CSE 4111/5111 —Winter 2014

Posted: March 24, 2014

Problem Set No. 2—Solutions

(1) Do Exercises 2.5.0.30 (p.171) and 2.6.0.33 (p.173).

- 2.5.0.30: Consider a set of *mutually exclusive* relations $R_i(\vec{x})$, $i = 1, \ldots, n$, that is, $R_i(\vec{x}) \wedge R_j(\vec{x})$ is false for each $\vec{x}$ as long as $i \neq j$.

  Then we can define a function f *by positive cases* R_i from given functions f_j by the requirement (for all $\vec{x}$) given below:

$$f(\vec{x}) = \begin{cases} f_1(\vec{x}) & \text{if } R_1(\vec{x}) \\ f_2(\vec{x}) & \text{if } R_2(\vec{x}) \\ \ldots & \ldots \\ f_n(\vec{x}) & \text{if } R_n(\vec{x}) \\ \uparrow & \text{otherwise} \end{cases}$$

  Prove that if each f_i is in $\mathcal{P}$ and each of the $R_i(\vec{x})$ is in $\mathcal{P}_*$, then $f \in \mathcal{P}$.
  Hint. Use the graph theorem along with closure properties of $\mathcal{P}_*$ relations to examine $y = f(\vec{x})$.

  Answer. I'll show that $y = f(\vec{x}) \in \mathcal{P}_*$ and will be done by the graph theorem.

  Indeed,

$$y = f(\vec{x}) \equiv y = f_1(\vec{x}) \wedge R_1(\vec{x}) \vee y = f_2(\vec{x}) \wedge R_2(\vec{x}) \vee \ldots \vee y = f_n(\vec{x}) \wedge R_n(\vec{x}) \tag{1}$$

  Since all graphs on the rhs of $\equiv$ are in $\mathcal{P}_*$ by the assumption on the f_i and by the graph theorem, we are done by closure of $\mathcal{P}_*$ under $\wedge, \vee$ and the assumption on the R_i.

  Wait a minute! Aren't we forgetting something like "$y =\uparrow \wedge$ oth" on the rhs? NO! $y =\uparrow$ is meaningless since a variable always holds a number. A number cannot be "undefined". How's this different from $f(\vec{x}) \uparrow$ or $f(\vec{x}) =\uparrow$? Well, a function call $f(\vec{x})$ depending on $\vec{x}$ *can* fail to give a numerical answer (when the program that computes the call never stops with input $\vec{x}$).

To sum up: $y = f(\vec{x})$ says (by virtue of y being a number) "$f(\vec{x}) \downarrow$ *and* equals **the number** y".

Last observation: The "oth" is **NOT** a "positive" case! It is the negation of the disjunction of all the others. Isn't it nice that by virtue of the "$\uparrow$" we do not have to explicitly deal with it!

And, btw, there is **no way** to do this using if-then-else. The R_i's being **NOT** necessarily recursive can lead to an infinite loop during evaluation (the no case). Imagine then that, for input $\vec{a}$, R_2 is true but all the others are false. Given that the if-then-else is **highly sequential** —if $R_1(\vec{a})$ then $f_1(\vec{a})$ else if $R_2(\vec{a})$ then $f_2(\vec{a})$ else if...— we will never answer $f_2(\vec{a})$, as we ought to do, because we are busy looping forever with $R_1(\vec{a})$!

- 2.6.0.33: What is $10 * 5$?

  Answer. First, $lh(10) =$ *the index of first prime that does not divide* 10: That is, 1. Similarly, $lh(5) = 0$, since $p_0 \nmid 5$.

  Now, recall

  $$x * y \stackrel{\text{Def}}{=} x \cdot \prod_{i < lh(y)} p_{i+lh(x)}^{\exp(i,y)} \tag{1}$$

  Thus,

  $$10 * 5 = 10 \cdot \prod_{i < lh(5)} p_{i+lh(10)}^{\exp(i,5)} = 10 \cdot \prod_{i<0} p_{i+lh(10)}^{\exp(i,5)} = 10^{*}$$

(2) From Section 2.12 (p.234 and onwards) do: 23, 24, 30, 31, 35, 42.

- #23: Once again, refer to Subsection 2.2.2 where we constructed the "universal" two-argument function $\lambda yx.f_y(x)$ that enumerates all one-argument primitive recursive functions. Prove
  - For all $\lambda x.h(x) \in \mathcal{PR}$, there is a an m such that $h(x) < f_m(x)$, for all x.
    Answer. Take an m such as $h(x) + 1 = f_m(x)$, all x. Such an m exists because $\lambda x.h(x) + 1$ is in $\mathcal{PR}$ too.
  - Base on the preceding bullet a new proof of the fact that $\lambda yx.f_y(x) \notin \mathcal{PR}$.
    Answer. Otherwise, $h = \lambda x.f_x(x) \in \mathcal{PR}$. By the previous question there is an m such that

    $$f_x(x) < f_m(x)$$

    for all x. Taking $x = m$ we see this cannot be! $\square$

*The empty product equals 1.

- # 24: Prove that it is impossible to form $\mathcal{PR}$ as the closure under *substitution* of some *finite* set of primitive recursive functions.

  Answer. Here's why: Suppose that for some $\mathcal{PR}$ functions $\mathscr{I} = \{f_1, f_2, \ldots, f_r\}$ we have that $\mathcal{PR}$ is equal to the closure of $\mathscr{I}$ under **substitution alone**.

  Then for some m,

  $$f_i(\vec{x}) \leq A_m^{k_i}(\max \vec{x}) \text{ for all } \vec{x} \text{ and all } i = 1, \ldots, r. \tag{1}$$

  Since Ackermann majorisation does not increase the lower index under substitution, we have

  $$g \in \mathrm{Cl}(\mathscr{I}, \text{subst}) \text{ implies } g(\vec{y}) \leq A_m^q(\max \vec{y})^\dagger \text{ for all } \vec{y} \tag{2}$$

  Here's the problem: $\lambda x.A_{m+1}(x) \in \mathcal{PR}$ as we know. If $\mathcal{PR} = \mathrm{Cl}(\mathscr{I}, \text{subst})$, then —by (2)— we must have

  $$A_{m+1}(x) \leq A_m^h(x) \text{ for some } h \text{ and all } x$$

  But this we know is not true ($A_m^h(x) < A_{m+1}(x)$ a.e. is true).

- # 30: Show that the set K_1 defined as $\{[x, y] : \phi_x(y) \downarrow\}$ is semi-computable.

  Proof. $z \in K_1 \equiv Seq(z) \wedge \phi_{(z)_0}((z)_1) \downarrow \equiv Seq(z) \wedge (\exists y) T((z)_0, (z)_1, y)$ and are done by strong projection and closure properties of $\mathcal{P}_*$. □

- # 31: Show that the set K_1 defined above is *not* recursive.
  Hint. **Caution**: Do not confuse coded pair $[x, y]$ with unpacked $\langle x, y\rangle$.
  K_1 is $\{z : \phi_{(z)_0}((z)_1) \downarrow\}$ —a set of numbers, not a set of pairs.

  Proof. If I can "solve" $\phi_{(z)_0}((z)_1) \downarrow$ then I can solve $x \in K$. That is,

  $$K \leq K_1$$

  How? Take $f(x) = [x, x](= 2^{x+1}3^{x+1})$, clearly a $\mathcal{PR}$ function.
  Then

  $$x \in K \equiv f(x) \in K_1$$

  □

- # 35: Prove that neither

  $$f(x) = \begin{cases} 0 & \text{if } x \in K \\ 42 & \text{otherwise} \end{cases}$$

†same m as in (1)!!

nor

$$g(x) = \begin{cases} 0 & \text{if } x \in K \\ x & \text{otherwise} \end{cases}$$

are in $\mathcal{P}$. This justifies our remarks in the text —about definition by positive cases— that the best we can suggest as "output" in the "otherwise" case is $\uparrow$. In general.

Why "in general"?

Because if the positive cases are actually recursive (here it is semi-recursive but not recursive), then so is the "otherwise" and a function call can correspond to this case rather than "$\uparrow$" and still have a computable function overall (definition by recursive cases).

Now to the two examples:

- **The first:** If $f \in \mathcal{P}$ then $f \in \mathcal{R}$ since it is total. But then

$$x \in K \equiv f(x) = 0$$

  making $K \in \mathcal{R}_*$ by a well known lemma. But this is absurd.

- **The second:** Seeing that it is **not** true that $x \in K \equiv g(x) = 0$ because of the x-response in the "otherwise", we need to be more subtle: Note that we have

$$x + 1 \in K \equiv g(x + 1) = 0 \tag{1}$$

  Noting that $\lambda x.g(x + 1) \in \mathcal{R}$ by substitution —if we **assume** $g \in \mathcal{R}$— **we only need to prove that the predicate $x+1 \in K$ is not recursive**, so that (1) can contradict the "red" assumption!

  Well, if we can compute the answer to

$$x + 1 \in K \tag{2}$$

  then

$$\text{we can compute the answer to } x \in K \tag{3}$$

  since $0 \notin K$ (**Why is $0 \notin K$?**)

  Informally, to decide $z \in K$, if $z = 0$ we say "no" and exit. If $z > 0$, then $z = x + 1$ for some x and we "call" the (assumed to exist) program for the problem (2).

  But (3) cannot be!

  Mathematically, if we denote the **assumed recursive** predicate (2) by $Q(x)$ —i.e., to avoid notational confusion we have defined $Q(x) \equiv x + 1 \in K$— then

$$z \in K \equiv z \neq 0 \wedge Q(z \dot{-} 1)$$

Thus if $Q \in \mathcal{R}_*$, then so is K!

- # 42: Prove that the set $E = \{\langle x, y\rangle : \phi_x = \phi_y\}$ is not semi-recursive. *Hint.* Fix ϕ_y to a conveniently simple function.

  Comment: This was done in class!

  Answer. If $E(x, y) \in \mathcal{P}_*$, then so is $E(x, 0)$, that is the set

  $$\{x : \phi_x = \phi_0\}$$

  Given that $\phi_0 = \emptyset$, the above is the set $\{x : \phi_x = \emptyset\}$ which we know from class (recall our reduction arguments!) is not semi-recursive.

(3) From Section 2.12 (p.237 and onwards) do: 46, 47, 54 **without Rice's Theorem!**

None of the 46, 47, 54 speak of recursiveness so Rice's Theorem is inapplicable anyway. Rice's Lemma applies to # 46, but not in any of # 47 or # 54.

- #46: Prove that the set $A = \{x : W_x = \{0, 1, 2\}\}$ is not c.e.

  Here $\mathscr{C}$ is the set of all ϕ_x that have exactly $\{0, 1, 2\}$ as their domain. So, $A = \{x : \phi_x \in \mathscr{C}\}$.

  Using Rice's **Lemma** was not forbidden, so using it we argue like this: First, let

  $$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{if } x = 1 \\ 2 & \text{if } x = 2 \\ \uparrow & \text{if } x \geq 3 \end{cases}$$

  Clearly $f \in \mathcal{P}$ and $\mathrm{dom}(f) = \{0, 1, 2\}$ thus $f \in \mathscr{C}$. Now the function $g = \lambda x.x$ extends f but is not in $\mathscr{C}$ since its domain is $\mathbb{N}$.

  By Rice's lemma, $A \notin \mathcal{P}_*$. □

- #47: Prove that the set $\{x : W_x = \mathbb{N}\}$ is not c.e.

  Answer. This set is the same as $A = \{x : \mathrm{dom}(\phi_x) = \mathbb{N}\}$. Piggy back on the argument in text/class that we did for the non-c.e.-ness of $\{x : \phi_x$ is a constant$\}$. We found there an $h \in \mathcal{PR}$ such that

  $$\phi_{h(x)} = \begin{cases} \lambda y.0 & \text{if } x \in \overline{K} \\ \text{a non constant} & \text{oth} \end{cases}$$

Note that $h(x) \in A$ precisely in the top case. Thus $h(x) \in \overline{K} \equiv x \in A$, that is, $A \leq \overline{K}$ and we are done.

- #54: Prove that $Q = \{x : \phi_x \in \mathcal{PR}\}$ is not c.e.

  Answer. The "ψ" we used in class to show that $A = \{x : \phi_x \text{ is a constant}\}$ is not c.e. once again works as is, since it led to

$$\phi_{h(x)} = \begin{cases} \lambda y.0 & \text{if } \phi_x(x) \uparrow \\ \text{a finite function} & \text{oth} \end{cases}$$

  Note that the top function is in $\mathcal{PR}$ but the bottom is not. Thus, $x \in \overline{K} \equiv h(x) \in Q$, i.e., $\overline{K} \leq_m Q$.