Exercises for Computability and Complexity, Spring 2018, Sheet 3 – Solutions

Please return next Thursday, March 1, in class. As usual you may form teams oft wo.

Note. All problems on this sheet are about showing that certain functions are primitive recursive. This is done by checking that the function in question can be "built up" by using the rules 1. – 5. from Definition 5.1 in the lecture notes. When you use one of these rules for an argument in your proof, please always specify the arities of the functions that you construct.

Problem 1 (easy) Show that the function *plus2*: $\mathbb{N} \to \mathbb{N}$, *plus2*($n$) = n + 2, is primitive recursive.

Solution. From rule 2 we know that the successor function σ is p.r. By rule 4, using $r = 1$ and $m = 1$, and setting $f = g = \sigma$, we get that $h(n) = \sigma(\sigma(n))$ is p.r., but obviously h is just the desired function that adds 2.

Problem 2 (a little easier than medium) Show that the function *minus1*: $\mathbb{N} \to \mathbb{N}$, *minus1*($n$) = max(0, n – 1) is primitive recursive. Hint: there is a very compact way of doing this, exploiting the fact that the primitive recursion scheme condition $h(n + 1, x) = g(n, h(n, x), x)$ has the "minus 1" operation already built in in the first arguments n +1 and n passed to h and g.

Solution. We use the rule 5. of primitive recursion with $r = 0$, put $f = \mathbf{0}$ and $g = p^2_1$. Then h = *minus1*.

Problem 3 (medium) Show that the function *evensquare*: $\mathbb{N} \to \mathbb{N}$, defined by *evensquare*(n) = n if n is uneven, else *evensquare*(n) = n^2, is primitive recursive. You may assume that *square*: $\mathbb{N} \to \mathbb{N}$, *square*($n$) = n^2, is primitive recursive. Hint: you may find it helpful to construct *evensquare* from a number of helper functions which you construct before assembling *evensquare*.

Solution. There are many ways to do this – and I would say this task has a "programming" flavour. Here is one way to do it.

We first procure a p.r. function *flip*: $\mathbb{N} \to \mathbb{N}$ that satisfies *flip*(0) = 1 and *flip*(1) = 0. Using the rule 5., put $r = 0$, $f = \sigma \circ \mathbf{0}$ (i.e., f is the constant 1), $g = p^2_1$. Then put *flip* = h.

Next, create a p.r. function *even*: $\mathbb{N} \to \mathbb{N}$, *even*($n$) = 1 if n is even, else = 0. Again we do this by using the recursion rule 5., putting $r = 0$, $f = \sigma \circ \mathbf{0}$, and g = *flip* $\circ$ p^2_2, which yields h = *even.*

Next, create a conditional function *cond*: $\mathbb{N}^3 \to \mathbb{N}$, which satisfies *cond*(0, n, m) = m and *cond*(1, n, m) = n (remark: *cond* can be seen as implementing an *if-then-else* operator.) Again using the rule 5., put $f = p^2_2$ and $g = p^5_4$ to obtain *cond* = h.

Finally, put *evensquare* = *cond*(*even*(·), *square*(·), *id*(·)) -- which is an obvious shorthand for an application of rule 4.