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

...

Consider

$\psi(x, y) = \begin{cases} y+1 & if\ \varphi_x(x)\downarrow \\ \uparrow & o.w. \end{cases}$, by the universal function interpreter, the $\varphi_x(x)\downarrow$ check is recursive (RZ: no, it is only RE). By the ITE lemma, since $y+1$ is recursive (costant 1 is recursive, input y is a number/recursive and addition is recursive and total) and the else branch has $\uparrow$ we conclude that $\psi(x, y)$ is recursive. Then by S-M-N theorem there exists a recursive (total) function g s.t. $\psi(x, y) = \varphi_{g(x)}(y)$.

$x \in K \Longrightarrow \varphi_x(x)\downarrow \Longrightarrow \forall y.\varphi_{g(x)}(y) = y+1 \Longrightarrow \forall y.\varphi_{g(x)}(y) = y+1 > y \Longrightarrow g(x) \in A$

$x \notin K \Longrightarrow \varphi_x(x)\uparrow \Longrightarrow \forall y.\varphi_{g(x)}(y)\uparrow \Longrightarrow \forall y.\varphi_{g(x)}(y)\uparrow \not> y \Longrightarrow g(x) \notin A$

Hence, $K \preceq_m A$ and $\overline{K} \preceq_m \overline{A}$.

...

Example of program implementing $\psi(x, y)$:

```
1: input:x,y
2: begin:
3:         eval1(x,x);
4:         return y + 1;
5: end
```

If $x \notin K$ the program goes stuck (loops forever) in line 3, otherwise ($x \in K$) it computes $y + 1$ and returns it.

2 Question

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

2.1 Answer

...

NOTATION TRIVIA: *here* $\varphi_x(y) \downarrow^k$ stands for "$\varphi_x(y) \downarrow$ in $\leq k$ steps". Similar interpretations for symbols $\not\uparrow$, $\not\downarrow$.

Note that $\overline{K} \preceq_m A \iff K \preceq_m \overline{A}$. Consider $\overline{A} = \{n \mid \exists x \in \mathbb{N}.\ \phi_n(x) \not\succ x\}$

$$\psi(x, y) = \begin{cases} y + 1 & if\ \varphi_x(x) \not\downarrow^y \\ \uparrow & o.w. \end{cases}$$

(RZ: OK. In an exam, invoke the if-then-else lemma (guard is recursive here))

Then by S-M-N theorem there exists a recursive (total) function g s.t. $\psi(x, y) = \varphi_{g(x)}(y)$.

$x \in K \implies \exists k \in \mathbb{N}.\varphi_x(x) \downarrow^k \implies \exists y \in \mathbb{N}.(\varphi_x(x) \downarrow^y \wedge \varphi_{g(x)}(y) \uparrow) \implies \exists y \in \mathbb{N}.\varphi_{g(x)}(y) \uparrow \not\succ y \implies g(x) \in \overline{A}$

$x \notin K \implies \nexists k \in \mathbb{N}.\varphi_x(x) \downarrow^k \implies \forall y \in \mathbb{N}.\varphi_{g(x)}(y) = y + 1 > y \implies \neg\exists y \in \mathbb{N}.\varphi_{g(x)}(y) \not\succ y \implies g(x) \notin \overline{A}$

Since we have found that $\overline{K} \preceq_m A \wedge \overline{K} \preceq_m \overline{A}$ we conclude that A, $\overline{A} \notin \mathcal{RE}$ and productive.

...

```
1: input: x,y
2: begin:
3:         if eq(1,termIn(x,x,y)) then // termIn(x,z,y) returns true if program x r
4:                 while(true);
5:         else
6:                 return y + 1;
7: end
```

If $x \in K$ then for some y the program loops forever on line 4, otherwise ($x \notin K$) it always returns a number greater than the one given in input with

line 6. The program never goes stuck in a loop in line 3, since *termIn* executes at most y steps of computation.

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

...

Again, it may be easier to tackle the problem $\overline{K} \preceq_m B \iff K \preceq_m \overline{B}$, with $\overline{B} = \{pair(n, m) \mid \phi_n(0) \neq \phi_m(0)\}$. (TRIVIA: when constructing $\overline{B}$ note that $\forall y \in \mathbb{N}.\exists! n, m \in \mathbb{N}.pair(n, m) = y$)

Consider

$\psi(x, y) = \begin{cases} 1 & if\ \varphi_x(x) \downarrow \wedge y = 0 \\ \uparrow & o.w. \end{cases}$ and $\gamma(z, t) = \begin{cases} 2 & if\ \varphi_z(z) \downarrow \wedge t = 0 \\ \uparrow & o.w. \end{cases}$

The check $\varphi_x(x) \downarrow$ is recursive by the universal program interpreter, the test $y = 0$ is recursive and since the set of recursive function is closed under $\wedge$ the entire guard is recursive. By the ITE lemma, given the recursive then branch (constant 1) and the $\uparrow$in the else branch, the function $\psi(x, y)$ is recursive. The same considerations hold for $\gamma(z, t)$. Therefore we can apply the S-M-N theorem on both functions and conclude that there exist two (total) recursive functions g, h s.t. $\psi(x, y) = \varphi_{g(x)}(y) \wedge \gamma(z, t) = \varphi_{h(z)}(t)$.

Note that is not restrictive to consider $x = z$, since these are arbitrarily chosen inputs given for our functions ψ, γ.

$x = z \in K \implies \varphi_{g(x)}(y) = \begin{cases} 1 & if\ true \wedge y = 0 \\ \uparrow & o.w. \end{cases} \wedge \varphi_{h(z)}(t) = \begin{cases} 2 & if\ true \wedge t = 0 \\ \uparrow & o.w. \end{cases} \implies$
$\varphi_{g(x)}(0) = 1 \neq 2 = \varphi_{h(z)}(0) \implies pair(g(x), h(z)) \in \overline{B}$

$x = z \notin K \implies \varphi_{g(x)}(y) = \begin{cases} 1 & if\ false \wedge y = 0 \\ \uparrow & o.w. \end{cases} \wedge \varphi_{h(z)}(t) = \begin{cases} 2 & if\ false \wedge t = 0 \\ \uparrow & o.w. \end{cases} \implies$
$\varphi_{g(x)}(0) \uparrow = \varphi_{h(z)}(0) \uparrow \implies pair(g(x), h(z)) \notin \overline{B}$

The *pair* function is recursive and total, the combination of total recursive functions is again a total recursive function and since we are assuming $x = z$ we can say that $(x \in K \iff pair(g(x), h(x)) \in \overline{B}) \implies (K \preceq_m \overline{B} \iff \overline{K} \preceq_m B)$.

...

(Example for ψ, the other is similar)

```
1: input: x,y
2: begin:
3:        eval1(x,x);
4:        if eq(0,y) then
```

```
6:                  return 1;
7:          else
8:                  while(true);
9:  end
```

If $x \notin K$, then the program loops forever in line 3. If $x \in K$ the instruction at line 3 eventually stops and the execution reaches the next line. If $y = 0$ the program returns 1 in line 6, otherwise it goes stuck in a loop at line 8.

V'Ger