make the sentence false.

**Problem 5 (Optional).** A sentence is in disjunctive normal form (DNF) if it is the disjunction of conjunctions of literals. For example, the sentence \( (A \land B \land \neg C) \lor \neg (A \land C) \lor (B \land \neg C) \) is in DNF.

a. Any propositional logic sentence is logically equivalent to the assertion that some possible world in which it would be true is in fact the case. From this observation, prove that any sentence can be written in DNF.

**Solution.** Each possible world can be expressed as the conjunction of all the literals that hold in the model. The sentence is then equivalent to the disjunction of all these conjunctions, i.e., a DNF expression.

b. Construct an algorithm that converts any sentence in propositional logic into DNF. (*Hint: The algorithm is similar to the algorithm for conversion to CNF*).

**Solution.** A trivial conversion algorithm would enumerate all possible models and include terms corresponding to those in which the sentence is true; but this is necessarily exponential time. We can convert to DNF using the same algorithm as for CNF except that we distribute \( \land \) over \( \lor \) at the end instead of the other way round.
c. Construct a simple algorithm that takes as input a sentence in DNF and returns a satisfying assignment if one exists, or reports that no satisfying assignments exists.

**Solution**. A DNF expression is satisfiable if it contains at least one term that has no contradictory literals. This can be checked in linear time, or even during the conversion process. Any completion of that term, filling in missing literals, is a model.

d. Apply the algorithms in (b) and (c) to the following set of sentences:

- \( A \Rightarrow B \)
- \( B \Rightarrow C \)
- \( C \Rightarrow \neg A \).

**Solution**. The first steps give \((\neg A \lor B) \land (\neg B \lor C) \land (\neg C \lor \neg A)\). Converting to DNF means taking one literal from each clause, in all possible ways, to generate the terms (8 in all). Choosing each literal corresponds to choosing the truth value of each variable, so the process is very like enumerating all possible models. Here, the first term is \((\neg A \land \neg B \land \neg C)\), which is clearly satisfiable.

e. Since the algorithm in (b) is very similar to the algorithm for conversion in to CNF, and since the algorithm in (c) is much simpler than any algorithm for solving a set of sentences in CNF, why is this technique not used in automated reasoning?

**Solution**. The problem is that the final step typically results in DNF expressions of exponential size, so we require both exponential time AND exponential space.