Gödel's 1st Non-completeness Theorem

https://math.uni-pannon.hu/~szalkai/Godel-sk.pdf (Godel-sk-jav.tex, 2019.05.10.)

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

First we code every expression $k \in K(\mathcal{L})$ and formulae $\varphi \in F(\mathcal{L})$ 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 ν 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* $A \subsetneqq \mathbb{N}$ is *recursive* (decidable), if its charasteristic function $\chi_A : \mathbb{N} \to \{0,1\}$ 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

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

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

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

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

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

Definition 2: Clearly β induces a coding function

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

$$s(\overrightarrow{a}) = c \; . \qquad \square$$

(For example, we can have: $s(a_1, ..., a_n) := p_1^{a_1+1} \cdot ... \cdot p_n^{a_n+1}$ where p_i is the i 'th prime number.)

Lemma 3: The *set* of codes

$$C := \{c \in \mathbb{N} : c \text{ is the code for some } \overrightarrow{a} \in \mathbb{N}^*\}$$

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 $C \subsetneqq \mathbb{N}$ 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 $\overrightarrow{a} \in \mathbb{N}^*$ and their codes $c = s(\overrightarrow{a}) \in \mathbb{N}$ at all, in what follows.

Now we code all the expresssions 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 $\rceil$, $\vee$, $\exists$, (,) , **,** and the variable symbols x_1 , ... x_i ... .

Let first

$$\nu_0 : \mathcal{L} \to \mathbb{N}$$

be any fixed bijection. Extend then ν_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)

$$\nu(k) := s\;(\;\nu_0(\mathbf{f}_i), \nu_0(\mathbf{(}), \nu(\mathbf{k}_1), \nu_0(\mathbf{,}), \nu(\mathbf{k}_2), \nu_0(\mathbf{,})\;, ..., \nu_0(\mathbf{,}), \nu(k_\mu), \nu_0(\mathbf{)})\;)$$

if $k = f_i(k_1, ..., k_\mu)$ is a $\iota + 1$ -order expression,

(iii)

$$\nu(\varphi) := s\;(\;\nu_0(\mathbf{P}_j), \nu_0(\mathbf{(}), \nu(\mathbf{k}_1), \nu_0(\mathbf{,}), \nu(\mathbf{k}_2), \nu_0(\mathbf{,})\;, ..., \nu_0(\mathbf{,}), \nu(k_\nu), \nu_0(\mathbf{)})\;)$$

if $\varphi = P_j(k_1, ..., k_\nu)$ is a 0 -order formula,

(iv)

$$\begin{aligned}
\nu(\;\rceil\boldsymbol{\psi}\;) \;&:= s\;(\;\nu_0(\;\rceil\;),\; \nu(\boldsymbol{\psi})\;)\;, \\
\nu(\;\boldsymbol{\psi} \vee \boldsymbol{\vartheta}\;) \;&:= s\;(\;\nu(\boldsymbol{\psi}), \nu_0(\vee), \nu(\boldsymbol{\vartheta}))\;, \\
\nu(\;\exists \mathbf{x}_i \boldsymbol{\psi}\;) \;&:= s\;(\;\nu_0(\exists), \nu_0(\mathbf{x}_i), \nu(\boldsymbol{\psi}))\;,
\end{aligned}$$

for the $\iota + 1$ -order expressions $\rceil\psi$, $\psi \vee \vartheta$ 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 $k = f_i(k_1, ..., k_\mu)$ we just code the sequence of the codes of the components of k : f_i , (, k_1 , **,** , ... , k_μ ,) . Or, in some more detail: we code the sequence $\nu_0(\mathbf{f}_i)$, $\nu_0(\mathbf{(})$, $\nu(\mathbf{k}_1)$, $\nu_0(\mathbf{,})$, $\nu(\mathbf{k}_2)$, $\nu_0(\mathbf{,})$, ..., $\nu_0(\mathbf{,})$, $\nu(k_\mu)$, $\nu_0(\mathbf{)})$, as it is written in the definition above. Further, please take care of when to use ν_0 and when ν .

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. $\square$

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

$$\mathrm{Im}(\nu) \subset \mathrm{Im}(s) = C \quad ,$$

we can consider the following predicates (questions) :

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

Var(c) := " c is a ν-code for a variable " ,

Kif(c) := " — ' ' — expression " ,

Fml(c) := " — ' ' — formula " ,

Free(e,x) := " Fml(e) and Var(x) and x is a ν-code for a free variable of the formula (coded by) e " ,

Subst(d,x,ℓ) := the code for the formula, obtained by the substitution $\varphi_{x_m}(k)$ where where $d = \nu(\varphi)$, $x = \nu(x_m)$, $\ell = \nu(k)$ and (of course) Kif(ℓ), Fml(d) and Var(x) yield,

AllSubst(h,d,x,ℓ) := " the substitution h=Subst(d,x,ℓ) is an allowed one " ,

LogAx(g) := " g is a ν-code for a logical axiom **" ,**

DedRul(u,w) := " $(\vartheta|\eta)$ is a deduction rule where $u = \nu(\vartheta)$ and $w = \nu(\eta)$ " **,**

DedRul(u,v,w) := " $(\vartheta, \tau|\eta)$ is a deduction rule where $u = \nu(\vartheta)$, $v = \nu(\tau)$ and $w = \nu(\eta)$ " **,**

Biz$_\Gamma$(a,b) := " b is a ν -code of a proof (sequence of formulas connected with & and deduction rules) from Γ of the formula coded by a " . $\square$

Let us note that Γ above is a *fixed* axiom system, and moreover the set

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

must be primitive recursive.

Definition 11: **Köv$_\Gamma$(a)**:= $\exists b \; Biz_\Gamma(a, b)$ (a is provable from Γ). $\square$

Definition 12: **Köv$_\Gamma$** := $\{a \in C : K\ddot{o}v_\Gamma(a)\}$ (the set of consequences of Γ). $\square$

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

In general, 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 charasteristic function $\chi_F : C \to \{0, 1\}$ is recursive. $\square$

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* Γ :

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:

$$\text{if} \quad \overrightarrow{b} \in Q \quad \text{then} \quad PA \vdash \varphi_Q\left(\overrightarrow{b}\right) ,$$

$$\text{if} \quad \overrightarrow{b} \notin Q \quad \text{then} \quad PA \vdash \rceil\, \varphi_Q\left(\overrightarrow{b}\right)$$

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

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

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 Γ in the above Theorem.

3. The 1st Non-completeness Theorem

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

Statement 15: Γ is decidable if and only if Köv_Γ is recursive. $\square$

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

Proof: Suppose on indirect way that Köv_Γ is recursive. Then the predicates

$$P(a, b) := K\ddot{o}v_\Gamma\left(Subst(a, x_0, b)\right)$$

and

$$Q(b) := \ \rceil\ P(b, b)$$

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. φ has exactly one free variable.

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

$$Q(b) = \uparrow \quad \textit{if and only if} \quad PA \vdash \varphi_{x_0}[b] = \uparrow \ .$$

Denote a the ν -code for φ : $\nu(\varphi) = a$.

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 $\rceil P(a, a)$ then $\rceil K\ddot{o}v_\Gamma\left(Subst(a, x_0, a)\right)$
then $\Gamma \nvdash \varphi_{x_0}[a]$ contradiction,

if $Q(a) = \downarrow$ then $\Gamma \vdash \rceil \varphi_{x_0}[a]$ then $\Gamma \nvdash \varphi_{x_0}[a]$ and $P(a, a)$ then $K\ddot{o}v_\Gamma\left(Subst(a, x_0, a)\right)$
then $\Gamma \vdash \varphi_{x_0}[a]$ contradiction. $\square$

Lemmae 17 and 18 below are, in some sense, the opposite of Theorem 16.

Lemma 17: If $R, Q \subseteqq \mathbb{N}^2$ are recursive sets and $P \subseteqq \mathbb{N}$ is any subset, such that for each $a \in \mathbb{N}$

$$\begin{array}{lcl} P(a) & \textit{if and only if} & \exists u\, Q(a,u) \\ \rceil P(a) & \textit{if and only if} & \exists v\, R(a,v) \end{array}$$

then P is recursive. $\square$

Lemma 18: If Γ is complete then it is decidable.

Proof: For any complete axiom system Γ we have

$$\begin{array}{lcl} K\ddot{o}v_\Gamma\,(a) & \textit{if and only if} & \exists u\, Biz_\Gamma(a,u) \quad , \\ \rceil K\ddot{o}v_\Gamma\,(a) & \textit{if and only if} & \exists v\, Biz_\Gamma(\,\rceil a, v) \quad . \end{array}$$

So $K\ddot{o}v_\Gamma$ 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* Γ *is consistent* then Γ *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.