Undecidability of the halting problem and Gödel's incompleteness theorems

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Tampere University, Finland

January 11, 2022

What is Gödel's incompleteness theorem about?

Theorem (Gödel's first incompleteness theorem, informal version)

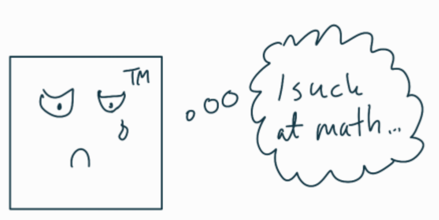
If T is a consistent set of axioms whose theorems can be enumerated by a Turing machine, then there are true statements about natural numbers that can not be deduced from T.

Reijo Jaakkola
reijo.jaakkola@tuni.fi

What is Gödel's incompleteness theorem about?

Theorem (Gödel's first incompleteness theorem, informal version)

If T is a consistent set of axioms whose theorems can be enumerated by a Turing machine, then there are true statements about natural numbers that can not be deduced from T.



Historical background

Kurt Gödel's achievement in modern logic is singular and monumental – indeed it is more than a monument, it is a landmark which will remain visible far in space and time.

John von Neumann

Historical background

Kurt Gödel's achievement in modern logic is singular and monumental – indeed it is more than a monument, it is a landmark which will remain visible far in space and time.

John von Neumann

1. 1900: Hilbert's second problem: Prove that arithmetic is consistent by using purely finitistic means.

Historical background

Kurt Gödel's achievement in modern logic is singular and monumental – indeed it is more than a monument, it is a landmark which will remain visible far in space and time.

John von Neumann

1. 1900: Hilbert's second problem: Prove that arithmetic is consistent by using purely finitistic means.
2. 1930: Gödel proves his incompleteness theorems without having a formal definition of computable function.

Historical background

> Kurt Gödel's achievement in modern logic is singular and monumental – indeed it is more than a monument, it is a landmark which will remain visible far in space and time.
>
> *John von Neumann*

1. 1900: Hilbert's second problem: Prove that arithmetic is consistent by using purely finitistic means.
2. 1930: Gödel proves his incompleteness theorems without having a formal definition of computable function.
3. 1934-1937: Gödel, Church and Turing each introduced models of computation (recursive functions, lambda calculus, Turing machines).

How to prove Gödel's first incompleteness theorem?

D.Hilbert asked if the formal arithmetic (PA: consisting of logic and algebraic axioms and an infinite family of Induction Axioms) can be consistently extended to a complete theory. The question was somewhat vague since an obvious answer was "yes": just add to PA axioms (assumed consistent) a maximal consistent set, clearly existing albeit hard to find. K.Goedel formalized this question as existence among such extensions of recursively enumerable ones and gave it a negative answer. **Its mathematical essence is the absence of total recursive extensions of universal partial recursive predicate.**

Leonid Levin

How to prove Gödel's first incompleteness theorem?

D.Hilbert asked if the formal arithmetic (PA: consisting of logic and algebraic axioms and an infinite family of Induction Axioms) can be consistently extended to a complete theory. The question was somewhat vague since an obvious answer was "yes": just add to PA axioms (assumed consistent) a maximal consistent set, clearly existing albeit hard to find. K.Goedel formalized this question as existence among such extensions of recursively enumerable ones and gave it a negative answer. **Its mathematical essence is the absence of total recursive extensions of universal partial recursive predicate.**

Leonid Levin

- Gödel's original proof was based on the observation that if T is sufficiently strong, then we can formalize in T deductions that can be done with T.

How to prove Gödel's first incompleteness theorem?

D.Hilbert asked if the formal arithmetic (PA: consisting of logic and algebraic axioms and an infinite family of Induction Axioms) can be consistently extended to a complete theory. The question was somewhat vague since an obvious answer was "yes": just add to PA axioms (assumed consistent) a maximal consistent set, clearly existing albeit hard to find. K.Goedel formalized this question as existence among such extensions of recursively enumerable ones and gave it a negative answer. **Its mathematical essence is the absence of total recursive extensions of universal partial recursive predicate.**

Leonid Levin

- Gödel's original proof was based on the observation that if T is sufficiently strong, then we can formalize in T deductions that can be done with T. This is analogous to the way how Turing machines can be given (encodings of) Turing machines as input.

How to prove Gödel's first incompleteness theorem?

D.Hilbert asked if the formal arithmetic (PA: consisting of logic and algebraic axioms and an infinite family of Induction Axioms) can be consistently extended to a complete theory. The question was somewhat vague since an obvious answer was "yes": just add to PA axioms (assumed consistent) a maximal consistent set, clearly existing albeit hard to find. K.Goedel formalized this question as existence among such extensions of recursively enumerable ones and gave it a negative answer. **Its mathematical essence is the absence of total recursive extensions of universal partial recursive predicate.**

Leonid Levin

- Gödel's original proof was based on the observation that if T is sufficiently strong, then we can formalize in T deductions that can be done with T. This is analogous to the way how Turing machines can be given (encodings of) Turing machines as input.
- However, an alternative proof of Gödel's first incompleteness theorem can be given by using the fact that the Halting problem of Turing machines is undecidable.

How to prove Gödel's first incompleteness theorem?

D.Hilbert asked if the formal arithmetic (PA: consisting of logic and algebraic axioms and an infinite family of Induction Axioms) can be consistently extended to a complete theory. The question was somewhat vague since an obvious answer was "yes": just add to PA axioms (assumed consistent) a maximal consistent set, clearly existing albeit hard to find. K.Goedel formalized this question as existence among such extensions of recursively enumerable ones and gave it a negative answer. **Its mathematical essence is the absence of total recursive extensions of universal partial recursive predicate.**

Leonid Levin

- Gödel's original proof was based on the observation that if T is sufficiently strong, then we can formalize in T deductions that can be done with T. This is analogous to the way how Turing machines can be given (encodings of) Turing machines as input.
- However, an alternative proof of Gödel's first incompleteness theorem can be given by using the fact that the Halting problem of Turing machines is undecidable.
- **This lecture:** We will formalize one variant of Gödel's first incompleteness theorem and prove it using the aforementioned approach.

First-order logic

- To formalize Gödel's incompleteness theorem, we will first need to formalize the background language.

Reijo Jaakkola
reijo.jaakkola@tuni.fi

First-order logic

- To formalize Gödel's incompleteness theorem, we will first need to formalize the background language. Theorems, axioms and statements will then be **sentences** of this formal language.

First-order logic

- To formalize Gödel's incompleteness theorem, we will first need to formalize the background language. Theorems, axioms and statements will then be **sentences** of this formal language.
- A standard choice of this formal language is the first-order logic $\mathcal{FO}$.

First-order logic

- To formalize Gödel's incompleteness theorem, we will first need to formalize the background language. Theorems, axioms and statements will then be **sentences** of this formal language.
- A standard choice of this formal language is the first-order logic $\mathcal{FO}$. We will start by defining its **syntax** and **semantics**.

Vocabularies

- First, we have a vocabulary which is a set of non-logical symbols.

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Vocabularies

- First, we have a vocabulary which is a set of non-logical symbols. Here we will consider only vocabularies which have **constant** symbols and **function** symbols.

Vocabularies

- First, we have a vocabulary which is a set of non-logical symbols. Here we will consider only vocabularies which have **constant** symbols and **function** symbols.

Example

As the vocabulary τ_{Ar} of arithmetic we can choose the set

$$\{0, S, +, \cdot\}$$

1. 0 is a constant.
2. S is a function symbol (the successor function).
3. $+$ and $\cdot$ are function symbols (addition and multiplication).

Syntax of first-order logic

- In addition to a vocabulary, we have an infinite set of variables $\{x, y, z, \dots\}$ and a set of logical symbols

$$\neg, \wedge, \vee, \rightarrow, \exists, \forall$$

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Syntax of first-order logic

- In addition to a vocabulary, we have an infinite set of variables $\{x, y, z, \dots\}$ and a set of logical symbols

  $$\neg, \wedge, \vee, \rightarrow, \exists, \forall$$

  Using symbols from our vocabulary, say τ_{Ar}, and variables, we can form **terms** such as $x + y$ and $S(0)$.

Syntax of first-order logic

- In addition to a vocabulary, we have an infinite set of variables $\{x, y, z, \dots\}$ and a set of logical symbols

$$\neg, \wedge, \vee, \rightarrow, \exists, \forall$$

Using symbols from our vocabulary, say τ_{Ar}, and variables, we can form **terms** such as $x + y$ and $S(0)$. Using terms, we can form equations such as $x + y = x \cdot y$ (also called atomic formulas).

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Syntax of first-order logic

- In addition to a vocabulary, we have an infinite set of variables $\{x, y, z, \dots\}$ and a set of logical symbols

$$\neg, \wedge, \vee, \rightarrow, \exists, \forall$$

Using symbols from our vocabulary, say τ_{Ar}, and variables, we can form **terms** such as $x + y$ and $S(0)$. Using terms, we can form equations such as $x + y = x \cdot y$ (also called atomic formulas). More complex formulas can be build from these using our logical symbols.

Syntax of first-order logic

- In addition to a vocabulary, we have an infinite set of variables $\{x, y, z, \dots\}$ and a set of logical symbols

$$\neg, \wedge, \vee, \rightarrow, \exists, \forall$$

Using symbols from our vocabulary, say τ_{Ar}, and variables, we can form **terms** such as $x + y$ and $S(0)$. Using terms, we can form equations such as $x + y = x \cdot y$ (also called atomic formulas). More complex formulas can be build from these using our logical symbols.

Example

The following strings are sentences of $\mathcal{FO}$ over the vocabulary τ_{Ar}.

1. $\forall x \forall y (x + y = y + x)$

Syntax of first-order logic

- In addition to a vocabulary, we have an infinite set of variables $\{x, y, z, \ldots\}$ and a set of logical symbols

$$\neg, \wedge, \vee, \rightarrow, \exists, \forall$$

Using symbols from our vocabulary, say τ_{Ar}, and variables, we can form **terms** such as $x + y$ and $S(0)$. Using terms, we can form equations such as $x + y = x \cdot y$ (also called atomic formulas). More complex formulas can be build from these using our logical symbols.

Example

The following strings are sentences of $\mathcal{FO}$ over the vocabulary τ_{Ar}.

1. $\forall x \forall y (x + y = y + x)$
2. $\neg \exists x \exists y \exists z (\neg x = 0 \wedge \neg y = 0 \wedge \neg z = 0 \wedge x^3 + y^3 = z^3)$, where $x^3 := x \cdot (x \cdot x)$

Syntax of first-order logic

- In addition to a vocabulary, we have an infinite set of variables $\{x, y, z, \dots\}$ and a set of logical symbols

  $$\neg, \wedge, \vee, \rightarrow, \exists, \forall$$

  Using symbols from our vocabulary, say τ_{Ar}, and variables, we can form **terms** such as $x + y$ and $S(0)$. Using terms, we can form equations such as $x + y = x \cdot y$ (also called atomic formulas). More complex formulas can be build from these using our logical symbols.

Example

The following strings are sentences of $\mathcal{FO}$ over the vocabulary τ_{Ar}.

1. $\forall x \forall y (x + y = y + x)$
2. $\neg \exists x \exists y \exists z (\neg x = 0 \wedge \neg y = 0 \wedge \neg z = 0 \wedge x^3 + y^3 = z^3)$, where $x^3 := x \cdot (x \cdot x)$
3. $\forall x (x = 0 \vee \exists y (x = S(y)))$

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Semantics

- Semantics define whether a sentence is **true** in a given **structure**.

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Semantics

- Semantics define whether a sentence is **true** in a given **structure**.
- Structures define how members of the underlying vocabulary are interpreted.

Semantics

- Semantics define whether a sentence is **true** in a given **structure**.
- Structures define how members of the underlying vocabulary are interpreted. For example $(\mathbb{N}, +^{\mathbb{N}}, \cdot^{\mathbb{N}}, S^{\mathbb{N}}, 0^{\mathbb{N}})$ is a structure over the vocabulary τ_{Ar}, which we will simply denote by $\mathbb{N}$.

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Semantics

- Semantics define whether a sentence is **true** in a given **structure**.
- Structures define how members of the underlying vocabulary are interpreted. For example $(\mathbb{N}, +^{\mathbb{N}}, \cdot^{\mathbb{N}}, S^{\mathbb{N}}, 0^{\mathbb{N}})$ is a structure over the vocabulary τ_{Ar}, which we will simply denote by $\mathbb{N}$.
- We do not formally define the semantics of $\mathcal{FO}$, but it is what you would expect.

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Semantics

- Semantics define whether a sentence is **true** in a given **structure**.
- Structures define how members of the underlying vocabulary are interpreted. For example $(\mathbb{N}, +^{\mathbb{N}}, \cdot^{\mathbb{N}}, S^{\mathbb{N}}, 0^{\mathbb{N}})$ is a structure over the vocabulary τ_{Ar}, which we will simply denote by $\mathbb{N}$.
- We do not formally define the semantics of $\mathcal{FO}$, but it is what you would expect. If a sentence φ of $\mathcal{FO}$ is true in a structure $\mathbb{A}$, then we denote this by $\mathbb{A} \vDash \varphi$.

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Semantics

- Semantics define whether a sentence is **true** in a given **structure**.
- Structures define how members of the underlying vocabulary are interpreted. For example $(\mathbb{N}, +^{\mathbb{N}}, \cdot^{\mathbb{N}}, S^{\mathbb{N}}, 0^{\mathbb{N}})$ is a structure over the vocabulary τ_{Ar}, which we will simply denote by $\mathbb{N}$.
- We do not formally define the semantics of $\mathcal{FO}$, but it is what you would expect. If a sentence φ of $\mathcal{FO}$ is true in a structure $\mathbb{A}$, then we denote this by $\mathbb{A} \vDash \varphi$.

Example

1. $\mathbb{N} \vDash \forall x \forall y (x + y = y + x)$, since addition of natural numbers is commutative.

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Semantics

- Semantics define whether a sentence is **true** in a given **structure**.
- Structures define how members of the underlying vocabulary are interpreted. For example $(\mathbb{N}, +^{\mathbb{N}}, \cdot^{\mathbb{N}}, S^{\mathbb{N}}, 0^{\mathbb{N}})$ is a structure over the vocabulary τ_{Ar}, which we will simply denote by $\mathbb{N}$.
- We do not formally define the semantics of $\mathcal{FO}$, but it is what you would expect. If a sentence φ of $\mathcal{FO}$ is true in a structure $\mathbb{A}$, then we denote this by $\mathbb{A} \vDash \varphi$.

Example

1. $\mathbb{N} \vDash \forall x \forall y (x + y = y + x)$, since addition of natural numbers is commutative.
2. $\mathbb{N} \vDash \neg \exists x \exists y \exists z (\neg x = 0 \wedge \neg y = 0 \wedge \neg z = 0 \wedge x^3 + y^3 = z^3)$, because Fermat's last theorem is true.

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Semantics

- Semantics define whether a sentence is **true** in a given **structure**.
- Structures define how members of the underlying vocabulary are interpreted. For example $(\mathbb{N}, +^{\mathbb{N}}, \cdot^{\mathbb{N}}, S^{\mathbb{N}}, 0^{\mathbb{N}})$ is a structure over the vocabulary τ_{Ar}, which we will simply denote by $\mathbb{N}$.
- We do not formally define the semantics of $\mathcal{FO}$, but it is what you would expect. If a sentence φ of $\mathcal{FO}$ is true in a structure $\mathbb{A}$, then we denote this by $\mathbb{A} \models \varphi$.

Example

1. $\mathbb{N} \models \forall x \forall y (x + y = y + x)$, since addition of natural numbers is commutative.
2. $\mathbb{N} \models \neg \exists x \exists y \exists z (\neg x = 0 \wedge \neg y = 0 \wedge \neg z = 0 \wedge x^3 + y^3 = z^3)$, because Fermat's last theorem is true.
3. $\mathbb{N} \models \forall x (x = 0 \vee \exists y (x = S(y)))$, because every natural number which is not zero has a predecessor.

Reijo Jaakkola
reijo.jaakkola@tuni.fi

First-order theories

- A set T of sentences of $\mathcal{FO}$ is called a **theory** and it is **consistent**, if there exists a structure $\mathbb{A}$ so that $\mathbb{A} \models \varphi$, for every $\varphi \in T$.

First-order theories

- A set T of sentences of $\mathcal{FO}$ is called a **theory** and it is **consistent**, if there exists a structure $\mathbb{A}$ so that $\mathbb{A} \vDash \varphi$, for every $\varphi \in T$. Note that T can be infinite!

Reijo Jaakkola
reijo.jaakkola@tuni.fi

First-order theories

- A set T of sentences of $\mathcal{FO}$ is called a **theory** and it is **consistent**, if there exists a structure $\mathbb{A}$ so that $\mathbb{A} \vDash \varphi$, for every $\varphi \in T$. Note that T can be infinite!
- **What are mathematicians doing**: Finding **logical** consequences of T, for various T.

First-order theories

- A set T of sentences of $\mathcal{FO}$ is called a **theory** and it is **consistent**, if there exists a structure $\mathbb{A}$ so that $\mathbb{A} \models \varphi$, for every $\varphi \in T$. Note that T can be infinite!
- **What are mathematicians doing**: Finding **logical** consequences of T, for various T.
- We say that φ is a **logical consequence** of T, denoted by $T \models \varphi$, if for every structure $\mathbb{A}$ we have that if $\mathbb{A} \models T$, then $\mathbb{A} \models \varphi$.

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Axioms of Peano Arithmetic

$$P_1 \quad \forall x\,(S(x) \neq 0)$$

$$P_2 \quad \forall x \forall y\,(S(x) = S(y) \rightarrow x = y)$$

$$P_3 \quad \forall x\,(x + 0 = x)$$

$$P_4 \quad \forall x \forall y\,(x + S(y) = S(x+y))$$

$$P_5 \quad \forall x\,(x \cdot 0 = 0)$$

$$P_6 \quad \forall x \forall y\,(x \cdot S(y) = (x \cdot y) + x)$$

$$(P_7^{\varphi}) \quad \forall \vec{y}\Big[\big(\varphi(0,\vec{y}) \wedge \forall x(\varphi(x,\vec{y}) \rightarrow \varphi(S(x),\vec{y}))\big) \rightarrow \forall x\,\varphi(x,\vec{y})\Big]$$

Reijo Jaakkola
reijo.jaakkola@tuni.fi

What is a proof system?

- **Essentially**: a system of **formal rules**

$$\varphi_1, \ldots, \varphi_n \vdash \varphi$$

which allow us to deduce new propositions from previously deduced propositions (including any assumptions).

What is a proof system?

- **Essentially**: a system of **formal rules**

$$\varphi_1, \ldots, \varphi_n \vdash \varphi$$

which allow us to deduce new propositions from previously deduced propositions (including any assumptions). Depends on the background logic!

What is a proof system?

- **Essentially**: a system of **formal rules**

$$\varphi_1, \ldots, \varphi_n \vdash \varphi$$

  which allow us to deduce new propositions from previously deduced propositions (including any assumptions). Depends on the background logic!

- Proofs of a proof system P are then sequences

$$\varphi_1, \varphi_2, \ldots, \varphi_n,$$

  where each φ_i is either an assumption or it was deduced from propositions that occurred before it using a rule of P.

Reijo Jaakkola
reijo.jaakkola@tuni.fi

What is a proof system?

- **Essentially**: a system of **formal rules**

$$\varphi_1, \ldots, \varphi_n \vdash \varphi$$

  which allow us to deduce new propositions from previously deduced propositions (including any assumptions). Depends on the background logic!

- Proofs of a proof system P are then sequences

$$\varphi_1, \varphi_2, \ldots, \varphi_n,$$

  where each φ_i is either an assumption or it was deduced from propositions that occurred before it using a rule of P. **Proofs are finite**!

What is a proof system?

- **Essentially**: a system of **formal rules**

$$\varphi_1, \ldots, \varphi_n \vdash \varphi$$

  which allow us to deduce new propositions from previously deduced propositions (including any assumptions). Depends on the background logic!

- Proofs of a proof system P are then sequences

$$\varphi_1, \varphi_2, \ldots, \varphi_n,$$

  where each φ_i is either an assumption or it was deduced from propositions that occurred before it using a rule of P. **Proofs are finite**!

- If rules are effective, which they should be since they are purely formal, then all the proofs of P can be enumerated by a Turing machine!!

Example of an $\mathcal{FO}$-deduction

$$\dfrac{\dfrac{\dfrac{\dfrac{[\neg\exists x A]^{2} \qquad \dfrac{[A]^{1}}{\exists x A}}{\exists x A \wedge \neg\exists x A}\;\wedge T}{\neg A}\;\neg T, 1}{\forall x \neg A}\;\forall T}{\neg\exists x A \to \forall x \neg A}\;\to T, 2$$

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Complete proof systems

- A remarkable property of $\mathcal{FO}$ is that there exists proof systems for which the following equivalence holds, for every set of $\mathcal{FO}$-sentences T and an $\mathcal{FO}$-sentence φ:

  $$T \models \varphi \iff T \vdash \varphi,$$

  where $T \vdash \varphi$ means that φ can be deduced from T.

Complete proof systems

- A remarkable property of $\mathcal{FO}$ is that there exists proof systems for which the following equivalence holds, for every set of $\mathcal{FO}$-sentences T and an $\mathcal{FO}$-sentence φ:

$$T \vDash \varphi \iff T \vdash \varphi,$$

where $T \vdash \varphi$ means that φ can be deduced from T. Such proof systems are called **complete**.

Complete proof systems

- A remarkable property of $\mathcal{FO}$ is that there exists proof systems for which the following equivalence holds, for every set of $\mathcal{FO}$-sentences T and an $\mathcal{FO}$-sentence φ:

$$T \vDash \varphi \iff T \vdash \varphi,$$

where $T \vdash \varphi$ means that φ can be deduced from T. Such proof systems are called **complete**.

- What this implies is that it does not really matter what proof system we fix as long as it is complete, since they all prove the same theorems.

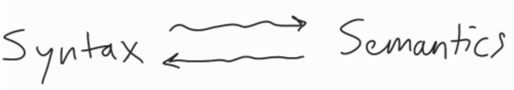
Reijo Jaakkola
reijo.jaakkola@tuni.fi

Complete proof systems

- A remarkable property of $\mathcal{FO}$ is that there exists proof systems for which the following equivalence holds, for every set of $\mathcal{FO}$-sentences T and an $\mathcal{FO}$-sentence φ:

$$T \vDash \varphi \iff T \vdash \varphi,$$

  where $T \vdash \varphi$ means that φ can be deduced from T. Such proof systems are called **complete**.
- What this implies is that it does not really matter what proof system we fix as long as it is complete, since they all prove the same theorems.



Back to Gödel's incompleteness theorem

Theorem (Gödel's first incompleteness theorem, formal version)

Suppose that T is a computable and consistent set of $\mathcal{FO}$-sentences over the vocabulary τ_{Ar}. Then

$$\{\varphi \in \mathcal{FO} \mid \mathbb{N} \vDash \varphi\} \nsubseteq \{\varphi \in \mathcal{FO} \mid T \vdash \varphi\}$$

Outline of the argument

Let $\mathrm{Th}(\mathbb{N}) := \{\varphi \in \mathcal{FO} \mid \mathbb{N} \vDash \varphi\}$ and $T^{\vdash} := \{\varphi \in \mathcal{FO} \mid T \vdash \varphi\}$.

Outline of the argument

Let $\mathrm{Th}(\mathbb{N}) := \{\varphi \in \mathcal{FO} \mid \mathbb{N} \vDash \varphi\}$ and $T^{\vdash} := \{\varphi \in \mathcal{FO} \mid T \vdash \varphi\}$.

1. Suppose that T is a consistent set of $\mathcal{FO}$-sentences over the vocabulary τ_{Ar} so that $\mathrm{Th}(\mathbb{N}) \subseteq T^{\vdash}$ and T is computable.

Outline of the argument

Let $\mathrm{Th}(\mathbb{N}) := \{\varphi \in \mathcal{FO} \mid \mathbb{N} \vDash \varphi\}$ and $T^{\vdash} := \{\varphi \in \mathcal{FO} \mid T \vdash \varphi\}$.

1. Suppose that T is a consistent set of $\mathcal{FO}$-sentences over the vocabulary τ_{Ar} so that $\mathrm{Th}(\mathbb{N}) \subseteq T^{\vdash}$ and T is computable.
2. One can show that there exists a computable mapping which maps each Turing machine M into an arithmetical $\mathcal{FO}$-sentence φ_M so that M halts if and only if $\mathbb{N} \vDash \varphi_M$. (This is the tricky part!)

Outline of the argument

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Let $\mathrm{Th}(\mathbb{N}) := \{\varphi \in \mathcal{FO} \mid \mathbb{N} \models \varphi\}$ and $T^{\vdash} := \{\varphi \in \mathcal{FO} \mid T \vdash \varphi\}$.

1. Suppose that T is a consistent set of $\mathcal{FO}$-sentences over the vocabulary τ_{Ar} so that $\mathrm{Th}(\mathbb{N}) \subseteq T^{\vdash}$ and T is computable.
2. One can show that there exists a computable mapping which maps each Turing machine M into an arithmetical $\mathcal{FO}$-sentence φ_M so that M halts if and only if $\mathbb{N} \models \varphi_M$. (This is the tricky part!)
3. This gives us a way to determine whether a given Turing machine M halts, since we can enumerate the sentences in $T^{\vdash}$ until we encounter either φ_M or $\neg\varphi_M$.

Outline of the argument

Let $\mathrm{Th}(\mathbb{N}) := \{\varphi \in \mathcal{FO} \mid \mathbb{N} \vDash \varphi\}$ and $T^{\vdash} := \{\varphi \in \mathcal{FO} \mid T \vdash \varphi\}$.

1. Suppose that T is a consistent set of $\mathcal{FO}$-sentences over the vocabulary τ_{Ar} so that $\mathrm{Th}(\mathbb{N}) \subseteq T^{\vdash}$ and T is computable.
2. One can show that there exists a computable mapping which maps each Turing machine M into an arithmetical $\mathcal{FO}$-sentence φ_M so that M halts if and only if $\mathbb{N} \vDash \varphi_M$. (This is the tricky part!)
3. This gives us a way to determine whether a given Turing machine M halts, since we can enumerate the sentences in $T^{\vdash}$ until we encounter either φ_M or $\neg\varphi_M$.
    a) We know that we will eventually encounter one of them, since $\mathrm{Th}(\mathbb{N}) \subseteq T^{\vdash}$.

Outline of the argument

Let $\mathrm{Th}(\mathbb{N}) := \{\varphi \in \mathcal{FO} \mid \mathbb{N} \vDash \varphi\}$ and $T^{\vdash} := \{\varphi \in \mathcal{FO} \mid T \vdash \varphi\}$.

1. Suppose that T is a consistent set of $\mathcal{FO}$-sentences over the vocabulary τ_{Ar} so that $\mathrm{Th}(\mathbb{N}) \subseteq T^{\vdash}$ and T is computable.
2. One can show that there exists a computable mapping which maps each Turing machine M into an arithmetical $\mathcal{FO}$-sentence φ_M so that M halts if and only if $\mathbb{N} \vDash \varphi_M$. (This is the tricky part!)
3. This gives us a way to determine whether a given Turing machine M halts, since we can enumerate the sentences in $T^{\vdash}$ until we encounter either φ_M or $\neg\varphi_M$.
    a) We know that we will eventually encounter one of them, since $\mathrm{Th}(\mathbb{N}) \subseteq T^{\vdash}$.
    b) Since T is consistent, we know that it can not contain both φ_M and $\neg\varphi_M$.

Outline of the argument

Let $\mathrm{Th}(\mathbb{N}) := \{\varphi \in \mathcal{FO} \mid \mathbb{N} \models \varphi\}$ and $T^{\vdash} := \{\varphi \in \mathcal{FO} \mid T \vdash \varphi\}$.

1. Suppose that T is a consistent set of $\mathcal{FO}$-sentences over the vocabulary τ_{Ar} so that $\mathrm{Th}(\mathbb{N}) \subseteq T^{\vdash}$ and T is computable.
2. One can show that there exists a computable mapping which maps each Turing machine M into an arithmetical $\mathcal{FO}$-sentence φ_M so that M halts if and only if $\mathbb{N} \models \varphi_M$. (This is the tricky part!)
3. This gives us a way to determine whether a given Turing machine M halts, since we can enumerate the sentences in $T^{\vdash}$ until we encounter either φ_M or $\neg\varphi_M$.
    a) We know that we will eventually encounter one of them, since $\mathrm{Th}(\mathbb{N}) \subseteq T^{\vdash}$.
    b) Since T is consistent, we know that it can not contain both φ_M and $\neg\varphi_M$.
4. Thus the halting problem is decidable, which is a contradiction.

Comparison to Gödel's original proof

- In the above argument the use of a diagonalization argument is hidden in the argument that the Halting problem for Turing machines is undecidable.

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Comparison to Gödel's original proof

- In the above argument the use of a diagonalization argument is hidden in the argument that the Halting problem for Turing machines is undecidable. Gödel's original proof used a similar diagonalization argument directly via the so-called fixed-point lemma.

Comparison to Gödel's original proof

- In the above argument the use of a diagonalization argument is hidden in the argument that the Halting problem for Turing machines is undecidable. Gödel's original proof used a similar diagonalization argument directly via the so-called fixed-point lemma.
- More precisely, Gödel proved that if T is sufficiently strong ($\mathrm{PA} \subseteq T$ is more than enough), then there are sentences which can speak about their own properties ("I am not provable").
- The proof of this is not too difficult, but it becomes more involved if you want to prove that a concrete theory, such as PA, is sufficiently strong.

Comparison to Gödel's original proof

- In the above argument the use of a diagonalization argument is hidden in the argument that the Halting problem for Turing machines is undecidable. Gödel's original proof used a similar diagonalization argument directly via the so-called fixed-point lemma.
- More precisely, Gödel proved that if T is sufficiently strong ($\mathrm{PA} \subseteq T$ is more than enough), then there are sentences which can speak about their own properties ("I am not provable").
- The proof of this is not too difficult, but it becomes more involved if you want to prove that a concrete theory, such as PA, is sufficiently strong. However, this approach naturally yields a proof of Gödel's second incompleteness theorem.

The second incompleteness theorem

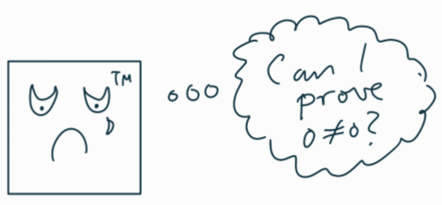
Theorem (Gödel's second incompleteness theorem, informal version)

Suppose that $T \supseteq \mathrm{PA}$ is a computable and consistent set of $\mathcal{FO}$-sentences over the vocabulary τ_{Ar}. Then T can not prove that it is consistent.

The second incompleteness theorem

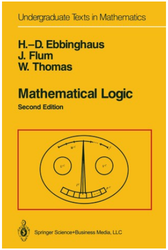
Theorem (Gödel's second incompleteness theorem, informal version)

Suppose that $T \supseteq \mathrm{PA}$ is a computable and consistent set of $\mathcal{FO}$-sentences over the vocabulary τ_{Ar}. Then T can not prove that it is consistent.



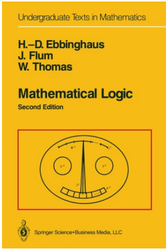
Pointers to literature

- This lecture was partially based on Chapter X in the book



Pointers to literature

- This lecture was partially based on Chapter X in the book



- Panu Raatikainen has an excellent entry on Gödel's incompleteness theorem in Stanford Encyclopedia of Philosophy, which contains several good references:
`https://plato.stanford.edu/entries/goedel-incompleteness/`.

Thanks! :) Questions?

Reijo Jaakkola
reijo.jaakkola@tuni.fi