Friedberg–Muchnik Theorem in Synthetic Computability

Current Progress: A Brief Overview

Haoyi Zeng
Saarland University
May 16, 2024

1 Introduction

"As a result we are left completely on the fence as to whether there exists a recursively enumerable set positive integers of absolutely lower degree of unsolvability than the complete set K, ..."

EMIL L. POST

Post's Problem, posed by Emil Post in 1944 [9], is a significant question in computability theory and the theory of degrees. The problem asks to prove the existence of a recursively enumerable degree between the degree of the empty set $\varnothing$ and the Turing jump of the empty set $\varnothing'$. Post himself made progress on the problem, but it remained unsolved until Friedberg [5] and Muchnik [8] independently provided a solution in 1956 and 1957, respectively.

The Friedberg–Muchnik Theorem, as a result of solving Post's Problem, is important because it marked a breakthrough in understanding the structure of recursively enumerable degrees. The theorem demonstrates that there exists an r.e. degree strictly between $\varnothing$ and $\varnothing'$, providing insight into the hierarchy of degrees and expanding our understanding of the computability landscape.

The methods used in the proof, known as finite priority arguments, involve recursive constructions with infinitely many stages and resolving conflicts among requirements according to a recursive scheme of priorities. This approach was significant not only for solving Post's Problem but also became a fundamental technique in local degree theory. The Friedberg–Muchnik method was later extended by other researchers, such as Lachlan, Sacks, and Yates, contributing to the development of this area of study.

2 Constructive Type Theory

These are some of the basic inductive types that will be used in this thesis:

- Unit: $\mathbb{1} : \mathfrak{T} \ ::= \ \star : \mathbb{1}$
- Boolean: $\mathbb{B} : \mathfrak{T} \ ::= \ \mathsf{tt} : \mathbb{B} \mid \mathsf{ff} : \mathbb{B}$
- Natural numbers: $\mathbb{N} : \mathfrak{T} \ ::= \ O : \mathbb{N} \mid S : \mathbb{N} \to \mathbb{N}$
- Option types: $\mathcal{O}(T : \mathfrak{T}) : \mathfrak{T} \ ::= \ \ulcorner\_\urcorner : T \to \mathcal{O}(T) \mid \mathsf{none} : \mathcal{O}(T)$
- Lists: $(T : \mathfrak{T})^* : \mathfrak{T} \ ::= \ \mathsf{nil} : T^* \mid \ :: \ : T \to T^* \to T^*$
- Sum types: $X + Y : \mathfrak{T} \ ::= \ \mathsf{injl} : X \to X + Y \mid \mathsf{injr} : Y \to X + Y$

We default the capital letters $X\ Y\ Z : \mathfrak{T}$ to arbitrary types, while $p\ q : X \to \mathfrak{P}$ generally express arbitrary predicates over X.

The characteristic relation $\hat{p} : X \to \mathbb{B} \to \mathfrak{P}$ of a predicate $p : X \to \mathfrak{P}$ defined by:

$$\hat{p} \ \coloneqq \ \lambda x\ b.\ \begin{cases} p\ x & \text{if } b = \mathsf{tt} \\ \neg\ p\ x & \text{if } b = \mathsf{ff} \end{cases}$$

3 Synthetic Computability

!!TODO:You can check Yannick's thesis [1], this paper [3] about Oracle Computability and also this paper [4] about Arithmetic hierarchy.

4 Limit Lemma

In the pursuit of exploring the Turing degree, the concept of limit computability serves as a fundamental tool for investigating reducibility and the jump operator. This notion was initially introduced by Gold [6] in 1964, although Shoenfield's work [10] in 1959 already proved the limit lemma.

4.1 Limit Computable

A predicate P is considered limit computable if there exists a computable and total guessing function $f(x, n)$ such that, for any fixing x, whether x belongs to P or not depends on whether the function $f(x, n)$ converges to tt or to ff.

In the context of synthetic computability, a function $f : X \to \mathbb{N} \to \mathbb{B}$ over arbitrary type X is employed as a uniform sequence of computable binary function, it can also be viewed as a uniform sequence of characteristic function of decidable sets. A uniform sequence is convergence to some value b at fixing x is defined as follows:

$$\exists N : \mathbb{N}.\ \forall n \geq N.\ f(x, n) = b$$

This can also be expressed as $\lim_{n\to\infty} f(x, n) = b$.

Definition 1 (Limit Computable)**.** A given characteristic relation $\hat{P} : X \to \mathbb{B} \to \mathfrak{P}$ is termed limit computable when there is a decider f s.t.:

$$\forall x\ b.\ \hat{P}(x, b) \iff \lim_{n\to\infty} f(x, n) = b$$

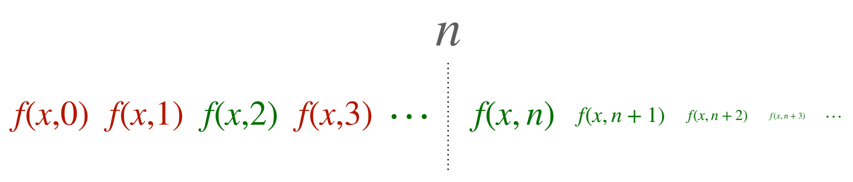

Figure 1: x is in P

Let P be a limit computable predicate defined by a function f. The determination of whether x is in P can be made by inserting x into f and then executing the function f stage by stage.

Next, we can execute this function up to some stage n. However, whether $f(x, n)$ returns tt (expressed in green) or ff (expressed in red) does not provide a means to determine whether x satisfies P. All that can be known about limit computable is that there exists a sufficiently large n such that, for any subsequent stage, the function always returns tt if and only if $P\ x$, with the opposite being ff for the negative case.

4.2 Limit Lemma 1

Limit computable is intuitively well-understood, while the limit lemma positions limit computable within the arithmetic hierarchy, specifically closely linked to the concept of a Δ_2-predicate. This lemma acts as a connection between computable approximations and the arithmetic hierarchy, playing a pivotal role in the examination of degrees of unsolvability and other areas within mathematical logic and recursion theory.

Lemma 1. For any predicate P, if P is limit computable, then both P and P are Σ_2-predicates.

Proof. Examining the definition of limit computable, we can express it as follows:

$$\begin{aligned} P\ x &\iff \exists n.\ \forall m \geq n.\ f(x, m) = \mathsf{tt} \\ \neg P\ x &\iff \exists n.\ \forall m \geq n.\ f(x, m) = \mathsf{ff} \end{aligned}$$

This shows that the limit computable predicate P is a Σ_2 predicate, and its complement is also Σ_2 predicate. □

Review the Post's Theorm that will be used.

Theorem 1 (Post's Theorem)**.** For any Σ_2-predicate P, P is semi-decidable in $\varnothing'$.

$$P \in \Sigma_2 \Rightarrow S_{\varnothing'}(P)$$

Under the classical assumption LEM_{Σ_1}:

$$S_Q(P) \land S_Q(\bar{P}) \Rightarrow P \preceq Q$$

Proof. Reading this paper [4]. □

Lemma 2 (Limit Lemma 1)**.** For any predicate P, if P is limit computable, then P is Turing reducible to $\varnothing'$ by assuming LEM_{Σ_1}.

Proof. Limit computable P and its complements are Σ_2 according to the Lemma 1.

By applying a lemma from the arithmetic hierarchy, a Σ_2 predicate is oracle semi-decidable in the $\varnothing'$, leading to reducibility to the $\varnothing'$ under the classical assumption LEM_{Σ_1}. $\square$

4.3 Limit Lemma 2

Before proving the other direction, it's worth noting that semi-decidable predicates can all be expressed as limit computable. Furthermore, we can establish an accumulative computable sequence, defined as follows.

Definition 2 (Σ_1 Approximation)**.** For any semi-decidable predicate, there exists a Σ_1 approximation, indicating the presence of an accumulative sequence $f : X \to \mathbb{N} \to \mathbb{B}$ such that:

$$P(x) \iff \exists n.\ f(x, n) = \mathsf{tt}$$

The concept of accumulation means that for any n and x:

$$f(x, n) = \mathsf{tt} \Rightarrow \forall m \geq n.\ f(x, m) = \mathsf{tt}$$

Proof. The proof can be executed straightforwardly by running the semi-decider for n iterations and gathering the outcomes of all terminations. For instance, in the halting problem, it can be defined as the assessment of the program for n steps of termination. $\square$

The Σ_1 approximation of predicate p is also denoted by $p[n]$, which is a family of decidable predicate such that $f(x, n) = \mathsf{tt} \iff p[n]\ x$.

This offers a fundamentally different perspective. Instead of examining a predicate statically, we approach it dynamically. That is, there exists a computable process to approximate its behavior, and we can straightforwardly derive a fact as follows.

Fact 2. In the context of a Σ_1 approximation f, for any given list of problems l, a sufficiently large number n such that for larger m, there consistently exists the same result within this list:

$$\exists n.\ \forall m \geq n.\ \forall x \in l.\ f(x, m) = f(x, n)$$

Proof. By definition. $\square$

Now, let's see how to execute the Turing machine with an Oracle. When dealing with synthetic computability, envisioning the execution of a Turing machine for n steps, i.e., running the corresponding partial function for n steps, is not challenging.

However, in the more complicate of Oracle computable, the computation of a Turing machine with an oracle is defined by a partial computable tree τ. Consequently, a step-index function that precisely indexes the steps of this function is not straightforward. Nevertheless, we can provide the following definition.

Definition 3 (Step-Index Function)**.** For some computable F, there exists a corresponding computable tree τ. Let's consider a semi-decidable predicate p, and the corresponding characteristic function $\chi_p : \mathbb{N} \to X \to \mathbb{B}$.

We insert this oracle p into our oracle machine by fixing a n, and subsequently run τ. Based on this effectively computable oracle, we can define a total function Φ as follows:

$$\Phi_\tau^{p[n]}\ x\ i\ j \ :=\ \begin{cases} \ulcorner \mathsf{out}\ o \urcorner & \text{if } (\tau\ x\ \mathsf{nil}) \rightsquigarrow_j \mathsf{out}\ o \\ \ulcorner \mathsf{ask}\ q \urcorner & \text{if } (\tau\ x\ \mathsf{nil}) \rightsquigarrow_j \mathsf{ask}\ q \text{ and } \ \ i = 0 \\ \Phi^{p[i']}_{\lambda r.\ \tau\ x\ \chi_p(q,n)::r}\ x\ i'\ j & \text{if } (\tau\ x\ \mathsf{nil}) \rightsquigarrow_j \mathsf{ask}\ q \text{ and } \ \ i = S\ i' \\ \mathsf{none} & \text{otherwise} \end{cases}$$

where the notation is defined as $x \rightsquigarrow_j o \ :=\ \texttt{seval}\ x\ j = \ulcorner o \urcorner$, means to perform the partial function within j steps.

This function is grounded in the intuition that, given our computable tree being partial, i represents the maximum depth that can be explored, and j is the maximum number of steps allowed to run at each node.

In short we want to encapsulate this function and give some of the properties we need.

Fact 3 (Monotonicity)**.** For any semi-decidable predicate p, the step-indexed function Φ is monotonic with respect to the arguments step i and deepth j:

$$\Phi_\tau^{p[n]}\ x\ i\ j = \ulcorner s \urcorner \Rightarrow \forall j' \geq j.\ \Phi_\tau^{p[n]}\ x\ i\ j' = \ulcorner s \urcorner$$
$$\Phi_\tau^{p[n]}\ x\ i\ j = \ulcorner \mathsf{out}\ o \urcorner \Rightarrow \forall i' \geq i.\ \Phi_\tau^{p[n]}\ x\ i'\ j = \ulcorner \mathsf{out}\ o \urcorner$$

A similar property holds for the oracle when it is a cumulative computable sequence, implying that the oracle is semi-decidable. For any successive oracle, it will consistently be closer to the predicate it is approximating than the previous one. In other words, if the oracle has a Σ_1 approximation, then the index of this approximation is also monotonic.

These observed monotonic indicate that these parameters can jointly approximate the expected output. Hence, we define a shorthand, also known as Lachlan notation [14]:

$$\Phi_\tau^p\ (x)[n] \ :=\ \Phi_\tau^{p[n]}\ x\ n\ n$$

The step-indexed oracle machine's completeness property can then be expressed as follows when taking a semi-decidable predicate as oracle.

Lemma 3. *Let p be a semi-decidable predicate, $p[n]$ is the n-th Σ_1-approximation of p.*

$$\Xi_e\ \hat{p}\ x\ \star \rightarrow \lim_{n\to\infty} \Phi_e^p(x)[n] = \ulcorner \star \urcorner$$

Now, we can set aside the definition of Φ while retaining its properties to finalize the proof of the Limit Lemma.

Lemma 4 (Limit Lemma 2)**.** *A predicate P is limit computable when $P \preceq \varnothing'$ by assuming LEM_{Σ_1} and P is logical decidable.*

Proof. Given that P is Turing reducible to $\varnothing'$, we obtain the computable tree τ. Building upon the step-index function described earlier, we define the following function:

$$\chi_P(s,x) \ :=\ \begin{cases} b & \text{if } \Phi_\tau^K(x)[s] = \ulcorner b \urcorner \\ \mathsf{tt} & \text{otherwise} \end{cases}$$

We show that:

$$\forall x\ b.\ \hat{P}\ x\ b \iff \lim_{s\to\infty} \chi_P(s,x) = b$$

If the Turing machine terminates with output o, then there exists a sufficiently accurate approximation of the oracle through continuous modules. This ensures that the machine terminates within a large enough number s of steps, guaranteeing that $\Phi_\tau^K(x)[s]$ converges to o.

Due to the logical decidability of P, we ascertain that $\chi_P(s,x)$ cannot simultaneously converge to both tt and ff. Therefore, if $\chi_P(s,x)$ converges to b, it implies the fact that $\hat{P}(x,b)$. $\square$

5 Low Simple Set

5.1 Introdution to Post's Problem

The concept of Turing degree, introduced by Post [9], plays a important role in clarifying undecidable problem. We commonly use the symbol $\varnothing$ to represent all decidable problems. Any problem P that is in the same degree as the halting problem K is denoted as Turing jump of decidable problem:

$$\varnothing' := \{P \mid P \equiv K\}$$

where $P \equiv Q \ := \ P \preceq Q \wedge Q \preceq P$ is denoted Turing reducible between the problem P and Q.

Particular attention is given to the degree that lies between $\varnothing$ and $\varnothing'$, also known as the local Turing degree [11]. Naturally, Post raises the question: Is there any semi-decidable problem Q strictly between $\varnothing$ and $\varnothing'$, i.e.,

$$\varnothing \prec Q \prec K.$$

The notation is defined as $P \prec Q \ := \ P \preceq Q \wedge P \not\equiv Q$, implying that, even Q as an oracle, cannot be used to solve the halting problem.

In Post's paper, he partially addressed this problem by constructing simple sets and hypersimple sets to answer the problem in the m-degree and truth-table degree, respectively. For discussions on synthetic computability, refer to the paper [2].

It wasn't until the 1950s that Friedberg [5] and Muchnik [8] independently provided a solution Post's Problem to Turing degrees using a new method called finite injury priority method. Subsequently, the finite injury priority method and it variants became one of the most important techniques in computability theory, widely employed in proving various theorems[12].

Moving forward, we aim to discuss the solution to Post's Problem in synthetic computability using the finite injury priority method and formalize it. This is a complex proof, posing a challenge in the field of mathematics formalization. Nevertheless, it marks a significant milestone, providing foundational techniques for exploring more advanced results in synthetic computability.

5.2 Finite Injuru Priority Method

We begin with a simplified version of the solution, constructing a low simple set. This construction, initially employed by Lerman and Soare in their paper [7], was later modified to become an example in standard textbooks [13, 14] and is considered one of the simplest solutions to Post's Problem using the finite injury priority method. Therefore, it can be used as a first step in how the finite injury priority method works and how powerful it is.

The idea of the priority method is to build the semi-decidable predicate step by step, adding a at most one element to the predicate at each step through a computable process that carries the information of current step, the constructed predicate should then be verified to meet a priority-ordered list of requirements.

We start from the way how a predicate be cosntructed. The semi-decider can be defined as a function $f : X \to \mathbb{N} \to \mathbb{B}$, which gives rise to the perspective that the priority method in synthetic computability is building a semi-decider recursively on the step index.

To make the proof simpler, we want to build and prove the predicate in a modular fashion, making it possible to construct a complex proof step by step. The inductive type $\rightsquigarrow$ is employed by abstracting the extension $\gamma : \mathbb{N}^* \to \mathbb{N} \to \mathbb{N} \to \mathfrak{P}$.

$$\frac{}{0 \rightsquigarrow [\,]} \qquad \frac{n \rightsquigarrow L \quad \gamma_n^L\ x}{n+1 \rightsquigarrow x :: L} \qquad \frac{n \rightsquigarrow L \quad \forall x.\ \neg\ \gamma_n^L\ x}{n+1 \rightsquigarrow L}$$

The predicate $n \rightsquigarrow L$ simply checks at each stage n whether a new element x can enter the L by asking the extension γ; if not, proceed to the next stage without changing the L; otherwise, concatenate x to the L.

In order for the L being accepted at each stage n to be decidable and unique, the extension must also be decidable and unique.

Definition 4 (Extension)**.** As long as the conditions below are met, we call the $\gamma : \mathbb{N}^* \to \mathbb{N} \to \mathbb{N} \to \mathfrak{P}$ an extension.

$$(\Sigma x.\ \gamma_n^L\ x) + (\forall x.\ \neg\ \gamma_n^L\ x)$$

$$\forall x\ y.\ \gamma_n^L\ x \to \gamma_n^L\ y \to x = y$$

Fact 4. For any extension γ, the following properties of $n{\rightsquigarrow}L$ hold.

- Unique: $\forall n\ L_1\ L_2.\ n \rightsquigarrow L_1 \to n \rightsquigarrow L_2 \to L_1 = L_2$
- Monotonic: For any stage n, m, if $n \rightsquigarrow L_1$ and $n \rightsquigarrow L_2$ for $n \leq m$, then $L1 \subseteq L_2$.
- Inversion: If $n \rightsquigarrow x :: L$ at some stage n, then there must be some stage m, such that $m \rightsquigarrow L$ and $\gamma_m^L\ x$ hold.

The predicate P_γ that is constructed over γ is defined by:

$$x \in P_\gamma \ := \ \exists n\ L.\ n \rightsquigarrow L \land x \in L$$

Lemma 5. For any extension γ, P_γ is semi-decidable, and there is a function $\Gamma : \mathbb{N} \to \mathbb{N}^*$ such that:

$$x \in P_\gamma \iff \exists n.\ x \in \Gamma_n$$

Notice that $x \in \Gamma_n$ is a Σ_1-approximation of P_γ.

We have the fact that any predicate constructed with any extension γ is semi-decidable. This is the first layer of the construction, which we can turn into the construction of the extension. As we do in this section, we begin with a simple property, literally "simple".

5.3 The Simple Extension

The simple predicate was defined and constructed by Post to demonstrate the existence of an m-incomplete, semi-decidable, and undecidable predicate. As an initial attempt at addressing Post's problem, the simple predicate provide a foundation by showing that such predicates are undecidable. The simple predicate we will construct, following Post's approach, is m-incomplete and Turing-complete. While we haven't solved Post's problem, this construction provides a framework that can be adjusted to create a Turing-incomplete simple predicate. Now, let's explore how to construct the extension γ so that P_γ is clear.

Definition 5 (Simple Predicate)**.** A predicate P is *simple* if p is semi-decidable, the complement of p, i.e. $\bar{p}$ is non-finite, and for any semi-predicate q, q is not a sub-predicate of $\bar{p}$.

$$\text{simple } P \;\coloneqq\; \mathcal{E}\ p \land \mathcal{X}(\bar{p}) \land \neg\exists q.\ \mathcal{E}\ q \land q \subseteq p$$

where the notation $q \subseteq p$ is defined as $\forall x.\ q\ x \to p\ x$.

Theorem 5. Simple predicate is are undecidable and m-incomplete.

In this section, we will reuse some of the findings from this work [2], where a simple predicate was established synthetically by explicitly defining it and verifying its simpleness. Instead of presenting the predicate directly, we'll build it withing the priority method using the following extension.

Definition 6 (Simple Extension)**.** The extension primarily follows Soare's design. At any stage n and for any list L, the extension checks whether L intersects with the first n elements of the e-th enumerator $W_e[n]$ or not. If there is such e and x such that $x \in W_e[n]$ and $2e < x$, we pick the least e and the corresponding least x as the next element to add to the predicate. Since the priority method always receives the list Γ_n, the extension extends the predicate at stage n by verifying all entries in P_γ before stage n.

$$\begin{aligned} \pi_n^L\ e\ x \;&\coloneqq\; x \in \mathcal{W}_e[n] \land 2e < x \\ \Pi_n^L\ e \;&\coloneqq\; L \cap \mathcal{W}_e[n] = \varnothing \land \exists x.\ \pi_n^L\ e\ x \\ \gamma_n^L\ x \;&\coloneqq\; \exists e.\ e < n \land \mathcal{L}\ e.\ \Pi_n^L \land \mathcal{L}\ x.\ \pi_n^L\ e \end{aligned}$$

The notation $\mathcal{L}\ x.\ p$ denotes that x is the least element that satisfies the predicate p as follows:

$$\mathcal{L}\ x.\ p \;\coloneqq\; p\ x \land \forall y < x.\ \neg p\ y$$

Fact 6. The predicate γ defined above is an extension.

To prove that the predicate P_γ is simple, we must prove three properties listed in the definition of the simple predicate. Because we construct the P_γ by instantiating γ, P_γ is semi-decidable.

$$P_e \;\coloneqq\; \mathcal{W}_e \text{ is infinite} \to \mathcal{W}_e \cap A \neq \varnothing$$

Lemma 6 (Verification)**.** The predicate P_γ satisfies the requirements P_e.

Fact 7 (Correctness)**.** $\bar{P}_\gamma$ is non-finite.

Fact 8 (Correctness)**.** For any semi-decidable predicate q, q is not a sub-predicate of $\bar{P}_\gamma$.

If we carefully examine the definition of extension, the requirement that the element x must exceed $2 \cdot e$ to progress to the next stage is actually aimed at ensuring that $\bar{P}_\gamma$ remains non-finite. This condition can be generalized to any function.

$$\pi_n^L \ e \ x \ := \ x \in \mathcal{W}_e[n] \land \omega_n^L(e) < x$$

Definition 7 (Wall Function)**.** The *wall function* is defined as a function $\omega : \mathbb{N} \to \mathbb{N}^* \to \mathbb{N} \to \mathbb{N}$ that meet the following conditions:

$$2 \cdot e \leq \omega_n^{\Gamma_n}(e)$$
$$\exists b. \ \lim_{n \to \infty} \omega_n^{\Gamma_n}(e) = b$$

where the second condition be relaxed to:

$$\neg\neg\exists b. \ \lim_{n \to \infty} \omega_n^{\Gamma_n}(e) = b$$

Lemma 7. For any wall function ω, the predicate P_γ is simple by instantiating the simple extension γ within ω.

5.4 The low wall

To construct a low simple predicate, we must define a concrete wall function. In this section, the wall function is defined as the maximum of the function $2 \cdot e$ and the use function. The definition of the *use function* involves the model of the oracle machine.

The use function, $\varphi_e^A(x)[n]$, captures the highest question asked during the computation of $\Phi_e^A(x)[n]$. A critical feature of this function is that if an element greater than $\varphi_e^A(x)[n]$ is added to the predicate A, it does not alter the outcome of the computation $\Phi_e^A(x)[n]$.

Lemma 8 (Completeness)**.** Let $k = \varphi_e^A(e)[n]$, the use function must satisfy the following condition:

$$\Phi_e^p(e)[n] = \ulcorner\star\urcorner \to \forall q. \ q \equiv_k p[n] \to \Xi_e \ \hat{q} \ e \ \star$$

Lemma 9 (Monotonic)**.** The intuition behind this property is that any element that does not affect the computation (i.e., is greater than the output of the use function) also does not alter the use function itself.

$$\varphi_e^p(x)[n] = k \to p[n] \equiv_k p[n+1] \to \varphi_e^p(x)[n+1] = k$$

Definition 8 (Low Wall)**.** The low wall function is defined as following:

$$\omega_n^L(e) \ := \ \max(2 \cdot e, \varphi_e^{x \in L}(e)[n])$$

Fact 9. The low wall function ω is convergent:

$$\neg\neg\exists b. \ \lim_{n \to \infty} \omega_n^{\Gamma_n}(n)(e) = b$$

Proof. TODO □

The requirements of lowness claim that the e-th oracle machine with oracle A terminates on e if it terminates on infinitely many approximations.

$$N_e \ := \ \exists^\infty s. \ \Phi_e^A(e)[s] \downarrow \to \Phi_e^A(e) \downarrow$$

Theorem 10 (Verification)**.** The predicate P_γ satisfies the requirements N_e.

Theorem 11 (Correctness)**.** Any predicate that satisfies both P_e and N_e is both low and simple.

5.5 Solution to Post's Problem

Since a simple predicate is undecidable yet semi-decidable, and because any predicate of lowness cannot be reduced from the halting problem, therefore, the low simple predicate provides an answer to Post's Problem, offering a positive solution.

6 Friedberg–Muchnik Theorem

!!TODO:The first solution to Post's problem independently proved by Friedberg and Muchnik

7 Conclusion

In this project, we formalise the construction of the low simple predicate in constructive type theory and mechanise our results in the Coq proof assistant, which give the first formalisation of the solution to Post's Problem.

References

1. Forster, Y. Computability in constructive type theory.

2. Forster, Y., and Jahn, F. Constructive and synthetic reducibility degrees: Post's problem for many-one and truth-table reducibility in coq. In *CSL 2023-31st EACSL Annual Conference on Computer Science Logic* (2023).

3. Forster, Y., Kirst, D., and Mück, N. Oracle computability and turing reducibility in the calculus of inductive constructions. *ArXiv abs/2307.15543* (2023).

4. Forster, Y., Kirst, D., and Mück, N. The kleene-post and post's theorem in the calculus of inductive constructions.

5. Friedberg, R. M. Two recursively enumerable sets of incomparable degrees of unsolvability (solution of post's problem, 1944). *Proceedings of the National Academy of Sciences 43*, 2 (1957), 236–238.

6. Gold, E. M. Limiting recursion. *The Journal of Symbolic Logic 30*, 1 (1965), 28–48.

7. Lerman, M., and Soare, R. d-simple sets, small sets, and degree classes. *Pacific Journal of Mathematics 87*, 1 (1980), 135–155.

8. Muchnik, A. A. On the unsolvability of the problem of reducibility in the theory of algorithms. In *Dokl. Akad. Nauk SSSR* (1956), vol. 108, pp. 194–197.

9. Post, E. L. Recursively enumerable sets of positive integers and their decision problems.

10. Shoenfield, J. R. On degrees of unsolvability. *Annals of mathematics 69*, 3 (1959), 644–653.

11. Simpson, S. G. Degrees of unsolvability: a survey of results. In *Studies in Logic and the Foundations of Mathematics*, vol. 90. Elsevier, 1977, pp. 631–652.

12. Soare, R. I. The infinite injury priority method1. *The Journal of Symbolic Logic 41*, 2 (1976), 513–530.

13. Soare, R. I. *Recursively enumerable sets and degrees: A study of computable functions and computably generated sets.* Springer Science & Business Media, 1999.

14. Soare, R. I. Turing computability. *Theory and Applications of Computability. Springer* (2016).