Formal Proof

수학문제연구회 2023F Seminar

36기 한승우

KAIST 수학문제연구회

2023년 11월 20일

Overview

이 세미나의 목적은...

1. Formal Proof의 (현실적/이론적) 필요성과 정의

Overview

이 세미나의 목적은...

1. Formal Proof의 (현실적/이론적) 필요성과 정의
2. ...를 알기 위한 배경(?) 지식

Overview

이 세미나의 목적은...

1. Formal Proof의 (현실적/이론적) 필요성과 정의
2. ...를 알기 위한 배경(?) 지식
3. ...을 새내기 친화적 언어로 설명하기!

Overview

이 세미나의 목적은...

1. Formal Proof의 (현실적/이론적) 필요성과 정의
2. ...를 알기 위한 배경(?) 지식
3. ...을 새내기 친화적 언어로 설명하기!
4. ...그리고 Gödel's First Incompleteness Theorem.

The most trustworthy types of proofs (1)

Proof:

(1) Included in the proof of 2.3.
(2) Similarly.
(3) Straightforward.
(4) Check.

The proof is trivial, and is left as an exercise.

The most trustworthy types of proofs (2)

129페이지짜리 페르마의 마지막 정리 증명

Annals of Mathematics, **141** (1995), 443-551

Pierre de Fermat

Modular elliptic curves and Fermat's Last Theorem

By Andrew John Wiles*

For Nada, Claire, Kate and Olivia

Andrew John Wiles

Cubum autem in duos cubos, aut quadratoquadratum in duos quadratoquadratos, et generaliter nullam in infinitum ultra quadratum potestatum in duos ejusdem nominis fas est dividere: cujes rei demonstrationem mirabilem sane detexi. Hanc marginis exiguitas non caperet.

- Pierre de Fermat ~ 1637

The most trustworthy types of proofs (2)

129페이지짜리 페르마의 마지막 정리 증명

Annals of Mathematics, **141** (1995), 443-551

Pierre de Fermat

Modular elliptic curves and Fermat's Last Theorem

By Andrew John Wiles*

For Nada, Claire, Kate and Olivia

Andrew John Wiles

Cubum autem in duos cubos, aut quadratoquadratum in duos quadratoquadratos, et generaliter nullam in infinitum ultra quadratum potestatum in duos ejusdem nominis fas est dividere: cujes rei demonstrationem mirabilem sane detexi. Hanc marginis exiguitas non caperet.

- Pierre de Fermat ~ 1637

이 증명을 검증하는 데에만 7+년이 걸림...

The most trustworthy types of proofs (3)

괭장히 간결하고 아름다우며 수학자들에게
10년간 받아들여지고 있는 4색 정리의 증명 [1]

On the Geographical Problem of the Four Colours.

By A. B. Kempe, B. A., *London*, *England*.

If we examine any ordinary map, we shall find in general a number of lines dividing it into districts, and a number of others denoting rivers, roads, etc. It frequently happens that the multiplicity of the latter lines renders it extremely difficult to distinguish the boundary lines from them. In cases where it is important that the distinction should be clearly marked the arti

[1] by Kempe, 1879

The most trustworthy types of proofs (3)

괭장히 간결하고 아름다우며 수학자들에게
10년간 받아들여지고 있는 4색 정리의 증명 [1]

On the Geographical Problem of the Four Colours.

By A. B. Kempe, B. A., *London*, *England*.

If we examine any ordinary map, we shall find in general a number of lines dividing it into districts, and a number of others denoting rivers, roads, etc. It frequently happens that the multiplicity of the latter lines renders it extremely difficult to distinguish the boundary lines from them. In cases where it is important that the distinction should be clearly marked the arti

(...곧 증명의 오류가 발표될 예정인)

[1] ~~by Kempe, 1879~~

The most trustworthy types of proofs (4)

매주 당신의 사무실에 도착하고 있는 리만 가설의 증명(들)...

Here comes the rescue!

The Idea

이미 컴퓨터는 인간에게 불가능한 양의 계산을 할 수 있음... 만약 증명을 computable object로 끌어내린다면...?

Here comes the rescue!

The Idea

이미 컴퓨터는 인간에게 불가능한 양의 계산을 할 수 있음... 만약 증명을 computable object로 끌어내린다면...?

1. 증명의 검증을 컴퓨터에게 맡길 수 있음.

Here comes the rescue!

The Idea

이미 컴퓨터는 인간에게 불가능한 양의 계산을 할 수 있음... 만약 증명을 computable object로 끌어내린다면...?

1. 증명의 검증을 컴퓨터에게 맡길 수 있음.
2. 잘만 하면 증명을 생성할 수 있을지도?

Here comes the rescue!

The Idea

이미 컴퓨터는 인간에게 불가능한 양의 계산을 할 수 있음... 만약 증명을 computable object로 끌어내린다면...?

1. 증명의 검증을 컴퓨터에게 맡길 수 있음.
2. 잘만 하면 증명을 생성할 수 있을지도?
3. 수학자들 실직시킬 수 있음 ~~self-unemployment~~

Here comes the rescue!

The Idea

이미 컴퓨터는 인간에게 불가능한 양의 계산을 할 수 있음... 만약 증명을 computable object로 끌어내린다면...?

1. 증명의 검증을 컴퓨터에게 맡길 수 있음.
2. 잘만 하면 증명을 생성할 수 있을지도?
3. 수학자들 실직시킬 수 있음 ~~self-unemployment~~

하지만 이를 위해서는 정리, 증명 등의 명확한(수학적) 정의가 필요!

Overview

오늘 우리가 알아볼 것:

Overview

오늘 우리가 알아볼 것:

1. First-Order Logic
2. Syntax(문법)과 Semantics(의미)
3. Proof Theory
4. Theory와 Theorem(정리)

Overview

오늘 우리가 알아볼 것:

1. First-Order Logic
2. Syntax(문법)과 Semantics(의미)
3. Proof Theory
4. Theory와 Theorem(정리)
5. Gödel's First/Second Incompleteness Theorem의 증명
   - 매우 non-rigorous함

Overview

오늘 우리가 알아볼 것:

1. First-Order Logic
2. Syntax(문법)과 Semantics(의미)
3. Proof Theory
4. Theory와 Theorem(정리)
5. Gödel's First/Second Incompleteness Theorem의 증명
   - 매우 non-rigorous함

Language

언어는 문장들로 이루어지고, 각 문장은 문자로 이루어짐.

- 한국어의 문장을 구성하는 문자는 한글+문장부호.
- 일본어의 문장을 구성하는 문자는 칸지+가나+문장부호.

Language

언어는 문장들로 이루어지고, 각 문장은 문자로 이루어짐.

- 한국어의 문장을 구성하는 문자는 한글+문장부호.
- 일본어의 문장을 구성하는 문자는 칸지+가나+문장부호.

Definition: Alphabet Σ

An alphabet is a (finite or countably infinite) set of symbols.

- e.g. $\Sigma = \{a, b, c\}$

Language

언어는 문장들로 이루어지고, 각 문장은 문자로 이루어짐.

- 한국어의 문장을 구성하는 문자는 한글+문장부호.
- 일본어의 문장을 구성하는 문자는 칸지+가나+문장부호.

Definition: Alphabet Σ

An alphabet is a (finite or countably infinite) set of symbols.

- e.g. $\Sigma = \{a, b, c\}$

Definition: Strings Σ^*

The set of strings: $\Sigma^* \triangleq \bigcup_{n=0}^{\infty} \Sigma^n$

- Σ^*는 Σ의 알파벳을 임의의 길이만큼 늘여 놓은 것들의 집합
- e.g. $\Sigma^* = \{\epsilon, a, b, c, aa, ab, ac, ba, \cdots\}$ [a]

[a] ϵ: zero-length empty string

Language

Definition: Language $\mathcal{L}$

A language is a subset of Σ^*.

- e.g. $\mathcal{L} = \{a, abab, aca\} \subseteq \{a, b, c\}^*$

Language

Definition: Language $\mathcal{L}$

A language is a subset of Σ^*.

- e.g. $\mathcal{L} = \{a, abab, aca\} \subseteq \{a, b, c\}^*$

아니 이러면 language가 너무 많지 않나?

Language

Definition: Language $\mathcal{L}$

A language is a subset of Σ^*.

- e.g. $\mathcal{L} = \{a, abab, aca\} \subseteq \{a, b, c\}^*$

아니 이러면 language가 너무 많지 않나?

- 그래서 우리는 특별하게 생긴 language만을 대상으로 해서 현실성 있으면서도 충분히 일반적인 결과를 얻고자 함.
- 그것들 중 하나가 First-Order Logic.

Preview

First-Order Logic(FOL)의 문장은 대충 이렇게 생김:

$$\neg(x > 0) \vee \exists z,\ z \times z = x$$

$$\forall x,\ \forall y,\ x \geq y \Longrightarrow x + 1 \geq y$$

$$\varepsilon > 0 \Longrightarrow \exists \delta,\ \delta > 0 \wedge \left(\forall x,\ |x - a| < \delta \Longrightarrow |f(x) - f(a)| < \varepsilon\right)$$

Preview

First-Order Logic(FOL)의 문장은 대충 이렇게 생김:

$$\neg(x > 0) \vee \exists z,\ z \times z = x$$

$$\forall x,\ \forall y,\ x \geq y \Longrightarrow x + 1 \geq y$$

$$\varepsilon > 0 \Longrightarrow \exists \delta,\ \delta > 0 \wedge \left(\forall x,\ |x - a| < \delta \Longrightarrow |f(x) - f(a)| < \varepsilon\right)$$

우리는 저 문장들의 참/거짓을 따지지 않음! 아직은:

- $a = b$에 "a와 b가 같다"의 의미가 없고,
- $|x|$에 "x의 절댓값"의 의미가 없으며,
- $A \Rightarrow B$에 "A이면 B이다"의 의미가 없음.

뒤의 semantics 부분까지 stay tuned…

Logical Symbol

$$(\neg x > 0) \vee \exists z,\ z \times z = x$$

$$\forall x,\ \forall y,\ x \geq y \Longrightarrow x + 1 \geq y$$

$$\varepsilon > 0 \Longrightarrow \exists \delta,\ \delta > 0 \wedge \left(\forall x,\ |x - a| < \delta \Longrightarrow |f(x) - f(a)| < \varepsilon\right)$$

빨간색으로 칠해진 것들은 logical symbol이라고 부름. 얘네들은 FOL language의 alphabet의 기본 구성품.

Definition: Logical Symbol

$$\{\ \underbrace{\forall, \exists,}_{\text{logical quantifiers}}\ \overbrace{\wedge, \vee, \Rightarrow, \neg,}^{\text{logical connectives}}\ \underbrace{(,), \text{“}, \text{”},}_{\text{punctuation symbols}}\ \overbrace{\top, \bot}^{\text{true/false}}\ \}$$

$$\cup \underbrace{(\text{a set of countably infinite letters})}_{\text{variables}} \subseteq \Sigma$$

Non-logical Symbol (Signature)

공통적인 alphabet 말고도 FOL Language마다 다를 수 있는 alphabet 도 넣을 수 있도록 하자. 이런 alphabet들을 non-logical symbol이라고 하는데, 이들을 통해 FOL language를 unique하게 정할 것이므로, 이들을 signature라고도 부른다.

Non-logical Symbol (Signature)

공통적인 alphabet 말고도 FOL Language마다 다를 수 있는 alphabet 도 넣을 수 있도록 하자. 이런 alphabet들을 non-logical symbol이라고 하는데, 이들을 통해 FOL language를 unique하게 정할 것이므로, 이들을 signature라고도 부른다.

Definition: Signature $\sigma = (\mathcal{F}, \mathcal{R}, \mathrm{ar})$

A signature is a tuple $\sigma = (\mathcal{F}, \mathcal{R}, \mathrm{ar})$ where

- $\mathcal{F}$ is a finite set of function symbols.
- $\mathcal{R}$ is a finite set of relation symbols.
- ar is a function $\mathrm{ar} : \mathcal{F} \cup \mathcal{R} \to \mathbb{N}_0$.

Non-logical Symbol (Signature)

Definition: Signature $\sigma = (\mathcal{F}, \mathcal{R}, \mathrm{ar})$

A signature is a tuple $\sigma = (\mathcal{F}, \mathcal{R}, \mathrm{ar})$ where

- $\mathcal{F}$ is a finite set of function symbols.
  - e.g. $\mathcal{F} = \{f, g, h, +, -, \times\}$.
- $\mathcal{R}$ is a finite set of relation symbols.
  - e.g. $\mathcal{R} = \{=, \leq\}$.
- ar is a function $\mathrm{ar} : \mathcal{F} \cup \mathcal{R} \to \mathbb{N}_0$.
  - Arity function ar는 function/relation이 몇 개의 인자를 받는지 나타냄.
  - 아직도 f, $+$, $\leq$ 등의 symbol에는 의미가 없으므로, 문법을 정의하기 위한 목적으로 필요함. (다음 페이지에서 설명)
  - e.g. $\mathrm{ar}(+) = 2, \mathrm{ar}(\leq) = 2$.

Non-logical Symbol (Signature)

Definition: Signature $\sigma = (\mathcal{F}, \mathcal{R}, \mathrm{ar})$

A signature is a tuple $\sigma = (\mathcal{F}, \mathcal{R}, \mathrm{ar})$ where

- $\mathcal{F}$ is a finite set of function symbols.
  - e.g. $\mathcal{F} = \{f, g, h, +, -, \times\}$.
- $\mathcal{R}$ is a finite set of relation symbols.
  - e.g. $\mathcal{R} = \{=, \leq\}$.
- ar is a function $\mathrm{ar} : \mathcal{F} \cup \mathcal{R} \to \mathbb{N}_0$.
  - Arity function ar는 function/relation이 몇 개의 인자를 받는지 나타냄.
  - 아직도 f, $+$, $\leq$ 등의 symbol에는 의미가 없으므로, 문법을 정의하기 위한 목적으로 필요함. (다음 페이지에서 설명)
  - e.g. $\mathrm{ar}(+) = 2, \mathrm{ar}(\leq) = 2$.

FOL Language의 signature $\sigma = (\mathcal{F}, \mathcal{R}, \mathrm{ar})$이 주어지면, language의 alphabets Σ는 (logical symbols) $\cup\, \mathcal{F} \cup \mathcal{R}$로 결정됨.

FOL Language

Definition: FOL Language $\mathcal{L}$

주어진 signature $\sigma = (\mathcal{F}, \mathcal{R}, \mathrm{ar})$에 대해, alphabet $\Sigma = (\text{logical symbols}) \cup \mathcal{F} \cup \mathcal{R}$ 위에서 σ에 대응되는 언어 $\mathcal{L} \subseteq \Sigma^*$은 '항'을 나타내는 집합 $\mathcal{L}_{\mathrm{Term}}$과 함께 다음을 만족하는 가장 작은 집합으로 정의된다.

- $\mathcal{L}_{\mathrm{Term}} \ni x$ for each variable x
- $\mathcal{L}_{\mathrm{Term}} \ni f(t_1, \cdots, t_{\mathrm{ar}(f)})$ for each $f \in \mathcal{F}$ and $t_i \in \mathcal{L}_{\mathrm{Term}}$
- $\mathcal{L} \ni \top, \bot$
- $\mathcal{L} \ni R(t_1, \cdots, t_{\mathrm{ar}(R)})$ for each $R \in \mathcal{R}$ and $t_i \in \mathcal{L}_{\mathrm{Term}}$
- $\mathcal{L} \ni \neg A,\ A \wedge B,\ A \vee B,\ A \Rightarrow B$ for each $A, B \in \mathcal{L}$
- $\mathcal{L} \ni \forall x\ A,\ \exists x\ A$ for each $A \in \mathcal{L}$ and variable x

FOL Language

Definition: FOL Language $\mathcal{L}$

주어진 signature $\sigma = (\mathcal{F}, \mathcal{R}, \mathrm{ar})$에 대해, alphabet $\Sigma = (\text{logical symbols}) \cup \mathcal{F} \cup \mathcal{R}$ 위에서 σ에 대응되는 언어 $\mathcal{L} \subseteq \Sigma^*$은 '항'을 나타내는 집합 $\mathcal{L}_{\mathrm{Term}}$과 함께 다음을 만족하는 가장 작은 집합으로 정의된다.

- $\mathcal{L}_{\mathrm{Term}} \ni x$ for each variable x
- $\mathcal{L}_{\mathrm{Term}} \ni f(t_1, \cdots, t_{\mathrm{ar}(f)})$ for each $f \in \mathcal{F}$ and $t_i \in \mathcal{L}_{\mathrm{Term}}$
- $\mathcal{L} \ni \top, \bot$
- $\mathcal{L} \ni R(t_1, \cdots, t_{\mathrm{ar}(R)})$ for each $R \in \mathcal{R}$ and $t_i \in \mathcal{L}_{\mathrm{Term}}$
- $\mathcal{L} \ni \neg A,\ A \wedge B,\ A \vee B,\ A \Rightarrow B$ for each $A, B \in \mathcal{L}$
- $\mathcal{L} \ni \forall x\ A,\ \exists x\ A$ for each $A \in \mathcal{L}$ and variable x

$\mathcal{L}_{\mathrm{Term}}$은 '항처럼 생긴 것들'의 집합, $\mathcal{L}$은 '명제처럼 생긴 것들'의 집합

FOL Language

- $\mathcal{L}_{\text{Term}} \ni x$ for each variable x
- $\mathcal{L}_{\text{Term}} \ni f(t_1, \cdots, t_{\text{ar}(f)})$ for each $f \in \mathcal{F}$ and $t_i \in \mathcal{L}_{\text{Term}}$
- $\mathcal{L} \ni \top, \bot$
- $\mathcal{L} \ni R(t_1, \cdots, t_{\text{ar}(R)})$ for each $R \in \mathcal{R}$ and $t_i \in \mathcal{L}_{\text{Term}}$
- $\mathcal{L} \ni \neg A,\ A \wedge B,\ A \vee B,\ A \Rightarrow B$ for each $A, B \in \mathcal{L}$
- $\mathcal{L} \ni \forall x\ A,\ \exists x\ A$ for each $A \in \mathcal{L}$ and variable x

FOL Language

- $\mathcal{L}_{\text{Term}} \ni x$ for each variable x
- $\mathcal{L}_{\text{Term}} \ni f(t_1, \cdots, t_{\text{ar}(f)})$ for each $f \in \mathcal{F}$ and $t_i \in \mathcal{L}_{\text{Term}}$
- $\mathcal{L} \ni \top, \bot$
- $\mathcal{L} \ni R(t_1, \cdots, t_{\text{ar}(R)})$ for each $R \in \mathcal{R}$ and $t_i \in \mathcal{L}_{\text{Term}}$
- $\mathcal{L} \ni \neg A,\ A \wedge B,\ A \vee B,\ A \Rightarrow B$ for each $A, B \in \mathcal{L}$
- $\mathcal{L} \ni \forall x\ A,\ \exists x\ A$ for each $A \in \mathcal{L}$ and variable x

- $\mathcal{L}_{\text{Term}} \cap \mathcal{L} = \emptyset$.

FOL Language

- $\mathcal{L}_{\text{Term}} \ni x$ for each variable x
- $\mathcal{L}_{\text{Term}} \ni f(t_1, \cdots, t_{\text{ar}(f)})$ for each $f \in \mathcal{F}$ and $t_i \in \mathcal{L}_{\text{Term}}$
- $\mathcal{L} \ni \top, \bot$
- $\mathcal{L} \ni R(t_1, \cdots, t_{\text{ar}(R)})$ for each $R \in \mathcal{R}$ and $t_i \in \mathcal{L}_{\text{Term}}$
- $\mathcal{L} \ni \neg A,\ A \wedge B,\ A \vee B,\ A \Rightarrow B$ for each $A, B \in \mathcal{L}$
- $\mathcal{L} \ni \forall x\ A,\ \exists x\ A$ for each $A \in \mathcal{L}$ and variable x

- $\mathcal{L}_{\text{Term}} \cap \mathcal{L} = \emptyset$.
- $\mathcal{L}_{\text{Term}}$의 원소는 공간의 원소를 의미하게 될 예정.
- $\mathcal{L}$의 원소는 명제의 진리치를 의미하게 될 예정.

FOL Language

- $\mathcal{L}_{\text{Term}} \ni x$ for each variable x
- $\mathcal{L}_{\text{Term}} \ni f(t_1, \cdots, t_{\text{ar}(f)})$ for each $f \in \mathcal{F}$ and $t_i \in \mathcal{L}_{\text{Term}}$
- $\mathcal{L} \ni \top, \bot$
- $\mathcal{L} \ni R(t_1, \cdots, t_{\text{ar}(R)})$ for each $R \in \mathcal{R}$ and $t_i \in \mathcal{L}_{\text{Term}}$
- $\mathcal{L} \ni \neg A,\ A \wedge B,\ A \vee B,\ A \Rightarrow B$ for each $A, B \in \mathcal{L}$
- $\mathcal{L} \ni \forall x\ A,\ \exists x\ A$ for each $A \in \mathcal{L}$ and variable x

- $\mathcal{L}_{\text{Term}} \cap \mathcal{L} = \emptyset$.
- $\mathcal{L}_{\text{Term}}$의 원소는 공간의 원소를 의미하게 될 예정.
- $\mathcal{L}$의 원소는 명제의 진리치를 의미하게 될 예정.
- $A \in \mathcal{L}$에서 $\forall, \exists$에 묶여 있지 않은 variable을 free-variable이라 함.
- $A \in \mathcal{L}$에 free variable이 없으면 A를 $\mathcal{L}$-sentence라고 부름.
  $\mathcal{L}_S = (\mathcal{L}\text{-sentence}$들의 집합).

FOL Language: Example

Let $\sigma = (\mathcal{F}, \mathcal{R}, \mathrm{ar})$ where

- $\mathcal{F} = \{S, \oplus, \overline{0}, \overline{1}\}$
- $\mathcal{R} = \{=, \leq\}$
- $\mathrm{ar}(\overline{0}) = \mathrm{ar}(\overline{1}) = 0$ i.e., constant symbols
  $\mathrm{ar}(S) = 1,\ \mathrm{ar}(\oplus) = \mathrm{ar}(=) = \mathrm{ar}(\leq) = 2.$

Then,

- $\mathcal{L}_{\mathsf{Term}} \ni \overline{0}, \overline{1}$
- $\mathcal{L}_{\mathsf{Term}} \ni \oplus\big(S(S(\overline{1})), \oplus(\overline{1}, S(\overline{1}))\big)$ $S(S(\overline{1})) \oplus \big(\overline{1} \oplus S(\overline{1})\big)$ in infix
- $\mathcal{L}_{\mathsf{Term}} \not\ni \oplus(\overline{0}, \overline{1}, S(x))$ violates arity
- $\mathcal{L} \ni \leq(S(x), S(y)) \Rightarrow \leq(x, y)$ $S(x) \leq S(y) \Rightarrow x \leq y$ in infix
- $\mathcal{L} \setminus \mathcal{L}_{\mathsf{S}} \ni \exists y \leq \big(S(y), \oplus(x, \overline{1})\big)$ x가 free variable
- $\mathcal{L}_{\mathsf{S}} \ni \forall x\ \exists y \leq \big(S(y), \oplus(x, \overline{1})\big)$ 위 예시의 universal closure

So far...

이때까지 한 것:

Signature $\sigma = (\mathcal{F}, \mathcal{R}, \mathrm{ar})$ 넣어서 FOL Language $\mathcal{L} \subseteq \Sigma^*$ 만들기

So far...

이때까지 한 것:

Signature $\sigma = (\mathcal{F}, \mathcal{R}, \mathrm{ar})$ 넣어서 FOL Language $\mathcal{L} \subseteq \Sigma^*$ 만들기

이제부터 할 것:

- 각 "function symbol" $f \in \mathcal{F}$를 실제 함수로 대응시키고,
- 각 "relation symbol" $R \in \mathcal{R}$를 실제 relation로 대응시키고,
- 각 "문자열" $t \in \mathcal{L}_{\mathrm{Term}}$을 전체 공간의 원소로 대응시켜서,
- 각 "문자열" $A \in \mathcal{L}$을 진리치(True/False)로 대응시킬 예정

전체 공간 $= U$, 대응 $=$???

Interpretation Function $\mathcal{I}$

Interpretation function $\mathcal{I}$의 역할:

> 각 function/relation symbol들을
> specific한 function/relation으로 대응시킴

즉, 문자열의 세계을 수학 세상으로 해석함.

Interpretation Function $\mathcal{I}$

Interpretation function $\mathcal{I}$의 역할:

> 각 function/relation symbol들을
> specific한 function/relation으로 대응시킴

즉, 문자열의 세계을 수학 세상으로 해석함.

상식적으로 interpretation function $\mathcal{I}$는 다음을 만족해야 함:

- $\mathcal{I}(f): U^{\mathrm{ar}(f)} \to U$ for each $f \in \mathcal{F}$
- $\mathcal{I}(R) \subseteq U^{\mathrm{ar}(R)}$ for each $R \in \mathcal{R}$

Interpretation Function $\mathcal{I}$

Interpretation function $\mathcal{I}$의 역할:

> 각 function/relation symbol들을
> specific한 function/relation으로 대응시킴

즉, 문자열의 세계을 수학 세상으로 해석함.

상식적으로 interpretation function $\mathcal{I}$는 다음을 만족해야 함:

- $\mathcal{I}(f): U^{\mathrm{ar}(f)} \to U$ for each $f \in \mathcal{F}$
- $\mathcal{I}(R) \subseteq U^{\mathrm{ar}(R)}$ for each $R \in \mathcal{R}$

Definition: $\mathcal{L}$-structure $\mathcal{M} = (U, \mathcal{I})$

FOL language $\mathcal{L}$에 대해, 전체 공간 U와 interpretation function $\mathcal{I}$가 주어지면 $\mathcal{M} = (U, \mathcal{I})$를 $\mathcal{L}$-structure라고 함.

Semantics $v_{\mathcal{M}}$

Definition: Semantics $v_{\mathcal{M}}$

$v_{\mathcal{M}} : \mathcal{L}_{\text{Term}} \cup \mathcal{L} \to U \cup \{T, F\}$ is defined inductively by:

Semantics $v_{\mathcal{M}}$

Definition: Semantics $v_{\mathcal{M}}$

$v_{\mathcal{M}} : \mathcal{L}_{\text{Term}} \cup \mathcal{L} \to U \cup \{T, F\}$ is defined inductively by:

- $v_{\mathcal{M}}(f(t_1, \cdots, t_n)) = \mathcal{I}(f)(v_{\mathcal{M}}(t_1), \cdots, v_{\mathcal{M}}(t_n)) \in U$
- $v_{\mathcal{M}}(R(t_1, \cdots, t_n)) = \begin{cases} T & \text{if } (v_{\mathcal{M}}(t_1), \cdots, v_{\mathcal{M}}(t_n)) \in \mathcal{I}(R) \\ F & \text{otherwise} \end{cases}$

Semantics $v_{\mathcal{M}}$

Definition: Semantics $v_{\mathcal{M}}$

$v_{\mathcal{M}} : \mathcal{L}_{\text{Term}} \cup \mathcal{L} \to U \cup \{T, F\}$ is defined inductively by:

- $v_{\mathcal{M}}(f(t_1, \cdots, t_n)) = \mathcal{I}(f)(v_{\mathcal{M}}(t_1), \cdots, v_{\mathcal{M}}(t_n)) \in U$
- $v_{\mathcal{M}}(R(t_1, \cdots, t_n)) = \begin{cases} T & \text{if } (v_{\mathcal{M}}(t_1), \cdots, v_{\mathcal{M}}(t_n)) \in \mathcal{I}(R) \\ F & \text{otherwise} \end{cases}$
- $v_{\mathcal{M}}(A \wedge B) = \begin{cases} T & \text{if } v_{\mathcal{M}}(A) = v_{\mathcal{M}}(B) = T \\ F & \text{otherwise} \end{cases}$
- $v_{\mathcal{M}}(A \Rightarrow B) = v_{\mathcal{M}}(\neg A \vee B)$
- $v_{\mathcal{M}}(\forall x\ A) = \begin{cases} T & \text{if } v_{\mathcal{M}}(A[a/x]) = T \text{ for all } a \in U^a \\ F & \text{otherwise} \end{cases}$
- $v_{\mathcal{M}}(A \vee B)$, $v_{\mathcal{M}}(\neg A)$, $v_{\mathcal{M}}(\exists x\ A)$ 도 비슷하게 정의

---

[a] $A[a/x]$는 A의 free variable x에 a를 대입한 것

Model

$A \in \mathcal{L}_S$ and $S \subseteq \mathcal{L}_S$ ($\mathcal{L}$-sentences)

Definition: Truth

Given an $\mathcal{L}$-structure $\mathcal{M}$,

- If $v_{\mathcal{M}}(A) = T$, we write $\mathcal{M} \vDash A$. (A is true in $\mathcal{M}$.)
- If $\mathcal{M} \vDash A$ for all $A \in S$, we write $\mathcal{M} \vDash S$.

1. A의 진리치는 $\mathcal{M}$에 의존함!
2. 모든 $A, \mathcal{M}$에 대해 $\mathcal{M} \vDash A$거나 $\mathcal{M} \vDash \neg A$임

Model

$A \in \mathcal{L}_S$ and $S \subseteq \mathcal{L}_S$ ($\mathcal{L}$-sentences)

Definition: Truth

Given an $\mathcal{L}$-structure $\mathcal{M}$,

- If $v_{\mathcal{M}}(A) = T$, we write $\mathcal{M} \vDash A$. (A is true in $\mathcal{M}$.)
- If $\mathcal{M} \vDash A$ for all $A \in S$, we write $\mathcal{M} \vDash S$.

1. A의 진리치는 $\mathcal{M}$에 의존함!
2. 모든 $A, \mathcal{M}$에 대해 $\mathcal{M} \vDash A$거나 $\mathcal{M} \vDash \neg A$임

Definition: Model

Given an $\mathcal{L}$-structure $\mathcal{M}$, if $\mathcal{M} \vDash S$, we say $\mathcal{M}$ is a model of S.

S의 model $\mathcal{M}$은 모든 $A \in S$를 참으로 만드는 해석!

Model: Example

Define a FOL Language $\mathcal{L}_0 = (\mathcal{F}, \mathcal{R}, \mathrm{ar})$ by

- $\mathcal{F} = \{\overline{0}, \mathrm{next}, \mathrm{add}\}$
- $\mathcal{R} = \{=\}$.
- $\mathrm{ar}(\overline{0}) = 0,\ \mathrm{ar}(\mathrm{next}) = 1,\ \mathrm{ar}(\mathrm{add}) = \mathrm{ar}(=) = 2$

Let $A =$ "$\exists x\ (\mathrm{next}(x) = \mathrm{add}(\overline{0}, \overline{0}))$" be an $\mathcal{L}$-sentence.

Model: Example

Define a FOL Language $\mathcal{L}_0 = (\mathcal{F}, \mathcal{R}, \mathrm{ar})$ by

- $\mathcal{F} = \{\overline{0}, \mathrm{next}, \mathrm{add}\}$
- $\mathcal{R} = \{=\}$.
- $\mathrm{ar}(\overline{0}) = 0,\ \mathrm{ar}(\mathrm{next}) = 1,\ \mathrm{ar}(\mathrm{add}) = \mathrm{ar}(=) = 2$

Let $A =$ "$\exists x\ (\mathrm{next}(x) = \mathrm{add}(\overline{0}, \overline{0}))$" be an $\mathcal{L}$-sentence.

$\mathcal{L}$-structure $\mathcal{M}_1 = (U_1, \mathcal{I}_1)$

- $U_1 = \mathbb{Z}$
- $\mathcal{I}_1(\overline{0}) = 0$
- $\mathcal{I}_1(\mathrm{next})(x) = x + 1$ $+$: usual addition
- $\mathcal{I}_1(\mathrm{add})(x, y) = x + y$ $+$: usual addition
- $\mathcal{I}_1(=) = \{(x, x) \mid x \in \mathbb{Z}\}$

이 해석 아래에서 A는 참임. ($\mathcal{M}_1 \vDash A$) 즉, $\mathcal{M}_1$은 $\{A\}$의 model.

Model: Example

Define a FOL Language $\mathcal{L}_0 = (\mathcal{F}, \mathcal{R}, \mathrm{ar})$ by

- $\mathcal{F} = \{\overline{0}, \mathrm{next}, \mathrm{add}\}$
- $\mathcal{R} = \{=\}$.
- $\mathrm{ar}(\overline{0}) = 0,\ \mathrm{ar}(\mathrm{next}) = 1,\ \mathrm{ar}(\mathrm{add}) = \mathrm{ar}(=) = 2$

Let $A =$ "$\exists x\ (\mathrm{next}(x) = \mathrm{add}(\overline{0}, \overline{0}))$" be an $\mathcal{L}$-sentence.

$\mathcal{L}$-structure $\mathcal{M}_2 = (U_2, \mathcal{I}_2)$

- $U_2 = \mathbb{N}_0$
- $\mathcal{I}_2(\overline{0}) = 0$
- $\mathcal{I}_2(\mathrm{next})(x) = x + 1$ — $+$: usual addition
- $\mathcal{I}_2(\mathrm{add})(x, y) = x + y$ — $+$: usual addition
- $\mathcal{I}_2(=) = \{(x, x) \mid x \in \mathbb{Z}\}$

이 해석 아래에서 A는 거짓임. ($\mathcal{M}_2 \not\models A$) 즉, $\mathcal{M}_2$는 A의 model이 아님.

Logical Validity/Consequence

$A, B \in \mathcal{L}_S$ and $S \subseteq \mathcal{L}_S$ ($\mathcal{L}$-sentences)

Definition: Logical Consequence

- If $\mathcal{M} \vDash B \implies \mathcal{M} \vDash A$ for every $\mathcal{L}$-structure $\mathcal{M}$, we say A is a logical consequence of B and we write $B \vDash A$.
- If $\mathcal{M} \vDash S \implies \mathcal{M} \vDash A$ for every $\mathcal{L}$-structure $\mathcal{M}$, we say A is a logical consequence of S and we write $S \vDash A$.

Definition: Logical Validity

If $\mathcal{M} \vDash A$ for every $\mathcal{L}$-structure $\mathcal{M}$, we say A is logically valid and we write $\vDash A$.

Logical Validity/Consequence

$A, B \in \mathcal{L}_S$ and $S \subseteq \mathcal{L}_S$ ($\mathcal{L}$-sentences)

Definition: Logical Consequence

- If $\mathcal{M} \vDash B \Longrightarrow \mathcal{M} \vDash A$ for every $\mathcal{L}$-structure $\mathcal{M}$, we say A is a logical consequence of B and we write $B \vDash A$.
- If $\mathcal{M} \vDash S \Longrightarrow \mathcal{M} \vDash A$ for every $\mathcal{L}$-structure $\mathcal{M}$, we say A is a logical consequence of S and we write $S \vDash A$.

Definition: Logical Validity

If $\mathcal{M} \vDash A$ for every $\mathcal{L}$-structure $\mathcal{M}$, we say A is logically valid and we write $\vDash A$.

- $S \vDash A$의 의미: S의 model $\mathcal{M}$은 A를 true로 해석할 수밖에 없음. (e.g. $A = B \vee C$)
- $\vDash A$의 의미: 모든 $\mathcal{L}$-structure $\mathcal{M}$은 A의 model임. 즉, A는 syntax 단계에서부터 참인 해석만 허용. (e.g. $A = B \vee \neg B$)

So far...

이때까지 한 것:

- Signature $(\mathcal{F}, \mathcal{R}, \mathrm{ar})$ 넣어서 FOL Language $\mathcal{L}$ 만들기
- $\mathcal{L}$에 $\mathcal{L}$-structure $\mathcal{M} = (U, \mathcal{I})$으로 의미를 부여하기

So far...

이때까지 한 것:

- Signature $(\mathcal{F}, \mathcal{R}, \mathrm{ar})$ 넣어서 FOL Language $\mathcal{L}$ 만들기
- $\mathcal{L}$에 $\mathcal{L}$-structure $\mathcal{M} = (U, \mathcal{I})$으로 의미를 부여하기

이제부터 할 것:

- $S \vDash A$인지 확인할 방법 찾으려고 시도...
  - $S \vDash A$와 equivalent한 좋은 명제 없을까?

Sequent

들어가기에 앞서

Proof theory는 semantics와 전혀 관련이 없음. 둘의 연관성은 나중에 볼 예정.

Sequent

들어가기에 앞서

Proof theory는 semantics와 전혀 관련이 없음. 둘의 연관성은 나중에 볼 예정.

Definition: Sequent $\Gamma \vdash A$

A sequent is a pair (Γ, A) where $\Gamma \subseteq \mathcal{L}$ and $A \in \mathcal{L}$. (보통 $\Gamma \vdash A$로 씀.)

Γ는 공리(axiom), A는 결론(conclusion)이라고 보면 됨.

Proof on Sequent (Gentzen-style)

Definition: Proof on Sequents

A proof of a sequent $\Gamma \vdash A$ is a tree such that

- its root is the sequent $\Gamma \vdash A$ and
- its subtrees are proofs of sequents $\Gamma_1 \vdash A_1, \cdots, \Gamma_k \vdash A_k$ where

$$\frac{\Gamma_1 \vdash A_1 \qquad \Gamma_2 \vdash A_2 \qquad \cdots \qquad \Gamma_k \vdash A_k}{\Gamma \vdash A}$$

is a natural deduction.

If $\Gamma \vdash A$ has a proof, then we say $\Gamma \vdash A$ is provable.

Proof on Sequent (Gentzen-style)

Definition: Proof on Sequents

A proof of a sequent $\Gamma \vdash A$ is a tree such that

- its root is the sequent $\Gamma \vdash A$ and
- its subtrees are proofs of sequents $\Gamma_1 \vdash A_1, \cdots, \Gamma_k \vdash A_k$ where

$$\frac{\Gamma_1 \vdash A_1 \qquad \Gamma_2 \vdash A_2 \qquad \cdots \qquad \Gamma_k \vdash A_k}{\Gamma \vdash A}$$

is a natural deduction.

If $\Gamma \vdash A$ has a proof, then we say $\Gamma \vdash A$ is provable.

- Natural deduction은 가장 작은 단위의 reasoning
  - 뒤에서 볼 예정
- $\Gamma \vdash A$: 증명하고자 하는 것
- $\Gamma_1 \vdash A_1, \cdots, \Gamma_k \vdash A_k$: $\Gamma \vdash A$의 근거

Natural Deduction

얘네들이 natural deduction (겁먹을 필요 없음):

$$\frac{}{\Gamma \vdash A}\ Axiom \quad if\ A \in \Gamma$$

$$\frac{}{\Gamma \vdash \top}\ \top\text{-}intro$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \bot\text{-}elim$$

$$\frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\ \neg\text{-}intro$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash \neg A}{\Gamma \vdash \bot}\ \neg\text{-}elim$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\text{-}intro$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\text{-}elim$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\text{-}elim$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee\text{-}intro$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee\text{-}intro$$

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}\ \vee\text{-}elim$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow\text{-}intro$$

$$\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}\ \Rightarrow\text{-}elim$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall x\ A}\ \forall\text{-}intro \quad if\ x \notin FV(\Gamma)$$

$$\frac{\Gamma \vdash \forall x\ A}{\Gamma \vdash (t/x)A}\ \forall\text{-}elim$$

$$\frac{\Gamma \vdash (t/x)A}{\Gamma \vdash \exists x\ A}\ \exists\text{-}intro$$

$$\frac{\Gamma \vdash \exists x\ A \quad \Gamma, A \vdash B}{\Gamma \vdash B}\ \exists\text{-}elim \quad if\ x \notin FV(\Gamma, B)$$

$$\frac{}{\Gamma \vdash A \vee \neg A}\ Excluded\ middle$$

Natural Deduction

얘네들이 natural deduction (겁먹을 필요 없음):

$$\frac{}{\Gamma \vdash A}\ Axiom \quad if\ A \in \Gamma$$

$$\frac{}{\Gamma \vdash \top}\ \top\text{-}intro$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \bot\text{-}elim$$

$$\frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}\ \neg\text{-}intro$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash \neg A}{\Gamma \vdash \bot}\ \neg\text{-}elim$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\text{-}intro$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\text{-}elim$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\text{-}elim$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee\text{-}intro$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee\text{-}intro$$

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}\ \vee\text{-}elim$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow\text{-}intro$$

$$\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}\ \Rightarrow\text{-}elim$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall x\ A}\ \forall\text{-}intro \quad if\ x \notin FV(\Gamma)$$

$$\frac{\Gamma \vdash \forall x\ A}{\Gamma \vdash (t/x)A}\ \forall\text{-}elim$$

$$\frac{\Gamma \vdash (t/x)A}{\Gamma \vdash \exists x\ A}\ \exists\text{-}intro$$

$$\frac{\Gamma \vdash \exists x\ A \quad \Gamma, A \vdash B}{\Gamma \vdash B}\ \exists\text{-}elim \quad if\ x \notin FV(\Gamma, B)$$

$$\frac{}{\Gamma \vdash A \vee \neg A}\ Excluded\ middle$$

- 가로선 위에서부터 아래로 deduction이 일어남.
  - 위의 sequent들이 provable하면 아래도 provable
- 이름이 natural deduction인 만큼 하나씩 뜯어보면 당연함.
  - 직관을 글로 explicit하게 쓰려고 하니 벌어지는 참사...

Formal Proof: Example

$\Gamma, A \vdash C$는 $\Gamma \cup \{A\} \vdash C$의 shortcut

Proof of $\Gamma \vdash A \Rightarrow (B \Rightarrow (A \wedge B))$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, A, B \vdash A}\text{ Axiom} \quad \dfrac{}{\Gamma, A, B \vdash B}\text{ Axiom}}{\Gamma, A, B \vdash A \wedge B}\ \wedge\text{-intro}}{\Gamma, A \vdash B \Rightarrow (A \wedge B)}\ \Rightarrow\text{-intro}}{\Gamma \vdash A \Rightarrow (B \Rightarrow (A \wedge B))}\ \Rightarrow\text{-intro}$$

Reference:

$$\frac{}{\Gamma \vdash A}\ Axiom \quad if\ A \in \Gamma$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\text{-}intro$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow\text{-}intro$$

Soundness of FOL

Theorem (Soundness of FOL)

For every $\Gamma \subseteq \mathcal{L}_S$ and $A \in \mathcal{L}_S$,

$$\Gamma \vdash A \text{ is provable} \implies \Gamma \vDash A.$$

Proof

Natural induction을 이 theorem이 성립하도록 잡았기 때문... 엄밀한 증명은 정말 귀찮음. □

Recall: Logical Consequence

If $\mathcal{M} \vDash S \implies \mathcal{M} \vDash A$ for every $\mathcal{L}$-structure $\mathcal{M}$, we say A is a logical consequence of S and we write $S \vDash A$.

증명의 존재는 진리치가 참임을 함의함!

Completeness of FOL

Theorem (Completeness of FOL) [Gödel, 1929]

For every $\Gamma \subseteq \mathcal{L}_S$ and $A \in \mathcal{L}_S$,

$$\Gamma \vDash A \Longrightarrow \Gamma \vdash A \text{ is provable.}$$

Proof

Completeness of FOL

Theorem (Completeness of FOL) [Gödel, 1929]

For every $\Gamma \subseteq \mathcal{L}_S$ and $A \in \mathcal{L}_S$,

$$\Gamma \vDash A \Longrightarrow \Gamma \vdash A \text{ is provable.}$$

Proof

...를 여기서 하면 세미나 n시간 추가됨.

- 1929년 괴델의 증명보다 1949년 Henkin의 증명이 더 쉽고 짧음.

아니 Gödel의 First Incompleteness Theorem에 의하면 ZFC에 증명도 반증도 할 수 없는 명제가 있는데 어떻게 된 것?

Completeness of FOL

Theorem (Completeness of FOL) [Gödel, 1929]

For every $\Gamma \subseteq \mathcal{L}_S$ and $A \in \mathcal{L}_S$,

$$\Gamma \vDash A \Longrightarrow \Gamma \vdash A \text{ is provable.}$$

Proof

...를 여기서 하면 세미나 n시간 추가됨.

- 1929년 괴델의 증명보다 1949년 Henkin의 증명이 더 쉽고 짧음.

아니 Gödel의 First Incompleteness Theorem에 의하면 ZFC에 증명도 반증도 할 수 없는 명제가 있는데 어떻게 된 것?

- 위 theorem은 Γ의 모든 model이 A의 model이기도 하다는 것
- 증명도 반증도 할 수 없는 명제 A와 $\neg A$ 둘 다 Γ의 logical consequence가 아님!

Automatic Proof Verification

우리는 방금:

Proof를 computable object(tree)로 formalize함

Proof tree의 특징

1. 모든 Proof Tree의 set은 countable.
   - Natural deduction의 개수가 finite
   - 한 번에 늘어나는 subtree의 개수가 finite
   - 모든 proof tree의 height는 finite
2. 어느 proof tree가 valid한지 쉽게 확인 가능
   - finite 개수의 natural deduction => finite 횟수의 확인

Automatic Theorem Proving

Theorem에서 proof 찾기

$\Gamma \subseteq \mathcal{L}_S, A \in \mathcal{L}_S$

1. 모든 Proof Tree를 열거하면서:
2. 그 tree T 가 $\Gamma \vdash A$인지 확인
3. 맞으면 T가 그 증명. 아니면 계속 진행...

Automatic Theorem Proving

Theorem에서 proof 찾기

$\Gamma \subseteq \mathcal{L}_S, A \in \mathcal{L}_S$

1. 모든 Proof Tree를 열거하면서:
2. 그 tree T 가 $\Gamma \vdash A$인지 확인
3. 맞으면 T가 그 증명. 아니면 계속 진행...

모든 Theorem 찾기

1. 모든 $A \in \mathcal{L}_S$와 $\Gamma \vdash A$를 root로 가지는 모든 Proof Tree를 열거하면서:
2. 그 tree T 가 $\Gamma \vdash A$인지 확인
3. 맞으면 A는 theorem. 아니면 계속 진행...

Automatic Theorem Proving

Theorem에서 proof 찾기

$\Gamma \subseteq \mathcal{L}_S, A \in \mathcal{L}_S$

1. 모든 Proof Tree를 열거하면서:
2. 그 tree T 가 $\Gamma \vdash A$인지 확인
3. 맞으면 T가 그 증명. 아니면 계속 진행...

모든 Theorem 찾기

1. 모든 $A \in \mathcal{L}_S$와 $\Gamma \vdash A$를 root로 가지는 모든 Proof Tree를 열거하면서:
2. 그 tree T 가 $\Gamma \vdash A$인지 확인
3. 맞으면 A는 theorem. 아니면 계속 진행...

만약 이게 가능하다면, 수학자들이 실직하지 않는 이유는 단순히 computing power의 부족 때문...

Automatic Theorem Proving : Example

Coq

Computer software의 하나:

- Proof tree를 쉽게(?) 구성할 수 있도록 해 줌.
- 엄청난 표현력...
- 부분적 automated theorem proving
- Cryptography 분야에서는 Coq proof 증명을 요구하기도...

Coq Proof of $\sum_{i=1}^{n}(2n^3+3n^2+n)/6$

```
Lemma sum3: forall n:nat, 6*sum n = 2*n*n*n + 3*n*n + n.
Proof.
    induction n.
    simpl (sum 0). ring; simpl (sum (S n)).
    ring simplify; rewrite IHn.
    ring.
Qed.
```

So far...

이때까지 한 것:

- Signature $(\mathcal{F}, \mathcal{R}, \mathrm{ar})$ 넣어서 FOL Language $\mathcal{L}$ 만들기
- $\mathcal{L}$에 $\mathcal{L}$-structure $\mathcal{M} = (U, \mathcal{I})$으로 의미를 부여하기
- $\Gamma \vDash A \Leftrightarrow \Gamma \vdash A$ (Soundness/Completeness of FOL)

So far...

이때까지 한 것:

- Signature $(\mathcal{F}, \mathcal{R}, \mathrm{ar})$ 넣어서 FOL Language $\mathcal{L}$ 만들기
- $\mathcal{L}$에 $\mathcal{L}$-structure $\mathcal{M} = (U, \mathcal{I})$으로 의미를 부여하기
- $\Gamma \vDash A \Leftrightarrow \Gamma \vdash A$ (Soundness/Completeness of FOL)

이제부터 할 것:

- PA/ZFC에서 Godel's First Incompleteness Theorem 증명
  - 증명 과정에서 computability theory 향 첨가 예정

So far...

이때까지 한 것:

- Signature $(\mathcal{F}, \mathcal{R}, \mathrm{ar})$ 넣어서 FOL Language $\mathcal{L}$ 만들기
- $\mathcal{L}$에 $\mathcal{L}$-structure $\mathcal{M} = (U, \mathcal{I})$으로 의미를 부여하기
- $\Gamma \vDash A \Leftrightarrow \Gamma \vdash A$ (Soundness/Completeness of FOL)

이제부터 할 것:

- PA/ZFC에서 Godel's First Incompleteness Theorem 증명
  - 증명 과정에서 computability theory 향 첨가 예정

Disclaimer

앞에서도 그랬지만, 이 부분에서 모든 것을 rigorous하게 설명할 경우 세미나 n시간 추가 예정.

- 따라서 proof idea와 납득할 수 있을 정도의 근거만을 제시
- 그러니 양해를...

Computable Functions

Definition: Partial Function

A partial function $f : X \rightharpoonup Y$ is a function but $f(x)$ may not be defined.

- $f(x)$가 정의되면 $f(x)\downarrow$,
- $f(x)$가 정의되지 않으면 $f(x)\uparrow$로 나타냄

Computable Functions

Definition: Partial Function

A partial function $f : X \rightharpoonup Y$ is a function but $f(x)$ may not be defined.

- $f(x)$가 정의되면 $f(x)\downarrow$,
- $f(x)$가 정의되지 않으면 $f(x)\uparrow$로 나타냄

Definition: Computable Function

A partial function $f : \mathbb{N} \rightharpoonup \mathbb{N}$ is computable if there exists a `Python` program M_f such that

- $M(x) = f(x)$ if $f(x)\downarrow$.
- $M(x)$ never halts if $f(x)\uparrow$.

- 당연히 보통 Turing machine으로 정의하지만 어차피 equivalent
- Total function $f : \mathbb{N}^k \to \mathbb{N}$에만 computability를 정의하는 책도

Halting Problem

$\mathfrak{M}$을 모든 Python 프로그램의 집합이라 하자.

Theorem (Halting Problem)

The total function $f : \mathfrak{M} \times X \to \{0, 1\}$ defined by

$$f(M, x) = \begin{cases} 1 & \text{if } M(x)\downarrow \\ 0 & \text{if } M(x)\uparrow \end{cases}$$

is not computable.

Proof (in the seminar on November 6th, 2023)

Define $\alpha(0) = 1$ and $\alpha(1)\uparrow$. If f is computable, then $g(M) = \alpha(f(M, M))$ is also computable. Then,

$$g(g)\downarrow \iff f(g, g) = 0 \iff g(g)\uparrow.$$

Consistency

Definition: Consistency

$\Gamma \subseteq \mathcal{L}_S$ is consistent if there is no $A \in \mathcal{L}_S$ such that

- $\Gamma \vdash A$ and $\Gamma \vdash \neg A$.

Consistency

Definition: Consistency

$\Gamma \subseteq \mathcal{L}_S$ is consistent if there is no $A \in \mathcal{L}_S$ such that

- $\Gamma \vdash A$ and $\Gamma \vdash \neg A$.

만약 Γ가 inconsistent하면

- 모든 $A \in \mathcal{L}_S$에 대해 $\Gamma \vdash A$가 provable함.
- Γ의 model $\mathcal{M}$이 존재하지 않음.
- 이 뒤의 모든 내용이 의미없어짐.

그러니 이제부터 Γ의 consistency는 깔고 감.

Representability

Lemma (Representability of Computable Functions)

Γ가 자연수를 표현할 수 있으면, 모든 computable function $f : \mathbb{N} \rightharpoonup \mathbb{N}$에 대해,

- $f(x) = y \implies \Gamma \vdash A_f(x, y)$
- $f(x) \neq y \implies \Gamma \vdash \neg A_f(x, y)$

인 $A_f \in \mathcal{L}$이 존재한다.

Representability

Lemma (Representability of Computable Functions)

Γ가 자연수를 표현할 수 있으면, 모든 computable function $f : \mathbb{N} \rightharpoonup \mathbb{N}$에 대해,

- $f(x) = y \implies \Gamma \vdash A_f(x, y)$
- $f(x) \neq y \implies \Gamma \vdash \neg A_f(x, y)$

인 $A_f \in \mathcal{L}$이 존재한다.

Proof Idea

- Computable function의 여러 equivalent한 정의 중 recursion으로 정의된 함수를 computable로 정의하는 게 있음. (μ-recursive functions)
- Recursive → induction을 쓰기 좋음; induction으로 증명.

Representability

$+ : \mathbb{N}^2 \to \mathbb{N}$ is μ-recursive

- $g(x) = x$
- $h_1(x_1, x_2, x_3) = x_3$
- $h_2(x) = x + 1$

는 μ-recursive function의 base case들. 그러면

- $h(x_1, x_2, x_3) = h_1(h_2(x_1, x_2, x_3)) = x_3 + 1$

도 μ-recursive. 그러면

- $+(x, 0) = g(x) = x$
- $+(x, y + 1) = h(x, y, +(x, y)) = +(x, y) + 1$

로 정의된 $+$도 μ-recursive.

Representability

Lemma

The partial function $f_{\mathsf{Halt}} : \mathfrak{M} \times X \to \{0, 1\}$ defined by

$$f_{\mathsf{Halt}}(M, x) = \begin{cases} 1 & \text{if } M(x)\downarrow \\ 0 \text{ or undefined} & \text{if } M(x)\uparrow \end{cases}$$

is computable. (주의: 위 Halting problem의 함수와 다름)

Representability

Lemma

The partial function $f_{\text{Halt}} : \mathfrak{M} \times X \to \{0, 1\}$ defined by

$$f_{\text{Halt}}(M, x) = \begin{cases} 1 & \text{if } M(x)\downarrow \\ 0 \text{ or undefined} & \text{if } M(x)\uparrow \end{cases}$$

is computable. (주의: 위 Halting problem의 함수와 다름)

Proof.

Simulate the `Python` program and wait until it halts.

```
subprocess.run(['python', 'my_program.py'])
print(1)
```

This is the `Python` program computes f_{Halt}. □

Undecidability of PA/ZFC

Definition: Decidability

$\Gamma \subseteq \mathcal{L}_S$ is decidable if the total function $f : \mathcal{L}_S \to \{0, 1\}$

$$f(A) = \begin{cases} 1 & \text{if } \Gamma \vdash A \\ 0 & \text{if } \Gamma \nvdash A \end{cases}$$

is computable.

Undecidability of PA/ZFC

Theorem (Undecidability of ZA/ZFC)

자연수를 표현하는 Γ는 모두 undecidable하다.

Undecidability of PA/ZFC

Theorem (Undecidability of ZA/ZFC)

자연수를 표현하는 Γ는 모두 undecidable하다.

Proof.

- $f_{\text{Halt}}(M,x)=y \implies \Gamma \vdash A(M,x,y)$
- $f_{\text{Halt}}(M,x)\neq y \implies \Gamma \vdash \neg A(M,x,y)$

인 $A \in \mathcal{L}_S$가 존재함. Computable function f에 대해,

$$\phi_{f,x} \equiv \text{“}\exists y\, A(M_f,x,y)\text{”}$$

라고 정의. ($\phi_{f,x}$는 “$M_f(x)\downarrow$”의 의미.) Halting 여부와 equivalent한 문장을 구성했음!

만약 Γ가 decidable이면 $g:\mathcal{L}_S \to \{0,1\}$, $g(A)=1$ iff $\Gamma \vdash A$인 g가 computable. 따라서, $M_f(x)\downarrow \iff g(\phi_{f,x})=1$, halting problem의 undecidability와 모순!! □

Semidecidability of PA/ZFC

Definition: Semidecidability

$\Gamma \subseteq \mathcal{L}_S$ is semidecidable if the partial function $f : \mathcal{L}_S \to \{0, 1\}$

$$f(A) = \begin{cases} 1 & \text{if } \Gamma \vdash A \\ 0 \text{ or undefined} & \text{if } \Gamma \nvdash A \end{cases}$$

is computable.

Semidecidability of PA/ZFC

Definition: Semidecidability

$\Gamma \subseteq \mathcal{L}_S$ is semidecidable if the partial function $f : \mathcal{L}_S \to \{0, 1\}$

$$f(A) = \begin{cases} 1 & \text{if } \Gamma \vdash A \\ 0 \text{ or undefined} & \text{if } \Gamma \nvdash A \end{cases}$$

is computable.

Theorem (Semidecidability is Global)

Every Γ is semidecidable.

Proof.

Sequent $\Gamma \vdash A$의 proof는 tree형태. 모든 tree의 집합은 countable 하니 하나씩 열거하면서 tree T가 $\Gamma \vdash A$인지 확인. (확인하는 작업은 computable.) 이런 program은 위 f를 계산함. □

Incompleteness of PA/ZFC

Definition: Completeness

$\Gamma \subseteq \mathcal{L}_S$ is complete if

- it is either $\Gamma \vdash A$ or $\Gamma \vdash \neg A$

for each $A \in \mathcal{L}_S$.

Incompleteness of PA/ZFC

Definition: Completeness

$\Gamma \subseteq \mathcal{L}_S$ is complete if

- it is either $\Gamma \vdash A$ or $\Gamma \vdash \neg A$

for each $A \in \mathcal{L}_S$.

Theorem

Γ_{PA} and Γ_{ZFC} are incomplete.

Incompleteness of PA/ZFC

Definition: Completeness

$\Gamma \subseteq \mathcal{L}_S$ is complete if

- it is either $\Gamma \vdash A$ or $\Gamma \vdash \neg A$

for each $A \in \mathcal{L}_S$.

Theorem

Γ_{PA} and Γ_{ZFC} are incomplete.

Proof.

Completeness를 가정하고 다음 program을 생각하자.

- 모든 proof tree T를 열거하면서:
- T가 $\Gamma \vdash A$ 혹은 $\Gamma \vdash \neg A$의 proof인지 확인.

Γ가 complete함을 가정했으므로, 언젠가 $\Gamma \vdash A$ 혹은 $\Gamma \vdash \neg A$의 증명을 찾음. 따라서, Γ가 decidable이 됨. 모순! □

Gödel's First Incompleteness Theorem

Theorem (Gödel's First Incompleteness Theorem)

Let Γ be any set of axioms such that

- Γ is consistent
- Γ is semidecidable
- can represent all computable functions.

Then, Γ is incomplete.

Proof.

위와 크게 다르지 않음. $\square$

QNA

질문 받습니다.