Math 260A — Mathematical Logic — Scribe Notes
UCSD — Spring Quarter 2012
Instructor: Sam Buss

Notes by: Thomas Barrett
April 9, 2012

1 Theorems on Decidability, Semi-Decidability, and Enumerability

Recall that last time we were talking about recursive, semi-decidable, and recursively enumerable relations/functions. Here we prove a number of theorems.

Theorem 1. *If R is recursive (i.e. decidable or computable), then R is recursively enumerable (i.e. computably enumerable, or equivalently, semi-decidable).*

Proof. If a Turing machine M decides R, then M semi-decides R. And since R is semi-decidable if and only if R is recursively enumerable (by a theorem last time), we conclude that R is recursively enumerable, as desired. □

We now need a couple of definitions:

Definition 1. Let $R \subset \Sigma^*$, then the *complement of* R, denoted $\bar{R}$, is defined by $\bar{R} := \Sigma^* - R$.

Definition 2. R is *corecursively enumerable* if and only if $\bar{R}$ is recursively enumerable.

The following theorem shows the relationship between recursive/corecursive enumerability and recursivity.

Theorem 2. *R is recursive if and only if R is recursively enumerable and corecursively enumerable.*

Proof. ($\Rightarrow$) Assume R is recursive. By Theorem 1, R is recursively enumerable. It is clear that if R is recursive, then $\bar{R}$ is recursive as well. For since R is recursive, there is a Turing machine M which decides R. Modify M by exchanging the states q_Y and q_N. Then this new Turing machine decides $\bar{R}$.

So R is recursive implies that $\bar{R}$ is recursive, which implies that $\bar{R}$ is recursively enumerable (by Theorem 1 again), which implies that R is corecursively enumerable.

($\Leftarrow$) Assume M_1 semi-decides R and M_2 semi-decides $\bar{R}$. We give the following algorithm to decide R.

Input $w \in \Sigma^*$. For each $i = 1, 2, 3, \ldots$ we do the following:

1. Run $M_1(w)$ for i steps, if it enters its q_Y state, then enter the 'accept' state (q_Y for our new machine).
2. Run $M_2(w)$ for i steps, if it enters its q_Y state, then enter the 'reject' state (q_N for our new machine).

For a given w, either 1 or 2 will eventually happen since M_1 and M_2 semi-decide R and $\bar{R}$, respectively. So our new machine will always halt with the correct answer, and hence it decides R. □

We will prove most of the following theorem, but part of it will be left for HW.

Theorem 3. *The following are equivalent:*

1. *R is semi-decidable.*
2. *R is recursively enumerable.*
3. *R is the range of a partial recursive function.*
4. *R is the domain of a partial recursive function.*
5. *$R = \emptyset$ or R is the range of a recursive function.*

Note that $\emptyset$ is decidable and consequently semi-decidable. And more generally, any finite set is recursive.

Proof. We proved last time that 1) $\Leftrightarrow$ 2).

We show that 1) $\Rightarrow$ 4). Assume that M semi-decides R. We can assume without loss of generality that if $w \in R$, then $M(w)$ enters q_Y and if $w \notin R$, then $M(w)$ diverges (i.e. it never halts).

Modify M to form M' in the following way. If M enters q_Y, then M' instead prints a 0 on the tape and enters q_H. So M' computes a partial recursive function whose domain is

$$\{w | M(w) \downarrow\} \ ,$$

as desired.

We show that 1) $\Rightarrow$ 3). Assume that M is as above, that is, M semi-decides R. Form M'' such that if $M(w)$ enters q_Y, M'' writes w as its output and enters q_H. It is fairly straightforward to see how this might be done. For example, one could have M'' make a copy of w on the tape, then run M on one of the two copies of w. If M enters state q_Y, then return to the left-most entry on the tape of the other copy of w.

We show that 1) $\Rightarrow$ 5). Assume again that M is as above, and furthermore that $R \neq \emptyset$. Let $w_0 \in R$. We give the following algorithm for deciding R:

Define the function f by:

$$f(w,i) = \begin{cases} w & \text{if } M(w) \text{ enters } q_Y \text{ in } \leq i \text{ steps} \\ w_0 & \text{otherwise} \end{cases} .$$

It is easy to see that f is recursive, since we can just run M on w for i steps. And range$(f) = R$. Clearly range$(f) \subset R$, and $R \subset$ range(f) since for all $w \in R$ there is some $i \in \mathbb{N}$ such that $M(w)$ accepts in less than or equal to i steps.

The rest of the theorem has been left for HW. □

One remark deserves to be made about our proof of 1) $\Rightarrow$ 5). It assumes that $|\Sigma| \geq 2$. But there are a couple of ways that we can use a single member of Σ^* to encode a pair of members in Σ^*.

For example, for $i, j \in \mathbb{N}$, we can give a single $k \in \mathbb{N}$ which encodes both i and j:

$i\backslash j$	1	2	3	4	...
1	0	2	5	9 ...	
2	1	4	8 ...		
3	3	7 ...			
4	6 ...				

So we can combine the two inputs that f needs above into simply 1 input. This means that we do not need the assumption that $|\Sigma| \geq 2$.

2 The Universal Turing Machine

Very roughly speaking, the Universal Turing Machine is a Turing machine that can do anything that any other Turing machine can do.

We can code Turing machines as strings. Fix Σ and assume that $\{0, 1\} \subset \Sigma$. Encode a Turing machine M by a string in Σ^*. Call this the Gödel number of M, in other words $\ulcorner M \urcorner$.

One way to do this is as follows. Given a fixed M, assume without loss of generality that $\Gamma = \Sigma$ for M. At first we use

$$\Sigma' = \Sigma \cup Q \cup \{R, L, N\} \cup \{\$, ,\} .$$

The Gödel number of M is a description of the transition function δ for M. Let $\delta(q, \sigma) = (\sigma', m, q')$, where $m \in \{R, L, N\}$, $q, q' \in Q$, and $\sigma, \sigma' \in \Sigma$. We can encode this as follows:

$	q	,	σ	,	σ'	,	m	,	q'	$

Now we concatenate entries like this for all of the values of δ (since δ is finite, this will work).

And we can now reduce the Σ that we were working with. Encode q, q' in binary notation. Also encode R, L, N as binary strings. For example, R as 00, L as 11, N as 01. So our tape

$	q	,	σ	,	σ'	,	m	,	q'	$

becomes a string of 0's, 1's, commas, and $'s. Now encode 0 as 00, 1 as 11, $ as 01, and 'comma' as 10 (for example). Then we have a string of symbols from Σ which completely encode our Turing machine M.

We now conclude with a more formal definition of the Universal Turing Machine.

Definition 3. The Universal Turing Machine, U, is a Turing machine with two inputs defined as follows:

$$U(\ulcorner M \urcorner, w) = M(w) .$$

For $\ulcorner M \urcorner$ the Gödel number of a Turing machine M, and $w \in \Sigma^*$. If M(w) halts in state q_Y or q_N, then so does U. And if M(w) halts in q_H and outputs v, so does U. And if $M(w) \uparrow$, then so does $U(\ulcorner M \urcorner, w)$.