Recursive and recursively enumerable sets

So far we have concentrated on the aspect of algorithms to do with computing functions from inputs to outputs.

Another important use of algorithms is to generate, or enumerate, the elements of some set of data.

One says that a set S is effectively enumerable if there is some algorithm A which lists the elements of S:

$$S = \{ A(0), A(1), A(2), \ldots \}$$

(It may well be that an element of S occurs many times in the list, but no matter.)

EXAMPLE :

The set PR of partial recursive functions is effectively enumerated by the algorithm A which, given input x,

decodes x as a pair $x = \langle n, e \rangle$, then

decodes e as a register machine program $Prog_e$,

and returns the n-ary computable (hence partial recursive) function $\varphi_e^{(n)}$, where

$$\varphi_e^{(n)}(x_1, \ldots, x_n) = y \overset{def}{\iff}$$ computation of $Prog_e$ started with $R1, \ldots, Rn$ set to $x_1, \ldots, x_n$ halts with $R0 = y$

(because every element of PR is of the form $\varphi_e^{(n)}$ for some n & e)

Clearly, S has to be a countable set if it is effectively enumerable.

[Recall:

S is <u>countably infinite</u> if there is some bijection (= one-one and onto function) between $\mathbb{N}$ and S.

S is <u>countable</u> if it is either finite or countably infinite.

S is <u>uncountable</u> if it is not countable.

Eg. $Fun(\mathbb{N}, \mathbb{N})$ is uncountable, by Cantor's Diagonal Argument.]

The notion of "effective enumerability" is an informal one, because it refers to the informal notion of "algorithm". We can formalize it using the notion of computable (= partial recursive) function provided we identify the set S to be enumerated with a subset of $\mathbb{N}$.... (Since S is necessarily countable, we can always do this some way).

DEFINITION:

A subset $S \subseteq \mathbb{N}$ of numbers is <u>recursively enumerable</u> (or <u>r.e.</u>, for short) if & only if <u>either</u> it is empty ($S = \emptyset$) <u>or</u> there is a (total) recursive function $f \in \mathrm{Fun}(\mathbb{N}, \mathbb{N})$ so that

$$S = \{ f(n) \mid n \in \mathbb{N} \}$$

Recall: $S \subseteq \mathbb{N}$ is <u>decidable</u> if & only if the <u>characteristic function</u> of S

$$\chi_S(x) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{if } x \notin S \end{cases}$$

is computable. (cf. p55)

Such sets are also called <u>recursive</u> (since χ_S is computable if & only if it is recursive, being a total function).

PROPOSITION:

Every recursive set is recursively enumerable.

__Proof__

Suppose S is recursive. If $S = \emptyset$, then S is r.e. by definition; otherwise we can find some $x_0 \in S$. Then since χ_S is recursive, so is

$$f(x) \stackrel{\text{def}}{=} \text{ifzero}(\chi_S(x), x_0, x)$$

and $S = \{f(x) \mid x \in \mathbb{N}\}$, so S is r.e. $\square$

In the section on the Halting Problem we saw that the set

$$\{e \in \mathbb{N} \mid \varphi_e \text{ is a total function}\}$$

is undecidable. In fact it is not even recursively enumerable ...

EXAMPLE of a non-r.e. set

$$\text{TOT} \stackrel{\text{def}}{=} \{e \in \mathbb{N} \mid \varphi_e \text{ is a total function}\}$$

is not recursively enumerable.

__Proof__

If TOT were r.e., then (since $\text{TOT} \neq \emptyset$) $\text{TOT} = \{f(x) \mid x \in \mathbb{N}\}$ for some recursive function $f \in \text{Fun}(\mathbb{N}, \mathbb{N})$.

Let $u \in \text{Pfn}(\mathbb{N}^2, \mathbb{N})$ be the partial function $u(e, x) \stackrel{\text{def}}{=} \varphi_e(x)$

CLAIM

(1) u is partial recursive; hence so is $g(x) \stackrel{\text{def}}{=} u(f(x), x) + 1$

(2) g is total; hence $g = \varphi_e$ for some $e \in \text{TOT}$, but

(3) $e \neq f(x)$ for any $x \in \mathbb{N}$ — contradiction!

Proof of the CLAIMS:

(1) follows from the work we did in the section on a universal register machine $\mathcal{U}$, since $u(e,x)$ is the result (if any) of running $\mathcal{U}$ starting with $P=e$ and $A=[x]$.
Thus u is computable, and hence is partial recursive.

(2) Since by assumption on f, for all $x \in \mathbb{N}$ $f(x) \in TOT$ so $\varphi_{f(x)}(x)\downarrow$, so $g(x)\downarrow$ (by definition of g). Thus g is total recursive, and hence $g=\varphi_e$ for some $e \in TOT$.

(3) If $e=f(x)$, then

$$
\begin{aligned}
g(x) &= \varphi_e(x) && \text{since } g=\varphi_e \\
&= u(e,x) && \text{by definition of } u \\
&\neq u(e,x)+1 && \text{since } u(e,x)=g(x)\downarrow \\
&= u(f(x),x)+1 && \text{since } e=f(x) \\
&= g(x) && \text{by definition of } g
\end{aligned}
$$

contradiction. So $e \neq f(x)$ for any x, contradicting the assumption that f enumerates TOT (since $e \in TOT$). $\square$

EXAMPLE of an r.e. set that is not recursive

is provided by the undecidability of the Halting Problem, which in particular implies that

$$H \stackrel{\text{def}}{=} \{e \in \mathbb{N} \mid \varphi_e(0)\downarrow\}$$ [cf. S_2 on p56]

is undecidable, i.e. is not recursive. But

H <u>is</u> r.e. because $H = Dom(f)$ the domain (of definedness) of the partial recursive function

$$f(x) \stackrel{\text{def}}{=} u(x,0)$$

(where u is as above)

and in general we have ...

PROPOSITION:

For a subset $S \subseteq \mathbb{N}$, the following are equivalent:

(1) S is recursively enumerable

(2) $S = \mathrm{Im}(f)$, the <u>image</u> of a (unary) partial recursive function

(3) $S = \mathrm{Dom}(f)$, the <u>domain</u> of a unary partial recursive function

(4) S is <u>semi-decidable</u>, meaning that the partial function

$$in_S(x) = \begin{cases} 1 & \text{if } x \in S \\ \text{undefined} & \text{if } x \notin S \end{cases}$$

is partial recursive.

NOTATION:

Given a partial function $f \in Pfn(X, Y)$

$\mathrm{Dom}(f) \stackrel{\text{def}}{=} \{x \in X \mid f(x)\downarrow\}$ the <u>domain</u> (of definedness) of f

$\mathrm{Im}(f) \stackrel{\text{def}}{=} \{y \in Y \mid \text{for some } x \in X,\ f(x) = y\}$ the <u>image</u> of f

<u>Proof</u> of the Proposition

We will show $(2) \Rightarrow (1) \Rightarrow (3) \Rightarrow (4) \Rightarrow (2)$.

In all cases the implications are trivial if S is empty (since $in_\emptyset$ = completely undefined function, is partial recursive and has domain & image $= \emptyset$). So we can assume $S \neq \emptyset$, say $x_0 \in S$.

(2)⇒(1) :

Let M be a register machine computing $f(a)$ in $R0$ when started with $R1 = a$.

Construct a new machine M' computing as follows:

decode $R1$ as a pair $\langle a, t \rangle$;
run M for t steps starting with $R1 = a$ and if it halts by then, set $R0$ to the value it computes in $R0$, else set $R0$ to x_0

Let f' be the unary function computed by M' (in $R0$, starting with input in $R1$).

By construction f' is total recursive and $f'(x) \in S$ for all $x \in \mathbb{N}$ (since M only computes values $f(a)$ that lie in S).
Conversely, if $y \in S = \mathrm{Im}(f)$, then $y = f(a)$ for some a.
Now M computes $f(a)$ in a finite number of steps starting from $R1 = a$, say t steps. Then by construction of M'
$f'(\langle a, t \rangle) = f(a) = y$. Thus every element of S is enumerated by the recursive function f' – so S is r.e.

[Remark: using the techniques of the proof of "f computable $\Rightarrow f \in PR$" one can show that S is enumerated by a primitive recursive function, since

$$f'(x) = \mathrm{ifzero}(I \dot{-} \mathrm{lab}(\mathrm{state}(\pi_1(x), \pi_2(x))), \mathrm{val}_0(\mathrm{state}(\pi_1(x), \pi_2(x))), x_0)$$

where π_1, π_2 are primitive recursive projection functions satisfying
$\pi_1(\langle a, t \rangle) = a$, $\pi_2(\langle a, t \rangle) = t$, & $\langle \pi_1(x), \pi_2(x) \rangle = x$.]

$(1) \Rightarrow (3)$:

Since we are assuming $S \neq \emptyset$,
$S = \{ f(n) \mid n \in \mathbb{N} \}$ for some recursive function f.

Then
$$g(x,y) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } f(y) = x \\ 1 & \text{if } f(y) \neq x \end{cases}$$
is also recursive, since $g(x,y) = 1 \dot{-} eq(f(y), x)$.
Thus $\mu(g)$ is partial recursive, and
$$\begin{aligned} x \in Dom(\mu(g)) &\Leftrightarrow \mu(g)(x)\downarrow \\ &\Leftrightarrow g(x,y) = 0 \text{ for some } y \\ &\Leftrightarrow f(y) = x \text{ for some } y \\ &\Leftrightarrow x \in S \end{aligned}$$
Thus $S = Dom(\mu(g))$, as required.

$(3) \Rightarrow (4)$:

If $S = Dom(f)$ with $f \in PR$, then
$$in_S(x) \equiv ifzero(f(x), 1, 1)$$
is also partial recursive, hence computable: so S is semi-decidable.

$(4) \Rightarrow (2)$:

Let M be a register machine computing $in_S(x)$ in $R0$ when started with x in $R1$.

Then

(where X is some register not mentioned in M's program.)

$$\text{START} \rightarrow \boxed{X := R1} \rightarrow \boxed{M} \rightarrow \boxed{R0 := X} \rightarrow \text{HALT}$$

computes the partial recursive function
$$f(x) \stackrel{\text{def}}{=} \begin{cases} x & \text{if } in_S(x)\downarrow \\ \uparrow & \text{if } in_S(x)\uparrow \end{cases}$$

and hence $Im(f) = S$.

$\square$

DEFINITION:

A subset $S \subseteq \mathbb{N}$ is called <u>co-r.e.</u> iff $\mathbb{N} \setminus S \ (\stackrel{\text{def}}{=} \{x \in \mathbb{N} \mid x \notin S\})$ is r.e.

PROPOSITION:

S is recursive if & only if it is both r.e and co-r.e.

PROOF:

$$\chi_S(x) = \begin{cases} 0 & \text{if } x \notin S \\ 1 & \text{if } x \in S \end{cases}$$

$$\chi_{\mathbb{N}\setminus S}(x) = \begin{cases} 0 & \text{if } x \in S \\ 1 & \text{if } x \notin S \end{cases}$$

$$= \mathrm{ifzero}(\chi_S(x), 1, 0)$$

So S recursive $\Rightarrow$ $\mathbb{N}\setminus S$ recursive

So S recursive $\Rightarrow$ S & $\mathbb{N}\setminus S$ both r.e.

Conversely...

Suppose

$$\left.\begin{array}{l} S \\ \mathbb{N}\setminus S \end{array}\right\} \text{enumerated by recursive function} \begin{cases} f \\ g \end{cases}$$

Let M be register machine which when started with x in $R1$:

computes successive values of the sequence
$$g(0), f(0), g(1), f(1), g(2), f(2), \ldots$$
halting (at n^{th} place in sequence, say) first time get a value $= x$, and returning $\begin{cases} 0 \\ 1 \end{cases}$ in $R0$ if n is $\begin{cases} \text{even} \\ \text{odd} \end{cases}$.

Then M decides membership of S, because...

M is guaranteed to halt because f and g are total; and then

either $x \in S$ – in which case $x = f(n)$, some n

or $x \notin S$ – in which case $x = g(n)$, some n.

More formally $\chi_S(x) \equiv mod_2(\mu(h)(x))$, where

$$h(x,y) \stackrel{\text{def}}{=} 1 \dot{-} eq(x, ifzero(mod_2(y), g(half(y)), f(half(y))))$$

and mod_2, $half$, eq were defined on pages 120 & 124.

Thus χ_S is recursive because f & g are and because eq, $ifzero$, mod_2 & $half$ are (primitive) recursive. $\square$

SUMMARY

- Formalization of intuitive notion of ALGORITHM in several <u>equivalent</u> ways

  cf. "Church-Turing Thesis"

- Limitative results : undecidable problems
  uncomputable functions

  "programs as data" + diagonalization