Chapter 1

Propositions and Formal Logic

1.1 Untitled 1

Important observation I wanted to highlight today:

> In math, we work with *functions* a lot. Recall the basic property of a function: it will take a value (a number, an animal, anything) and output another value (a letter of the alphabet, the number of the animal's legs, anything). Later on we will learn a more rigorous definition than that described above.

Then a boolean expression is really a function. Each variable takes in either true or false, at which point we can *evaluate* the expression and we get some boolean value (either true or false). Consider the expression

$$(p \wedge q) \vee r.$$

Then we can feed in any combination of trues or falses. For example, if $p \equiv T$, $q \equiv F$, and $r \equiv T$, then the expression above will evaluate these values to

$$(T \wedge F) \vee T \equiv F \vee T \equiv T,$$

or true.

A **truth table** then is really just a way of recording the output values of this function.

1. Create truth tables for the following expressions.

    - $p \wedge (\sim p)$ (This formula is called *unsatisfiable*. Why is this?)
    - $p \wedge T$ (Here, T represents true. Similarly, F represents false).
    - $p \wedge F$
    - $p \vee T$ and $p \vee F$.

> Two logical expressions are called **logically equivalent** if they have the same truth value given a set of truth values for each variable.
> Equivalently, two logical expressions are logically equivalent if they have the same truth table.

2. Let's consider the logical expressions in problem 1 again. Figure out simpler expressions which are equivalent to these.

3. Consider the following chunk of code. Here, `var1`, `var2` are boolean variables (ie, they are true or false).

```
if (var1 && (var1 || var2)) { printf("She loves me!"); }
else { printf("She loves me not..."); }
```

   Simplify the code.

4. In logical expressions, parentheses matter!

   Build the truth tables for $(\sim a) \wedge b$ and $\sim (a \wedge b)$. Compare it to the viral facebook math problem below:

Figure 1.1: Viral Facebook Math Meme

5. This problem mostly deals with the implies ($\implies$) logical operator. The motivation behind this operator is that eventually we want to do **proofs**. That is, given a set of initial hypotheses, we want to make deductions and logically deduce that a conclusion is true. For example, eventually all of you will be able to prove, roughly stated:

$$(n \text{ is even}) \implies (n+1 \text{ is odd}).$$

   Construct the truth tables for all the logical expressions below.

   - $p \implies (p \wedge q)$ and $p \implies (p \vee q)$.

- $(p \wedge q) \implies (p \vee q)$. This formula is an example of a *tautology*.
- $(p \vee q) \implies (p \wedge q)$.
- $(p \implies q) \implies (p \iff q)$.
- $(p \iff q) \implies (p \implies q)$.
- $p \implies q \implies r$ and $p \implies r$.

1.2 Recursive definition of formal expressions

In this worksheet, we will briefly discuss a property of boolean formulas called **satisfiability**. This was briefly discussed on the last worksheet, with respect to the boolean formula $p \wedge (\neg p)$.

> A boolean formula is **satisfiable** if there exists an assignment of true or false to each of its variables which makes the formula evaluate to true. Note that we only need *one* such assignment of variables. A boolean formula is **unsatisfiable** if it isn't satisfiable. In particular, all assignments of variables will evaluate to false.

For example, the formula $p \wedge q \wedge r$ is satisfiable (with satisfying assignment $p \equiv q \equiv r \equiv T$. And this is the ONLY satisfying assignment).

1. Determine whether or not the following Boolean formulae are satisfiable:

   - $x_1 \wedge x_2 \wedge \cdots \wedge x_n$, and $x_1 \vee x_2 \vee \cdots \vee x_n$, where n is a natural number.
   - $(x_1 \wedge (\neg x_2)) \vee (x_2 \wedge (\neg x_3)) \vee \cdots \vee (x_{n-1} \wedge (\neg x_n))$.
   - $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$.
   - $(x_1 \vee (\neg x_2) \vee x_3) \wedge (x_1 \vee (\neg x_2) \vee (\neg x_3)) \wedge ((\neg x_1) \vee x_2 \vee (\neg x_3))$.

2. Explain why a boolean formula with n variables has 2^n different assignments.

   a) Suppose I was to tell you that a certain boolean formula was satisfiable. What would I need to tell/demonstrate to you in order to convince you of this fact? How much time would it take?

   b) Suppose on the other hand I was to tell you that a certain boolean formula was *unsatisfiable*. What would I need to demonstrate to you in order to convince you of this fact? How much time would it take?

3. Sometimes, despite the computational time it takes to find a satisfying assignment, some unfortunate circumstance may require that you do so. This problem will outline one or two heuristics in order to possibly simplify your job a little further.

   Consider the boolean formula

   $$(x_1 \wedge x_5 \wedge x_2) \vee (x_3 \wedge (\neg x_3) \wedge x_5) \vee (x_2 \wedge (\neg x_7) \wedge x_6 \wedge x_1 \wedge (\neg x_4)).$$

   a) Notice that this formula is in *disjunctive normal form*. That is, it can be represented as an OR of ANDS. Explain how this might simplify our problem a little bit.

   b) Simplify the formula by considering the variable x_3.

   c) Discuss (with your neighbors) the theoretical or practical use of such simplifications.