Homework 6
Due: Wednesday, March 23, 2016

All homeworks are due at 12:55 PM in the CS22 bin on the CIT second floor, next to the Fishbowl.

Include our cover sheet or equivalent, write your Banner ID (but not your name or your CS login) on each page of your homework, label all work with the problem number, and staple the entire handin before submitting.

Be sure to fully explain your reasoning and show all work for full credit. Consult the style guide for more information.

Problem 1

a. As discussed in class, boolean formulas can be represented algebraically where true and false are mapped to 0/1 bits. On these bits, addition is defined where \(0 + 0 = 0, 1 + 0 = 1, 0 + 1 = 1, \text{ and } 1 + 1 = 1\), which is equivalent to OR, and multiplication is defined as where \(0 \cdot 0 = 0, 1 \cdot 0 = 0, 0 \cdot 1 = 0, \text{ and } 1 \cdot 1 = 1\), which is equivalent to AND.

In this problem, we consider another way to represent boolean formulas. They can be expressed equivalently as integers modulo 2 where 0 represents false and 1 represents true. When translating to this form, AND is represented by multiplication, XOR is represented by addition, and NOT is represented by adding one. For example, \(p \land q\) is congruent to \(pq\) (mod 2), \(p \oplus q\) is congruent to \((p + q)\) (mod 2), and \(\neg p\) is congruent to \((p + 1)\) (mod 2).

Convert the following propositions over the set \(S = \{p, q, r, s\}\) to algebraic expressions mod 2.

i. \((p \land q \land r) \oplus s\)

ii. \((p \land \neg q) \oplus (q \land \neg r)\)

iii. \((p \lor q)\)

iv. \((p \implies q)\)

b. Let \(S = \{p_1, p_2, \ldots, p_n\}\) be a set of propositions.

Consider propositions formed from the elements of \(S\) in the following way:

- You can use any \(p_i\), but not \(\neg p_i\).
- You can AND together any propositions into a term. A term can also be the tautology, 1.
• You can XOR together terms described above.

For example, \((p_1 \land p_2 \land p_3) \oplus (p_3 \land p_4) \oplus p_2 \oplus 1\) is of the correct form. \(p_1 \Rightarrow p_2\), \((p_1 \land \neg p_2) \oplus p_1\), and \(p_1 \lor (p_2 \land p_3)\) are not of the correct form.

Express the follow propositions over the set \(S = \{p, q, r, s\}\) in the given form, showing all steps.

i. \(p \land \neg q \land r \land s\)
ii. \((p \Rightarrow q) \Rightarrow r\)
iii. \((p \oplus q) \lor r\)

Solution

a. i. \((p \land q \land r) \oplus s\) is congruent to \((pqr + s) \mod 2\).

ii. \((p \land \neg q) \oplus (q \land \neg r)\) is congruent to
\[
(p(q + 1) + q(r + 1)) \equiv (pq + p + qr + q) \mod 2.
\]

By using De Morgan’s laws, \(p \lor q = \neg(\neg p \land \neg q)\). That is congruent to
\[
((p + 1)(q + 1) + 1) \equiv (pq + p + q + 1 + 1) \mod 2
\]
\[\equiv (pq + p + q) \mod 2.
\]

Because \(p \Rightarrow q = \neg p \lor q\), it is congruent to
\[
((p + 1) + q + (p + 1)q) \equiv (p + 1 + q + pq + q)
\]
\[\equiv (pq + p + 1) \mod 2.
\]

b. We can express each boolean formula in the desired form by converting it to mod 2 form and converting that formula such that it is composed of sums of products of propositions. Then, those can be converted back to the boolean formula form using XOR and AND.

i. \(p \land \neg q \land r \land s\) is congruent to
\[
p(q + 1)rs \equiv (pqrs + prs) \mod 2.
\]

Then, we can covert back to a boolean formula: \((p \land q \land r \land s) \oplus (p \land r \land s)\).
ii. 
\[(p \Rightarrow q) \Rightarrow r = \neg(p \vee q) \vee r\]
\[
(p(q + 1) + r + p(q + 1)r) \equiv (pq + p + r + pqr + pr) \pmod{2}
\]
\[
p \oplus r \oplus (p \land q) \oplus (p \land r) \oplus (p \land q \land r)
\]

iii. 
\[(p \oplus q) \vee r\]
\[
((p + q) + r + (p + q)r) \equiv (p + q + r + pr + qr) \pmod{2}
\]
\[
p \oplus q \oplus r \oplus (p \land r) \oplus (q \land r)
\]

This way of representing boolean formulas is called Algebraic Normal Form because of this connection to the integers mod 2.

Problem 2

For each of the following, answer the questions and provide your reasoning.

a. Can AND (\&) be expressed using only OR (\lor) and NOT (\neg)?

b. Can OR (\lor) be expressed using only AND (\&) and NOT (\neg)?

c. Can NOT (\neg) be expressed using only AND (\&) and OR (\lor)?

d. Recall that a set of Boolean operators is complete if all possible truth tables can be expressed using those operators. Is the set \{OR, AND\} complete?

e. Is the set \{OR, AND, NOT\} complete?

f. Recall the \star operator from Homework 5 where \(p \star q = \neg(p \land q)\). Is the set \{\star\} complete?

Solution

a. Yes. Using de Morgan’s laws, \((x \land y)\) can be written as \(\neg(\neg x \lor \neg y)\)

b. Yes. Using de Morgan’s laws, \((x \lor y)\) can be written as \(\neg(\neg x \land \neg y)\)

c. No. NOT only operates on one variable, say, \(x\). Thus, to emulate NOT, we can only operate on one variable

We know the following: if \(x\) is true, \(x \lor x\) and \(x \land x\) are also true. If \(x\) is false, \(x \lor x\) and \(x \land x\) are false. \(\lor\) and \(\land\) will output the value of their inputs when
their inputs are the same.

Suppose that we construct a proposition that has only one input and uses a combination of $\lor$, $\land$ operations. Consider when the input, $x$, is false. We can split our operations into two types:

- Operations that have $x$ directly as an input
- Operations that have the output of another operation as an input

All operations of the former will output false (as reasoned above). Since all the operations of type 2 will only have false inputs, all operations of type 2 will output false. Since every operation outputs false, we can conclude that every combination of $\land$, $\lor$ operations will output false when the input is false.

In order to represent negation, we must output true when given a false input. Since no sequence of $\land$, $\lor$ operations can do this, we can conclude that we cannot express negation using only $\land$, $\lor$.

\textbf{d. No.} Because we showed that \textit{Not} cannot be represented by \textit{And} and \textit{Or}, we cannot represent the following truth table using only \textit{And} and \textit{Or}:

\begin{tabular}{c|c}
$p$ & $\neg p$ \\
\hline
T & F \\
F & T \\
\end{tabular}

Therefore, we cannot represent all truth tables using \textit{And} and \textit{Or}, so they are not complete.

\textbf{e. Yes.} Any Boolean function can be written using only these three operators.

Consider some Boolean function $f(p_1, \ldots, p_k)$ on $k$ propositions. This function’s input/output behavior can be completely defined by using a truth table with $2^k$ rows; we want to show that we can also write this function as some Boolean formula $F$ using only the operators \textit{And}, \textit{Or}, and \textit{Not}. We can do this by using the truth table to convert the function to \textit{disjunctive normal form}.

Intuitively, disjunctive normal form enumerates every assignment of the input propositions that make the function return \textit{true}. Each assignment of the input propositions corresponds to a single row of the truth table; if we have $t$ rows of the table which have \textit{true} in the output column, we will be setting

$$F = r_1 \lor r_2 \lor \cdots \lor r_t,$$

where each $r_i$ is an expression corresponding to one of these \textit{true} rows. Since we are then disjuncting ("ORing") them together, our final expression will be true if and only if the inputs $p_1, \ldots, p_k$ match one of these assignments.
For the $i$th row of the truth table which evaluates to true, let $v_1, \ldots, v_k$ be the values which are assigned to each of the input propositions, where each $v$ is either 0 or 1. We will define the expression $r_i$ for this row by conjuncting (“Anding”) every proposition $p_j$ where $v_j = 1$, and the negation of every proposition $p_j$ (i.e. $\neg p_j$) where $v_j = 0$. This expression will be true if and only if the propositions have exactly the values $v_1, \ldots, v_k$, which is what we hoped to find.

We now have an $F$, written using only And, Or, and Not, which is true if and only if $f$ is true. However, if $t$ is 0 – in other words, if the function $f$ is always false – then $F$ is currently completely empty! Thus, we need one slight modification: we can instead write

$$F = (p_1 \land \neg p_1) \lor r_1 \lor r_2 \lor \cdots \lor r_t.$$  

Since the first expression is always false, we have not changed $F$ if $t > 0$, but $F$ is false if $t = 0$.

Since all Boolean functions can be written out in a truth table, and we have given a general-purpose algorithm for constructing an equivalent expression from a truth table, the set $\{\land, \lor, \neg\}$ is complete.

f. Yes. From Homework 5, we showed that $\neg p = p \star p$, $p \land q = (p \star q) \star (p \star q)$, and $p \lor q = (p \star p) \star (q \star q)$. Because we can express And, Or, and Not using $\star$ and we can express all truth tables using those three operations, we can express all truth tables using only the $\star$ operator.

Thus, $\star$ is complete.

Problem 3

Six of the employees from the Office go to a nearby store to buy groceries. They claim to have paid already, but the money doesn’t add up, so they conclude that one of the employees hasn’t paid yet. The employee who has not paid always lies, but the other five are always honest.

- Pam: Jim has already paid.
- Dwight: Either Jim or Ryan hasn’t paid yet.
- Michael: Angela has already paid.
- Jim: Pam has already paid.
- Ryan: Dwight hasn’t paid yet.
- Angela: Either Michael or Ryan hasn’t paid yet.
Which employee hasn’t paid yet? Justify your answer.

**Solution**

For each employee, we can suppose they are telling the truth and see if it leads to a contradiction. Here’s the supposition that gives you the solution:

1) Suppose Ryan is a non-liar.
2) .∴ Dwight is a liar *(what Ryan said)*
3) .∴ What Dwight says is false *(definition of liar)*
4) .∴ Angela is telling the truth *(there’s only one liar)*
5) .∴ Either Michael or Ryan hasn’t paid yet *(what Angela said)*
6) .∴ Michael is a liar *(by statement 5, statement 1, and elimination)*
7) .∴ We have a contradiction: Dwight and Michael are both liars but there’s only one liar.
8) .∴ The supposition that Ryan is a non-liar is false *(by contradiction rule)*
9) .∴ Ryan is the liar *(by negation of supposition)*

**Problem 4**

Michael can’t decide whether he should fire Creed or not, so he gets Jim, Pam, and Dwight to help him decide.

a. Say he asks each of his chosen three employees to vote (either yes or no). Michael will go with whichever option receives the majority of the votes.
   
   Model this problem as a circuit using only AND, OR, and NOT gates.

b. Michael decides this is too simple. Instead, he will switch his choice every time one of the three employees switches his/her choice. The initial state is that all three employees and Michael say no.
   
   For example, say Michael wants to fire Creed, and there is currently exactly one employee who thinks he should. If one of the other two employees changes their mind, and decides that Michael *should* fire Creed, then Michael will also change his mind – he will not fire Creed.
   
   Model this problem as a circuit using only AND, OR, and NOT gates.
Problem 5

Dwight has had enough of Jim stealing his desk supplies. He decides to build an electronic lock for his desk drawers. Help him build a combinatorial circuit to keep his stuff safe.

The password will be represented as a three bit integer. When the correct password is entered into the three input wires, the output wire will activate to unlock his drawers. In order to prevent Jim’s cunning attempts to discover the password, Dwight also
wants to be able to change the password regularly.

a. Build a circuit in Logisim with three “password entry” input wires for inputting the password, and three more “password definition” wires whose state indicates what the password actually is. When these match pairwise, the output wire should active.

b. Now copy your circuit and extend it to allow saving the password persistently. To do this, add a wire to signify that the password should be changed. When the correct old password has been entered on the first three wires and the password change wire is activated, the password entered on the three password definition wires will become the new password. (This means there are 7 input wires in total.) Use D flip-flops (but no clock) and logic gates to model this circuit in Logisim.

For this problem you can use all logic gates and more than two input wires to a gate if necessary.
Solution