Friedberg–Muchnik Theorem in Synthetic Computability

Current Progress: A Brief Overview

Haoyi Zeng
Saarland University
November 30, 2023

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 [Pos44], 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 $\emptyset$ and the Turing jump of the empty set $\emptyset'$. Post himself made progress on the problem, but it remained unsolved until Friedberg [Fri57] and Muchnik [Muc56] 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 $\emptyset$ and $\emptyset'$, 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: $\mathcal{L}(T : \mathfrak{T}) : \mathfrak{T} ::= [] : \mathcal{L}(T) \mid :: : T \to \mathcal{L}(T) \to \mathcal{L}(T)$
- 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:

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

3 Synthetic Computability

TODO: You can check Yannick's thesis [For21], this paper [FKM23] about Oracle Computability and also this paper [FKM24] 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 [Gol65] in 1964, although Shoenfield's work [Sho59] 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.:

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

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.

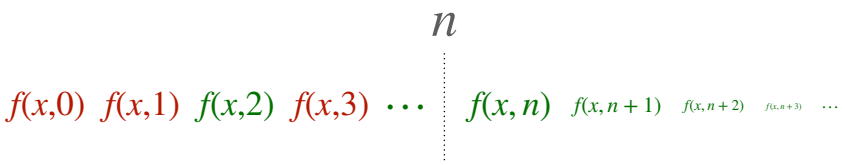

Figure 1: x is in P

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 $\overline{P}$ 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 $\emptyset'$.

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

Under the classical assumption LEM_{Σ_1}:

$$S_Q(P) \wedge S_Q(\bar{P}) \Rightarrow P \leq_T Q$$

Proof. Reading this paper [FKM24]. □

Lemma 2 (Limit Lemma 1)**.** For any predicate P, if P is limit computable, then P is Turing reducible to $\emptyset'$ 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 $\emptyset'$, leading to reducibility to the $\emptyset'$ under the classical assumption LEM_{Σ_1}. □

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 Approximatiom)**.** 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. □

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. □

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 computable sequence $O : \mathbb{N} \to X \to \mathfrak{P}$, or the corresponding characteristic function $\chi_O : \mathbb{N} \to X \to \mathbb{B}$.

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

$$\Phi_\tau^{O(n)}\ x\ i\ j\ :=\ \begin{cases} \ulcorner \mathsf{out}\ o \urcorner & \text{if } (\tau\ x\ []) \rightsquigarrow_j \mathsf{out}\ o \\ \ulcorner \mathsf{ask}\ q \urcorner & \text{if } (\tau\ x\ []) \rightsquigarrow_j \mathsf{ask}\ q \text{ and } i = 0 \\ \Phi_{\tau@[\chi_O\ n\ q]}^{O(n)}\ x\ i'\ j & \text{if } (\tau\ x\ []) \rightsquigarrow_j \mathsf{ask}\ q \text{ and } i = S\ i' \\ \mathsf{none} & \text{otherwise} \end{cases}$$

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, and it is not hard to see that this function is momotonic with respect to the arguments i and j:

Fact 3 (Monotonicity)**.**

$$\Phi_\tau^{O(n)}\ x\ i\ j = \ulcorner s \urcorner \Rightarrow \forall j' \geq j.\ \Phi_\tau^{O(n)}\ x\ i\ j' = \ulcorner s \urcorner$$
$$\Phi_\tau^{O(n)}\ x\ i\ j = \ulcorner \mathsf{out}\ o \urcorner \Rightarrow \forall i' \geq i.\ \Phi_\tau^{O(n)}\ x\ i'\ j = \ulcorner \mathsf{out}\ o \urcorner$$

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.

Lemma 3. If $\sigma\ x\ ;\ (\lambda x b.\ f x = b) \vdash \mathsf{qs}; \mathsf{ans}$ and $\sigma\ x\ \mathsf{ans} \rhd o$, then for any function g such that $\sigma\ x\ ;\ (\lambda x b.\ g x = b) \vdash \mathsf{qs}; \mathsf{ans}$, there is some i, j:

$$\Phi_\tau^{(\lambda x b.\ g x = b)}\ x\ i\ j = \ulcorner o \urcorner$$

These observed monotonic indicate that these parameters can jointly approximate the expected output. Hence, we define a shorthand, also known as Lachlan notation [Soa16]:

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

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

TODO: More convenient lemma about Φ!

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

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

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

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 $\mathtt{tt}$ and $\mathtt{ff}$. Therefore, if $\chi_P(s,x)$ converges to b, it implies the fact that $\hat{P}(x,b)$. □

5 Low Simple Set

5.1 Introdution to Post's Problem

The concept of Turing degree, introduced by Post [Pos44], plays a important role in clarifying undecidable problem. We commonly use the symbol $\emptyset$ 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:

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

where $P \equiv Q \coloneqq P \preceq Q \land Q \preceq P$ is denoted Turing reducible between the problem P and Q.

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

$$\emptyset \prec Q \prec K.$$

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

In Post's paper, he partially addressed this problem using the priority method, constructing simple sets and hypersimple sets to answer Post's Problem in the m-degree and truth-table degree, respectively. For discussions on synthetic computability, refer to the paper [FJ23].

It wasn't until the 1950s that Friedberg [Fri57] and Muchnik [Muc56] independently provided a solution 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[Soa76].

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 [LS80], was later modified to become an example in standard textbooks [Soa99, Soa16] 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.

6 Friedberg–Muchnik Theorem

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

7 Conclusion

TODO

References

FJ23. Yannick Forster and Felix Jahn. 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.

FKM23. Yannick Forster, Dominik Kirst, and Niklas Mück. Oracle computability and turing reducibility in the calculus of inductive constructions. *ArXiv*, abs/2307.15543, 2023.

FKM24. Yannick Forster, Dominik Kirst, and Niklas Mück. The kleene-post and post's theorem in the calculus of inductive constructions. 2024.

For21. Yannick Forster. Computability in constructive type theory. 2021.

Fri57. Richard M Friedberg. Two recursively enumerable sets of incomparable degrees of unsolvability (solution of post's problem, 1944). *Proceedings of the National Academy of Sciences*, 43(2):236–238, 1957.

Gol65. E Mark Gold. Limiting recursion. *The Journal of Symbolic Logic*, 30(1):28–48, 1965.

LS80. Manuel Lerman and Robert Soare. d-simple sets, small sets, and degree classes. *Pacific Journal of Mathematics*, 87(1):135–155, 1980.

Muc56. Albert A Muchnik. On the unsolvability of the problem of reducibility in the theory of algorithms. In *Dokl. Akad. Nauk SSSR*, volume 108, pages 194–197, 1956.

Pos44. Emil L Post. Recursively enumerable sets of positive integers and their decision problems. 1944.

Sho59. Joseph R Shoenfield. On degrees of unsolvability. *Annals of mathematics*, 69(3):644–653, 1959.

Soa76. Robert I Soare. The infinite injury priority method1. *The Journal of Symbolic Logic*, 41(2):513–530, 1976.

Soa99. Robert I Soare. *Recursively enumerable sets and degrees: A study of computable functions and computably generated sets*. Springer Science & Business Media, 1999.

Soa16. Robert I Soare. Turing computability. *Theory and Applications of Computability. Springer*, 2016.