Q1. Disjunctive Normal Form

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. 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.)

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 assignment exists.

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 the previous two parts to the following set of sentences:

\[
\begin{align*}
A & \implies B \\
B & \implies C \\
C & \implies \neg A
\end{align*}
\]

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 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?

The problem is that the final step typically results in DNF expressions of exponential size, so we require both exponential time AND exponential space.
Q2. In What Worlds?

(a) We wish to come up with hypotheses that entail the following sentences:

- \( S_1: X_1 \land X_2 \implies Y \)
- \( S_2: \neg X_1 \lor X_2 \implies Y \)

In this problem, we want to come up with a hypothesis \( H \) such that \( H \models S_1 \land H \models S_2 \).

(i) Assume we have the hypothesis \( H: Y \iff X_1 \lor X_2 \).

Does \( H \) entail \( S_1 \)?  
- Yes  
- No

Does \( H \) entail \( S_2 \)?  
- Yes  
- No

By looking at the truth table, you see that for all worlds \( H \) is true, \( S_1 \) is true. However, \( H \) does not entail \( S_2 \). One example is \( X_1 = \text{false}, X_2 = \text{false}, Y = \text{true} \).

(ii) Pretend that we have obtained a magical solver, \( SAT(s) \) which takes in a sentence \( s \) and returns true if \( s \) is satisfiable and false otherwise. We wishes to use this solver to determine whether a hypothesis \( H' \) entails the two sentences \( S_1 \) and \( S_2 \). Mark all of the following expressions that correctly return true if and only if \( H' \models S_1 \land H' \models S_2 \). If none of the expressions are correct, select “None of the above”.

- \( SAT(H' \land \neg (S_1 \land S_2)) \)
- \( SAT(H' \lor (S_1 \land S_2)) \)
- \( \neg SAT(H' \land \neg (S_1 \land S_2)) \)
- \( \neg SAT(H' \lor (S_1 \land S_2)) \)
- None of the above

Recall for \( H' \) to entail both \( S_1 \) and \( S_2 \), it must hold that \( H' \land \neg (S_1 \land S_2) \) is not satisfiable.

(b) Select which first-order logic sentences are syntactically correct translations for the following English sentences. You must use the syntax shown in class (eg. \( \forall, \exists, \land, \lor, \implies \)). Please select all that apply.

(i) At least one dish cooked by Alex was brought by Betty.

- \( \exists d \text{ Cooked}(A, d) \land \text{Brought}(B, d) \)
- \( \exists d \text{ Cooked}(A, d) \land [\exists d' (d' \neq d) \text{ Brought}(B, d')] \)
- \( \neg [\forall d \text{ Cooked}(A, d) \lor \text{Brought}(B, d)] \)
- \( \exists d_1, d_2 \text{ Cooked}(A, d_1) \land (d_2 = d_1) \land \text{Brought}(B, d_2) \)

(ii) At least one game played by Cathy is only played by people who brought dishes.

- \( \neg [\forall g \text{ Played}(C, g) \lor \exists p \text{ Played}(p, g) \implies \exists d \text{ Brought}(p, d)] \)
- \( \forall p \exists g \text{ Played}(C, g) \land \text{Played}(p, g) \implies \exists d \text{ Brought}(p, d) \)
- \( \exists g \text{ Played}(C, g) \implies \forall p \exists d \text{ Played}(p, g) \land \text{Brought}(p, d) \)
- \( \exists g \text{ Played}(C, g) \land [\forall p \text{ Played}(p, g) \implies \exists d, \text{ Brought}(p, d)] \)

(c) Assume we have the following sentence with variables \( A, B, C, \) and \( D \), where each variable takes Boolean values:

\[ S3 : (A \lor B \lor \neg C) \land (A \lor \neg B \lor D) \land (\neg B \lor \neg D) \]

(i) For the above sentence \( S3 \), state how many worlds make the sentence true. [Hint: you can do this and the next part without constructing a truth table]
(1) Clauses disjoint (2) clauses with \( k \) literals remove \( 2^{n-k} \) models.

The first clause removes \((A=F, B=F, C=T, D=T/F)\), second clause removes \((A=F, B=T, C=T/F, D=F)\), and third clause removes \((A=T/F, B=T, C=T/F, D=T)\) The three of them are mutually disjoint - so removed 2 + 2 + 4 = 8 in total.

(ii) Does \( S3 \models (A \land B \land D) ? \)

- [ ] Yes
- [x] No