2/12/25 GSO

Daniel Rui

Case: "computers can do arithmetic; but also, arithmetic can compute."

0.1 Exhibit A: origins I (Herbrand-Gödel)

(Soare) Dedekind 1888 "proved, using accepted axioms, that such a definition defines a unique function, and he applied it to the definition of the functions $m+n, m\times n$, and m^n. Based on this work of Dedekind, Peano 1889 and 1891 wrote the familiar five axioms for the positive integers. As a companion to his fifth axiom, mathematical induction, Peano used definition by induction, which has been called primitive recursion".

(Andrew) Historically, several mathematicians in the early 1930s (Church, Gödel, Herbrand, and others) gave what we now recognize to be the correct definition of computability (Herbrand-Gödel/partial recursive class). However, they were unable to give a convincing argument that their definition included all possible algorithms.

The smallest collection of functions closed under items 1-5 are called the primitive recursive functions. They were first defined by Gödel in his 1931 paper on incompleteness.

(Soare) Gödel realized, however, that the primitive recursive functions did not include all effectively calculable functions (Ackermann in 1928 had produced a function defined by double recursion which was not primitive recursive), and in 1934 Gödel proposed a wider class of functions based on an earlier suggestion of Herbrand (letter from 1931). Gödel called these the general recursive functions.

Definition 0.1: Partial recursive functions

The set of **partial recursive functions** is the smallest set of (partial) functions $\mathbb{N}^*\to\mathbb{N}$ containing the **initial functions**:

- (zero) $\mathbf{0}:=[x\mapsto 0]:\mathbb{N}\to\mathbb{N}$,
- (successor) $\textcircled{+1}:=[x\mapsto x+1]:\mathbb{N}\to\mathbb{N}$,
- (coordinate projections) $\mathrm{Proj}_i^k:=[(x_1,\dots,x_k)\mapsto x_i]:\mathbb{N}^k\to\mathbb{N}$

and closed under the following 3 operations:

- (composition)
$$\left.\begin{matrix} f:\mathbb{N}^k\to\mathbb{N} \\ g_1,\dots,g_k:\mathbb{N}^l\to\mathbb{N}\end{matrix}\right\}\rightsquigarrow [x\mapsto f(g_1(x),\dots,g_k(x))]:\mathbb{N}^l\to\mathbb{N}$$
- (primitive recursion)
$$\left.\begin{matrix} f_0:\mathbb{N}^k\to\mathbb{N} \\ r:\mathbb{N}^{k+2}\to\mathbb{N}\end{matrix}\right\}\rightsquigarrow f:\mathbb{N}^{k+1}\to\mathbb{N},\left[\begin{matrix} f(0,x):=f_0(x) \\ f(i+1,x):=r(i,f(i,x),x)\end{matrix}\right]$$
- (minimization/μ-recursion, denoted $f:=\mu i.g(i,x)$)
$$g:\mathbb{N}^{k+1}\to\mathbb{N}\rightsquigarrow f:\mathbb{N}^k\to\mathbb{N},\left[f(x):=\min\left\{i:\begin{matrix} g(i,x)=0\wedge \\ g([1..i),x)\exists\end{matrix}\right\}\right]$$

Single-variable partial recursive functions $\mathbb{N}\to\mathbb{N}$ can recover the whole set, by the use of Cantor's pairing function $\mathsf{pair}:=[(x,y)\mapsto\frac{1}{2}(x+y)(x+y+1)+y]:\mathbb{N}\times\mathbb{N}\to\mathbb{N}$ (the amazingly simple (polynomial!) formula for the (inverse of the) standard zigzag pattern mapping $\mathbb{N}\to\mathbb{N}^2$) and its inverse $(\mathsf{1st},\mathsf{2nd})(\vec{z}):=\left(\mu x_{\leqslant|\vec{z}|}[(\exists y\leqslant|\vec{z}|)(z=\mathsf{pair}(x,y))],\quad \mu y_{\leqslant|\vec{z}|}[(\exists x\leqslant|\vec{z}|)(z=\mathsf{pair}(x,y))]\right)$

0.2 Exhibit B: origins II (Turing)

(Andrew) In 1936, Alan Turing suggested a new definition of computability via what are called now Turing machines. Turing's model leads to the same definition as the Herbrand-Gödel recursive functions (see Exercise 2.8). However, what was important about Turing's model is that it included a convincing philosophical argument that it encompassed all possible algorithms.

Definition 0.2: Turing machine (intuitively)

A (deterministic) TM ("idealized agent executing a (deterministic) algorithm") should have:

- an unlimited supply of time and scratch paper; and
- one agent who can [look at]/read/write/erase (only) one memory cell/piece of paper [at a time]/[during each single clock tick]; and
- only finitely (bounded) many possible symbols Σ to write in each memory cell/paper; and
- only finitely many mental thoughts/states/memories; and
- the ability to [know/remember the path to] and [move to] (only) finitely (bounded) many "neighboring" memory cells/papers in each clock tick.

- Can always reduce symbol set Σ to $\{0, 1, _\}$ by grouping together big blocks of these binary digits (+ space) encode each "super-symbol".
- Can always reduce m-tape TM to single-tape by expanding the symbol set to Σ^m (+ underlined symbols to denote where "head" is on that tape).
- Can reduce moving abilities to just {left, right}, by additionally mentally keeping track of how many steps you are on your way to the desired "neighbor" cell, e.g. replacing states $\mathtt{q} \in \mathtt{Q}$ with $(\mathtt{q}, \text{"I'm on step } i \text{ on the path to destination } d\text{"})$.

Definition 0.3: Turing machine (more rigorously)

The **(intrinsic) data of a TM** consists of:

- $\mathtt{Q}$ (a finite set of "mental states"), which contain 2 distinguished states $\mathtt{q}_0$ (start state) and $\mathtt{q}_{\mathtt{halt}}$ (halt state),
- a symbol set $\Sigma := \{0, 1, _\}$,
- and a transition function $t : (\mathtt{Q} \times \Sigma) \to (\Sigma \times \{\leftarrow, \rightarrow\} \times \mathtt{Q})$ (let us conflate $\overleftrightarrows$ with ± 1) (where $(\mathtt{q}, i) \mapsto (i', \pm 1, \mathtt{q}')$ read as "in mental state $\mathtt{q}$ and read i (at current position/cell), overwrite current cell to symbol i', move left/right, and adopt mental state $\mathtt{q}'$"

An **execution/tape configuration of a TM** consists of:

- a clock starting at 0, going up by 1 every clock tick/cycle
- the tape at time i, $\mathsf{T}_i : \mathbb{Z} \to \Sigma$,
- the head at time i, $\blacktriangledown_i \in \mathbb{Z}$ (referring to a current position on the tape T)
- the state $\mathtt{q}_\mathtt{i}$ at time i. A **configuration** consists of the latter 3 data: $(\mathtt{q}, \mathsf{T}, \blacktriangledown)$.

s.t. at time 0, we start with state $\mathtt{q}_0$ and $\blacktriangledown_0 = 0$, and for all times $i \geqslant 0$,

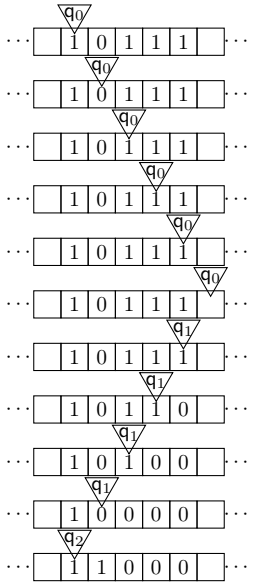
$$t(\mathtt{q}_\mathtt{i}, \quad \mathsf{T}_i(\blacktriangledown_i)) = (\mathsf{T}_{i+1}(\blacktriangledown_i), \quad (\blacktriangledown_{i+1} - \blacktriangledown_i), \quad \mathtt{q}_{\mathtt{i+1}}).$$

Spaces allow us to consider multiple inputs, but as with partial recursive functions, can always encode away this annoyance, so let's focus on single input TMs, and symbol space simplified to $\Sigma := \{0, 1\}$.

0.3 Exhibit C: example of TM

Theorem 0.1: TM computable $\iff$ partial recursive

A (partial) function $f : \mathbb{N} \to \mathbb{N}$ is Turing computable $\iff$ it is partial recursive



$$\begin{aligned}
\checkmark\mathsf{step}(c, c') := \checkmark\mathsf{config}(c) \quad &\wedge \quad \checkmark\mathsf{config}(c') \quad \wedge \\
\mathsf{state}(c) = 1 \quad &\wedge \quad \mathsf{state}(c') = 2 \quad \wedge \\
2 \cdot \mathsf{left}(c') = \mathsf{left}(c) \quad &\wedge \\
\mathsf{right}(c') = 2 \cdot \mathsf{right}(c) + 1 &
\end{aligned}$$

See https://www.cmi.ac.in/~madhavan/courses/pl2024/lectures/RecursiveFunctionsAndTMs.pdf#page=8 for more formula gore.

0.4 Exhibit D: connection to arithmetical truth

Logical symbols (common to all languages, hence often left unwritten), all distinct:

- parentheses (and).
- the set of (pairwise distinct) variables $\mathscr{V} := \{\mathsf{v}_n : n \in \mathbb{N}\}$ (not dependent on $\mathscr{L}$, so as opposed to like $\mathscr{C}^{\mathscr{L}}$, only write as $\mathscr{V}$)
- the equality symbol =
- logical connectives: negation ¬, conjunction ∧
- the existential quantifier ∃.

$$\mathscr{L}_{\text{arith}} := [\text{logical part}] \cup \{ \underset{\in \mathscr{C}}{\mathsf{0}}, \underset{\in \mathscr{F}_1}{\mathsf{S}}, \underbrace{\mathsf{+},\cdot}_{\in \mathscr{F}_2}, \underset{\in \mathscr{R}_2}{<} \}$$

Lemma 0.1: Gödel β encoding, reducing any sequence to data of just 2 numbers

There exists $\beta^{b,c}(i)$ from $\mathbb{N}^3 \to \mathbb{N}$ s.t. for any sequence $a_0, \ldots, a_n$ of natural numbers, there exist $b, c \in \mathbb{N}$ s.t. $\beta_{b,c}(i) = a_i$ for $i = 0, \ldots, n$.

0.4.1 Example: (primitive recursion)

$$\begin{bmatrix} f(0,x) := f_0(x) \\ f(i+1,x) := r(i, f(i,x), x) \end{bmatrix}$$

(Andrew) Now note that $f(j, x_1, \ldots, x_k) = y$ if and only if there is a sequence $a_0, \ldots, a_j$ so that $a_0 = f_0(\vec{x})$, and for every $i < j$, $r(i, a_i, \vec{x}) = a_{i+1}$, and $y = a_j$. So using Gödel's β lemma to represent this sequence $a_0, \ldots, a_j$, we can represent f by the formula $\varphi_f(j, \vec{x}; y)$ (the goal is this formula is true $\iff$ $f(j, x_1, \ldots, x_k) = y$)

$$\begin{aligned} (\exists b, c)\Big[& (\exists \mathsf{a}_0 < b)\big[\beta^{b,c}(0) = \mathsf{a}_0 \quad \wedge \quad \varphi_{f_0}(\vec{x}; \mathsf{a}_0)\big] \quad \wedge \\ & (\forall i < j)(\exists \mathsf{a} < b)(\exists \mathsf{a}_+ < b)\big[\beta^{b,c}(i) = \mathsf{a} \quad \wedge \quad \beta^{b,c}(i+1) = \mathsf{a}_+ \quad \wedge \quad \varphi_r(i, \mathsf{a}, \vec{x}; \mathsf{a}_+)\big] \\ & \wedge \beta^{b,c}(j) = y \Big] \end{aligned}$$

See https://math.berkeley.edu/~marks/notes/computability_notes_v1.pdf#page=46 for more formula gore (in particular **bounded quantifiers!**).

Definition 0.4: Σ_1 sentences

A sentence (in the language $\mathscr{L}_{\text{arith}}$; no free variables) is called Σ_1 if it is of the form $\exists \vec{x} \psi(\vec{x})$ where $\psi(\vec{x})$ is a formula containing only **bounded quantifiers**.

Theorem 0.2: Partial recursive functions reduce to Σ_1 sentences

For every partial computable function $f : \mathbb{N}^k \to \mathbb{N}$, there is a Σ_1 formula $\psi_f(\vec{x}; y)$ s.t. $f(\vec{x}) = y$ iff $\psi_f(\vec{x}; y)$ is true (i.e. a true statement about natural numbers).
Furthermore, if one has a natural number y that does indeed equal $= f(\vec{x})$, Arithmetic can prove this: $\mathsf{PA}^- \vdash \psi_f(\overline{x_1}, \ldots, \overline{x_k}; \overline{y})$ and also $\mathsf{PA}^- \vdash \forall \mathsf{y}'\big[\psi_f(\overline{x_1}, \ldots, \overline{x_k}; \overline{y}) \to \mathsf{y}' = \overline{y}\big]$.
The previous paragraph is the definition of f **representable in** PA^-. This theorem essentially says a function f is partial computable $\iff$ it is representable in PA^-.

(Andrew) Definition 10.1 Peano Arithmetic or PA is the following theory in the language of arithmetic. First, the theory contains the axioms of the positive part of a discrete ordered semiring. We call these following 16 axioms PA^-:

1. $(\forall x, y, z)((x + y) + z = x + (y + z))$.
2. $(\forall x, y)(x + y = y + x)$.
3. $(\forall x, y, z)((x \cdot y) \cdot z = x \cdot (y \cdot z))$.
4. $(\forall x, y)((x \cdot y) = (y \cdot x))$.
5. $(\forall x, y, z)(x \cdot (y + z) = x \cdot y + x \cdot z)$.
6. $(\forall x)(x + 0 = x)$.
7. $(\forall x)(x \cdot 0 = 0)$.
8. $(\forall x)(x \cdot 1 = x)$.
9. $(\forall x, y, z)(x < y \wedge y < z \to x < z)$.
10. $(\forall x)(-x < x)$.
11. $(\forall x, y, z)(x < y \vee y < x \vee x = y)$.
12. $(\forall x, y, z)(x < y \to x + z < y + z)$.
13. $(\forall x, y, z)(0 < z \wedge x < y \to x \cdot z < y \cdot z)$.
14. $(\forall x, y)(x < y \to (\exists z)x + z = y)$.
15. $0 < 1 \wedge (\forall x)[x > 0 \to x \geqslant 1)]$.
16. $(\forall x)(x \geqslant 0)$.

Second, the theory contains an induction axiom for each formula φ :

$$\varphi(0) \wedge \forall k[\varphi(k) \to \varphi(k + 1)] \to \forall n\varphi(n) \qquad (\mathrm{Ind}_\varphi)$$

So $\mathsf{PA} = \mathsf{PA}^- + \bigcup_\varphi \{\mathrm{Ind}_\varphi\}$.
For each $n \in \mathbb{N}$, we use the overline notation $\overline{n}$ to denote the **formal syntax** `1+...+1` (n many `1`'s).