The Friedberg-Muchnik Theorem

Mathematical Logic for Computer Science

Simone Bianco, 1986936

Sapienza Università di Roma, Italy

June 11, 2025

Contents

1 Introduction

1.1 Decidability and semi-decidability

In 1936, Alan Turing's groundbreaking work introduced the concept of a function being computable by a **Turing machine** (TM), an abstract model of a computer that is capable of capturing the modern concept of computation with high precision while maintaining a simple interpretation. During his work, together with his mentor Alonzo Church who had already explored these concepts, Turing was able to use his notion of **computability** to prove that some problems are not algorithmically solvable, i.e. not computable by a Turing machine (and thus by a modern computer).

The most famous example of uncomputable problem is the Halting Problem, which asks to determine if a given program will halt or not for a given input. Formally, this problem can be described as the set $H = \{\langle M, x\rangle \mid M(x) \downarrow\}$, where $\langle M, x\rangle$ denotes a string encoding of the TM M and the input x, while $M(x) \downarrow$ denotes that M will halt when x is given as input. After Turing's work gave birth to the field of computability theory, researchers began to explore what is computable and what isn't. In particular, they were able to distinguish two types of computability, known as **decidability** and **semi-decidability**.

Definition 1 (Semi-decidability)**.** Given a subset $S \subseteq \Sigma^*$, we say that S is *semi-decidable* if there is an algorithmic procedure $\mathcal{A} : \Sigma^* \to \{0, 1\}$ such that $\forall x \in S$ it holds that $A(x) \downarrow = 1$.

In the above definition, we denote with Σ^* be the set of all strings over the alphabet Σ. We observe that this definition implies that for each string $x \in \overline{S}$ it can either hold that $A(x) \downarrow = 0$ or that $A(x) \uparrow$, where $A(x) \uparrow$ denotes that M will go into an infinite loop when x is given as input. When for all $x' \in \overline{S}$ it holds that $A(x) \downarrow = 0$ is the only option, we say that S is *decidable*.

Definition 2 (Decidability)**.** Given a subset $S \subseteq \Sigma^*$, we say that S is *decidable* if there is an algorithmic procedure $\mathcal{A} : \Sigma^* \to \{0, 1\}$ such that $\forall x \in S$ it holds that $A(x) \downarrow = 1$ while $\forall x \in \overline{S}$ it holds that $A(x) \downarrow = 0$.

In other words, semi-decidability implies that an algorithm is capable of recognizing a positive instance for a problem but cannot always recognize a negative instance, while decidability implies that an algorithm can distinguish between positive and negative instances. In fact, it's easy to see that a set $S \subseteq \Sigma^*$ is decidable if and only if both S and $\overline{S}$ are semi-decidable.

Turing showed that the set H describing the Halting problem is semi-decidable, but not decidable. In particular, both result were proven through the use of an Universal Turing Machine (UTM), i.e. a Turing machine that can take the description of a TM and simulate its execution for a given input. The semi-decidability of H can be simply proven through an UTM that takes the pair $\langle M, x\rangle$, simulates $M(x)$ and returns 1 if it halts, or going into an infinite loop otherwise. The undecidability of H, instead, can be proven through the use of the diagonalization technique: if we assume the existence of a TM M that decides H, then there must also be a machine $\overline{M}$ that returns the opposite result of M on any input, making the computation $M(\langle\overline{M}\rangle, \langle\overline{M}\rangle)$ halt if and only if it doesn't halt, raising a clear contradiction.

Semi-decidability can also be described with an equivalent concept, that being **recursive enumeration** (or r.e. for short). This latter concepts describes the natural property of a set of elements to be enumerated by an algorithm, meaning that there is a procedure that prints all the elements of the set. We observe that such procedure doesn't have to halt, as some sets are clearly infinite: the definition simply requires that an algorithm can achieve such task, even if it takes an infinite amount of time.

Definition 3 (Recursive enumerability)**.** Given a subset $S \subseteq \Sigma^*$, we say that S is *recursively enumerable* if there is an algorithmic procedure $\mathcal{A} : \mathbb{N} \to \Sigma^*$ that produces a list of all and only the elements inside it, meaning that $S = \{\mathcal{A}(0), \mathcal{A}(1), \ldots\}$.

From this very definition, it should seem natural that a set is semi-decidable if and only if it is recursively enumerable: if a set has a semi-deciding procedure then the latter can be used to test all inputs in parallel and print only those with a positive answer, while if a set has an enumerating procedure then all positive inputs will be eventually printed by such procedure.

1.2 Degrees of unsolvability

2 The Friedberg-Muchnik Theorem

2.1 Post's problem

2.2 The priority method