Applied Synthetic Computability Theory

Gödel's Incompleteness and Post's Problem

Dominik Kirst

Logic and Computation Team Seminar
LIPN, February 27th, 2025

There are non-computable functions!

There are non-computable functions!

Standard example: the characteristic function of the halting problem

$$\chi_K(M) := \begin{cases} 1 & \text{if } M \text{ terminates} \\ 0 & \text{if } M \text{ diverges} \end{cases}$$

There are non-computable functions!

Standard example: the characteristic function of the halting problem

$$\chi_K(M) := \begin{cases} 1 & \text{if } M \text{ terminates} \\ 0 & \text{if } M \text{ diverges} \end{cases}$$

Three properties ensure that this relation is a function:

There are non-computable functions!

Standard example: the characteristic function of the halting problem

$$\chi_K(M) := \begin{cases} 1 & \text{if } M \text{ terminates} \\ 0 & \text{if } M \text{ diverges} \end{cases}$$

Three properties ensure that this relation is a function:

- Functionality: obvious

There are non-computable functions!

Standard example: the characteristic function of the halting problem

$$\chi_K(M) := \begin{cases} 1 & \text{if } M \text{ terminates} \\ 0 & \text{if } M \text{ diverges} \end{cases}$$

Three properties ensure that this relation is a function:

- Functionality: obvious
- Totality: given M, since M either terminates or diverges by the law of excluded middle, we have either $\chi_K(M) = 1$ or $\chi_K(M) = 0$, respectively

There are non-computable functions!

Standard example: the characteristic function of the halting problem

$$\chi_K(M) := \begin{cases} 1 & \text{if } M \text{ terminates} \\ 0 & \text{if } M \text{ diverges} \end{cases}$$

Three properties ensure that this relation is a function:

- Functionality: obvious
- Totality: given M, since M either terminates or diverges by the law of excluded middle, we have either $\chi_K(M) = 1$ or $\chi_K(M) = 0$, respectively
- Unique choice: total functional relations are actually functions

There are non-computable functions!

Standard example: the characteristic function of the halting problem

$$\chi_K(M) := \begin{cases} 1 & \text{if } M \text{ terminates} \\ 0 & \text{if } M \text{ diverges} \end{cases}$$

Three properties ensure that this relation is a function:

- Functionality: obvious
- Totality: given M, since M either terminates or diverges by the law of excluded middle, we have either $\chi_K(M) = 1$ or $\chi_K(M) = 0$, respectively
- Unique choice: total functional relations are actually functions

So classical logic is needed to show that $\chi_K(M)$ really is a function!

There are no non-computable functions!

There are no non-computable functions!

Non-computable functions are an artefact of classical logic!

There are no non-computable functions!

Non-computable functions are an artefact of classical logic!

In foundations without LEM like IZF, CZF, MLTT, HoTT, CIC:

- Proving totality is the similar to constructing a computable function
- Type-theoretic foundations actually identify the two

There are no non-computable functions!

Non-computable functions are an artefact of classical logic!

In foundations without LEM like IZF, CZF, MLTT, HoTT, CIC:

- Proving totality is the similar to constructing a computable function
- Type-theoretic foundations actually identify the two

In foundations without LEM and UC like CIC:

- Even if totality is given, it still does not induce a function
- So we can even assume LEM and still don't obtain new functions!

There are no non-computable functions!

Non-computable functions are an artefact of classical logic!

In foundations without LEM like IZF, CZF, MLTT, HoTT, CIC:

- Proving totality is the similar to constructing a computable function
- Type-theoretic foundations actually identify the two

In foundations without LEM and UC like CIC:

- Even if totality is given, it still does not induce a function
- So we can even assume LEM and still don't obtain new functions!

In any of those: no need for Turing machines, simply treat all functions as computable

Some Synthetic Definitions[1]

$P \subseteq X$ is decidable if there exists $d : X \to \mathbb{B}$ with $x \in P \leftrightarrow d\,x = \mathsf{true}$

[1] cf. Richman (1983), Bauer (2006), Forster, Kirst, Smolka (2019)

Some Synthetic Definitions[1]

$P \subseteq X$ is decidable if there exists $d : X \to \mathbb{B}$ with $x \in P \leftrightarrow d\,x = \mathsf{true}$

$P \subseteq X$ is semi-decidable if there exists $s : X \times \mathbb{N} \to \mathbb{B}$ with $x \in P \leftrightarrow \exists n.\, s\,(x, n) = \mathsf{true}$

[1] cf. Richman (1983), Bauer (2006), Forster, Kirst, Smolka (2019)

Some Synthetic Definitions[1]

$P \subseteq X$ is decidable if there exists $d : X \to \mathbb{B}$ with $x \in P \leftrightarrow d\,x = \text{true}$

$P \subseteq X$ is semi-decidable if there exists $s : X \times \mathbb{N} \to \mathbb{B}$ with $x \in P \leftrightarrow \exists n.\, s\,(x, n) = \text{true}$

$P \subseteq X$ is enumerable if there exists $e : \mathbb{N} \to X \,\dot{\cup}\, \{*\}$ with $x \in P \leftrightarrow \exists n.\, e\,n = x$

[1] cf. Richman (1983), Bauer (2006), Forster, Kirst, Smolka (2019)

Some Synthetic Definitions[1]

$P \subseteq X$ is decidable if there exists $d : X \to \mathbb{B}$ with $x \in P \leftrightarrow d\,x = \text{true}$

$P \subseteq X$ is semi-decidable if there exists $s : X \times \mathbb{N} \to \mathbb{B}$ with $x \in P \leftrightarrow \exists n.\, s\,(x, n) = \text{true}$

$P \subseteq X$ is enumerable if there exists $e : \mathbb{N} \to X \,\dot{\cup}\, \{*\}$ with $x \in P \leftrightarrow \exists n.\, e\,n = x$

$P \subseteq X$ reduces to $Q \subseteq Y$ if there exists $r : X \to Y$ with $x \in P \leftrightarrow r\,x \in Q$

[1] cf. Richman (1983), Bauer (2006), Forster, Kirst, Smolka (2019)

Some Synthetic Proofs

1. Decidable sets are semi-decidable and co-semi-decidable

Some Synthetic Proofs

1. Decidable sets are semi-decidable and co-semi-decidable

   $\Rightarrow$ Given $d : X \to \mathbb{B}$ pick $s(x, n) := d\,x$ and $s(x, n) := \neg d\,x$, respectively

Some Synthetic Proofs

1. Decidable sets are semi-decidable and co-semi-decidable

   $\Rightarrow$ Given $d : X \to \mathbb{B}$ pick $s(x, n) := d\,x$ and $s(x, n) := \neg d\,x$, respectively

2. A set (of numbers) is semi-decidable if and only if it is enumerable

Some Synthetic Proofs

1. Decidable sets are semi-decidable and co-semi-decidable

   $\Rightarrow$ Given $d : X \to \mathbb{B}$ pick $s(x,n) := d\,x$ and $s(x,n) := \neg d\,x$, respectively

2. A set (of numbers) is semi-decidable if and only if it is enumerable

   $\Rightarrow$ Given a semi-decider $s : \mathbb{N} \times \mathbb{N} \to \mathbb{B}$ pick the enumerator

$$e\,n := \begin{cases} n_2 & n = \langle n_1, n_2 \rangle \text{ and } s(n_1, n_2) = \text{true} \\ * & \text{otherwise} \end{cases}$$

   and given an enumerator $e : \mathbb{N} \to \mathbb{N} \cup \{*\}$ pick the semi-decider $s(x,n) := e\,n \stackrel{?}{=} x$

Some Synthetic Proofs

1. Decidable sets are semi-decidable and co-semi-decidable

   $\Rightarrow$ Given $d : X \to \mathbb{B}$ pick $s(x, n) := d\,x$ and $s(x, n) := \neg d\,x$, respectively

2. A set (of numbers) is semi-decidable if and only if it is enumerable

   $\Rightarrow$ Given a semi-decider $s : \mathbb{N} \times \mathbb{N} \to \mathbb{B}$ pick the enumerator

$$e\,n := \begin{cases} n_2 & n = \langle n_1, n_2 \rangle \text{ and } s(n_1, n_2) = \text{true} \\ * & \text{otherwise} \end{cases}$$

   and given an enumerator $e : \mathbb{N} \to \mathbb{N} \cup \{*\}$ pick the semi-decider $s(x, n) := e\,n \stackrel{?}{=} x$

3. If $P \subseteq X$ reduces to $Q \subseteq Y$ and Q is decidable, then so is P

Some Synthetic Proofs

1. Decidable sets are semi-decidable and co-semi-decidable

   $\Rightarrow$ Given $d : X \to \mathbb{B}$ pick $s(x, n) := d\,x$ and $s(x, n) := \neg d\,x$, respectively

2. A set (of numbers) is semi-decidable if and only if it is enumerable

   $\Rightarrow$ Given a semi-decider $s : \mathbb{N} \times \mathbb{N} \to \mathbb{B}$ pick the enumerator

$$e\,n := \begin{cases} n_2 & n = \langle n_1, n_2 \rangle \text{ and } s(n_1, n_2) = \text{true} \\ * & \text{otherwise} \end{cases}$$

   and given an enumerator $e : \mathbb{N} \to \mathbb{N} \cup \{*\}$ pick the semi-decider $s(x, n) := e\,n \stackrel{?}{=} x$

3. If $P \subseteq X$ reduces to $Q \subseteq Y$ and Q is decidable, then so is P

   $\Rightarrow$ Given $d : Y \to \mathbb{B}$ and $f : X \to Y$ pick $d \circ f$

Post's Theorem

Lemma (Post)

If $P \subseteq X$ and is bi-semi-decidable and definite $(\forall x.\, x \in P \vee x \notin P)$, then P is decidable.

Post's Theorem

Lemma (Post)

If $P \subseteq X$ *and is bi-semi-decidable and definite* $(\forall x.\, x \in P \lor x \notin P)$, *then* P *is decidable.*

Proof.

Post's Theorem

Lemma (Post)

If $P \subseteq X$ and is bi-semi-decidable and definite $(\forall x.\, x \in P \lor x \notin P)$, then P is decidable.

Proof.

1. Let s semi-decide P and s' semi-decide $\overline{P}$.

Post's Theorem

Lemma (Post)

If $P \subseteq X$ and is bi-semi-decidable and definite $(\forall x.\, x \in P \vee x \notin P)$, then P is decidable.

Proof.

1. Let s semi-decide P and s' semi-decide $\overline{P}$.
2. Define d by running both s and s', return true if $s(x) \downarrow$ true and false if $s'(x) \downarrow$ true.

Post's Theorem

Lemma (Post)

If $P \subseteq X$ and is bi-semi-decidable and definite $(\forall x.\, x \in P \lor x \notin P)$, then P is decidable.

Proof.

1. Let s semi-decide P and s' semi-decide $\overline{P}$.
2. Define d by running both s and s', return true if $s(x) \downarrow$ true and false if $s'(x) \downarrow$ true.
3. Since not both $x \in P$ and $x \notin P$, observe that d is well-defined.

Post's Theorem

Lemma (Post)

If $P \subseteq X$ and is bi-semi-decidable and definite $(\forall x.\, x \in P \lor x \notin P)$, then P is decidable.

Proof.

1. Let s semi-decide P and s' semi-decide $\overline{P}$.
2. Define d by running both s and s', return true if $s(x) \downarrow$ true and false if $s'(x) \downarrow$ true.
3. Since not both $x \in P$ and $x \notin P$, observe that d is well-defined.
4. Since either $x \in P$ or $x \notin P$ by definiteness, observe that d is total.

Post's Theorem

Lemma (Post)

If $P \subseteq X$ and is bi-semi-decidable and definite $(\forall x.\, x \in P \vee x \notin P)$, then P is decidable.

Proof.

1. Let s semi-decide P and s' semi-decide $\overline{P}$.
2. Define d by running both s and s', return true if $s(x) \downarrow$ true and false if $s'(x) \downarrow$ true.
3. Since not both $x \in P$ and $x \notin P$, observe that d is well-defined.
4. Since either $x \in P$ or $x \notin P$ by definiteness, observe that d is total.
5. Conclude that d decides P. □

Internalising Computability

So far everything we did is borderline meaningless as we kept our foundation neutral:

$$\mathsf{LEM} + \mathsf{UC} \;\Rightarrow\; \text{every set is decidable}$$

Internalising Computability

So far everything we did is borderline meaningless as we kept our foundation neutral:

$$\text{LEM} + \text{UC} \;\Rightarrow\; \text{every set is decidable}$$

Therefore we can't show any set undecidable without contradicting LEM + UC:

Internalising Computability

So far everything we did is borderline meaningless as we kept our foundation neutral:

$$\text{LEM} + \text{UC} \;\Rightarrow\; \text{every set is decidable}$$

Therefore we can't show any set undecidable without contradicting LEM + UC:

1. Assume that every function $\mathbb{N} \to \mathbb{N}$ is Turing-computable (CT)

Internalising Computability

So far everything we did is borderline meaningless as we kept our foundation neutral:

$$\text{LEM} + \text{UC} \;\Rightarrow\; \text{every set is decidable}$$

Therefore we can't show any set undecidable without contradicting LEM + UC:

1. Assume that every function $\mathbb{N} \to \mathbb{N}$ is Turing-computable (CT)
2. Observe that most diagonalisations just rely on an enumeration of computable functions

Internalising Computability

So far everything we did is borderline meaningless as we kept our foundation neutral:

$$\text{LEM} + \text{UC} \;\Rightarrow\; \text{every set is decidable}$$

Therefore we can't show any set undecidable without contradicting LEM + UC:

1. Assume that every function $\mathbb{N} \to \mathbb{N}$ is Turing-computable (CT)
2. Observe that most diagonalisations just rely on an enumeration of computable functions
3. Be careful not to assume an enumeration of the total function space $\mathbb{N} \to \mathbb{N}$

Internalising Computability

So far everything we did is borderline meaningless as we kept our foundation neutral:

$$\mathsf{LEM} + \mathsf{UC} \;\Rightarrow\; \text{every set is decidable}$$

Therefore we can't show any set undecidable without contradicting LEM + UC:

1. Assume that every function $\mathbb{N} \to \mathbb{N}$ is Turing-computable (CT)
2. Observe that most diagonalisations just rely on an enumeration of computable functions
3. Be careful not to assume an enumeration of the total function space $\mathbb{N} \to \mathbb{N}$
4. Assume an enumeration of the partial function space $\mathbb{N} \rightharpoonup \mathbb{N}$

EPF and the Halting Problem

Axiom (EPF, Richman (1983); Bauer (2006); Forster (2022))

There is a universal function $\Theta : \mathbb{N} \to (\mathbb{N} \rightharpoonup \mathbb{N})$ *enumerating all partial functions:*

$$\forall f : \mathbb{N} \rightharpoonup \mathbb{N}. \exists c : \mathbb{N}. \forall xb. \Theta_c\, x \downarrow b \leftrightarrow f\, x \downarrow b$$

EPF and the Halting Problem

Axiom (EPF, Richman (1983); Bauer (2006); Forster (2022))

There is a universal function $\Theta : \mathbb{N} \to (\mathbb{N} \rightharpoonup \mathbb{N})$ enumerating all partial functions:

$$\forall f : \mathbb{N} \rightharpoonup \mathbb{N}.\, \exists c : \mathbb{N}.\, \forall xb.\, \Theta_c\, x \downarrow b \leftrightarrow f\, x \downarrow b$$

Lemma

The self-halting problem $\mathsf{K} := \{c \in \mathbb{N} \mid \Theta_c\, c \downarrow\}$ is semi-decidable but undecidable.

EPF and the Halting Problem

Axiom (EPF, Richman (1983); Bauer (2006); Forster (2022))

There is a universal function $\Theta : \mathbb{N} \to (\mathbb{N} \rightharpoonup \mathbb{N})$ *enumerating all partial functions:*

$$\forall f : \mathbb{N} \rightharpoonup \mathbb{N}.\, \exists c : \mathbb{N}.\, \forall xb.\, \Theta_c\, x \downarrow b \leftrightarrow f\, x \downarrow b$$

Lemma

The self-halting problem $\mathsf{K} := \{c \in \mathbb{N} \mid \Theta_c\, c \downarrow\}$ *is semi-decidable but undecidable.*

Proof.

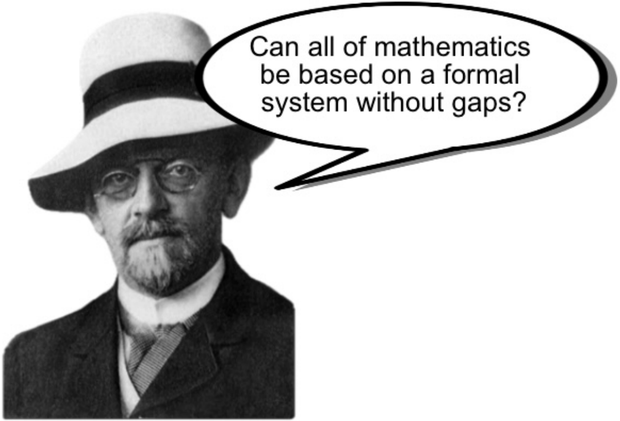
Assume $d : \mathbb{N} \to \mathbb{B}$ decides K. Consider the function $f : \mathbb{N} \rightharpoonup \mathbb{B}$ with $f\, c \uparrow$ if $d\, c = \mathsf{true}$ and $f\, c \downarrow \mathsf{true}$ otherwise. Let c be the code of f given by EPF, we derive a contradiction:

$$d\, c = \mathsf{true} \;\Leftrightarrow\; c \in K \;\Leftrightarrow\; \Theta_c\, c \downarrow \Leftrightarrow\; f\, c \downarrow \Leftrightarrow\; f\, c \downarrow \mathsf{true} \;\Leftrightarrow\; d\, c = \mathsf{false} \qquad \square$$

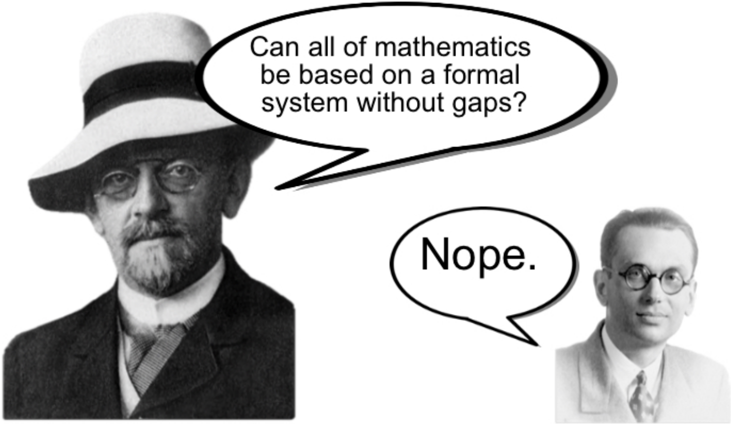
Application 1: Gödel's Incompleteness

(jww. Marc Hermes and Benjamin Peters)

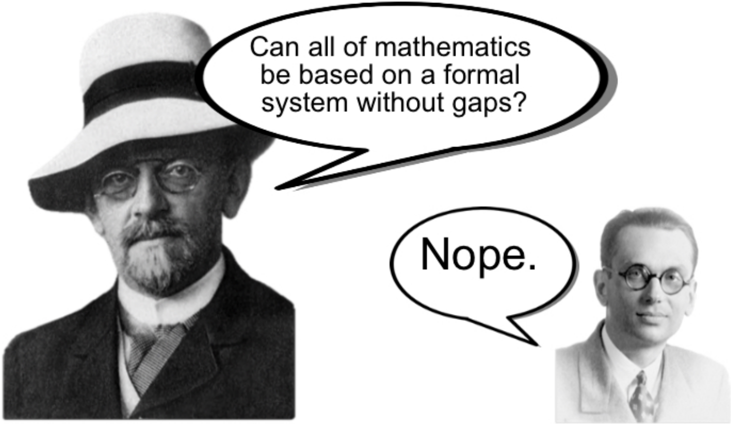
Historical Motivation



Historical Motivation



Historical Motivation



"Every sufficiently strong formal system admits independent sentences."

The First Incompleteness Theorem

Which formal systems $\mathcal{S}$ admit sentences φ with both $\mathcal{S} \nvdash \varphi$ and $\mathcal{S} \nvdash \neg\varphi$?

The First Incompleteness Theorem

Which formal systems $\mathcal{S}$ admit sentences φ with both $\mathcal{S} \nvdash \varphi$ and $\mathcal{S} \nvdash \neg\varphi$?

- Gödel: all sound, sufficiently expressive ones (Gödel, 1931)

The First Incompleteness Theorem

Which formal systems $\mathcal{S}$ admit sentences φ with both $\mathcal{S} \nvdash \varphi$ and $\mathcal{S} \nvdash \neg\varphi$?

- Gödel: all sound, sufficiently expressive ones (Gödel, 1931)
- Rosser: all consistent, sufficiently expressive ones (Rosser, 1936)

The First Incompleteness Theorem

Which formal systems $\mathcal{S}$ admit sentences φ with both $\mathcal{S} \nvdash \varphi$ and $\mathcal{S} \nvdash \neg\varphi$?

- Gödel: all sound, sufficiently expressive ones (Gödel, 1931)
- Rosser: all consistent, sufficiently expressive ones (Rosser, 1936)
- Turing(/Post): Gödel's incompleteness follows from undecidability (Turing, 1937)

The First Incompleteness Theorem

Which formal systems $\mathcal{S}$ admit sentences φ with both $\mathcal{S} \nvdash \varphi$ and $\mathcal{S} \nvdash \neg\varphi$?

- Gödel: all sound, sufficiently expressive ones (Gödel, 1931)
- Rosser: all consistent, sufficiently expressive ones (Rosser, 1936)
- Turing(/Post): Gödel's incompleteness follows from undecidability (Turing, 1937)
- Kleene: Rosser's incompleteness follows from recursive inseparability (Kleene, 1951)

Matrix of Incompleteness Theorems

	Disprove completeness	Independent sentence
Soundness	Turing	Gödel
ω-consistency		Gödel
Consistency		Rosser/Kleene

Motivational Testimonies

Motivational Testimonies

Computational proofs of Rosser's strength are not well-known but desirable:

Motivational Testimonies

Computational proofs of Rosser's strength are not well-known but desirable:

- *"Recently I was struck to discover just such a proof laid out..."*
  Anatoly Vorobey on the FOM mailing list

Motivational Testimonies

Computational proofs of Rosser's strength are not well-known but desirable:

- *"Recently I was struck to discover just such a proof laid out..."*
  Anatoly Vorobey on the FOM mailing list

- *"A few months ago, I found a short, simple, Turing-machine-based proof of Rosser's Theorem... So, will Gödel's Theorem always and forevermore be taught as a centerpiece of computability theory, and will the Gödel numbers get their much-deserved retirement?"*
  Scott Aaronson on his blog

Motivational Testimonies

Computational proofs of Rosser's strength are not well-known but desirable:

- *"Recently I was struck to discover just such a proof laid out..."*
  Anatoly Vorobey on the FOM mailing list
- *"A few months ago, I found a short, simple, Turing-machine-based proof of Rosser's Theorem... So, will Gödel's Theorem always and forevermore be taught as a centerpiece of computability theory, and will the Gödel numbers get their much-deserved retirement?"*
  Scott Aaronson on his blog
- *"Here I shall present very simple computability-based proofs of Gödel/Rosser's incompleteness theorem, which require only basic knowledge about programs. I feel that these proofs are little known despite giving a very general form of the incompleteness theorems, and also easy to make rigorous without even depending on much background knowledge in logic."*
  User21820 on StackExchange

Motivational Testimonies (ctd.)

Gödel's incompleteness theorems [edit]

The concepts raised by Gödel's incompleteness theorems are very similar to those raised by the halting problem, and the proofs are quite similar. In fact, a weaker form of the First Incompleteness Theorem is an easy consequence of the undecidability of the halting problem. This weaker form differs from the standard statement of the incompleteness theorem by asserting that an axiomatization of the natural numbers that is both complete and sound is impossible. The "sound" part is the weakening: it means that we require the axiomatic system in question to prove only *true* statements about natural numbers. Since soundness implies consistency, this weaker form can be seen as a corollary of the strong form. It is important to observe that the statement of the standard form of Gödel's First Incompleteness Theorem is completely unconcerned with the truth value of a statement, but only concerns the issue of whether it is possible to find it through a mathematical proof.

`https://en.wikipedia.org/wiki/Halting_problem`

Abstract Formal Systems

Abstract Formal Systems

Definition

A formal system $\mathcal{S} = (\mathbb{S}, \vdash, \neg)$ consists of:

- $\mathbb{S}$ is a set we consider the collection of formal sentences
- $\vdash$ is a semi-decidable subset we consider the provable sentences
- $\neg : \mathbb{S} \to \mathbb{S}$ is a computable function we consider negation, satisfying consistency:

$$\forall \varphi \in \mathbb{S}.\ \neg(\vdash \varphi \land \ \vdash \neg\varphi)$$

$\mathcal{S}$ is complete if for every $\varphi \in \mathbb{S}$ either $\vdash \varphi$ or $\vdash \neg\varphi$.

Abstract Formal Systems

Definition

A formal system $\mathcal{S} = (\mathbb{S}, \vdash, \neg)$ consists of:

- $\mathbb{S}$ is a set we consider the collection of formal sentences
- $\vdash$ is a semi-decidable subset we consider the provable sentences
- $\neg : \mathbb{S} \to \mathbb{S}$ is a computable function we consider negation, satisfying consistency:

$$\forall \varphi \in \mathbb{S}.\ \neg(\vdash \varphi \wedge \ \vdash \neg\varphi)$$

$\mathcal{S}$ is complete if for every $\varphi \in \mathbb{S}$ either $\vdash \varphi$ or $\vdash \neg\varphi$.

Lemma (Refutation)

In a complete formal system we have $\nvdash \varphi$ iff $\vdash \neg\varphi$.

Abstract Formal Systems

Definition

A formal system $\mathcal{S} = (\mathbb{S}, \vdash, \neg)$ consists of:

- $\mathbb{S}$ is a set we consider the collection of formal sentences
- $\vdash$ is a semi-decidable subset we consider the provable sentences
- $\neg : \mathbb{S} \to \mathbb{S}$ is a computable function we consider negation, satisfying consistency:

$$\forall \varphi \in \mathbb{S}.\ \neg(\vdash \varphi \land \ \vdash \neg\varphi)$$

$\mathcal{S}$ is complete if for every $\varphi \in \mathbb{S}$ either $\vdash \varphi$ or $\vdash \neg\varphi$.

Lemma (Refutation)

In a complete formal system we have $\nvdash \varphi$ iff $\vdash \neg\varphi$.

Proof.

That $\nvdash \varphi$ implies $\vdash \neg\varphi$ is by completeness, the other direction by consistency. $\square$

Gödel à la Turing

Gödel à la Turing

Theorem

Let P be some undecidable set, for instance the halting problem K*. If P reduces to the provable sentences $\vdash$ of a formal system $\mathcal{S} = (\mathbb{S}, \vdash, \neg)$, then $\mathcal{S}$ cannot be complete.*

Gödel à la Turing

Theorem

Let P be some undecidable set, for instance the halting problem K. If P reduces to the provable sentences $\vdash$ of a formal system $\mathcal{S} = (\mathbb{S}, \vdash, \neg)$, then $\mathcal{S}$ cannot be complete.

Proof.

Gödel à la Turing

Theorem

Let P be some undecidable set, for instance the halting problem K. If P reduces to the provable sentences $\vdash$ of a formal system $\mathcal{S} = (\mathbb{S}, \vdash, \neg)$, then $\mathcal{S}$ cannot be complete.

Proof.

1. Assume that P is undecidable and that $\mathcal{S}$ were complete.

Gödel à la Turing

Theorem

Let P be some undecidable set, for instance the halting problem K. *If P reduces to the provable sentences $\vdash$ of a formal system $\mathcal{S} = (\mathbb{S}, \vdash, \neg)$, then $\mathcal{S}$ cannot be complete.*

Proof.

1. Assume that P is undecidable and that $\mathcal{S}$ were complete.
2. Obtain that that $\nvdash \varphi$ iff $\vdash \neg\varphi$ by (Refutation).

Gödel à la Turing

Theorem

Let P be some undecidable set, for instance the halting problem K. If P reduces to the provable sentences $\vdash$ of a formal system $\mathcal{S} = (\mathbb{S}, \vdash, \neg)$, then $\mathcal{S}$ cannot be complete.

Proof.

1. Assume that P is undecidable and that $\mathcal{S}$ were complete.
2. Obtain that that $\nvdash \varphi$ iff $\vdash \neg\varphi$ by (Refutation).
3. Observe that the complement of $\vdash$ is semi-decidable.

Gödel à la Turing

Theorem

Let P be some undecidable set, for instance the halting problem K*. If P reduces to the provable sentences $\vdash$ of a formal system $\mathcal{S} = (\mathbb{S}, \vdash, \neg)$, then $\mathcal{S}$ cannot be complete.*

Proof.

1. Assume that P is undecidable and that $\mathcal{S}$ were complete.
2. Obtain that that $\nvdash \varphi$ iff $\vdash \neg\varphi$ by (Refutation).
3. Observe that the complement of $\vdash$ is semi-decidable.
4. Derive that $\vdash$ is decidable by (Post).

Gödel à la Turing

Theorem

Let P be some undecidable set, for instance the halting problem K. If P reduces to the provable sentences $\vdash$ of a formal system $\mathcal{S} = (\mathbb{S}, \vdash, \neg)$, then $\mathcal{S}$ cannot be complete.

Proof.

1. Assume that P is undecidable and that $\mathcal{S}$ were complete.
2. Obtain that that $\nvdash \varphi$ iff $\vdash \neg\varphi$ by (Refutation).
3. Observe that the complement of $\vdash$ is semi-decidable.
4. Derive that $\vdash$ is decidable by (Post).
5. Conclude that P must be decidable by (Reduction).

Gödel à la Turing

Theorem

Let P be some undecidable set, for instance the halting problem K*. If P reduces to the provable sentences $\vdash$ of a formal system $\mathcal{S} = (\mathbb{S}, \vdash, \neg)$, then $\mathcal{S}$ cannot be complete.*

Proof.

1. Assume that P is undecidable and that $\mathcal{S}$ were complete.
2. Obtain that that $\nvdash \varphi$ iff $\vdash \neg\varphi$ by (Refutation).
3. Observe that the complement of $\vdash$ is semi-decidable.
4. Derive that $\vdash$ is decidable by (Post).
5. Conclude that P must be decidable by (Reduction).
6. Contradiction. $\square$

Matrix of Incompleteness Theorems

	Disprove completeness	Independent sentence
Soundness	Turing ($\checkmark$)	Gödel
ω-consistency		Gödel
Consistency		Rosser/Kleene

Halting Problem (Refined)

Lemma

For every partial decider $d : \mathbb{N} \rightharpoonup \mathbb{B}$ *for* $\mathsf{K} = \{c \in \mathbb{N} \mid \Theta_c\, c \downarrow\}$ *with*

$$\forall x.\, x \in \mathsf{K} \;\leftrightarrow\; d\, x \downarrow \mathsf{true}$$

one can construct a concrete value c *such that* $d\, c$ *diverges.*

Halting Problem (Refined)

Lemma

For every partial decider $d : \mathbb{N} \rightharpoonup \mathbb{B}$ *for* $\mathsf{K} = \{c \in \mathbb{N} \mid \Theta_c\, c \downarrow\}$ *with*

$$\forall x.\, x \in \mathsf{K} \;\leftrightarrow\; d\, x \downarrow \mathsf{true}$$

one can construct a concrete value c *such that* $d\, c$ *diverges.*

Proof.

We first define a partial function $f : \mathbb{N} \rightharpoonup \mathbb{B}$ diagonalising against d by:

$$f\, x := \begin{cases} \mathsf{true} & \text{if } d\, x \downarrow \mathsf{false} \\ \uparrow & \text{otherwise} \end{cases}$$

Now using EPF we obtain a code c for f and deduce that $d\, c \uparrow$ by:

$$d\, c \downarrow \mathsf{true} \;\Leftrightarrow\; c \in \mathsf{K} \;\Leftrightarrow\; \Theta_c\, c \downarrow \Leftrightarrow\; f\, c \downarrow \Leftrightarrow\; f\, c \downarrow \mathsf{true} \;\Leftrightarrow\; d\, c \downarrow \mathsf{false} \qquad \square$$

Post's Theorem (Refined)

Theorem

Given disjoint semi-decidable sets $P, Q \subseteq X$, there is a partial decider $d : X \rightharpoonup \mathbb{B}$ with:

$$\forall x.\, (x \in P \leftrightarrow d\,x \downarrow \mathsf{true}) \land (x \in Q \leftrightarrow d\,x \downarrow \mathsf{false})$$

Post's Theorem (Refined)

Theorem

Given disjoint semi-decidable sets $P, Q \subseteq X$, there is a partial decider $d : X \rightharpoonup \mathbb{B}$ with:

$$\forall x.\, (x \in P \leftrightarrow d\,x \downarrow \mathsf{true}) \land (x \in Q \leftrightarrow d\,x \downarrow \mathsf{false})$$

Proof.

Given s_1 semi-deciding P and s_2 semi-deciding Q, define d by:

$$d\,x\,n := \begin{cases} \mathsf{true} & \text{if } s_1\,x\,n \\ \mathsf{false} & \text{if } s_2\,x\,n \\ \uparrow & \text{otherwise} \end{cases}$$

Then use disjointness to verify well-definedness and specification. $\square$

Partial Deciders of Formal Systems

Since formal systems have two canonical disjoint semi-decidable sets:

Lemma

For every formal system $\mathcal{S} = (\mathbb{S}, \neg, \vdash)$ there is a partial function $d_{\mathcal{S}} : \mathbb{S} \rightharpoonup \mathbb{B}$ with:

$$\forall \varphi.\, (\vdash \varphi \leftrightarrow d_{\mathcal{S}}\, \varphi \downarrow \mathsf{true}) \land (\vdash \neg\varphi \leftrightarrow d_{\mathcal{S}}\, \varphi \downarrow \mathsf{false})$$

Moreover, because of consistency we have $d_{\mathcal{S}}\, \varphi \uparrow$ exactly if φ is an independent sentence.

Partial Deciders of Formal Systems

Since formal systems have two canonical disjoint semi-decidable sets:

Lemma

For every formal system $\mathcal{S} = (\mathbb{S}, \neg, \vdash)$ *there is a partial function* $d_{\mathcal{S}} : \mathbb{S} \rightharpoonup \mathbb{B}$ *with:*

$$\forall \varphi.\, (\vdash \varphi \leftrightarrow d_{\mathcal{S}}\, \varphi \downarrow \mathsf{true}) \land (\vdash \neg\varphi \leftrightarrow d_{\mathcal{S}}\, \varphi \downarrow \mathsf{false})$$

Moreover, because of consistency we have $d_{\mathcal{S}}\, \varphi \uparrow$ *exactly if* φ *is an independent sentence.*

- If $\mathcal{S}$ is complete, then $d_{\mathcal{S}}$ induces a decider for representable problems

Partial Deciders of Formal Systems

Since formal systems have two canonical disjoint semi-decidable sets:

Lemma

For every formal system $\mathcal{S} = (\mathbb{S}, \neg, \vdash)$ *there is a partial function* $d_{\mathcal{S}} : \mathbb{S} \rightharpoonup \mathbb{B}$ *with:*

$$\forall \varphi. (\vdash \varphi \leftrightarrow d_{\mathcal{S}}\, \varphi \downarrow \mathsf{true}) \land (\vdash \neg\varphi \leftrightarrow d_{\mathcal{S}}\, \varphi \downarrow \mathsf{false})$$

Moreover, because of consistency we have $d_{\mathcal{S}}\, \varphi \uparrow$ *exactly if* φ *is an independent sentence.*

- If $\mathcal{S}$ is complete, then $d_{\mathcal{S}}$ induces a decider for representable problems
- Even without completeness, $d_{\mathcal{S}}$ is a partial decider for representable problems...

Gödel à la Turing (Refined)

Theorem

Every formal system $\mathcal{S}$ admitting a reduction from K *has an independent sentence.*

Gödel à la Turing (Refined)

Theorem

Every formal system $\mathcal{S}$ admitting a reduction from K *has an independent sentence.*

Proof.

If $r : \mathbb{N} \to \mathbb{S}$ reduces K to $\mathcal{S}$, then $d_{\mathcal{S}} \circ r$ is a candidate decider for K. Thus there is some code c with $d_{\mathcal{S}}\,(r\,c)\uparrow$, so $r\,c$ is must be independent. □

Gödel à la Turing (Refined)

Theorem

Every formal system $\mathcal{S}$ admitting a reduction from K has an independent sentence.

Proof.

If $r : \mathbb{N} \to \mathbb{S}$ reduces K to $\mathcal{S}$, then $d_{\mathcal{S}} \circ r$ is a candidate decider for K. Thus there is some code c with $d_{\mathcal{S}}\,(r\,c)\uparrow$, so $r\,c$ is must be independent. □

Implicitly, the formal system is assumed to be sound due to the reducibility property:

$$\vdash r(x) \;\to\; x \in \mathsf{K}$$

Matrix of Incompleteness Theorems

	Disprove completeness	Independent sentence
Soundness	Turing (✓)	Gödel (✓)
ω-consistency		Gödel
Consistency		Rosser/Kleene

Recursive Inseparability

To avoid soundness, we would like that $c \in \overline{\mathsf{K}}$ implies $\vdash \neg r\, c$...

Recursive Inseparability

To avoid soundness, we would like that $c \in \overline{\mathsf{K}}$ implies $\vdash \neg r\, c$...

- $\overline{\mathsf{K}}$ is not semi-decidable, so can't be recognised in a formal system

Recursive Inseparability

To avoid soundness, we would like that $c \in \overline{\mathsf{K}}$ implies $\vdash \neg r\, c$...

- $\overline{\mathsf{K}}$ is not semi-decidable, so can't be recognised in a formal system
- So we would like a semi-decidable subset of $\overline{\mathsf{K}}$

Recursive Inseparability

To avoid soundness, we would like that $c \in \overline{\mathsf{K}}$ implies $\vdash \neg r\, c$...

- $\overline{\mathsf{K}}$ is not semi-decidable, so can't be recognised in a formal system
- So we would like a semi-decidable subset of $\overline{\mathsf{K}}$
- Doesn't work for $\overline{\mathsf{K}}$ directly but there are other examples

Recursive Inseparability

To avoid soundness, we would like that $c \in \overline{\mathsf{K}}$ implies $\vdash \neg r\, c$...

- $\overline{\mathsf{K}}$ is not semi-decidable, so can't be recognised in a formal system
- So we would like a semi-decidable subset of $\overline{\mathsf{K}}$
- Doesn't work for $\overline{\mathsf{K}}$ directly but there are other examples
- Recursive Inseparability: disjoint sets P, Q that are not separable by $d : X \to \mathbb{B}$

$$\forall x.\, (x \in P \to d\, x = \mathsf{true}) \land (x \in Q \to d\, x = \mathsf{false})$$

Canonical Inseparable Sets

Lemma

The sets $\mathsf{K}^1 := \{c \in \mathbb{N} \mid \Theta_c\, c \downarrow \mathsf{true}\}$ *and* $\mathsf{K}^0 := \{c \in \mathbb{N} \mid \Theta_c\, c \downarrow \mathsf{false}\}$ *are semi-decidable but recursively inseparable, in fact for every partial separation* $d : \mathbb{N} \rightharpoonup \mathbb{B}$ *with*

$$x \in \mathsf{K}^1 \;\rightarrow\; d\, x \downarrow \mathsf{true} \qquad \textit{and} \qquad x \in \mathsf{K}^0 \;\rightarrow\; d\, x \downarrow \mathsf{false}$$

one can construct a concrete value c *such that* $d\, c$ *diverges.*

Canonical Inseparable Sets

Lemma

The sets $\mathsf{K}^1 := \{c \in \mathbb{N} \mid \Theta_c\, c \downarrow \mathsf{true}\}$ *and* $\mathsf{K}^0 := \{c \in \mathbb{N} \mid \Theta_c\, c \downarrow \mathsf{false}\}$ *are semi-decidable but recursively inseparable, in fact for every partial separation* $d : \mathbb{N} \rightharpoonup \mathbb{B}$ *with*

$$x \in \mathsf{K}^1 \;\to\; d\,x \downarrow \mathsf{true} \qquad \textit{and} \qquad x \in \mathsf{K}^0 \;\to\; d\,x \downarrow \mathsf{false}$$

one can construct a concrete value c *such that* $d\,c$ *diverges.*

Proof.

We first define a partial function $f : \mathbb{N} \rightharpoonup \mathbb{B}$ diagonalising against s by:

$$f\,x := \begin{cases} \mathsf{true} & \text{if } d\,x \downarrow \mathsf{false} \\ \mathsf{false} & \text{if } d\,x \downarrow \mathsf{true} \\ \uparrow & \text{otherwise} \end{cases}$$

Now using EPF we obtain a code c for f and deduce that $d\,c \uparrow$ by similar equivalences. □

Gödel à la Kleene

We say that a formal system $\mathcal{S}$ separates sets $P, Q \subseteq X$ if there is a function $r : X \to \mathbb{S}$ with

$$\forall x. (x \in P \to \vdash r\,x) \land (x \in Q \to \vdash \neg r\,x).$$

Gödel à la Kleene

We say that a formal system $\mathcal{S}$ separates sets $P, Q \subseteq X$ if there is a function $r : X \to \mathbb{S}$ with

$$\forall x.\, (x \in P \to \vdash r\, x) \wedge (x \in Q \to \vdash \neg r\, x).$$

Theorem

Every formal system $\mathcal{S}$ separating K^1 and K^0 has an independent sentence.

Gödel à la Kleene

We say that a formal system $\mathcal{S}$ separates sets $P, Q \subseteq X$ if there is a function $r : X \to \mathbb{S}$ with

$$\forall x.\, (x \in P \to \vdash r\,x) \land (x \in Q \to \vdash \neg r\,x).$$

Theorem

Every formal system $\mathcal{S}$ separating K^1 and K^0 has an independent sentence.

Proof.

If $r : \mathbb{N} \to \mathbb{S}$ separates K^1 and K^0, then $d_{\mathcal{S}} \circ r$ is a partial separation of K^1 and K^0. Thus there is some code c with $d_{\mathcal{S}}\,(r\,c)\uparrow$, so $r\,c$ is must be independent. $\square$

Gödel à la Kleene

We say that a formal system $\mathcal{S}$ separates sets $P, Q \subseteq X$ if there is a function $r : X \to \mathbb{S}$ with

$$\forall x.\, (x \in P \to \vdash r\,x) \land (x \in Q \to \vdash \neg r\,x).$$

Theorem

Every formal system $\mathcal{S}$ separating K^1 and K^0 has an independent sentence.

Proof.

If $r : \mathbb{N} \to \mathbb{S}$ separates K^1 and K^0, then $d_{\mathcal{S}} \circ r$ is a partial separation of K^1 and K^0. Thus there is some code c with $d_{\mathcal{S}}\,(r\,c) \uparrow$, so $r\,c$ is must be independent. □

Corollary

If $\mathcal{S}$ separates K^1 and K^0, then every extension $\mathcal{S}' \supseteq \mathcal{S}$ has an independent sentence.

Matrix of Incompleteness Theorems

	Disprove completeness	Independent sentence
Soundness	Turing (✓)	Gödel (✓)
ω-consistency		Gödel
Consistency		Rosser/Kleene (✓)

Essential Incompleteness of Robinson's Q

To instantiate these abstract proofs to Q, we need a stronger assumption than EPF:

Axiom ($\mathsf{CT_Q}$)

For every $f : \mathbb{N} \rightharpoonup \mathbb{B}$ there is a Σ_1-formula φ with: $f\,x \downarrow b \;\leftrightarrow\; \mathsf{Q} \vdash \forall b'.\, \varphi(\overline{x}, b') \leftrightarrow b' = \overline{b}$

Essential Incompleteness of Robinson's Q

To instantiate these abstract proofs to Q, we need a stronger assumption than EPF:

Axiom (CT_Q)

For every $f : \mathbb{N} \rightharpoonup \mathbb{B}$ there is a Σ_1-formula φ with: $f\,x \downarrow b \;\leftrightarrow\; \mathsf{Q} \vdash \forall b'.\, \varphi(\overline{x}, b') \leftrightarrow b' = \overline{b}$

CT_Q implies that Q and every consistent extension of it has an independent sentence:

Essential Incompleteness of Robinson's Q

To instantiate these abstract proofs to Q, we need a stronger assumption than EPF:

Axiom ($\mathsf{CT_Q}$)

For every $f : \mathbb{N} \rightharpoonup \mathbb{B}$ there is a Σ_1-formula φ with: $f\,x \downarrow b \;\leftrightarrow\; \mathsf{Q} \vdash \forall b'.\, \varphi(\overline{x}, b') \leftrightarrow b' = \overline{b}$

$\mathsf{CT_Q}$ implies that Q and every consistent extension of it has an independent sentence:

- $\mathsf{CT_Q}$ implies EPF

Essential Incompleteness of Robinson's Q

To instantiate these abstract proofs to Q, we need a stronger assumption than EPF:

Axiom ($\mathrm{CT_Q}$)

For every $f : \mathbb{N} \rightharpoonup \mathbb{B}$ there is a Σ_1-formula φ with: $f\,x \downarrow b \;\leftrightarrow\; \mathrm{Q} \vdash \forall b'.\, \varphi(\overline{x}, b') \leftrightarrow b' = \overline{b}$

$\mathrm{CT_Q}$ implies that Q and every consistent extension of it has an independent sentence:

- $\mathrm{CT_Q}$ implies EPF
- $\mathrm{CT_Q}$ implies that Q separates the problems K^1 and K^0

Essential Incompleteness of Robinson's Q

To instantiate these abstract proofs to Q, we need a stronger assumption than EPF:

Axiom (CT_Q)

For every $f : \mathbb{N} \rightharpoonup \mathbb{B}$ there is a Σ_1-formula φ with: $f\,x \downarrow b \;\leftrightarrow\; Q \vdash \forall b'.\, \varphi(\overline{x}, b') \leftrightarrow b' = \overline{b}$

CT_Q implies that Q and every consistent extension of it has an independent sentence:

- CT_Q implies EPF
- CT_Q implies that Q separates the problems K^1 and K^0
- Thus Q is essentially incomplete by instantiation of the abstract proof

Essential Incompleteness of Robinson's Q

To instantiate these abstract proofs to Q, we need a stronger assumption than EPF:

Axiom (CT_Q)

For every $f : \mathbb{N} \rightharpoonup \mathbb{B}$ *there is a* Σ_1*-formula* φ *with:* $f\,x \downarrow b \;\leftrightarrow\; \mathsf{Q} \vdash \forall b'.\, \varphi(\overline{x}, b') \leftrightarrow b' = \overline{b}$

CT_Q implies that Q and every consistent extension of it has an independent sentence:

- CT_Q implies EPF
- CT_Q implies that Q separates the problems K^1 and K^0
- Thus Q is essentially incomplete by instantiation of the abstract proof

- CT_Q also implies the diagonal lemma and therefore the Gödel/Rosser stragety

Essential Incompleteness of Robinson's Q

To instantiate these abstract proofs to Q, we need a stronger assumption than EPF:

Axiom ($\mathsf{CT_Q}$)

For every $f : \mathbb{N} \rightharpoonup \mathbb{B}$ there is a Σ_1-formula φ with: $f\,x \downarrow b \;\leftrightarrow\; \mathsf{Q} \vdash \forall b'.\, \varphi(\overline{x}, b') \leftrightarrow b' = \overline{b}$

$\mathsf{CT_Q}$ implies that Q and every consistent extension of it has an independent sentence:

- $\mathsf{CT_Q}$ implies EPF
- $\mathsf{CT_Q}$ implies that Q separates the problems K^1 and K^0
- Thus Q is essentially incomplete by instantiation of the abstract proof

- $\mathsf{CT_Q}$ also implies the diagonal lemma and therefore the Gödel/Rosser stragety
- Sanity check: $\mathsf{CT_Q}$ is equivalent to CT

Ongoing Work

Ongoing Work

Get more out of $\mathsf{CT_Q}$:

- Tarski's undefinability theorem
- Gödel's second incompleteness theorem
- Löb's theorem

Ongoing Work

Get more out of $\mathsf{CT_Q}$:

- Tarski's undefinability theorem
- Gödel's second incompleteness theorem
- Löb's theorem

Circumvent CT:

- Work against abstract notion of computable functions
- Instantiate trivially with CT for the constructively minded
- Instantiate with Turing-computability for the classically minded

Application 2: Post's Problem

(jww. Yannick Forster, Niklas Mück, Takako Nemoto, Haoyi Zeng)

Post's Problem

Is there a semi-decidable yet undecidable set S with $H \not\leq_T S$?

Post's Problem

Is there a semi-decidable yet undecidable set S with $H \not\preceq_T S$?

- Left as an open problem by Post (1944)
- Positive solution by Friedberg (1957) and Muchnik (1956)
- Low simple set construction by Lerman and Soare (1980)

Post's Problem

Is there a semi-decidable yet undecidable set S with $H \not\leq_T S$?

- Left as an open problem by Post (1944)
- Positive solution by Friedberg (1957) and Muchnik (1956)
- Low simple set construction by Lerman and Soare (1980)

- Synthetic proof mechanised in Coq by Zeng et al. (2024a), relying on Σ_2-LEM
- Analytic proof given by Nemoto (2024), relying only on Σ_1-LEM aka LPO
- Combination yields a synthetic and mechanised proof using LPO (Zeng et al., 2024b)

Synthetic Oracle Computability

Synthetic Oracle Computability

Definition (Forster, Kirst and Mück (2023))

An oracle computation is a functional $F\colon (Q \to A \to \mathbb{P}) \to I \to O \to \mathbb{P}$ captured by a computation tree $\tau\colon I \to A^* \rightharpoonup Q + O$ and its induced interrogation relation $\tau i ; R \vdash qs ; as$ as follows:

$$\frac{}{\sigma\,;\,R \vdash [\,]\,;\,[\,]} \qquad\qquad \frac{\sigma\,;\,R \vdash qs\,;\,as \qquad \sigma\, as \triangleright \mathsf{ask}\ q \qquad R q a}{\sigma\,;\,R \vdash qs +\!\!+ [q]\,;\,as +\!\!+ [a]}$$

$$F\,R\,i\,o \;\leftrightarrow\; \exists qs\, as.\ \tau i ; R \vdash qs ; as \land \tau\, i\, as \triangleright \mathsf{out}\ o$$

Synthetic Oracle Computability

Definition (Forster, Kirst and Mück (2023))

An oracle computation is a functional $F\colon (Q \to A \to \mathbb{P}) \to I \to O \to \mathbb{P}$ captured by a computation tree $\tau\colon I \to A^* \rightharpoonup Q + O$ and its induced interrogation relation $\tau i ; R \vdash qs ; as$ as follows:

$$\frac{}{\sigma\,;R \vdash [\,]\,;[\,]} \qquad \frac{\sigma\,;R \vdash qs\,;as \qquad \sigma\, as \triangleright \mathsf{ask}\ q \qquad R q a}{\sigma\,;R \vdash qs \mathbin{+\!\!+} [q]\,;as \mathbin{+\!\!+} [a]}$$

$$F\,R\,i\,o \;\leftrightarrow\; \exists qs\,as.\ \tau i ; R \vdash qs ; as \land \tau\ i\ as \triangleright \mathsf{out}\ o$$

$$P \preceq_T Q \;:=\; \text{there is an oracle computation } F\colon (\mathbb{N} \to \mathbb{B} \to \mathbb{P}) \to \mathbb{N} \to \mathbb{B} \to \mathbb{P} \text{ with } F\,Q = P$$

Synthetic Oracle Computability

Definition (Forster, Kirst and Mück (2023))

An oracle computation is a functional $F\colon(Q\to A\to\mathbb{P})\to I\to O\to\mathbb{P}$ captured by a computation tree $\tau\colon I\to A^*\rightharpoonup Q+O$ and its induced interrogation relation $\tau i\,;R\vdash qs\,;as$ as follows:

$$\frac{}{\sigma\,;R\vdash[\,]\,;[\,]}\qquad\qquad\frac{\sigma\,;R\vdash qs\,;as\qquad \sigma\,as\triangleright\mathrm{ask}\ q\qquad Rqa}{\sigma\,;R\vdash qs+\!\!+[q]\,;as+\!\!+[a]}$$

$$F\,R\,i\,o\ \leftrightarrow\ \exists qs\,as.\ \tau i\,;R\vdash qs\,;as\wedge\tau\ i\ as\triangleright\mathrm{out}\ o$$

$P\preceq_T Q\ :=$ there is an oracle computation $F\colon(\mathbb{N}\to\mathbb{B}\to\mathbb{P})\to\mathbb{N}\to\mathbb{B}\to\mathbb{P}$ with $F\,Q=P$

$\mathcal{S}_Q(P)\ :=$ there is an oracle computation $F\colon(\mathbb{N}\to\mathbb{B}\to\mathbb{P})\to\mathbb{N}\to\mathbb{1}\to\mathbb{P}$ with $\mathrm{dom}(F\,Q)=P$

Continuity of Oracle Computations

Continuity of Oracle Computations

Our employed notion of sequential continuity is strictly stronger than modulus-continuity:

Lemma (Forster, Kirst and Mück (2023))

1. *Every oracle computation F is modulus-continuous:*

$$F\,R\,i\,o \;\rightarrow\; \exists qs \subseteq \mathrm{dom}(R).\,\forall R'.\,(\forall q \in qs.\,\forall a.\; Rqa \leftrightarrow R'qa) \rightarrow F\,R'\,i\,o$$

2. *Not every modulus-continuous functional is an oracle computation.*

Continuity of Oracle Computations

Our employed notion of sequential continuity is strictly stronger than modulus-continuity:

Lemma (Forster, Kirst and Mück (2023))

1. *Every oracle computation F is modulus-continuous:*

$$F\,R\,i\,o \;\rightarrow\; \exists qs \subseteq \mathrm{dom}(R).\, \forall R'.\, (\forall q \in qs.\, \forall a.\; Rqa \leftrightarrow R'qa) \rightarrow F\,R'\,i\,o$$

2. *Not every modulus-continuous functional is an oracle computation.*

Proof.

1. From a terminating run $F\,R\,i\,o$ we obtain an interrogation $\tau\,i; R \vdash qs; as$ and can easily show that qs is a modulus of continuity.
2. The modulus-continuous functional $F\,R\,i\,o \;:=\; \exists q.\, R\,q\,\text{true}$ is not an oracle computation as for any computation tree τ we can define a suitably blocking oracle. □

Enumerating Oracle Computations

We need an enumeration of oracle computations for diagonalisations / Turing jump...

Enumerating Oracle Computations

We need an enumeration of oracle computations for diagonalisations / Turing jump...

For consistency (with LEM), we want to stick to EPF as only assumption:

Enumerating Oracle Computations

We need an enumeration of oracle computations for diagonalisations / Turing jump...

For consistency (with LEM), we want to stick to EPF as only assumption:

- EPF in particular enumerates trees $\tau : \mathbb{N} \to \mathbb{B}^* \rightharpoonup \mathbb{N} + \mathbb{B}$

Enumerating Oracle Computations

We need an enumeration of oracle computations for diagonalisations / Turing jump...

For consistency (with LEM), we want to stick to EPF as only assumption:

- EPF in particular enumerates trees $\tau : \mathbb{N} \to \mathbb{B}^* \rightharpoonup \mathbb{N} + \mathbb{B}$
- Due to the defining property of sequential continuity this is enough

Enumerating Oracle Computations

We need an enumeration of oracle computations for diagonalisations / Turing jump...

For consistency (with LEM), we want to stick to EPF as only assumption:

- EPF in particular enumerates trees $\tau : \mathbb{N} \to \mathbb{B}^* \rightharpoonup \mathbb{N} + \mathbb{B}$
- Due to the defining property of sequential continuity this is enough

Theorem (Forster, Kirst and Mück (2024))

There is an enumerator of functionals $\Phi : \mathbb{N} \to (\mathbb{N} \to \mathbb{B} \to \mathbb{P}) \to \mathbb{N} \to \mathbb{B} \to \mathbb{P}$ *such that*

1. Φ_e *is an oracle computation for all* $e : \mathbb{N}$
2. *Given an oracle computation* F *there is* $e : \mathbb{N}$ *such that* $\forall R x b.\, \Phi_e^R(x) \downarrow b \leftrightarrow F\, R\, x\, b$

Enumerating Oracle Computations

We need an enumeration of oracle computations for diagonalisations / Turing jump...

For consistency (with LEM), we want to stick to EPF as only assumption:

- EPF in particular enumerates trees $\tau : \mathbb{N} \to \mathbb{B}^* \rightharpoonup \mathbb{N} + \mathbb{B}$
- Due to the defining property of sequential continuity this is enough

Theorem (Forster, Kirst and Mück (2024))

There is an enumerator of functionals $\Phi : \mathbb{N} \to (\mathbb{N} \to \mathbb{B} \to \mathbb{P}) \to \mathbb{N} \to \mathbb{B} \to \mathbb{P}$ *such that*

1. Φ_e *is an oracle computation for all* $e : \mathbb{N}$
2. *Given an oracle computation* F *there is* $e : \mathbb{N}$ *such that* $\forall Rxb.\, \Phi_e^R(x) \downarrow b \leftrightarrow F\, R\, x\, b$
3. *The Turing jump* $P'\, x := \Phi_x^P(x) \downarrow \mathrm{true}$ *of* P *is strictly harder than* P

Low Simple Sets and Limit Computability

Low Simple Sets and Limit Computability

Definition (Lerman and Soare (1980) and Post (1944))

$P : X \to \mathbb{P}$ is low if $P' \preceq_T H$ and simple if it is co-infinite, semi-decidable, and for W_e being the e-th enumerable set we have $W_e \cap P \neq \emptyset$ whenever W_e is infinite.

Low Simple Sets and Limit Computability

Definition (Lerman and Soare (1980) and Post (1944))

$P : X \to \mathbb{P}$ is low if $P' \preceq_T H$ and simple if it is co-infinite, semi-decidable, and for W_e being the e-th enumerable set we have $W_e \cap P \neq \emptyset$ whenever W_e is infinite.

$\Rightarrow$ Every low simple set is a solution to Post's problem!

Low Simple Sets and Limit Computability

Definition (Lerman and Soare (1980) and Post (1944))

$P : X \to \mathbb{P}$ is low if $P' \preceq_T H$ and simple if it is co-infinite, semi-decidable, and for W_e being the e-th enumerable set we have $W_e \cap P \neq \emptyset$ whenever W_e is infinite.

$\Rightarrow$ Every low simple set is a solution to Post's problem!

Definition (Shoenfield (1959) and Gold (1965))

$P : X \to \mathbb{P}$ is limit-computable if there exists a function $f : X \to \mathbb{N} \to \mathbb{B}$ with

$$x \in P \;\leftrightarrow\; \exists n.\forall m > n.\ f(x, m) = \text{true} \quad \wedge \quad x \notin P \;\leftrightarrow\; \exists n.\forall m > n.\ f(x, m) = \text{false}.$$

Low Simple Sets and Limit Computability

Definition (Lerman and Soare (1980) and Post (1944))

$P : X \to \mathbb{P}$ is low if $P' \preceq_T H$ and simple if it is co-infinite, semi-decidable, and for W_e being the e-th enumerable set we have $W_e \cap P \neq \emptyset$ whenever W_e is infinite.

$\Rightarrow$ Every low simple set is a solution to Post's problem!

Definition (Shoenfield (1959) and Gold (1965))

$P : X \to \mathbb{P}$ is limit-computable if there exists a function $f : X \to \mathbb{N} \to \mathbb{B}$ with

$$x \in P \;\leftrightarrow\; \exists n.\forall m > n.\ f(x, m) = \text{true} \quad \wedge \quad x \notin P \;\leftrightarrow\; \exists n.\forall m > n.\ f(x, m) = \text{false}.$$

$\Rightarrow$ Limit-computability provides easy way to prove lowness...

Limit Lemma

Limit Lemma

Lemma (1)

If $\mathcal{S}_Q(P)$ *and* $\mathcal{S}_Q(\overline{P})$ *then* $P \preceq_T Q$.

Limit Lemma

Lemma (1)

If $\mathcal{S}_Q(P)$ *and* $\mathcal{S}_Q(\overline{P})$ *then* $P \preceq_T Q$.

Lemma (2)

Assuming Σ_n*-LEM, if* P *is* Σ_{n+1} *and* Q *is* Σ_n*, then* $\mathcal{S}_Q(P)$.

Limit Lemma

Lemma (1)

If $\mathcal{S}_Q(P)$ and $\mathcal{S}_Q(\overline{P})$ then $P \preceq_T Q$.

Lemma (2)

Assuming Σ_n-LEM, if P is Σ_{n+1} and Q is Σ_n, then $\mathcal{S}_Q(P)$.

Lemma (Limit Lemma)

Assuming LPO, if P is limit computable, then $P \preceq_T H$.

Limit Lemma

Lemma (1)

If $\mathcal{S}_Q(P)$ and $\mathcal{S}_Q(\overline{P})$ then $P \preceq_T Q$.

Lemma (2)

Assuming Σ_n-LEM, if P is Σ_{n+1} and Q is Σ_n, then $\mathcal{S}_Q(P)$.

Lemma (Limit Lemma)

Assuming LPO, if P is limit computable, then $P \preceq_T H$.

Proof.

If P is limit computable, then immediately by definition both P and $\overline{P}$ are Σ_2. Moreover, since the halting problem H is Σ_1, Lemma 2 together with LPO yields both $\mathcal{S}_H(P)$ and $\mathcal{S}_H(\overline{P})$. From there we conclude $P \preceq_T H$ with Lemma 1. □

The Priority Method

The Priority Method

Fix step function $\gamma : \mathbb{N}^* \to \mathbb{N} \to \mathbb{N} \to \mathbb{P}$, approximate S inductively:

$$\frac{}{0 \leadsto [\,]} \qquad \frac{n \leadsto L \quad \gamma_n^L\ x}{n+1 \leadsto x :: L} \qquad \frac{n \leadsto L \quad \forall x.\ \neg\ \gamma_n^L\ x}{n+1 \leadsto L}$$

The Priority Method

Fix step function $\gamma : \mathbb{N}^* \to \mathbb{N} \to \mathbb{N} \to \mathbb{P}$, approximate S inductively:

$$\frac{}{0 \rightsquigarrow [\,]} \qquad \frac{n \rightsquigarrow L \quad \gamma_n^L\, x}{n+1 \rightsquigarrow x :: L} \qquad \frac{n \rightsquigarrow L \quad \forall x.\, \neg\, \gamma_n^L\, x}{n+1 \rightsquigarrow L}$$

Depending on properties of γ we obtain for $S\,x := \exists n, L.\, n \rightsquigarrow L \land x \in L$ that:

The Priority Method

Fix step function $\gamma : \mathbb{N}^* \to \mathbb{N} \to \mathbb{N} \to \mathbb{P}$, approximate S inductively:

$$\frac{}{0 \rightsquigarrow [\,]} \qquad \frac{n \rightsquigarrow L \quad \gamma_n^L\, x}{n+1 \rightsquigarrow x :: L} \qquad \frac{n \rightsquigarrow L \quad \forall x.\, \neg\, \gamma_n^L\, x}{n+1 \rightsquigarrow L}$$

Depending on properties of γ we obtain for $S\,x := \exists n, L.\, n \rightsquigarrow L \land x \in L$ that:

- γ is computable $\;\Rightarrow\;$ S is semi-decidable

The Priority Method

Fix step function $\gamma : \mathbb{N}^* \to \mathbb{N} \to \mathbb{N} \to \mathbb{P}$, approximate S inductively:

$$\frac{}{0 \rightsquigarrow [\,]} \qquad \frac{n \rightsquigarrow L \quad \gamma_n^L\ x}{n+1 \rightsquigarrow x :: L} \qquad \frac{n \rightsquigarrow L \quad \forall x.\ \neg\ \gamma_n^L\ x}{n+1 \rightsquigarrow L}$$

Depending on properties of γ we obtain for $S\,x := \exists n, L.\ n \rightsquigarrow L \land x \in L$ that:

- γ is computable $\Rightarrow$ S is semi-decidable
- S satisfies $P_e := W_e$ is infinite $\to W_e \cap S \neq \emptyset$ $\Rightarrow$ S is simple

The Priority Method

Fix step function $\gamma : \mathbb{N}^* \to \mathbb{N} \to \mathbb{N} \to \mathbb{P}$, approximate S inductively:

$$\frac{}{0 \rightsquigarrow [\,]} \qquad \frac{n \rightsquigarrow L \quad \gamma_n^L\ x}{n+1 \rightsquigarrow x :: L} \qquad \frac{n \rightsquigarrow L \quad \forall x.\ \neg\ \gamma_n^L\ x}{n+1 \rightsquigarrow L}$$

Depending on properties of γ we obtain for $S\,x := \exists n, L.\ n \rightsquigarrow L \wedge x \in L$ that:

- γ is computable $\Rightarrow$ S is semi-decidable
- S satisfies $P_e :=$ W_e is infinite $\to W_e \cap S \neq \emptyset$ $\Rightarrow$ S is simple
- S satisfies $N_e := (\exists^\infty n.\ \Phi_e^S(e)[n]\downarrow) \to \Phi_e^S(e)\downarrow$ $\Rightarrow$ S' is limit computable (using LPO)

Wall Functions

Wall Functions

Definition

The use function $U_e^P(x)$ approximates the continuity information of the oracle computation $\Phi_e^P(x)$ for (semi-)decidable oracles P in a step-indexed way.

Wall Functions

Definition

The use function $U_e^P(x)$ approximates the continuity information of the oracle computation $\Phi_e^P(x)$ for (semi-)decidable oracles P in a step-indexed way.

Define suitable γ again relative to a wall function ω of same type:

- $\omega_n^L(e) \geq 2 \cdot e \;\Rightarrow\; S$ satisfies the requirements P_e
- $\omega_n^L(e) \geq \max_{e' \leq e} U_{e'}^L(e')[n] \;\Rightarrow\; S$ satisfies the requirements N_e (using LPO)

Wall Functions

Definition

The use function $U_e^P(x)$ approximates the continuity information of the oracle computation $\Phi_e^P(x)$ for (semi-)decidable oracles P in a step-indexed way.

Define suitable γ again relative to a wall function ω of same type:

- $\omega_n^L(e) \geq 2 \cdot e \;\Rightarrow\; S$ satisfies the requirements P_e
- $\omega_n^L(e) \geq \max_{e' \leq e} U_{e'}^L(e')[n] \;\Rightarrow\; S$ satisfies the requirements N_e (using LPO)

Theorem

Assuming LPO, a low simple set exists.

Wall Functions

Definition

The use function $U_e^P(x)$ approximates the continuity information of the oracle computation $\Phi_e^P(x)$ for (semi-)decidable oracles P in a step-indexed way.

Define suitable γ again relative to a wall function ω of same type:

- $\omega_n^L(e) \geq 2 \cdot e \;\Rightarrow\;$ S satisfies the requirements P_e
- $\omega_n^L(e) \geq \max_{e' \leq e} U_{e'}^L(e')[n] \;\Rightarrow\;$ S satisfies the requirements N_e (using LPO)

Theorem

Assuming LPO, a low simple set exists.

Proof.

Choose the wall function $\omega := \max(2 \cdot e, \max_{e' \leq e} U_{e'}^L(e')[n])$. □

Ongoing Work

Ongoing Work

Reverse analysis:

- LPO needed for limit lemma?
- LPO needed to show that S' is limit computable?
- LPO needed to construct a low simple set?

Ongoing Work

Reverse analysis:

- LPO needed for limit lemma?
- LPO needed to show that S' is limit computable?
- LPO needed to construct a low simple set?

Generalisation:

- Friedberg-Muchnik theorem
- Low basis theorem
- Connections to true second-order arithmetic

Conclusion

Synthetic Computability Propaganda

Reviewer 1: “Clearly, synthetic computability is trivializing things that should have been trivial from the beginning.”

1. Guides towards the computational essence of proofs
2. Allows concise but precise formalisation
3. Makes mechanisation feasible

What else could we discuss?

- Constructive status of completeness theorems
  - Model existence is equivalent to WLEM
  - Quasi-completeness is equivalent to WLEMS
- Constructive status of Löwenheim-Skolem theorems
  - Downwards part needs "DC-CC" and "LEM-MP"
  - Upwards part is usually proved via compactness
- Realisability models of constructive type theory and HOL
  - Type theories with choice sequences to separate formulations of MP
  - Effectful realisability interpretation of HOL to unify realisability variants
- Completeness theorems for bi-intuitionistic logic
  - Semantics in (constant-domain) Kripke models and (complete) bi-Heyting algebras

Bibliography I

Bauer, A. (2006). First steps in synthetic computability theory. *Electronic Notes in Theoretical Computer Science*, 155:5–31.

Forster, Y. (2022). Parametric Church's thesis: Synthetic computability without choice. In *International Symposium on Logical Foundations of Computer Science*, pages 70–89. Springer.

Forster, Y., Kirst, D., and Mück, N. (2023). Oracle computability and turing reducibility in the calculus of inductive constructions. In *Asian Symposium on Programming Languages and Systems*. Springer.

Forster, Y., Kirst, D., and Mück, N. (2024). The kleene-post and post's theorem in the calculus of inductive constructions. In *32nd EACSL Annual Conference on Computer Science Logic (CSL 2024)*.

Forster, Y., Kirst, D., and Smolka, G. (2019). 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*.

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

Gödel, K. (1931). Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für mathematik und physik*, 38(1):173–198.

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

Bibliography II

Kleene, S. C. (1951). A symmetric form of Gödel's theorem. *Journal of Symbolic Logic*, 16(2).

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

Muchnik, A. A. (1956). On the unsolvability of the problem of reducibility in the theory of algorithms. In *Dokl. Akad. Nauk SSSR*, volume 108, page 1.

Nemoto, T. (2024). Computability theory over intuitionistic logic. Logic Colloquium 2024, European Summer Meeting of the Association for Symbolic Logic, Gothenburg, Sweden.

Post, E. L. (1944). Recursively enumerable sets of positive integers and their decision problems. *bulletin of the American Mathematical Society*, 50(5):284–316.

Richman, F. (1983). Church's thesis without tears. *The Journal of symbolic logic*, 48(3):797–803.

Rosser, B. (1936). Extensions of some theorems of Gödel and Church. *The journal of symbolic logic*, 1(3):87–91.

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

Turing, A. M. (1937). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London mathematical society*, 2(1):230–265.

Zeng, H., Forster, Y., and Kirst, D. (2024a). Post's problem and the priority method in cic. In *30th International Conference on Types for Proofs and Programs TYPES 2024–Abstracts*, page 27.

Zeng, H., Forster, Y., Kirst, D., and Nemoto, T. (2024b). Post's problem in constructive mathematics. In *Continuity, Computability, Constructivity – From Logic to Algorithms*.