In this chapter we prove Gödel’s 1’st Non-Completeness Theorem:
If $\Gamma$ is recursive, consistent and $\Gamma \vdash PA$ then $\Gamma$ is not complete.

First we code every expression $k \in K(L)$ and formulae $\varphi \in F(L)$ with a natural number $\nu(k)$ and $\nu(\varphi) \in \mathbb{N}$, then we prove Gödel’s 1st Non-completeness Theorem. Let us emphasize in advance that not the technical details of the coding $\nu$ are important but only the existence of such a coding! In other words, other coding functions also would do. In fact, every coding function of finite sequences (strings) with natural numbers are usually called ”Gödel-coding”.

Before we need the notion and properties of primitive and general recursive functions.

1. Primitive recursive functions

For the definitions and explanations of primitive/partial recursive and recursive-enumerable functions and sets please refer to the 3rd Part of the green book ”Diszkrét matematika és az algoritmuselmélet alapjai” by I.Szalkai (in Hungarian).

Definition: Any set $A \subseteq \mathbb{N}$ is recursive (decidable), if its characteristic function $\chi_A : \mathbb{N} \rightarrow \{0, 1\}$ is recursive, i.e. there is a T.M. which decides "$n \in A$" for every $n \in \mathbb{N}$.

2. Gödel-coding

Lemma 1: There is a primitive recursive function

$$\beta : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$$

with the property: for every number $n \in \mathbb{N}$ and every finite sequence of natural numbers length of $n$

$$\vec{a} = (a_1, \ldots, a_n) \in \mathbb{N}^n$$

there is a $c \in \mathbb{N}$, the code of $\vec{a}$, such that

$$\beta(c, 0) = n \quad \text{and} \quad \beta(c, i) = a_i \quad (i \leq n) \quad \Box$$

Definition 2: Clearly $\beta$ induces a coding function

$$s : \mathbb{N}^* \rightarrow \mathbb{N}$$

$$s(\vec{a}) = c \quad \Box$$

(For example, we can have: $s(a_1, \ldots, a_n) := p_1^{\alpha_1+1} \cdot \ldots \cdot p_n^{\alpha_n+1}$ where $p_i$ is the $i$ ‘th prime number.)
Lemma 3: The set of codes

\[ C := \{ c \in \mathbb{N} : c \text{ is the code for some } \overline{a} \in \mathbb{N}^* \} \]

is primitive recursive (i.e. the statement "\( c \in \mathbb{N} \) is a code for a finite sequence" is primitive recursive decidable). \( \square \)

Lemma 4: The decoding function

\[ B : C \to \mathbb{N}^* \]

is primitive recursive. \( \square \)

Lemma 5: The predicate "\( B(c) \) is an initial segment of \( B(d) \)" is primitive recursive, too. (We mean that \( B(c) = (a_1, ..., a_n) \) and \( B(d) = (a_1, ..., a_m) \) where \( n \leq m \).) \( \square \)

Lemma 6: The function \( \ell : C \to \mathbb{N} \) where \( \ell(c) \) is the length of the sequence \( B(c) \) (coded by \( c \)) is primitive recursive. \( \square \)

Note 7: Since \( C \subseteq \mathbb{N} \) is primitive recursive and the functions \( s : \mathbb{N}^* \to C \), \( B : C \to \mathbb{N}^* \) both are bijective (one-to-one and onto) and primitive recursive, we do not distinguish the sequences \( \overline{a} \in \mathbb{N}^* \) and their codes \( c = s(\overline{a}) \in \mathbb{N} \) at all, in what follows.

Now we code all the expressions and formulas.
Let \( \mathcal{L} = (f_1, ..., f_n, P_1, ..., P_m) \) be any first order (fixed) language. Clearly \( \mathcal{L} \) contains also the symbols \( \land, \lor, \exists, (,) \), and the variable symbols \( x_1, ..., x_i, ... \).

Let first

\[ \nu_0 : \mathcal{L} \to \mathbb{N} \]

be any fixed bijection. Extend then \( \nu_0 \) to \( K(\mathcal{L}) \cup F(\mathcal{L}) \) as (for example):

Definition 8: (i) \( \nu(x_0) := s(\nu_0(x_i)) \) if \( k = x_i \) is a 0-order expression,

(ii)

\[ \nu(k) := s(\nu_0(f_1), \nu_0(), \nu(k_1), \nu_0(), \nu(k_2), \nu_0(), ..., \nu_0(), \nu(k_m), \nu_0()) \]

if \( k = f_1(k_1, ..., k_m) \) is a \( \ell + 1 \)-order expression,

(iii)

\[ \nu(\varphi) := s(\nu_0(P_1), \nu_0(), \nu(k_1), \nu_0(), \nu(k_2), \nu_0(), ..., \nu_0(), \nu(k_v), \nu_0()) \]

if \( \varphi = P_1(k_1, ..., k_v) \) is a 0-order formula,

(iv)

\[ \nu(\varphi) : = s(\nu_0(\varphi), \nu(\psi)) \]

\[ \nu(\psi \lor \vartheta) : = s(\nu(\psi), \nu_0(\lor), \nu(\vartheta)) \]

\[ \nu(\exists x_i \psi) : = s(\nu_0(\exists), \nu_0(x_i), \nu(\psi)) \]

for the \( \ell + 1 \)-order expressions \( \varphi \), \( \psi \lor \vartheta \) and \( \exists x_i \psi \). \( \square \)

Please observe and understand the trivial base idea of coding all expressions and formulas: eg. for coding the expression \( k = f_1(k_1, ..., k_m) \) we just code the sequence of the codes of the components of \( k \) : \( f_1, (, k_1, , , ..., , k_m , ) \). Or, in some more detail: we code the sequence \( \nu_0(f_1), \nu_0(), \nu(k_1), \nu_0(), \nu(k_2), \nu_0(), ..., \nu_0(), \nu(k_m), \nu_0() \), as it is written in the definition above. Further, please take care of when to use \( \nu_0 \) and when \( \nu \).

Let us emphasize again, that not the details of the coding

\[ \nu : K(\mathcal{L}) \cup F(\mathcal{L}) \to \mathbb{N} \]
but the existence of such coding is important. Moreover, the main aim of such codings is: to represent and examine formulas, proofs, axiom systems (everything) with natural numbers.

**Theorem 9:** The function $\nu: K(\mathcal{L}) \cup F(\mathcal{L}) \to \mathbb{N}$ is one-to-one. □

Now we go on. All the proofs below are omitted because of their simplicity, unless it is stated otherwise. Since

$$\text{Im}(\nu) \subseteq \text{Im}(s) = C,$$

we can consider the following predicates (questions):

**Theorem 10:** The following predicates and functions on $C$ are primitive recursive $(c, e, x, ..., \in C)$:

- $\text{Var}(c) := " c$ is a $\nu$-code for a variable "$
- \text{Kif}(c) := " \ell \text{ is an } \nu\text{-expression}"$
- $\text{Fml}(c) := " \ell \text{ is a } \nu\text{-formula}"$
- $\text{Free}(e, x) := $ " $Fml(e)$ and $\text{Var}(x)$ and $x$ is a $\nu$-code for a free variable of the formula (coded by $e$)"
- $\text{Subst}(d, x, \ell) :=$ the code for the formula, obtained by the substitution $\varphi_{x_m}(k)$ where $d = \nu(\varphi)$, $x = \nu(x_m)$, $\ell = \nu(k)$ and (of course) $\text{Kif}(\ell)$, $\text{Fml}(d)$ and $\text{Var}(x)$ yield,
- $\text{AllSubst}(h, d, x, \ell) := $ " the substitution $h=\text{Subst}(d, x, \ell)$ is an allowed one",
- $\text{LogAx}(g) := " g$ is a $\nu$-code for a logical axiom",
- $\text{DedRul}(u, w) := " (\vartheta | \varphi) \text{ is a deduction rule where } u = \nu(\vartheta) \text{ and } w = \nu(\varphi)"
- $\text{DedRul}(u, v, w) := " (\vartheta, \tau | \varphi) \text{ is a deduction rule where } u = \nu(\vartheta), v = \nu(\tau) \text{ and } w = \nu(\varphi)"
- $\text{Biz}_T(a, b) := "$ b is a $\nu$-code of a proof (sequence of formulas connected with & and deduction rules) from $\Gamma$ of the formula coded by $a$"

Let us note that $\Gamma$ above is a fixed axiom system, and moreover the set

$$\{\nu(\gamma) : \gamma \in \Gamma\} \subseteq C$$

must be primitive recursive.

**Definition 11:** $\text{Köv}_\Gamma(a) := \exists b \text{Biz}_T(a, b)$ (a is provable from $\Gamma$). □

**Definition 12:** $\text{Köv}_\Gamma := \{a \in C : \text{Köv}_\Gamma(a)\}$ (the set of consequences of $\Gamma$). □

Please keep in mind that $\Gamma$ is a fixed axiom system, and $\text{Köv}_\Gamma$ is the set of $(\nu$-codes of) the formulas which are provable from $\Gamma$ ("corollaries of $\Gamma"$). This is not the set of $(\nu$-codes of) formulas decidable by $\Gamma$, but ... think a little bit on this question, please.

In general, $\text{Köv}_\Gamma$ is even not general recursive (see 15, 16 below).

**Definition 13:** Any set of formulas $F$ is recursive if and only if its characteristic function $\chi_F : C \to \{0, 1\}$ is recursive. □
The following theorem reveals the real importance and strength of PA, Peano’s Axiom system for arithmetic): we can talk about recursive sets and formulas inside $\Gamma$:

**Theorem 14:** (Representation Theorem for Recursive Sets) For any recursive set $Q \subset \mathbb{N}^n$ (predicate over $\mathbb{N}^n$) there is a formula $\varphi = \varphi_Q \in F(\mathcal{L}_{PA})$ such that:

- if $\overline{b} \in Q$ then $PA \vdash \varphi_Q(\overline{b})$,
- if $\overline{b} \notin Q$ then $PA \vdash \varphi_Q(\overline{b})$

for every $\overline{b} = (b_1, ..., b_n) \in \mathbb{N}^n$.

**Proof:** Easy but boring a bit: using the inductive definition of recursive functions and sets (basic functions, opertors, ...) we can actually construct the formula $\varphi_Q$ itself (see the 3rd Part of the green book ”*Diszkrét matematika és az algoritmuselmélet alapjai*” by I. Szalkai).

14.b.) **Remark**, that we can not write ”if and only if” in none of the statement lines of the previous Theorem.

Further, in the case $\Gamma \vdash PA$ (possibly after a neccesary conservative extension) we can replace $PA$ by $\Gamma$ in the above Theorem.

### 3. The 1st Non-completeness Theorem

**Definition:** $\Gamma$ is decidable if for every $\varphi$ the question ”$\Gamma \vdash \varphi$” can be decided.

**Statement 15:** $\Gamma$ is decidable if and only if $\text{Köv}_\Gamma$ is recursive. □

**Theorem 16:** (A. Church) If $\Gamma \vdash PA$ and $\Gamma$ is consistent then $\Gamma$ is not decidable.

**Proof:** Suppose on indirect way that $\text{Köv}_\Gamma$ is recursive. Then the predicates

$$P(a, b) := \text{Köv}_\Gamma(Subst(a, x_0, b))$$

and

$$Q(b) := \varphi \text{ with free variable } x_0$$

both are recursive, too. Now let the formula $\varphi \in F(\mathcal{L}_\Gamma)$ represent $Q$ as in Theorem 14. Clearly $V(\varphi) = \{x_0\}$, i.e. $\varphi$ has exactly one free variable.

This means, for every $b \in C$ :

$$Q(b) = \uparrow \text{ if and only if } PA \vdash \varphi_{x_0}[b] = \uparrow.$$ Denote $\nu$ the $\nu$-code for $\varphi$ : $\nu(\varphi) = a$.

Now either $Q(a) = \uparrow$ or $Q(a) = \downarrow$ we reach to a contradiction:

- if $Q(a) = \uparrow$ then $\Gamma \vdash \varphi_{x_0}[a]$ then $\|P(a, a)$ then $\|\text{Köv}_\Gamma(Subst(a, x_0, a))$
  - then $\Gamma \not\vdash \varphi_{x_0}[a]$ contradiction,

- if $Q(a) = \downarrow$ then $\Gamma \vdash \varphi_{x_0}[a]$ then $\Gamma \not\vdash \varphi_{x_0}[a]$ and $P(a, a)$ then $\text{Köv}_\Gamma(Subst(a, x_0, a))$
  - then $\Gamma \vdash \varphi_{x_0}[a]$ contradiction. □

Lemmas 17 and 18 below are, in some sense, the opposite of Theorem 16.
Lemma 17: If \( R, Q \subseteq \mathbb{N}^2 \) are recursive sets and \( P \subseteq \mathbb{N} \) is any subset, such that for each \( a \in \mathbb{N} \)

\[
P(a) \text{ if and only if } \exists u \ Q(a, u) \\
\neg P(a) \text{ if and only if } \exists v \ R(a, v)
\]

then \( P \) is recursive. \( \square \)

Lemma 18: \( \) If \( \Gamma \) is complete then it is decidable.

Proof: For any complete axiom system \( \Gamma \) we have

\[
K\bar{\nu}_\Gamma(a) \text{ if and only if } \exists u \ Biz_\Gamma(a, u)
\]
\[
\neg K\bar{\nu}_\Gamma(a) \text{ if and only if } \exists v \ Biz_\Gamma(\neg a, v)
\]

So \( K\bar{\nu}_\Gamma \) must be recursive by Lemma 17, and use Statement 15. \( \square \)

Theorem 19: \( \) (Gödel’s 1’st Non-Completeness Theorem)

If \( \Gamma \vdash PA \) and \( \Gamma \) is consistent then \( \Gamma \) is not complete.

Proof: Lemma 18 contradicts to Church’s Theorem 16. \( \square \)

Note that Gödel’s Theorem 19. is a strenghtening of Church’s Theorem 16.