THEORY OF COMPUTATION
Recursively Enumerable Sets - 10 part 1

Prof. Dan A. Simovici

UMB

Predicates can be used to define sets.

Definition

If $P(x_1, \ldots, x_n)$ is a predicate, the set B_P defined by P is:

$$B_P = \{(x_1, \ldots, x_n) \mid P(x_1, \ldots, x_n) = \text{TRUE}\}.$$

P is the characteristic predicate of the set B_P.
The set B_P is defined as computable or recursive if its characteristic predicate is computable.
B_P is primitive recursive if P is a primitive recursive predicate.

In other words, B_P is recursive if we can give a yes/no answer to the question "$x \in B_P$". This follows from the fact that P is computable.

Example

The set

$$B = \{(x, y) \mid \text{ the program } \mathcal{P} \text{ with } \#(\mathcal{P}) = y \text{ halts on } x\}$$

has HALT(X, Y) as its characteristic predicate. Since *HALT* is not computable, the set B is not recursive.

Definition

A set B belongs to a class of functions if its characteristic predicate belongs to that set.

Theorem

Let $\mathcal{C}$ be a PRC class. If B, C belong to $\mathcal{C}$, then so do the sets $B \cup C, B \cap C$ and $\overline{B}$.

Proof.

If P_B, P_C are the characteristic predicates of B and C, respectively, and $P_B, P_C \in \mathcal{C}$, then the characteristic predicates of $B \cup C, B \cap C$ and $\overline{B}$ are $P_B \vee P_C$, $P_B \& P_C$, and $\sim P_B$, respectively, and we saw that they belong to $\mathcal{C}$. □

Theorem

Let $\mathcal{C}$ be a PRC class, and let $B \subseteq \mathbb{N}^m$, where $m \geqslant 1$. Then $B \in \mathcal{C}$ if and only it the set of numbers

$$B' = \{[x_1, \ldots, x_m] \mid (x_1, \ldots, x_m) \in B\}$$

belongs to $\mathcal{C}$.

Proof.

If $P_B(x_1, \ldots, x_m)$ is the characteristic function of B, then

$$P_{B'}(x) \Leftrightarrow P_B((x)_1, \ldots, (x)_m) \& \mathrm{Lt}(x) = m,$$

and $P_{B'}$ clearly belongs to $\mathcal{C}$ if $P_B \in \mathcal{C}$.
On the other hand, $P_B(x_1, \ldots, x_m) \Leftrightarrow P_{B'}([x_1, \ldots, x_n])$, hence $P_{B'} \in \mathcal{C}$ implies $P_B \in \mathcal{C}$. □

Definition

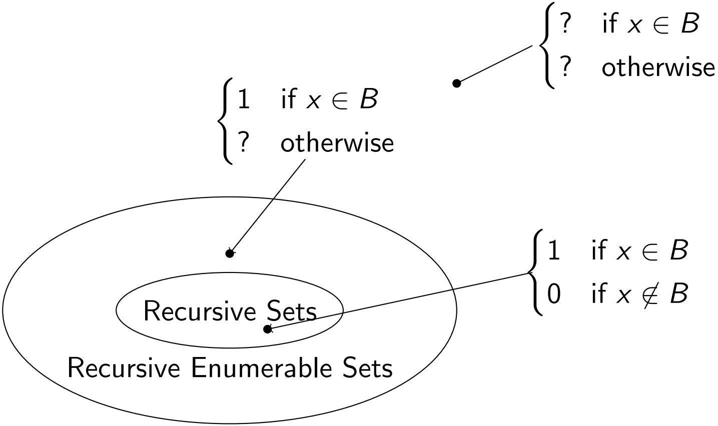
The set $B \subseteq \mathbb{N}$ is recursively enumerable if there is a partially computable function $g(x)$ such that

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

The term recursively enumerable is abbreviated as r.e.

A set is recursively enumerable when it the domain of a partially computable function. Equivalently, B is r.e. if it is just the set of inputs on which some program $\mathcal{P}$ halts.

- If $\mathcal{P}$ is an algorithm for testing the membership in B, $\mathcal{P}$ will provide an *yes* answer for any x in B.
- If $x \notin B$ the algorithm $\mathcal{P}$ will never terminate. This is why $\mathcal{P}$ is also called a semidecision procedure for B.


$\begin{cases} ? & \text{if } x \in B \\ ? & \text{otherwise} \end{cases}$
$\begin{cases} 1 & \text{if } x \in B \\ ? & \text{otherwise} \end{cases}$
$\begin{cases} 1 & \text{if } x \in B \\ 0 & \text{if } x \notin B \end{cases}$
Recursive Sets
Recursive Enumerable Sets

Theorem

If B is a recursive set, then B is r.e.

Proof.

Since B is recursive, the predicate $x \in B$ is computable, so we can write the program $\mathcal{P}$:

$$[A] \quad \text{IF } \sim (X \in B) \text{ GOTO } A$$

If $h(x)$ is computed by this program then
$B = \{x \in \mathbb{N} \mid h(x)\downarrow\}$. □

Theorem

The set B is recursive if and only if both B and $\overline{B}$ are both r.e.

Proof.

If B is recursive, then so is $\overline{B}$, hence both B and $\overline{B}$ are r.e. Conversely, suppose that B and $\overline{B}$ are both r.e., that is

$$\begin{aligned} B &= \{x \in \mathbb{N} \mid g(x)\downarrow\}, \\ \overline{B} &= \{x \in \mathbb{N} \mid h(x)\downarrow\}, \end{aligned}$$

where g and h are both partially computable. □

Proof cont'd

Proof.

Let g be the function computed by program $\mathcal{P}$ and h be the function computed by program $\mathcal{Q}$, where $\#(\mathcal{P}) = p$ and $\#(\mathcal{Q}) = q$. The next program computes the characteristic function of B:

$$
\begin{array}{ll}
[A] & \text{IF STP}^{(1)}(X, p, T) \text{ GOTO } C \\
 & \text{IF STP}^{(1)}(X, q, T) \text{ GOTO } E \\
 & T \leftarrow T + 1 \\
 & \text{GOTO } A \\
[C] & Y \leftarrow 1
\end{array}
$$

□

The technique used in the previous proof is known as dovetailing. It combines the algorithms for computing g and h by running the two algorithms for longer and longer times until one of them terminates.

Theorem

If B and C are r.e. sets, then so are $B \cup C$ and $B \cap C$.

Proof.

Let

$$B = \{x \in \mathbb{N} \mid g(x)\downarrow\} \text{ and } C = \{x \in \mathbb{N} \mid h(x)\downarrow\},$$

where g and h are partially computable. Let f be computed by

$$Y \leftarrow g(X)$$
$$Y \leftarrow h(X)$$

Note that $f(x)\downarrow$ if and only if $g(x)\downarrow$ and $h(x)\downarrow$. Hence $B \cap C = \{x \in \mathbb{N} \mid f(x)\downarrow\}$, so $B \cap C$ is r.e. □

Proof cont'd

Proof.

For $B \cup C$ we use dovetailing again. Let g be the function computed by program $\mathcal{P}$ and h be the function computed by program $\mathcal{Q}$, where $\#(\mathcal{P}) = p$ and $\#(\mathcal{Q}) = q$. Let $k(x)$ be computed by

$$
\begin{array}{ll}
[A] & \text{IF STP}^{(1)}(X, p, T) \text{ GOTO } E \\
 & \text{IF STP}^{(1)}(X, q, T) \text{ GOTO } E \\
 & T \leftarrow T + 1 \\
 & \text{GOTO } A
\end{array}
$$

Thus, $k(x)\downarrow$ just when either $g(x)\downarrow$ or $h(x)\downarrow$, that is $B \cup C = \{x \in \mathbb{N} \mid k(x)\downarrow\}$. □

If $\Phi(x, n)$ is the universal function, n is the program code and x is the input. Alternatively, we use the notation

$$\Phi_n(x)$$

for $\Phi(x, n)$.
The definition domain of $\Phi_n(x)$ is the set denoted as W_n.
Equivalently,

$$W_n = \{x \in \mathbb{N} \mid \Phi(x, n) \downarrow\}.$$

or

$$W_n = \{x \in \mathbb{N} \mid \Phi_n(x) \downarrow\}.$$

Theorem

Enumeration Theorem: *A set B is r.e. if and only if there is an n for which $B = W_n$.*

Proof.

This follows immediately from the definition of $\Phi(x, n)$. □

The theorem gets its name from the fact that

$$W_0, W_1, \ldots, W_n, \ldots$$

is an enumeration of all r.e. sets.