

### Recursively Enumerable Sets

As long as the Gödel numbering functions  $[x_1, \dots, x_n]$  and  $(x)_i$  are available, we only need to consider subsets of  $N$  instead of subsets of  $N^m$ .

Then we have:

**Theorem 4.2:** Let  $C$  be a PRC class, and let  $B$  be a subset of  $N^m$ ,  $m \geq 1$ .

Then  $B$  belongs to  $C$  if and only if

$B' = \{[x_1, \dots, x_m] \in N \mid (x_1, \dots, x_m) \in B\}$  belongs to  $C$ .

October 29, 2009

Theory of Computation  
Lecture 13: A Universal Program V

1

### Recursively Enumerable Sets

**Proof:** If  $P_B(x_1, \dots, x_m)$  is the characteristic function of  $B$ , then

$P_{B'}(x) \Leftrightarrow P_B((x)_1, \dots, (x)_m) \ \& \ Lt(x) \leq m \ \& \ x > 0$

is the characteristic function of  $B'$ , and  $P_{B'}$  clearly belongs to  $C$  if  $P_B$  belongs to  $C$ .

On the other hand, if  $P_B(x)$  is the characteristic function of  $B$ , then

$P_B(x_1, \dots, x_m) \Leftrightarrow P_{B'}([x_1, \dots, x_m])$

is the characteristic function of  $B$ , and  $P_B$  clearly belongs to  $C$  if  $P_{B'}$  belongs to  $C$ .

For example,  $\{[x, y] \in N \mid \text{HALT}(x, y)\}$  is not a computable set.

October 29, 2009

Theory of Computation  
Lecture 13: A Universal Program V

2

### Recursively Enumerable Sets

**Definition:** The set  $B \subseteq N$  is called **recursively enumerable** if there is a partially computable function  $g(x)$  such that

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

The term recursively enumerable is abbreviated **r.e.**

A set is r.e. just when it is the domain of a partially computable function.

If  $\mathcal{P}$  is a program that computes the function  $g$  (see above), then  $B$  is simply the set of all inputs to  $\mathcal{P}$  for which  $\mathcal{P}$  eventually halts.

October 29, 2009

Theory of Computation  
Lecture 13: A Universal Program V

3

### Recursively Enumerable Sets

If we think of  $\mathcal{P}$  as providing an algorithm for testing for membership in  $B$ , we see that

- if a number belongs to  $B$ , the algorithm will provide a positive answer,
- if a number does not belong to  $B$ , the algorithm will never terminate.

Such algorithms are called **semi-decision procedures**.

They can be considered an "approximation" to solving the problem of testing membership in  $B$ .

October 29, 2009

Theory of Computation  
Lecture 13: A Universal Program V

4

### Recursively Enumerable Sets

**Theorem 4.3:** If  $B$  is a recursive set, then  $B$  is r.e.

**Proof:** Consider the following program  $\mathcal{P}$ :

[A] IF  $\sim(X \in B)$  GOTO A

Since  $B$  is recursive, the predicate  $X \in B$  is computable and  $\mathcal{P}$  can be expanded to a program of  $\mathcal{L}$ .

Let  $\mathcal{P}$  compute the function  $h(x)$ . Then, clearly,

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

October 29, 2009

Theory of Computation  
Lecture 13: A Universal Program V

5

### Recursively Enumerable Sets

If  $B$  and  $\neg B$  are both r.e., then we can devise **two algorithms**:

- one algorithm that terminates if a given input is in  $B$ , and
- another algorithm that terminates if a given input is not in  $B$ .

Can we find a way to **combine** these two algorithms to obtain a single algorithm that **always terminates** and tells us whether a given input is in  $B$ ?

The trick is to let the two algorithms run for more and more steps until one of them terminates (**dovetailing**).

October 29, 2009

Theory of Computation  
Lecture 13: A Universal Program V

6

### Recursively Enumerable Sets

**Theorem 4.4:** The set  $B$  is recursive if and only if  $B$  and  $\neg B$  are both r.e.

**Proof:** If  $B$  is recursive, then by Theorem 4.1 so is  $\neg B$ , and hence by Theorem 4.3, they are both r.e.

Conversely, if  $B$  and  $\neg B$  are both r.e., we may write

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

$$\neg B = \{x \in \mathbb{N} \mid h(x) \downarrow\},$$

where  $g$  and  $h$  are both partially computable.

October 29, 2009

Theory of Computation  
Lecture 13: A Universal Program V

7

### Recursively Enumerable Sets

Now let  $g$  be computed by program  $\mathcal{P}$  and  $h$  be computed by program  $\mathcal{Q}$ , and let  $p = \#(\mathcal{P})$  and  $q = \#(\mathcal{Q})$ .

Then the following program computes  $B$ :

```
[A]  IF STP(1)(X, p, T) GOTO C
      IF STP(1)(X, q, T) GOTO E
      T ← T+1
      GOTO A
[C]  Y ← 1
```

October 29, 2009

Theory of Computation  
Lecture 13: A Universal Program V

8