Computability Assignment
Year 2012/13 - Number 8

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.

1 Question

Prove that the following set is **not** λ-definable.

$$A = \{\#M \mid \exists n \in \mathbb{N}.\ M\ulcorner n \urcorner =_{\beta\eta} \ulcorner 5 \urcorner\}$$

1.1 Answer

Let's use the Rice's theorem:

A is not empty: for example $K\ulcorner 5\urcorner$is a valid M

A is not equal to N: for example $K\ulcorner 4\urcorner$is not a valid M

A is semantically closed, because if $\#M \in A$ and another program P exists and it is beta-equivalent to M, then it will also return $\ulcorner 5 \urcorner$, so it is also included in the set.

This means that A is not λ-definable.

2 Question

Prove that the following set is semantically closed. Then, prove that it **is** λ-definable.

$$A = \{\#M \mid \forall N \in \Lambda.\ N\, M =_{\beta\eta} \mathbf{I}\}$$

2.1 Answer

The set is empty, because if we select an unsolvable N, the result of its application with M is also unsolvable (hence it is different from $\mathbf{I}$).

Because it is empty, it is also semantically closed (it contains no lambda programs) (RZ: you don't need this), and it isλ-defined by $\lambda x.F$.

Note.

The following exercise is harder. Feel free to skip it.

3 Question

Prove **whether** the following set is λ-definable.

$$A = \{\#M \mid M\ulcorner M\urcorner =_{\beta\eta} M\}$$

(Note: there is at least one simple solution to this. You do not need to try huge formulae for this.)

3.1 Answer

There are no verifiers for the specified set.

Assume that such a verifier exists: it could be used to construct a program like this:

$M = \lambda m.V_A m\,\mathbf{I}\,(E\,m)$

If the verifier (RZ: specify you are considering $V_A\ulcorner M\urcorner$) answers "true", M will return the identity function, proving that the verifier is wrong. (RZ: ok, you need to show that $M \neq I$, but it looks simple)

If the verifier answers "false", M will return the evaluation of the source code of M, proving that the verifier is wrong.