thy.1 Equivalent Defininitions of Computably Enumerable Sets

cmp:thy:eqc:sec The following gives a number of important equivalent statements of what it means to be computably enumerable.

cmp:thy:eqc:thm:ce-equiv **Theorem thy.1.** *Let S be a set of natural numbers. Then the following are equivalent:*

1. *S is computably enumerable.*
2. *S is the range of a* partial *computable function.*
3. *S is empty or the range of a primitive recursive function.*
4. *S is the* domain *of a partial computable function.*

The first three clauses say that we can equivalently take any non-empty computably enumerable set to be enumerated by either a computable function, a partial computable function, or a primitive recursive function. The fourth clause tells us that if S is computably enumerable, then for some index e, explanation

$$S = \{x : \varphi_e(x) \downarrow\}.$$

In other words, S is the set of inputs on for which the computation of φ_e halts. For that reason, computably enumerable sets are sometimes called *semi-decidable*: if a number is in the set, you eventually get a "yes," but if it isn't, you never get a "no"!

Proof. Since every primitive recursive function is computable and every computable function is partial computable, (3) implies (1) and (1) implies (2). (Note that if S is empty, S is the range of the partial computable function that is nowhere defined.) If we show that (2) implies (3), we will have shown the first three clauses equivalent.

So, suppose S is the range of the partial computable function φ_e. If S is empty, we are done. Otherwise, let a be any element of S. By Kleene's normal form theorem, we can write

$$\varphi_e(x) = U(\mu s\, T(e, x, s)).$$

In particular, $\varphi_e(x) \downarrow$ and $= y$ if and only if there is an s such that $T(e, x, s)$ and $U(s) = y$. Define $f(z)$ by

$$f(z) = \begin{cases} U((z)_1) & \text{if } T(e, (z)_0, (z)_1) \\ a & \text{otherwise.} \end{cases}$$

Then f is primitive recursive, because T and U are. Expressed in terms of Turing machines, if z codes a pair $\langle (z)_0, (z)_1 \rangle$ such that $(z)_1$ is a halting computation of machine e on input $(z)_0$, then f returns the output of the computation; otherwise, it returns a.We need to show that S is the range of f, i.e., for any natural number y, $y \in S$ if and only if it is in the range of f. In the forwards direction, suppose $y \in S$. Then y is in the range of φ_e, so for some x and s, $T(e,x,s)$ and $U(s) = y$; but then $y = f(\langle x, s\rangle)$. Conversely, suppose y is in the range of f. Then either $y = a$, or for some z, $T(e,(z)_0,(z)_1)$ and $U((z)_1) = y$. Since, in the latter case, $\varphi_e(x) \downarrow= y$, either way, y is in S.

(The notation $\varphi_e(x) \downarrow= y$ means "$\varphi_e(x)$ is defined and equal to y." We could just as well use $\varphi_e(x) = y$, but the extra arrow is sometimes helpful in reminding us that we are dealing with a partial function.)

To finish up the proof of Theorem thy.1, it suffices to show that (1) and (4) are equivalent. First, let us show that (1) implies (4). Suppose S is the range of a computable function f, i.e.,

$$S = \{y : \text{for some } x, f(x) = y\}.$$

Let

$$g(y) = \mu x\; f(x) = y.$$

Then g is a partial computable function, and $g(y)$ is defined if and only if for some x, $f(x) = y$. In other words, the domain of g is the range of f. Expressed in terms of Turing machines: given a Turing machine F that enumerates the elements of S, let G be the Turing machine that semi-decides S by searching through the outputs of F to see if a given element is in the set.

Finally, to show (4) implies (1), suppose that S is the domain of the partial computable function φ_e, i.e.,

$$S = \{x : \varphi_e(x) \downarrow\}.$$

If S is empty, we are done; otherwise, let a be any element of S. Define f by

$$f(z) = \begin{cases} (z)_0 & \text{if } T(e,(z)_0,(z)_1) \\ a & \text{otherwise.} \end{cases}$$

Then, as above, a number x is in the range of f if and only if $\varphi_e(x) \downarrow$, i.e., if and only if $x \in S$. Expressed in terms of Turing machines: given a machine M_e that semi-decides S, enumerate the elements of S by running through all possible Turing machine computations, and returning the inputs that correspond to halting computations. □

The fourth clause of Theorem thy.1 provides us with a convenient way of enumerating the computably enumerable sets: for each e, let W_e denote the domain of φ_e. Then if A is any computably enumerable set, $A = W_e$, for some e.

The following provides yet another characterization of the computably enumerable sets.

Theorem thy.2. *A set S is computably enumerable if and only if there is a computable relation $R(x,y)$ such that* cmp:thy:eqc:thm:exists-char

$$S = \{x : \exists y\, R(x,y)\}.$$

Proof. In the forward direction, suppose S is computably enumerable. Then for some e, $S = W_e$. For this value of e we can write S as

$$S = \{x : \exists y\, T(e, x, y)\}.$$

In the reverse direction, suppose $S = \{x : \exists y\, R(x, y)\}$. Define f by

$$f(x) \simeq \mu y\ AtomRx, y.$$

Then f is partial computable, and S is the domain of f. □

Photo Credits

Bibliography