Computability Assignment
Year 2012/13 - Number 11

Please keep this file anonymous: do not write your name inside this file.

More information about assignments at http://disi.unitn.it/~zunino/teaching/computability/assignments

Please do not submit a file containing only the answers; edit this file, instead, filling the answer sections.

Note

Remember that $undefined \not> x$ for any natural x.

1 Question

Consider the set

$$A = \{n \mid \forall x \in \mathbb{N}.\ \phi_n(x) > x\}$$

Prove that $\mathsf{K} \leq_m A$.

1.1 Answer

If we could verify whether a number belongs to A or not, we could create a verifier for K:

$V_{\mathsf{K}} = \lambda p.V_A\#(\lambda x.\phi_p(p) + x + 1)$

We are asking the original verifier to check whether our custom function (that we built using the s-m-n theorem) always outputs a number which is greater than the input value. Because of the way it has been constructed, it always behaves in that way, provided that the program we are asked to verify actually terminates, when its own source code is given as input. Otherwise, the two verifiers will correctly return false.

(RZ: the question was not about proving whether a verifier exists, but to construct a m-reduction. The function $h(p) = \#(\lambda x.\phi_p(p) + x + 1)$ is indeed a reduction. You should prove just that, without trying to construct verifiers when the question does not ask you to do so).

2 Question

Prove that $\bar{\mathsf{K}} \leq_m A$, with the above A.

2.1 Answer

We can just negate the result of the verifier for K from the previous exercise:

$V_{\bar{\mathsf{K}}} = \lambda p.Not\ (V_{\mathsf{K}} p)$

(RZ: no, this does not prove that an m-reduction exists. Also, you never used the set A in your answer!)

3 Question

Consider the set

$$B = \{\mathsf{pair}(n, m) \mid \phi_n(0) = \phi_m(0)\}$$

Prove that $\bar{\mathsf{K}} \leq_m B$.

3.1 Answer

If we had a verifier for B, we could use it to create a verifier for $\bar{\mathsf{K}}$.

$V_{\bar{\mathsf{K}}} = \lambda p.Not\,(V_B \mathsf{pair}\,(\#\,(\lambda z.\phi_p(p))\,,\#\,(\lambda z.\phi_p(p))))$

Since it is not true that $undefined = undefined$ (RZ: no, the convention we have used in this course is that undefined=undefined), we can ask the original verifier to check whether two copies of the same program actually return a value. If this is not the case (because the program loops forever when given as input its own source code - note that the z argument is ignored), we know that the program does not belong to K (otherwise, it will). Then, we negate the answer in order to give an answer about$\bar{\mathsf{K}}$.