XX

Twentieth lecture
Automata & Formal Languages
20 November 2023

REMARK

$$\#v \neq |v|$$

Isomorphism between $(\mathbb{N}, <) \cong (\mathbb{N}, <)$

Length

Recap §4.7 Universality

Alphabet Σ; expanded to Σ' to allow encodings.

Lecture XIX page 8

Proposition The operation

$$w, v, v \longmapsto \text{code}(C(M, \vec{w}, \#v))$$

where $\text{code}(M) = w$, $\text{code}(\vec{w}) = v$

can be performed by a RM.

... terms of the TRANS-

Lecture XIX page 9

Corollary 4.25. The total operation "check whether M has halted with input $\vec{w}$ after at most $\#v$ steps" can be performed by a register machine. We call the corresponding total (characteristic) function

$$t_{M,k}(\vec{w}, v) := \begin{cases} a & M \text{ has halted with input } \vec{w} \text{ after at most } \#v \text{ steps and} \\ \varepsilon & \text{otherwise} \end{cases}$$

the *truncated computation of* M.

Lecture XIX, page 10

Theorem 4.26 (The Software Principle)**.** There is a register machine U, called a *universal register machine* such that for every register machine M and sequence of words $\vec{w}$, we have that

$$f_{U,2}(v,u) = \begin{cases} f_{M,k}(\vec{w}) & \text{if } v = \text{code}(M) \text{ for a register machine } M \\ & \quad \text{and } u = \text{code}(\vec{w}) \text{ for a sequence of words of length } k, \\ \uparrow & \text{otherwise.} \end{cases}$$

We discussed the fundamental importance of the Software Principle!

Proof. Take v and u as input.
Check whether $v = \text{code}(M)$ for some M
$u = \text{code}(\vec{w})$ for some $\vec{w}$.
If not $\uparrow$.
If so, use scratch registers k, l, m. (initially empty)

Repeat SUBROUTINE
until reg. l contains $\text{code}(q_{HT})$
After that, output content of reg. m.

SUBROUTINE
Compute $C(M, \vec{w}, v) = (q, \vec{v})$
where v is the content of reg. k
Write $\text{code}(q)$ into reg. l
Write v_0 into reg. m
Apply s to register k.

q.e.d.

Let's use this to improve notation:

Let U be a universal RM

Define $\quad v \in \mathbb{N} \quad \vec{w} \in \mathbb{N}^k$

$$f_{v,k}(\vec{w}) := f_{U,2}(v, \mathrm{code}(\vec{w})).$$

$$= f_{M,k}(\vec{w}) \quad \text{where } v = \mathrm{code}(M)$$

by Software principle

No need to bother with machines in the notation.

Note that if v is not a code for a RM, then $f_{v,k}$ ↑ everywhere.

Theorem 4.27 (The s-m-n Theorem). Let $g : \mathbb{W}^{k+1} \dashrightarrow \mathbb{W}$ be any partial computable function. Then there is a total computable function $h : \mathbb{W} \to \mathbb{W}$ such that for all $v \in \mathbb{W}$ and all $\vec{w} \in \mathbb{W}^k$, we have $f_{h(v),k}(\vec{w}) = g(\vec{w}, v)$.

if $g : \mathbb{W}^{k+1} \dashrightarrow \mathbb{W}$ computable, then there is total computable h s.t.

$$g(\vec{w}, v) = f_{h(v),k}(\vec{w}).$$

[The name derives from the original notation when it was proved.]

The theorem "pulls one variable into the index": known as CURRYING.

Proof. Note that $g_v : \vec{w} \longmapsto g(\vec{w}, v)$ is computable, so obviously there is some v s.t.

$$f_{v,k}(\vec{w}) = g(\vec{w}, v)$$

But s-m-n requires that this can be obtained by a total computable fn.

(1) M RM performing f, N RM performing g $\longrightarrow$ we had RM performing $f \circ g$

Let call it $M \circ N$

Then

$$(\text{code}(M), \text{code}(N)) \longmapsto \text{code}(M \circ N)$$

is a total computable fn.

(2) Fix v, the map

$$\vec{w} \longmapsto (\vec{w}, v)$$

is computable. This is performed by an explicit RM M_v.

Therefore $v \longmapsto \text{code}(M_v)$

is total computable.

(3) $v \xrightarrow{(2)} \text{code}(M_v) \xrightarrow{(2)} (\text{code}(M_v), \text{code}(M))$

Fix some RM M.

$$\xrightarrow{(1)} \text{code}(M \circ M_v)$$

Thus $h_M(v) := \text{code}(M \circ M_v)$

is a computable function (total).

(4) Let g be as in the theorem. Fix M performing g.

CLAIM h_M does the job.

$$f_{h_M(v),k}(\vec{w}) = f_{M \circ M_v, k}(\vec{w}) = g(\vec{w}, v)$$

q.e.d.

§4.8 Computably enumerable sets

Reminder If $X \subseteq \mathbb{N}^k$, then TFAE:

(i) X is c.e.

(ii) there is $f : \mathbb{N}^k \dashrightarrow \mathbb{N}$ computable s.t. $X = \mathrm{dom}(f)$

(iii) there is RM M s.t. $X = \mathrm{dom}(f_{M,k})$

(iv) there is $w \in \mathbb{N}$ s.t. $X = \mathrm{dom}(f_{w,k})$.

HALTING PROBLEM

Introduce two sets

$$K := \{ w : f_{w,1}(w)\downarrow \}$$

$$K_0 := \{ \langle w,v \rangle : f_{w,1}(v)\downarrow \}$$

two-variable HALTING PROBLEM version of the

Prop. K & K_0 are c.e.

Proof. By definition, $K_0 = \mathrm{dom}(f_{U,2})$ where U is the universal RM, so c.e.

Clearly, $w \longmapsto (w,w)$ can be performed by a RM, so $w \longmapsto f_{U,2}(w,w)$ is computable & its domain is K. q.ed.

Theorem

K & K_0 are not computable.

⟶ Limits of computation

230 A. M. TURING. [Nov. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable numbers, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.

Proof.

Suppose either χ_K or χ_{K_0} is computable,

then the fn

$$f(w) := \begin{cases} \uparrow & \text{if } \chi_{K_0}(w,w) = \chi_K(w) = a \\ \varepsilon & \text{if } \chi_{K_0}(w,w) = \chi_K(w) = \varepsilon. \end{cases}$$

is computable. Let $d \in W$ be s.t.

$$f(w) = f_{d,1}(w).$$

$f(d)\uparrow \iff \chi_K(d) = a \iff d \in K$

$\iff f_{d,1}(d)\downarrow$

$\iff f(d)\downarrow$

Contradiction!

q.e.d.

Corollary There are c.e. sets that are not computable.

Goal. Understand c.e. sets better.

By previous notation, we know

$X \subseteq \mathbb{N}$ $\quad X$ c.e. $\iff \exists v \;\; X = \mathrm{dom}(\varphi_{v,1})$

Write

$W_v := \mathrm{dom}(\varphi_{v,1})$

for

the c.e. set coded by v.

$\{W_v ; v \in \mathbb{N}\} = \{A \subseteq \mathbb{N} ; A \text{ is c.e.}\}$

Definition 4.30. A set $X \subseteq \mathbb{W}^k$ is called Σ_1 if there is a computable set $Y \subseteq \mathbb{W}^{k+1}$ such that for all $\vec{w} \in \mathbb{W}^k$, we have

$$\vec{w} \in X \iff \exists v (\vec{w}, v) \in Y.$$

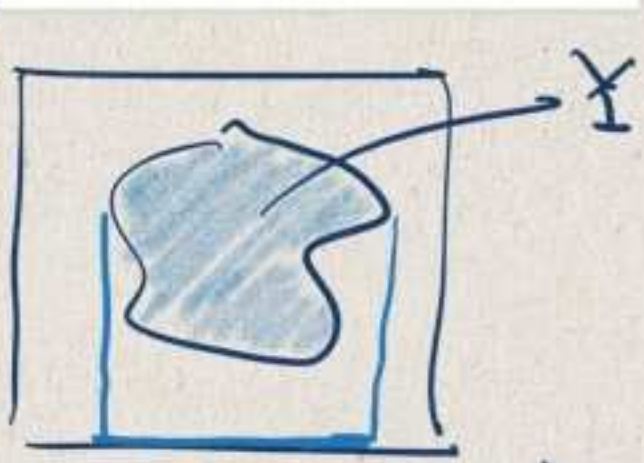
It is called Π_1 if it is the complement of a Σ_1 set; it is called Δ_1 if it is both Σ_1 and Π_1.

The terminology derives from the fact that Σ_1 sets are defined using one existential quantifier and logicians tend to think of existential quantifiers as analogues of sums; similarly, Π_1 sets are defined using one universal quantifier and logicians tens to think of these as analogies of products. The letter Δ comes from the German word "*Durchschnitt*" (intersection) since the class of Δ_1 sets is the intersection of the classes of Σ_1 and Π_1 sets.

Σ ⟷ "sum"

Π ⟷ "product"

Δ ⟷ intersection "Durchschnitt"



X is the projection of Y to the first k coordinates

Theorem If X is computable, then X is Δ_1.

Proof in Lecture XXI.