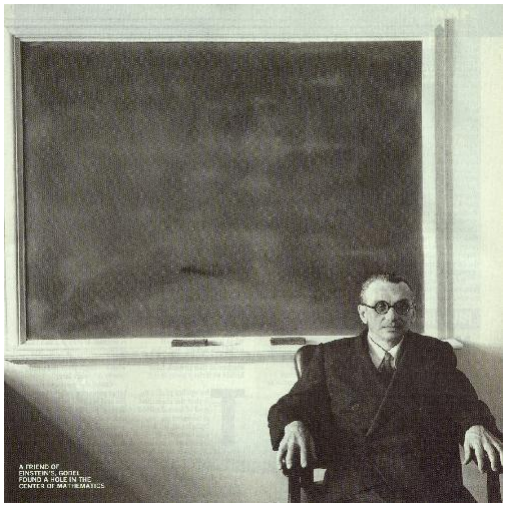
Gödel and recursivity

Prof. J. Duparc
TEXed by Cédric Ho Thanh

Spring 2013



Contents

1 Introduction **5**

2 Towards Turing machines **7**
2.1 Finite automatons 7
2.2 Counter automatons and pushdown automatons 11
2.3 Turing machines 13

3 From Turing machines to recursive functions **17**
3.1 Primitive recursive functions 17
3.2 Coding sequences of integers 22
3.3 Recursive functions 22

4 Arithmatical hierarchy **25**
4.1 Σ_n^0, Π_n^0 and Δ_n^0 25
4.2 Turing machine with oracle 27

5 Arithmetisation of the first order logic and Gödels first incompleteness theorem **29**
5.1 Representable functions 31
5.2 Arithmetization of syntax 33
5.3 Coding proofs 37

Chapter 1

Introduction

1.0.1 Definition (Goodstein sequence)**.** Let $m \in \mathbb{N}$ be an integer from which we define the **Goodstein sequence** $(G^m_{(i)})_{i\in\mathbb{N}}$. For instance, if $m = 4$:

- $G^4_{(0)}$: write 4 in hereditary base 2 : $G^4_{(0)} = 2^2 = 4$,
- $G^4_{(1)}$: replace base 2 by base 3 and substract 1 : $G^4_{(1)} = 3^3 - 1 = 2\times 3^2 + 2\times 3^1 + 2\times 3^0 = 26$,
- $G^4_{(2)}$: increase the base and substract 1 : $G^4_{(2)} = 2\times 4^2 + 2\times 4^1 + 2\times 4^0 - 1 = 41$,
- and so on...

Elements of G^4 continue to increase until $G^4_{(3\times 2^{402653209})}$ where it reaches $3\times 2^{402653210} - 1$, it stays there for the next 3×2^{402653209} steps and then starts to decrease.

1.0.2 Theorem (Goodstein, 1944)**.** For any $m \in \mathbb{N}$

$$\lim_{i\to\infty} G^m_{(i)} = 0.$$

1.0.3 Theorem (Kiny, Paris, 19820)**.** The Goodstein theorem cannot be proved in Peano aritmetic.

Chapter 2

Towards Turing machines

2.1 Finite automatons

2.1.1 Definition (Deterministic finite automaton, DFA)**.** **(DFA)** $\mathfrak{A}$ is defined by :

- A **deterministic finite automaton**

$$\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$$

where

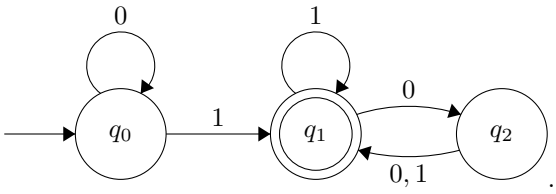
1. Q is a finite set of **states** ,
2. Σ is a nonempty set called the alphabet,
3. $\delta : Q \times \Sigma \longrightarrow Q$ is a function,
4. $q_0 \in Q$ is the **initial state**
5. $F \subseteq Q$ is the set of **accepting states** .

- The automaton $\mathfrak{A}$ **accepts** a word $w \in \Sigma^{<\omega}$ iff :
  - $\exists a_0, \ldots, a_n \in \Sigma$ such that $w = a_0 \cdots a_n$,
  - $\exists r_0, \ldots, r_{n+1} \in Q$ such that $r_0 = q_0$, $\delta(r_i, a_i) = r_{i+1}, \quad \forall i \leq n$,
  - $r_{n+1} \in F$.
- The langage accepted by $\mathfrak{A}$ is defined by

$$\mathcal{L}(\mathfrak{A}) = \left\{ w \in \Sigma^{<\omega} \mid \mathfrak{A} \text{ accepts } w \right\}.$$

2.1.2 Examples.

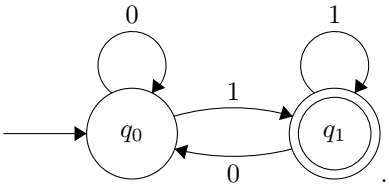
- Let $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0, 1\}$, $F = \{q_1\}$ and $\mathfrak{A}$ be defined by :



.

Then, $\mathfrak{A}$ accepts 0101 but rejects 010.

- Let $Q = \{q_0, q_1\}$, $\Sigma = \{0, 1\}$, $F = \{q_1\}$ and $\mathfrak{B}$ be defined by :



  Then, $\mathcal{L}(\mathfrak{B}) = \{w \in \Sigma^{<\omega} \mid w \text{ ends with a } 1\}$.

- Let $\mathcal{L} = \{w \in \{0,1\}^{<\omega} \mid w \text{ contains as many 0's as 1's}\}$. There is no DFA $\mathfrak{C}$ such that $\mathcal{L}(\mathfrak{C}) = \mathcal{L}$.

2.1.3 Definition (Regular langage)**.** A langage is **regular** if it is recognized by a DFA (i.e. all of its words are accepted by this DFA). The class of all regural langages is called Reg .

2.1.4 Notation. Let Σ be an alphabet, let ε stands for the empty word and $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$.

2.1.5 Definition (Nondeterministic finite automaton, NFA)**.**

- A **non deterministic finite automaton (NFA)** $\mathfrak{A}$ is defined by :

$$\mathfrak{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$$

  where

  1. Q is a finite set of **states** ,
  2. Σ is a nonempty set called the alphabet,
  3. $\delta : Q \times \Sigma \longrightarrow \mathcal{P}(Q)$ is a function,
  4. $q_0 \in Q$ is the **initial state**
  5. $F \subseteq Q$ is the set of **accepting states** .

- The automaton $\mathfrak{A}$ **accepts** a word $w \in \Sigma^{<\omega}$ iff
  - $\exists a_0, \ldots, a_n \in \Sigma$ such that $w = a_0 \cdots a_n$,
  - $\exists r_0, \ldots, r_{n+1} \in Q$ such that $r_0 = q_0$, $r_{i+1} \in \delta(r_i, a_i), \quad \forall i \leq n$,
  - $r_{n+1} \in F$.

- The langage accepted by $\mathfrak{A}$ is defined by

$$\mathcal{L}(\mathfrak{A}) = \left\{w \in \Sigma^{<\omega} \mid \mathfrak{A} \text{ accepts } w\right\}.$$

2.1.6 Theorem. For every NFA, there exists an equivalent DFA, i.e. the two automata recognize the same langage.

Proof. Let $\mathfrak{N} = \langle Q, \Sigma, \delta, q_0, F \rangle$ be a NFA. Suppose that $\mathfrak{N}$ has no ε-transition. Then define the DFA

$$\mathfrak{D} = \langle \mathcal{P}(Q), \Sigma, \delta', \{q_0\}, F' \rangle$$

where

- $\delta'(R, a) = \bigcup_{r \in R} \delta(r, a) = \{q \in Q \mid \exists r \in R \text{ such that } q \in \delta(r, a)\}$,
- $F' = \{R \in \mathcal{P}(Q) \mid R \cap F \neq \emptyset\}$.

If $\mathfrak{N}$ has ε-transitions, then define

$$\mathrm{Eps}(R) = \{q \in Q \mid q \text{ can be reached from } R \text{ by traveling along } \varepsilon\text{-transition(s)}\}.$$

Then define $\mathfrak{D}$ as above with the following changes :

- the initial state is $\mathrm{Eps}(\{q_0\})$,
- $\delta'(R, a) = \mathrm{Eps}\left(\bigcup_{r \in R} \delta(r, a)\right)$.

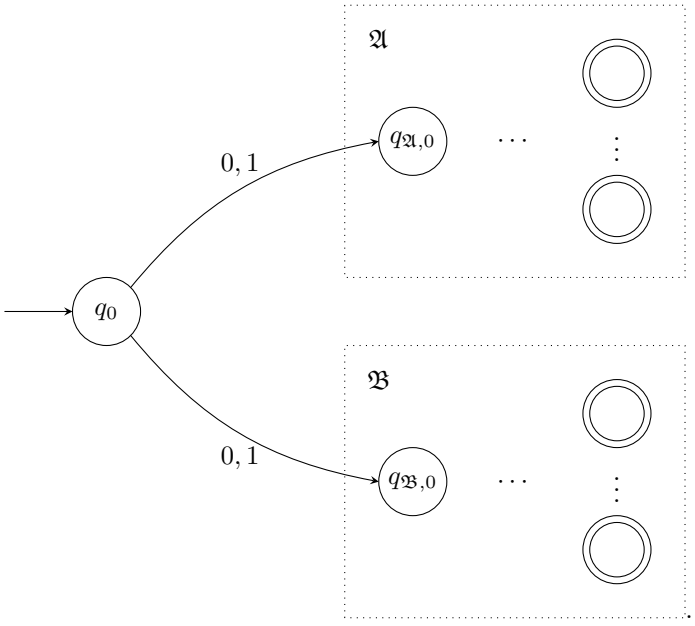
□

2.1.7 Theorem. The class Reg is closed under the following operations :

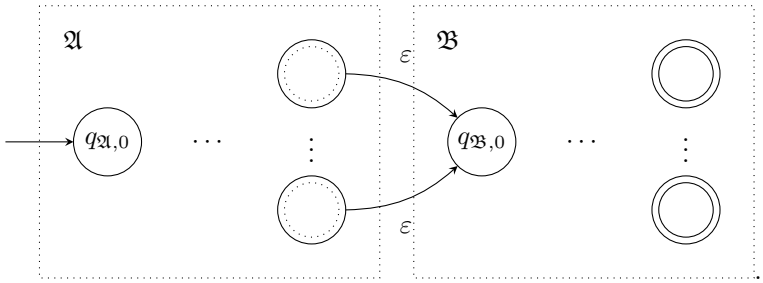
1. Union : $\mathcal{L}_A \cup \mathcal{L}_B = \{w \in \Sigma^{<\omega} \mid w \in \mathcal{L}_A \text{ or } w \in \mathcal{L}_B\}$,
2. Concatenation : $\mathcal{L}_A \cdot \mathcal{L}_B = \{w \in \Sigma^{<\omega} \mid \exists u \in \mathcal{L}_A, \exists v \in \mathcal{L}_B \text{ such that } w = uv\}$,
3. Star operation : $\mathcal{L}_A^* = \{w \in \Sigma^{<\omega} \mid \exists k \in \mathbb{N}, \exists u_1, \ldots, u_k \in \mathcal{L}_A \text{ such that } w = u_1 \cdots u_k\} \cup \{\varepsilon\}$.

Proof. Let $\mathfrak{A}$ and $\mathfrak{B}$ be two DFAs such that $\mathcal{L}(\mathfrak{A}) = \mathcal{L}_A$ and $\mathcal{L}(\mathfrak{B}) = \mathcal{L}_B$.

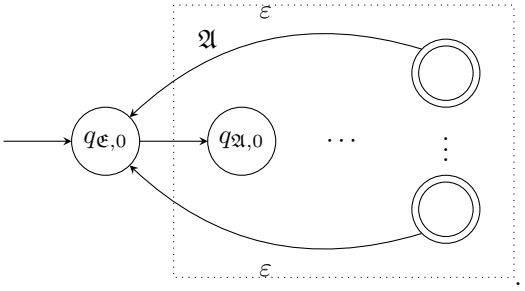
1. Define $\mathfrak{C}$ by



2. Define $\mathfrak{D}$ by :



3. Define $\mathfrak{E}$ by



□

2.1.8 Definition (Regular expression)**.** R is a **regular expression** if either

- $\emptyset$,
- a letter $a \in \Sigma$,
- ε,
- $R_1 \cup R_2$, where R_1 and R_2 are regular expressions,
- $R_1 \cdot R_2$, where R_1 and R_2 are regular expressions,
- R_1^* where R_1 is a regular expression.

We define $\mathcal{L}(R)$ as follow :

- $\mathcal{L}(\emptyset) = \emptyset$,
- $\mathcal{L}(a) = a$ for all $a \in \Sigma$,
- $\mathcal{L}(\varepsilon) = \{\varepsilon\}$,
- $\mathcal{L}(R_1 \cup R_2) = \mathcal{L}(R_1) \cup \mathcal{L}(R_2)$,
- $\mathcal{L}(R_1 \cdot R_2) = \mathcal{L}(R_1) \cdot \mathcal{L}(R_2)$,
- $\mathcal{L}(R_1^*) = \mathcal{L}(R_1)^*$.

2.1.9 Theorem. A langage $\mathcal{L}$ is regular iff there exists some regular expression R such that $\mathcal{L}(R) = \mathcal{L}$.

Proof. Exercise. □

Fix Σ a finite alphabet. Hom many regular langages does it admit ? Let $\mathfrak{D} = \langle Q, \Sigma, \delta, q_0, F\rangle$ be a DFA with $Q = \{q_0, \ldots, q_n\}$, $\Sigma = \{a_1, \ldots, a_k\}$, $F = \{q_{i_1}, \ldots, q_{i_m}\}$. We code Q by $n_Q = n + 1$, Σ by $n_\Sigma = k$, q_0 by $n_{q_0} = 0$, F by

$$n_F = \prod_{j=1}^{m} p_j^{i_j}, \qquad \text{where } \{p_1, \ldots\} = \mathbb{P}.$$

To code δ, remind that $\delta : Q \times \Sigma \longrightarrow Q$ can be viewed as a subset $\delta \subset Q \times \Sigma \times Q$. A triple (q_i, a_j, q_l) can be coded by $n_{(q_i,a_j,q_l)} = 2^i 3^j 5^l$. So δ can be coded by

$$n_\delta = \prod_{(q_i,a_j,q_l)\in\delta} p_{n_{(q_i,a_j,q_l)}}.$$

A DFA is then coded by a sequence of 5 integers, which can itself be coded as an integer. We have an injection from the set of all the DFAs built on Σ to $\mathbb{N}$.

2.1.10 Corollary. There are countably many regular langages.

Hom many langages does Σ admit ? If Σ is finite, then $\operatorname{Card}\Sigma^{<\omega} = \aleph_0$ and $\operatorname{Card}\mathcal{P}(\Sigma^{<\omega}) = \operatorname{Card}\mathbb{R}$. So most of the langages are not regular.

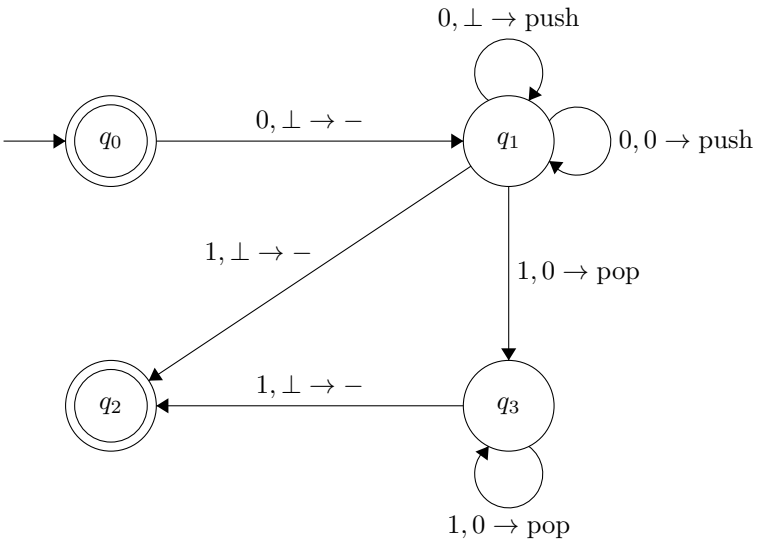
2.2 Counter automatons and pushdown automatons

2.2.1 Definition (Counter automaton)**.** A **counter automaton** is the same as a DFA or a NFA to which we add a **counter** (or **stack**) :

$$\mathfrak{C} = \langle Q, \Sigma, \delta, q_0, F, \bot, 0\rangle$$

- $\bot \neq 0$,
- the automaton cas push or pop the letter 0 inside the stack,
- the automaton can read the top of the stack,
- $\bot$ at the bottom of the stack and cannot be pushed nor popped,
- $\delta : Q \times \Sigma \times \{\bot, 0\} \longrightarrow Q \times \{\text{push}, \text{pop}, -\}$ if the automaton is deterministic,
- $\delta : Q \times \Sigma \times \{\bot, 0\} \longrightarrow \mathcal{P}(Q \times \{\text{push}, \text{pop}, -\})$ is the automaton is non deterministic.

2.2.2 Example. Consider



in addition that for any missing edge, the machine goes to a "trash" state which is not accepting and from which it can't escape. This CDFA recognises $\mathcal{L}(\mathfrak{C}) = \{0^n 1^n \mid n \in \mathbb{N}\}$, which is not regular.

2.2.3 Definition (Pushdown automaton)**.** A **pushdown automaton** is given by

$$\mathfrak{P} = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$$

where

- Γ is a finite alphabet called the **stack alphabet** ,
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow Q \times \Gamma_\varepsilon$ for a deterministic automaton,
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$ for a non deterministic automaton.

The automaton $\mathfrak{P}$ computes as follow : a word $w = w_1 \cdots w_n$ is accepted iff $\exists r_0, \ldots, r_n \in Q$, $\exists s_0, \ldots, s_n \in \Gamma^{<\omega}$ such that

1. $r_0 = q_0$,
2. $s_0 = \varepsilon$,
3. $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$, $\forall i \leq n-1$, where $s_i = at$, $s_{i+1} = bt$, for some $a, b \in \Gamma_\varepsilon$ (t is the begining of the stack and a its last letter which is replaced by b),
4. $r_n \in F$.

2.2.4 Definition (Context free grammar)**.** A **context free grammar** $G = \langle V, \Sigma, R, S \rangle$ is defined by :

1. V is a finite set of **variables** ,
2. Σ is a finite set of **terminal** such that $\Sigma \cap V = \emptyset$,
3. R is a finite set of **rules** ,
4. $S \in V$ is the start variable.

The langage it generates is the set of all words that can be derived from it.

2.2.5 Example. Consider the following context free grammar : $V = \{A, B\}$, $\Sigma = \{0, 1, \sharp\}$, $S = A$ and the rules are :

$$\begin{aligned} A &\longrightarrow 0A1 \\ A &\longrightarrow 0B1 \\ B &\longrightarrow \sharp. \end{aligned}$$

The generated langage is $\{0^n \sharp 1^n \mid n \geq 1\}$.

2.2.6 Theorem. A langage is recognized by a NPDA iff it is context free, i.e. generated by a CFG.

Le langage $\mathcal{L} = \{a^i b^j k \mid i = j \text{ or } i = k\}$. is recognized by a NPFA but not by a DPFA. $\updownarrow' = \{a^i b^i c^i \mid i \in \mathbb{N}\}$ is not recognized by a NPFA but easily recognized by a Turing machine.

2.3 Turing machines

2.3.1 Definition (Deterministic Turing machine)**.** A **deterministic Turing machine** (TM or DTM) is a 7-tuple

$$\mathfrak{T} = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}} \rangle$$

where

1. Q is a finite set of **states** , or **control state** ,
2. Σ is a finite alphabet, not containing the **blank symbol** $\sqcup$,
3. Γ is the **working alphabet** , where $\Sigma \subset \Gamma$ and $\sqcup \in \Gamma$,
4. $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$ is the **transition function** ,
5. $q_0 \in Q$ is the **initial state** ,
6. $q_{\text{accept}} \in Q$ is the **accepting state** ,
7. $q_{\text{reject}} \in Q$ is the **rejecting state** .

If the head of the machine is at the left-hand end of the tape, it cannot move left. If δ says so, it stays put. A **configuration** of a TM is a snapshot :

$$1011q_701111$$

means that

- the current control state is q_7,
- the word written is 101101111,
- the position of the head is

$$\begin{array}{cccccccccc} & & & & q_7 & & & & & \\ & & & & \downarrow & & & & & \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & \end{array} .$$

The initial configuration on input $w \in \Sigma^*$ is q_0w. A halting configuration is a configuration which state is either q_{accept} or q_{reject}. Given $a, b, c \in \Gamma$, $u, v \in \Gamma^*$, $q_i, q_j \in Q$, we say that

- uaq_ibv yields to uq_jacv if $\delta(q_i, b) = (q_j, c, L)$,
- uaq_ibv yields to $uacq_jv$ if $\delta(q_i, b) = (q_j, c, R)$.

The TM accepts the input w if there is a sequence of configurations $c_1, \ldots, c_k$ such that

- $c_1 = q_0w$,
- c_i yields to c_{i+1},
- c_k is an accepting configuration.

The set of words acepted by a TM $\mathfrak{T}$ is

$$\mathcal{L}(\mathfrak{T}) = \{w \in \Sigma^* \mid \mathfrak{T} \text{ accepts } w\} .$$

2.3.2 Definition (Turing-recognizable, Turing-decidable)**.** A langage $\mathcal{L}$ is :

- **Turing-recognizable** if there exist a TM $\mathfrak{T}$ such that $\mathcal{L} = \mathcal{L}(\mathfrak{T})$,
- **Turing-decidable** if there exists a **decider** , i.e. a TM that halts on all inputs and that recognizes $\mathcal{L}$.

Hilbert's 10th problen (1900, Paris) : " Given a Diaphantine equation with any number of unknown quantities and with rational integral numerical coefficients, devise a process according to which it can be determined in a finie number of operations wether the equation is solvable in rational integers ".

Proving that such an algorithm exists requires a formal definition. The Church-Turing thesis says that the intuitive notion of an algorithm is captured by the notion of a Turing machine.

Yuri Matyasevic (1970) : There is no decider of $\{P \mid P \text{ is a polynomial with integral root}\}$. Therefor, Hilbert's 10th problem is undecidable. It is recognizable though.

2.3.3 Proposition. Given a TM

- there exist an equivalent TM with a bi-infinite tape,
- there exists an equivalent multitape TM,
- there exists an equivalent multitape TM with bi-infinite tapes.

2.3.4 Definition (k-tape Turing machine)**.** A **k-tape Turing machine** is the same as a TM exept that it has k tapes, k independent heads and

$$\delta : Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{L, R\}^k.$$

2.3.5 Theorem. Avery multitape TM has an equivalent 1-tape TM.

2.3.6 Definition (Non deterministic Turing machine)**.** A **nondeterministic Turing machine** (NTM) is the same as a deterministic TM exept

$$\delta : Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\}).$$

2.3.7 Theorem. Every NTM has an equivalent DTM.

Proof. We build a 3-tapes DTM $\mathfrak{D}$ that simulates a NTM $\mathfrak{N}$. The tapes are :

1. the input tape : contains the input word w,
2. the simulation tape : Empty at the begening,
3. the adress tape : empty at the begining.

First, $\mathfrak{D}$ copies the input w from tape 1 to tape 2. Then, tape 3 uses tape 2 to simulate $\mathfrak{N}$ with input w on one branch of its non deterministic computation. For this, consider

$$n = \max \{\text{Card}\, \delta(q, \gamma) \mid (q, \gamma) \in Q \times \Gamma\}.$$

We consider the infinitary n-ary tree of non deterministic choices. For each $(q, \gamma) \in Q \times \Gamma$, we fix a linear ordering of transitions $\delta(q, \gamma)$ (this ordering has length at most n). The idea is to try all possible branch, not by a depth first search but in a breadth first search. Before each step of $\mathfrak{N}$, $\mathfrak{D}$ consults the next symbol on tape 3 to determine which choice to make among the allowed $\mathfrak{N}$'s transitions. If the k-th choice is invalid, $\mathfrak{D}$ aborts this branch and goes to the next one. If an accepting state is encountered, it accepts. □

2.3.8 Definition (Enumerator)**.** An **enumerator** of a langage $\mathcal{L}$ is a TM $\mathfrak{E}$ such that the result of its (possibly infinite) computation is

$$w_0 \sqcup w_1 \sqcup w_2 \sqcup \cdots ,$$

where $\mathcal{L} = \{w_n\}_{n\in\mathbb{N}}$.

2.3.9 Definition (Recursively enumerable langage)**.** A langage is **recursively enumerable** if there is an enumerator that enumerates it.

2.3.10 Theorem. A langage is Turing-recognizable if and only if it is recursively enumerable.

Proof. $\Rightarrow$: From $\mathfrak{M}$ a TM that recognizes $\mathcal{L} = \mathcal{L}(\mathfrak{M})$, we build $\mathfrak{E}$ that enumerates $\mathcal{L}$:

- fix a recursive enumeration of $\Sigma^* = \{s_i\}_{i\in\mathbb{N}}$,
- for $i = 1, 2, \ldots$, repeat the following : run $\mathfrak{M}$ for i steps successively on inputs $s_0, \ldots, s_i$. If any computation accepts, print the corresponding input s_k.

$\Leftarrow$: From $\mathfrak{E}$ an enumerator, build $\mathfrak{M}$ as follow : on input w, run $\mathfrak{E}$ and every time it prints something, compare it to w.

□

2.3.11 Exercise. Let Σ be an ordered alphabet and consider the lexicographical ordering of Σ^*. Prove that a langage is decidable iff there exists an enumerator that prints out $\mathcal{L}$ in this ordering.

Consider all DTM with fixed alphabets $\Sigma = \{0, 1\}$ and $\Gamma = \{0, 1, \sqcup\}$. Any such Turung machine is of the form

$$\mathfrak{M} = \langle \{q_0, \ldots, q_k\}, \{0, 1\}, \{0, 1, \sqcup\}, \delta, q_0, q_i = q_i, q_j \rangle$$

where δ is just a set of 5-tuples. The description of $\mathfrak{M}$ is a finite sequence on a finite alphabet

$$\mathcal{A} = \Big\{ \rangle, \langle, \_, 0, \ldots, k, \sqcup, \}, \{,), (\Big\}.$$

Get a recursive coding of $\mathcal{A}^* \longrightarrow \Sigma^*$ (i.e. an injective mapping that can be computed by a TM). For instance, if you have $\operatorname{Card} \mathcal{A} < 2^n$ for a $n \in \mathbb{N}$, code any letter of $\mathcal{A}$ by a sequence of n many 0's and 1's. Let $\lceil w \rceil$ be the code of w. The langage

$$\{\lceil \mathfrak{M} \rceil \mid \mathfrak{M} \text{ is a TM with } \Sigma = \{0, 1\}, \Gamma = \{0, 1, \sqcup\}\}$$

is decidable.

2.3.12 Definition (Universal Turung machine)**.** A **universal Turing machine** $\mathfrak{U}$ on alphabets $\Sigma_{\mathfrak{U}} = \{0, 1\}$, $\Gamma_{\mathfrak{U}} = \{0, 1, \sqcup\}$ is a TM such that on each input of the form $\langle \lceil \mathfrak{M} \rceil, w \rangle$, $\mathfrak{U}$ works as $\mathfrak{M}$ on w.

2.3.13 Theorem. There exists a universal Turing machine.

Proof. Construct the following 4-tapes TM :

1. input w,
2. w copied from tape 1,
3. the transition relation of $\mathfrak{M}$,
4. the sequence $\lceil q_0 \rceil \lceil q_{\text{accept}} \rceil \lceil q_{\text{reject}} \rceil$.

□

2.3.14 Theorem. The langage $\mathcal{A}_{\text{TM}} = \{\langle \lceil \mathfrak{M} \rceil, w\rangle \mid \mathfrak{M} \text{ accepts } w\}$ is recognizable but not decidable.

Proof. The fact that $\mathcal{A}_{\text{TM}}$ is recognizable is obvious. Suppose it is decidable by $\mathfrak{D}$. Build a TM $\mathfrak{H}$ tht works the following way on input w :

- $\mathfrak{H}$ never stops on w iff $\mathfrak{D}$ accepts $\langle w, w\rangle$,
- $\mathfrak{H}$ accepts w iff $\mathfrak{D}$ rejects $\langle w, w\rangle$.

One has that

$$\begin{aligned} \mathfrak{H} \text{ accepts } \lceil \mathfrak{H} \rceil &\iff \mathfrak{D} \text{ rejects } \langle \lceil \mathfrak{H} \rceil, \lceil \mathfrak{H} \rceil \rangle \\ &\iff \mathfrak{H} \text{ does not accept } \lceil \mathfrak{H} \rceil. \end{aligned}$$

□

The same king of proof shows that

$$\{\langle \lceil \mathfrak{M} \rceil, w\rangle \mid \mathfrak{M} \text{ stops on } w\}$$

is not decidable.

We have that

$$\text{Reg} \subsetneq \text{CFG} \subsetneq \text{Turing-decidable} \subsetneq \text{Turing recognizable}.$$

Chapter 3

From Turing machines to recursive functions

3.1 Primitive recursive functions

3.1.1 Definitions. 1. The i**-th projection** P_p^i is the function

$$\begin{aligned} P_p^i : \mathbb{N}^p &\longrightarrow \mathbb{N} \\ (x_1, \ldots, x_p) &\longmapsto x_i. \end{aligned}$$

2. The **successor function** is noted $s : \mathbb{N} \longrightarrow \mathbb{N}$.

3. If $f_1, \ldots, f_n : \mathbb{N}^p \longrightarrow \mathbb{N}$ and $g : \mathbb{N}^n \longrightarrow \mathbb{N}$, then the **composition** $g(f_1, \ldots, f_n)$ is defined as

$$\begin{aligned} g(f_1, \ldots, f_n) : \mathbb{N}^p &\longrightarrow \mathbb{N} \\ (x_1, \ldots, x_p) &\longmapsto g(f_1(x_1, \ldots, x_p), \ldots, f_n(x_1, \ldots, x_p)). \end{aligned}$$

4. Given $g : \mathbb{N}^p \longrightarrow \mathbb{N}$ and $h : \mathbb{N}^{p+2} \longrightarrow \mathbb{N}$, there exist a unique $f : \mathbb{N}^{p+1} \longrightarrow \mathbb{N}$ such that $\forall \vec{x} \in \mathbb{N}^p$, $\forall y \in \mathbb{N}$ we have :

$$\begin{aligned} f(\vec{x}, 0) &= g(\vec{x}) && \text{Initial condition} \\ f(\vec{x}, y+1) = f(\vec{x}, s(y)) &= h(\vec{x}, y, f(\vec{x}, y)) && \text{Induction step.} \end{aligned}$$

We say that f is **inductively defined** by g.

3.1.2 Definition (Primitive recursive function)**.** The set of all **primitive recursive functions** (PR functions) is the set that contains

1. all constants functions $\mathbb{N}^p \longrightarrow \mathbb{N}$,
2. all projections $P_p^i : \mathbb{N}^p \longrightarrow \mathbb{N}$,
3. the successor function $s : \mathbb{N} \longrightarrow \mathbb{N}$,

and that is closed under composition and induction. In other words, let R_0 be the set of all constants functions, projections and the successor function,

$$R_{n+1} = \{\text{functions built by composition and induction of functions of } R_n\},$$

$$\mathrm{PR} = \bigcup_{n\in\mathbb{N}} R_n.$$

3.1.3 Example. • The addition $a : (x, y) \longmapsto x + y$ is primitive recursive :

$$a(x, 0) = P_1^1(x, 0)$$
$$a(x, y + 1) = a(x, s(y)) = s \circ P_3^3(x, y, a(x, y)).$$

- Multiplication is PR :

$$x \cdot 0 = 0$$
$$x \cdot (y + 1) = x \cdot y + x.$$

- Exponentiation is PR :

$$x^0 = 1$$
$$x^{y+1} = x^y \cdot x.$$

- Define

$$x\dot{-}y = \begin{cases} x - y & \text{if } x - y \in \mathbb{N} \\ 0 & \text{otherwise.} \end{cases}$$

It is a PR function :

$$0\dot{-}1 = 0$$
$$(x + 1)\dot{-}1 = x,$$
$$x\dot{-}0 = x$$
$$x\dot{-}(y + 1) = (x\dot{-}y)\dot{-}1.$$

3.1.4 Definition (Primitive recursive set)**.** A subset $A \subseteq \mathbb{N}^p$ is **primitive recursive** if it's characteristic function χ_A is a PR function.

3.1.5 Example. The set $<= \{(x, y) \in \mathbb{N} \mid x < y\}$ is PR. Indeed,

$$\chi_<(x, y) = 1\dot{-}(1\dot{-}(y\dot{-}x)).$$

3.1.6 Definition (Partial function, total function)**.** We say that (D_f, f) is a **partial function** $\mathbb{N}^p \longrightarrow \mathbb{N}$ if f is a mapping $D_f \longrightarrow \mathbb{N}$, where $D_f \subset \mathbb{N}$. It is **total** if $D_f = \mathbb{N}$.

3.1.7 Definition (Turing computable function)**.** A partial function $f : D_f \longrightarrow \mathbb{N}$ with $D_f \subset \mathbb{N}^p$ is **Turing computable** if there exist a TM $\mathfrak{M}$ such that on the input $(x_1, \ldots, x_p)$,

- if $f(\vec{x})$ is not defined (i.e. $\vec{x} \notin D_f$), then $\mathfrak{M}$ never stops,
- if $\vec{x} \in D_f$, then $\mathfrak{M}$ stops with $f(\vec{x})$ written on the tape.

3.1.8 Proposition. A partial function f is Turing computable iff its graph

$$G_f = \{(\vec{x}, f(\vec{x})) \mid \vec{x} \in D_f\}$$

is Turing recognizable.

Proof. $\Rightarrow$: From a machine that computes f, we build a machine $\mathfrak{N}$ that recognizes G_f on input $(\vec{x}, y)$. It simulates $\mathfrak{M}$ on input $\vec{x}$. If $\mathfrak{M}$ stops, it compares $f(\vec{x})$ to y and accepts if they are equal.

$\Leftarrow$: From $\mathfrak{N}$ that recognizes the graph of f, we build $\mathfrak{M}$ such thatit computes f as follow : on input $\vec{x}$, repeatidly, for $i = 1, 2, \ldots$n it simulates $\mathfrak{N}$ for i-many steps on input $(\vec{x}, 0), \ldots, (\vec{x}, i)$. If it accepts $(\vec{x}, m)$, then $\mathfrak{M}$ outputs m.

□

3.1.9 Remark. If f is total and Turing computable, then G_f is Turing decidable.

f is a partial TC function	G_f is recursively enumerable	G_f is Turing recognizable
f is a total TC function	G_f is recursive	G_f is Turing decidable

3.1.10 Remark. All functions un R_0 are total TC. By induction on n, it is easy to see that $\mathrm{PR} = \bigcup_{n\in\mathbb{N}} R_n$ contains only total TC functions.

$$\mathrm{PR} \subsetneq \text{Total TC functions} \subsetneq \text{TC functions}.$$

3.1.11 Proposition. The set PR is closed under **variable substitution** : given $f : \mathbb{N}^p \longrightarrow \mathbb{N}$ a PR function and any $\sigma : \{1, \ldots, p\} \longrightarrow \{1, \ldots, p\}$ (not necessarily injective), we have

$$\begin{aligned} g : \mathbb{N}^p &\longrightarrow \mathbb{N} \\ (x_1, \ldots, x_p) &\longmapsto f(x_{\sigma(1)}, \ldots, x_{\sigma(p)}) \end{aligned}$$

and $g \in \mathrm{PR}$.

Proof. The function g is the following composite :

$$g(\vec{x}) = f(P^p_{\sigma(1)}(\vec{x}), \ldots, P^p_{\sigma(p)}(\vec{x})).$$

□

3.1.12 Proposition. If $A \subset \mathbb{N}^n$ is a PR set and $f_1, \ldots, f_n : \mathbb{N}^p \longrightarrow \mathbb{N}$ are PR functions, then

$$E = \{\vec{x} \in \mathbb{N}^p \mid (f_1(\vec{x}), \ldots, f_n(\vec{x})) \in A\}$$

is a PR set.

Proof. Consider its characteristic function :

$$\chi_E(\vec{x}) = \chi_A(f_1(\vec{x}), \ldots, f_n(\vec{x})).$$

□

3.1.13 Corollary. Let f and g be PR functions. Then the following sets are PR :

$$\begin{aligned} &\{\vec{x} \mid f(\vec{x}) < g(\vec{x})\} \\ &\{\vec{x} \mid f(\vec{x}) = g(\vec{x})\}. \end{aligned}$$

3.1.14 Proposition. If $A, B \subseteq \mathbb{N}^p$ are PR sets, then

$$A \cup B, \quad A \cap B, \quad A \Delta B, \quad \mathbb{N}^p \setminus A, \quad A \setminus B$$

are PR sets.

3.1.15 Proposition (Case study)**.** If $f_1, \dots, f_{n+1} : \mathbb{N}^p \longrightarrow \mathbb{N}$ are PR functions and $A_1, \dots, A_n \subset \mathbb{N}^p$ are all PR sets, then

$$g : \mathbb{N}^p \longrightarrow \mathbb{N}$$
$$\vec{x} \longmapsto \begin{cases} f_1(\vec{x}) & \text{if } \vec{x} \in A_1 \\ f_2(\vec{x}) & \text{if } \vec{x} \in A_2 \setminus A_1 \\ f_3(\vec{x}) & \text{if } \vec{x} \in A_3 \setminus (A_1 \cup A_2) \\ \vdots & \\ f_{n+1}(\vec{x}) & \text{if } \vec{x} \notin \bigcup_{k=1}^{n} A_n \end{cases}$$

is a PR function.

Proof. Consider the following form for g :

$$g(\vec{x}) = f_1(\vec{x})\chi_{A_1}(\vec{x}) + f_2(\vec{x})\chi_{A_2 \setminus A_1} + \cdots .$$

□

3.1.16 Corollary. Functions $\sup, \inf : \mathbb{N}^p \longrightarrow \mathbb{N}$ are PR.

3.1.17 Proposition. If $f : \mathbb{N}^{p+1} \longrightarrow \mathbb{N}$ is PR, then

$$g(x_1, \dots, x_p, y) = \sum_{t=0}^{y} f(x_1, \dots, x_p, t),$$
$$h(x_1, \dots, x_p, y) = \prod_{t=0}^{y} f(x_1, \dots, x_p, t)$$

are PR functions.

3.1.18 Proposition (Bounded minimisation)**.** Given a PR set $A \subseteq \mathbb{N}^{p+1}$, then

$$f : \mathbb{N}^{p+1} \longrightarrow \mathbb{N}$$
$$(\vec{x}, z) \longmapsto \begin{cases} 0 & \text{if there exist no integer } t \leq z \text{ such that } (\vec{x}, t) \in A \\ \text{the least } t \leq z \text{ such that } (\vec{x}, t) \in A & \text{otherwise.} \end{cases}$$

is a PR function.

Proof. The function f is defined by

$$f(\vec{x}, 0) = 0$$
$$f(\vec{x}, z+1) = \begin{cases} f(\vec{x}, z) & \text{if } \sum_{y=0}^{z} \chi_A(\vec{x}, y) \geq 1 \\ f(\vec{x}, z) & \text{otherwise an if } f(\vec{x}, z) \in A \\ 0 & \text{otherwise.} \end{cases}$$

□

3.1.19 Notation. We denote $f(\vec{x}, z)$ by

$$f(\vec{x}, z) = \mu t \leq z, \quad (\vec{x}, z) \in A.$$

3.1.20 Proposition. The set of all **predicates** (i.e. PR sets) is closed under bounded quantification : if $A \leq \mathbb{N}^{p+1}$ is a PR set, then

$$B = \{(\vec{x}, z) \mid \exists t \leq z \text{ such that } (\vec{x}, t) \in A\},$$
$$C = \{(\vec{x}, z) \mid \forall t \leq z \text{ such that } (\vec{x}, t) \in A\}$$

are PR sets.

Proof. Define the characteristic functions :

$$\chi_B(\vec{x}, z) = 1\dot{-}\left(1\dot{-}\sum_{t=0}^{z}\chi_A(\vec{x}, t)\right),$$
$$\chi_C(\vec{x}, z) = \prod_{t=0}^{z}\chi_A(\vec{x}, t).$$

□

3.1.21 Examples. 1. $\mathbb{N}$ is a PR set.

2. $\{2n \mid n \in \mathbb{N}\}$ is PR :

$$\chi(0) = 1$$
$$\chi(n+1) = 1\dot{-}\chi(n).$$

3. The function

$$q(x, y) = \begin{cases} 0 & \text{if } y = 0 \\ \lceil \frac{x}{y} \rceil & \text{otherwise} \end{cases}$$

is PR. Indeed :

$$q(x, y) = \mu t \leq x, \quad (t+1)y > x.$$

4. $\{(x, y) \mid x \text{ is divided by } y\}$ is PR. Indeed :

$$\chi(x, y) = 1\dot{-}(x\dot{-}q(x, y) \cdot y).$$

5. $\{x \mid x \text{ si prime}\}$ is PR.

6. The function

$$\pi : \mathbb{N} \longrightarrow \mathbb{N}$$
$$n \longmapsto \text{the } (n+1)^{\text{th}} \text{ prime number}$$

is PR. Indeed :

$$\pi(0) = 2$$
$$\pi(n+1) = \mu z \leq \pi(n)! + 1, \quad z > \pi(n) \text{ and } z \text{ prime.}$$

3.2 Coding sequences of integers

3.2.1 Proposition. For every non zero $p \in \mathbb{N}$, there exist PR functions $\alpha_p : \mathbb{N}^p \longrightarrow \mathbb{N}$, $\beta_p^1, \ldots, \beta_p^p : \mathbb{N} \longrightarrow \mathbb{N}$ such that

- α_p is a bijection,
- $\alpha_p^{-1}(x) = (\beta_p^1(x), \ldots, \beta_p^p(x))$.

Proof. We define those functions by induction on $p > 0$.

- Define $\alpha_1 = \beta_1^1 = \mathrm{id}_{\mathbb{N}}$,
- Define $\alpha_2(x, y) = \alpha_2(x + y, 0) + y$. It is a PR function. Define

$$\begin{aligned} \beta_2^1(n) &= \mu x \leq n, \quad \exists t \leq n \text{ such that } \alpha_2(x, t) = n \\ \beta_2^2(n) &= \mu y \leq n, \quad \exists t \leq n \text{ such that } \alpha_2(t, y) = n. \end{aligned}$$

- We define α_{p+1} by induction :

$$\begin{aligned} \alpha_{p+1}(x_1, \ldots, x_{p+1}) &= \alpha_p(x_1, \ldots, x_{p-1}, \alpha_2(x_p, x_{p+1})), \\ \beta_{p+1}^1 &= \beta_p^1, \\ &\vdots \\ \beta_{p+1}^{p-1} &= \beta_p^{p-1}, \\ \beta_{p+1}^p &= \beta_2^1 \circ \beta_p^p, \\ \beta_{p+1}^{p+1} &= \beta_2^2 \circ \beta_p^p. \end{aligned}$$

□

3.3 Recursive functions

3.3.1 Notation. Let $\mathcal{F}_p$ be the set of all partial functions $\mathbb{N}^p \longrightarrow \mathbb{N}$.

If (D_f, f) is a partial function and if $\vec{x} \notin D_f$, we say that f is not defined on $\vec{x}$, or that $f(\vec{x})$ is not defined. Two partial functions (D_f, f) and (D_g, g) are equal iff $D_f = D_g$ and if $f(\vec{x}) = g(\vec{x})$ for all $\vec{x} \in D_f$. Recall that f is total if its domain is $D_f = \mathbb{N}^p$.

3.3.2 Definition (Composition of partial functions)**.** Let $f_1, \ldots, f_n \in \mathcal{F}_p$, $g \in \mathcal{F}_n$. The **composition** $h = g(f_1, \ldots, f_n)$ is the element of $\mathcal{F}_p$ defined by

- $h(\vec{x})$ is not defined if one $f_i(\vec{x})$ is not defined or if $g(f_1(\vec{x}), \ldots, f_n(\vec{x}))$ is not defined,
- otherwise, $h(\vec{x}) = g(f_1(\vec{x}), \ldots, f_n(\vec{x}))$.

3.3.3 Proposition. For all $g \in \mathcal{F}_p$ and $h \in \mathcal{F}_{p+2}$, there exists a unique $f \in \mathcal{F}_{p+1}$ such that $\forall \vec{x} \in \mathbb{N}^p$

- $f(\vec{x}, 0) = g(\vec{x})$,
- $f(\vec{x}, y + 1) = h(\vec{x}, y, f(\vec{x}, y))$.

If $f(\vec{x}, y)$ is not defined, then $f(\vec{x}, y + 1)$ is not defined.

3.3.4 Definition (Minimisation)**.** Let $f \in \mathcal{F}_{p+1}$, then the partial function $g(\vec{x}) = \mu y,\ f(\vec{x}, y) = 0$ is defined by

- if there exists z such that $f(\vec{x}, z) = 0$ and $\forall z' < z,\ f(\vec{x}, z')$ is defined, then $g(\vec{x})$ is the least such z,
- otherwise, $g(\vec{x})$ is not defined.

3.3.5 Remark. Given $A \subseteq \mathbb{N}^{p+1}$:

$$\mu y,\ (\vec{x}, y) \in A = \mu y,\ 1 \dot{-} \chi_A(\vec{x}, y) = 0.$$

3.3.6 Definition (Recursive function)**.** The set of all **recursive functions** is the least subset of $\bigcup_{n\in\mathbb{N}} \mathcal{F}_n$ that contains

- all constants functions,
- all projections,
- the successor function,

and that is closed under

- composition,
- induction,
- minimisation.

3.3.7 Theorem. A function $f \in \mathcal{F}_p$ is (partial) recursive iff f is Turing computable.

Proof. $\Rightarrow$: Clear.

$\Leftarrow$: The idea is to take a TM $\mathfrak{M}$ that computes a partial function $f \in \mathcal{F}_p$. The machine $\mathfrak{M}$ starts with $x_1 \cdots x_p$ as input and

- if $\vec{x} \notin D_f$, then $\mathfrak{M}$ never stops,
- if $\vec{x} \in D_f$, then $\mathfrak{M}$ stops, accepts and $f(\vec{x})$ is written on the tape.

We show a more general result. Let $\mathfrak{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}} \rangle$ be a TM such that $\Gamma = \{0, \ldots, k-1\}$, where $0 = \sqcup$, and $Q = \{q_0, \ldots, q_l\}$. We define a PR coding of Γ^*

$$\begin{aligned} \Gamma &\longrightarrow \mathbb{N} \\ u &\longmapsto \lceil u \rceil. \end{aligned}$$

We show that ther eist a recursive function $f_{\mathfrak{M}} \in \mathcal{F}_1$ such that on input u

- if $\mathfrak{M}$ doesn't stop or reject, then $\lceil u \rceil \notin D_{f_{\mathfrak{M}}}$,
- if $\mathfrak{M}$ accept with v as output, then $f_{\mathfrak{M}}(\lceil u \rceil) = \lceil v \rceil$.

We say that $f_{\mathfrak{M}}$ works as $\mathfrak{M}$ "in the code". We now define a coding for $w \in \Gamma^*$: we read w as a number in base k : if $w = w_0 \cdots w_n$, then

$$\lceil w \rceil = \lceil w_0 \cdots w_n \rceil = \sum_{i=0}^{k} w_i k^i.$$

This coding is surjective but not injective. Given $\lceil w \rceil$, we recover the letter w_j from $\lceil w \rceil$ by

$$\text{letter}(w, j) = \text{Reminder}(q(\lceil w \rceil, k^j), k),$$

where q is the quotient and Reminder the reminder of the euclidian division. If j is greater than the length of w, then $\text{letter}(w, j) = 0$, i.e. the blank symbol. At any time of the computation, a configuration of the machine will be of the form

$$(\text{word of the tape},\ \text{current state},\ \text{position of the head})\,.$$

The tape word is coded by $\lceil w \rceil$, the state q_i is coded by i and the position of the head is a integer $p \in \mathbb{N}$. The code for such a configuration is $\alpha_3(\lceil w \rceil, i, p)$. Recall that from a code x of a configuration, we recover $\lceil w \rceil = \beta_3^1(x)$, $i = \beta_3^2(x)$ and $p = \beta_3^3(x)$. We define the set $\text{Config}_{\mathfrak{M}} \subseteq \mathbb{N}$ the set of correct codes of configurations of $\mathfrak{M}$, i.e.

$$x \in \text{Config}_{\mathfrak{M}} \iff \beta_3^2(x) < |Q|.$$

We define a PR predicate $\text{NextConfig}_{\mathfrak{M}} \subseteq \mathbb{N}^2$ such that $(a, b) \in \text{NextConfig}_{\mathfrak{M}}$ iff

- $a, b \in \text{Config}_{\mathfrak{M}}$,
- if a codes the configuration (w, i, p) and if b codes the configuration (w', i', p'), then δ yields in one step from (w, i, p) to (w', i', p').

There is also a PR predicate $\text{InitConfig}_{\mathfrak{M}} \subseteq \mathbb{N}$ such that $x \in \text{InitConfig}_{\mathfrak{M}}$ iff

- $\beta_3^1(x) = 0$ or the input coded by $\beta_3^1(x)$ contains no 0,
- $\beta_3^2(x) = \beta_3^3(x) = 0$.

Define a PR function $\text{Compute}_{\mathfrak{M}} : \mathbb{N}^2 \longrightarrow \mathbb{N}$ such that if a is the code of an initial configuration of $\mathfrak{M}$ and $t \in \mathbb{N}$, then $\text{Compute}_{\mathfrak{M}}(a, t)$ will return the code of a configuration that $\mathfrak{M}$ is in after t many steps. Fix an integer e such that e is not the code of a configuration (for instance $e = \alpha_3(0, |Q|, 0)$). Then

$$\text{Compute}_{\mathfrak{M}}(a, 0) = \begin{cases} a & \text{if } a \in \text{InitConfig}_{\mathfrak{M}}, \\ e & \text{otherwise,} \end{cases}$$

$$\text{Compute}_{\mathfrak{M}}(a, t+1) = b \iff (\text{Compute}_{\mathfrak{M}}(a, t), b) \in \text{NextConfig}_{\mathfrak{M}}\,.$$

Define a partial function $\text{Time}_{\text{accept}} : \mathbb{N} \longrightarrow \mathbb{N}$ that returns thenumber os steps needed by $\mathfrak{M}$ to accept the output. There is 2 cases :

- if q_0 is the accpeting state, then the function is constant 0,
- otherwise,

$$\text{Time}_{\text{accept}}(a) = \mu t \begin{cases} a \in \text{InitConfig}_{\mathfrak{M}}, \\ \text{Compute}_{\mathfrak{M}}(a, t) \neq e, \\ \beta_3^2(\text{Compute}_{\mathfrak{M}}(a, t)) = \text{ the number that codes } q_{\text{accept}}. \end{cases}$$

Finally,

$$f_{\mathfrak{M}}(a) = \beta_3^1\left(\text{Compute}_{\mathfrak{M}}(c, \text{Time}_{\text{accept}}(c))\right),$$

where $c = \alpha_3(a, 0, 0)$.

□

3.3.8 Corollary. The proof uses minimisation only once. Consequently, every TC function is a partial function where minimisation is used only once.

Chapter 4

Arithmatical hierarchy

The goal is to classify the sets of integers by formulas of some kind and by oracle Turing machines.

4.1 Σ_n^0, Π_n^0 and Δ_n^0

4.1.1 Definition (Arithmetical hierarchy)**.** The **arithmetical hierarchy** $(\Sigma_n^0, \Pi_n^0, \Delta_n^0)$ is defined as follow :

- Σ_1^0 is the class of all recursively enumerable sets,
- Π_n^0 is the class of all $A \subseteq \mathbb{N}^p$ such that $A^C \in \Sigma_n^0$,
- $\Delta_n^0 = \Sigma_n^0 \cap \Pi_n^0$,
- $\Sigma_{n+1}^0 = \left\{\exists x\, A \mid A \in \Pi_n^0\right\}$.

4.1.2 Examples.

- $\{p \mid p \text{ prime}\} \in \Delta_1^0$,
- $\{n \mid n \text{ codes a TM that halts on empty input}\} \in \Sigma_1^0 \setminus \Pi_1^0$.

4.1.3 Lemma. Let $A \subseteq \mathbb{N}^{p+k}$, $A \in \Sigma_1^0$. Then

$$B = \{\vec{y} \in \mathbb{N}^k \mid \exists \vec{x} \in \mathbb{N}^p,\ (\vec{x}, \vec{y}) \in A\} \in \Sigma_1^0.$$

Proof. Given an enumerator $\mathfrak{E}_A$ of A, we construct $\mathfrak{E}_B$, an enumerator of B as follow : it works juste like $\mathfrak{E}_A$ but esases the first p components of eveny $p+k$ tuples that $\mathfrak{E}_A$ prints. □

If $B = \{0\}$ and $A = C \times B$ for some $C \subseteq \mathbb{N}^p$, we have $B \in \Sigma_1^0$ and C does not need to be recursively enumerable, neither does A.

4.1.4 Corollary. Let $D \subseteq \mathbb{N}^{p+k}$, $D \in \Pi_1^0$. Then

$$E = \{\vec{y} \in \mathbb{N}^k \mid \forall \vec{x} \in \mathbb{N}^p,\ (\vec{x}, \vec{y}) \in D\} \in \Pi_1^0.$$

Proof. $E^C \in \Sigma_1^0$. □

4.1.5 Proposition. If $B \in \mathbb{N}^p$, then $B \in \Sigma_1^0$ iff $\exists A \subseteq \mathbb{N}^{p+1}$, $A \in \Delta_1^0$ such that

$$B(\vec{y}) \iff \exists x\, A(x, \vec{y}).$$

Proof. It goes back to the proof that a function if recursive if it is Turing computable. The whole proof used only one minimization, all the rest was primitive recursive. We can reconstruct the proof so that minimization is applied to a primitive recursive predicate $\mu t\, P(t, \vec{x}) = 1$ may be expressed by $\mu t\, P(t, \vec{x}) = 0$. □

4.1.6 Proposition. The classes Σ^0_n and Π^0_n are closed under finite union and finite intersection.

Proof. By orevious result, $B_0, B_1 \in \Sigma^0_n$ iff $\exists A_0, A_1 \in \Delta^0_1$ such that

$$B_0 = \exists x_1 \forall x_2 \exists x_3 \cdots Q_n x_n\, A_0,$$
$$B_1 = \exists x_1 \forall x_2 \exists x_3 \cdots Q_n x_n\, A_1.$$

By induction on n :

- If $n = 1$, then $B_0 = \exists x\, A_0$ and $B_1 = \exists x\, A_1$.
  - $B_0 \cup B_1 = \exists x\, (A_0 \cup A_1)$.
  - $(B_0 \cap B_1)(\vec{y}) \iff \exists x\, A_0(\beta^1_2(x), \vec{y}) \wedge A_1(\beta^2_2(x), \vec{y})$.
- If $n > 1$, we assume that the result holds for n, and we show it holds for $n + 1$.
  -
  $$(B_0 \cup B_1)(\vec{y}) \iff \underbrace{\exists x_1\ \underbrace{\underbrace{(\forall x_2 \exists x_3 \forall x_4 \cdots\ A_0(\vec{x}, \vec{y}))}_{\in \Pi^0_n} \vee \underbrace{(\forall x_2 \exists x_3 \forall x_4 \cdots\ A_1(\vec{x}, \vec{y}))}_{\in \Pi^0_n}}_{\in \Pi^0_n}}_{\in \Sigma^0_{n+1}}.$$
  -
  $$(B_0 \cap B_1)(\vec{y}) \iff \underbrace{\exists x_1\ \underbrace{\underbrace{(\forall x_2 \exists x_3 \forall x_4 \cdots\ A_0(\beta^1_2(x_1), x_2, \ldots, \vec{y}))}_{\in \Pi^0_n} \vee \underbrace{(\forall x_2 \exists x_3 \forall x_4 \cdots\ A_1(\beta^2_2(x_1), x_2, \ldots, \vec{y}))}_{\in \Pi^0_n}}_{\in \Pi^0_n}}_{\in \Sigma^0_{n+1}}.$$

It is similar to show that Π^0_n is also closed under finite union and finite intersection. □

4.1.7 Proposition. Let $A \subseteq \mathbb{N}^{p+k}$, $n > 0$.

- If $A \in \Sigma^0_n$, then if
  $$B(\vec{y}) \iff \exists \vec{x} \in \mathbb{N}^p\ A(\vec{x}, \vec{y})$$
  we have $B \in \Sigma^0_n$.
- If $A \in \Pi^0_n$, then if
  $$B(\vec{y}) \iff \forall \vec{x} \in \mathbb{N}^p\ A(\vec{x}, \vec{y})$$
  we have $B \in \Pi^0_n$.

Proof.

- If $B \in \Sigma_n^0$, then there exists $A \in \Delta_1^0$ such that

$$B(\vec{x}, \vec{y}) \iff \exists z_1 \forall z_2 \exists z_3 \cdots Q_n z_n \; A(\vec{z}, \vec{x}, \vec{y}).$$

  Hence

$$\begin{aligned} \exists \vec{x} \, B(\vec{x}) &\iff \exists \vec{x} \exists z_1 \forall z_2 \exists z_3 \cdots Q_n z_n \; A(\vec{z}, \vec{x}, \vec{y}) \\ &\iff \exists x \forall z_2 \exists z_3 \cdots Q_n z_n \; A(\beta_{p+1}^1(x), \beta_{p+1}^2(x), \cdots, \vec{y}) \\ &\in \Sigma_n^0. \end{aligned}$$

- It is imediate by complementation.

□

4.1.8 Proposition. For all $n > 0$, $\Sigma_n^0 \cup \Pi_n^0 \subseteq \Delta_{n+1}^0$.

Proof. By induction on n.

- If $n = 1$.
  - Let $B \in \Sigma_1^0$, $B = \exists x \; A$ for some $A \in \Delta_1^0$. Hence $A \in \Pi_1^0$, and therefore $B \in \Pi_2^0$. Si $\Sigma_1^0 \subseteq \Pi_2^0$.
  - If $C \subseteq \mathbb{N}^p$, $C \in \Pi_1^0$, then $C' = \mathbb{N} \times C \in \Pi_1^0$ and $\exists x \; C' = C$. So $C \in \Sigma_2^0$ and $\Pi_1^0 \subseteq \Sigma_2^0$.
- Assume the hypothesis holds for n.
  - Let $B \in \Sigma_{n+1}^0$, $B = \exists x \; A$ for some $A \in \Pi_n^0$. Hence $B = \exists x \; A$, $A \in \Pi_{n+1}^0$ (by induction hypothesis) and so $B \in \Sigma_{n+2}^0$.
  - Let $C \in \Pi_{n+1}^0$, $C' = \mathbb{N} \times C \in \Pi_{n+1}^0$. Hence $C = \exists x \; C'$, $C' \in \Pi_{n+1}^0$ (by induction hypothesis) and so $C \in \Sigma_{n+2}^0$.

□

4.2 Turing machine with oracle

4.2.1 Definition (Turing machine with oracle)**.** A **Turing machine with an orable** $\mathcal{O} \subseteq \mathbb{N}$ is a regular TM wath an extra tape on which at the begening of the computation, the caracteristic function of $\mathcal{O}$ is written (it's an infinite word !).

4.2.2 Definition (Jump)**.** Given $\mathcal{O} \subseteq \mathbb{N}$, the **jump** if $\mathcal{O}$ is the subset of $\mathbb{N}$ defined by

$$\mathcal{O}' = \left\{ i \in \mathbb{N} \mid i = \lceil \mathfrak{M}^{\mathcal{O}} \rceil, \text{ and } \lceil \mathfrak{M}^{\mathcal{O}} \rceil \text{ stops on input } i \right\}.$$

Here, i is the code of a TM, and the oracle $\mathcal{O}$ does not belong to the code.

4.2.3 Remark. $\emptyset'$ correspond to the halting problem. So $\emptyset' \in \Sigma_1^0 \setminus \Pi_1^0$.

4.2.4 Theorem. $\Sigma_n^0 \neq \Pi_n^0$.

Proof. 1. $\forall n \geq 0$, $\forall k > 0$, we show that there exist a universal set for Σ_n^0 subsets of $\mathbb{N}$. We only do it for $n = 3$ and $k = 1$, for simplicity of the notations. Let $A \subseteq \mathbb{N}$.

$$A \in \Sigma_3^0 \text{iff} A(x) \iff \exists y_3 \forall y_2 \exists y_1 \; B(y_1, y_2, y_3, x),$$

where $B \in \Sigma_1^0$, i.e. B is a recursively enumerable predicate. Take a universal set for RE subsets of $\mathbb{N}^4$, i.e. $\mathcal{U} \subseteq \mathbb{N}^5$ with

$$\mathcal{U}(i, y_1, y_2, y_3, x) \iff i \text{ codes a TM that accepts } (y_1, y_2, y_3, x).$$

We have $\mathcal{U} \in \Sigma_1^0$. We may view $\mathcal{U}$ as the langage recognized by a universal TM that first checks weather i is the code of a TM, than works as i on input (y_1, y_2, y_3, x). Define $\mathcal{U}_3 \subseteq \mathbb{N}^2$ by

$$\mathcal{U}_3(i, x) \iff \exists y_3 \forall y_2 \exists y_1 \, \mathcal{U}(i, y_1, y_2, y_3, x).$$

Then $\mathcal{U}_3 \in \Sigma_3^0$ abs is universal for Σ_3^0 subsets of $\mathbb{N}$.

- $\mathcal{U}_3 \in \Sigma_3^0$ by definition.
- $\forall A \subseteq \mathbb{N}$ that is Σ_3^0, there is an index i of A such that $\mathcal{U}(i, x) \iff A(x)$.

Now, consider the following predicate $P \subseteq \mathbb{N}$:

$$P(i) \iff \mathcal{U}_3(i, i).$$

We have that $P \in \Sigma_3^0 \setminus \Pi_3^0$ By contradition, if $P \in \Pi_3^0$, than we would have $P^C \in \Sigma_3^0$. Si P_C has an index $j \in \mathbb{N}$, i.e. $\forall x \in \mathbb{N}$, $P^C(x) \iff \mathcal{U}_3(j, x)$. But

$$\begin{aligned} j \in P^C &\iff \mathcal{U}_3(j, j) \\ &\iff j \in P, \end{aligned}$$

which is absurd. So $P \notin \Pi_3^0$.

□

4.2.5 Corollary. The arothmetical hierarchy is strict.

Proof. $\forall n > 0$, $\exists A \in \Sigma_n^0 \setminus \Pi_n^0$. Hence $A^C \in \Pi_n^0 \setminus \Sigma_n^0$. We know that

$$\begin{aligned} \Delta_1^0 &\subsetneq \Sigma_1^0, \Pi_1^0, \\ \Delta_{n+1}^0 &= \Sigma_{n+1}^0 \cap \Pi_{n+1}^0, \\ \Sigma_n^0 \cup \Pi_n^0 &\subseteq \Sigma_{n+1}^0, \Pi_{n+1}^0. \end{aligned}$$

Hence $\Delta_n^0 \subseteq \Delta_{n+1}^0 \subsetneq \Sigma_{n+1}^0, \Pi_{n+1}^0$. □

Chapter 5

Arithmetisation of the first order logic and Gödels first incompleteness theorem

Let $\mathcal{P}$ denote the Peano arithmetic. The set $\mathcal{L}_{\mathcal{P}}$ contains

- the constant symbols $\underline{0}$ and $\underline{1}$,
- the function symbol $\underline{s}$ of arity 1,
- two function symbols $\underline{+}$ and $\underline{\times}$ of arity 2.

This is a langage with equality. From now on, $\mathbb{N} = \langle \mathbb{N}, 0, 1, s, +, \times \rangle$ is the standard model.

5.0.6 Axioms (Peano)**.**

(A_1) : $\forall x_0 \, \neg \underline{s}(x_0) = \underline{0}$.

(A_2) : $\forall x_0 \exists x_1 \ (\neg x_0 = \underline{0} \to \underline{s}(x_1) = x_0)$.

(A_3) : $\forall x_0, \forall x_1 \ (\underline{s}(x_0) = \underline{s}(x_1) \to x_0 = x_1)$.

(A_4) : $\forall x_0 \, x_0 \underline{+} \underline{0} = x_0$.

(A_5) : $\forall x_0 \forall x_1 \, x_0 \underline{+} \underline{s}(x_1) = \underline{s}(x_0 \underline{+} x_1)$.

(A_6) : $\forall x_0 \, x_0 \underline{\times} \underline{0} = \underline{0}$.

(A_7) : $\forall x_0 \forall x_1 \, x_0 \underline{\times} \underline{s}(x_1) = (x_0 \underline{\times} x_1) \underline{+} x_0$.

Induction scheme : $\forall n \in \mathbb{N}$, $\forall \phi \in \mathcal{F}(\mathcal{L}_{\mathcal{P}})$ whose free variables are amond $x_0, \ldots, x_n$,

$$\forall x_1 \cdots \forall x_n \left((\phi[\underline{0}, x_1, \ldots, x_n] \wedge \forall x_0 \ (\phi[x_0, \ldots, x_n] \to \phi[\underline{s}(x_0), x_1, \ldots, x_n])) \to \forall x_0 \, \phi[x_0, \ldots, x_n] \right).$$

5.0.7 Remark. Up to isomorphism, there exist $2^{\aleph_0}$ countable models of $\mathcal{P}$ that are not isomorphic. $\mathbb{N}$ s one of them.

5.0.8 Notation. For $n \in \mathbb{N}$, we write

$$\underline{n} = \underbrace{\underline{s}(\underline{s}(\cdots \underline{s}}(\underline{0}) \cdots)).$$

Here is a bunch of interesting facts :

1. $\mathcal{P} \vdash \forall x_0\ \underline{0}\underline{+}x_0 = x_0$.
2. $\mathcal{P} \vdash \forall x_0 \forall x_1\ \underline{s}(x_1)\underline{+}x_0 = \underline{s}(x_1\underline{+}x_0)$.
3. $\mathcal{P} \vdash \forall x_0\ \underline{1}\underline{+}x_0 = \underline{s}(x_0)$.
4. $\mathcal{P} \vdash \forall x_0 \forall x_1\ x_0\underline{+}x_1 = x_1\underline{+}x_0$.
5. $\mathcal{P} \vdash \forall x_0 \forall x_1 \forall x_2\ x_0\underline{+}(x_1)\underline{+}x_2) = (x_0\underline{+}x_1)\underline{+}x_2$.
6. $\mathcal{P} \vdash \forall x_0\ \underline{0}\underline{\times}x_0 = \underline{0}$.
7. $\mathcal{P} \vdash \forall x_0\ x_0\underline{\times}\underline{1} = x_0$.
8. $\mathcal{P} \vdash \forall x_0\ \underline{1}\underline{\times}x_0 = x_0$.
9. $\mathcal{P} \vdash \forall x_0 \forall x_1 \forall x_2\ x_0\underline{\times}(x_1\underline{+}x_2) = (x_0\underline{\times}x_1)\underline{+}(x_0\underline{\times}x_2)$.
10. $\mathcal{P} \vdash \forall x_0 \forall x_1 \forall x_2\ (x_0\underline{\times}x_1)\underline{\times}x_0 = x_0\underline{\times}(x_1\underline{\times}x_2)$.
11. $\mathcal{P} \vdash \forall x_0 \forall x_1\ x_0\underline{\times}x_1 = x_1\underline{\times}x_0$.
12. $\mathcal{P} \vdash \forall x_0 \forall x_1 \forall x_2\ (x_0\underline{+}x_2 = x_1\underline{+}x_2) \to x_0 = x_1$.
13. $\mathcal{P} \vdash \forall x_0 \forall x_1\ \neg x_1 = \underline{0} \to \neg x_0\underline{+}x_1 = \underline{0}$.
14. $\mathcal{P} \vdash \forall x_0 \forall x_1\ x_0\underline{+}x_1 = \underline{0} \to x_0 = \underline{0} \wedge x_1 = \underline{0}$.
15. $\mathcal{P} \vdash \forall x_0 \forall x_1\ x_0\underline{+}x_1 = x_0 \to x_1 = \underline{0}$.
16. For now, we write $x_0 \leq x_1$ to abbreviate $\exists x_2\ x_2\underline{+}x_0 = x_1$, and $x_0 < x_1$ to abbreviate $x_0 \leq x_1 \wedge \neg x_0 = x_1$. We have $\mathcal{P} \vdash \forall x_0\ x_0 \leq x_0$ (reflexivity).
17. (Transitivity) $\mathcal{P} \vdash \forall x_0 \forall x_1 \forall x_2\ (x_0 \leq x_1 \wedge x_1 \leq x_2) \to x_0 \leq x_2$.
18. (Antisymetry) $\mathcal{P} \vdash \forall x_0 \forall x_1 \forall x_2\ (x_0 \leq x_1) \wedge (x_1 \leq x_0) \to x_0 = x_1$.
19. (Compatibility with $\underline{+}$) $\mathcal{P} \vdash \forall x_0 \forall x_1 \forall x_2\ (x_0\underline{+}x_2 \leq x_1\underline{+}x_2) \leftrightarrow x_0 \leq x_1$.
20. $\mathcal{P} \vdash \forall x_0 \forall x_1 x_0 \leq x_1 \vee x_1 \leq x_0$.
21. (Compatibility with $\underline{\times}$) $\mathcal{P} \vdash \forall x_0 \forall x_1 \forall x_2\ x_0 \leq x_1 \to x_0\underline{\times}x_2 \leq x_1\underline{\times}x_2$.
22. $\mathcal{P} \vdash \forall x_0 \forall x_1 (\neg x_0 = \underline{0} \wedge \neg x_1 = \underline{0}) \to \neg x_0\underline{\times}x_1 = 0$.
23. $\mathcal{P} \vdash \forall x_0 \forall x_1 \forall x_2 (x_0\underline{\times}x_2 = x_1\underline{\times}x_2) \to (x_0 = x_1 \vee x_2 = \underline{0})$.
24. $\forall n \in \mathbb{N}$, $\mathcal{P}_0 \vdash \underline{n+1} = \underline{s}(\underline{n})$.
25. $\forall m, n \in \mathbb{N}$, $\mathcal{P}_0 \vdash \underline{m} + \underline{n} = \underline{m+n}$.
26. $\forall m, n \in \mathbb{N}$, $\mathcal{P}_0 \vdash \underline{m}\underline{\times}\underline{n} = \underline{m \cdot n}$.
27. $\forall n \in \mathbb{N} \setminus \{0\}$, $\mathcal{P}_0 \vdash \neg\underline{n} = \underline{0}$.
28. $\forall m, n \in \mathbb{N}$, $m \neq n$, $\mathcal{P}_0 \vdash \neg\underline{m} = \underline{n}$.
29. $\forall n \in \mathbb{N}$, $\mathcal{P}_0 \vdash \forall x_0\ x_0 \leq \underline{n} \to \bigvee_{i=0}^{n} x_0 = \underline{i}$.

30. $\forall n \in \mathbb{N}$, $\mathcal{P}_0 \vdash \forall x_0\ (x_0 \leq \underline{n} \vee \underline{n} \leq x_0)$.

We denote $\mathcal{P}_0$ as the Peano arithmetic without the Induction Scheme. This is a very weak theory, in which one cannot prove that $\underline{+}$ is commutative. We first show that every model of $\mathcal{P}_0$ "starts" with a copy of $\mathbb{N}$.

5.0.9 Definition (Initial segment, final extension)**.** Let $\mathcal{M}$ and $\mathcal{N}$ be two models of $\mathcal{P}_0$, such that $\mathcal{N}$ is a substructure of $\mathcal{M}$. We say that $\mathcal{N}$ is an **initial segment** of $\mathcal{M}$ if $\forall a \in |\mathcal{N}|$, $\forall b \in |\mathcal{M}|$,

1. if $\mathcal{M} \models b \leq a$, then $b \in |\mathcal{N}|$,
2. if $b \notin |\mathcal{N}|$, then $\mathcal{M} \models a \leq b$.

We also say that $\mathcal{M}$ is a **final extension** of $\mathcal{N}$.

5.0.10 Theorem. Let $\mathcal{M}$ be a model of $\mathcal{P}_0$. Then

$$\mathcal{N} = \{a \in |\mathcal{M}| \mid \exists n \in \mathbb{N} \text{ such that } a = \underline{n}\}$$

is

1. isomorphic to $\mathbb{N}$,
2. an initial segment of $\mathcal{M}$.

Proof. 1. By 25., 26. and 28.

2. By 29. and 30.

□

5.1 Representable functions

Recall that $\mathcal{F}_p$ is the set of all total functions $\mathbb{N}^p \longrightarrow \mathbb{N}$.

5.1.1 Definition (Representable function)**.** Let $f \in \mathcal{F}_p$ be a total function, and $\Phi[x_0, \ldots, x_p]$ be a formula of $\mathcal{L}_0$, the langage of $\mathcal{P}_0$. We say that $\Phi[x_0, \ldots, x_p]$ **represents** f if for all p-tuple $(n_1, \ldots, n_p)$

$$\mathcal{P}_0 \vdash \forall x_0\ \Phi[x_0, \underline{n_1}, \ldots, \underline{n_p}] \leftrightarrow x_0 = \underline{f(\vec{n})}.$$

We say that f is **representable** by Φ. We say that a total function g is **representable** if it is representable by some formula of $\mathcal{L}_0$.

5.1.2 Definition (Representable set)**.** Let $A \subseteq \mathbb{N}^p$ be a set, and $\Phi[x_1, \ldots, x_p]$ be a formula of $\mathcal{L}_0$. We say that $\Phi[x_1, \ldots, x_p]$ **represents** A if $\forall \vec{n} \in \mathbb{N}^p$,

- if $\vec{n} \in A$, then $\mathcal{P}_0 \vdash \Phi[\underline{n_1}, \ldots, \underline{n_p}]$,
- if $\vec{n} \notin A$, then $\mathcal{P}_0 \vdash \neg\Phi[\underline{n_1}, \ldots, \underline{n_p}]$.

5.1.3 Remark. A set $A \subseteq \mathbb{N}^p$ is representable iff χ_A is representable. Indeed :

- If $\Phi[x_1, \ldots, x_p]$ represents A, then

  $$\Theta[x_0, \ldots, x_p] : (\Phi[x_1, \ldots, x_p] \wedge x_0 = \underline{1}) \vee (\neg\Phi[x_1, \ldots, x_p] \wedge x_0 = \underline{0})$$

  represents χ_A.

- If $\Psi[x_0, \ldots, x_p]$ represents χ_A, then

$$\Theta[x_1, \ldots, x_p] : \Psi[\underline{1}, x_1, \ldots, x_p]$$

represents A.

5.1.4 Theorem (Chinese reminder theorem)**.** Let $n_1, \ldots, n_k \in \mathbb{N} \setminus \{0\}$ be pairwise coprimes. Then, $\forall a_1, \ldots, a_k \in \mathbb{N}$, $\exists a \in \mathbb{N}$ such that

$$a \equiv a_i \ (n_i).$$

5.1.5 Lemma. There exists a total function $\beta : \mathbb{N}^3 \longrightarrow \mathbb{N}$ such that

1. β is recursive,
2. β is representable,
3. β has the property that $\forall p \in \mathbb{N}$ and for every sequence $(n_1, \ldots, n_p) \in \mathbb{N}^p$, there exists $a, b \in \mathbb{N}$ such that

$$\beta(i, a, b) = n_i, \qquad \forall 1 \leq i \leq p.$$

Proof. This makes use of the Chinese reminder theorem. We define $\beta(i, a, b)$ to be the reminder of b by $a(i + 1) + 1$. It is a recursive function. Define

$$\begin{aligned} \Phi[x_0, \ldots, x_4] : x_3 = (x_4 \underline{\times} \underline{s}(x_2 \underline{\times} \underline{s}(x_1))) \underline{+} x_0 \\ \wedge\, x_0 < \underline{s}(x_2 \underline{\times} \underline{s}(x_1)). \end{aligned}$$

Then β is represented by

$$\begin{aligned} \Phi_\beta[x_0, \ldots, x_3] : \exists x_4\, \Phi[x_0, \ldots, x_3] \\ \wedge\, (\forall x_4 < x_0\, \neg\Phi[x_0, \ldots, x_4]. \end{aligned}$$

Given a sequence $(a_0, \ldots, a_n)$, we choose $m > n + 1$ such that $a = m! > a_i$, $\forall 0 \leq i \leq n$. Then, all integer $a(i+1)+1$ are pairwise coprime, since if $c|a(i+1)+1$, $c|a-j+1)+1$, then $c|a(j-1) = m!(j-1)$. Hence $c \leq m$ which contradicts the hypothesis that $c|m!(i+1)+1$. By the Chinese reminder theorem, we get $b \in \mathbb{N}$ such that $b \equiv a_i \ (a(i + 1) + 1)$. Since $a_i \leq a(i + 1) + 1$, we get $a_i = \beta(i, a, b)$. □

5.1.6 Theorem (Representation theorem)**.** Every (total) recursive function is representable.

Proof.

- The successor function s is represented by

$$\Theta[x_0, x_1] : x_0 = \underline{s}(x_1).$$

- The projection function P_p^i is represented by

$$\Theta[x_0, \ldots, x_p] : x_0 = x_i.$$

- The constant function $x \longmapsto n$ is represented by

$$\Theta[x_0, \ldots, x_p] : x_0 = \underline{n}.$$

- We show that the set of representable functions is closed under composition. Let $f_1, \ldots, f_n \in \mathcal{F}_p$, $g \in \mathcal{F}_n$ be represented by $\Phi_1, \ldots, \Phi_n, \Psi$. Then $g(f_1, \ldots, f_n)$ is represented by

$$\Theta[x_0, \ldots, x_p] : \exists y_1 \cdots \exists y_n\, \Phi[x_0, y_1, \ldots, y_n] \wedge \bigwedge_{i=1}^{n} \Phi_i[y_i, x_1, \ldots, x_p].$$

- We show that given a subset $A \subseteq \mathbb{N}^{p+1}$ represented by $\Phi[x_0, \ldots, x_p]$, the function

$$f(x_1, \ldots, x_p) = \mu y \ (y, \vec{x}) \in A$$

  is representable, if it is total. Indeed, it is represented by

$$\Theta[x_0, \ldots, x_p] : \Phi[x_0, \ldots, x_p] \wedge (\forall x < x_0 \ \neg \Phi[x, x_1, \ldots, x_p]) .$$

- We show that recursion of representable function is still a representable function. We make use of the previous lemma. Given $g \in \mathcal{F}_p$, $h \in \mathcal{F}_{p+2}$ that are represented by $\Phi_g[x_0, \ldots, x_p]$ and $\Phi_h[x_0, \ldots, x_{p+2}]$. The function $f \in \mathcal{F}_{p+1}$ defined by
  - $f(\vec{x}, 0) = g(\vec{x})$,
  - $f(\vec{x}, x_{p+1} + 1) = h(\vec{x}, x_{p+1}, f(\vec{x}, x_{p+1}))$

  is representable. We define

$$\begin{aligned}\Phi_f[x_0, \vec{x}, x_{p+1}] : \exists y_1 \exists y_2 \ (&\exists y_0 \ \Phi_\beta[y_0, 1, y_1, y_2]) \\ &\wedge \Phi_g[y_0, \vec{x}] \\ &\wedge \Phi_\beta[x_0, \underline{s}(x_{p+1}), y_1, y_2] \\ &\wedge \forall y_3 < x_{p+1} \exists y_4 \exists y_5 (\Phi_\beta[y_4, \underline{s}(y_3), y_1, y_2] \\ &\qquad \wedge \Phi_\beta[y_5, \underline{s}(\underline{s}(y_2)), y_1, y_2] \\ &\qquad \wedge \Phi_h[y_5, \vec{x}, y_3, y_4]).\end{aligned}$$

□

5.2 Arithmetization of syntax

We code terms, formulas and proofs by integers.

5.2.1 Lemma. Let $p, n \in \mathbb{N}$, $k_1, \ldots, k_n \in \mathcal{F}_1$, $g \in \mathcal{F}_p$, $h \in \mathcal{F}_{n+1}$, all PR functions, and $\forall y > 0$, $\forall 1 \leq i \leq n$, $k_i(y) \leq y$. Then the unique function defined by

- $f(0, \vec{x}) = g(\vec{x})$,
- $f(y, \vec{x}) = h(y, f(k_1(y), \vec{x}), \ldots, f(k_n(y), \vec{x}))$

is PR.

Proof. Define ϕ by

- $\phi(0, \vec{x}) = 2^{g(\vec{x})}$,
- $\phi(y + 1, \vec{x}) = \phi(y, \vec{x}) p_{y+1}^{\gamma}$,

where p_i is the i-th prime number,

$$\gamma = h(y + 1, \delta(k_1(y + 1), \phi(y, \vec{x})), \cdots, \delta(k_n(y + 1), \phi(y, \vec{x}))),$$

and $\delta(i, x) = \mu z \ x$ is not divided by p_i^{z+1}. Then ϕ is PR and

$$f(y, \vec{x}) = \delta(y, \phi(y, \vec{x})).$$

□

5.2.2 Definition (Gödel number of a term)**.** Given a term t from $\mathcal{L}_0$, we define $\sharp t$, its **Gödel number** :

- if $t = \underline{0}$, then $\sharp t = \alpha_3(0, 0, 0)$,
- if $t = x_n$, then $\sharp t = \alpha_3(n + 1, 0, 0)$,
- if $t = \underline{s}(t_1)$, then $\sharp t = \alpha_3(\sharp t_1, 0, 1)$,
- if $t = t_1 \underline{+} t_2$, then $\sharp t = \alpha_3(\sharp t_1, \sharp t_2, 2)$,
- if $t = t_1 \underline{\times} t_2$, then $\sharp t = \alpha_3(\sharp t_1, \sharp t_2, 3)$.

5.2.3 Remark. This coding is injective.

5.2.4 Lemma. The set $\{\sharp t \mid t \text{ is a term of } \mathcal{L}_0\}$ is PR.

Proof. Exercise. □

5.2.5 Definition (Gödel number of a formula)**.** Given a formula ϕ from $\mathcal{L}_0$, we define $\sharp\phi$, its **Gödel number** :

- if $\phi : t_1 = t_2$, then $\sharp\phi = \alpha_3(\sharp t_1, \sharp t_2, 0)$,
- if $\phi : \neg\psi$, then $\sharp\phi = \alpha_3(\sharp\psi, 0, 1)$,
- if $\phi : \psi_1 \wedge \psi_2$, then $\sharp\phi = \alpha_3(\sharp\psi_1, \sharp\psi_2, 2)$,
- if $\phi : \psi_1 \vee \psi_2$, then $\sharp\phi = \alpha_3(\sharp\psi_1, \sharp\psi_2, 3)$,
- if $\phi : \psi_1 \to \psi_2$, then $\sharp\phi = \alpha_3(\sharp\psi_1, \sharp\psi_2, 4)$,
- if $\phi : \psi_1 \leftrightarrow \psi_2$, then $\sharp\phi = \alpha_3(\sharp\psi_1, \sharp\psi_2, 5)$,
- if $\phi : \exists x_n\, \psi$, then $\sharp\phi = \alpha_3(\sharp\psi, n, 6)$,
- if $\phi : \forall x_n\, \psi$, then $\sharp\phi = \alpha_3(\sharp\psi, n, 7)$.

5.2.6 Remark. This coding is also injective.

5.2.7 Lemma. The set $\{\sharp\phi \mid \phi \text{ is a formula of } \mathcal{L}_0\}$ is PR.

Proof. Exercise. □

5.2.8 Lemma. The following sets are PR :

- $\Theta_0 = \{(\sharp t, n) \mid t \text{ is a term in which } x_n \text{ does not occur}\}$,
- $\Theta_1 = \{(\sharp t, n) \mid t \text{ is a term in which } x_n \text{ does occur}\} = \Theta_0^C$,
- $\Omega_0 = \{(\sharp\phi, n) \mid \phi \text{ is a formula in which } x_n \text{ does not occur}\}$,
- $\Omega_0' = \{(\sharp\phi, n) \mid \phi \text{ is a formula in which } x_n \text{ does occur}\} = \Omega_0^C$,
- $\Omega_1 = \{(\sharp\phi, n) \mid \phi \text{ is a formula in which } x_n \text{ does not have free occurence}\}$,
- $\Omega_1 = \{(\sharp\phi, n) \mid \phi \text{ is a formula in which } x_n \text{ does not have bounded occurence}\}$,
- $\Omega_3 = \{\sharp\phi \mid \phi \text{ is a closed formula}\}$,

- $\Omega_4 = \{(\sharp\phi, n) \mid \phi \text{ is a formula in which } x_n \text{ does have a free occurence}\} = \Omega_1^C$.

5.2.9 Lemma. There exist two PR functions $\text{Subst}_t, \text{Subst}_f \in \mathcal{F}_3$ such that for all terms t and u, and for all formula ϕ,

$$\text{Subst}_t(n, \sharp t, \sharp u) = \sharp u_{[t/x_n]}$$
$$\text{Subst}_f(n, \sharp t, \sharp \phi) = \sharp \phi_{[t/x_n]}.$$

We make the following Hilbert style notion of a proof.

5.2.10 Definition (Proof)**.** Let T be a $\mathcal{L}$-theory. A **formal proof** from T of a formula ϕ is a finite sequence $\langle \phi_0, \ldots, \phi_n \rangle$ such that $\forall 0 \leq i \leq n$, either

- $\phi_i \in T$
- ϕ_i is a logical axiom,
- $\exists j, k < i$ such that $\phi_k : \phi_j \to \phi_i$ (modus ponens),
- $\exists j < i$ and a variable x such that $\phi_i : \forall x\, \phi_j$ (generalization).

The set of logical axioms contains exactly all formulas of the form

- $\exists x\, \phi \leftrightarrow \neg \forall x\, \neg \phi$,
- $\forall x\, (\phi \to \psi) \to (\phi \to \forall x\, \psi)$, where x has no free occurence in ψ,
- $\forall x\, \phi \to \phi_{[t,x]}$, where t is a term, ϕ without free occurence of x that lies in the scope of a quantifier that bounds a variable from t,
- all tautologies from 1st order logic.

We code formulas of propositional calculus on variables $A_1, A_2, \ldots$ in the same way as for forlumas inf 1st order logic. Without surprise,

$$\text{Prop} = \{\sharp P \mid P \text{ is a propositional formula}\}$$

is PR.

5.2.11 Lemma. The function

$$E : \mathbb{N}^2 \longrightarrow \mathbb{N}$$
$$(x, k) \longmapsto \begin{cases} \delta_k(P) & \text{if } x = \sharp P \\ 0 & \text{if } x \notin \text{Prop} \end{cases}$$

is PR.

Proof. Suppose that $x = \sharp P$, for some formula P (so $x \in \text{Prop}$).

- If $\beta_3^3(x) = 0$, then $P = A_n$, for some $n \in \mathbb{N}$. So

$$E(x, k) = \begin{cases} 1 & \text{if} \pi(\beta_3^1(x)) | k \\ 0 & \text{otherwise.} \end{cases}$$

- If $\beta_3^3(x) = 1$, then $P = \neg Q$, for some formula Q. So

$$E(x, k) = 1 \dot{-} E(\beta_3^1(x), k).$$

- If $\beta_3^3 = 2$, then $P = Q_1 \wedge Q_2$, for some formulas Q_1 and Q_2. So

$$E(x,k) = E(\beta_3^1(x),k)E(\beta_3^2(x),k).$$

- ...

□

5.2.12 Theorem (Decidability of propositional calculus)**.** The set of teutologies

$$\text{PropTaut} = \{\sharp P \mid P \text{ is a tautology}\}$$

is PR.

Proof. For all $k \in \mathbb{N}$ we define a thruth value function $\delta_k : \{A_n\}_{n\in\mathbb{N}} \longrightarrow 2$ by

$$\delta_k(A_n) = \begin{cases} 1 & \text{if } \pi(n)|k \\ 0 & \text{otherwise,} \end{cases}$$

implicitly extended to all formulas. Every truth value function $\delta : \{A_n\}_{n\in\mathbb{N}} \longrightarrow 2$ and every $c \in \mathbb{N}$, there exists $k \in \mathbb{N}$ such that $\forall i \leq c$ we have $\delta(A_i) = \delta_k(A_i)$. Indeed, take $k = \prod_{i=0}^{c} \pi(i)^{\delta(A_i)}$. Remark that $k \leq \pi(c)!$. Whenever A_n occurs in P, we have $\sharp A_n = n < \sharp P$. If P is a formula, then P is a tautology iff $\forall k \leq \pi(\sharp P)$ we have $\delta_k(P) = 1$. To finish up the proof, we make ue of the previous lemma. Remark that $\forall x \in \mathbb{N}$, $x \in \text{PropTaut}$ iff $x \in \text{Prop}$ and $\forall k \leq \pi(x)!$ we have $E(x,k) = 1$. □

5.2.13 Theorem. The set

$$\text{Taut} = \{\sharp\phi \mid \phi \text{ is a 1st order tautology}\}$$

is PR.

Proof. We define $\gamma : \mathbb{N} \longrightarrow \mathbb{N}$, a PR function such that if $x = \sharp\phi$, where ϕ is a 1st order formula, then $\gamma(x) = \sharp P_\phi$, where P_ϕ is a propositional formula with variables $A_{n_0}, \ldots, A_{n_k}$, such that

1. $n_i = \sharp\phi_i$, $0 \leq i \leq k$,
2. ϕ_i is either atomic, of the form $\exists x\, \theta$ or of the form $\forall x\, \theta$,
3. $(P_\phi)_{[\phi_i/A_{n_i}]} = \phi$.

Then, for all 1st order formula, we have $\sharp\phi \in \text{Taut}$ iff $\gamma(\sharp\phi) \in \text{PropTaut}$. Define γ as follow : if $x \in \text{Form}$

- if $\beta_3^3(x) \in \{0,6,7\}$, then $\gamma = \alpha_3(x,0,0)$,
- if $\beta_3^3(x) = 1$, then $P_{\neg\phi} = \neg P_\phi$, and $\gamma(x) = \alpha_3(\gamma(\beta_3^1(x)),0,1)$,
- if $\beta_3^3(x) \in \{2,3,4,5\}$, then $\gamma(x) = \alpha_3(\gamma(\beta_3^1(x)),\gamma(\beta_3^2(x)),\gamma(\beta_3^3(x)))$,
- if $\beta_3^3(x) > 7$, then $\gamma(x) = 0$.

□

Define

$$\begin{aligned}
\mathrm{Ax}_1 &= \{\sharp(\exists x\ \phi \leftrightarrow \neg\forall x\neg\phi) \mid \phi \text{ is a formula, and } x \text{ is a variable}\},\\
\mathrm{Ax}_2 &= \{\sharp((\forall x\ (\phi \to \psi)) \to (\phi \to \forall x\ \psi))) \mid x \text{ doesn't occur free in } \phi\},\\
\mathrm{Ax}_3 &= \{\sharp((\forall x\ \phi) \to \phi_{[t/x]}) \mid t \text{ is a term, and no variable catpure occurs}\},\\
\mathrm{Ax} &= \mathrm{Ax}_1 \cup \mathrm{Ax}_2 \cup \mathrm{Ax}_3 \cup \mathrm{Taut}\,.
\end{aligned}$$

5.2.14 Theorem. The set Ax is PR.

5.2.15 Remark. By a theory we mean a 1st order theory on some finite langage, assuming that we have coded fomulas in the same fashion as for $\mathcal{L}_0$.

5.2.16 Definition (Recursive theory, decidable theory)**.**

- A theory T is called **recursive** if the set of codes $\{\sharp\phi \mid \phi \in T\}$ is recursive.
- A theory T is called **decidable** if

$$\mathrm{Th}(T) = \{\sharp\phi \mid T \vdash \phi, \text{ and } \phi \text{ is closed}\}$$

is PR.

5.2.17 Examples.

1. The theory $\emptyset$ is recursive and $\mathrm{Th}(\emptyset)$ is the set of clodes valid formulas.
2. $\mathcal{P}_0$ and $\mathcal{P}$ are recursive.
3. The theory $\mathcal{P}_0 \cup \{\underline{0} = \underline{1}\}$ is decidable.

5.3 Coding proofs

Define

$$\begin{aligned}
\Omega : \mathbb{N}^{<\omega} &\longrightarrow \mathbb{N}\\
\varepsilon &\longmapsto 1\\
(x_0, \ldots, x_k) &\longmapsto \prod_{i=0}^{k} \pi(i)^{x_i}.
\end{aligned}$$

Let $\delta : \mathbb{N}^2 \longrightarrow \mathbb{N}$ be the PR function defined by

$$\delta(i, x) = \mu y \leq x\ \pi(i)^{y+1} | x.$$

We have $\delta(i, \Omega(x_0, \ldots, x_k)) = x_i$, $\forall 0 \leq i \leq k$. Define the PR function

$$\begin{aligned}
\mathrm{lg} : \mathbb{N} &\longrightarrow \mathbb{N}\\
x &\longmapsto \begin{cases} 1 + \max\{i \mid \pi(i) | x\} & \text{if } x > 1\\ 0 & \text{if } x \leq 1. \end{cases}
\end{aligned}$$

Remark that $\mathrm{lg}(\Omega(x_0, \ldots, x_k)) = k + 1$ if $x_k \neq 0$.

5.3.1 Notation. For $\langle\phi_0, \ldots, \phi_k\rangle = p$ a finite sequence of formulas, we note

$$\sharp\sharp p = \Omega(\sharp\phi_0, \ldots, \sharp\phi_k).$$

5.3.2 Theorem. Let T be a recursive theory. Then

$$\text{Proof}(T) = \{(m,n) \mid n = \sharp\phi, m = \sharp\sharp p, p \text{ is a proof of } \phi \text{ from } T\}$$

is recursive.

Proof. $(m,n) \in \text{Proof}(T)$ iff

- $\forall i \leq \lg(m)$ we have $\delta(i,m) \in \text{Form}$,
- $\delta(\lg(m) \cdot -1, m) = n$ (the last formula of the proof is ϕ),
- $\forall i < \lg(m)$, one of the following holds :
  - $\delta(i,m) \in \text{Ax}$,
  - $\delta(i,m) \in \{\sharp\psi \mid \psi \in T\}$,
  - $\exists j,k < i$ such that
    $$\delta(j,m) = \alpha_3(\delta(k,m), \delta(i,m), 4)$$
    (modus ponens),
  - $\exists j < i$, $\exists p < m$ such that $\delta(i,m) = \alpha_3(\delta(m,j), p, 6)$.

□

5.3.3 Corollary. If T is a recursive theory, then $\text{Th}(T)$ is recursively enumerable.

Proof. We have $\text{Th}(T) = \{n \mid \exists m \text{ such that } (n,m) \in \text{Proof}(T)\}$. □

5.3.4 Corollary. If T is recursive and complete theory, then T is decidable.

Proof. By previous corollary, $\text{Th}(T)$ is recursively enumerable. We show that $\mathbb{N} \setminus \text{Th}(T)$ is also recursively enumerable. $\forall x \in \mathbb{N}$, $x \notin \text{Th}(T)$ iff $x \notin \Omega_3$ (x doesn't code a closed formula) or $\alpha_3(x,0,1) \in \text{Th}(T)$ (the code of the negation of the formula coded by x is in $\text{Th}(T)$), since T is complete. □

5.3.5 Theorem. Assume T is recursive, consistant ($T \nvdash_c \bot$), and that $\mathcal{P}_0 \subseteq T$. Then T is not decidable.

Proof. Towards a contradiction, assume that T is decidable. We consider

$$\Theta = \{(m,n) \mid m = \sharp\Phi[x_0] \text{ where } x_0 \text{ is the only (if any) } ???\}.$$

Since T is decidable, Θ is decidable also because

1. one can decide wether m is the Gödel number of a formula $\Phi[x_0]$,
2. one can decide wether $\text{Subst}_f(0, \sharp\underline{n}, m) \in \text{Th}(T)$.

Therefore $B = \{n \in \mathbb{N} \mid (n,n) \notin \Theta\}$ is recursive. By the representation theorem, B is representable by a formula $\Phi_B[x_0]$, i.e. $\forall n \in \mathbb{N}$

- if $n \in B$, then $\mathcal{P}_0 \vdash \Phi_B[\underline{n}]$,
- if $n \notin B$, then $\mathcal{P}_0 \vdash \neg\Phi_B[x_0]$.

Consider $g = \sharp\Phi_B[x_0]$. We see that

- if $g \in B$, then $\mathcal{P}_0 \vdash \Phi_B[x_0]$, so $(g,g) \notin \Theta$, which implies $T \nvdash \Phi_B[x_0]$. But $\mathcal{P}_0 \subseteq T$, so it is a contradiction.
- if $g \notin B$, then $\mathcal{P}_0 \vdash \neg\Phi_B[x_0]$, then also $(g,g) \in \Theta$ which implies $T \vdash \Phi_B[x_0]$, another contradiction.

□

5.3.6 Corollary (Gödel's first incompleteness theorem)**.** The set

$$S = \{\sharp\Phi \mid \Phi \text{ is a closed universaly valid formula}\}$$

is not recursive.

Proof. Since $\mathcal{P}_0 = \{A_1, \ldots, A_7\}$ is finite, it is equivalent to $\Phi_{\mathcal{P}_0} = \bigwedge_{i=1}^{7} A_i$. The formula $\Phi_{\mathcal{P}_0} \to \Phi$ belongs to S iff $\mathcal{P}_0 \vdash \Phi$ (by Gödel's completeness theorem). Therefore, decidability of the membership in S implies decidability of $\mathcal{P}_0$. □

5.3.7 Theorem (Gödel's second incompleteness theorem)**.** Let T be a consistant, recursive theory such that $\mathcal{P} \subseteq T$. Then T doesn't prove its own consistency.

Proof. We define

$$\begin{aligned} P &= \text{Proofs} \\ &= \{(a,b) \mid a = \sharp\Phi, \text{ a closed formula, and } b \text{ is the Gödel number of a proof of } \Phi \text{ in } T\}, \\ P_0 &= \text{Proofs}_0 \\ &= \{(a,b) \mid a = \sharp\Phi, \text{ a closed formula, and } b \text{ is the Gödel number of a proof of } \Phi \text{ in } \mathcal{P}_0\}. \end{aligned}$$

Both Proofs and Proofs_0 are recursive, so representable by $P[-,-]$ and $P_0[-,-]$, formulas of $\mathcal{L}_0$. Define

$$\text{Cons}(T) : \neg\exists x_0 \; P(\sharp(\underline{0} = \underline{1}), x_0).$$

We have that $T \vdash \text{Cons}(T)$ iff T is consistant. Notice that if $\exists x_0 \in \mathbb{N}$ such that $\mathbb{N} \models P(\sharp(\underline{0} = \underline{1}), x_0)$, then $\mathbb{N} \models \neg\,\text{Cons}(T)$, and so $\exists n \in \mathbb{N}$ which is the Gödel numbering of a formal proof of the formula $\underline{0} = \underline{1}$ inside T. However, on can have $\mathcal{M} \models \mathcal{P}$ and $\mathcal{M} \models \neg\,\text{Cons}(T)$, together with T being consistant. The reason is that if one takes the "integer" (that can be non standard) in $\mathcal{M}$ that witnesses the proof of $\underline{0} = \underline{1}$, the integer being non standard will not lead to the proof of $\underline{0} = \underline{1}$. □

To proceed further, we need P and P_0 to statisfies additional properties :

1. $\vdash \forall x_0 \forall x_1 \; (P_0(x_0, x_1) \to P(x_0, x_1))$,
2. P and P_0 are Σ_1^0 formulas,
3. if Φ is a closed Σ_1^0 formula, then

$$\mathcal{P} \vdash \Phi \to \exists x_1 \; P_0(\sharp\Phi, x_1).$$

Point 1. comes from the proof of the representation theorem : when we proved it, we proved something stringer : every recursive total function is representable by a Σ_1^0 formula.

5.3.8 Lemma. Let $\mathcal{N}$ be a $\mathcal{L}_0$-structure and $\mathcal{M}$ be a final extension of $\mathcal{N}$. Let $\Phi[x_1, \ldots, x_p]$ be a Σ_1^0 formula, and $a_1, \ldots, a_p \in |\mathcal{N}|$. Then

$$\mathcal{N} \models \Phi[a_1, \ldots, a_p] \quad \text{implies} \quad \mathcal{M} \models \Phi[a_1, \ldots, a_p].$$

Proof. We show that the set of all formulas $\Psi[x_1, \ldots, x_p]$ satisfying the proposition's conditions contains all Σ_1^0 formulas. Remark that

- this set contains all quantifiers-free formulas,
- this set is closed under $\wedge$ and $\vee$,
- this set is closed under $\exists$ (because $\mathcal{N} \preceq \mathcal{M}$),
- this set is closed under bounded universal quantifiers (because $\mathcal{M}$ is a final extension of $\mathcal{N}$).

□

Define

$$\mathcal{P}_1 = \mathcal{P}_0 \cup \{\Phi \to \exists x_1 \; P_0(\sharp\Phi, x_1) \mid \Phi \text{ is a closed formula}\}.$$

5.3.9 Lemma. If $\Psi \in \mathcal{P}_1$, then $\mathcal{P} \vdash \Psi$.

5.3.10 Lemma. If T is a consistant recursive theory that proves all formulas of $\mathcal{P}_1$, then T does not prove its own consistency.

5.3.11 Proposition. Let Φ be a closed Σ_1^0 formula of $\mathcal{L}_0$. Then

$$\mathbb{N} \models \Phi \to \exists x_1 \; P_0(\sharp\Phi, x_1).$$

Proof. Assume Φ holds in $\mathbb{N}$. We show that $\mathcal{P}_0 \vdash \Phi$. For this, weshow that Φ holdsin every model of $\mathcal{P}_0$. □

Index

— Symbols —

Reg 8
Δ_n^0 25
Π_n^0 25
Σ_ε 8
Σ_n^0 25
$\dot{-}$ 18
ε 8
$\sharp\sharp p$ 37
$\sharp$ 34

— A —

Accepting state 13
Arithmetical hierarchy 25

— B —

Blank symbol 13
Bounded minimisation 20

— C —

Case study 20
Composition 17, 22
Concatenation 9
Configuration 13
Control state 13
Counter 11
    automaton 11

— D —

Decidable
    theory 37
Decider 14
DFA 7, 8

— E —

Enumerator 15

— F —

Final extension 31
Finite automaton
    deterministic 7
    non deterministic 8
$\mathcal{F}_p$ 22

— G —

Gödel
    number
        of a formula 34
        of a term 34
Goodstein sequence 5

— I —

Induction 17
Induction scheme 29
Initial segment 31
Initial state 13

— J —

Jump 27

— K —

k-tape Turing machine 14

— L —

$\mathcal{L}$ 7, 8, 13

— M —

μ 20
Multitape Turing machine 14

— N —

Non deterministic
    Turing machine 14

— P —

Partial function 18
PR 18
Predicate 21
Primitive recursive
    function 17
    set 18
Projection 17
Proof 35
Pushdown automaton 12

— R —

Recursive
    theory 37
Recursively enumerable langage 15
Regular expression 10
Regular lagage 8
Rejecting state 13
Representable
    function 31
    set 31
R_n 17
Rule 12

— S —

Stack 11
    alphabet 12
Star operation 9
State 7, 8, 13
    accepting 7, 8
    initial 7, 8
Successor function 17

— T —

Terminal 12
Total function 18
Transition function 13
Turing
    decidable 14
    recognizable 14
Turing computable function 18
Turing machine 13
    with oracle 27

— U —

Universal Turing machine 15

— V —

Variable 12
Variable substitution 19

— W —

Working alphabet 13