Undecidability of the Halting Problem

Alethfeld Proof System

Graph ID: graph-837d62-79a574, Version 38

Status: 18 nodes verified, 0 tainted

Lean 4: Fully verified (0 sorries, 0 axioms)

January 2026

Abstract

We prove that no total program can decide its own halting problem, using a diagonalization argument. The proof is structured in Lamport's hierarchical notation and has been verified by the Alethfeld adversarial proof system. **This proof has been fully formalized in Lean 4 with zero sorries and zero non-standard axioms.**

1 Axioms and Definitions

We work in an axiomatic model of computation $\mathcal{M}$ with the following components:

Definition 1 (DiagModel)**.** *A* diagonal model of computation *consists of:*

- *A type* Data *used for both programs and inputs*
- *A proposition* $\mathsf{halts}(p, x)$ *asserting that program* p *halts on input* x
- *A function* $\mathsf{eval}(p, x, h) : \mathsf{Bool}$ *returning the result of* p *on* x*, given a proof* h *that* p *halts on* x
- *A constructor* $\mathsf{if\_run\_else\_halt} : \mathsf{Data} \to \mathsf{Data}$

subject to the following axioms:

$$\mathsf{eval}(c, x, h) = \mathsf{true} \Rightarrow \neg\mathsf{halts}(\mathsf{if\_run\_else\_halt}(c), x) \tag{ireh\_runs\_of\_true}$$

$$\mathsf{eval}(c, x, h) = \mathsf{false} \Rightarrow \mathsf{halts}(\mathsf{if\_run\_else\_halt}(c), x) \tag{ireh\_halts\_of\_false}$$

Definition 2 (Total Program)**.** *A* total program *is a pair* $(\mathsf{prog}, \mathsf{htotal})$ *where* $\mathsf{prog} : \mathsf{Data}$ *and* $\mathsf{htotal} : \forall x.\, \mathsf{halts}(\mathsf{prog}, x)$*.*

We define $\mathsf{eval\_total}(p, x) := \mathsf{eval}(p.\mathsf{prog}, x, p.\mathsf{htotal}(x))$*.* `[Lean: L29--38]`

2 Main Result

Theorem 3 (Undecidability of the Halting Problem)**.** *For any total program* $\mathsf{candidate}$*, there exists a program* $\mathsf{spoiler}$ *such that:*

$$\big(\mathsf{eval\_total}(\mathsf{candidate}, \mathsf{spoiler}) = \mathsf{true} \wedge \neg\mathsf{halts}(\mathsf{spoiler}, \mathsf{spoiler})\big)$$

$$\vee\ \big(\mathsf{eval\_total}(\mathsf{candidate}, \mathsf{spoiler}) = \mathsf{false} \wedge \mathsf{halts}(\mathsf{spoiler}, \mathsf{spoiler})\big)$$

`[Lean: L58--83]`

Proof. ⟨1⟩**1.** Let candidate : TotalProgram be arbitrary. *[assumption]* [Lean: L67]

⟨1⟩**2.** Define spoiler := if_run_else_halt(candidate.prog). *[definition]* [Lean: L72]

⟨1⟩**3.** $\mathsf{eval\_total}(\mathsf{candidate}, \mathsf{spoiler}) = \mathsf{true} \vee \mathsf{eval\_total}(\mathsf{candidate}, \mathsf{spoiler}) = \mathsf{false}$. *[Boolean exhaustion]* [Lean: L75]

⟨1⟩**4.** If $\mathsf{eval\_total}(\mathsf{candidate}, \mathsf{spoiler}) = \mathsf{true}$, then $\neg\mathsf{halts}(\mathsf{spoiler}, \mathsf{spoiler})$. *[implication-intro from 2, 3]* [Lean: L80--83]

⟨2⟩**4.1.** Assume $\mathsf{eval\_total}(\mathsf{candidate}, \mathsf{spoiler}) = \mathsf{true}$. *[local assumption H_{true}]* [Lean: L80]

⟨2⟩**4.2.** $\mathsf{halts}(\mathsf{candidate.prog}, \mathsf{spoiler})$ holds. *[universal-elim from* candidate.htotal*]* [Lean: L32]

⟨2⟩**4.3.** $\mathsf{eval\_total}(\mathsf{candidate}, \mathsf{spoiler}) = \mathsf{eval}(\mathsf{candidate.prog}, \mathsf{spoiler}, \mathsf{candidate.htotal}(\mathsf{spoiler}))$. *[definition expansion]* [Lean: L35--38]

⟨2⟩**4.4.** By axiom IREH_RUNS_OF_TRUE: $\mathsf{eval}(\mathsf{candidate.prog}, \mathsf{spoiler}, h) = \mathsf{true} \Rightarrow \neg\mathsf{halts}(\mathsf{if\_run\_else\_halt}(\mathsf{candida}$ *[axiom application]* [Lean: L63--64]

⟨2⟩**4.5.** By definition of spoiler and H_{true}: $\neg\mathsf{halts}(\mathsf{spoiler}, \mathsf{spoiler})$. *[modus ponens from 4.1, 4.3, 4.4]* [Lean: L83]

⟨2⟩**4.6.** Discharge H_{true}. *[discharge]* [Lean: L82--83]

⟨1⟩**5.** If $\mathsf{eval\_total}(\mathsf{candidate}, \mathsf{spoiler}) = \mathsf{false}$, then $\mathsf{halts}(\mathsf{spoiler}, \mathsf{spoiler})$. *[implication-intro from 2, 3]* [Lean: L76--79]

⟨2⟩**5.1.** Assume $\mathsf{eval\_total}(\mathsf{candidate}, \mathsf{spoiler}) = \mathsf{false}$. *[local assumption H_{false}]* [Lean: L76]

⟨2⟩**5.2.** $\mathsf{halts}(\mathsf{candidate.prog}, \mathsf{spoiler})$ holds. *[universal-elim from* candidate.htotal*]* [Lean: L32]

⟨2⟩**5.3.** $\mathsf{eval\_total}(\mathsf{candidate}, \mathsf{spoiler}) = \mathsf{eval}(\mathsf{candidate.prog}, \mathsf{spoiler}, \mathsf{candidate.htotal}(\mathsf{spoiler}))$. *[definition expansion]* [Lean: L35--38]

⟨2⟩**5.4.** By axiom IREH_HALTS_OF_FALSE: $\mathsf{eval}(\mathsf{candidate.prog}, \mathsf{spoiler}, h) = \mathsf{false} \Rightarrow \mathsf{halts}(\mathsf{if\_run\_else\_halt}(\mathsf{candida}$ *[axiom application]* [Lean: L65--66]

⟨2⟩**5.5.** By definition of spoiler and H_{false}: $\mathsf{halts}(\mathsf{spoiler}, \mathsf{spoiler})$. *[modus ponens from 5.1, 5.3, 5.4]* [Lean: L79]

⟨2⟩**5.6.** Discharge H_{false}. *[discharge]* [Lean: L78--79]

⟨1⟩**6. QED:** $(\mathsf{eval\_total}(\mathsf{candidate}, \mathsf{spoiler}) = \mathsf{true} \wedge \neg\mathsf{halts}(\mathsf{spoiler}, \mathsf{spoiler})) \vee (\mathsf{eval\_total}(\mathsf{candidate}, \mathsf{spoiler}) = \mathsf{false} \wedge \mathsf{halts}(\mathsf{spoiler}, \mathsf{spoiler}))$. *[disjunction-intro from 3, 4, 5]* [Lean: L68--70]

□

3 Classical Formulation

Theorem 4 (Undecidability – Negation Form)**.** *There does not exist a total program* decider *such that for all programs p:*

$$\mathsf{eval\_total}(\mathsf{decider}, p) = \mathsf{true} \iff \mathsf{halts}(p, p)$$

[Lean: L93--114]

Proof. Suppose for contradiction that such a decider exists. By Theorem 3, there exists spoiler such that:

- **Case 1:** $\mathsf{eval\_total}(\mathsf{decider}, \mathsf{spoiler}) = \mathsf{true}$ and $\neg\mathsf{halts}(\mathsf{spoiler}, \mathsf{spoiler})$.

  But by the specification of decider, $\mathsf{eval\_total}(\mathsf{decider}, \mathsf{spoiler}) = \mathsf{true}$ implies $\mathsf{halts}(\mathsf{spoiler}, \mathsf{spoiler})$. Contradiction. [Lean: L111--112]

- **Case 2:** $\mathsf{eval\_total}(\mathsf{decider}, \mathsf{spoiler}) = \mathsf{false}$ and $\mathsf{halts}(\mathsf{spoiler}, \mathsf{spoiler})$.

  But by the specification of decider, $\mathsf{halts}(\mathsf{spoiler}, \mathsf{spoiler})$ implies $\mathsf{eval\_total}(\mathsf{decider}, \mathsf{spoiler}) = \mathsf{true}$. This contradicts $\mathsf{eval\_total}(\mathsf{decider}, \mathsf{spoiler}) = \mathsf{false}$. [Lean: L113--114]

In both cases we reach a contradiction, so no such decider exists. □

4 Lean 4 Formalization

The proof has been fully formalized in Lean 4 using Mathlib. The formalization is available at:

`lean/AlethfeldLean/Computability/HaltingUndecidability.lean`

Key Definitions

```
structure TotalProgram (Program : Type)
    (halts : Program → Program → Prop) where
  prog : Program
  htotal : Program →
  terminates :  input, halts prog input

def eval_total (eval : Program → Program →  → Bool)
    (candidate : TotalProgram Program halts)
    (input : Program) : Bool :=
  eval candidate.prog input (candidate.htotal input)
```

Main Theorem

```
theorem halting_undecidability
    (eval : Program → Program →  → Bool)
    (if_run_else_halt : Program → Program)
    (ireh_runs_of_true :  dec input h,
      eval dec input h = true →
        ¬halts (if_run_else_halt dec) input)
    (ireh_halts_of_false :  dec input h,
      eval dec input h = false →
        halts (if_run_else_halt dec) input)
    (candidate : TotalProgram Program halts) :
     spoiler,
      (eval_total eval candidate spoiler = true
        ¬halts spoiler spoiler)
      (eval_total eval candidate spoiler = false
        halts spoiler spoiler)
```

Verification Status

- **Sorries:** 0
- **Axioms used:** None (fully constructive)
- **Dependencies:** Mathlib.Tactic

Verification Status

Metric	**Value**
Graph ID	graph-837d62-79a574
Version	38
Total nodes	18
Verified	18
Admitted	0
Tainted	0
Obligations	0
Lean 4 Status	**Fully Verified**
Lean file	`HaltingUndecidability.lean`
Sorries	0
Non-standard axioms	0

Generated by Alethfeld Proof Orchestrator v5.1
Lean 4 formalization verified with `lake build`