Recursion-theoretic background for Learning Theory

Nina Gierasimczuk, Dick de Jongh

February 12, 2013

1 Coding with $\mathbb{N}$

We will introduce the notions we need in a rather informal manner. It may be helpful to consult in addition a basic introduction to recursion theory. Let us first introduce *coding* by natural numbers, The *natural numbers* are the elements of the set $\mathbb{N} = \{0, 1, 2, \dots\}$. Natural numbers can be used to code all kinds of discrete objects and processes so that one can talk about these objects and processes and their properties as if one talks about natural numbers and their properties, a very great advantage both mathematically and in its unifying effect. The first basic idea is the *pairing function*, a bijection from $\mathbb{N} \times \mathbb{N}$ onto $\mathbb{N}$. Usually one takes for this the *Cantor function* $j(x, y) = 1/2((x + y)^2 + 3y + x)$. The function j has *inverses* j_0 and j_1 such that for all natural numbers x, y:

$$j_0(j(x, y)) = x, \;\; j_1(j(x, y)) = y, \;\; j(j_0(x), j_1(x)) = x.$$

This function enables us to behave as if we talk about *pairs* of natural numbers while actually mentioning directly only the natural numbers themselves.

Next one can introduce *triples* of natural numbers, and n-tuples of natural numbers in general. A triple $j^3(x, y, z)$ can simply be introduced by means of the pairing function:

$$j^3(x, y, z) = j(j(x, y), z).$$

In general: $j^{n+1}(x_1, \dots, x_{n+1}) = j(j^n(x_1, \dots, x_n), x_{n+1})$. A last step is the introduction *finite sequences* of natural numbers of varied length. Even that can be done with the pairing function:

$$\langle x_0, \dots, x_n \rangle = j(n + 1, j^{n+1}(x_0, \dots, x_n)),$$

with $j^1(x)$ interpreted as x. For these sequences the length is indicated by lh. Thus, we have $lh(\langle x_0, \dots, x_n \rangle) = n + 1$, and in this particular representation also the equation $lh(x) = j_0(x)$. Note that we are using $\langle x_0, \dots, x_n \rangle$ ambiguously between the sequence of numbers and the number that codes it. Such

ambiguities are useful. There are *inverses* $(x)_i$ such that, if $x = \langle x_0, \ldots, x_n \rangle$ and $i \leq n$, then $(x)_i = x_i$.

So, now any number can be used as a code for a sequence of some arbitrary fixed length and of a sequence of varied length. Other objects can be coded similarly easily. As an example, one can code the symbols $p, \wedge, \vee, \rightarrow, \neg,), ($ of a propositional language by $0, 1, 2, 3, 4, 5, 6, 7$ respectively, the propositional variables as pairs of p and a natural number, and formulas as the codes of certain sequences of such symbols. There are various ways to also introduce codings for finite sets of natural numbers and thereby for finite sets all kinds of discrete objects which we will not spell out.

2 Partial Recursive Functions

All this means that in studying effective processes and procedures one can, for a large part, restrict one's attention to the natural numbers. It turns out that very many attempts to describe the notion of effective function from natural numbers to natural numbers lead to an extension to *partial functions*: $\varphi(n)$ may be *defined* ($\varphi(n)\downarrow$) or *undefined* ($\varphi(n)\uparrow$). For such partial functions F and G one defines $F(x) \simeq G(x)$ iff ($F(x)\downarrow$ iff $G(x)\downarrow$, and $F(x) = G(x)$ if both are defined). Also the class of effectively computable partial functions invariably turns out to be the same. The functions in this class are called *partial recursive functions*. The (*total*) *recursive functions* are the members of this class that are defined everywhere. One of the many ways of doing this is the introduction of a theoretical type of machine that calculates functions, Turing machines, register machines, the Post processes, etc.

Another important point to note is that we need to study only unary functions. Functions of two or more arguments can be studied as functions applied to pairs or longer sequences and thereby by the coding as unary functions. This will not inhibit us at all in speaking liberally of functions with sequences as arguments or values. It may be easier to do so as a manner of speaking. By coding e.g. the Turing machines by natural numbers one can give each partial recursive function the code of the Turing machine that computes it. We write φ_e for the function with code e, and e is the called the *index* of the function, or better *an* index, one function will have may different indices.

It is a fact of experience called the *Church-Turing thesis* that functions that can be seen to be effectively computable are recursive. Thus $+$ and $\times$ are recursive functions and also the Cantor pairing function, lh and the inverses of the sequences. There are many closure operations that lead from recursive functions to recursive functions. For example, the pairing function was obtained by *composition* from $+$ and $\times$ and other recursive functions. The recursive functions are closed under composition. Another important closure operation is *recursion*. Its basic form is that, from recursive functions G and H, one can define a recursive function F as follows:

$$F(x, 0) = G(x)$$

$$F(x, y+1) = H(F(x, y), x, y).$$

It is easy to see that F can be computed if G and H can. One may also start with a finite number of values of F "given", e.g. $F(x,0), F(x,1), F(x,2)$ instead of only one. For us a particular form is most useful, recursion over sequences. For example, if G and H are recursive, the F is, given the following equations:

$$F(x, \langle y \rangle) = G(x, y)$$
$$F(x, \sigma \,^\frown \langle y \rangle) = H(F(x, \sigma), x, y).$$

3 Recursive Sets

With regard to *sets*, or equivalently, *predicates* one is in first instance interested in the sets for which one can decide membership. In this set up these are the so-called *recursive sets*, the sets A for which the characteristic function K_A is recursive:

$$K_A(n) = 1 \text{ if } n \in A,\ K_A(n) = 0 \text{ if } n \notin A.$$

An important recursive predicate is *Kleene's T-predicate*; $Texy$ expresses that e is (the code of) a Turing-machine that on input of the number x leads to a concluded computation (coded by) y. It is combined with a recursive *outcome function* U, Uy is the outcome of the concluded computation y. This leads to

Theorem 1 (Kleene's Normal Form Theorem) *For each partial recursive function F there exists an e such that $F = \varphi_e$ and, for all x, $F(x) \simeq U(\mu y\, Texy)$.*

The complexity of the computations of $F = \varphi_e$ can then be measured by the function $\Phi_e(x) \simeq \mu y\, Texy$. This function produces the whole computation instead of only its outcome.

4 Recursively Enumerable Sets

A wider class than the recursive sets is of great interest, the class of the *recursively enumerable* or *r.e.* sets. This is the most general type of language in the so-called Chomsky-hierarchy and it will be the most general type of language we will consider. Recursively enumerable sets can be defined in various ways. They can be defined as the ranges of partial recursive functions, as the domains of partial recursive functions (i.e. as the domain on which the function is defined) and, thirdly as the ranges of total recursive functions, but then the empty set has to be added as an additional possibility. The last possibility is the most intuitive, one can imagine the set as being generated by some recursive process. The set (of codes) of the provable formulas of some finitely axiomatizable theory is a natural example. But the standard way is as the domains of the partial recursive functions. Naturally, they are the coded by the number of the function: $W_e = \{n \,|\, \varphi_e(n)\!\downarrow\}$. In that case e is called an *index* for the language $L = W_e$. Tu put it in the form of a theorem:

Definition 1 *A subset A of the natural numbers is* recursively enumerable (r.e.) *iff*

1. *A is $dom(F)$ for F partial recursive, i.e. $A = \{x \mid \exists y\, Texy\}$ for some e in which case we write $A = W_e$, OR*
2. *A is $range(F)$ for F partial recursive, i.e. $A = \{u \mid \exists x, y\, (Texy \wedge u = Uy)\}$ for some e, OR*
3. *A is $range(F)$ for F total recursive, or $A = \emptyset$.*

By means of the coding we can consider languages over arbitrary alphabets as (r.e.) subsets of the natural numbers. The set Σ^* of all strings over Σ is then simply $\mathbb{N}$. It is a very important fact that there exist sets that are recursively enumerable but not recursive. We will give an example presently. Let us first note some important closure properties of recursive and r.e. sets.

Theorem 2

1. *The recursive sets are closed under* intersection, union and complement*. This means that the formulas describing the corresponding predicates are closed under all the usual connectives. These predicates are also closed under* bounded quantification*, i.e. quantifiers of the form $\forall x \leq F(y)$ and $\exists x \leq F(y)$ for some recursive function F.*
2. *The r.e. sets are closed under* intersection and union*. This means that the formulas describing the corresponding predicates are closed under $\wedge$ and $\vee$. These predicates are also closed under bounded quantification.*
3. (Definition by cases, example) *If G, H and K are (partial) recursive functions and A and B are recursive predicates, then F, defined by the following equations, is a (partial) recursive function.*

$$F(x) = G(x) \text{ if } A(x)$$
$$F(x) = H(x) \text{ if } \neg A(x) \text{ and } B(x)$$
$$F(x) = K(x) \text{ else.}$$

There are important connections between recursively enumerable and recursive sets.

Theorem 3 (Post's theorem) *The set A is recursive iff both A and its complement $\overline{A} = \mathbb{N} - A$ are recursively enumerable.*

The idea of the proof is that one can decide the membership of n in a set A if both A and $\overline{A}$ are recursively enumerable by simultaneously generating the elements of both A and $\overline{A}$ and noting in which of the two n turns up. Another important characterization of r.e. sets is given by the following.

Theorem 4 *The set A is recursively enumerable iff there exists a 2-place recursive predicate B such that $x \in A$ iff $\exists y\, B(x, y)$.*

The proof from right to left of this theorem consists of noting that the function $\varphi(x) \simeq \mu y\, B(x,y)$ is partial recursive and has as its domain the set $\{x \mid \exists y\, B(x,y)\}$. Here $\mu y\, B(x,y)$ means the smallest y such that $B(x,y)$ if such a value exists and undefined otherwise; it can be computed by checking $B(x,0), B(x,1), B(x,2), \ldots$. Hence, in general, partial recursive functions are closed under the notation $\mu y\,(y \leq F(x) \wedge B(x,y))$.

One should also note that two or more existential quantifiers in sequence can be contracted to one by means of the pairing function. For example, if B is recursive, $\exists y, z\, B(x,y,z)$ is r.e. because it is equivalent to $\exists y\, B(x, j_1(y), j_2(y))$ (look e.g. at Definition **??**(2)).

Theorem 5 *The set $K = \{x \mid \varphi_x(x)\downarrow\}$ is r.e., but not recursive.*

Proof The set K is r.e., because it is the domain of a partial recursive function. Assume that K is recursive. Then $\overline{K}$ is recursive as well, and hence the domain of some partial recursive function φ_e. Applying φ_e to e itself leads to a contradiction: $\varphi_e(e)\downarrow$ iff $\varphi_e(e)\uparrow$. □

Of course, this shows more in general that the so-called *halting problem*, to determine of a certain machine (or function) whether it stops on a certain input, is undecidable. For this reason, it is sometimes good to have a closer look and rely on the fact that (partial) recursive functions are computed in *stages*. We will write $\varphi_e^s(x) = y$ to state that the value of $\varphi_e(x)$ is y and that the computation has been completed by the s-th stage. Under such an interpretation $\varphi_e^s(x)\downarrow$ is a decidable predicate. Also r.e. sets can then be thought of as being generated in stages: $W_{e,s} = \{x \mid \varphi_e^s(x)\downarrow\}$ is always a finite, decidable set, and the $W_{e,s}$ are weakly increasing in the s so that W_e is the limit of the $W_{e,s}$.

A more complicated example of r.e. sets which are not recursive is given by the following.

Theorem 6 *There exist two disjoint r.e. sets A and B which are* recursively inseparable*, i.e. there exists no recursive C such that $A \subseteq C$ and $C \cap B = \emptyset$.*

5 Two Important Theorems of RT

For completeness' sake we now give the two most important theorems of elementary recursion theory. The S-m-n-theorem is not used explicitly by us, but it is actually needed to prove things that we assume without proof like the fact that indices of the W_e coding finite sets can be found recursively in the codes of these finite sets. We will give one example of the use the recursion theorem to get the feeling of this magical theorem, but we will hardly use it in the course.

Theorem 7 (S-m-n-theorem) *There exists a recursive function S such that, for each e, x, y, $\varphi_{S(e,x)}(y) \simeq \varphi_e(x,y)$.*

Theorem 8 (Recursion Theorem) *For each total recursive function F there exists an e such that $\varphi_{F(e)} = \varphi_e$.*

Lemma 9 *For each infinite r.e. set W_e there exists a strictly increasing total recursive function F such that $range(F) \subseteq W_e$.*

Proof Let $\langle s_0, x_0\rangle, \langle s_1, x_1\rangle, \langle s_2, x_2\rangle, \ldots$ enumerate all pairs of natural numbers recursively. Define $F(0)$ to be x_i if $\langle s_i, x_i\rangle$ is the first member of the sequence such that $\varphi_e^{s_i}(x_i)$ is defined, and define $F(x+1)$ to be x_j if $\langle s_j, x_j\rangle$ is the first member of the sequence such that $\varphi_e^{s_j}(x_j)$ is defined for which $x_j > F(x)$.

One may note that $F(0)$ can be taken to be arbitrarily large, that $range(F)$ is a recursive set, and that in case W_e is finite the definition of F still works, but gives a partial recursive function. □

Definition 2 *An index e is* minimal *if for no $f < e$, $W_f = W_e$.*

Lemma 10 *If A is an r.e. set consisting of minimal indices, the A is finite.*

Proof Assume that A is an r.e. set consisting of minimal indices and A is infinite. By Lemma **??** there exists a strictly increasing, recursive F with $range(F) \subseteq A$. We noted that $F(0)$ can be taken to be greater than 0, which means that, for each x, $F(x) > x$. Apply the recursion theorem to obtain an e such that $\varphi_{F(e)} = \varphi_e$ and hence that $W_{F(e)} = W_e$. Then $F(e) > e$ contradicts the fact that $F(e)$ as a member of A is supposed to be minimal. □