A Computational and Abstract Approach to Gödel's First Incompleteness Theorem

Benjamin Peters
Universität des Saarlandes

December 13, 2021

Gödel's first incompleteness theorem [4] in its modern form states that any consistent and sufficiently powerful formal system is incomplete. His original proof requires ω-consistency instead of consistency, a slightly stronger and more technical condition, but a few years after its publication Rosser [8] discovered a way to strengthen Gödel's result, giving it its modern form.

The theorem is also often presented in conjunction with computability theory: Assuming a sound and sufficiently powerful formal system, this approach usually shows that the halting problem would be decidable. This proof requires soundness instead of consistency or ω-consistency and does not construct an independent sentence, which both Gödel's and Rosser's proofs do.

Kleene [6] shows the incompleteness theorem in its modern form using computability theory by using a slightly different approach than usual, although it requires a different notion of power for the formal system. This proof was then presented in a different form in a blog post by Scott Aaronson [1], although his presentation does not mention the possibility of finding an explicit independent sentence. This approach was improved upon by an anonymous user on the Mathematics Stack Exchange [10] which was picked up by Anatoly Vorobey [11], who gave an overview on the different strengths in which the incompleteness theorem can be shown.

We give and compare two abstract proofs of Gödel's first incompleteness theorem using computability theory based on [10] in a constructive type theory, one in its weakest (soundness and no explicit independent sentence) and one in its strongest and modern (consistency and an explicit independent sentence) form.

A similar, but weaker statement has been shown and mechanized in [5].

1 Preliminaries

We work in the framework of a constructive type theory such as the one implemented in Coq. In such a theory all definable functions are total. To represent partial functions we need to resort to step-indexing:

Definition 1. *A function $f : X \to \mathbb{N} \to \mathcal{O}(Y)$ is called partial, if it is stationary, i.e.*

$$\forall xykk'.\, f\,x\,k = {}^{\circ}y \;\longrightarrow\; k' \geq k \;\longrightarrow\; f\,x\,k' = {}^{\circ}y.$$

In this case we write $f : X \rightharpoonup Y$.

Let $f, g : X \rightharpoonup Y$. We say that

- *f evaluates on x to y, written $f\,x \downarrow y$, if $\exists k.\, f\,x\,k = {}^{\circ}y$,*
- *f halts on x, written $f\,x\downarrow$, if $\exists y.\, f\,x \downarrow y$,*
- *f diverges on x, written $f\,x\uparrow$, if $\forall k.\; fx = \emptyset$,*
- *f is total, if $\forall x.\, f\,x\downarrow$,*
- *f and g return the same value on x, written $f\,x \equiv g\,x$, if $\forall y.\, f\,x \downarrow y \;\longleftrightarrow\; g\,x \downarrow y$,*
- *f and g are equivalent, written $f \equiv g$, if $\forall x.\, f\,x \equiv g\,x$.*

Lemma 1. *For any function $f : X \to \mathbb{N} \to \mathcal{O}(Y)$ that is functional, that is*

$$\forall x k_1 k_2 y_1 y_2.\, f\,x\,k_1 = y_1 \;\longrightarrow\; f\,x\,k_2 = y_2 \;\longrightarrow\; y_1 = y_2,$$

there is an equivalent partial function.

In our type theory it is consistent to assume that all functions that can be explicitly defined are computable. We make this formal by assuming a variant of Church's thesis [7, 9] as formulated by [2] in constructive type theory:

Assumption 1 (Church's thesis)**.** *There is a function $\theta : \mathbb{N} \to \mathbb{N} \rightharpoonup \mathbb{B}$, such that*

$$\forall f.\, \exists c.\, \forall x.\, f\,x \equiv \theta_c(x).$$

It is important to keep in mind that the conversion from functions to codes is not necessarily computable. For our purposes it suffices to consider Assumption 1 as shown above, which can be shown equivalent to a more canonical one with $f : \mathbb{N} \rightharpoonup \mathbb{N}$ and $\theta : \mathbb{N} \to \mathbb{N} \rightharpoonup \mathbb{N}$. We will write ! for boolean negation, `T` for boolean true and `F` for boolean false.

2 Formal systems

Definition 2 (Formal systems)**.** *A formal system* $\mathrm{FS} = (S, \neg, \vdash)$ *consists of a type* $S : \mathbb{T}$ *of logical sentences, a negation function* $\neg : S \to S$ *and a provability predicate* $\vdash\, : S \to \mathbb{P}$ *fulfilling the following properties:*

- S *is enumerable and discrete.*
- $\vdash$ *is enumerable.*
- FS *is consistent:* $\forall s.\, \neg(\vdash s \land \vdash \neg s)$.

FS can easily be instantiated with any reasonable formal logic with respect to a set of axioms, e.g. first- or second-order logic with a natural deduction system and the axioms of Robinson's Q or Peano arithmetic.

Definition 3 (Completeness)**.** *A formal system* $(S, \neg, \vdash)$ *is complete if every sentence can either be proven or disproven:* $\forall s.\, \vdash s \lor \vdash \neg s$.

Lemma 2 (Deep negations)**.** *Let* $(S, \neg, \vdash)$ *be a complete formal system and* $s : S$ *be a sentence. Then* $\vdash \neg s \longleftrightarrow\; \nvdash s$.

Proof. $\longrightarrow$ by consistency, $\longleftarrow$ by completeness. □

Lemma 3 (Co-enumerability)**.** *In any complete formal system* $(S, \neg, \vdash)$, $\vdash$ *is co-enumerable.*

Proof. We can construct a semi-decider for $\lambda s.\, \vdash \neg s$ instead of a co-enumerator for $\vdash$ using Theorem 2 and enumerability of S. Given a sentence s, enumerate all provable sentences s'. If $\neg s = s'$, then accept, otherwise continue searching.

Now, s is disprovable iff the semi-decider accepts s. □

Theorem 4 (Decidability)**.** *In any complete formal system* $\mathrm{FS} = (S, \neg, \vdash)$, $\vdash$ *is decidable.*

Proof. We get enumerability from the definition of FS and co-enumerability by Theorem 3. To constructively get a decider from this we need a form of Post's theorem. We use a formulation by [3] that additionally only requires discreteness of S and logical decidability of $\vdash$, which we get using the definition of FS and using completeness respectively. □

This result can also be shown by directly giving a decider: Given a sentence s we enumerate all provable sentences s' and check whether $s = s'$ or $\neg s = s'$ using discreteness. Correctness is easy to show and totality is by completeness. Note that this does not require enumerability of S. However, this approach is tedious to handle in Coq as it requires working with a total function $S \rightharpoonup \mathbb{B}$. To our knowledge it would again require enumerability of S to convert it to a function $S \to \mathbb{B}$ or a decider.

3 Weak representability

Definition 4 (Weak representability)**.** *Let* FS $= (S, \neg, \vdash)$ *be a formal system,* $X : \mathbb{T}, P : X \to \mathbb{P}$ *be a predicate,* $r : X \to S$*. We say* r *weakly represents* P *in* FS*, if*

$$\forall x.\, P\, x \;\longleftrightarrow\; \vdash r\, x.$$

If such an r *exists, we call* P *weakly representable in* FS*.*

From a computational perspective, r would be considered a many-one reduction from P to $\vdash$.

Lemma 5 (Special halting problem)**.** *The special halting problem for* θ*, that is*

$$H_0\, c := \theta_c(c)\!\downarrow,$$

is undecidable.

Proof. Let $f : \mathbb{N} \to \mathbb{B}$ be a function such that $\forall c.\, f\, c = \mathsf{T} \longleftrightarrow H_0\, c$. Choose

$$g : \mathbb{N} \rightharpoonup \mathbb{B}, g\, c := \begin{cases} 0 & \text{if } f\, c = \mathsf{F} \\ \text{undefined} & \text{if } f\, c = \mathsf{T} \end{cases}$$

and let c be the code of g. We have

$$f\, c = \mathsf{F} \;\longleftrightarrow\; g\, c = 0 \;\longleftrightarrow\; \theta_c(c)\!\downarrow \;\longleftrightarrow\; H_0\, c \;\longleftrightarrow\; f\, c = \mathsf{T}$$

Therefore, H_0 is undecidable. □

Theorem 6. *Let* FS $= (S, \neg, \vdash)$ *be a complete formal system that can weakly represent the special halting problem for* θ*. Then* H_0 *is decidable.*

Proof. By Theorem 4, $\vdash$ is decidable, and, because decidability transports across equivalences, also H. □

Corollary 6.1 (Weak Gödel's first incompleteness theorem)**.** *Any formal system that can weakly represent* H_0 *is incomplete.*

There are many formal systems that fulfill the requirements of Corollary 6.1, such as any reasonable deduction system for first-order logic with the axioms of Robinson's Q as well as its sound extensions.

4 Value-representability

Definition 5 (Value-representability)**.** *Let* FS $= (S, \neg, \vdash)$ *be a formal system,* $f : \mathbb{N} \rightharpoonup \mathbb{B}$*, and* $r : \mathbb{N} \to \mathbb{B} \to S$*. We say* r *value-represents* f*, if*

$$\forall xy.\, f\, x \!\downarrow\! y \;\longrightarrow\; (\vdash r\, x\, y) \land (\vdash \neg r\, x\, (!y))$$

If such an r *exists,* FS *value-represents* f*.*

Definition 6. *A formal system value-represents all computable functions, if*

$$\forall c.\, \Sigma r.\, r \text{ value-represents } \theta_c.$$

It would be possible to quantify over functions instead of codes in Definition 6. This would essentially form a variant of Church's thesis for formal systems with a computable mapping from functions f to their codes r. We believe that this would still be consistent, but it might force us to interpret functions intensionally and prevent us from assuming some common axioms, such as functional extensionality.

Note that the definition of weak representability requires a form of soundness: From the provability of a sentence in a formal system we need to deduce a truth in our meta-system. Value-representability does not require this. Instead, we only need to be able to correctly reason about programs that actually halt: We do not pose restrictions on a representation $\vdash r\,x\,y$ if $f\,x$ is undefined.

Most importantly, if a theory T value-represents a function f in first-order logic, all of its consistent extensions also value-represent f. Weak representability only preserves along sound extensions.

It is difficult to compare the notions of weak and value-representability. If we restrict ourselves to functions of the form θ_c and predicates $P(x) = \theta_c(x) \downarrow \mathsf{T}$, generally both notions are incomparable. In complete formal systems however, weak representability of such a predicate implies value-representability of the respective function.

Definition 7 (Consistent guessing). *A language $L \subseteq \mathbb{N}$ fulfills consistent guessing if*

$$\{(c,x) \mid \theta_c(x) \downarrow \mathsf{T}\} \subseteq L \quad \wedge \quad \{(c,x) \mid \theta_c(x) \downarrow \mathsf{F}\} \cap L = \emptyset,$$

or equivalently,

$$\forall cv.\, (\theta_c(x) \downarrow \mathsf{T} \longrightarrow (c,x) \in L) \quad \wedge \quad (\theta_c(x) \downarrow \mathsf{F} \longrightarrow (c,x) \notin L).$$

Note that this definition does not place any restrictions on tuples (c,x) such that $\theta_c(x)$ is undefined, or rather, such that c does not halt on input x.

Lemma 7 (Consistent guessing is undecidable). *Any language $L \subseteq \mathbb{N}$ that fulfills consistent guessing is undecidable.*

Proof. Let $f : \mathbb{N} \to \mathbb{N} \to \mathbb{B}$ be a function such that $\forall cx.\, f\,c\,x = \mathsf{T} \longleftrightarrow (c,x) \in L$. Choose

$$g : \mathbb{N} \to \mathbb{N}, g\,c := !f\,c\,c$$

and let c be the code of g. We have

$$f\,c\,c = \mathsf{T} \longleftrightarrow g\,c = \mathsf{F} \longleftrightarrow \theta_c(c) \downarrow \mathsf{F} \longrightarrow (c,c) \notin L \longleftrightarrow f\,c\,c = \mathsf{F}$$

and

$$f\,c\,c = \mathsf{F} \longleftrightarrow g\,c = \mathsf{T} \longleftrightarrow \theta_c(c) \downarrow \mathsf{T} \longrightarrow (c,c) \in L \longleftrightarrow f\,c\,c = \mathsf{T}.$$

Therefore, L is undecidable. □

In essence, Theorem 7 shows that the following sets are recursively inseparable:

$$\{(c,x) \mid \theta_c(x) \downarrow \mathsf{T}\} \qquad \{(c,x) \mid \theta_c(x) \downarrow \mathsf{F}\}$$

5 Main result

Theorem 8 (Consistent guessing is decidable)**.** *Let* FS $= (S, \neg, \vdash)$ *be a complete formal system that can value-represent all computable functions. Then there is a decidable language that fulfills consistent guessing.*

Proof. We write r_c for the value-representation of a code c. Let $h : \mathbb{N} \to \mathbb{N} \to \mathbb{B}$ be the following function:

$$h\,c\,x := \begin{cases} \mathsf{T} & \text{if } r_c\,x\,\mathsf{T} \text{ is provable} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

h is total by Theorem 4 and completeness. It suffices to show that $L := \{(c, x) \mid h\,c\,x = \mathsf{T}\}$ fulfills consistent guessing.

Let c, x be such that $\theta_c(x) \downarrow \mathsf{T}$. By value-representability we have $\vdash r_c\,x\,\mathsf{T}$, and therefore $h\,c\,x = \mathsf{T}$ and $(c, x) \in L$.

Let c, x be such that $\theta_c(x) \downarrow \mathsf{F}$. Similarly, we have $\vdash \neg r_c\,x\,\mathsf{T}$ and therefore $h\,c\,x = \mathsf{F}$ by consistency and $(c, x) \notin L$. □

Corollary 8.1. *There is no complete formal system that can value-represent all functions.*

Corollary 8.2 (Gödel's first incompleteness theorem)**.** *Any formal system that can value-represent all functions is incomplete.*

Theorem 9 (Explicit incompleteness)**.** *Let* FS $= (S, \neg, \vdash)$ *be a formal system that value-represents all computable functions. We write r_c for the value-representation of a function c. Consider the following program $f(c, x)$:*

```
1. enumerate all provable sentences $s$.
2.    if $s = r_c\,x\,\mathsf{T}$, accept.
3.    if $s = \neg r_c\,x\,\mathsf{T}$, reject.
4.    otherwise, continue searching
```

and the function g':

$$g'\,c := \begin{cases} \mathsf{F} & \textit{if } f(c, c) \downarrow \mathsf{T} \\ \mathsf{T} & \textit{if } f(c, c) \downarrow \mathsf{F} \\ \textit{undefined} & \textit{if } f(c, c) \uparrow \end{cases}$$

Note that g' is not immediately monotonic. By Lemma 1, let $g : \mathbb{N} \rightharpoonup \mathbb{B}$ be an equivalent partial and therefore monotonic function, and c its code.

Now, $r_c\,c\,\mathsf{T}$ is independent in FS*, that is $\nvdash r_c\,c\,\mathsf{T}$ and $\nvdash \neg r_c\,c\,\mathsf{T}$.*

Proof. Assume $\vdash r_c\,c\,\mathsf{T}$. It suffices to show $\theta_c(c) \downarrow \mathsf{F}$ by consistency and value-representability. We have $\theta_c(c) \downarrow \mathsf{F} \longleftrightarrow g\,c = \mathsf{F} \longleftrightarrow f(c, c) \downarrow \mathsf{T} \longleftrightarrow \vdash r_c\,c\,\mathsf{T}$.

Assume $\vdash \neg r_c\,c\,\mathsf{T}$. It suffices to show $\vdash r_c\,c\,\mathsf{T}$ by consistency. We have $\vdash r_c\,c\,\mathsf{T} \longleftarrow \theta_c(c) \downarrow \mathsf{T} \longleftrightarrow g\,c = \mathsf{T} \longleftrightarrow f(c, c) \downarrow \mathsf{F} \longleftrightarrow \vdash \neg r_c\,c\,\mathsf{T}$. □

Note that f would compute a language that fulfills consistent guessing if FS were complete and that g is the function constructed in the undecidability proof of consistent guessing.

As opposed to the Corollary 6.1, Corollary 8.1 and Theorem 9 can both not only establish incompleteness of a theory, but also of its consistent extensions, that is essential incompleteness.

We believe that enumerability of sentences is not required for showing Theorem 9.

Kleene [6] shows the same result by more abstractly using two recursively enumerable and recursively inseparable sets, which correspond exactly to the two sets in Definition 7, and exploiting the fact that any computable characteristic function separating both sets must diverge on an input, in this case g.

References

[1] Scott Aaronson. Rosser's theorem via turing machines. Shtetl-Optimized. URL:`https://scottaaronson.blog/?p=710` (version: 2021-11-25).

[2] Yannick Forster. Parametric church's thesis: Synthetic computability without choice. In *Logical Foundations of Computer Science: International Symposium, LFCS 2022, January 10-13, 2022*, 2022.

[3] Yannick Forster, Dominik Kirst, and Gert Smolka. On synthetic undecidability in coq, with an application to the entscheidungsproblem. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2019, page 38–51, New York, NY, USA, 2019. Association for Computing Machinery.

[4] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931.

[5] Dominik Kirst and Marc Hermes. Synthetic Undecidability and Incompleteness of First-Order Axiom Systems in Coq. In Liron Cohen and Cezary Kaliszyk, editors, *12th International Conference on Interactive Theorem Proving (ITP 2021)*, volume 193 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[6] Stephen Cole Kleene. *Mathematical Logic*. Dover Publications, 1967.

[7] Georg Kreisel. Mathematical logic. *Journal of Symbolic Logic*, 32(3):419–420, 1967.

[8] Barkley Rosser. Extensions of some theorems of gödel and church. *The Journal of Symbolic Logic*, 1(3):87–91, 1936.

[9] A.S. Troelstra, D. van Dalen, and L.D. Beklemishev. *Constructivism in Mathematics, Vol 1*. Constructivism in Mathematics. Elsevier Science, 1988.

[10] user21820 (`https://math.stackexchange.com/users/21820/user21820`). Computability viewpoint of godel/rosser's incompleteness theorem. Mathematics Stack Exchange. URL:`https://math.stackexchange.com/q/2486349` (version: 2017-12-31).

[11] Anatoly Vorobey. First incompleteness via computation: an explicit construction. Foundations of Mathematics mailing list. URL:`https://cs.nyu.edu/pipermail/fom/2021-September/022872.html` (version: 2021-11-25).