THEORY OF COMPUTATION
Recursively Enumerable Sets - 13 part 4

Prof. Dan A. Simovici

UMB

$\mathcal{S}$ programs can be encoded as numbers, hence every one-argument function computed by an $\mathcal{S}$ program appears in the list

$$\psi^{(1)}_{\mathcal{P}_0}, \psi^{(1)}_{\mathcal{P}_1}, \ldots$$

The proof of the fact that $\mathrm{HALT}(x, y)$ is not computable is actually a proof by diagonalization. Recall that

$$\mathrm{HALT}(x, y) \Leftrightarrow \psi^{(1)}_{\mathcal{P}}(x)\downarrow, \text{ where } \#(\mathcal{P}) = y$$

is not computable.

Suppose that HALT(x, y) were computable by a program $\mathcal{P}$ with $\#(\mathcal{P}) = y$, that is,

$$\text{HALT}(x, y) = \textit{TRUE} \text{ if } \psi_{\mathcal{P}}^{(1)}(x)\downarrow,$$

and

$$\text{HALT}(x, y) = \textit{FALSE} \text{ if } \psi_{\mathcal{P}}^{(1)}(x)\uparrow.$$

The set of $\mathcal{S}$ programs is countable, so we can arrange it in a list:

$$\mathcal{P}_0, \mathcal{P}_1, \ldots$$

Consider a list of all one-variable functions computed by these programs $\psi^{(1)}_{\mathcal{P}_0}, \psi^{(1)}_{\mathcal{P}_1}, \ldots$ and construct the array:

$$\begin{array}{cccc}
\boxed{\psi^{(1)}_{\mathcal{P}_0}(0)} & \psi^{(1)}_{\mathcal{P}_0}(1) & \psi^{(1)}_{\mathcal{P}_0}(2) & \cdots \\
\psi^{(1)}_{\mathcal{P}_1}(0) & \boxed{\psi^{(1)}_{\mathcal{P}_1}(1)} & \psi^{(1)}_{\mathcal{P}_1}(2) & \cdots \\
\psi^{(1)}_{\mathcal{P}_2}(0) & \psi^{(1)}_{\mathcal{P}_2}(1) & \boxed{\psi^{(1)}_{\mathcal{P}_2}(2)} & \cdots \\
\vdots & \vdots & \vdots &
\end{array}$$

Each row represents one computable function.
Recall that we considered the program $\mathcal{P}$:

$$[A] \quad \text{IF HALT}(X, X) \text{ GOTO } A$$

that computed $\psi_{\mathcal{P}}^{(1)}(x)$. We claim that there is no row in the previous table that corresponds to $\mathcal{P}$. Note that

$$\psi_{\mathcal{P}}^{(1)}(x) \downarrow \text{ if and only if } \psi_{\mathcal{P}_x}^{(1)}(x) \uparrow .$$

Thus, the row that would correspond to $\psi_{\mathcal{P}}^{(1)}$ will differ from the row that corresponds to $\mathcal{P}_x$ in the diagonal entry. This makes impossible for $\mathcal{P}$ to correspond to a row in this table!

Let $\text{TOT} = \{z \in \mathbb{N} \mid (\forall x)\Phi(x, z)\downarrow\}$. In other words, TOT is the set of program codes z that compute total functions.

Theorem

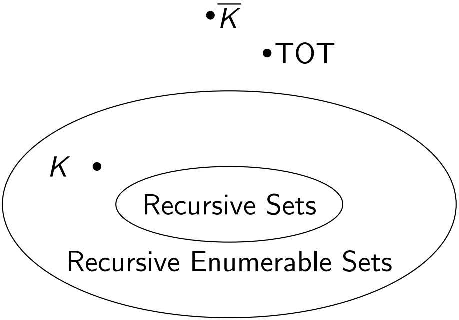
The set TOT is not r.e.

Proof.

Suppose TOT were r.e. Since $\text{TOT} \neq \emptyset$, there exists a computable function g such that $\text{TOT} = \{g(0), g(1), g(2), \ldots\}$. Define $h(x) = \Phi(x, g(x)) + 1$.
Since $g(x)$ is the number of a program that computes a total function, $\Phi(x, g(x))\downarrow$ for all x. In particular, $h(x)\downarrow$ for all x.
Suppose that h is computed by $\mathcal{P}$ with $p = \#(\mathcal{P})$. Then, $p \in \text{TOT}$, so $p = g(i)$ for some i. Then,

$$h(i) = \Phi(i, g(i)) + 1 = \Phi(i, p) + 1 = h(i) + 1,$$

which is a contradiction. □


$\bullet\overline{K}$
•TOT
K •
Recursive Sets
Recursive Enumerable Sets

Definition

Let A, B be sets. The set A is many-one reducible to B, written $A \leqslant_m B$ if there exists a computable function f such that

$$A = \{x \in \mathbb{N} \mid f(x) \in B\}.$$

If $A \leqslant_m B$, testing membership in A is no harder than testing membership in B because to test if $x \in A$, compute $f(x)$ and test whether $f(x) \in B$.

Theorem

Suppose $A \leqslant_m B$.
If B is recursive, then A is recursive.
If B is r.e., then A is r.e.

Proof.

Part 1: Since $A \leqslant_m B$ there exists f such that $A = \{x \mid f(x) \in B\}$. If P_B is the characteristic predicate of B, then $A = \{x \mid P_B(f(x)) = 1\}$, which show that $P_A(x) = P_B(f(x))$. Thus, if B is recursive, then P_A is computable so A is recursive. □

Proof.

Part 2: Suppose B is r.e. Then,

$$B = \{x \in \mathbb{N} \mid g(x) \downarrow\}$$

for some partially computable function g. Therefore,

$$A = \{x \in \mathbb{N} \mid g(f(x)) \downarrow\}.$$

Since $g(f(x))$ is partially computable, A is r.e. □

Example

The set

$$K_0 = \{x \in \mathbb{N} \mid \Phi_{r(x)}(\ell(x)) \downarrow\} = \{\langle x, y \rangle \mid \Phi_y(x) \downarrow\}$$

is r.e. but it is not recursive. K_0 is clearly r.e. We will prove that $K \leqslant_m K_0$ which should imply that K_0 is not recursive.

Example cont'd

Example

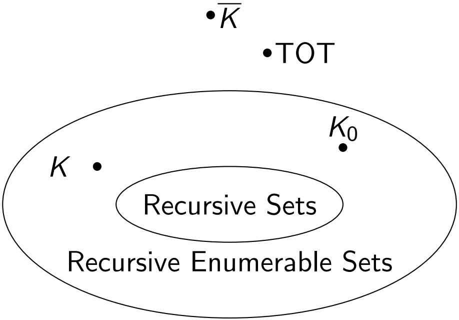
Facts:

- $x \in K$ if and only if $\langle x, x\rangle \in K_0$.
- $f(x) = \langle x, x\rangle$ is computable.

Claim: if A is r.e. then $A \leqslant_m K_0$:

$$\begin{aligned} A &= \{x \in \mathbb{N} \mid g(x)\downarrow\} \text{ for some partially computable } g \\ &= \{x \in \mathbb{N} \mid \Phi(x, z_0)\downarrow\} \text{ for some } z_0 \\ &= \{x \in \mathbb{N} \mid \langle x, z_0\rangle \in K_0\}. \end{aligned}$$

In particular, $K \leqslant_m K_0$.


$\overline{K}$
TOT
K_0
K
Recursive Sets
Recursive Enumerable Sets

Definition

A set A is *m-complete* if

1. A is r.e., and
2. for every r.e. set B we have $B \leqslant_m A$.

Example

The set K_0 is m-complete.

Theorem

If $A \leqslant_m B$ and $B \leqslant_m C$, then $A \leqslant_m C$.

Proof.

Let

$$\begin{aligned} A &= \{x \in \mathbb{N} \mid f(x) \in B\}, \text{ and} \\ B &= \{x \in \mathbb{N} \mid g(x) \in C\}. \end{aligned}$$

Then, $A = \{x \in \mathbb{N} \mid g(f(x)) \in C\}$. □

Corollary

If A is m-complete, B is r.e. and $A \leqslant_m B$, then B is m-complete.

Proof.

If C is r.e., then $C \leqslant_m A$ and $A \leqslant_m B$, so $C \leqslant_m B$, so B is m-complete. □

Note: testing membership in an m-complete set is at least as difficult as testing membership in any r.e. set.

Theorem

The set K is m-complete.

Proof.

We will show that $K_0 \leqslant_m K$. To this end, we start with a pair $\langle n, q \rangle$ and transform it into a number $f(\langle n, q \rangle)$ of a program such that

$$\Phi_q(n)\downarrow \text{ if and only if } \Phi_{f(\langle n,q \rangle)}(f(\langle n, q \rangle))\downarrow .$$

In other words, $\langle n, q \rangle \in K_0$ if and only if $f(\langle n, q \rangle) \in K$. □

Proof cont'd

Proof:
Let $\mathcal{P}$ be the $\mathcal{S}$ program $Y \leftarrow \Phi^{(1)}(\ell(X_2), r(X_2))$ and let $p = \#(\mathcal{P})$.
Then, $\psi_{\mathcal{P}}(x_1, x_2) = \Phi^{(1)}(\ell(x_2), r(x_2))$, and

$$\psi_{\mathcal{P}}(x_1, x_2) = \Phi^{(2)}(x_1, x_2, p) = \Phi^{(1)}(x_1, S_1^1(x_2, p))$$

for all values of x_1. This holds for all values of x_1, so in particular,

$$\Phi^{(1)}(n, q) = \Phi^{(1)}_{S_1^1(\langle n,q \rangle, p)}(S_1^1(\langle n, q \rangle, p)).$$

Therefore, $\Phi^{(1)}(n, q) \downarrow$ if and only if $\Phi^{(1)}_{S^1_1(\langle n,q \rangle, p)}(S^1_1(\langle n, q \rangle, p)) \downarrow$, so

$$\langle n, q \rangle \in K_0 \text{ if and only if } S^1_1(\langle n, q \rangle, p) \in K.$$

With p held constant, $S^1_1(x, p)$ is a computable unary function. Thus, $K_0 \leqslant_m K$.

Definition

$A \equiv_m B$ means that $A \leqslant_m B$ and $B \leqslant_m A$.

$A \equiv_m B$ means that testing membership in A has the same difficulty as testing membership in B.
We proved that both K and K_0 are m-complete and that $K \equiv_m K_0$.

Definition

Let

$$\text{EMPTY} = \{x \in \mathbb{N} \mid W_x = \emptyset\}.$$

Theorem

The set EMPTY is not r.e.

Proof.

We show that $\overline{K} \leqslant_m$ EMPTY. Since $\overline{K}$ is not r.e, it will follow that EMPTY is not r.e.

Let $\mathcal{P}$ be the $\mathcal{S}$ program $Y \leftarrow \Phi(X_2, X_2)$ with $p = \#(\mathcal{P})$. $\mathcal{P}$ does not use X_1, so

$$\psi_{\mathcal{P}}^{(2)}(x, z)\downarrow \text{ if and only if } \Phi(z, z)\downarrow.$$

By the smn theorem

$$\psi_{\mathcal{P}}^{(2)}(x, z) = \Phi^{(2)}(z, z, p) = \Phi^{(1)}(x_1, S_1^1(x_2, p)).$$

□

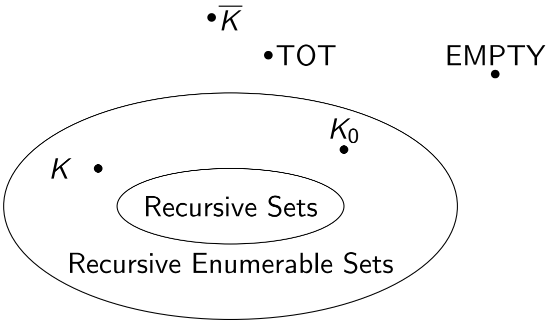
Proof cont'd

Proof.

For any z we have

$$\begin{aligned} z \in \overline{K} &\Leftrightarrow \Phi(z,z)\uparrow \\ &\Leftrightarrow \psi_{\mathcal{P}}^{(2)}(x,z)\uparrow \text{ for all } x \\ &\Leftrightarrow \Phi^{(1)}(x, S_1^1(z,p))\uparrow \text{ for all } x \\ &\Leftrightarrow W_{S_1^1(z,p)} = \emptyset \\ &\Leftrightarrow S_1^1(z,p) \in \text{EMPTY}. \end{aligned}$$

Since $f(z) = S_1^1(z,p)$ is computable, we have $\overline{K} \leqslant_m$ EMPTY. □


$\overline{K}$
TOT
EMPTY
K_0
K
Recursive Sets
Recursive Enumerable Sets