Assignment 8

Alex Clemmer
CS 3100
Student number: u0458675

Problem 1:

(a) True, but must can be extended. If $1 = 2$, then $3 = 4$. That is, if two numbers are equal, they all must be equal. But this is also extensible: if this is true, then all numbers are equal, and you can point the arrow the other direction, too.

(b) True. $2 = 3$ iff $1 = 2$, because if one number is equal to another number, all numbers must be equal. This means a necessary condition is that if one of the two conditions is true, so must both be.

(c) True, but must be extended. If $gcd(x, y) = z$, then $gcd(x + y, y) = z$; that is, the way to increase the gcd is pick a new number that is smaller than both, but which divides into them. If you simply move the number up by the amount of the second number, you are not adding a significant new factor. But this can also be extended: if the second one is true, then so is the first one.

(d) True. Subtracting the two numbers cannot give you a number smaller than the gcd. Since this equation goes both ways, this is entirely true.

(e) True. If a graph has a clique of more than one vertex, then it also has a clique that is one vertex smaller than that. But this is not necessarily true in reverse: a graph with a clique of k vertices does not necessarily have a clique of $k + 1$ vertices.

(f) False. If a graph has a clique of k vertices, it does not necessarily have a clique of $k + 1$ vertices. But this is true in reverse, as we have seen above: if we have a clique of $k + 1$ vertices, as long as $k > 1$, this graph also has a clique of k vertices.

(g) True. If c logically follows from b, and the conclusion that $b \Rightarrow c$ follows logically from a, then both b and c must logically follow from a. But this is not necessarily true the other way: if b and c follow from a, it may not be the case that c also follows from b, let alone that the conclusion that $b \Rightarrow c$ follows from a.

Problem 2:

(a) This is a mapping reduction. $1, 2$, and 3 are all mapped to 1. A mapping reduction essentially asks if we can express one set in terms of another set. It's not important that numbers be mapped to different numbers, as if all $\{1, 2, 3\}$ can be expressed in terms of 1 number, the translation of principles between the two sets is still the same.

(b) This is **not** a mapping reduction. A function cannot map one input to two outputs.

(c) This reduction is not that hard. We can do it by functionally erasing every output from A_{tm}, then passing that input to A_{bt}, and accepting any time A_{bt} accepts. Since A_{bt} accepts only when the input $\in A_{tm}$, we can make similar statements about both machines.

The point of a mapping reduction is to take one set of outputs and transform it with some function into another set of outputs; even if the outputs of the first set are mapped all to one output of the second set, it is a mapping reduction, because we have shown that we can express one set purely in terms of another set. Thus similar principles will hold for both.

Here we are mapping all inputs of A_{tm} into A_{bt}, and therefore, similar principles will apply both ways. Since A_{tm} is undecidable, so too must be A_{bt}.

Problem 3:

Assuming that there are 2 basic operations (AND, OR), there are $N \cdot (N-1)^2$ possible functions. That is, we have N different numbers, which means that there are $N-1$ different operators relating them.

We are ignoring what each of the input's possible numbers are because we are calculating the number of possible Boolean functions, not the number of possible equations. So the next thing is, each operator can be one of thee things, so we can represent that with $(N-1)^2$.

If we would like to consider Boolean functions with more than 2 operators (*e.g.*, including XOR), we can just generalize the number as so: $N \cdot (N-1)^I$, where I is the number of operators.

Problem 4:

We know from 1(c) that we cannot create a TM that can tell whether any arbitrary machine accepts ε. These are similar cases, although 1(c) is more specific.

Because 1(c) is more specific, a reduction from A_{tm} is possible. So if we create this reduction, let's say that the function takes the output from A_{tm}, erases the tape, and prints out some CFG. If the machine that accepts CFG_{tm} accepts this, it accepts only when A_{tm} has accepted. We know that this is a contradiction, and thus, this too is not decidable.

Problem 5:

So given some CNF formula, we make a graph that has a vertex for every ordered pair (v, c), where v represents the variable, and c is the clause in the equation that has v. We form an edge wherever there the variable assignments are the same, but not negations of each other.

Now, given this, if we have k clauses in this CNF formula, then the k-vertex cliques in the graph end up representing ways of building the corresponding truth table. Thus, some formula is satisfiable iff there exists some k-vertex clique. Since we have shown that 3-SAT can thus be expressed using graphs, we have shown that the Clique problem and 3-SAT are equivalent.

This is clearly a many-to-one mapping reduction because we are taking boolean satisfiability and showing that it (and its important qualities) can be expressed in terms of a graph. By showing that these graphs have certain qualities, we can show that k-clique graphs have similar properties to Boolean satisfiability, and thus, that this problem is NPC.