Chapter 6

Recursive and Semi-Recursive Relations

(Rogers chapter 5, Cutland Chapters 6, 7.)

Recall that the *characteristic function* χ_R for a k-ary relation R
is a function $f : \mathbf{N}^k \to [0,1]$ such that for all $\overrightarrow{x}$,

$$R(x) \Rightarrow f(\overrightarrow{x}) = 1 \ \wedge \ \neg R(\overrightarrow{x}) \Rightarrow f(\overrightarrow{x}) = 0$$

A *verifying function* for a relation R
is a partial function $\phi : \mathbf{N}^k \to \{0,1\}$ such that

$$R(\overrightarrow{x}) \iff \phi(\overrightarrow{x}) \simeq 1$$

A *refuting function* for a relation R
is a partial function $\phi : \mathbf{N}^k \to \{0,1\}$ such that

$$\neg R(\overrightarrow{x}) \iff \phi(\overrightarrow{x}) \simeq 0$$

A *computably decidable* or *recursive* relation
is a relation whose characteristic function is total recursive.

A *computably verifiable* or *semi-recursive* relation
is a relation with a partial recursive verifying function.

A *computably refutable* or *co-semi-recursive* relation
is a relation with a partial recursive refuting function.

The "computably decidable, verifiable, refutable" notation is not standard, but is more apt than the standard "recursive, semi-recursive" terminology.

Examples

- computably decidable: Gödel numbers of valid formulas in a propositional language.
- computably verifiable: Gödel numbers of valid formulas in the language of arithmetic.
- computably refutable: first-order consistency in the language of arithmetic.
- none of the above: the complete theory of arithmetic.

6.1 Arity and Clutter Reduction

From now on, you may drop the arity parameter when working with unary functions.

$$\phi_n = \phi_n^1$$

You can handle k-ary functions and relations in the same numbering by using a k-ary coding function.

$$\begin{aligned} \phi_n^k(\vec{x}) &\simeq \phi_m(\langle\vec{x}\rangle); \\ R(\vec{x}) &\simeq S(\langle\vec{x}\rangle). \end{aligned}$$

Exercise 6.1 *Show that there is an effective conversion from index n to index m and conversely. That is, there exist total recursive f, g such that for all k-ary $\vec{x}$,*

$$\begin{aligned} \phi_n^k(\vec{x}) &\simeq \phi_{f(n)}(\langle\vec{x}\rangle); \\ \phi_{g(n)}^k(\vec{x}) &\simeq \phi_n(\langle\vec{x}\rangle). \end{aligned}$$

Hint: use the universal construction to set up the application of an index to a coded sequence of arguments like this:

$$\psi(i, \vec{x}) \simeq ((\mu z)\ U(i, (z)_0, (z)_1, \langle\vec{x}\rangle))_1$$

Apply the s-m-n *theorem and see what you get. For the other side, do something similar. Whenever you need a "uniform" transformation of indices, seek the "right" application of the universal construction and then apply* s-m-n *to the index position.*

6.2 Alternate Characterizations of Verifiability

The following results and concepts are used and referred to constantly. Please take special effort to master them right away. The following, odd notation is ubiquitous. Just grin and memorize it.

$$\begin{aligned} W_n &= dom(\phi_n); \\ E_n &= rng(\phi_n). \end{aligned}$$

Proposition 6.1 *Relation R is computably verifiable $\iff$ there exists an n, such that for all x,*

$$R(\vec{x}) \iff W_n(\langle \vec{x} \rangle)$$

Exercise 6.2 *Prove it. Hint: Use a μ operator to produce infinite loops in the right places and conversely, use sg to cut outputs higher than 1 down to 1.*

In set theory, a countable set S can be presented as a list $\{x_i : i \in \mathbf{N}\}$. Since each $i \in \mathbf{N}$ is associated with at most one element of S, there exists a total, onto mapping $f : S \to \mathbf{N}$ such that $x_i = f(i)$. Indeed,

> a set S is **set-theoretically enumerable** iff either S is empty or there exists a total mapping $f : \mathbf{N} \to S$ such that f is onto. Then you may think of the set as being enumerated as $\{f(i) : i \in \mathbf{N}\}$.

From the point of view of computability, the idea of an arbitrary, set-theoretic f is uninteresting. It would be more interesting if f were computable, so that by computing $f(i)$, one can effectively obtain x_i. This "motorized" version of enumerability is called "recursive enumerability".

S is *recursively enumerable* (r.e.) $\iff$ $S = \emptyset$ or $(\exists$ tot. rec. $f)$ $S = rng(f)$.

The following result says that this amounts to the same thing as computable verifiability.

Proposition 6.2 *(The fundamental theorem of r.e. sets)*

$$S \text{ is r.e.} \iff (\exists n)\ S = W_n$$

Proof. Suppose that S is r.e. If $S = \emptyset$, then $S = dom(\emptyset)$. But $\emptyset =$ the function defined by $(\mu z)(x \neq x)$, and hence is partial recursive.

Now suppose that $S = rng(f)$ where f is total recursive. One must show that for some i, $S = W_i$. Define:

$$\psi(x) = (\mu z)(f(z) = x).$$

Function ψ is partial recursive since f is recursive. Hence there exists i such that $\psi = \phi_i$.

Note that:

$$\begin{aligned} x \in S &\iff (\exists y) f(y) = x \\ &\iff (\mu z)(f(z) = x) \downarrow \\ &\iff x \in dom(\phi_i) \\ &\iff x \in W_i. \end{aligned}$$

Hence, $S = W_i$.

Conversely, suppose that $S = W_i = dom(\phi_i)$. If $S = \emptyset$, then we are done. So suppose $S \neq \emptyset$. So there is at least one k such that $k \in S$. Now we need to produce a total recursive enumeration$f$ of S. We can't assume that $\phi_i(x)$ will halt, so some dovetailing is required. Also, we can't assume that W_i is infinite, so perhaps at some point we run out of new things to enumerate. We must therefore continue to output some previously enumerated number until a new element of W_i is detected. That sounds like a job for recursion.

$$\begin{array}{rcl} f(0) & = & k; \\ f(n+1) & = & \begin{cases} (n)_2 & \text{if } (\forall z \leq n)\ (((n)_2)_0 \neq f(z) \wedge U(w, (n)_0, (n)_1, (n)_2); \\ f(n) & \text{otherwise.} \end{cases} \end{array}$$

This is a course of values recursion over primitive recursive constructions and hence is not only total recursive, but primitive recursive, which is always nice to know. $\dashv$

Corollary 6.3 *Every r.e. set is enumerated by a primitive recursive function.*

This follows from the fact that total recursive enumeration constructed in the preceding proof is in fact primitive recursive. $\dashv$

Corollary 6.4 *The sequence*

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

is an enumeration of the computably verifiable sets.

The corollary provides a way of using our effective numbering of *Part* to number the verifiable sets without having to do it again from scratch. According to the following result, there is an equivalent, effectively intercompilable enumeration of the r.e. sets based on ranges instead of domains.

Proposition 6.5 *Effective conversions between domains and ranges.*

$$\exists \text{ total recursive } f\ \forall n\ W_n = E_{f(n)}$$

$$\exists \text{ total recursive } g\ \forall n\ W_{g(n)} = E_n$$

Exercise 6.3 *Prove it. Hint: use the* s-m-n *theorem over a universal dovetailing construction analogous to the one given in the preceding exercise. This is yet another application of the universal and s-m-n theorems. Optional bonus question: how about a total recursive translation between ranges of primitive recursive functions and domains of partial recursive functions? Now you are dealing with different enumerations, so there is no universal theorem in the primitive recursive enumeration. One way is to generate a primitive recursive index for the course-of-values recursion over the simultaneous recursion defining U.*

Corollary 6.6 *The sequence*

$$E_0, E_1, \ldots, E_n, \ldots$$

is also an enumeration of the computably verifiable sets.

Recall that the primitive recursive relations are closed under bounded existential quantification. The computably verifiable relations are closed under full existential quantification over **N**. And every partial recursive relation results from an unbounded existential quantification over a (primitive) recursive relation. So unbounded existential quantification has a role similar to that of unbounded minimalization: it suffices to extend primitive recursive relations to recursively enumerable relations just as minimalization extends primitive recursive functions to partial recursive functions. This suggests a handy table of analogies that may prove useful as you continue. Incidentally, why is there no version of $\exists$ and μ for the middle line?

$$\begin{array}{rcl} \text{set} & : & \text{function} :: \\ \exists & : & \mu :: \\ \text{verifiable} & : & \text{partial} :: \\ \text{decidable} & : & \text{total} :: \\ \text{prim. rec.} & : & \text{prim. rec} :: \\ \exists \leq x & : & \mu \leq x. \end{array}$$

Proposition 6.7 ***The Projection Theorem*** *S is computably verifiable* $\iff$

$$(\exists \text{ primitive recursive } R)(\forall x)\ (S(x) \iff (\exists y)\ R(x, y)).$$

Proof. Let S be computably verifiable. So $\exists n(S = W_n)$, by proposition 6.1. Thus, for all x,

$$\begin{array}{rcl} S(x) & \iff & W_n(x) \\ & \iff & \phi_n(x) \downarrow \\ & \iff & \exists z\ U(n, (z)_0, (z)_1, \langle x \rangle) \end{array}$$

Recall that U is primitive recursive.

Conversely, suppose that

$$S(x) \iff (\exists z)\ R(z, x),$$

where R is r.e. So for some n, $R(\langle z, x \rangle) \iff W_n(\langle z, x \rangle)$. Define

$$\psi(x) \simeq (\mu w)\ U(n, (w)_0, (w)_1, \langle (w)_2, x \rangle)$$

This is partial recursive, so for some m,

$$\phi_m = \psi.$$

Now for the fun:

$$\begin{aligned}
W_m(x) &\iff \phi_m(x)\downarrow \\
&\iff \psi(x)\downarrow \\
&\iff (\mu w)\ U(n,(w)_0,(w)_1,\langle(w)_2,x\rangle)\downarrow \\
&\iff \exists w\ U(n,(w)_0,(w)_1,\langle(w)_2,x\rangle) \\
&\iff \exists z\ \phi_n(\langle z,x\rangle)\downarrow \\
&\iff \exists z\ W_n(\langle z,x\rangle) \\
&\iff \exists z\ R(z,x) \\
&\iff S(x)
\end{aligned}$$

So $S = W_i$ is computably verifiable, by proposition 6.1. $\dashv$

Corollary 6.8 *(Summary) The following are equivalent:*

S *is computably verifiable*

S *is r.e.*

$(\exists n)\ S = E_n$

$(\exists n)\ S = W_n$

$\exists$ *primitive recursive* R *such that* $\forall x(S(x) \leftrightarrow (\exists y)\ R(x,y))$.

6.3 The Halting Problem

What can you do with the W_n notation? By Corollary 6.8, you have a nice, effective enumeration of the computationally verifiable sets. Thus, you can form a two-dimensional table of zeros and ones as follows:

$$T[n,m] = \chi_{W_n}(m)$$

Thus, each row of the table is the characteristic function of a computably verifiable set and the characteristic function of each computably verifiable set occurs at least once in the table (infinitely often, actually).

Now define the set K whose characteristic function is the diagonal of the table:

$$\begin{aligned}
\chi_k(n) &= T[n,n] \\
&= \chi_{W_n}(n)
\end{aligned}$$

Then the characteristic function of $\neg K = N - K$ is the counter-diagonal of the table:

$$\begin{aligned}
\chi_{\neg K}(n) &= \overline{sg}(T[n,n]) \\
&= \overline{sg}(\chi_{W_n}(n))
\end{aligned}$$

Then $\neg K$ is not a computably verifiable set, since its characteristic function is concocted to differ from each row of the table along the diagonal. Thus:

Proposition 6.9 *$\neg K$ is not r.e.*

The proof of proposition 6.9 is almost the same as Cantor's argument that the power set of **N** is not enumerable. The difference is that in Cantor's argument, you know that the diagonal's characteristic function yields a subset of **N** so the enumeration assumption is rejected. In this case, you know from the fundamental theorem of r.e. sets that the effective enumeration $W_0, W_1, W_2, \ldots$ exists, so the diagonal characteristic function yields a non-r.e. set.

Unwinding the definition of K yields:

$$\begin{aligned} K(n) &\iff \chi_{W_n}(n) = 1 \\ &\iff W_n(n) \\ &\iff n \in dom(\phi_n) \\ &\iff \phi_n(n) \downarrow \end{aligned}$$

Thus, K is the set of all indices that return an output or "halt" when given themselves as input. For this reason, K is called the **halting problem**.

Exercise 6.4 *You have just seen that $\neg K$ is not computably verifiable. Show that K is computably verifiable. Hint: use an existential quantifier over an r.e. relation.*

The halting problem talks about a computation never halting. That sounds like a "universal law" governing what a computation will do for eternity. Hume's problem of induction arises when one tries to empirically verify a universal law governing the future. The halting problem is not computably verifiable either. In either case, laws governing the unbounded future aren't verifiable. That is the fundamental analogy we wish to explore this term.

6.4 Alternate Characterizations of the Recursive Sets

Decision intuitively requires that one be able to verify whether yes and verify whether no.

Proposition 6.10 *S is recursive $\iff$ $S, \overline{S}$ are both r.e.*

Exercise 6.5 *Prove it. Hint: One way is immediate using results proved above. The other requires a nice dovetailing construction, because you don't want to commit a suki by focusing on S forever before checking $\overline{S}$. But you can count on one or the other halting because each number is either in S or $\overline{S}$.*

Corollary 6.11 *K, $\neg K$ are not recursive.*

The next result is very interesting because it links the straightforwardly epistemological concept of decidability with the function theoretic concept of monotonicity. The idea is that you can use a monotone enumeration to decide the set: once you see a bigger thing than what you are looking for, you know you won't ever find what you are looking for. Conversely, if you can decide a set, then you can make sure that nothing smaller than x will have to be added to the enumeration before you add x. Compare this with the fundamental theorem for r.e. sets. In that case, you can't. So here is another nifty analogy:

$$\begin{array}{rcl} \text{verifiable} & : & \text{decidable} :: \\ \text{partial} & : & \text{total} :: \\ \text{nondecreasing} & : & \text{monotone increasing.} \end{array}$$

Proposition 6.12 *S is recursive* $\iff$

$$S \text{ is finite } \vee \ (\exists \text{ monotone, total recursive } f)\ (S = rng(f)).$$

Proof: Suppose S is recursive. If S is finite, we are done. So suppose that S is infinite. Define

$$\begin{array}{rcl} f(0) & = & (\mu x)\ (S(x)) \\ f(n+1) & = & (\mu x)\ (S(x) \wedge (\forall_{z \leq n})\ (f(n) \neq x)) \end{array}$$

Conversely, suppose that S is finite. Then a case definition (using one case per element) yields a program for the characteristic function of S, so S is primitive recursive, and hence recursive.

Finally, suppose that S is the range of monotone, total recursive f .... $\dashv$

Exercise 6.6 *Complete the proof. Hint: refer to the intuitive motivation above.*

6.5 Closure Laws

These are very important. So you get to prove them!

Proposition 6.13

1. The r.e. sets are closed under unbounded existential quantification under unbounded existential quantification.

2. The r.e. sets are not closed under complementation (think of your one example so far of a non-r.e. set.

3. The recursive sets form a Boolean algebra $(Rec, \cap, \cup, \mathbf{N}, \emptyset)$, where $\cap, \cup$ are finitary.

4. The recursive and r.e. relations are both closed
   under Cartesian product $\times$.

Exercise 6.7 *Prove it. Hint: For a Boolean algebra it suffices to show closure under $\wedge$, $\neg$. Conjunction over r.e. relations requires dovetailing. Doesn't the $\exists$ case sound familiar?*

Exercise 6.8 *(optional) Show that the recursive sets are not closed under $\exists$ or $\forall$ and the r.e. sets are not closed under $\forall$. Try showing these facts using only results on the table. Hint: express the halting problem and its complement using quantifiers over the Kleene predicate.*