Computability Theory **Fall 2020**

Lecture 11: The Fixed-point Theorem

Lecturer: Renjie Yang

Lemma 11.1 *The following statements are equivalent:*

1. *For every partial computable function $g(x, y)$, there is an index e such that for every y,*

$$\varphi_e(y) \simeq g(e, y).$$

2. *For every partial computable function $f(x)$, there is an index e such that for every y,*

$$\varphi_e(y) \simeq \varphi_{f(e)}(y).$$

Alternative formulation:

1. Let T be some Turing machine that computes some function $t : \Sigma^* \times \Sigma^* \to \Sigma^*$. Then there will always exist another Turing machine R that does the same thing as t when t is applied to a description of itself. That is, R computes the function $r : \Sigma^* \to \Sigma^*$ and for every w, $r(w) = t(\langle R \rangle, w)$

2. Let t be any computable function $t : \Sigma^* \to \Sigma^*$. Then there is a Turing machine F such that $t(\langle F \rangle)$ is equivalent to F.

Proof: $1 \Rightarrow 2$: Given f, define g by $g(x, y) \simeq Un(f(x), y)$. Use 1 to get an index e such that for every y, $\varphi_e(y) = Un(f(e), y) = \varphi_{f(e)}(y)$.

$2 \Rightarrow 1$: Given g, use the s-m-n theorem to get f such that for every x and y, $\varphi_{f(x)}(y) \simeq g(x, y)$. Use 2 to get an index e such that $\varphi_e(y) = \varphi_{f(e)}(y) = g(e, y)$. ■

Theorem 11.2 *The two statements in Lemma 11.1 are true.*

Proof: It suffices to prove statement 1. Define $s(x, y) \simeq Un(x, x, y)$. By the s-m-n theorem, we can find aprimitive recursive function $diag$ satisfying

$$\varphi_{diag(x)}(y) \simeq s(x, y)$$

Now define the function l by

$$l(x, y) \simeq g(diag(x), y)$$

and let $\ulcorner l \urcorner$ be an index for l. Let $e = diag(\ulcorner l \urcorner)$, then for every y, we have

$$\begin{aligned} \varphi_e(y) &\simeq \varphi_{diag(\ulcorner l \urcorner)}(y) \\ &\simeq \varphi_{\ulcorner l \urcorner}(\ulcorner l \urcorner, y) \\ &\simeq l(\ulcorner l \urcorner, y) \\ &\simeq g(diag(\ulcorner l \urcorner), y) \\ &\simeq g(e, y). \end{aligned} \tag{11.1}$$

■

Example A program that print itself:
x ← ‘print ‘x ← ‘’ print x print ‘” print x’
print ‘x ←‘’
print x
print ‘”
print x

Algorithm 1: An algorithm that prints itself