Mathematical Logic

Lecture 7

Harbin, 2023

Computable functions

Notation. We let $\mu x(-x-)$ denote the least $x \in \mathbb{N}$ for which $-x-$ holds. Here $-x-$ is some condition on natural numbers x. For example $\mu x(x^2 > 10) = 4$
For $a \in \mathbb{N}$ we also let $\mu x_{<a}(-x-)$ be the least $x < a$ in $\mathbb{N}$ such that $-x-$ holds if there is such an x, and if there is no such x we put $\mu x_{<a}(-x-) := a$. For example, $\mu x_{<3}(x^2 > 10) = 3$.

Definition 1. For $R \subseteq \mathbb{N}^n$, we define $\chi_R : \mathbb{N}^n \to \mathbb{N}$ by

$$\chi_R(a) = \begin{cases} 1, & \text{if } a \in R \\ 0, & \text{if } a \notin R \end{cases}$$

Think of such R as an n-ary relation on $\mathbb{N}$. We call χ_R the characteristic function of R, and often write $R(a_1, \ldots, a_n)$ instead of $(a_1, \ldots, a_n) \in R$.
For example, $\chi_<(m, n) = 1$ iff $m < n$, and $\chi_<(m, n) = 0$ iff $m \geq n$.

Definition 2. For $i = 1, \ldots, n$ we define $I_i^n : \mathbb{N}^n \to \mathbb{N}$ by $I_i^n(a_1, \ldots, a_n) = a_i$. These functions are called coordinate functions.

Definition 3. The computable functions (or recursive functions) are the functions from $\mathbb{N}^n$ to $\mathbb{N}$ (for $n = 0, 1, 2 \ldots$) obtained by inductively applying the following rules:

(R1) $+ : \mathbb{N}^2 \to \mathbb{N}$, $\cdot : \mathbb{N}^2 \to \mathbb{N}$ and the coordinate functions I_i^n are computable.

(R2) If $G : \mathbb{N}^m \to \mathbb{N}$ is computable and $H_1, \ldots, H_m : \mathbb{N}^n \to \mathbb{N}$ are computable, then so is the function $F = G(H_1, \ldots, H_m) : \mathbb{N}^n \to \mathbb{N}$ defined by

$$F(a) := G(H_1(a), \ldots, H_m(a))$$

(R3) If $G : \mathbb{N}^{n+1} \to \mathbb{N}$ is computable, and for all $a \in \mathbb{N}^n$ there exists $x \in \mathbb{N}$ such that $G(a, x) = 0$, then the function $F : \mathbb{N}^n \to \mathbb{N}$ given by

$$F(a) = \mu x(G(a, x) = 0)$$

is computable.

A relation $R \subseteq \mathbb{N}^n$ is said to be computable (or recursive) if its characteristic function $\chi_R : \mathbb{N}^n \to \mathbb{N}$ is computable.

Example. If $F : \mathbb{N}^3 \to \mathbb{N}$ and $G : \mathbb{N}^2 \to \mathbb{N}$ are computable, then so is the function $H : \mathbb{N}^4 \to \mathbb{N}$ defined by $H(x_1, x_2, x_3, x_4) = F(G(x_1, x_4), x_2, x_4)$. This follows from (R2) by noting that $H(x) = F(G(I_1^4(x), I_4^4(x)), I_2^4(x), I_4^4(x))$ where $x = (x_1, x_2, x_3, x_4)$.

Lemma 1

Let $H_1, \ldots, H_m : \mathbb{N}^n \to \mathbb{N}$ and $R \subseteq \mathbb{N}^m$ be computable. Then $R(H_1, \ldots, H_m) \subseteq \mathbb{N}^n$ is computable, where for $a \in \mathbb{N}^n$ we put $R(H_1, \ldots, H_m)(a) \Leftrightarrow R(H_1(a), \ldots, H_m(a))$

Proof. Observe that $\chi_{R(H_1,\ldots,H_m)} = \chi_R(H_1, \ldots, H_m)$. Now apply (R2).■

Lemma 2

The functions $\chi_{\geq}$ and $\chi_{=}$ on $\mathbb{N}^2$ are computable.

Proof. The function $\chi_{\geq}$ is computable because $\chi_{\geq}(m,n) = \chi_{\leq}(n,m) = \chi_{\leq}(I_2^2(m,n), I_1^2(m,n))$ which enables us to apply (R1) and (R2). Similarly, $\chi_{=}$ is computable: $\chi_{=}(m,n) = \chi_{\geq}(m,n) \cdot \chi_{\leq}(m,n)$ ■

For $k \in \mathbb{N}$ we define the constant function $c_k^n : \mathbb{N}^n \to \mathbb{N}$ by $c_k^n(a) = a$.

Lemma 3

Every constant function c_k^n is computable.

Proof. By induction on k. For $k = 0$ we use $c_0^n(a) = \mu x(I_{n+1}^{n+1}(a,x) = 0)$. For the step from k to $k+1$, observe that

$$c_{k+1}^n(a) = \mu x(c_k^n(a) < x) = \mu x(\chi_{\geq}(c_k^{n+1}(a,x), I_{n+1}^{n+1}(a,x)) = 0)$$

for $a \in \mathbb{N}^n$ ■

Let P, Q be $n-$ary relations on $\mathbb{N}$. Then we can form the $n-$ary relations

$$\neg P := \mathbb{N}^n \setminus P, P \vee Q := P \cup Q, P \wedge Q := P \cap Q$$

$$P \to Q := \neg P \vee Q, P \leftrightarrow Q := (P \to Q) \wedge (Q \to P)$$

Lemma 4

Suppose P, Q are computable. Then $\neg P, P \vee Q, P \wedge Q, P \to Q, P \leftrightarrow Q$ are also computable.

Proof. Let $a \in \mathbb{N}^n$. Then $\neg P(a)$ iff $\chi_P(a) = 0$ iff $\chi_P(a) = c_0^n(a)$, so $\chi_{\neg P}(a) = \chi_=(\chi_P(a), c_0^n(a))$. Hence $\neg P$ is computable by (R2) and Lemma 2. Next, the relation $P \wedge Q$ is computable since $\chi_{P \wedge Q} = \chi_P \cdot \chi_Q$. By De Morgan's Law, $P \vee Q = \neg(\neg P \wedge \neg Q)$. Thus, $P \vee Q$ is computable. The rest follows from the equalities above.■

Lemma 5

The binary relations $<, \leq, =, >, \geq, \neq$ on $\mathbb{N}$ are computable.

Proof. The relations $\leq, \geq$ and $=$ have already been taken care of by Lemma 2 and (R1). The remaining relations are complements of these three, so by Lemma 4 they are also computable. ■

Lemma 6

Let $R_1, \ldots, R_k \subseteq \mathbb{N}^n$ be computable such that for each $a \in \mathbb{N}^n$ exactly one of $R_1(a), \ldots, R_k(a)$ holds, and suppose that $G_1, \ldots, G_k : \mathbb{N}^n \to \mathbb{N}$ are computable. Then $G : \mathbb{N}^n \to \mathbb{N}$ given by

$$G(a) = \begin{cases} G_1(a) & \text{if } R_1(a) \\ \cdots\cdots\cdots & \\ G_k(a) & \text{if } R_k(a) \end{cases}$$ *is computable.*

Proof. This follows from $G = G_1 \cdot \chi_{R_1} + \ldots + G_k \cdot \chi_{R_k}$ ■

Lemma 7

Let $R_1, \ldots, R_k \subseteq \mathbb{N}^n$ be computable such that for each $a \in \mathbb{N}^n$ exactly one of $R_1(a), \ldots, R_k(a)$ holds. Let $P_1, \ldots, P_k \subseteq \mathbb{N}^n$ be computable. Then the relation $P \subseteq \mathbb{N}^n$ defined by

$$P(a) \Leftrightarrow \begin{cases} P_1(a) & \text{if } R_1(a) \\ \ldots\ldots\ldots & \\ P_k(a) & \text{if } R_k(a) \end{cases} \quad \textit{is computable.}$$

Proof. Use that $P = (P_1 \wedge R_1) \vee \ldots \vee (P_k \wedge R_k)$. ■

Lemma 8

Let $R \subseteq \mathbb{N}^{n+1}$ be computable such that for all $a \in \mathbb{N}^n$ there exists $x \in \mathbb{N}$ with $(a, x) \in R$. Then the function $F : \mathbb{N}^n \to \mathbb{N}$ given by $F(a) = \mu x(R(a, x))$ is computable.

Proof. Note that $F(a) = \mu x(\chi_{\neg R}(a, x) = 0)$ and apply (R3). ■

Lemma 9

Let $F : \mathbb{N}^n \to \mathbb{N}$. Then F is computable if and only if its graph is computable.

Proof. Let $R \subseteq \mathbb{N}^{n+1}$ be the graph of F. Then for all $a \in \mathbb{N}^n$ and $b \in \mathbb{N}$, $R(a, b) \Leftrightarrow F(a) = b, \quad F(a) = \mu x R(a, x)$, from which the lemma follows immediately. ■

Lemma 10

If $R \subseteq \mathbb{N}^{n+1}$ is computable, then the function $F_R : \mathbb{N}^{n+1} \to mathbbN$ defined by $F_R(a, y) = \mu x_{<y} R(a, x)$ is computable.

Proof. Use that $F_R(a, y) = \mu x(R(a, x)$ or $x = y)$. ■

Lemma 11

Suppose $R \subseteq \mathbb{N}^{n+1}$ is computable. Let $P, Q \subseteq \mathbb{N}^{n+1}$ be the relations defined by

$$P(a, y) \iff \exists x_{<y} R(a, x)$$

$$Q(a, y) \iff \forall x_{<y} R(a, x)$$

for $(a, y) = (a_1, \ldots, a_n, y) \in \mathbb{N}^{n+1}$. Then P and Q are computable.

Proof. Note that $P(a, y)$ iff $F_R(a, y) < y$. Hence $\chi_P(a, y) = \chi_<(F_R(a, y), y)$. For Q, note that $\neg Q(a, y)$ iff $\exists x_{<y} \neg R(a, x)$.

Lemma 12

The function $\dot{-} : \mathbb{N}^2 \to \mathbb{N}$ defined by

$$a \dot{-} b = \begin{cases} a - b & \text{if } a \geq b \\ 0, & \text{if } a < b \end{cases} \quad \textit{is computable}$$

Proof. Use that $a \dot{-} b = \mu x(b + x = a \text{ or } a < b)$. ■

Definition 4. Define the function $Pair : \mathbb{N}^2 \to \mathbb{N}$ by

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

We call Pair the pairing function.

Lemma 13

The function Pair is bijective and computable.

Proof. Easy exercise.

Definition 5. Since Pair is a bijection we can define functions

$$Left, Right : \mathbb{N} \to \mathbb{N}$$

by

$$Pair(x, y) = a \iff Left(a) = x \text{ and } Right(a) = y$$

Note that $Left(a), Right(a) \leq a$ for $a \in \mathbb{N}$, and $Left(a) < a$ if $0 < a \in \mathbb{N}$.

Lemma 14

The functions Left and Right are computable.

Proof. Use Lemma 9 in combination with

$$Left(a) = \mu x(\exists y_{<a+1} Pair(x, y) = a)$$

$$Right(a) = \mu y(\exists x_{<a+1} Pair(x, y) = a) \blacksquare$$

Lemma 15

The relation $a \equiv b(mod\ c)$ on $\mathbb{N}$ is computable.

Proof. Use that for $a, b, c \in \mathbb{N}$ we have
$a \equiv b(mod\ c) \iff (\exists x_{<a+1} a = x \cdot c + b \text{ or } \exists x_{<b+1} b = x \cdot c + a) \blacksquare$

We can now introduce Gödel's function $\beta : \mathbb{N}^2 \to \mathbb{N}$

Definition 6. For $a, i \in \mathbb{N}$ we let $\beta(a, i)$ be the remainder of $Left(a)$ upon division by $1 + (1 + i)Right(a)$, that is,

$$\beta(a, i) := \mu x(x \equiv Left(a) \ mod \ 1 + (i + 1)Right(a))$$

Proposition 16

The function β is computable, and $\beta(a, i) \leq a \boxminus 1$ for all $a, i \in \mathbb{N}$. For any $a_0, \ldots, a_n \in \mathbb{N}$ there exists $a \in \mathbb{N}$ such that $\beta(a, 0) = a_0, \ldots, \beta(a, n) = a_n$

Proof. The computability of β is clear from earlier results. We have

$$\beta(a, i) \leq Left(a) \leq a \boxminus 1$$

Let $a_0, \ldots, a_n \in \mathbb{N}$. Take $N \in \mathbb{N}$ such that $a_i \leq N$ for all $i \leq n$ and N is a multiple of every prime number $\leq n$.

We claim that then

$$1 + N, 1 + 2N, \ldots, 1 + nN, 1 + (n + 1)N$$

are pairwise relatively prime. To see this, suppose p is a prime number such that $p|1 + iN$ and $p|1 + jN$ $(1 \leq i < j \leq n + 1)$; then p divides their difference $(j - i)N$, but $p \equiv 1(mod\ N)$, so p does not divide N, hence $p|j - i \leq n$. But all prime numbers $\leq n$ divide N, and we have a contradiction.
By the Chinese Remainder Theorem there exists an $M \in \mathbb{N}$ such that

$$M \equiv a_0\ mod\ 1 + N$$

$$M \equiv a_1\ mod\ 1 + 2N$$

$$\ldots\ldots\ldots$$

$$M \equiv a_n\ mod\ 1 + (n + 1)N$$

Put $a := Pair(M, N)$; then $Left(a) = M$ and $Right(a) = N$, and thus $\beta(a, i) = a_i$ as required. ■

Remark. Proposition 16 shows that we can use β to encode a sequence of numbers $a_0, \ldots, a_n$ in terms of a single number a. We use this as follows to show that the function $n \mapsto 2^n$ is computable.
If $a_0, \ldots, a_n$ are natural numbers such that $a_0 = 1$ and $a_{i+1} = 2a_i$ for all $i < n$, then necessarily $a_n = 2^n$. Hence by Proposition 16 we have $\beta(a, n) = 2^n$, where

$$a := \mu x(\beta(x, 0) = 1 \text{ and } \forall i_{<n}\beta(x, i + 1) = 2\beta(x, i)),$$

that is,

$$2^n = \beta(a, n) = \beta(\mu x(\beta(x, 0) = 1 \text{ and } \forall i_{<n}\beta(x, i + 1) = 2\beta(x, i)), n)$$

It follows that $n \mapsto 2^n$ is computable.

The above suggests a general method, which we develop next. To each sequence $(a_1, \ldots, a_n)$ of natural numbers we assign a sequence number, denoted $\langle a_1, \ldots, a_n \rangle$, and defined to be the least natural number a such that $\beta(a, 0) = n$ (the length of the sequence) and $\beta(a, i) = a_i$ for $i = 1, \ldots, n$. For $n = 0$ this gives $\langle\rangle = 0$, where $\langle\rangle$ is the sequence number of the empty sequence. We define the length function $lh : \mathbb{N} \to \mathbb{N}$ by $lh(a) = \beta(a, 0)$, so lh is computable. Observe that $lh(\langle a_1, \ldots, a_n \rangle) = n$.

Put $(a)_i := \beta(a, i + 1)$. The function $(a, i) \mapsto (a)_i : \mathbb{N}^2 \to \mathbb{N}$ is computable, and $(\langle a_1, \ldots, a_n \rangle)_i = a_{i+1}$ for $i < n$. Finally, let $Seq \subseteq \mathbb{N}$ denote the set of sequence numbers. The set Seq is computable since

$$a \in Seq \iff \forall x_{<a}(lh(x) \neq lh(a) \text{ or } \exists i_{<lh(a)}(x)_i \neq (a)_i)$$

Lemma 17

For any n, the function $(a_1, \ldots, a_n) \mapsto \langle a_1, \ldots, a_n \rangle : \mathbb{N}^n \to \mathbb{N}$ is computable, and $a_i < \langle a_1, \ldots, a_n \rangle$ for $(a_1, \ldots, a_n) \in \mathbb{N}^n$ and $i = 1, \ldots, n$.

Proof. Use $\langle a_1, \ldots, a_n \rangle = \mu a(\beta(a, 0) = n, \beta(a, 1) = a_1, \ldots, \beta(a, n) = a_n)$, and apply Lemmas 8, 4 and 15. ■

Lemma 18

We have computable binary operations $\mathit{In} : \mathbb{N}^2 \to \mathbb{N}$ and $\star : \mathbb{N}^2 \to \mathbb{N}$ such that for all $a_1, \ldots, a_m, b_1, \ldots, b_n \in \mathbb{N}$,

$$\mathit{In}(\langle a_1, \ldots, a_m \rangle, i) = \langle a_1, \ldots, a_i \rangle \text{ for } i \leq m$$

$$\langle a_1, \ldots, a_m \rangle \star \langle b_1, \ldots, b_n \rangle = \langle a_1, \ldots, a_m, b_1, \ldots, b_n \rangle$$

Proof. Such functions are obtained by defining

$$\mathit{In}(a, i) = \mu x(\mathit{lh}(x) = i \text{ and } \forall j_{<i}(x)_j = (a)_j),$$

$$a \star b = \mu x(lh(x) = lh(a) + lh(b) \text{ and } \forall i_{<lh(a)}(x)_i = (a)_i$$

$$\text{and } \forall j_{<lh(b)}(x)_{lh(a)+j} = (b)_j) \blacksquare$$

Definition 7. For $F : \mathbb{N}^{n+1} \to \mathbb{N}$, let $\bar{F} : \mathbb{N}^{n+1} \to \mathbb{N}$ be given by

$$\bar{F}(a, b) = \langle F(a, 0), \ldots, F(a, b-1) \rangle \quad (a \in \mathbb{N}^n, b \in \mathbb{N}).$$

Note that $\bar{F}(a, 0) = \langle\rangle = 0$.

Lemma 19

Let $F : \mathbb{N}^{n+1} \to \mathbb{N}$. Then F is computable if and only if $\bar{F}$ is computable.

Proof. Suppose F is computable. Then $\bar{F}$ is computable since

$$\bar{F}(a, b) = \mu x(lh(x) = b \text{ and } \forall i_{<b}(x)_i = F(a, i)).$$

In the other direction, suppose $\bar{F}$ is computable. Then F is computable since $F(a, b) = (\bar{F}(a, b+1))_b$ $\blacksquare$

Given $G : \mathbb{N}^{n+2} \to \mathbb{N}$ there is a unique function $F : \mathbb{N}^{n+1} \to \mathbb{N}$ such that

$$F(a, b) = G(a, b, \bar{F}(a, b)) \quad (a \in \mathbb{N}^n, b \in \mathbb{N}).$$

This will be clear if we express the requirement on F as follows:

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

$$F(a, b+1) = G(a, b+1, \langle F(a, 0), \ldots, F(a, b)\rangle).$$

The next result is important because it allows us to introduce computable functions by recursion on its values at smaller arguments.

Proposition 20

Let G and F be as above and suppose G is computable. Then F is computable.

Proof. Note that

$$\bar{F}(a,b) = \mu x(Seq(x) \text{ and } lh(x) = b \text{ and } \forall i_{<b}(x)_i = G(a,i,In(x,i)))$$

for all $a \in \mathbb{N}^n, b \in \mathbb{N}$. It follows that $\bar{F}$ is computable, and thus by the previous lemma F is computable. ■

Definition 8. Let $A : \mathbb{N}^n \to \mathbb{N}$ and $B : \mathbb{N}^{n+2} \to \mathbb{N}$ be given. Let a range over $\mathbb{N}^n$, and define the function $F : \mathbb{N}^{n+1} \to \mathbb{N}$ by

$$F(a,0) = A(a),$$

$$F(a,b+1) = B(a,b,F(a,b)).$$

We say that F is obtained from A and B by primitive recursion.

Proposition 21

Suppose A, B, and F are as above, and A and B are computable. Then F is computable.

Proof. Define $G : \mathbb{N}^{n+2} \to \mathbb{N}$ by

$$G(a,b,c) = \begin{cases} A(a), & \text{if } b = 0 \\ B(a, b-1, (c)_{b-1}), & \text{if } b > 0 \end{cases}$$

Clearly, G is computable. We claim that $F(a,b) = G(a,b,\bar{F}(a,b))$. This claim yields the computability of F, by Proposition 20. We have

$$F(a,0) = A(a) = G(a,0,\bar{F}(a,0)), \text{ and}$$

$$F(a,b+1) = B(a,b,F(a,b)) = B(a,b,(\bar{F}(a,b+1))_b) = G(a,b+1,\bar{F}(a,b+1)$$

■

Example. Let $G : \mathbb{N} \to \mathbb{N}$ and $H : \mathbb{N}^2 \to \mathbb{N}$ be computable. There is clearly a unique function $F : \mathbb{N}^2 \to \mathbb{N}$ such that for all $a, b \in \mathbb{N}$

$$F(a,b) = \begin{cases} F(a, G(b)), & \text{if } G(b) < b \\ H(a,b), & \text{otherwise} \end{cases}$$

In particular $F(a,0) = H(a,0)$. We claim that F is computable.

According to Proposition 20 this claim will follow if we can specify a computable function $K : \mathbb{N}^3 \to \mathbb{N}$ such that $F(a,b) = K(a,b,\bar{F}(a,b))$ for all $a, b \in \mathbb{N}$. Such a function K is given by

$$K(a,b,c) = \begin{cases} (c)_{G(b)}, & \text{if } G(b) < b \\ H(a,b), & \text{otherwise} \end{cases}$$

The Church-Turing Thesis

The computable functions as defined in the last section are also computable in the informal sense that for each such function $F : \mathbb{N}^n \to \mathbb{N}$ there is an algorithm that on any input $a \in \mathbb{N}$ stops after a finite number of steps and produces an output $F(a)$. An algorithm is given by a finite list of instructions, a computer program, say. These instructions should be deterministic (leave nothing to chance or choice). We deliberately neglect physical constraints of space and time: imagine that the program that implements the algorithm has unlimited access to time and space to do its work on any given input.

Let us write "calculable"for this intuitive, informal, idealized notion of computable. The Church-Turing Thesis asserts

each calculable function $F : \mathbb{N} \to \mathbb{N}$ is computable.

Call a set $P \subseteq \mathbb{N}$ calculable if its characteristic function is calculable.

While the Church-Turing Thesis is not a precise mathematical statement, it is an important guiding principle, and has never failed in practice: any function that any competent person has ever recognized as being calculable, has turned out to be computable, and the informal grounds for calculability have always translated routinely into an actual proof of computability. Here is a heuristic (informal) argument that might make the Thesis plausible.

Let an algorithm be given for computing $F : \mathbb{N} \to \mathbb{N}$. We can assume that on any input $a \in \mathbb{N}$ this algorithm consists of a finite sequence of steps, numbered from 0 to n, say, where at each step i it produces a natural number a_i, with $a_0 = a$ as starting number. It stops after step n with $a_n = F(a)$.

We assume that for each $i < n$ the number a_{i+1} is calculated by some fixed procedure from the earlier numbers $a_0, \ldots, a_i$, that is, we have a calculable function $G : \mathbb{N} \to \mathbb{N}$ such that $a_{i+1} = G(\langle a_0, \ldots, a_i \rangle)$ for all $i < n$.

The algorithm should also tell us when to stop, that is, we should have a calculable $P \subseteq \mathbb{N}$ such that $\neg P(\langle a_0, \ldots, a_i \rangle)$ for $i < n$ and $P(\langle a_0, \ldots, a_i \rangle)$.

Since G and P describe only single steps in the algorithm for F it is reasonable to assume that they at least are computable. Once this is agreed to, one can show easily that F is computable as well (Exercise!).

The above is only a rather narrow version of the Church-Turing Thesis, but it suffices for our purpose. There are various refiniments and more ambitious versions. Also, our Church-Turing Thesis does not characterize mathematically the intuitive notion of algorithm, only the intuitive notion of function computable by an algorithm that produces for each input from $\mathbb{N}$ an output in $\mathbb{N}$.