

Solutions to In-Class Problems Week 2, Wed.

Problem 1.

Prove by truth table that OR distributes over AND:

$$[P \text{ OR } (Q \text{ AND } R)] \text{ is equivalent to } [(P \text{ OR } Q) \text{ AND } (P \text{ OR } R)] \quad (1)$$

Solution.

| <u>P</u> | <u>OR</u> | (Q | AND | R) |
|----------|-----------|----|-----|----|
| T | T | T | T | T |
| T | T | T | F | F |
| T | T | F | F | T |
| T | T | F | F | F |
| F | T | T | T | T |
| F | F | T | F | F |
| F | F | F | F | T |
| F | F | F | F | F |

| (P | OR | Q) | <u>AND</u> | (P | OR | R) |
|----|----|----|------------|----|----|----|
| T | T | T | T | T | T | T |
| T | T | T | T | T | T | F |
| T | T | F | T | T | T | T |
| T | T | F | T | T | T | F |
| F | T | T | T | F | T | T |
| F | T | T | F | F | F | F |
| F | F | F | F | F | T | T |
| F | F | F | F | F | F | F |

The two columns for the principle operator (underlined) are the same, and therefore the corresponding propositional formulas are equivalent. ■

Problem 2.

This problem¹ examines whether the following specifications are *satisfiable*:

1. If the file system is not locked, then
 - (a) new messages will be queued.
 - (b) new messages will be sent to the messages buffer.
 - (c) the system is functioning normally, and conversely, if the system is functioning normally, then the file system is not locked.
2. If new messages are not queued, then they will be sent to the messages buffer.
3. New messages will not be sent to the message buffer.

(a) Begin by translating the five specifications into propositional formulas using four propositional variables:

$L ::=$ file system locked,
 $Q ::=$ new messages are queued,
 $B ::=$ new messages are sent to the message buffer,
 $N ::=$ system functioning normally.

Solution. The translations of the specifications are:

| | |
|---------------------|---------------|
| NOT L IMPLIES Q | (Spec. 1.(a)) |
| NOT L IMPLIES B | (Spec. 1.(b)) |
| NOT L IFF N | (Spec. 1.(c)) |
| NOT Q IMPLIES B | (Spec. 2.) |
| NOT B | (Spec. 3.) |

■

(b) Demonstrate that this set of specifications is satisfiable by describing a single truth assignment for the variables L, Q, B, N and verifying that under this assignment, all the specifications are true.

Solution. An assignment that works is

$L =$ True
 $N =$ False
 $Q =$ True
 $B =$ False.

To find this assignment, we could have started constructing the sixteen line truth table—one line for each way of assigning truth values to the four variables $L, N, Q,$ and B —and calculated the

¹From Rosen, 5th edition, Exercise 1.1.36

truth value of the AND of all the five specifications under that assignment, continuing until we got one that made the AND-formula true.

If for every one of the sixteen possible truth assignments, the AND-formula was false, then the system is not satisfiable. ■

(c) Argue that the assignment determined in part (b) is the only one that does the job.

Solution. We can avoid calculating all 16 rows of the full truthtable calculation suggested in the previous solution to part (b) by reasoning as follows. In any truth assignment that makes all five specifications true,

- B must be false, or the last specification, (Spec. 3.), would be false.
- Given that B is false, (Spec. 2.) and (Spec. 1.(b)) can be true only if Q and L are true.
- Given that L is true, (Spec. 1.(c)) can be true only if N is false.

Thus, in order for all five specifications to be true, the assignment must be:

$$\begin{aligned} L &= \text{True} \\ N &= \text{False} \\ Q &= \text{True} \\ B &= \text{False.} \end{aligned}$$

■

Problem 3.

When the Mathematician says to his student, "If a function is not continuous, then it is not differentiable," then letting D stand for "differentiable" and C for continuous, the only proper translation of the Mathematician's statement would be

$$\text{NOT}(C) \text{ IMPLIES } \text{NOT}(D),$$

or equivalently,

$$D \text{ IMPLIES } C.$$

But when a Mother says to her son, "If you don't do your homework, then you can't watch TV," then letting T stand for "watch TV" and H for "do your homework," a reasonable translation of the Mother's statement would be

$$\text{NOT}(H) \text{ IFF } \text{NOT}(T),$$

or equivalently,

$$H \text{ IFF } T.$$

Explain why it is reasonable to translate these two IF-THEN statements in different ways into propositional formulas.

Solution. We know that a differentiable function must be continuous, so when a function is not continuous, it is also not differentiable. Now Mathematicians use IMPLIES in the technical way given by its truth table. In particular, if a function *is* continuous then to a Mathematician, the implication

$$\text{NOT}(C) \text{ IMPLIES } \text{NOT}(D),$$

is automatically true since the hypothesis (left hand side of the IMPLIES) is false. So whether or not continuity holds, the Mathematician could comfortably assert the IMPLIES statement knowing it is correct.

And of course a Mathematician does *not* mean IFF, since she knows a function that is not differentiable may well be continuous.

On the other hand, while the Mother certainly means that her son cannot watch TV if he does not do his homework, both she and her son *most likely* understand that if he *does* do his homework, then he *will* be allowed watch TV. In this case, even though the Mother uses an IF-THEN phrasing, she really means IFF.

On the other hand, circumstances in the household might be that the boy may watch TV when he has not only done his homework, but *also* cleaned up his room. In this case, just doing homework would not imply being allowed to watch TV –the boy won't be allowed to watch TV if he hasn't cleaned his room, even if he has done his homework.

The general point here is that semantics (meaning) trumps syntax (sentence structure): even though the Mathematician's and Mother's statements have the same structure, their meaning may warrant different translations into precise logical language.

■

Problem 4.

Propositional logic comes up in digital circuit design using the convention that **T** corresponds to 1 and **F** to 0. A simple example is a 2-bit half-adder circuit. This circuit has 3 binary inputs, a_1 , a_0 and b , and 3 binary outputs, c , o_1 , o_0 . The 2-bit word a_1a_0 gives the binary representation of an integer, s between 0 and 3. The 3-bit word co_1o_0 gives the binary representation of $s + b$. The output c is called the *final carry bit*.

So if s and b were both 1, then the value of a_1a_0 would be 01 and the value of the output co_1o_0 would 010, namely, the 3-bit binary representation of $1 + 1$.

In fact, the final carry bit equals 1 only when all three binary inputs are 1, that is, when $s = 3$ and $b = 1$. In that case, the value of co_1o_0 is 100, namely, the binary representation of $3 + 1$.

This 2-bit half-adder could be described by the following formulas:

$$\begin{aligned} c_0 &= b \\ o_0 &= a_0 \text{ XOR } c_0 \\ c_1 &= a_0 \text{ AND } c_0 && \text{the carry into column 1} \\ o_1 &= a_1 \text{ XOR } c_1 \\ c_2 &= a_1 \text{ AND } c_1 && \text{the carry into column 2} \\ c &= c_2. \end{aligned}$$

(a) Generalize the above construction of a 2-bit half-adder to an $n + 1$ bit half-adder with inputs a_n, \dots, a_1, a_0 and b for arbitrary $n \geq 0$. That is, give simple formulas for o_i and c_i for $0 \leq i \leq n + 1$, where c_i is the carry into column i and $c = c_{n+1}$.

Solution. The $n + 1$ -bit word $a_n \dots a_1 a_0$ will be the binary representation of an integer, s , between 0 and $2^{n+1} - 1$. The circuit will have $n + 2$ outputs c, o_n, \dots, o_1, o_0 where the $n + 2$ -bit word $co_n \dots o_1 o_0$ gives the binary representation of $s + b$.

Here are some simple formulas that define such a half-adder:

$$\begin{aligned} c_0 &= b, \\ o_i &= a_i \text{ XOR } c_i && \text{for } 0 \leq i \leq n, \\ c_{i+1} &= a_i \text{ AND } c_i && \text{for } 0 \leq i \leq n, \\ c &= c_{n+1}. \end{aligned}$$

■

(b) Write similar definitions for the digits and carries in the sum of two $n + 1$ -bit binary numbers $a_n \dots a_1 a_0$ and $b_n \dots b_1 b_0$.

Solution. Define

$$\begin{aligned} c_0 &= 0 \\ o_i &= a_i \text{ XOR } b_i \text{ XOR } c_i && \text{for } 0 \leq i \leq n, \\ c_{i+1} &= (a_i \text{ AND } b_i) \text{ OR} \\ &\quad (a_i \text{ AND } c_i) \text{ OR } (b_i \text{ AND } c_i) && \text{for } 0 \leq i \leq n, \\ c &= c_{n+1}. \end{aligned}$$

■

Visualized as digital circuits, the above adders consist of a sequence of single-digit half-adders or adders strung together in series. These circuits mimic ordinary pencil-and-paper addition, where a carry into a column is calculated directly from the carry into the previous column, and the carries have to ripple across all the columns before the carry into the final column is determined. Circuits with this design are called “ripple-carry” adders. Ripple-carry adders are easy to understand and remember and require a nearly minimal number of operations. But the higher-order output bits and the final carry take time proportional to n to reach their final values.

(c) How many of each of the propositional operations does your adder from part (b) use to calculate the sum?

Solution. The scheme given in the solution to part (b) uses $3(n + 1)$ AND’s, $2(n + 1)$ XOR’s, and $2(n + 1)$ OR’s for a total of $7(n + 1)$ operations.²

■

²Because c_0 is always 0, you could skip all the operations involving it. Then the counts are $3n + 1$ AND’s, $2n + 1$ XOR’s, and $2n$ OR’s for a total of $7n + 2$ operations.

The Propositional Operations

| P | NOT P |
|----------|----------|
| T | F |
| F | T |

| P | Q | P AND Q |
|----------|----------|-------------|
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | F |

| P | Q | P OR Q |
|----------|----------|------------|
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |

| P | Q | P XOR Q |
|----------|----------|-------------|
| T | T | F |
| T | F | T |
| F | T | T |
| F | F | F |

| P | Q | P IMPLIES Q |
|----------|----------|-----------------|
| T | T | T |
| T | F | F |
| F | T | T |
| F | F | T |

| P | Q | P IFF Q |
|----------|----------|-------------|
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | T |

MIT OpenCourseWare
<http://ocw.mit.edu>

6.042J / 18.062J Mathematics for Computer Science
Spring 2010

For information about citing these materials or our Terms of Use, visit: <http://ocw.mit.edu/terms>.