The Friedberg-Muchnik Theorem

Mathematical Logic for Computer Science

Simone Bianco, 1986936

Sapienza Università di Roma, Italy

June 11, 2025

Contents

1 Introduction **2**

1.1 Decidability and semi-decidability 2

1.2 Degrees of unsolvability 3

2 The Friedberg-Muchnik Theorem **5**

2.1 Post's problem 5

2.2 The priority method 5

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

From Turing's work and the underlying separation between computable and uncomputable problems, a natural question arises: is there a good measure of how much a problem is uncomputable? It turns out that a good enough measure can be obtained by other concepts that were developed by Turing himself, in particular **Turing reductions**. This reducibility notion describes the idea of a problem being solvable through the use of a machine that knows how to solve another problem. In particular, Turing formulated such notion through the use of *oracle machines*, that being Turing machines that have access to a black-box oracle (which can be seen as a magic tool or simply as another TM whose internal workings aren't known).

Definition 4 (Turing reduction)**.** Given two sets $A, B \subseteq \Sigma^*$, we say that A is *Turing reducible* to B, denoted with $A \leq_T B$, if there is an oracle TM M^B that has access to an oracle for B and that decides A.

Turing reductions are a generalized version of more commonly used *many-one reductions*, denoted with $A \leq_m B$, where the oracle TM is restricted to one single final query to the oracle. It's easy to see that Turing reductions define a partial order on the set 2^{Σ^*}. We also observe that there are clearly some problems that are equivalent to each other, either due to their definition (e.g. pairs of problems that ask the same question in two different ways) or to their intrinsic nature (e.g pairs of problems whose instances can be transformed back and forth). In particular, two problems $A, B \subseteq \Sigma^*$ are said to be **Turing equivalent**, written as $A \equiv_T B$, when $A \leq_T B$ and $B \leq_T A$. Intuitively, Turing equivalence defines an equivalence relation $\equiv_T$ that partitions the set 2^{Σ^*} into various classes of problems. Each class describes the complexity of the problems lying inside of it: for a problem $X \subseteq \Sigma^*$, each problem $Y \in [X]$ (where $[X]$ denotes the class of X over $\equiv_T$) is X*-complete*, meaning that it is perfectly interchangeable with the problem X.

When two problems lie in two different classes, at least one of them cannot be reduced to the other, implying that there is some intrinsic property that differentiates them. This allows us to view all of these classes as a *reduction hierarchy.* This hierarchy can be described in a natural way by a partial order over the classes and Turing reductions among them. In particular, given two classes $[X], [Y]$, we say that $[X] \leq [Y]$ if and only if $X \leq_T Y$. Intuitively, if $[X] \leq [Y]$ holds then each problem in the class $[Y]$ is "at least as hard" as each problem in the class $[X]$. This raises a natural concept of *hardness degree*, known as the **Turing degree**.

Definition 5 (Turing degree)**.** Given a subset $X \subseteq \Sigma^*$, the *Turing degree* of X is the equivalence class $[X]$ over the relation $\equiv_T$. We denote with $\mathcal{D}$ the set of all Turing degrees, i.e. the quotient set $\mathcal{D} = 2^{\Sigma^*}/_{\equiv_T}$.

The Turing degree acts as a measure that perfectly encapsulates the concept of algorithmic unsolvability. In fact, it's easy to see that any decidable problem has the same degree since for any pair $A, B \subseteq \Sigma^*$ of decidable sets we can decide A through a TM provided with an oracle for B simply by ignoring the oracle. We refer to the unique class of decidable problems as the degree 0 (or the degree $[\varnothing]$). Clearly, any problem that can be decided through an oracle for any set X of degree 0 is also also of degree 0. This implies that any problem outside of 0 must be undecidable under every single decidable oracle. The largest class of problems that lie outside of 0 is known as the class $0'$, where $0'$ denotes the **Turing jump** of 0.

Turing jump. Given a subset $X \subseteq \Sigma^*$, the *Turing jump* of X is a problem X' that cannot be Turing reduced to X, i.e. $X' \not\leq_T X$. $\square$

This definition of jump comes from the fact that the Turing degree strictly increases, meaning that $d < d'$ for any degree $d \in \mathcal{D}$. In particular, we observe that the famous Halting problem H is $0'$-complete, meaning that $0' = [H]$. To prove this, we give a stronger result.

Theorem 1. *A problem is recursively enumerable if and only if it can be Turing reduced to the Halting problem.*

Proof. Given $A \subseteq \Sigma^*$, suppose that $A \leq_T H$. Since H is recursively enumerable, its semi-decider can be used to semi-decide the instances of A also is. Vice versa, suppose that A is recursively enumerable. Then, there is a semi-decider M_1 for A. We can build a new machine M_2 that on input x simulates $M_1(x)$ and accepts if $M_1(x) = 1$, otherwise it loops. It's easy to see that $x \in A$ if and only if $\langle M_2, x\rangle \in H$, concluding that $A \leq_m H$ and thus that $A \leq_T H$. $\square$

The above theorem concludes that $0'$ is exactly the class of semi-decidable and undecidable problems. This also implies that every degree containing a recursively enumerable set is below $0'$. Such degrees are referred to r.e. degrees.

2 The Friedberg-Muchnik Theorem

2.1 Post's problem

2.2 The priority method