Staff Solutions to In-Class Problems Week 3, Tue.

Problem 1.
For each of the logical formulas, indicate whether or not it is true when the domain of discourse is \( \mathbb{N} \) (the nonnegative integers 0, 1, 2, \ldots), \( \mathbb{Z} \) (the integers), \( \mathbb{Q} \) (the rationals), \( \mathbb{R} \) (the real numbers), and \( \mathbb{C} \) (the complex numbers). Add a brief explanation to the few cases that merit one.

\[
\begin{align*}
\exists x. x^2 &= 2 \\
\forall x. \exists y. x^2 &= y \\
\forall y. \exists x. x^2 &= y \\
\forall x \neq 0. \exists y. xy &= 1 \\
\exists x. \exists y. x + 2y &= 2 \text{ AND } 2x + 4y &= 5
\end{align*}
\]

**STAFF NOTE:** The few brief explanations for entries below are sufficient.

Intervene if teams start to go overboard with adding explanations (unlikely). After the problem has been team-approved (team check on their board), you can challenge a team member to provide an omitted explanation. If they had sufficient explanations (common), I like to challenge a team member with a “meta”-question, “Which was the hardest entry to fill in, and why?”

Solution.

<table>
<thead>
<tr>
<th>Statement</th>
<th>( \mathbb{N} )</th>
<th>( \mathbb{Z} )</th>
<th>( \mathbb{Q} )</th>
<th>( \mathbb{R} )</th>
<th>( \mathbb{C} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \exists x. x^2 = 2 )</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>( \forall x. \exists y. x^2 = y )</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>(( y = x^2 )</td>
<td>T</td>
</tr>
<tr>
<td>( \forall y. \exists x. x^2 = y )</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>( \forall x \neq 0. \exists y. xy = 1 )</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>(( y = 1/x ))</td>
<td>T</td>
</tr>
<tr>
<td>( \exists x. \exists y. x + 2y = 2 \text{ AND } 2x + 4y = 5 )</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Problem 2.
The goal of this problem is to translate some assertions about binary strings into logic notation. The domain of discourse is the set of all finite-length binary strings: \( \lambda, 0, 1, 00, 01, 10, 11, 000, 001, \ldots \) (Here \( \lambda \) denotes the empty string.) In your translations, you may use all the ordinary logic symbols (including =), variables, and the binary symbols 0, 1 denoting 0, 1.

A string like 01x0y of binary symbols and variables denotes the concatenation of the symbols and the binary strings represented by the variables. For example, if the value of \( x \) is 011 and the value of \( y \) is 1111, then the value of 01x0y is the binary string 0101101111.

Here are some examples of formulas and their English translations. Names for these predicates are listed in the third column so that you can reuse them in your solutions (as we do in the definition of the predicate NO-1S below).
Meaning | Formula | Name
---|---|---
x is a prefix of $y$ | $\exists z\ (xz = y)$ | PREFIX\((x, y)\)
x is a substring of $y$ | $\exists u\exists v\ (uxv = y)$ | SUBSTRING\((x, y)\)
x is empty or a string of 0's | NOT\((\text{SUBSTRING}(1, x))\) | NO-1S\((x)\)

(a) $x$ consists of three copies of some string.

**Solution.** $\exists y\ (x = yy)$

(b) $x$ is an even-length string of 0's.

**Solution.** NO-1S\((x)\) AND $\exists y\ (x = yy)$

Some students mentioned $\lambda$ in their formulas. Technically, this is not allowed, so they need to justify it by giving a formula that means “$x = \lambda$.” This is easy, for example: $x = xx$.

A serious mistake was to try writing a recursive definition of a predicate calculus formula, as in

$$P(x) ::= x = \lambda \text{ OR } \exists y. x = 00y \text{ AND } P(y). \tag{1}$$

Such recursive formulas are, by definition, not part of predicate calculus— with good reason. Definition 1 resembles a simple recursive definition of a procedure to test if $x$ is an even length string of 0's, and its meaning might be explained in procedural terms. But it’s hard to figure out in general what recursively defined formulas mean. For example, let $n$ be an integer-valued variable, and suppose we tried to define a formula, $Q(n)$, that means $n$ is positive:

$$Q(n) ::= (n = 0 \text{ OR } \text{NOT} (Q(n + 1))) \text{ AND } (n = 1 \text{ OR } Q(n - 1)).$$

might succeed in giving a procedural explanation for this example, but it is not immediately clear whether the definition is inconsistent with itself (as, for example, $Q(n) ::= \text{NOT } Q(n)$ is).

(c) $x$ does not contain both a 0 and a 1.

**Solution.**

$$\text{NOT}[\text{SUBSTRING}(0, x) \text{ AND SUBSTRING}(1, x)]$$

(d) $x$ is the binary representation of $2^k + 1$ for some integer $k \geq 0$.

**Solution.** $(x = 10) \text{ OR } (\exists y\ (x = 1y1 \text{ AND NO-1S}(y)))$

(e) An elegant, slightly trickier way to define NO-1S\((x)\) is:

$$\text{PREFIX}(x, 0x). \tag{*}$$

Explain why (*) is true only when $x$ is a string of 0's.
Solution. Prefixing $x$ with 0 rightshifts all the bits. So the $n$th symbol of $x$ shifts into the $(n + 1)$st symbol of $0x$. Now for $x$ to be a prefix of $0x$, the $n$th symbol of $0x$ must match the $(n + 1)$st symbol of $x$. So if $x$ satisfies (*), the $n$th and $(n + 1)$st symbols of $x$ must match. This holds for all $n \geq 0$ up to the length of $x$, that is, *all* the symbols of $x$ must be the same. In addition, if $x \neq \lambda$, it must start with 0. Therefore, if $x$ satisfies (*), all its symbols must be 0’s.

Note that it’s easy to see, conversely, that if $x = \lambda$ or $x$ is all 0’s, then of course it satisfies (*).

STAFF NOTE: Explain why can’t we define “$x$ is an even-length string of 0’s,” by

$$\text{PREFIX}(x, \text{00}x).$$

Problem 3.

Provide a counter model for the invalid implication. Informally explain why the other one is valid.

1. $\forall x. \exists y. P(x, y) \text{ IMPLIES } \exists y. \forall x. P(x, y)$

2. $\exists y. \forall x. P(x, y) \text{ IMPLIES } \forall x. \exists y. P(x, y)$

Solution. The first implication, $\forall x. \exists y. P(x, y) \text{ IMPLIES } \exists y. \forall x. P(x, y)$, is invalid.

One counter model is the predicate $P(x, y) ::= y < x$ where the domain of discourse is the real numbers, $\mathbb{R}$. For every real number $x$, there exists a real number $y$ which is strictly less than $x$, so the antecedent of the implication is true. But there is no minimum real number, so the consequent is false.

The second implication is valid. Let’s say that “$x$ is good for $y$” when $P(x, y)$ is true. The hypothesis says that there is some element, call it $g$, that is good for everything. The conclusion is that every element has something that is good for it, which of course is true since $g$ will be good for it.

STAFF NOTE: It’s not clear students will be able to articulate the validity explanation. If they get stuck, offer them the “$x$ is good for $y$” phrase as helpful. If it doesn’t help, then explain the answer.