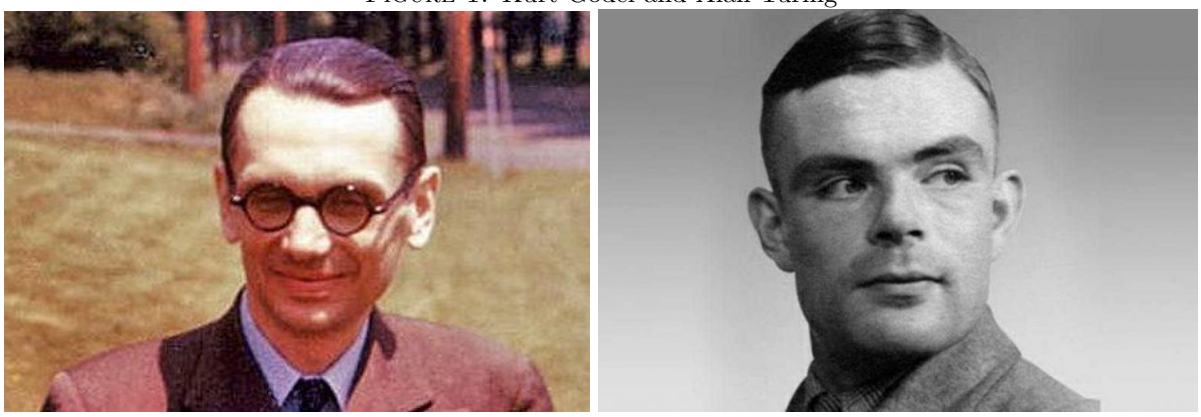
MATHEMATICAL LOGIC FOR COMPUTER SCIENCE
(A.Y. 20/21)

HANDOUT N. 12

Abstract. We show how the (Gödel's) Characterization Theorem and (Turing's) Undecidability of the Halting Set give important corollaries about the possibility of algorithmically deciding sentences true in the natural numbers (i.e., the first-order formal counterpart of what we usually call Number Theory).

Figure 1. Kurt Gödel and Alan Turing



1. Computably Enumerable Sets and Undecidability

The following definition is basically a repetition of the definition of semi-decidable set, which we gave in the context of Turing Machines.

Definition 1.1 (Computably Enumerable Sets)**.** A set is *computably enumerable* if is the domain of a computable function (with one argument).

A similar definition can be given for relations of arbitrary size. Synonyms of computably enumerable are: recursively enumerable (r.e.), algorithmically enumerable.

It can be easily proved that a set is enumerable if it is the image of a total computable function, i.e., if there is an algorithm defined on all natural that produces a list of all and only the elements of the set. (Exercise)

If a set S and its complement $\mathbf{N}\backslash S$ are both computably enumerable, then the set is decidable (or *recursive* or *computable*), i.e. there is a decision algorithm for the the membership problem for S. The viceversa also holds. (Exercise)

By the Characterization Theorem we have the following logical characterization of c.e. sets.

Corollary 1.2 (c.e.= Σ_1-definable)**.** *A set S is computably enumerable if and only if for some formula $F(x) \in \Sigma_1$ (with only one free variable x) for every $n \in \mathbf{N}$*

$$n \in S \iff \mathbf{N} \models F(x)[n].$$

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

Proof. Let S be c.e. Then S is the domain of a computable function φ. Let $F_\varphi(x,y)$ be a Σ_1-formula that defines in $\mathbf{N}$ the function φ. Then for each $n \in \mathbf{N}$ we have

$$n \in S \iff \mathbf{N} \models \exists y F_\varphi(x,y)[n].$$

The formula $\exists y F_\varphi(x,y)$ is obviously Σ_1.

Let $S = \{n \in \mathbf{N} : \mathbf{N} \models F[n]\}$, where $F(x)$ is Σ_1 with the only free variable x. We have to find a computable function whose domain is S. The formula

$$G(x,y) := ((y = 0) \wedge F(x))$$

is also Σ_1 and it defines a partial function. Consider indeed the set of ordered pairs

$$X_G = \{(a,b) : \mathbf{N} \models G(x,y)[a,b]\}.$$

This set is clearly functional, since it contains only pairs $(a,0)$, where a satisfies $F(x)$, that is

$$X_G = \{(a,0) : \mathbf{N} \models F(x)[a]\},$$

which by hypothesis coincides with

$$X_G = \{(a,0) : n \in S\}.$$

In other words, it's the graph of the following function:

$$\varphi(n) = \begin{cases} 0 \text{ if } n \in S, \\ \uparrow \text{ otherwise.} \end{cases}$$

Then, since φ is Σ_1-definable in $\mathbf{N}$, it is computable (by the Characterization Theorem). Obviously S is the domain of φ.

□

Remark 1.3. The fact that every enumerable collection can be expressed by a formula of type "exist x ..." is intuitively clear if we think of any characterization of computable functions in terms of programming systems (e.g., Turing machines, machines RAM, etc.). A set is enumerable if it is the set of inputs on which a certain program terminates. Therefore a set S is enumerable if for some program P, $x \in S$ if and only if **there is a time** t within which the computation of the program P on input x terminates. To make the argument rigorous we should encode in the language of arithmetic the syntactic constructs of the programming language in which P is written and prove that it is possible to express in the language of arithmetic formulas of low complexity (Δ_0) facts such as "The program P on input x terminates within t steps".

The previous corollary already gives you an idea of why the Characterization Theorem is useful to solve issues of decidability and undecidability. Let $(\varphi_n)_{n\in\mathbf{N}}$ be a programming system (model of computation) – for example the of all and only the (unary) functions in the class $\mathcal{C}$. The Diagonal Halting problemis the set

$$K = \{n : \varphi_n(n) \downarrow\}.$$

We know that K is **computably enumerable** but not **computable**. Its complement,

$$\overline{K} = \{n : \varphi_n(n) \uparrow\}$$

is **not computably enumerable** (otherwise K would be decidable!).

By the Corollary above, there is a Σ_1-formula, say $F(x)$, such that

$$n \in K \iff \mathbf{N} \models F[n].$$

Then also

$$n \in \bar{K} \iff \mathbf{N} \not\models F[n] \iff \mathbf{N} \models \neg F[n].$$

It should be noted that for each formula $G(x)$ with one free variable x and for every $a \in \mathbf{N}$, $\mathbf{N} \models G(x)[a]$ if and only if $\mathbf{N} \models G(\bar{a})$, where $\bar{a}$ is the cloed term in the language of arithmetic associated with the number a (i.e., $1+1+1\ldots$, a times). Therefore there is a Σ_1-formula $F(x)$ such that, for all $n \in \mathbf{N}$.

$$n \in K \iff \mathbf{N} \models F(\bar{n}),$$

and

$$n \in \bar{K} \iff \mathbf{N} \models \neg F(\bar{n}).$$

From the comments above we have the following Corollary showing that there is no algorithmic solution to the Decision Problem for truth of sentences in the language of arithmetic over the standard model (i.e. arithmetical truth is undecidable).

Corollary 1.4 (True Arithmetic is Undecidable)**.** *The theory* $Th(\mathbf{N}) = \{E : \mathbf{N} \models E\}$ *is algorithmically undecidable.*

The above important result is strictly connected to the famous Gödel's Incompleteness Theorem which was proved **without** assuming the existence of an algorithmically undecidable problems such as K.

What we observed above in fact allows us to give two more refined corollaries.

From the fact that $n \in K$ is equivalent to $\mathbf{N} \models F(\bar{n})$, where $F(x)$ is a fixed Σ_1-formula, we have the following corollary, showing that undecidability of arithmetic hits already at a low level of quantifier complexity.

Corollary 1.5 (True Σ_1-Arithmetic is Undecidable)**.** *The set of* Σ_1*-sentences true in* $\mathbf{N}$ *is not decidable.*

From the fact that $n \in \bar{K}$ is equivalent to $\mathbf{N} \not\models F(\bar{n})$, where $F(x)$ is a fixed Σ_1-formula, we have the following.

Corollary 1.6 (False Σ_1-Arithmetic is Undecidable)**.** *The set of* Σ_1*-sentences false in* $\mathbf{N}$ *is not computably enumerable.*

2. Descriptive Complexity of computable, c.e and co-c.e. sets.

Let F be a Σ_1-formula. Then F is of the form $\exists x_1 \dots \exists x_t G$ with G a Δ_0-formula.

Then $\neg F$ equivalent to $\forall x_1 \dots \forall x_t \neg G$ (the equivalence holds over all structures by logical laws of quantifiers, and in particular it holds over $\mathbf{N}$). Since G is a Δ_0-formula, also $\neg G$ is a Δ_0-formula. This leads us the definition of the following class of formulas.

Definition 2.1 (Π_1-formulas)**.** A formula of arithmetic is a Π_1-formula if it is Δ_0 or of the form $\forall x_1 \dots \forall x_t G$ with G a Δ_0-formula.

We are interested in the relations between the syntactic complexity of a formula defining a set and the computational complexity of the set itself. From the above facts and definitions we can gather a number of observations.

Definition 2.2. If $\mathcal{C}$ is a class of formulas (in the language of arithmetic), we say that a set S is $\mathcal{C}$-definable (over $\mathbf{N}$) if there exists a formula $F \in \mathcal{C}$ that defines S (over $\mathbf{N}$).

We will use ambiguously the name of the class to refer both to the class of formulas and to the class of sets definable by formulas in the class. Therefore, saying that a formula is Σ_1 means that it has a particular syntactical form, while saying that a set is Σ_1 means that there exists a Σ_1-formula that defines it.

Thus it makes sense *a priori* that a set S can be both Σ_1 and Δ_0, or both Σ_1 and Π_1.

From the definitions we have the following trivial inclusions for Δ_0-, Σ_1- and Π_1-definable sets:

$$\Delta_0 \subseteq \Sigma_1; \Delta_0 \subseteq \Pi_1.$$

Are these inclusions strict? And what is the relations between Σ_1 and Π_1?

Since K is **computablly enumerable** but **not computable**, we have that K is Σ_1 (Characterization Theorem) but not Δ_0 (the truth of Δ_0 formulas in $\mathbf{N}$ is computable).

Therefore the semantic classes are strictly separated:

$$\Sigma_1 \setminus \Delta_0 \neq \varnothing.$$

Since $\bar{K}$ is **not computably enumerable** (and *a fortiori* **not computable**), it is neither Δ_0 (else it would be computable) nor Σ_1 (else it would be computably enumerable).

From one of the above equivalences we have that $\bar{K}$ is Π_1. Therefore

$$\Pi_1 \setminus \Delta_0 \neq \varnothing$$

, and

$$\Pi_1 \setminus \Sigma_1 \neq \varnothing.$$

Moreover, since K is Σ_1 but is not computable, we can deduce that K is not Π_1. In fact, if it were, then $\bar{K}$ would be Σ_1 (by taking the negation of the Π_1-formula defining K) and therefore computably enumerable. So we also have

$$\Sigma_1 \setminus \Pi_1 \neq \varnothing.$$

$\bar{K}$ is the complement of a c.e. set (K). Let's call complements of c.e. sets **co-c.e.** sets. Just as for $\bar{K}$, it is easy to see that any co-c.e. set is Π_1 and viceversa.

Also, recalling that a set S is computable if and only if both S and $\bar{S}$ are c.e., we immediately deduce that the computable sets are exactly those in the intersection $\Pi_1 \cap \Sigma_1$. Since it coincides with computable sets, this class deserves a proper name: Δ_1. Obviously we have

$$\Pi_1 \setminus \Delta_1 \neq \varnothing,$$

and

$$\Sigma_1 \setminus \Delta_1 \neq \varnothing.$$

The classes Δ_1, Σ_1 and Π_1 are the first three levels of an infinite hierarchy of families of sets defined in terms of the complexity of their defining formula. This hiearchy is called the Arithmetical Hierarchy and we picture its first levels as follows, where the edges indicate proper inclusion, reading from the bottom up. The class Δ_0 defined above is properly included in Δ_1, although we will not prove it.

FIGURE 2. Arithmetical Hiearchy, initial levels