8. Semicomputability

[Rogers, Ch. 5]

8.1 Semicomputable relations

Let B be an n-ary relation on $\mathbb{N}$. We say that B is

- ($\mathcal{G}$-)***semicomputable*** (***s/comp***) or ($\mathcal{G}$-)***computably enumerable*** (***c.e.***) or ***recursively enumerable*** (***r.e.***) iff B is the ***halting set*** of a $\mathcal{G}$-program, i.e., $B = \mathrm{HS}(\mathcal{P})$ for some $\mathcal{G}$-program $\mathcal{P}$, where

  $$\mathrm{HS}(\mathcal{P}) \ =_{df}\ \{\vec{x} \mid \mathcal{P} \text{ halts on input } \vec{x}\},$$

  or equivalently, iff $B = \boldsymbol{dom}(g)$ for some $\mathcal{G}$-computable function g.

- ***semi-decidable*** (***s/dec***) or ***semi-effective*** iff there is an algorithm which gives ***positive information*** (only) on membership of B, i.e. with input $\vec{x}$, the algorithm halts iff $\vec{x} \in B$.

Notation. We will sometimes drop the '$\mathcal{G}$' in front of '***comp***' and '***s/comp***'.

NOTES:

1. By CT, B is ($\mathcal{G}$-)***s/comp*** iff B is ***s/dec***.

2. If B is ***dec***, then B is certainly ***s/dec***, since an algorithm which decides B can easily be modified to one which gives positive information only on B. (However, the converse is not true, as we will see!)
   The analogous result for $\mathcal{G}$-***comp*** B is:

Theorem 8.1. *If B is comp, then B is s/comp.*

Proof: Let P be a $\mathcal{G}$-program that computes χ_B. Then the program

$$\boxed{\begin{array}{c} P \\ [A]\ \mathsf{if}\ Y = 0\ \mathsf{goto}\ A \end{array}}$$

halts on B. $\square$

Theorem 8.2 (Post). *B is comp iff B and $\overline{B}$ are s/comp.*

Proof: ($\Rightarrow$) Suppose B is comp. By Cor. 7.2, $\overline{B}$ is comp.
The result follows from Thm 8.1.

($\Leftarrow$) Suppose B and $\overline{B}$ are s/comp. Then for some indices p, q,

$$B = \mathrm{HS}(\mathcal{P}_p) \qquad \text{and} \qquad \overline{B} = \mathrm{HS}(\mathcal{P}_q)$$

Intuitively, on any input $\vec{x}$, we ***merge*** or ***interleave*** executions of $\mathcal{P}_p$ and $\mathcal{P}_q$ until one of them halts. Note that, by Theorem 7.6, there is a macro for $\boldsymbol{stp}^{(n)}$. So the program

```
[A]  if stp^(n)(X⃗, p, T) goto C
     if stp^(n)(X⃗, q, T) goto E
     T++
     goto A
[C]  Y++
```

computes χ_B. $\square$

Theorem 8.3. *If B, C are s/comp, then so are $B \cap C$ and $B \cup C$.*

Proof: Suppose $B = \mathrm{HS}(\mathcal{P}_p)$ and $C = \mathrm{HS}(\mathcal{P}_q)$. Then the program

```
P_p
... (re-initialise variables)
P_q
```

halts for inputs in $\mathrm{HS}(\mathcal{P}_p) \cap \mathrm{HS}(\mathcal{P}_q) = B \cap C$.
On the other hand, *merging* $\mathcal{P}_p$ and $\mathcal{P}_q$, the program

```
[A]  if stp^(n)(X⃗, p, T) goto E
     if stp^(n)(X⃗, q, T) goto E
     T++
     goto A
```

halts for inputs in $\mathrm{HS}(\mathcal{P}_p) \cup \mathrm{HS}(\mathcal{P}_q) = B \cup C$. $\square$

Intuitively, if B and C are ***s/dec***, then so are $B \cap C$, and $B \cup C$.

Let ($\mathcal{G}$-)COMP and ($\mathcal{G}$-)SCOMP be the classes of $\mathcal{G}$-comp and $\mathcal{G}$-s/comp sets respectively. Then, clearly,

$$\boxed{\mathrm{PR} \;\subseteq\; \mathcal{G}\text{-COMP} \;\subseteq\; \mathcal{G}\text{-SCOMP} \;\subseteq\; \wp(\mathbb{N})}$$

We devote the rest of the section to the questions concerning the properness of the above "$\subseteq$" inclusions (except for the leftmost one, which will be answered later, in §11, Exercise 3).

By Corollary 7.2, $\mathcal{G}$-COMP is closed under $\cup$, $\cap$ and $^{-}$,
and by Thm 8.3, $\mathcal{G}$-SCOMP is closed under $\cup$ and $\cap$.
The obvious question now is: Is $\mathcal{G}$-SCOMP closed under $^{-}$?
The answer to this question also resolves the question concerning the second "$\subseteq$" inclusion.

Let $W_n = \mathrm{HS}(\mathcal{P}_n) = \boldsymbol{dom}(\varphi_n)$. So for all x,

$$x \in W_n \iff \varphi_n(x)\downarrow,$$

yielding an ***effective listing*** of $\mathcal{G}$-SCOMP:

$$W_0, W_1, W_2, \ldots$$

Now let $K = \{x \mid x \in W_x\}$. Then

$$x \in K \iff x \in W_x \iff \varphi_x(x)\downarrow. \tag{12}$$

Lemma 8.4. *$\overline{K}$ is not s/comp.*

Proof: Suppose $\overline{K}$ is s/comp. Then for some n,

$$\overline{K} = W_n. \tag{13}$$

So for all x,

$$x \in W_n \overset{(13)}{\iff} x \in \overline{K} \overset{(12)}{\iff} x \notin W_x.$$

Putting $x = n$,

$$n \in W_n \iff n \notin W_n,$$

a contradiction. $\square$

Theorem 8.5. *K is s/comp, but not comp.*

Proof:
K is the domain of $\lambda x \cdot \Phi(x, x)$, which is comp, by the UFT (Thm 7.5). So K is s/comp. Suppose K is comp.
Then by Thm 8.2, $\overline{K}$ is s/comp, contradicting Lemma 8.4. $\square$

NOTES:

1. Note again the use of ***diagonalisation*** (or ***self-reference***, or ***self-application***) in the proof of Lemma 8.4.

2. The non-computability of K is just another formulation of the unsolvability of HP (see Exercise, p. 7-4).

3. $\mathcal{G}$-COMP $\subset$ $\mathcal{G}$-SCOMP by Thm 8.5, with witness K.

4. Similarly, $\mathcal{G}$-SCOMP $\subset \wp(\mathbb{N})$, by Lemma 8.4, with witness $\overline{K}$.

5. Alternatively, we can argue that $\mathcal{G}$-SCOMP $\subset \wp(\mathbb{N})$ since $\mathcal{G}$-SCOMP is *countable* by the enumeration $W_0, W_1, \ldots$ whereas $\wp(\mathbb{N})$ is *uncountable* by Cantor's theorem (Thm 1.12(b)). Hence:

$$\boxed{\text{PR} \;\subseteq\; \mathcal{G}\text{-COMP} \;\subset\; \mathcal{G}\text{-SCOMP} \;\subset\; \wp(\mathbb{N})}$$

EXERCISE:

By re-proving Cantor's theorem in the present context, produce a witness that $\mathcal{G}$-SCOMP $\subset \wp(\mathbb{N})$. What is the connection between this witness and the one in Note 4?

8.2 Characterisation of semicomputable sets using CT

Although the theorems in this section do not depend on CT, we will give proofs using CT for simplicity (following [Rog67]). (Remember, functions are assumed to be partial, unless explicitly called "total".)

Theorem 8.6. *If f is **total comp**, then **ran**(f) is **s/comp**.*

Proof$^{\mathbf{CT}}$: Suppose that f is total comp. The following algorithm *halts only on inputs in **ran**(f)*:

"With ***input*** x, compute (in turn):
$f(0)$, $f(1)$, $f(2), \ldots$
until you find an i with $f(i) = x$;
then ***halt***."

By CT there is a $\mathcal{G}$-program corresponding to this algorithm. □

Theorem 8.7. *If f is **comp**, then **ran**(f) is **s/comp**.*

Proof$^{\mathbf{CT}}$: By modifying the algorithm in the proof of Thm 8.6 as follows:

"With ***input*** x, generate **ran**(f) by
dovetailing (***interleaving***) i.e. in stages:
at ***stage n*** :
do ***n steps*** in the computation of
$f(0)$, $f(1)$, $f(2), \ldots, f(n-1)$;
halt (if and) when you find an i with $f(i) = x$." □

Theorem 8.8. *If f is **total comp** and **strictly increasing**, then **ran**(f) is **comp**.*

Proof$^{\mathbf{CT}}$: By modifying the algorithm in the proof of Thm 8.6 as follows:

"With ***input*** x, compute (in turn):
$f(0)$, $f(1)$, $f(2), \ldots$
until you find an i with $f(i) \geq x$;
if $f(i) = x$: ***output*** 1;
if $f(i) > x$: ***output*** 0." □

The next two theorems can be considered a ***converse*** to Theorem 8.6.

Theorem 8.9. *If B is* ***s/comp*** *and $B \neq \emptyset$, then there exists a* ***total comp*** *f such that $B =$* ***ran***(f).

Proof$^{\mathbf{CT}}$: Let g be comp with ***dom***$(g) = B$.
The following algo computes a total function f with ***ran***$(f) = B$:

> "With ***input*** x, generate list of elements of B by ***dovetailing***:
> at ***stage n*** :
> do n steps in the computation of
> $g(0), g(1), g(2), \ldots, g(n-1)$;
> for all $i < n$ such that $g(i)\downarrow$ in $\leq n$ steps,
> add i to list;
> ***output*** element number x in the list. $\square$

NOTE 1: f is an ***effective listing*** of B. It is ***total*** (even if B is finite), since it has ***repetitions***.

Theorem 8.10. *If B is* ***s/comp*** *and* ***infinite****, then there exists a* ***total 1-1 comp*** *f such that $B =$* ***ran***(f).

Proof$^{\mathbf{CT}}$: EXERCISE. $\square$

NOTE 2: Here f is an ***effective listing*** of B ***w/o reps***.

By combining the above results, we get:

Theorem 8.11.

(a) *Suppose $B \neq \emptyset$. Then*
B is ***s/comp*** *iff B is the range of a* ***total comp*** *function.*

(b) *B is* ***s/comp*** *iff B is the range of a* ***comp*** *function.*

Proof: (a) From Thms 8.6 and 8.9.

(b) From Thms 8.7 and 8.9, and since $\emptyset =$ ***dom***$(\lambda x\cdot\uparrow) =$ ***ran***$(\lambda x\cdot\uparrow)$. $\square$

NOTE: This theorem gives the justification for the terminology "***computably enumerable***" or "***recursively enumerable***".
It can be viewed as a ***constructive version of Thm 1.11***, part (1)$\Leftrightarrow$(2).
It says (using CT): ***"s/dec $\iff$ eff. listable"***

EXERCISES:

1. Prove Theorem 8.10.

2. Prove: B is ***s/comp*** iff B is the range of a ***1-1 computable*** function. (This can be viewed as a ***constructive version of Thm 1.11***, part (1)⇔(3).)

8.3 Enumerability of total computable functions

In §7.4 we defined an (n+1-ary) ($\mathcal{G}$-computable) ***universal function*** for $\mathcal{G}$-COMP$^{(n)}$ in terms of an enumeration $\varphi_0^{(n)}, \varphi_1^{(n)}, \ldots$ of $\mathcal{G}$-COMP$^{(n)}$. In this section we show that this ***cannot*** be done for $\mathcal{G}$-TCOMP$^{(n)}$ (even when $n = 1$). It is for this reason that we consider (***partial***) $\mathcal{G}$-computable functions as ***more fundamental*** than ***total*** $\mathcal{G}$-computable functions.

For any binary function F and $n \in \mathbb{N}$, let

$$F_n =_{df} \lambda x \cdot F(n, x).$$

We now investigate whether the UFT holds for $\mathcal{G}$-TCOMP$^{(1)}$, i.e. whether there is a ***universal function*** $F \in \mathcal{G}$-TCOMP$^{(2)}$, for which the sequence

$$F_0, F_1, F_2, \ldots \tag{14}$$

enumerates all of $\mathcal{G}$-TCOMP$^{(1)}$.
(Note there *is* a UFT for $\mathcal{G}$-COMP, by Thm 7.5.)

Theorem 8.12. *If $F \in \mathcal{G}$-TCOMP$^{(2)}$, then*

(a) *for all n, $F_n \in \mathcal{G}$-TCOMP$^{(1)}$, but*

(b) *we can find a function $h \in \mathcal{G}$-TCOMP$^{(1)}$ which is **outside** the enumeration (14), i.e. for all n, $F_n \neq h$.*

Proof: (a) Clear.

(b) Define $h(x) = F(x, x) + 1$, i.e., ***diagonalize out!*** □

Corollary 8.13. *There exists no UFT for $\mathcal{G}$-TCOMP.*

NOTES:

1. Note the use of ***diagonalisation*** again in the proof of Theorem 8.12.

2. By CT this theorem says: Given any effective enumeration of some class of total computable functions, we can "diagonalise out" to obtain a total computable function outside the class!

3. Thus, although $\mathcal{G}$-TCOMP is ***enumerable*** by ***classical reasoning*** (being a subset of the enumerable set $\mathcal{G}$-COMP), it is (by CT) ***not effectively enumerable***! (See also Exercise 3 below.)

4. Why can the method of "diagonalising out" not be used to contradict the UFT for $\mathcal{G}$-COMP? Because the definition $h(x) \simeq \varphi_x(x) + 1$ does *not* imply that for all y, $\varphi_y \neq h$. For suppose $h = \varphi_n$. Then the equation
$$\varphi_n(n) \simeq h(n) \simeq \varphi_n(n) + 1$$
just means that $\varphi_n(n) \uparrow$.

EXERCISES:

1. Let $\mathcal{G}$-COMP-PRED be the class of $\mathcal{G}$-comp predicates, i.e. ***total*** functions $P : \mathbb{N} \to \mathbb{2}$. Is there a UFT for $\mathcal{G}$-COMP-PRED?

2. (a) Let PR-DERIV be the set of all PR-derivations. Show how (by Gödel numbering or otherwise) to give an ***effective enumeration*** of PR-DERIV, and hence (as a sublist) an effective enumeration of the set PR-DERIV$^{(1)}$ of PR-derivations of unary functions. This induces an ***effective enumeration*** $f_0, f_1, f_2, \ldots$ of PR$^{(1)}$.

   (b) Let F be the binary ***universal function*** for PR$^{(1)}$ under the enumeration in (a), i.e. for all m and n, $F(m, n) = f_m(n)$. Clearly F is ***effective***, and hence in $\mathcal{G}$-TCOMP, by CT. But is F **PR**?
   More generally, is there a UFT for PR at all?

3. Show that the set $\{y \mid \varphi_y \text{ is total}\}$ is not s/comp.
(*Hint*: Otherwise there would be a UFT for $\mathcal{G}$-TCOMP).