Discussion Problems 4

Problem One: Connective Completeness

Show that using just the $\land$, $\lor$, $\neg$, $\to$, $\leftrightarrow$, $\top$, and $\bot$ connectives, it is possible to express any possible binary connective. That is, if you were to write out the truth table for an arbitrary binary connective $p \star q$, you could always find a formula in propositional logic that was equivalent to evaluating $p \star q$.

Problem Two: Disjunctive Normal Form

In lecture, we talked about propositional formulas in *conjunctive normal form* (CNF), in which all formulas were the many-way $\land$ of the many-way $\lor$ of literals (which were either variables or their negations). For example, the following formula is in CNF:

$$(x \lor \neg y) \land (x \lor \neg z \lor w) \land (w \lor x \lor y \lor z)$$

Another normal form for propositional formulas is *disjunctive normal form* (DNF), in which all formulas are the many-way $\lor$ of the many-way $\land$ of literals (which are either variables or their negations). For example, the following formula is in DNF:

$$(x \land y \land \neg z) \lor (\neg x \land y \land \neg z) \lor (\neg x \land \neg y)$$

i. Come up with a very efficient algorithm to solve SAT on formulas in DNF.

Suppose that you have an algorithm A that, given a formula in DNF, can count the number of satisfying assignments to A. That is, $A(\varphi)$ returns the total number of satisfying assignments to the DNF formula φ.

ii. Show how to write the negation of any CNF formula as a DNF formula.

iii. Create an algorithm that uses A as a subroutine to solve SAT for CNF formulas. Prove that your algorithm is correct. *(Hint: If there are n variables in a propositional formula, there are 2^n possible variable assignments.)*

Problem Three: Translating into Logic

i. Given the predicate *Person(x)*, which states that x is a person, and *Muggle(x)*, which states that x is a muggle, write a statement in first-order logic that says "some (but not all) people are muggles."

ii. Given the predicate *Person(x)*, which states that x is a person, and *Commoner(x)*, which states that x is a commoner, write a statement in first-order logic that says "there are either zero or one people who are not commoners."