CSE 311: Foundations of Computing I

Homework 1 (due Friday, October 8th at 11:00 PM)

Directions: *Write up carefully argued solutions to the following problems. Your solution should be clear enough that it should explain to someone who does not already understand the answer why it works. However, you may use results from lecture, the theorems handout, and previous homeworks without proof.*

1. Feedback (0 points)

How much time (in minutes) did you spend on each problem of this homework? This information is useful for us to calibrate the difficulty of the problems for future offerings of this course.

2. I Can't Blab Such Blibber Blubber! (30 points)

Translate these English statements into logic, making the atomic propositions as simple as possible and exposing as much of the logic via symbols as possible.

(a) [5 Points] Define a set of *at most three* atomic propositions. Then, use them to translate "If you do not pay attention in class or start the assignment the day it is due, then you will not complete it on time."

(b) [10 Points] Define a set of *at most three* atomic propositions. Then, use them to translate all of these sentences about saving the village:

- i) If the dragons do not attack, the town will be safe.
- ii) The town will not be safe unless the wizards are awake.
- iii) The town will not be safe if and only if the dragons attack and the wizards are not awake.

(c) [10 Points] Define a set of *at most five* atomic propositions. Then, use them to translate all of these sentences about unicorns:

- i) If it has four hooves and says "neigh", then it either has a horn or it doesn't.
- ii) It has four hooves and says "neigh" if it is a horse or unicorn.
- iii) Only if it has four hooves, says "neigh", and has no horn is it a horse.

(d) [5 Points] Your professor says "You will get participation credit if you come to class and do not fall asleep."

Define a set of *at most three* atomic propositions. Then, use them give two different (i.e., non-equivalent) translations of the sentence that it is reasonable to believe might represent what the professor means. (If it is not obvious why those are reasonable, you can add some explanation.)

3. Thing One and Thing Two (20 points)

Use truth assignments (i.e., an assignment of a truth value to each variable) to demonstrate that the two propositions in each part are **not** logically equivalent (i.e., they don't always have the same truth value).

(a) [5 Points] $(p \vee r) \wedge p$ vs. $r \vee (p \wedge r)$

(b) [5 Points] $p \rightarrow (r \rightarrow s)$ vs. $(p \rightarrow r) \rightarrow s$

(c) [5 Points] $p \rightarrow (r \oplus s)$ vs. $(p \oplus r) \rightarrow (p \oplus s)$

(d) [5 Points] $(p \rightarrow r) \rightarrow (r \rightarrow p)$ vs. $(r \rightarrow p) \rightarrow (p \rightarrow r)$

4. You Can Make a Quick Trick Lock Stack (10 points)

Searching for a secure encryption function, you find a GitHub repo with three implementations: `A`, `B`, and `C`. The documentation in the repo tells you that **only one of the implementations is secure** — the other two are not! All three implementations are obfuscated, so you cannot tell which one is secure by examining the source. However, each implementation has a documentation sentence:

`A` "This implementation (`A`) is not secure."

`B` "This implementation (`B`) is not secure."

`C` "Implementation `A` is secure."

Unfortunately, the documentation in the repo also tells you that **only one documentation sentence is true** — the other two are false!

Which implementation is the *secure* one? (Not necessarily the one whose documentation sentence is true!) Justify your answer by considering each possibility for which implementation is the secure one, and showing which of the sentences would be true in each case. Only one possibility should match the description above.

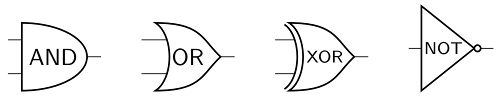
5. One, Two Three... How Many Fingers Do I See? (10 points)

Find a compound proposition involving the propositional variables r, s, and t that is true precisely when a majority of r, s, and t are true. (I.e., a proposition which is true in that case and false in all other cases.) Use a truth table to show that your answer works.

6. Would You, Could You, With a Gate? (10 points)

(a) Write a table showing the values of the boolean function $B(r, s, t)$ defined as follows. If r and s are both 1, then $B(r, s, t) = \bar{t}$. If $r = 1$ but $s = 0$, then $B(r, s, t) = t$. In all other cases, $B(r, s, t) = 1$.

(b) Draw the diagram of a circuit with three inputs (r, s, and t) that computes the function $B(r, s, t)$ using **at most one** of each of the following gates:



Hint: Start by trying to build circuits for the top half and bottom half of the table separately (pretending there were 0s in the other half). Then, try combining those circuits with an additional gate to get the correct output for the whole table.

7. Aunt Annie's Alligator (20 points)

Define the three-input "A" gate by the rules $A(F, r, s) = r$ and $A(T, r, s) = s$. In words, when the first input is F, the output of the A gate has the same value as its second input (r), and when the first input is T, the output of the A gate has the same value as its third input (s). For example, we have $A(F, T, F) = T$ since output is the same as the second input and $A(T, T, F) = F$ since the output is the same as the third input.

Show how to implement the following gates using only A's. You are allowed to use the constants T and F and the inputs a and b, as many times as you want, but you cannot use other gates like `AND`, `OR`, and `NOT`.

You must *justify your answers* if they are not obvious (to the grader).

(a) [4 Points] $\neg a$, using only **one** A gate

(b) [4 Points] $a \vee b$, using only **one** A gate

(c) [4 Points] $a \wedge b$, using only **one** A gate

(d) [4 Points] $a \rightarrow b$, using only **one** A gates

(e) [4 Points] $\neg a \wedge \neg b$, using at most **two** A gates

8. EXTRA CREDIT: XNORing (0 points)

Imagine a computer with a fixed amount of memory. We give names, $R_1, R_2, R_3, \ldots$, to each of the locations where we can store data and call these "registers." The machine can execute instructions, each of which reads the values from some register(s), applies some operation to those values to calculate a new value, and then stores the result in some register. For example, the instruction $R_4 := \texttt{AND}(R_1, R_2)$ would read the values stored in R_1 and R_2, compute the logical and of those values, and store the result in register R_4.

We can perform more complex computations by using a sequence of instructions. For example, if we start with register R_1 containing the value of the proposition p and R_2 containing the value of the proposition q, then the following instructions:

1. $R_3 := \texttt{NOT}(R_1)$
2. $R_4 := \texttt{AND}(R_1, R_2)$
3. $R_4 := \texttt{OR}(R_3, R_4)$

would leave R_4 containing the value of the expression $\neg p \vee (p \wedge q)$. Note that this last instruction reads from R_4 and also stores the result into R_4. This is allowed.

Now, assuming p is stored in register R_1 and q is stored in register R_2, give a sequence of instructions that

- only uses the `XNOR` operation (no `AND`, `OR`, etc.),
- only uses registers R_1 and R_2 (no extra space), and
- ends with q stored in R_1 and p stored in R_2 (i.e., with the original values in R_1 and R_2 swapped).