1 First order logic and Peano arithmetics

In order to make this presentation more self-contained, we first quickly recap some notions and results from formal logic. In the following, we use results and formalism of first order logic and a standard axiomatisation of arithmetics—Peano arithmetics.

First order logic The syntax of first order logic is determined by a signature $\sigma = \langle \mathcal{C}; \mathcal{F}; \mathcal{P} \rangle$ consisting of constant symbols $\mathcal{C}$, function symbols $\mathcal{F}$ and predicate symbols $\mathcal{P}$. The formulas of first order logic is defined rather standard way by combining constants, free variables, quantifier, logic connectives and braces. However, a priory none of the symbols or formulas have any kind of semantics—only an interpretation can transform syntactic gibberish into a meaningful mathematical proposition.

Definition 1. *An interpretation I of a signature $\sigma = \langle \mathcal{C}; \mathcal{F}; \mathcal{P} \rangle$ consists of nonempty domain $\mathcal{D}$ and a collection of mappings that:*

- *assign a domain element to each constant symbol,*
- *assign a n-ary function $f : \mathcal{U}^n \rightarrow \mathcal{U}$ to each n-ary function symbol,*
- *assign a n-ary predicate $p : \mathcal{U}^n \rightarrow \{0, 1\}$ to each n-ary predicate symbol.*

All well-formed formulas naturally split into three subclasses: *consistent*, *inconsistent* and *valid* formulas. Valid formulas are true in every interpretation. There are several formal techniques (proof systems) that allow to algorithmically derive all valid formulas via infinite computing process, i.e. the set of all valid formulas is recursively enumerable.

In order to use any proof system for deriving some nontrivial mathematical facts, one has to provide a set of axioms $\mathcal{T}$ that define the mathematical structure in question. We slightly abjuse the notation and talk about theory $\mathcal{T}$, as all true stetements are determined by the axiom set $\mathcal{T}$. We say that a formula ϕ follows from the axioms $\mathcal{T}$ iff for all interpretations where axioms $\mathcal{T}$ are true also the formula ϕ is true, and denote it by $\mathcal{T} \models \phi$. If the set of axioms $\mathcal{T}$ is finite, then $\mathcal{T} \models \phi$ iff the formula $\mathcal{T} \supset \phi$ is valid, and thus also provable by the proof system. However, for many practical theories, we use infinite system of axioms $\mathcal{T}$ and latter opens a gap between provable statements and true statements. Therefore, we use notation $\mathcal{T} \vdash \phi$ to state that exist a (finite) proof for ϕ in theory $\mathcal{T}$. Obviously, for any consistent theory $\mathcal{T}$, the fact $\mathcal{T} \vdash \phi$ implies $\mathcal{T} \models \phi$.

Sometimes adding axioms along with constant, function or predicate symbols does not change the set of representable statements and the set of valid statements in the theory. Then we talk about *conservative extensions.* In a way, the conservative extension is just a more convenient reformulation of axioms.

Peano arithmetics A standard signature of arithmetics is $\sigma = \langle \mathsf{0}; +, \cdot; = \rangle$, where $\mathsf{0}$, $\mathsf{1}$ are the only constants, $+$ and $\cdot$ denote standard arithmetical operations, and $=$ stands for equality. The semantics of the Peano arithmetics is captured by the following set PA of axioms

EQUALITY AXIOMS

$\forall x(x = x)$

$\forall x \forall y(x = y \supset y = x)$

$\forall x \forall y \forall z((x = y \wedge y = z) \supset x = z)$

$\forall x \forall y(\phi(\ldots, x, \ldots) \supset \phi(\ldots, y, \ldots))$

SUCCESSOR AXIOMS

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

$\forall x \forall y(x + 1 = y + 1 \supset y = x)$

$(\phi(0) \wedge \forall x(\phi(x) \supset \phi(x + 1)) \supset \forall x \phi(x)$

ADDITION AXIOMS

$\forall x(x + 0 = x)$

$\forall x \forall y(x + (y + 1) = (x + y) + 1)$

MULTIPLICATION AXIOMS

$\forall x(x \cdot 0 = x)$

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

where ϕ can be any well-formed formula in the signature σ. However, the formalism is not precise enough—first order Peano axiomatics has infinite number of different non-equivalent models. The latter underlines the fact that first order logic is not too descriptive and mathematicians use implicitly more refined logic. On the other hand, the set of axioms is infinite, and thus we loose the completeness—some true formulas cannot be proved. The gap between true and provable formulas emerges, as the Peano arithmetic PA is rich enough to embed all primitively recursive relations, functions and predicates.

Clearly, one can express all natural numbers as a sum of ones, let n be the shorthand of such a sum[1]. Now, we can formally specify, what the embedding means.

Definition 2. *Let $r \subseteq \mathbb{N}^k$ be a k-ary relation. Then a formula $\rho_r(x_1, \ldots, x_n)$ with free variables $x_1, \ldots, x_k$ represents the relation R in theory $\mathcal{T}$ iff for for any tuple $(a_1, \ldots, a_k) \in \mathbb{N}^k$*

$$\begin{aligned} (a_1, \ldots, a_k) \in r \quad &\Rightarrow \quad \mathcal{T} \vdash \rho_r(\mathsf{a}_1, \ldots, \mathsf{a}_\mathsf{n}) \ , \\ (a_1, \ldots, a_k) \notin r \quad &\Rightarrow \quad \mathcal{T} \vdash \neg\rho_r(\mathsf{a}_1, \ldots, \mathsf{a}_\mathsf{n}) \ . \end{aligned}$$

We call such relations representable in theory $\mathcal{T}$.

Theorem 1 (Kurt Gödel). *Every primitively recursive function or predicate is representable in the Peano arithmetics.*

Let f be a primitively recursive function and ρ_f a formula that represents f. Then we can add a functional symbol f along with a defining axiom

$$\forall x_1 \forall x_2 \ldots \forall x_k \rho_f(x_1, x_2, \ldots, x_k, \mathsf{f}(x_1, x_2, \ldots, x_k))$$

to PA without increasing the set of representable or provable formulas.

[1] Later we use more space efficient representation.

2 Turing machines. Polynomial-time proofs

There is nothing mystical about Turing machines—reader has probably even seen several different formalisations. Nevertheless, we stress some aspects. First, for each a machine $\mathcal{M}$, let $\mathsf{code}_{\mathcal{U}}(\mathcal{M})$ denote the natural number that codes $\mathcal{M}$. In particular, let $\mathcal{U}$ be the universal Turing machine that interpretes and executes $\mathsf{code}_{\mathcal{U}}(\mathcal{M})$, i.e. $\mathcal{U}(\mathsf{code}_{\mathcal{U}}(\mathcal{M}), w) = \mathcal{M}(w)$ for all inputs $w \in \mathbb{N}$. Here, we assume that $\mathsf{code}_{\mathcal{U}}(\mathcal{M})$ is self-delimiting, i.e. the interpretator $\mathcal{U}$ can split its single binary input into the code part $\mathsf{code}_{\mathcal{U}}(\mathcal{M})$ and the argument w. Therefore, formally we have defined a computable function $\mathcal{U} : \mathbb{N} \to \mathbb{N}$. We use same convention for multiargument functions, i.e. a tuple $(w_1, \ldots, w_n)$ is actually a concatenation of self-delimiting descriptions of w_i.

We also use a dedicated universal Turing machine $\mathcal{U}_{\mathsf{p}}$ for polynomial time algorithms. The input of $\mathcal{U}_{\mathsf{p}}$ is a triple $(\mathsf{code}_{\mathcal{U}}(\mathcal{M}), c, w)$, where c is a time-bound constant and w input of $\mathcal{M}$. Given input the interpreter $\mathcal{U}_{\mathsf{p}}$ first computers a time-bound $t = \textsc{size}(w_1, \ldots, w_n)^c$ and then executes at most t instructions of $\mathcal{M}$ and halts. W.l.o.g. we can assume that $\mathcal{U}_{\mathsf{p}}$ is foolproof, i.e. halts when the input is invalid or time-bound is exceeded. The function $\textsc{size} : \mathbb{N}^k \to \mathbb{N}$ must be a polynomial-time computable function that is hardwired into the description of $\mathcal{U}_{\mathsf{p}}$, for example it can be the summary bit-length of inputs.

Similarly, we introduce codes for the predicate formulas ϕ in Peano arithmetics. Let $\mathsf{code}_{P}(\phi)$ be an efficient (polynomial-time decodable) representation of ϕ as natural number, for example a self-delimiting ASCII encoding of predicate formula. Hence, we can also encode proofs and use a dedicated Tueing machine $\mathcal{V}$ to verify formal proofs. We do not specify the proof system or the format of proof. Though, implicitly we assume that proofs are represented as a trees and a single Hilbert or Gentzen style rule is applied to derive parent from child nodes. However, we impose a efficiency and soundness requirement to the proof system.

Definition 3. *The proof system is sound, if it allows to prove only true formulas. The proof system is polynomial-time verifiable iff there exists a polynomial time algorithm $\mathcal{V}$ such that $\mathcal{V}(\rho, \mathsf{proof}(\rho)) = 1$ iff $\mathsf{proof}(\rho)$ is a valid proof of ρ.*

It is curious fact that given a set formulas that have polynomial size proofs w.r.t. the formula size, the proving itself is $\mathcal{NP}$-hard problem. Still, for suitable sub-classes of formulas one can device a polynomial-time prover. More formally, we can talk about time-complexities of some language L of provable formulas.

Definition 4. *A language L has a polynomial-time proofs iff there exists a polynomial-time prover $\mathcal{P}$ that generates a valid proof for every problem instance $\phi \in L$. The time-complexity of $\mathcal{P}$ must $\mathcal{O}(\textsc{size}(\mathsf{code}_{P}(\phi))^c)$, where $\textsc{size}$ can be defined in a problem specific way as long as $\textsc{size}(w) = \mathcal{O}(|w|)$. Let $\mathcal{T} \wedge \mathcal{P} \Vdash L$ denote that $\mathcal{P}$ is a polynomial-time prover for the language L. Let $\mathcal{T} \wedge \mathcal{P} \Vdash \phi$ denote that $\mathcal{P}$ generates valid proof for ϕ, it may fail for other instances.*

Especially interesting languages are generated by formulas $\phi(x_1, \ldots, x_k)$, namely let $L_\phi = \left\{\phi(\mathsf{a}_1, \ldots, \mathsf{a}_k) : (a_1, \ldots, a_k) \in \mathbb{N}^k \ \wedge\ \mathcal{T} \vdash \phi(\mathsf{a}_1, \ldots, \mathsf{a}_k)\right\}$. Standard proof systems for the first order Peano arithmetics are very inefficient, thus the set of polynomial-time provable languages is not very rich. It is conjectured that no formula $\phi(n, x)$, that represents fact $x = 2^n$, forms a polynomial-time provable language. Hence, we have to optimise the basic proof system. We consider such subtleties in the next section.

3 Representations with polynomial-time proofs

As said before, standard proof systems are too inefficient, therefore we consider an extended signature for arithmetics $\sigma_e = \langle 0, 1; +, \cdot, \mathsf{exp}; = \rangle$. Let $\rho_{\mathsf{exp}}(x, y)$ be a representation of $2^x = y$. Then the additional axiom $\forall n \rho_{\mathsf{exp}}(n, \mathsf{exp}(n))$ fixes uniquely the value $\mathbf{E}\left(()\, n\right)$. We enchance our proof system so that all $\rho_{\mathsf{exp}}(\mathsf{n}, 2^{\mathsf{n}})$ are axioms. Moreover, we use more compact encoding for integers $\mathsf{n} = \mathsf{n}_0 + \mathsf{n}_1 \cdot \mathsf{exp}(1) + \cdots + \mathsf{n}_\mathsf{k} \cdot \mathsf{exp}(\mathsf{k})$, where $n_k \dots n_0$ is the binary representation of n. Hence, $|\mathsf{code}_P(n)| = \mathcal{O}(\log^2 n)$ and simple arithmetic realations $x + y = z$, $xy = z$ and $x^y = z$ have polynomial-time proofs. The change is only cosmetic, the extended theory PA^{e} is clearly a conservative extension of Peano arithmetics. The functional symbol exp is just for convenience: we could drop it and use only ρ_{exp}, the class of polynomially provable formulas would not change.

Definition 5. *Let $r \subseteq \mathbb{N}^k$ be a k-ary relation. Then a formula $\rho_r(x_1, \dots, x_n)$ with free variables $x_1, \dots, x_k$ polynomially represents the relation r in theory $\mathcal{T}$ iff languages L_ϕ and $L_{\neg\phi}$ have polynomial-time proofs. We call such representation efficient.*

Next, we give a proof sketch for the following profound theorem by introducing additional functional symbols.

Theorem 2. *Any polynomial-time computable relation $r \subseteq \mathbb{N}^k$ can be efficiently represented as ρ_r. Moreover, if r is a graph of $(k-1)$-ary function, then*

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall x_1 \dots \forall x_{k-1} \exists! y \rho_r(x_1, \dots, x_{k-1}, y) \ .$$

Proof sketch We do not give the complete proof, we give only main ideas and let reader to fill the missing gaps. The improved proof system is actually a very efficient computational device. The latter allows us to use efficient analogue for Gödel β-function and the rest follows straightforwardly.

First, we introduce additional functions len, bit and $\boldsymbol{\beta}_{\mathsf{e}}$ by adding axioms

$$\forall x (\mathsf{exp}(\mathsf{len}(x)) \leq 2 \cdot x \wedge x < \mathsf{exp}(\mathsf{len}(x)))$$
$$\forall x \forall i \exists y (\rho_{\mathsf{rem}}(x, \mathsf{exp}(i+1), y) \wedge \rho_{\mathsf{div}}(y, \mathsf{exp}(i), \mathsf{bit}(x, i)))$$
$$\forall x \forall i \forall k \exists y (\rho_{\mathsf{rem}}(x, \mathsf{exp}((i+1) \cdot k), y) \wedge \rho_{\mathsf{div}}(\mathsf{exp}(k+1) \cdot y, \mathsf{exp}((i+1) \cdot k), \boldsymbol{\beta}_{\mathsf{e}}(x, i, k)))$$

where div and rem stand for integer quotient and reminder. In other words, we enforce $\mathsf{len}(\mathsf{x}) = |x|$, $\mathsf{bit}(\mathsf{x}, \mathsf{i}) = x_i$ and $\boldsymbol{\beta}_{\mathsf{e}}(\mathsf{x}, \mathsf{i}, \mathsf{k}) = x_{(i+1)k} \dots x_{ik+1}$, where x_i denotes the ith bit of x. It is straightforward but tiresome to prove that languages $L_{\mathsf{len}}, L_{\neg\mathsf{len}}, L_{\mathsf{bit}}, L_{\neg\mathsf{bit}}, L_{\boldsymbol{\beta}_{\mathsf{e}}}, L_{\neg\boldsymbol{\beta}_{\mathsf{e}}}$ have polynomial-time proofs. Now, for any Turing machine $\mathcal{M}$, we get the following representation

$$\exists a \exists t(\rho_{\mathsf{init}}(\boldsymbol{\beta}_{\mathsf{e}}(a, \mathsf{0}, t), x) \wedge \forall t_1 < t \rho_{\mathsf{tran}}(\boldsymbol{\beta}_{\mathsf{e}}(a, t_1, t), \boldsymbol{\beta}_{\mathsf{e}}(a, t_1 + \mathsf{1}, t)) \wedge \rho_{\mathsf{ends}}(\boldsymbol{\beta}_{\mathsf{e}}(a, t, t), y))$$

where $\rho_{\mathsf{tran}}(c_0, x) = 1$ iff the initial configuration of tape is x and $\mathcal{M}$ is in state q_1, $\rho_{\mathsf{tran}}(c_1, c_2) = 1$ iff $\mathcal{M}$ moves from configuration c_1 to c_2 and $\rho_{\mathsf{ends}}(c, y) = 1$ iff c is final configuration of $\mathcal{M}$ with the output y. A configuration c must capture the configuration of tape and the internal state of $\mathcal{M}$. For example, $c = c_1 + c_2 \cdot 2^t + c_3 \cdot 2^{2t}$, where c_1, c_2, c_3 are descritpions of the internal state q, head location and tape configuration. Therefore, if $\mathcal{M}$ is a polynomial time algorithm, then exists t that is polynomial in $\textsc{size}(x)$ such that adequate description fits into t-bit block and $\mathcal{M}$ does less than t steps.

Again, it is somewhat tedious to prove that $\rho_{\mathsf{init}}, \rho_{\mathsf{tran}}, \rho_{\mathsf{ends}}$ are polynomially representable. Now, observe a crucial detail. If $\mathcal{M}$ is polynomial time algorithm, then the prover can compute a by simulating the run of $\mathcal{M}$. As the bit-length of a is polynomial in $\textsc{size}(x)$ and the prover can prove or disprove the formula by considering only polynomial number of sub formulas

$$\begin{cases} \rho_{\mathsf{init}}(\boldsymbol{\beta}_{\mathsf{e}}(a, \mathsf{0}, t), x) \\ \rho_{\mathsf{tran}}(\boldsymbol{\beta}_{\mathsf{e}}(a, t_1, t), \boldsymbol{\beta}_{\mathsf{e}}(a, t_1 + \mathsf{1}, t)), \qquad t_1 = 0, \ldots, t-1 \\ \rho_{\mathsf{ends}}(\boldsymbol{\beta}_{\mathsf{e}}(a, t, t), y)) \end{cases} .$$

As each of those is a polynomial size formula, it has a polynomial size proof. The first claim is proven.

We do not give a formal proof to the second claim. The idea is simple. First, one devices a formula that says in each time step the Turing machine $\mathcal{M}$ is in a unique state. Next, it proves it using induction axiom. Finally, states additionally that after halting the state remains same. These two proofs combined in proper way form a solid proof of required triviality. □

4 Proofs of proofs

Now, it is time to raise to the meta-level. Following the footsteps of Gödel, we formalise the notion of provability in the framework of polynomial proofs.

Definition 6. *Let $\mathcal{V}$ be a polynomial-time verifier of a a first order theory $\mathcal{T}$, L be a class of well-formed true formulas and $\mathcal{P}$ a polynomial-time prover. Then the predicate $[\![\mathcal{T} \wedge \mathcal{P} \Vdash \phi]\!]$ characterises correctness of proofs*

$$[\![\mathcal{T} \wedge \mathcal{P} \Vdash \phi]\!] = \begin{cases} 1, & \text{if } \mathcal{U}_{\mathsf{p}}(m, c, \mathsf{code}_P(\phi)) \text{ produces a valid proof of } \phi, \\ 0, & \text{otherwise.} \end{cases}$$

Polynomial provability predicate has several interesting properties that are worth mentioning. Basically, polynomial provability is closed under various logical proving schemes, as long as we can compose provers to a polynomial-time superprover. Note that the formula $\forall x [\![\mathrm{PA}^{\mathsf{e}} \wedge \mathcal{P} \Vdash \phi(\mathsf{x})]\!]$ actually means, that given x we substitute x into ϕ and obtain $\phi(\mathsf{x})$ and then test $[\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P} \Vdash \phi(\mathsf{x})]\!]$, i.e. $[\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P} \Vdash \phi(\mathsf{x})]\!]$ is a shorthand for more complicated predicate depending on x.

Lemma 1. *Let $\phi(x_1, \dots, x_k)$ be a well-formed formula in the extended Peano arithmetics PA^{e} and let $\mathcal{P}$ be a polynomial-time algorithm. Let $\psi(x_1, \dots, x_k) = [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P} \Vdash \phi(\mathsf{x}_1, \dots, \mathsf{x}_\mathsf{k})]\!]$. Then there exists a suitable repsresentation of ψ such that there exists a polynomial-time prover $\mathcal{P}_\circ$ and $\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P}_\circ \Vdash L_\psi$.*

Proof. Clearly, $[\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P} \Vdash \phi(\mathsf{x}_1, \dots, \mathsf{x}_\mathsf{k})]\!]$ is a polynomial-time computable predicate: one first runs $\mathcal{P}$ to get a proof π and then runs $\mathcal{V}(\pi)$ to test the proof. Thus by Theorem 4, it is also polynomially-representable. $\square$

Lemma 2. *Let $\phi(x)$ and $\psi(x)$ be a well-formed formula in the extended Peano arithmetics PA^{e} and let $\mathcal{P}_1$ and $\mathcal{P}_2$ be a polynomial-time provers. Then there exists a polynomial-time prover $\mathcal{P}_3$ such that*

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall x([\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P}_1 \Vdash \phi(\mathsf{x})]\!] \wedge [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P}_2 \Vdash \phi(\mathsf{x}) \supset \psi(\mathsf{x})]\!] \supset [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P}_3 \Vdash \psi(\mathsf{x})]\!]).$$

Proof. This is evident, but the formal proof is somewhat complicated. First, the construction $\mathcal{P}_3$ is following, given input $\mathsf{code}_P(\psi(\mathsf{x}))$

- Runs $\mathcal{P}_1(\mathsf{code}_P(\phi(\mathsf{x})))$ and stores the output proof π_1.
- Runs $\mathcal{P}_2(\mathsf{code}_P(\phi(\mathsf{x}) \supset \psi(\mathsf{x})))$ and stores the output proof π_2.
- If verification of proofs fails:
  - $\mathcal{V}(\mathsf{code}_P(\phi(\mathsf{x})), \pi_1) = 0$ or
  - $\mathcal{V}(\mathsf{code}_P(\phi(\mathsf{x}) \supset \psi(\mathsf{x})), \pi_2) = 0$

  then halts with failure.
- Otherwise uses Modus Ponens rule $A, A \supset B \vdash B$ to merge π_1 and π_2 into a single proof π of $\psi(\mathsf{x})$. Outputs π.

Informally, we know that for all inputs $x_1, \dots, x_k$ either $\mathcal{P}_1$ or $\mathcal{P}_2$ fail or $\mathcal{P}_3$ produces a valid proof, however we need formal proof in PA^{e}.

Let $\textsc{out}_{\mathcal{P}_1}(x, y)$ be an efficient representation of $\mathcal{P}_1(x) = y$ and $\textsc{out}_{\mathcal{P}_2}(x, y)$ be an efficient representation of $\mathcal{P}_2(x) = y$. Then form Theorem follows

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall x_1 \forall x_2 \exists! y_1 \exists! y_2 (\textsc{out}_{\mathcal{P}_1}(x_1, y_1) \wedge \textsc{out}_{\mathcal{P}_1}(x_2, y_2))$$

On the other hand, from the specification of $\mathcal{V}$ on can obtain

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall a \forall b (\exists u \mathcal{V}(a, u) \wedge \exists v \mathcal{V}(b, v) \wedge \textsc{are}_{\phi, \phi \supset \psi}(a, b)) \supset \mathcal{V}(\textsc{com}(a, b), \textsc{mp}(u, v)),$$

where $\textsc{are}_{\phi, \phi \supset \psi}(a, b) = 1$ iff $a = \mathsf{code}_P(\phi(\mathsf{x}))$ and $b = \mathsf{code}_P(\phi(\mathsf{x})) \supset \psi(\mathsf{x})$ for some $x \in \mathbb{N}$, $\textsc{com}(a, b) = \mathsf{code}_P(\psi(\mathsf{x}))$ iff $\textsc{are}_{\phi, \phi \supset \psi}(a, b) = 1$ and $\textsc{mp}$ combines proofs using the Modus Ponents rule. Clearly, the proof is not trivial for $\mathcal{V}$, but it can be still proven by casting structural induction into PA^{e} famework.

Combining this formulas gives exactly the required proof. Though, it is completely straightforward it is far from trivial. $\square$

Corollary 1. *Let ϕ and ψ be a well-formed formula in the extended Peano arithmetics* PA^{e} *and let $\mathcal{P}$ a polynomial-time prover that provides valid proofs for L_ϕ. If there exists a proof*

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall x(\phi(x) \supset \psi(x))$$

then there exists a polynomial-time prover $\mathcal{P}_\circ$ such that

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall x([\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P} \Vdash \phi(\mathsf{x})]\!] \supset [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P}_\circ \Vdash \psi(\mathsf{x})]\!]) \ .$$

Proof. Direct corollary from the previous lemma. As $\mathrm{PA}^{\mathrm{e}} \vdash \forall x(\phi(x) \supset \psi(x))$ is a finite proof, there exists another polynomial-time algorithm $\mathcal{P}_2$ such that $\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P}_2 \Vdash \phi(\mathsf{x}) \supset \psi(\mathsf{x})$. Now, from Lemma 2 we obtain

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall x([\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P} \Vdash \phi(\mathsf{x})]\!] \wedge [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P}_2 \Vdash \phi(\mathsf{x}) \supset \psi(\mathsf{x})]\!] \supset [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P}_\circ \Vdash \psi(\mathsf{x})]\!]).$$

Since $[\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P}_2 \Vdash \phi(\mathsf{x}) \supset \psi(\mathsf{x})]\!] = 1$, we can omit it. □

Corollary 2. *Let ϕ and ψ be a well-formed formula in the extended Peano arithmetics* PA^{e} *and let $\mathcal{P}_1$ and $\mathcal{P}_2$ be polynomial-time provers. Then exists a polynomial-time prover $\mathcal{P}_3$ such that*

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall x([\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P}_1 \Vdash \phi(\mathsf{x})]\!] \wedge [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P}_2 \Vdash \psi(\mathsf{x})]\!] \supset [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P}_3 \Vdash \phi(\mathsf{x}) \wedge \phi(\mathsf{x})]\!])$$

Proof. The proof is analogous to Lemma2. Alternatively, it can be obtained from Corollary 1 by using tautology

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall x(\phi(x) \supset (\psi(x) \supset (\phi(x) \wedge \psi(x))))$$

□

Corollary 3. *Let ϕ be a well-formed formulas in the extended Peano arithmetics* PA^{e} *and let $\mathcal{P}$ be a polynomial-time prover. Then exists a polynomial-time prover $\mathcal{P}_\circ$ such that*

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall x \forall y([\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P} \Vdash \phi(\mathsf{x})]\!] \wedge [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P} \Vdash \phi(\mathsf{y})]\!] \supset [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P}_\circ \Vdash \phi(\mathsf{x}) \wedge \phi(\mathsf{y})]\!] \ .$$

Proof. Simple corollary of Corollary 5. □

Lemma 3. *Let $r \subseteq \mathbb{N}$ be a polynomial-time computable relation and ρ_r corresponding polynomial representation in theory* PA^{e}*. Then there exists a polynomial-time prover $\mathcal{P}$ such that*

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall x(\rho_r(x) \sim [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P} \Vdash \rho_r(\mathsf{x})]\!] \ .$$

Proof. Actually, this is a clear tautology. The relation is defined via a polynomial-time Turing machine $\mathcal{M}$ and we can take the representation ρ_r from Theorem 4. The prover $\mathcal{P}$ is just the same as in proof of Theorem 4. Therefore, we have to formally prove that simulation of $\mathcal{M}$ gives always the same results as running $\mathcal{M}$. The proof is fairly straightforward, however non-trivial. The proper way to formally prove this is to define sematics of instructions and then prove that for any Turing machine such formal proof exists by a structural induction. □

5 Incompleteness theorems

In order to postulate Gödel-style incompleteness theorems for polynomial time proofs, we have to sharpen the Kleene's recursion theorem. Still, lets start from additional notation before stating the both form of recursion theorems. Also, recall that by our convention universal Turing machines work with all inputs.

Definition 7. *Let* $\rho_{\mathsf{u\text{-}ptm}} : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ *be a polynomial representation of a universal polynomial Turing machine* $\mathcal{U}_{\mathsf{p}} : \mathbb{N} \to \mathbb{N}$. *Let* $\rho_{\mathsf{p\text{-}utm\text{-}p}} : \mathbb{N} \to \{0, 1\}$ *be a polynomial representation of a universal polynomial Turing machine restricted to Boolean outputs* $\mathcal{U}_{\mathsf{p}} : \mathbb{N} \to \{0, 1\}$.

Theorem 3 (Kleene). *For any self-delimiting argument* $m \in \mathbb{N}$ *there exist* $k \in \mathbb{N}$ *such that* $\mathcal{U}(k, w) = \mathcal{U}(m, k, w)$ *for all input values* $w \in \mathbb{N}$.

Theorem 4 (Polynomial-time Recursion Theorem). *For any* $m \in \mathbb{N}$ *and* $c_1 \in \mathbb{N}$ *there exist a code-constant* k *and a time-bound constant* $c_2 > c_1$ *such that*

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall w \forall y (\rho_{\mathsf{u\text{-}ptm}}(\mathsf{k}, \mathsf{c}_2, w, y) \sim \rho_{\mathsf{u\text{-}ptm}}(\mathsf{m}, \mathsf{k}, \mathsf{c}_1, w, y)).$$

Proof. Let $\mathcal{M}$ be a Turing machine such that $m = \mathsf{code}_{\mathcal{U}}(\mathcal{M})$. First, lets give an explicit description of a Turing machine $\mathcal{K}$ such that $k = \mathsf{code}_{\mathcal{U}}(\mathcal{K})$ and

$$\forall w \in \mathbb{N} : \qquad \mathcal{U}_{\mathsf{p}}(k, c_1, w) = \mathcal{U}_{\mathsf{p}}(m, k, w) \ .$$

Recall that constants c_1 and c_2 determine the time-bounds for $\mathcal{M}$ and $\mathcal{K}$. The algorithm $\mathcal{K}$ executes following steps:

1. Write t to the working tape.
2. Copy its own code k to the working tape.
3. Copy the inputs w to the working tape.
4. Interptete the input (t, k, c_1, w) as $\mathcal{U}_{\mathsf{p}}$.

Clearly, the code of $\mathcal{K}$ is finite and thus the second step requires only constant amount of time. Therefore, the interpeting the input (t, k, c_1, w) requires only a polynomial time and thus we can specify the time-bound constant c_2.

To complete the proof, we must formally show that interpreting the code and running the Turing machine produce the same result for every input. We omit the proof, as it is straightforward but non-trivial. □

And now, we are in good shape to construct the classical Gödel sentences: the sentence is true, when it is not provable in polynomial time. In order to avoid obscure and esoteric relational notation, we consider only accepting-rejecting universal Turing machine.

Corollary 4 (Reformulation of Polynomial-time Recursion Theorem). *For any* $m \in \mathbb{N}$ *and* $c_1 \in \mathbb{N}$ *there exist a code-constant* k *and a time-bound constant* $c_2 > c_1$ *such that*

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall w (\rho_{\mathsf{p\text{-}utm\text{-}p}}(\mathsf{k}, \mathsf{c}_2, w) \sim \rho_{\mathsf{p\text{-}utm\text{-}p}}(\mathsf{m}, \mathsf{k}, \mathsf{c}_1, w)).$$

Proof. Direct conclusion from Polynomial-time Recursion Theorem. □

Lemma 4 (Gödel Sentence). *For any polynomial-time accepting-rejecting Turing machine $\mathcal{M}$ there exist a formula $\rho_{\mathcal{M}}$ such that*

$$\mathrm{PA}^{\mathsf{e}} \vdash \forall w(\rho_{\mathcal{M}}(w) \sim \neg[\![\mathrm{PA}^{\mathsf{e}} \wedge \mathcal{M} \Vdash \rho_{\mathcal{M}}(\mathsf{w})]\!]$$

For all x the formula $\rho_{\mathcal{M}}$ is called a Gödel sentence with respect to $\mathcal{M}$.

Proof. First consider a two-argument Turing machine $\mathcal{T}$ that given input (k, w)

- Construct the formula $\rho_{\mathsf{p\text{-}utm\text{-}p}}(\mathsf{k}, \mathsf{c}_1, w)$
- Run $\mathcal{M}(\mathsf{code}_P(\rho_{\mathsf{p\text{-}utm\text{-}p}}(\mathsf{k}, \mathsf{c}_1, \mathsf{w})))$ and thest wheter the output π is a valid proof of $\rho_{\mathsf{p\text{-}utm\text{-}p}}(\mathsf{k}, \mathsf{c}_1, w, y)$.
- Return $\neg[\![\mathrm{PA}^{\mathsf{e}} \wedge \mathcal{M} \Vdash \rho_{\mathsf{p\text{-}utm\text{-}p}}(\mathsf{k}, \mathsf{c}_1, \mathsf{w})]\!]$.

By Polynomial-time Recursion Theorem there are values k and c_2 such that

$$\mathrm{PA}^{\mathsf{e}} \vdash \forall w(\rho_{\mathsf{p\text{-}utm\text{-}p}}(\mathsf{k}, \mathsf{c}_2, w) \sim \rho_{\mathsf{p\text{-}utm\text{-}p}}(\mathsf{t}, \mathsf{k}, \mathsf{c}_1, w))$$

and thus by construction

$$\mathrm{PA}^{\mathsf{e}} \vdash \forall w(\rho_{\mathsf{p\text{-}utm\text{-}p}}(\mathsf{k}, \mathsf{c}_2, w) \sim \neg[\![\mathrm{PA}^{\mathsf{e}} \wedge \mathcal{M} \Vdash \rho_{\mathsf{p\text{-}utm\text{-}p}}(\mathsf{k}, \mathsf{c}_1, \mathsf{w})]\!] \ .$$

Since $\mathcal{T}$ is a polynomial-time algorithm there exists a value c such that

$$\mathrm{PA}^{\mathsf{e}} \vdash \forall w(\rho_{\mathsf{p\text{-}utm\text{-}p}}(\mathsf{t}, \mathsf{k}, \mathsf{c}, w) \sim \rho_{\mathsf{p\text{-}utm\text{-}p}}(\mathsf{t}, \mathsf{k}, \mathsf{c}_1, w)), \qquad c < c_1 \ .$$

Hence, we can set $\rho_{\mathcal{M}}(w) = \rho_{\mathsf{p\text{-}utm\text{-}p}}(\mathsf{k}, \mathsf{c}_2, w)$, where c_2 corresponds to sufficiently large c, and the claim is proved. □

Theorem 5 (First Incompleteness Theorem). *Let $\mathcal{M}$ be a polynomial-time Turing machine and $\rho_{\mathcal{M}}(w)$ the corresponding Gödel sentence. Then for all inputs $w \in \mathbb{N}$*

$$\mathrm{PA}^{\mathsf{e}} \wedge \mathcal{M} \not\Vdash \rho_{\mathcal{M}}(\mathsf{w})$$

and

$$\mathrm{PA}^{\mathsf{e}} \wedge \mathcal{M} \not\Vdash \neg[\![\mathrm{PA}^{\mathsf{e}} \wedge \mathcal{M} \Vdash \rho_{\mathcal{M}}(\mathsf{w})]\!]$$

unless PA^{e} *is inconsistent.*

Proof. We know that

$$\mathrm{PA}^{\mathsf{e}} \vdash \forall w(\rho_{\mathcal{M}}(w) \sim \neg[\![\mathrm{PA}^{\mathsf{e}} \wedge \mathcal{M} \Vdash \rho_{\mathcal{M}}(\mathsf{w})]\!]$$

and thus for all $w \in \mathbb{N}$ such that $\rho_{\mathcal{M}}(w) = 1$, the prover $\mathcal{M}$ produces invalid proof for $\rho_{\mathcal{M}}(w)$. On the other hand, if PA^{e} is consistent then $\mathcal{M}$ cannot provide a valid proof when $\rho(w) = 0$.

For a shake of contradiction, assume that

$$\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \Vdash \neg [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \Vdash \rho_{\mathcal{M}}(\mathsf{w})]\!] \enspace .$$

Then from the consistency follows $[\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \Vdash \rho_{\mathcal{M}}(\mathsf{w})]\!] = 1$ that is

$$\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \Vdash \rho_{\mathcal{M}}(\mathsf{w})$$

and we have contradiction with the first claim. □

Theorem 6 (Second Incompleteness Theorem). *Let $\phi(w)$ be a well-formed formula with a single free variable w and $\mathcal{M}$ a polynomial-time Turing machine. Then there exists a Turing machine $\mathcal{M}_\circ$ such that for all $w \in \mathbb{N}$*

$$\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \nVdash \neg [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M}_\circ \Vdash \phi(\mathsf{w})]\!]$$

unless PA^{e} *is inconsistent.*

Proof. We start with the Gödel sentence $\rho_{\mathcal{M}}(w)$ for $\mathcal{M}$. And in a long run, we want to construct $\mathcal{M}_\circ$ such that

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall w([\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \Vdash \rho_{\mathcal{M}}(\mathsf{w})]\!] \supset [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M}_\circ \Vdash \phi(\mathsf{w})]\!]) \tag{1}$$

as it gives an immediate contradiction. More precisely, if we have

$$\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \Vdash \neg [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M}_\circ \Vdash \phi(\mathsf{w})]\!] \tag{2}$$

then combining formal proof (1), we can construct a polynomial-time prover $\mathcal{M}_\star$ such that if condition (2) is satisfied, then

$$\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M}_\star \Vdash \neg [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \Vdash \rho_{\mathcal{M}}(\mathsf{w})]\!] \enspace .$$

But this is impossible accoding to First Incompleteness Theorem.

To give a proof for line (1), we have to give an explicit construction to $\mathcal{M}_\circ$ and the corresponding formal proof. Nevertheless, we give only a traditional handwritten proof and leave it to the reader to completely formalise it. From Lemma 3, we get that exist a polynomial-time prover $\mathcal{P}_1$ such that

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall w([\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \Vdash \rho_{\mathcal{M}}(\mathsf{w})]\!] \supset [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P}_1 \Vdash [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \Vdash \rho_{\mathcal{M}}(\mathsf{w})]\!]]\!])$$

From Lemma 4, we get

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall w([\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \Vdash \rho_{\mathcal{M}}(\mathsf{w})]\!] \supset \neg\rho_{\mathcal{M}}(\mathsf{w}))$$

and thus exists a polynomial-time prover $\mathcal{P}_2$ such that

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall w([\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \Vdash \rho_{\mathcal{M}}(\mathsf{w})]\!] \supset [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P}_2 \Vdash \neg\rho_{\mathcal{M}}(\mathsf{w})]\!]) \enspace .$$

Combining provers $\mathcal{M}$ and $\mathcal{P}_2$, Lemma assures existance of a polynomial-time prover $\mathcal{P}_3$ that can prove *falsum*

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall w([\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \Vdash \rho_{\mathcal{M}}(\mathsf{w})]\!] \supset [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{P}_3 \Vdash \rho_{\mathcal{M}}(\mathsf{w}) \wedge \neg\rho_{\mathcal{M}}(\mathsf{w})]\!] \enspace .$$

Since

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall w(\phi_{\mathcal{M}}(\mathsf{w}) \wedge \neg\rho_{\mathcal{M}}(\mathsf{w}) \supset \phi(\mathsf{w}))$$

is a tautology, Corollary 1 assures existance of a polynomial-time prover $\mathcal{M}_\circ$ such that

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall w([\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \Vdash \rho_{\mathcal{M}}(\mathsf{w})]\!] \supset [\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M}_\circ \Vdash \phi(\mathsf{w})]\!]$$

and this completes our proof. □

6 The formalization of SAT problem

Before considering the $\mathcal{P} = \mathcal{NP}$ problem, we follow the traditional path and define 3-SAT problem. Though, formally we do not have explicit propositional variables in the formulas of first order logic, we can circumvent the restriction and use $x_i = 1$ for each propositional variable. Hence, it is rather straightforward to define language L_{3CNF} of 3-CNF formulas in PA. By a standard convention all ill-formed or non-3-CNF formulas are considered unsatisfiable.

Definition 8 (3SAT relation). *Let $r_{\mathsf{3SAT}} \subseteq \mathbb{N}$ be a relation such that $x \in r_{\mathsf{3SAT}}$ only if there exist a satisfiable formula $\phi \in L_{\mathrm{3CNF}}$ and $x = \mathsf{code}_P(\phi)$. Let ρ_{3SAT} denote an inefficient representation of r_{3SAT} as a formula in the first order Peano arithmetics.*

Definition 9 (Language of satisfiable formulas). *We denote the set of satisfiable formulas by $L_{\mathrm{SAT}} = \{\rho_{\mathsf{3SAT}}(\mathsf{a}) : \rho_{\mathsf{3SAT}}(\mathsf{a}) \wedge \mathrm{PA} \vdash \rho_{\mathsf{3SAT}}(\mathsf{a})\}$ and its complement by $L_{\mathrm{CO\text{-}SAT}} = \{\rho_{\mathsf{3SAT}}(\mathsf{a}) : \rho_{\mathsf{3SAT}}(\mathsf{a}) \wedge \mathrm{PA} \vdash \neg\rho_{\mathsf{3SAT}}(\mathsf{a})\}$. For all $x \in \mathbb{N}$ that have corresponding 3CNF formulas, let $\textsc{size}(x) = 2n$, where n is the number of free variables in the 3CNF formula. Let $\textsc{size}(x) = 2 \cdot |x|$ otherwise.*

Wheter to include incorrect formulas into $L_{\mathrm{CO\text{-}SAT}}$ is a matter of choice. The main main motivation behind our choice is conceptual simplicity. If such incorrect formulas are excluded, then there is an explicit need for efficient (polynomial-time in $\textsc{size}(\phi)$) enumeration of correct formulas. Though, it is clearly doable, it just complicates the matters.

To formalise the guestion $\mathcal{P} = \mathcal{NP}$ or not, we need a efficient representation of such a test. We provide actually two such tests: binary predicates solve-sat and prove-sat.

Definition 10. *A predicate $\rho_{\mathsf{prove\text{-}sat}} : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ characterises the ability of Turing machines to prove $\phi \in L_{\mathrm{SAT}}$, i.e. $\rho_{\mathsf{solve\text{-}sat}}(m, w)$ is an efficient representation of $[\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \Vdash \rho_{\mathsf{3SAT}}(\mathsf{w})]\!]$ where $m = \mathsf{code}_{\mathcal{U}}(\mathcal{M})$ and $w = \mathsf{code}_P(\phi)$. A predicate $\rho_{\mathsf{solve\text{-}sat}} : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ characterises the ability of Turing machines to recognise the language L_{SAT}, i.e. $\rho_{\mathsf{solve\text{-}sat}}(m, w) = \rho_{\mathsf{p\text{-}utm\text{-}p}}(m, w) \sim \rho_{\mathsf{3SAT}}(w)$.*

Theorem 7 (Main Negative Result). *For all polynomial-time provers $\mathcal{M}$ there exist a polynomial-time Turing machine $\mathcal{M}_\circ$ such that for all $w \in \mathbb{N}$*

$$\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \not\Vdash \neg\rho_{\mathsf{solve\text{-}sat}}(\mathsf{m}_\circ, \mathsf{w}) \enspace ,$$

where $m_\circ = \mathsf{code}_{\mathcal{U}}(\mathcal{M}_\circ)$.

In the first version of the article, authors claimed that Second Incompleteness theorem is sufficient to prove the claim. Shortly put, there is a polynomial-time Turing machine $\mathcal{M}_\circ$ such that

$$\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M} \not\Vdash \neg[\![\mathrm{PA}^{\mathrm{e}} \wedge \mathcal{M}_\circ \Vdash \rho_{\mathsf{3SAT}}(\mathsf{w})]\!]$$

but this is not enough for contradiction, as inability of $\mathcal{M}$ to prove something about $\mathcal{M}_\circ$ proofs does not lead to anywhere. Hence, we need stronger incompleteness theorems that deal with inability to proove correct polynomial-time descisions.

7 Polynomial-time descisions

Formalising correctness of polynomial-time descisions is strickingly simple. Nevertheless, it is possible to obscure matters beyound recognition by introducing complex notation. Therefore, we take anoter and more simple path. Consider a Turing machine $\mathcal{M}$ that given an input $\mathsf{code}_P(\phi)$ accepts or rejects input. Clearly, $\mathcal{M}$ correctly accepts ϕ if $\mathrm{PA}^{\mathrm{e}} \models \phi$ or alternatively

$$\mathrm{PA} \models \rho_{\mathsf{p\text{-}utm\text{-}p}}(\mathsf{code}_{\mathcal{U}}(\mathcal{M}), \mathsf{code}_P(\phi)) \wedge \phi \enspace .$$

Still, the notation is to cumbersome and hence, we use

$$\mathrm{PA} \models \mathcal{M}(\phi) \wedge \phi \enspace .$$

To stress the meaning and separate from other formulas, let $[\![\mathcal{M}(\phi) \wedge \phi]\!]$ be the corresponding predicate, i.e.

$$[\![\mathcal{M}(\phi) \wedge \phi]\!] = \begin{cases} 1, & \text{if } \mathrm{PA} \models \mathcal{M}(\phi) \wedge \phi, \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, the proof of $[\![\mathcal{M}(\phi) \wedge \phi]\!]$ in PA is actually

$$\mathrm{PA} \vdash \mathcal{M}(\phi) \wedge \phi \enspace .$$

The following definition summarises the notation and extends it to rejections and descisions.

Definition 11 (Absolute corretness). *Let L be a set of formulas in* PA^{e} *and let $\mathcal{M}$ be a polynomial-time Turing machine. Then $\mathcal{M}$ correctly accepts $\phi \in L$ iff* $\mathrm{PA} \models \mathcal{M}(\phi) \wedge \phi$*, correctly rejects iff* $\mathrm{PA} \models \neg\mathcal{M}(\phi) \wedge \neg\phi$*, and correctly descides iff* $\mathrm{PA} \models \mathcal{M}(\phi) \sim \phi$*. The corresponding notations are $[\![\mathcal{M}(\phi) \wedge \phi]\!]$, $[\![\neg\mathcal{M}(\phi) \wedge \neg\phi]\!]$ and $[\![\mathcal{M}(\phi) \sim \phi]\!]$.*

Definition 12 (Relative correctness). *Let L be a set of formulas in* PA^{e} *and let $\mathcal{M}$ be a polynomial-time Turing machine and $\mathcal{V}$ a descision verifier. Then with respect to $\mathcal{V}$ $\mathcal{M}$ correctly accepts $\phi \in L$ iff* $\mathrm{PA} \models \mathcal{M}(\phi) \wedge \mathcal{V}(\phi)$*, correctly rejects iff* $\mathrm{PA} \models \neg\mathcal{M}(\phi) \wedge \neg\mathcal{V}(\phi)$*, and correctly descides iff* $\mathrm{PA} \models \mathcal{M}(\phi) \sim \mathcal{V}(\phi)$*. The corresponding notations are $[\![\mathcal{M}(\phi) \wedge \mathcal{V}(\phi)]\!]$, $[\![\neg\mathcal{M}(\phi) \wedge \neg\mathcal{V}(\phi)]\!]$ and $[\![\mathcal{M}(\phi) \sim \mathcal{V}(\phi)]\!]$.*

Let r be the primitively recursive relation and $\mathcal{M}$ corresponding Turing machine that evaluates ρ. then the representation theorem assures that there is a formula ρ that simulates the run of $\mathcal{M}$. Let us call ρ to *canonic representation* of r. Then there exists a universal prover that for each canonic representation produces proof $\mathrm{PA} \vdash \rho(\mathsf{x})$, whenever $\rho(x)$ is true.

Lemma 5. *Let ϕ be a primitively recursive predicate in a canonic form. Then there exist a descision verifier V_ϕ such that*

$$\begin{array}{ccc} [\![\mathcal{M}(\phi(\mathsf{x})) \wedge \mathcal{V}_\phi(\phi(\mathsf{x}))]\!] & \Leftrightarrow & [\![\mathcal{M}(\phi(\mathsf{x})) \wedge \phi(\mathsf{x})]\!] \\ [\![\neg\mathcal{M}(\phi(\mathsf{x})) \wedge \neg\mathcal{V}_\phi(\phi(\mathsf{x}))]\!] & \Leftrightarrow & [\![\neg\mathcal{M}(\phi(\mathsf{x})) \wedge \neg\phi(\mathsf{x})]\!] \end{array}$$

Proof. Obvious, since we can use universal prover to test $\phi(\mathsf{x})$. □

Lemma 6. *Let $\phi(x)$ be a primitively recursive predicate in a canonical form. Then there exists a primitively recursive upper bound f_ϕ for proof of descisions, i.e.*

$$\begin{array}{ccc} \exists y < f_\phi(\textsc{size}(\phi(\mathsf{a}))) \wedge \mathcal{V}_{\mathsf{pa}}(\phi(\mathsf{a}), y) = 1 & \Leftrightarrow & \mathrm{PA} \vdash \phi(\mathsf{a}) \\ \exists y < f_\phi(\textsc{size}(\phi(\mathsf{a}))) \wedge \mathcal{V}_{\mathsf{pa}}(\neg\phi(\mathsf{a}), y) = 1 & \Leftrightarrow & \mathrm{PA} \vdash \neg\phi(\mathsf{a}) \end{array}$$

provided that SIZE *majories primitively recursive monotonically growing function that is unbounded.*

Proof. Obvious, since we can list all proofs for all formulas with fixed size and then take the maximum from proofs. □

Now, by some reason—still unknown to author of tis review—we consider a dedicated verifiers.

Definition 13 (Dedicated verifier for acceptance).

Definition 14 (Dedicated verifier for rejection).

Definition 15 (Soundness). *We say that verifier $\mathcal{V}$ soundly accepts iff*

$$\mathrm{PA} \models \mathcal{V}(\phi) \supset \phi$$

and soundly rejects

$$\mathrm{PA} \models \neg\mathcal{V}(\phi) \supset \neg\phi.$$

Lemma 7. *Deticated acceptance verifier $\mathcal{V}_\phi^+$ and rejection verifier $\mathcal{V}_\phi^-$ are correct, i.e.*

$$\mathrm{PA} \models \forall a(\mathcal{V}_\phi^+(\phi(\mathsf{a}) \sim \phi(\mathsf{a}))$$
$$\mathrm{PA} \models \forall a(\neg\mathcal{V}_\phi^-(\phi(\mathsf{a}) \sim \neg\phi(\mathsf{a}))$$

and also satisfy

$$\begin{aligned}
&\mathrm{PA} \vdash \forall x(\mathcal{V}_\phi^+(\phi_1(\mathsf{x}) \wedge \phi_2(\mathsf{x})) \sim \mathcal{V}_\phi^+(\phi_1(\mathsf{x})) \wedge \mathcal{V}_\phi^+(\phi_2(\mathsf{x})))\\
&\mathrm{PA} \vdash \forall x(\mathcal{V}_\phi^+(\llbracket \mathcal{M}(\psi(\mathsf{x})) \wedge \mathcal{V}_\phi^+(\psi(\mathsf{x})\rrbracket) \sim \llbracket \mathcal{M}(\psi(\mathsf{x})) \wedge \mathcal{V}_\phi^+(\psi(\mathsf{x}))\rrbracket\\
&\mathrm{PA} \vdash \forall x(\neg\mathcal{V}_\phi^-(\phi_1(\mathsf{x}) \vee \phi_2(\mathsf{x})) \sim \neg\mathcal{V}_\phi^-(\phi_1(\mathsf{x})) \wedge \neg\mathcal{V}_\phi^-(\phi_2(\mathsf{x})))\\
&\mathrm{PA} \vdash \forall x(\neg\mathcal{V}_\phi^-(\llbracket \mathcal{M}(\psi(\mathsf{x})) \wedge \mathcal{V}_\phi^-(\psi(\mathsf{x})\rrbracket) \sim \neg\llbracket \mathcal{M}(\psi(\mathsf{x})) \wedge \mathcal{V}_\phi^-(\psi(\mathsf{x}))\rrbracket
\end{aligned}$$

Proof. Nothing to proove, follows from the constructions of verifiers. □

Lemma 8. *Let ϕ be a primitively recursive predicate in a canonic form. Then it is sufficient to test descisions w.r.t. the deticated verifiers $\mathcal{V}_\phi^+$ and $\mathcal{V}_\phi^-$, i.e.*

$$\begin{aligned}
\llbracket \mathcal{M}(\phi(\mathsf{x})) \wedge \mathcal{V}_\phi^+(\phi(\mathsf{x}))\rrbracket \quad &\Leftrightarrow \quad \llbracket \mathcal{M}(\phi(\mathsf{x})) \wedge \phi(\mathsf{x})\rrbracket\\
\llbracket \neg\mathcal{M}(\phi(\mathsf{x})) \wedge \neg\mathcal{V}_\phi^-(\phi(\mathsf{x}))\rrbracket \quad &\Leftrightarrow \quad \llbracket \neg\mathcal{M}(\phi(\mathsf{x})) \wedge \neg\phi(\mathsf{x})\rrbracket
\end{aligned}$$

Proof. Nothing to prove. □

Lemma 9. *Let ϕ and ψ be a primitively recursive predicates in a canonic form. Then for any polynomial-time Turing machine $\mathcal{M}$, we have*

$$\begin{aligned}
&\mathrm{PA} \vdash \forall x(\llbracket \mathcal{M}(\phi(\mathsf{x})) \wedge \mathcal{V}_\psi^+(\phi(\mathsf{x})\rrbracket \supset \llbracket \mathcal{M}(\phi(\mathsf{x})) \wedge \mathcal{V}_\phi^+(\phi(\mathsf{x})\rrbracket)\\
&\mathrm{PA} \vdash \forall x(\llbracket \neg\mathcal{M}(\phi(\mathsf{x})) \wedge \neg\mathcal{V}_\psi^-(\phi(\mathsf{x})\rrbracket \supset \llbracket \neg\mathcal{M}(\phi(\mathsf{x})) \wedge \neg\mathcal{V}_\phi^-(\phi(\mathsf{x})\rrbracket)
\end{aligned}$$

Proof. Nothing to proove. □

8 Descisions about descisions

Lemma 10. *Let $\phi(x)$ be a well-formed formula in Peano Arithmetics. Then for any polynomial-time Turing machine $\mathcal{M}$ and for any $a \in \mathbb{N}$, we have*

$$\begin{aligned}
\llbracket \mathcal{M}(\phi(\mathsf{a})) \wedge \mathcal{V}(\phi(\mathsf{a}))\rrbracket \quad &\Rightarrow \quad \mathrm{PA} \vdash \llbracket \mathcal{M}(\phi(\mathsf{a})) \wedge \mathcal{V}(\phi(\mathsf{a}))\rrbracket\\
\llbracket \neg\mathcal{M}(\phi(\mathsf{a})) \wedge \neg\mathcal{V}(\phi(\mathsf{a}))\rrbracket \quad &\Rightarrow \quad \mathrm{PA} \vdash \llbracket \neg\mathcal{M}(\phi(\mathsf{a})) \wedge \neg\mathcal{V}(\phi(\mathsf{a}))\rrbracket
\end{aligned}$$

Proof. Nothing to proove. □

Lemma 11. *Let $\phi(x)$ and $\psi(x)$ be a well-formed formulas in Peano Arithmetics. Then for any polynomial-time Turing machines $\mathcal{M}_1$ and $\mathcal{M}_2$ there exist a polynomial time turing machine $\mathcal{M}_3$ such that and for any $a \in \mathbb{N}$, we have*

$$\mathrm{PA} \vdash \forall x(\llbracket \mathcal{M}_1(\phi(\mathsf{x})) \wedge \mathcal{V}_\Omega^+(\phi(\mathsf{x}))\rrbracket \wedge \llbracket \mathcal{M}_2(\psi(\mathsf{a})) \wedge \mathcal{V}_\Omega^+(\psi(\mathsf{a}))\rrbracket \supset \llbracket \mathcal{M}_3(\phi(\mathsf{a}) \wedge \psi(\mathsf{a})) \wedge \mathcal{V}_\Omega^+(\phi(\mathsf{a}) \wedge \psi(\mathsf{a}))\rrbracket$$
$$\mathrm{PA} \vdash \forall x(\llbracket \neg\mathcal{M}_1(\phi(\mathsf{x})) \wedge \neg\mathcal{V}_\Omega^+(\phi(\mathsf{x}))\rrbracket \wedge \llbracket \neg\mathcal{M}_2(\psi(\mathsf{a})) \wedge \neg\mathcal{V}_\Omega^+(\psi(\mathsf{a}))\rrbracket \supset \llbracket \neg\mathcal{M}_3(\phi(\mathsf{a}) \vee \psi(\mathsf{a})) \wedge \neg\mathcal{V}_\Omega^+(\phi(\mathsf{a}) \vee \psi(\mathsf{a})$$

Proof. Nothing to proove, follows from constructions of dedicated verifiers. □

Lemma 12. *Let $\phi(x)$ and $\psi(x)$ be a well-formed formulas in Peano Arithmetics. Assume that* PA $\vdash \forall x(\phi(x) \supset \psi(x))$ *is installed into $\mathcal{V}_\Omega^+$. Then for any polynomial-time Turing machines $\mathcal{M}_1$ there exist a polynomial time turing machine $\mathcal{M}_2$ such that and for any $a \in \mathbb{N}$, we have*

$$\mathrm{PA} \vdash \forall x([\![\mathcal{M}_1(\phi(\mathsf{x})) \wedge \mathcal{V}_\Omega^+(\phi(\mathsf{x}))]\!] \supset [\![\mathcal{M}_1(\psi(\mathsf{a})) \wedge \mathcal{V}_\Omega^+(\psi(\mathsf{a}))]\!]$$

Proof. Nothing to proove, follows from constructions of dedicated verifiers. □

Lemma 13. *Let $\phi(x)$ and $\psi(x)$ be a well-formed formulas in Peano Arithmetics. Assume that* PA $\vdash \forall x(\neg\phi(x) \supset \neg\psi(x))$ *is installed into $\mathcal{V}_\Omega^-$. Then for any polynomial-time Turing machines $\mathcal{M}_1$ there exist a polynomial time turing machine $\mathcal{M}_2$ such that and for any $a \in \mathbb{N}$, we have*

$$\mathrm{PA} \vdash \forall x([\![\neg\mathcal{M}_1(\phi(\mathsf{x})) \wedge \neg\mathcal{V}_\Omega^+(\phi(\mathsf{x}))]\!] \supset [\![\neg\mathcal{M}_1(\psi(\mathsf{a})) \wedge \neg\mathcal{V}_\Omega^+(\psi(\mathsf{a}))]\!]$$

Proof. Nothing to proove, follows from constructions of dedicated verifiers. □

Lemma 14. *Let $\phi(x)$ and $\psi(x)$ be a well-formed formulas in Peano Arithmetics. Assume that* PA $\vdash \forall x(\phi(x) \supset \psi(x))$ *is installed into $\mathcal{V}_\Omega^+$. Then for any polynomial-time Turing machines $\mathcal{M}_1$ there exist a polynomial time turing machine $\mathcal{M}_2$ such that and for any $a \in \mathbb{N}$, we have*

$$\mathrm{PA} \vdash \forall x([\![\mathcal{M}_1(\phi(\mathsf{x})) \wedge \mathcal{V}_\Omega^+(\phi(\mathsf{x}))]\!] \supset [\![\mathcal{M}_2([\![\mathcal{M}_1(\phi(\mathsf{x})) \wedge \mathcal{V}_\Omega^+(\phi(\mathsf{x}))]\!]) \wedge \mathcal{V}_\Omega^+([\![\mathcal{M}_1(\phi(\mathsf{x})) \wedge \mathcal{V}_\Omega^+(\phi(\mathsf{x}))]\!])]\!]$$

Proof. Nothing to proove, follows from constructions of dedicated verifiers. □

Lemma 15. *Let $\phi(x)$ and $\psi(x)$ be a well-formed formulas in Peano Arithmetics. Assume that* PA $\vdash \forall x(\phi(x) \supset \psi(x))$ *is installed into $\mathcal{V}_\Omega^+$. Then for any polynomial-time Turing machines $\mathcal{M}_1$ there exist a polynomial time turing machine $\mathcal{M}_2$ such that and for any $a \in \mathbb{N}$, we have*

$$\mathrm{PA} \vdash \forall x([\![\neg\mathcal{M}_1(\phi(\mathsf{x})) \wedge \neg\mathcal{V}_\Omega^-(\phi(\mathsf{x}))]\!] \supset [\![\neg\mathcal{M}_2([\![\neg\mathcal{M}_1(\phi(\mathsf{x})) \wedge \neg\mathcal{V}_\Omega^+(\phi(\mathsf{x}))]\!])) \wedge \neg\mathcal{V}_\Omega^+(\neg[\![\mathcal{M}_1(\phi(\mathsf{x})) \wedge \neg\mathcal{V}_\Omega^+(\phi(\mathsf{x}$$

Proof. Nothing to proove, follows from constructions of dedicated verifiers. □

9 Gödel sentemces about descisions

Lemma 16 (Gödel sentences for accepting descisions). *For any polynomial-time Turing machine $\mathcal{M}$ and unbounded verifier $\mathcal{V}$, there exist a formula $\rho_{\mathcal{M},\mathcal{V}}(x)^+$ such that*

$$\mathrm{PA} \vdash \forall x(\rho^+_{\mathcal{M},\mathcal{V}}(x) \sim \neg[\![\mathcal{M}(\rho^+_{\mathcal{M},\mathcal{V}}(\mathsf{x})) \wedge \mathcal{V}(\rho^+_{\mathcal{M},\mathcal{V}}(\mathsf{x}))]\!]) \ .$$

Proof. First consider a two-argument Turing machine $\mathcal{T}$ that given input (k, w)

- Construct a formula $\rho_{\mathsf{utm\text{-}p}}(\mathsf{k}, w)$.
- Output $\neg(\mathcal{M}(\rho_{\mathsf{utm\text{-}p}}(\mathsf{k}, \mathsf{w})) \wedge \mathcal{V}(\rho_{\mathsf{utm\text{-}p}}(\mathsf{k}, \mathsf{w})))$.

By Kleene Recursion Theorem there is a value k such that

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall w(\rho_{\mathsf{utm\text{-}p}}(\mathsf{k}, w) \sim \rho_{\mathsf{utm\text{-}p}}(\mathsf{t}, \mathsf{k}, w))$$

and thus by construction

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall w(\rho_{\mathsf{utm\text{-}p}}(\mathsf{k}, w) \sim \neg[\![\mathcal{M}(\rho_{\mathsf{utm\text{-}p}}(\mathsf{k}, \mathsf{w})) \wedge \mathcal{V}(\rho_{\mathsf{utm\text{-}p}}(\mathsf{k}, \mathsf{w}))]\!]) \ .$$

Hence, the claim is proved for $\rho^{+}_{\mathcal{M},\mathcal{V}}(w) = \rho_{\mathsf{utm\text{-}p}}(\mathsf{k}, w)$. □

Lemma 17 (Gödel sentences for rejecting descisions). *For any polynomial-time Turing machine $\mathcal{M}$ and unbounded verifier $\mathcal{V}$, there exist a formula $\rho^{-}_{\mathcal{M},\mathcal{V}}(x)$ such that*

$$\mathrm{PA} \vdash \forall x(\rho^{-}_{\mathcal{M},\mathcal{V}}(x) \sim \neg[\![\neg\mathcal{M}(\rho^{-}_{\mathcal{M},\mathcal{V}}(\mathsf{x})) \wedge \neg\mathcal{V}(\rho^{-}_{\mathcal{M},\mathcal{V}}(\mathsf{x}))]\!]) \ .$$

Proof. First consider a two-argument Turing machine $\mathcal{T}$ that given input (k, w)

- Construct a formula $\rho_{\mathsf{utm\text{-}p}}(\mathsf{k}, w)$.
- Output $\neg(\neg\mathcal{M}(\rho_{\mathsf{utm\text{-}p}}(\mathsf{k}, \mathsf{w})) \wedge \neg\mathcal{V}(\rho_{\mathsf{utm\text{-}p}}(\mathsf{k}, \mathsf{w})))$.

By Kleene Recursion Theorem there is a value k such that

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall w(\rho_{\mathsf{utm\text{-}p}}(\mathsf{k}, w) \sim \rho_{\mathsf{utm\text{-}p}}(\mathsf{t}, \mathsf{k}, w))$$

and thus by construction

$$\mathrm{PA}^{\mathrm{e}} \vdash \forall w(\rho_{\mathsf{utm\text{-}p}}(\mathsf{k}, w) \sim \neg[\![\neg\mathcal{M}(\rho_{\mathsf{utm\text{-}p}}(\mathsf{k}, \mathsf{w})) \wedge \neg\mathcal{V}(\rho_{\mathsf{utm\text{-}p}}(\mathsf{k}, \mathsf{w}))]\!]) \ .$$

Hence, the claim is proved for $\rho^{-}_{\mathcal{M},\mathcal{V}}(w) = \rho_{\mathsf{utm\text{-}p}}(\mathsf{k}, w)$. □

Theorem 8 (First Incompleteness Theorem). *For any polynomial-time Turing machine $\mathcal{M}$ and unbounded sound verifier $\mathcal{V}$, $\mathcal{M}$ cannot correctly accept Gödel sentences $\rho^{+}_{\mathcal{M},\mathcal{V}}(x)$, i.e. for all $x \in \mathbb{N}$*

$$\mathrm{PA} \models \neg[\![\mathcal{M}(\rho^{+}_{\mathcal{M},\mathcal{V}}(\mathsf{x})) \wedge \mathcal{V}(\rho^{+}_{\mathcal{M},\mathcal{V}}(\mathsf{x}))]\!]$$

unless PA *is inconsistent.*

Proof. For a shake of contradiction, assume that $\mathcal{M}$ correctly accepts, then Lemma 10

$$\mathrm{PA} \vdash [\![\mathcal{M}(\rho^{+}_{\mathcal{M},\mathcal{V}}(\mathsf{a})) \wedge \mathcal{V}(\rho^{+}_{\mathcal{M},\mathcal{V}}(\mathsf{a}))]\!]$$

As $\mathcal{V}$ soundly accepts, we get

$$\mathrm{PA} \models \rho^{+}_{\mathcal{M},\mathcal{V}}(\mathsf{a}) \qquad \Rightarrow \qquad \mathrm{PA} \models \neg[\![\mathcal{M}(\rho^{+}_{\mathcal{M},\mathcal{V}}(\mathsf{x})) \wedge \mathcal{V}(\rho^{+}_{\mathcal{M},\mathcal{V}}(\mathsf{x}))]\!]$$

and thus we have a proof of a false claim. □

Theorem 9 (First Incompleteness Theorem). *For any polynomial-time Turing machine $\mathcal{M}$ and unbounded sound verifier $\mathcal{V}$, $\mathcal{M}$ cannot correctly reject Gödel sentences $\rho^-_{\mathcal{M},\mathcal{V}}(x)$, i.e. for all $x \in \mathbb{N}$*

$$\mathrm{PA} \models \neg[\![\neg\mathcal{M}(\rho^-_{\mathcal{M},\mathcal{V}}(\mathsf{x})) \wedge \neg\mathcal{V}(\rho^-_{\mathcal{M},\mathcal{V}}(\mathsf{x}))]\!]$$

unless PA *is inconsistent.*

Proof. For a shake of contradiction, assume that $\mathcal{M}$ correctly rejects, then Lemma 10

$$\mathrm{PA} \vdash \neg[\![\neg\mathcal{M}(\rho^-_{\mathcal{M},\mathcal{V}}(\mathsf{a})) \wedge \neg\mathcal{V}(\rho^-_{\mathcal{M},\mathcal{V}}(\mathsf{a}))]\!]$$

As $\mathcal{V}$ soundly rejects, we get

$$\mathrm{PA} \models \neg\rho^-_{\mathcal{M},\mathcal{V}}(\mathsf{a}) \qquad \Rightarrow \qquad \mathrm{PA} \models [\![\mathcal{M}(\neg\rho^-_{\mathcal{M},\mathcal{V}}(\mathsf{x})) \wedge \neg\mathcal{V}(\rho^-_{\mathcal{M},\mathcal{V}}(\mathsf{x}))]\!]$$

and thus we have a proof of a false claim. □

10 Towards second incompleteness theorem

Lemma 18. *Fix two dedicated verifiers $\mathcal{V}^+_\Omega$ and $\mathcal{V}^-_\Omega$. Then for any Turing machine $\mathcal{M}$ and $\phi(x)$, there exists another $\mathcal{M}_\circ$ such that*

$$\mathrm{PA} \vdash \forall x(\neg[\![\mathcal{M}_\circ(\phi(\mathsf{x})) \wedge \mathcal{V}^+_\Omega(\phi(\mathsf{x}))]\!] \supset \rho^+_{\mathcal{M},\mathcal{V}_\Omega}(\mathsf{x})$$
$$\mathrm{PA} \vdash \forall x(\neg[\![\neg\mathcal{M}_\circ(\phi(\mathsf{x})) \wedge \neg\mathcal{V}^-_\Omega(\phi(\mathsf{x}))]\!] \supset \neg\rho^-_{\mathcal{M},\mathcal{V}_\Omega}(\mathsf{x}))$$

Proof. For brevity let $\psi^+ = \rho^+_{\mathcal{M},\mathcal{V}_\Omega}(x)$ and $\psi^- = \rho^-_{\mathcal{M},\mathcal{V}_\Omega}(\mathsf{x})$.
Lemma 14/15 we get that there is $\mathcal{M}_1$ such that

$$\mathrm{PA} \vdash \forall x([\![\mathcal{M}(\psi^+(\mathsf{x})) \wedge \mathcal{V}^+_\Omega(\phi(\mathsf{x}))]\!] \supset [\![\mathcal{M}_1([\![\mathcal{M}(\phi(\mathsf{x})) \wedge \mathcal{V}^+_\Omega(\phi(\mathsf{x}))]\!]) \wedge \mathcal{V}^+_\Omega([\![\mathcal{M}(\phi(\mathsf{x})) \wedge \mathcal{V}^+_\Omega(\phi(\mathsf{x}))]\!])]\!]$$
$$\mathrm{PA} \vdash \forall x([\![\neg\mathcal{M}(\psi^-(\mathsf{x})) \wedge \neg\mathcal{V}^+_\Omega(\phi(\mathsf{x}))]\!] \supset [\![\neg\mathcal{M}_1([\![\neg\mathcal{M}(\phi(\mathsf{x})) \wedge \neg\mathcal{V}^+_\Omega(\phi(\mathsf{x}))]\!]) \wedge \neg\mathcal{V}^+_\Omega([\![\neg\mathcal{M}(\phi(\mathsf{x})) \wedge \neg\mathcal{V}^+_\Omega(\phi(\mathsf{x}))$$

Next, recursiverly install definitions of Gödel sentences[2]

$$\mathrm{PA} \vdash \forall x(\rho^+_{\mathcal{M},\mathcal{V}}(x) \sim \neg[\![\mathcal{M}(\rho^+_{\mathcal{M},\mathcal{V}}(\mathsf{x})) \wedge \mathcal{V}(\rho^+_{\mathcal{M},\mathcal{V}}(\mathsf{x}))]\!])$$
$$\mathrm{PA} \vdash \forall x(\rho^-_{\mathcal{M},\mathcal{V}}(x) \sim \neg[\![\neg\mathcal{M}(\rho^-_{\mathcal{M},\mathcal{V}}(\mathsf{x})) \wedge \neg\mathcal{V}(\rho^-_{\mathcal{M},\mathcal{V}}(\mathsf{x}))]\!])\ .$$

into the verifiers. Hence Lemma 12/13 allows to build

□

[2]See wether such Verifier can be built at all