CS 135 Spring 2018: Problem Set 2.

Problem 1. (10 points) Let $Fool(x, y, d)$ be a predicate that represents the statement "*x makes a fool of y on day d.*" Thus, for example, $\exists x\ \forall d\ Fool(x, Lem, d)$ means that there is someone who fools Lem every day.

Express each of the following statements as a quantified predicate.

a. Every day Lem fools someone. $\forall d \exists x : Fool(Lem, x, d)$

b. There is a person who, on each day, fools someone other than himself.
$\exists x \forall d \exists y : (x \neq y \wedge Fool(x, y, d)$

c. Everyone fools someone someday. $\forall x \exists y \exists d : Fool(x, y, d)$

d. On any day a person who is fooled does not fool anyone that day.
$\forall d \exists x \exists y : (Fool(x, y, d) \Rightarrow \forall z : \neg Fool(y, z, d))$

e. Lem never fools himself. $\forall d : \neg Fool(Lem, Lem, d)$

Now, let $Wise(x)$, $Future(d_1, d_2)$ respectively denote the predicates "x is wise" and "on day d_1, day d_2 lies in the future" (i.e., day d_2 comes after day d_1). Use these in addition to $Fool(x, y, d)$ to express the following statements.

f. A wise person never fools himself. $\forall d\ \forall x: W(x) \Rightarrow \neg Fool(x, x, d)$

g. If Lem fools a wise person someday then he never fools that person on any future day.
$\exists d_1 \exists x: \big(W(x) \wedge Fool(Lem, x, d_1)\big) \Rightarrow (\forall d_2 : Future(d_1, d_2) \Rightarrow \neg Fool(Lem, x, d_2))$

h. If someone fools Lem someday then Lem fooled himself someday in the past.
$\exists x \exists d_1 : Fool(x, Lem, d_1) \Rightarrow \big(\exists d_2 : Future(d_2, d_1) \wedge Fool(Lem, Lem, d_2)\big)$

i. Anyone who is fooled by the same person on more than one day is not wise.
$\forall x \exists y \exists d_1 \exists d_2 : ((d_1 \neq d_2) \wedge Fool(y, x, d_1) \wedge Fool(y, x, d_2)) \Rightarrow \neg W(x)$

j. Lem was fooled two different people other than himself, each on a different day.
$\exists d_1 \exists d_2 \exists x \exists y : (x \neq y) \wedge (d_1 \neq d_2) \wedge Fool(x, Lem, d_1) \wedge Fool(y, Lem, d_2)$

Problem 2. (10 points) In this problem we explore the use of logic in computer arithmetic. As you know, computers represent numbers using the bits (binary digits) 0 and 1. These bits represent (obviously!) the numbers zero and one respectively. The number two is represented as 10, three is 11, and four is 100. The four bit sequence $b_3 b_2 b_1 b_0$ represents the number $2^0 b_0 + 2^1 b_1 + 2^2 b_2 + 2^3 b_3$. As examples, the sequence 1001 represents $(2^3 \times 1) + (2^0 \times 1) = 8 + 1 = 9$, while the number thirteen is represented as 1101.

Suppose we have to design an addition circuit to add three one-bit numbers a_0, b_0, c_0. The possible values for the sum are 0, 1, 2, and 3. So the sum will be represented by a 2-bit number $s_1 s_0$. For example, 0+1+1 = 10, while 1+0+0 = 01.

To see the connection to logic, let us correspond the bit value 1 with the logical value T (True) and 0 with F (False). So we can consider a_0, b_0, c_0, s_1, s_0 to be logical variables.

a. Construct the truth tables for s_1 and s_0 in terms of the inputs a_0, b_0, c_0. For example, when $a_0 = T, b_0 = T, c_0 = F$, the truth values of $s_1 = T$ and $s_0 = F$.
b. Express s_0 as a proposition using the variables a_0, b_0, c_0 and the logical connectives $\neg, \wedge, \vee$. Write a similar expression for s_1.
c. Express s_0 and s_1 using only the NAND connective discussed in lecture.
d. Next, let's see how to add two 2-bit numbers a_1a_0, b_1b_0 to produce the 3-bit result $s_2s_1s_0$. Recall how we usually add numbers – we first add the lowest order bits (a_0 and b_0) to get the value s_0 as well as a "carry bit" which when added with a_1 and b_1 produces s_1 and a carry bit (which is s_2 for 2-bit numbers). Express each of s_2, s_1, s_0 in terms of a_1, a_0, b_1, b_0 using the NAND connective.