MATHEMATICAL LOGIC FOR COMPUTER SCIENCE (A.Y. 20/21)

HANDOUT N. 11

Abstract. We give a characterization of computable functions and computably enumerable sets in terms of first-order definability in $\mathbf{N}$. In other words we give a *machine-independent* characterization of these fundamental notions of Computability Theory.

1. Computable Functions

We fix a new definition of (partial) computable function. This definition is close to some of the first proposed formal definitions of the informa notion of "algorithm". The definition is phrased in standard mathematical terms and does not refer to machines or computing devices. Definitions of this kind were proposed and studied by Gödel, Herbrand, and Kleene in the 1930s.

Figure 1. Kurt Gödel (1906-1978)

Definition 1.1 (The class $\mathcal{C}$)**.** The class $\mathcal{C}$ is the class of possibly partial functions of type $\mathbf{N}^k \to \mathbf{N}$ defined as follows: $\mathcal{C}$ is the smallest class containing addition, multiplication, projections, the characteristic function of the predicate of numerical identity, and closed under composition and minimalization.

We use Greek lowercase letters φ, ψ, etc. to denote partial functions. We use latin lowercase letters f, g, etc. to denote total functions.

The following claim explains the interest of the class $\mathcal{C}$.

Claim 1.2. *The class $\mathcal{C}$ coincides with the class of computable functions.*

The above Claim admits no proper mathematical proof, in that it equates a formal notion (being in $\mathcal{C}$) with an informal one. What can be proved instead, is that $\mathcal{C}$ coincides with the class of functions computable

Notes by Lorenzo Carlucci, carlucci@di.uniroma1.it.

by Turing Machines. Similarly, it coincides with the class of functions defined by other well-known models of computation.

The composition operator refers to the following generalized form of composition: let $\theta_1, \dots, \theta_k$ be k functions of n arguments and let ψ be a function of k arguments. The composition of $\phi, \theta_1, \dots, \theta_k$ is the function φ of n arguments defined as follows:

$$a_1, \dots, a_n \mapsto \psi(\theta_1(a_1, \dots, a_n), \dots, \theta_n(a_1, \dots, a_n)).$$

It is reasonable to argue that if $\psi, \theta_1, \dots, \theta_k$ are computable then φ is also computable.

The operator of minimalization is defined as follows. If $\psi(\vec{x}, y)$ is a function, then the function $\varphi(\vec{x})$ denoted by $\min z(\psi(\vec{x}, z) = 0)$ is defined as the minimum z such that the values $\psi(\vec{x}, 0), \psi(\vec{x}, 1), \dots, \psi(\vec{x}, z-1)$ are defined and different from 0 and $\psi(\vec{x}, z)$ is defined and equal to 0. If such a z does not exist then the function is undefined.

Remark 1.3. The operator of minimalization corresponds to an unlimited and possibly not ending search. The class of computable functions therefore contains partial functions. In general it is possible to prove that no list of total functions $(f_i)_{i \in \mathbf{N}}$ such that the operation $i \mapsto f_i(i)$ is algorithmic can contain the class of algorithmically computable functions. In fact, if this were the case, then the function $i \mapsto f_i(i) + 1$ would be algorithmically computable. But this function does not belong to the list (Exercise).

Remark 1.4. Modify as follows the definition of minimization:

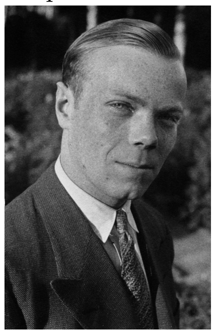
If $\psi(\vec{x}, y)$ is a function, then the function $\varphi(\vec{x})$ denoted by $\mu y(\psi(\vec{x}, y) = 0)$ is defined as the minimum y such that $\psi(\vec{x}, y)$ is defined and equal to 0. If such a y does not exist then the function is undefined.

For this notion of minimalization it is *not* easy to argue that if ψ is computable then ϕ is computable. The problem has to do with the possibility that $\psi(\vec{x}, y)$ is undefined for some values of $\vec{x}, y$. Suppose we try to find the minimum y such that $\psi(\vec{x}, y) = 0$ by a complete search, trying values $\psi(\vec{x}, 0), \psi(\vec{x}, 1), \psi(\vec{x}, 2), \dots$ If the minimum y such that $\psi(\vec{x}, y) = 0$ is $y = 4$ but, say, $\psi(\vec{x}, 2)$ is undefined, we will be stuck in computing the latter and never get to try $y = 3$.

This could be patched for by resorting to approximations of the computations of $\psi(\vec{x}, n)$ for $n = 0, 1, 2, \dots$. For example, at stage s we compute s steps of computation of $\psi(\vec{x}, 0), \dots, \psi(\vec{x}, s)$, and so on. Also in this case, however, we might not be able to find the minimum y we search for: suppose that the minimum such y is $y = 3$ but that $\psi(\vec{x}, 6) = 0$. Also, suppose that $\psi(\vec{x}, 6) = 0$ takes 8 steps of computation while to compute $\psi(\vec{x}, 3)$ it takes 100 steps. Then at stage $s = 8$ of our approximate computation we would find out that $\psi(\vec{x}, 6) = 0$ while the computation of $\psi(\vec{x}, 3)$ would still be incomplete, so that we would misleadingly output 6.

This indicates that it is not easy to argue that this concept of minimalization preserves computability. In fact, we can prove that requiring closure under this operator takes us out of the class of computable functions: a decision procedure for the Halting Set can be designed. (Exercise).

FIGURE 2. Jacques Herbrand (1908-1931)



The notion of computable function is used to define the fundamental concept of decidable (or algorithmic or computable or recursive) set.

Definition 1.5 (Algorithmically Decidable Set). A set is *algorithmically decidable* if its characteristic function is a computable function.

Obviously the characteristic function of a set is a total function. Synonyms of algorithmically decidable are: decidable, computable, recursive.

Definition 1.6 (Limited Quantifiers). Limited universal quantification indicates an occurrence in a formula of a universal quantifier in a context of the following type:

$$(\forall x)(x < t \to G),$$

where t is a term that does not contain x and G a formula. Limited existential quantification indicates an occurrence in a formula of an existential quantifier in a context of the following type:

$$(\exists x)(x < t \land G),$$

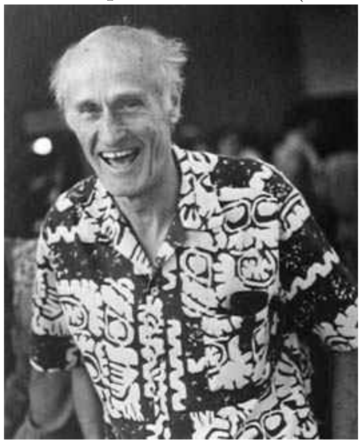
where t is a term that does not contain x.

We abbreviate $(\forall x)(x < t \to G)$ by $(\forall x < t)G$ and the construct $(\exists x)(x < t \land G)$ by $(\exists x < t)G$.

Definition 1.7 (Δ_0 and Σ_1 Formulas). A formula is Δ_0 if it contains no quantifiers or if it contains only limited quantifiers. A formula is Σ_1 if is Δ_0 or of type $\exists x_1 \dots \exists x_n G$ with G a Δ_0 formula.

It is easily observed that the class of Δ_0 formulas is closed under all Boolean connectives. The class of Σ_1 formulas is closed under disjunction (by the properties of the existential quantifier) but also under conjunction (a conjunction of Σ_1 formulas, $\exists x_1 \dots \exists x_n G(x_1, \dots, x_n) \land \exists x_1 \dots \exists x_n H(x_1, \dots, x_n)$ is equivalent to a Σ_1 formula, $\exists x_1 \dots \exists x_n \exists y_1, \dots, \exists y_n (G(x_1, \dots, x_n) \land H(y_1, \dots, y_n)$ where the y_is are new variables).

Figure 3. Stephen C. Kleene (1909-1994)



2. Characterization Theorem

Definition 2.1 (Definability of a function in a structure). Let A be a set and $\varphi : A^k \to A$ a function, $k \geq 1$. Let $\mathfrak{A}$ be a structure with domain A. We say that φ *is definable* in $\mathfrak{A}$ by a formula $F(x_1, \dots, x_k, y)$ (with exactly $k+1$ free variables) if, for each $a_1, \dots, a_k, b \in A$, the following holds:

$$\varphi(a_1, \dots, a_k) = b \iff \mathfrak{A} \models F[a_1, \dots, a_k, b],$$

i.e., if the $k+1$-ples $(a_1, \dots, a_k, b)$ belonging to the graph of φ are exactly those that satisfy $F(x_1, \dots, x_k, y)$ in $\mathfrak{A}$ assigning a_i to x_i and b to y.

Theorem 2.2 (Characterization Theorem)**.** *A function (possibly partial) $\varphi : \mathbf{N}^k \to \mathbf{N}$ is computable if and only if it is definable in $\mathbf{N}$ by a Σ_1 formula (in the language of arithmetic).*

Proof. We prove that each computable function is definable by a Σ_1 formula. We prove that the class of functions definable by a Σ_1 formula contains the basic functions and is closed under composition and and minimalization.

Claim 2.3 (Basic Functions)**.** *Basic functions are definable in $\mathbf{N}$ by Σ_1 formulas.*

Proof. Addition is defined by the formula $x + y = z$, multiplication by the formula $x \times y = z$, projection $\pi_i^n(x_1, \ldots, x_n) = x_i$ (where $i \in [1, n]$) is definable by the formula $x_1 = x_1 \wedge \cdots \wedge x_i = z \wedge \cdots \wedge x_n = x_n$, the characteristic function of equality is definable by the formula $(x = y \wedge z = 1) \vee (x \neq y \wedge z = 0)$. $\square$

Claim 2.4 (Closure under composition)**.** *Functions definable by Σ_1 formulae in $\mathbf{N}$ are closed under composition.*

Proof. Let $G_i(\vec{x}, y_i)$ be a Σ_1 formula that defines θ_i, for $i \in [1, k]$, and $H(y_1, \ldots, y_k, z)$ a Σ_1 formula that defines ψ. Then

$$\exists y_1 \ldots \exists y_k (G_1(\vec{x}, y_1) \wedge \cdots \wedge G_k(\vec{x}, y_k) \wedge H(y_1, \ldots, y_k, z)$$

is Σ_1 and defines $\varphi(\vec{x}) = \psi(\theta_1(\vec{x}), \ldots, \theta_k(\vec{x}))$.

If $\varphi(\vec{a}) = p$ then there are $q_1, \ldots, q_k$ such that $\theta_i(\vec{a}) = q_i$ and $\psi(q_1, \ldots, q_k) = p$. Therefore $\mathbf{N} \models G_i[\vec{a}, q_i]$ and $\mathbf{N} \models H[q_1, \ldots, q_k, p]$ by hypotheses on g_i and H. $\square$

Claim 2.5 (Closure under minimalization)**.** *Functions definable in $\mathbf{N}$ by Σ_1 formulae are closed under minimalization.*

Proof. Let $\varphi(\vec{x})$ be defined as $(\min z)(\psi(\vec{x}, z) = 0)$. Let $F_\psi(\vec{x}, z, y)$ be a Σ_1 formula that defines ψ. Then the following formula defines φ.

$$(\forall w \leq z)(\exists y)(F_\psi(\vec{x}, w, y) \wedge (y = 0 \leftrightarrow w = z)).$$

We prove that this formula is equivalent (over the standard model of arithmetic) to a Σ_1 formula. Let $F_\psi(\vec{x}, z, y)$ of form $(\exists v)F'(\vec{x}, z, y, v)$ with $F' \in \Delta_0$. The formula above is equivalent in $\mathbf{N}$ to

$$(\exists u)(\forall w \leq z)(\exists y, v \leq u)(F'(\vec{x}, z, y, v) \wedge (y = 0 \leftrightarrow w = z)).$$

Just choose the value of u large enough to allow to find below it witnesses for y and v that satisfy $F'(\vec{x}, z, y, v) \wedge (y = 0 \leftrightarrow w = z)$, for each choice of $w \leq z$. $\square$

The proof of the first part of the Theorem is thus completed. We shall now proceed with the proof of the second part.

We prove that every function definable in $\mathbf{N}$ by a Σ_1 formula is computable. Let $F(x_1, \ldots, x_k, y)$ be a Σ_1 formula that defines a function in $\mathbf{N}$. Let $\varphi_F : \mathbf{N}^k \to \mathbf{N}$ be the function defined in $\mathbf{N}$ by $F(x_1, \ldots, x_k, y)$. I.e.,

$$\varphi_F(n_1, \ldots, n_k) = m \iff \mathbf{N} \models F[n_1, \ldots, n_k, m].$$

F is of the form $(\exists z)H(x_1, \ldots, x_k, y, z)$ with $H \in \Delta_0$. Note that for each $\vec{n}$ there is at most one m such that $\mathbf{N} \models F_\varphi[\vec{n}, m]$. Then the function φ_F is equivalent to

$$\vec{n} \;\mapsto\; \text{the only } m \ \mathbf{N} \models F_\varphi[\vec{n}, m], \text{ if exists, otherwise undefined.}$$

We want to prove that the map just defined is computable. We give an informal proof. The details needed to make it completely rigorous are given below.

We can think to calculate φ_F as follows. For the sake of simplicity we write n instead of $\vec{n}$. Given the input n, we ask in sequence

$$\mathbf{N} \models F[n, 0]?, \mathbf{N} \models F[n, 1]?, \mathbf{N} \models F[n, 2]?, \ldots, \mathbf{N} \models F[n, i]? \ldots$$

If there exists a i SUCH THAT $\mathbf{N} \models F[n,i]$, then the procedure returns the minimum, otherwise it does not terminate. But let us see what it takes to answer questions $\mathbf{N} \models F[n,i]?$. Since F is Σ_1, we have that $F(x,y)$ is $\exists \vec{w} H(x,y,\vec{w}$ with H a Δ_0 formula. For the sake of simplicity we use w instead of $\vec{w}$. To answer the question $\mathbf{N} \models F[n,i]?$ we must answer the questions

$$\mathbf{N} \models H[n,i,0]?, \mathbf{N} \models H[n,i,1]?, \mathbf{N} \models H[n,i,2]?, \dots, \mathbf{N} \models H[n,i,j]?, \dots$$

The answer to $\mathbf{N} \models F[n,i]?$ is YES if and only if there exists a j such that $\mathbf{N} \models H[n,i,j]$.

Can organize the infinite succession of questions $\mathbf{N} \models H[n,i,j]?$ for $i,j \in \mathbf{N}$ in such a way as to make the procedure described above algorithmic? And in this case, for $i,j \in \mathbf{N}$, can we answer algorithmically the question $\mathbf{N} \models H[n,i,j]?$. In both cases the answer is positive. For the first point we just need an algorithmic coding of pairs (and more generally of finite sequences) of natural numbers (there are many, see *infra*). For the second point we observe that the satisfaction in $\mathbf{N}$ a formula of complexity Δ_0 is algorithmically decidable.

Proposition 2.6. *If $F(\vec{x})$ is Δ_0, then the set of $\vec{m}$ that satisfies $F(\vec{x})$ in $\mathbf{N}$ is decidable.*

Proof. We argue as follows (using Church's Thesis[1]). We first observe that if $t(x_1,\dots,x_n)$ is a term of the language of arithmetic and $(m_1,\dots,m_n)$ is a sequence of natural numbers, then the value of $t^{\mathbf{N}}(m_1,\dots,m_n)$ is calculable (with input $m_1,\dots,m_n$). We then show that if $F(x_1,\dots,x_n)$ is Δ_0 and $(m_1,\dots,m_n)$ are natural numbers, then it is algorithmically decidable if $\mathbf{N} \models F[m_1,\dots,m_n]$. We proceed by induction on the structure of the formula F. If F is an identity between terms, i.e., is of the form $t(\vec{x}) = s(\vec{x})$, it is sufficient to calculate the values of $t(m_1,\dots,m_n)$, $s(m_1,\dots,m_n)$ and to verify the numerical equality. If F is a boolean combination of formulas, it is sufficient to calculate the corresponding truth function. If F is of the form $\exists y < t(x_1,\dots,x_n) G(x_1,\dots,x_n,y)$, it is sufficient to calculate the value of $t^{\mathbf{N}}(m_1,\dots,m_n)$ (call this value p) and decide, for every $i < p$, if $\mathbf{N} \models G[m_1,\dots,m_n,i]$ or not. $\square$

In conclusion, the decision procedure of the function φ defined by F in $\mathbf{N}$ proceeds as follows. On input n, the algorithm checks for $m \in \mathbf{N}$, if $\mathbf{N} \models H[n,a,b]$ holds, where a,b are the only values such that m encodes the ordered pair (a,b) with respect to the fixed algorithmic encoding of pairs. If the answer is in the affirmative then the algorutm returns a. $\square$

3. Coding sequences and primitive recursion [Extra Material]

We integrate the missing details to make some of the above arguments perfectly rigorous. In particular we show that there is an algorithmic coding of the pairs of natural (of low syntactical complexity) and that the satisfaction of a formula Δ_0 is algorithmically decidable (and with a procedure of low complexity).

We introduce a function of *pairing* of low quantifier complexity. Define

$$(x,y) = \frac{(x+y)(x+y+1)}{2} y.$$

This function is a bijection between $\mathbf{N}^2$ and $\mathbf{N}$ and $x, y \le (x,y) < (x+y+1)^2$ holds. The relation $(x,y) = z$ is definable by the formula $(x+y)(x+y+1) + 2y = 2z$, which is Δ_0. The encoding extends to ordered n-tuples by setting $(x_1, x_2, \dots, x_n) := (x_1, (x_2, \dots, x_n)$. The maps obtained are all not decreasing in their arguments, and each is defined by a formula Δ_0. For precision

$$x_1, \dots, x_n \le (x_1, \dots, x_n) < (x_1 + x_2 + \cdots + x_n + 1)^{2^n}.$$

The encoding functions of pairs, triples, etc. thus defined are clearly computable (it is also possible to prove that there is a Δ_0 formula which expresses a coding of n-ples *uniformly* in n, see *infra*). In fact, these maps belong to a very interesting subclass of the computable functions, the .

[1]Recall that Church's Thesis is the thesis according to which each algorithmic function in intuitive sense is computable in the formal sense. Church's Thesis says that the formal definition of computable function properly captures the intuitive notion of algorithmic procedure. Invoking Church's Thesis allows avoiding tedious formalizations for proving that a certain function is computable

Definition 3.1 (Primitive Recursive Functions)**.** The class of *Primitive Recursive Functions* is the smallest class containing $Z : x \mapsto 0$, $S : x \mapsto x + 1$ and $\pi_i^n : (x_1, \ldots, x_n) \mapsto x_i$ $(1 \leq i \leq n)$ and closed under composition and primitive recursion, i.e., the schema

$$h(\vec{x}, y) = \begin{cases} f(\vec{x}) & \text{if } y = 0 \\ g(\vec{x}, y - 1, h(\vec{x}, y - 1)) & \text{if } y > 0, \end{cases}$$

where h in the second case is undefined if g is undefined. If h is defined according to the previous schema we say that h is defined by primitive recursion from f and g.

Primitive recursive functions are all computable and are all total. To prove that they are computable we must deal with encodings of sequences. The totality is obvious. Primitive Recursive Functions enjoy a remarkable property: They are *computationally honest*, in the sense of the following proposition.

Proposition 3.2. *If f is a function calculated from a program in a certain model of universal calculation (e.g. Turing machines, lambda calculus, machines to registers, etc.) in time or space limited by a primitive function recursive, then f is primitive recursive.*

We now prove that the problem with input $\vec{m}$

$$\mathbf{N} \models F[\vec{m}]?$$

is decidable by a primitive recursive decision procedure, if F is Δ_0.

Proposition 3.3. *If $F(\vec{x})$ is Δ_0, then the set of $\vec{m}$ that satisfy $F(\vec{x})$ in $\mathbf{N}$ is decidable by a primitive recursive procedure.*

Proof. You can associate inductively to every formula Δ_0 a primitive recursive function that decides its satisfaction in $\mathbf{N}$ (Exercise) or you can use the previous theorem and informally describe a decision procedure for Δ_0 formulas that operates in time limited by a primitive recursive function (Exercise). $\square$

The characterization of computable functions that we used for the Characterization Theorem hides a technique to eliminate primitive recursions. This technique requires certain procedures for encoding and decoding that are elementary but rather boring (more suitable for an introductory Computability course!). One can prove that all primitive recursive functions are computable (see *infra*).

In the most standard approach computable functions are defined as the smallest class of functions that contains the function zero, the successor, projections, and are closed under composition, primitive recursion and minimization. If you use this definition and you want to prove the characterization theorem you must explicitly deal with the case of Σ_1-definability in $\mathbf{N}$ of a function obtained by primitive recursion, and this requires the manipulation of formulas that define the coding of sequences.

The essential ingredient is a formula for uniformly representing the encoding of finite sequences (of variable length). Let us see why.

If f is defined by primitive recursion from g and h, then

$$f(x_1, \ldots, x_n, y) = z$$

if and only if **exists a sequence** of numbers $b_0, \ldots, b_y$ such that

$$b_0 = g(x_1, \ldots, x_n)$$

$$b_y = z$$

and for $i < y$,

$$b_{i+1} = h(x_1, \ldots, x_n, i, b_i).$$

Lemma 3.4 (Gödel)**.** *There is a function $\beta(x, y, z)$ such that for every $n \in \mathbf{N}$, for each sequence of natural numbers $s_0, \ldots, s_\ell$ there exist b, c such that*

$$\beta(b, c, i) = s_i,$$

for $i \in [0, \ell]$.

Proof. Let $\beta(x, y, z)$ equal the remainder of the division of x by $1 + (z + 1) \cdot y$. Let $m = \max(\ell, s_0, \ldots, s_\ell)$. Let $c = M!$. We define the sequence $(u_i)_{i \in [0,\ell]}$ as follows.

$$u_i = 1 + (i + 1) \cdot c.$$

We observe that the u_is are two by two relatively prime. Moreover, for every $i \in [0, \ell]$, $s_i < u_i$, because

$$s_i \leq m \leq m! = c < 1 + (i + 1)c = u_i.$$

by the Chinese Remainder Theorem[2] there exists $b < \Pi_{i=0}^{\ell} u_i$ such that for every $i \in [0, \ell]$,

$$b \equiv s_i \mod u_i.$$

Since $u_i < s_i$, it follows that s_i is the remainder of the division of b by u_i. Then by definition of β and since $s_i < u_i$, we have

$$\beta(b, c, i) = s_i.$$

□

Lemma 3.5. *The function β is primitive recursive and definable in $\mathbf{N}$ by a Σ_1 formula.*

Proof. For the first point it suffices to show that the function that given x, y computes the remainder of the division of y by x is primitive recursive. For the second point we consider the formula $B(x, y, z, v)$.

$$(\exists w)(x = (1 + (z + 1) \cdot y) \cdot w + v \wedge v < 1 + (z + 1) \cdot y).$$

□

The β function can be used to treat explicitly the case of primitive recursion in the Characterization Theorem. On the other hand, given the Characterization Theorem, the following proposition implies that the computable functions are closed under primitive recursion and that all primitive recursive functions are computable.

Proposition 3.6. *Functions definable in $\mathbf{N}$ from Σ_1 formulas are closed under primitive recursion.*

Proof. Let $f(x_1, \ldots, x_n, y)$ be defined by primitive recursion from $g(x_1, \ldots, x_n)$ and $h(x_1, \ldots, x_n, y, z)$. Then (as noted above)

$$f(x_1, \ldots, x_n, y) = z$$

if and only if there exists a sequence of numbers $b_0, \ldots, b_y$ such that

$$b_0 = g(x_1, \ldots, x_n)$$
$$b_y = z,$$

and $i < y$,

$$b_1 = h(x_1, \ldots, x_n, i, b_i).$$

If G is Σ_1 and defines g in $\mathbf{N}$ and H is Σ_1 and defines h in $\mathbf{N}$, then the following formula $F(x_1, \ldots, x_n)$ is Σ_1 and defines f in $\mathbf{N}$.

$$\begin{aligned}
&(\exists u)(\exists v)(((\exists w)(B(u, v, 0, w) \wedge G(x_1, \ldots, x_n, w)))\wedge \\
&\qquad B(u, v, y, z) \wedge (\forall i)(i < y \rightarrow \\
&\qquad (\exists w)(\exists w')(B(u, v, i, w) \wedge B(u, v, i + 1, w') \wedge H(x_1, \ldots, x_n, i, w, w'))))
\end{aligned}$$

□

Corollary 3.7. *Primitive Recursive Functions are computable.*

[2]For every n, for every $s_0, \ldots, s_\ell$, for every $u_0, \ldots, u_\ell$ pairwise relatively prime, there exists b such that $b \equiv s_i \mod u_i$. You can also always find such a $b < \Pi_{i=0}^{\ell} u_i$.