Math 114C, Winter 2019, Solutions to HW #3
Note: The optional problem was meant to be 2C.8.

---

x2B.3. For the following three recursive programs in $\mathbf{N}_0$,

$$(E_1) \qquad p(x) = S(p(x)),$$

$$(E_2) \qquad p(x) = p(q(x)), \quad q(x) = x,$$

$$(E_3) \qquad p(x,y) = p_1(p(x,y),y), \quad p_1(x,y) = x.$$

(1) Which partial functions satisfy them, as systems of equations?

(2) Which partial functions do they compute, and how do their computations differ?

Solution. (1) The program E_1 is satisfied only by the empty partial function ε, because if $f(x)$ is any solution and $f(n)\downarrow$ for some n, then we have $f(n) = f(n)+1$, which is absurd. The equation of E_2 is just the identity $p(x) = p(x)$, and it is satisfied by every partial function. The same for E_3, which simply expresses $p(x,y) = p(x,y)$ in a different way.

(2) All three programs define the empty partial function, of one variable for the first two and of two variables for the third, but their computations differ significantly. This becomes apparent from their first steps which are as follows (omitting some obvious transitions):

$$p : x \to S(p(x)) : \to S\ p(x) : \to S\ p\ x : \to S\ p : x \to \cdots S\ S\ p : x \to \cdots S\ S\ S\ p : x \to \cdots$$

$$p\ : x \to p\ q\ x : \to p : x \cdots \to p : x \to \cdots$$

$$p : x\ y \to p_1(p(x,y),y) : \to p_1\ p\ x\ y\ y : \to p_1\ p : x\ y\ y \to p_1\ p_1\ p\ x\ y\ y : y \to p_1\ p_1 p : x\ y\ y\ y \cdots$$

The obvious (and significant) differences of the computations of these programs is that the state remains bounded (in length at most 4) in the computations of E_2; it grows indefinitely on its left-hand side in the computations of E_1; and it grows indefinitely on both sides in the computations of E_3.

x2B.5. Construct a recursive program of $\mathbf{N}_0$ which computes the Ackermann function, 1A.6.

Solution. The recursive definition of the Ackermann function is easily expressed in the language R:

$$\begin{aligned} A(n,x) = {} & \mathsf{if}\ (n=0)\ \mathsf{then}\ x = 1 \\ & \mathsf{else}\ \ \mathsf{if}\ (x=0)\ \mathsf{then}\ A(n \dot{-} 1, 1) \\ & \mathsf{else}\ A(n \dot{-} 1, A(n, x \dot{-} 1). \end{aligned}$$

By the Theorem 2C.2, if we consider it as a program, this equation computes a function $\overline{A}(n,x)$ which satisfies it. By (easy, double) induction on n we show that for every n, x, $\overline{A}(n,x)\downarrow$; so this $\overline{A}(n,x)$ is the Ackermann function, by the definition of $A(n,x)$ (as the unique, total function which satisfies the three characteristic equations).

x2B.6. Construct a recursive program which computes the function $p(x, y)$ in Problem x1C.8*.

Solution. We consider the following program, with four equations:

$$\begin{aligned} p(x,y) &= \mathsf{if}\ (x=0)\ \mathsf{then}\ y \\ &\qquad \mathsf{else}\ \ \mathsf{if}\ (y=0)\ \mathsf{then}\ S(p_2(Pd(x))) \\ &\qquad \mathsf{else}\ S(S(p_+(p_3(p(Pd(x),Pd(y))),p(Pd(y),Pd(x))))) \\ p_+(x,y) &= \mathsf{if}\ (x=0)\ \mathsf{then}\ y\ \mathsf{else}\ S(p_+(Pd(x),y)) \\ p_2(x) &= p_+(x,x) \\ p_3(x) &= p_+(p_2(x),x) \end{aligned}$$

The problem requires us to prove that $\overline{p}(x, y)$ is the function $p(x, y)$ of Problem x1C.8*, and this is easier to do by appealing to the Theorem of Computational Soundness 2C.2: from this and familiar arguments, we infer that

$$\overline{p_+}(x,y) = x+y, \quad \overline{p_2}(x) = 2x, \quad \overline{p_3}(x) = 3x$$

and so the equation

$$\begin{aligned} \overline{p}(x,y) &= \mathsf{if}\ (x=0)\ \mathsf{then}\ y \\ &\qquad \mathsf{else}\ \ \mathsf{if}\ (y=0)\ \mathsf{then}\ 2Pd(x)+1 \\ &\qquad \mathsf{else}\ 3\overline{p}(Pd(x),Pd(y)) + \overline{p}(Pd(y),Pd(x)) + 2 \end{aligned}$$

which is easily equivalent to the system of the three equations of Problem x1C.8*, holds.

x2B.7. Construct a recursive program E in the expansion $(\mathbf{N}_0, g, h, \tau)$ of $\mathbf{N}_0$ which computes the partial function f defined from the functions g, h, τ by nested recursion, Problem x1B.19*.

Solution. As for the Ackermann function, we consider the program

$$\begin{aligned} p(n,y) = \mathsf{if}\ (n=0)\ \mathsf{then}\ g(y) \\ \mathsf{else}\ h(p(x \dot{-} 1, \tau(x \dot{-} 1, y \dot{-} 1)), x \dot{-} 1, y \dot{-} 1). \end{aligned}$$

By Theorem 2C.2, the recursive partial function computed by this program satisfies it, therefore it suffices to show that it is *total*, and so it is also the function defined by nested recursion from the given g, h, τ, by definition. To see this, we show by induction on n, that "for every x, $\overline{p}(n, x)\downarrow$"; we will omit this induction, which is easy.

x2B.9. Prove or give a counterexample for each of the following two propositions:

(1) For every partial algebra $\mathbf{M}$ and every $x_0 \in M$, the constant, one-place function $f(x) = x_0$ is $\mathbf{M}$-recursive.

(2) For every number $x_0 \in \mathbb{N}$, the constant, one-place function $f(x) = x_0$ on the natural numbers is recursive.

Solution. (1) This does not hold in the (simplest) partial algebra $(\mathbb{N}, 0, 1,)$ which has no primitives! To prove this by contradiction, we assume that the constant function $f(x) = 2$ is computed by some program E with main symbol

p, and we examine the computation of the value $f(0)$ which is of the form

$$p : 0 \to s_1 \to \cdots \to \ : 2.$$

We observe that 2 does not occur in the first state but occurs in the last one. Let s_{i+1} be the first state in which 2 occurs; but there are no transitions in this algebra which introduce 2 (by simple inspection of the table), and so we arrived at a contradiction.

(2) This is true, because each number n is defined by the closed, pure term $S(S(\cdots n \text{ times}(0))\cdots)$, and so it is computed by the program with the single equation

$$p(x) = S(S(\cdots n \text{ times}(0))\cdots)$$

x2B.10. Prove that for every partial algebra $\mathbf{M}$ and $A \subseteq M$, the set $\overline{A}$ is the smallest $\mathbf{M}$-closed subset of M which contains A, that is: $A \subseteq \overline{A}$, $\overline{A}$ is $\mathbf{M}$-closed, and for every $X \subseteq M$, if $A \subseteq X$ and X is $\mathbf{M}$-closed, then $\overline{A} \subseteq X$.

Solution. Suppose first that $X \subseteq M$, $A \subseteq X$ and X is $\mathbf{M}$-closed, and prove by induction that for each k, $\overline{A}^{(k)} \subseteq X$; this shows that $\overline{A} \subseteq X$. For the converse inclusion, check (easily) that $\overline{A}$ contains A and is $\mathbf{M}$-closed.

x2B.8. Prove that the "equation"

$$f(x) = \begin{cases} 0, & \text{if } g(x)\uparrow, \\ g(x)+1, & \text{otherwise.} \end{cases}$$

"cannot be formalized in R", but first explain what needs to be proved.

Solution. What we have to show is that there exists no term A of R with one function constant g and one number variable v, such that for every partial function g, if $\mathbf{M} = (\mathbb{N}, S, Pd, g)$, then for every number x,

$$\mathbf{val}_{\mathbf{M}}(A\{\mathsf{v} :\equiv x\}) = \begin{cases} 0, & \text{if } g(x)\uparrow, \\ g(x)+1, & \text{otherwise.} \end{cases}$$

Assume that there exists such a term A, let B be the term obtained from the replacement of the constant g by some function variable p, and let E be the program

$$p(\mathsf{v}) = B.$$

It follows that for the partial function $g = \overline{p}$ computed by this program

$$(\mathbb{N}, S, Pd, g) \models p(\mathsf{v}) = B,$$

from which we conclude that

$$g(x) = \begin{cases} 0, & \text{if } g(x)\uparrow, \\ g(x)+1, & \text{otherwise} \end{cases}$$

and it is easy to see that such a partial function does not exist. (If it existed it would be total, by the equation, and so, for every x, we would have $g(x) = g(x)+1$, which is of course absurd.)

Observation: the left-hand side $f(x)$ of the equation we examine here does not play any role! (And yet, this makes the problem difficult.)

`Let me know of errors or better solutions.`