The Recursion theorem

P program $\langle P \rangle$: code of P

Two most practical

Obtain $\langle P \rangle$: allocate a program to access its own source code.

Run $\langle P \rangle$

Ex1 P_1

Obtain $\langle P_1 \rangle$, Output $\langle P_1 \rangle$

EXAMPLE of a self-reproducing program in English:

Write the following sentence twice, the second time in quotes.

"Write the following sentence twice, the second time in quotes."

Ex2 P_2 input: word in Σ^*

if $w \in$ the language

else { obtain $\langle P_2 \rangle$,

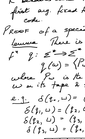
$\langle P_2 \rangle$ = $\text{fail}(w)$

: if you get n return $n+1$ }

P_2 recursively computes the length of w.

THM Let T be a TM that computes $t: \Sigma^* \times \Sigma^* \to \Sigma^*$. Then there is a TM, R, which computes $r: \Sigma^* \to \Sigma^*$ where $r(w) = t(\langle R \rangle, w)$.

How does R behave?



R behaves like T but with the first arg fixed to be its own source code.

Proof of a special case:

Lemma There is a total comp $f: \Sigma^* \to \Sigma^*$ s.t

$f(w) = \langle P_w \rangle$

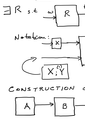
where P_w is the TM that writes w on its tape & stops.

ex 1. $\delta(s_1, w) = (s_1, r, R)$

$\delta(s_1, u) = (s_1, 0, R)$

$\delta(s_1, v) = (s_1, 0, R)$

$\delta(s_1, w) = (s_1, w, S)$



B reads output of A

A;B

B appends its input to be of the form $\langle M \rangle$

B outputs $\langle P_{\langle M \rangle}; M \rangle$

[LEMMA]

B outputs a string

$\langle B \rangle$ is just a string.

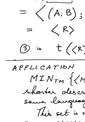
A in $P_{\langle B \rangle}$

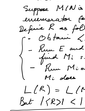
output $\langle P_{\langle B \rangle}; B \rangle = \langle A; B \rangle$

output of A is $\langle A; B \rangle$

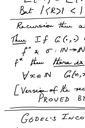
General case of the recursion thm:

$\forall T \; \exists r$ s.t

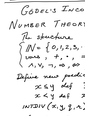




Notation

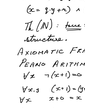


CONSTRUCTION OF R gives T

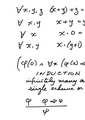


$\langle A; B \rangle$; σ^* T

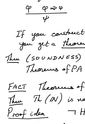
A is $P_{\langle B_{\sigma^*} T \rangle}$



B is



output of B is a description of



① output of A : $\langle B_{\sigma^*} T \rangle$

② output of B :

$$\langle P_{\langle B_{\sigma^*} T \rangle}; \langle B_{\sigma^*} T \rangle \rangle$$
$$= \langle \langle P_{\langle B_{\sigma^*} T \rangle}; B \rangle ; \sigma^* T \rangle$$
$$= \langle \langle A; B \rangle ; \sigma^* T \rangle$$
$$= \langle R \rangle$$

③ is $t(\langle R \rangle, w)$

APPLICATION

$MIN_{TM} = \{\langle M \rangle \mid$ No TM with a shorter description recognizes same language$\}$

This set is not CE

Suppose MIN is CE & E enum enumerates its L.

Define R as follows:

- Obtain $\langle R \rangle$
- Run E and wait until you find M: $|\langle M \rangle| > |\langle R \rangle|$
- Run M on w & its subroutine M_1 states

$L(R) = L(M_1)$

But $|\langle R \rangle| < |\langle M_1 \rangle|$ ⊗

Recursion thm as a fixed point thm

Thm If $G(\cdot,\cdot)$ is a total numerical f & $\sigma: \mathbb{N} \to \mathbb{N}$ is a total computable f then there is some n s.t.

$\forall x \in \mathbb{N} \quad G(n, x) = G(\sigma(n), x)$

[Version of the recursion thm.]

PROVED BY KLEENE]

GODEL'S INCOMPLETENESS THM

NUMBER THEORY

The structure

$\{\mathbb{N}, +, \{0, 1, 2, 3, \cdots\}, \{\text{mult}, +, =, \cdot, \forall, \exists, \wedge, \vee, \neg, \Rightarrow, (,)\}$

Define new predicates

$x \leq y$ def $\exists z \; x+z = y$

$x < y$ def $x \leq y \wedge x \neq y$

$\text{INDIV}(x, y, z) \stackrel{\text{def}}{=}$

$(x = y \vee y = z) \wedge (x < y)$

$\mathcal{TL}(N)$: have statements about its structure.

AXIOMATIC FRAMEWORK

PEANO ARITHMETIC : FOL +

$\forall x \; \neg(x+1) = 0$

$\forall x, y \quad (x+1) = (y+1) \Rightarrow x = y$

$\forall x \quad x + 0 = x$

$\forall x, y, z \quad (x+y) + z = x + (y+z)$

$\forall x, y \quad x + y = y + x$

$\forall x \quad x \cdot 0 = 0$

$\forall x, y \quad x \cdot (y+1) = x \cdot y + x$

$(\varphi(0) \wedge \forall x \; (\varphi(x) \Rightarrow \varphi(x+1))) \Rightarrow \forall x \; \varphi(x)$

INDUCTION

infinitely many axioms but in a single schema or $\forall \varphi$: 2^{Σ^*} order.

$\varphi \Rightarrow \psi$ $\qquad$ $\varphi(x)$

ψ $\qquad$ $\forall x \; \varphi(x)$

If you construct a proof in PA you get a theorem

Thm (SOUNDNESS)

Theorems of PA $\subset \mathcal{TL}(N)$

FACT Theorems of PA is a CE set.

Thm $\mathcal{TL}(N)$ is not CE

Proof idea $\neg HT \leq_m \mathcal{TL}(N)$

We must construct a formula χ of NT s.t. $(M, w) \in HT \Leftrightarrow \chi$ is true

We can encode valid computations of a Turing m/c in NT.

VALCOMP(M, w, y): you valid computation of M on w

$y \stackrel{\text{def}}{=} \exists$ VALCOMP(M, w, y)

No finite proof system can capture all true statements of NT.

Idea of Godel's Proof:

- Every formula can be encoded as a number.

$\varphi \longrightarrow \ulcorner \varphi \urcorner$

FORMULA $\quad$ Godel number of φ

- Given any proof system, he showed how to define a formula that represents PROVABLE($\ulcorner \varphi \urcorner$).
- For any formula $\psi(x)$ with x a free variable $\exists$ a formula θ s.t. $\theta \Leftrightarrow \psi(\ulcorner \theta \urcorner)$

θ says "my Godel number satisfies ψ".

A proof this lemma:

$\neg$ PROVABLE(x)

We get a formula θ s.t.

$\vdash P \Leftrightarrow \neg$ PROVABLE($\ulcorner P \urcorner$)

P cannot be provable

- P is false ??

Then $\neg$ PROVABLE($\ulcorner P \urcorner$) is false

i.e. PROVABLE($\ulcorner P \urcorner$) is true

so P is provable

by soundness of the proof system

- P is true

then P cannot be provable.

There exists a true statement of NT which cannot be proved in the specific proof system which you are using.

THEOREM

Thm No predicate defined in NT can capture the truth of arithmetic formulas.

Idea: Any arithmetic formula will at some finite level of the arithmetic hierarchy so it cannot capture truth of formulas at a higher level.