Homework 2: Predicate Logic

Due date: Wednesday January 18th at 10 PM
If you work with others (and you should!), remember to follow the collaboration policy outlined in the syllabus.
In general, you are graded on both the clarity and accuracy of your work. Your solution should be clear enough that someone in the class who had not seen the problem before would understand it.
We sometimes describe approximately how long our explanations are. These are intended to help you understand approximately how much detail we are expecting. You are allowed to have longer explanations, but explanations significantly longer than necessary may receive deductions.

Be sure to read the grading guidelines on the assignments page for more information on what we're looking for.

1. Proof [23 points]

In Lecture 3 we gave a symbolic proof that $(p \wedge q) \vee (\neg p \wedge q) \vee (\neg p \wedge \neg q) \equiv (\neg p \vee q)$. In this problem we will give another proof.

(a) Our intuition for the proof in class was "the last two pieces of the formula correspond to vacuous truth." Identify a commonality in the first two pieces of the formula, and describe it.
(Your description should be similar in spirit to the one from class, but it's ok if you don't have a technical term like 'vacuous truth') [4 points]

(b) Give another proof of the formula that matches the intuition from part a instead of the intuition from class. [16 points]
Read the symbolic proof guidelines on the assignments page before you start.
Hint: your proof, if it matches your intuition from (a) will be different from the one from class – at least some of the intermediate expressions will be different.

(c) In class we labeled portions of the proof in purple with high-level descriptions of what they are doing (lecture 3 slide 36, left side). Produce similar labels for your proof. Submit your answer in the form "Steps [X] to [Y]: [label]" for each part. [3 points]
Note: The goal here is to give intuition for what is happening at a higher level than individual steps.

2. Circuit du Soleil [10 points]

In this problem, we'll construct two propositions in terms of the variables x, y, z and then use these propositions to build a circuit that computes a binary function $M(x, y, z)$.

(a) Give a propositional logic formula containing only the variables y and z which evaluates to $\neg y$ when z is false and evaluates to true when z is true. [2 points]

(b) Give a propositional logic formula containing only the variables z and x which evaluates to x when z is true and evaluates to true when z is false. [2 points]

(c) Now consider the binary function $M(x, y, z)$ which is defined as:

$$M(x, y, 1) := x$$

$$M(x, y, 0) := \neg y$$

Draw a circuit that takes x, y, z as input, uses only AND, OR, and NOT gates, and outputs $M(x, y, z)$. Your gates should not take more than two inputs.

Your answer for this part must combine your answers from (a) and (b)!) [6 points]

3. Think Contrapositive Be Contrapositive [15 points]

(a) You can easily become a multi-millionaire and take an infinitely long vacation at a Cancun resort, if you just solve one of the Millennium Problems.

   (i) convert this sentence to propositional logic (as on homework 1, ensure you're giving variables to **atomic propositions**, not compound ones). [2 points]

   (ii) take the contrapositive symbolically, and simplify so that $\neg$ signs are next to atomic propositions (i.e. only single variables). You are not required to show work for this part [2 points]

   (iii) translate the contrapositive back to English. [3 points]

   (iv) Compare English you've written down to original implication. Do they mean the same thing? (Just say "yes" or "no" here) [0.5 points]

(b) We will not reach artificial general intelligence, unless we design a better model.
Repeat steps (i)-(iv) from (a) for this sentence.

4. Two of a kind [20 points]

(a) Translate the Boolean Algebra expression $A \cdot (B + (B' + A)) + (A \cdot (A + B))'$ to Propositional Logic. Use the variables a and b to represent the propositions $A = 1$ and $B = 1$, respectively. [2 points]

(b) Prove that your solution to (a) is a tautology using a chain of equivalences. [16 points]

(c) Why do we know that the Boolean Algebra expression from part (a) will always evaluate to 1? Explain (1-2 sentences). [2 points]

5. A tale of two $\forall$ [12 points]

Consider the following two expressions:

$$\forall x(\mathsf{P}(x) \lor \mathsf{Q}(x)) \qquad \forall x(\mathsf{P}(x)) \lor \forall x(\mathsf{Q}(x))$$

(a) Give a domain of discourse and definitions of P and Q such that these expressions are **not** equivalent. Explain why your examples work (1-2 sentences). [6 points]

(b) Give a domain of discourse and definitions of P and Q such that these expressions **are** equivalent. Explain why your examples work (1-2 sentences). [6 points]

6. There is an implication [8 points]

Implications are uncommon under existential quantifiers. Consider this expression (which we'll call "the original expression"): $\exists x(\mathsf{P}(x) \to \mathsf{Q}(x))$

(a) Suppose that $\mathsf{P}(x)$ is not always true (i.e. there is an element in the domain for which $\mathsf{P}(x)$ is false). Explain why the original expression is true in this case. (1-2 sentences should suffice. If you prefer, you may give a formal proof instead).

(b) Suppose that $\mathsf{P}(x)$ is always true (i.e. $\forall x\, \mathsf{P}(x)$). There is a simpler statement which conveys the meaning of the original expression (i.e. is equivalent to it for all domains and predicates. By simpler, we mean "uses fewer symbols"). Give that expression, and briefly (1-2 sentences) explain why it works.

(c) When we do domain restriction incorrectly we'll often get an expression like the original expression – it doesn't usually mean what it might look like at first glance. Ponder, based on the last two parts, why it's very uncommon to write the original expression. You do not have to write anything for this part, simply ponder. [0 points]

7. [Extra Credit] 1111+1111 = Integer Overflow?

In this question you will construct a binary calculator equipped with the addition operator. For ease of understanding and writing, feel free to use Java syntax for loop structures and method structures.
Assume you are given two binary integers in the form of `boolean[]`s, where the first one is some `a = boolean[n]`, of length n, the second is some `b = boolean[m]`, of length m. You may assume that $m \leq n$. In this representation, every entry is a boolean (we're in binary!) $a[0]$ is the "least-significant-bit" (the "one's place").

your goal is to return their sum in binary (return it as a `boolean[]` as well, in the same format). For this problem you may **only** use the boolean operators $\neg, \wedge, \vee$ in finding the sum of these two binary integers.

You are free to use java-like syntax (including java structures like loops) along with propositional logic notation. But you may not use any operator other than $\neg, \wedge, \vee$ to combine/alter the booleans.

(for a greater challenge, limit yourself to only two of these operators, and consider why you cannot solve this task using only one of these operators).

Hint: Consider breaking this problem down. First, consider the maximal possible size of the output array. Then, isolate a single "column" in the addition, and consider what information you need to find the corresponding output digit. Then, try adding small binary numbers by hand and seeing how you would construct these addition operations using only boolean operators. Good luck!

8. Feedback [1 point]

Please keep track of how much time you spend on this homework and answer the following questions. This can help us calibrate future assignments and future iterations of the course, and can help you identify which areas are most challenging for you.

- How many hours did you spend working on this assignment (excluding any extra credit questions, if applicable)? Report your estimate to the nearest hour.
- Which problem did you spend the most time on?
- Any other feedback for us?