逻辑、程序与形式化验证

Logic, Program and Formal Verification

李嘉图

清华大学交叉信息研究院

2021 年 2 月 5 日

主要内容 ~ Outline

自我介绍 ~ Self-introduction

1. NOI 2018 金牌，清华大学交叉信息研究院本科二年级
2. 参与命题：十二省联考 2019 D2T2（春节十二响），D2T3（希望）；NOI 2020 D1T2（命运），D2T2（超现实树）
3. 为什么要做这次报告：

自我介绍 ~ Self-introduction

1. NOI 2018 金牌，清华大学交叉信息研究院本科二年级
2. 参与命题：十二省联考 2019 D2T2（春节十二响），D2T3（希望）；NOI 2020 D1T2（命运），D2T2（超现实树）
3. 为什么要做这次报告：

NOI2020 D2T2 Surreal 主定理的一个形式化证明

2020-09-05 10:27:09 By ljt12138

最近试玩了一下一个叫做 Lean 的定理证明器，并尝试在里面证明了 Surreal 的主定理：一个树的集合是完备的当且仅当它能够生成每一棵高度为 h 的树，其中 h 为树的集合中包含的最高的树的高度。整个证明大概有 1k lines，花了大约 15 小时。

证明放在这里，欢迎大家来验证/基于里面的定理继续搞事

（正在考虑要不要冬令营来做一个关于定理证明器的 talk hh）

形式化证明 修改 删除 好评 差评 [+18]

逻辑 ~ Logic

什么是逻辑

研究对象	代表人物	例子
非形式逻辑	亚里士多德	人都会死；苏格拉底是人；因此苏格拉底会死
命题逻辑	乔治 · 布尔	$p \to q, p \models q$
谓词逻辑	戈特洛布 · 弗雷格	$\forall x(P(x) \to Q(x)), P(c) \models Q(c)$

逻辑学的研究还涉及：

1. 对算术系统的研究
2. 高阶逻辑
3. 模态逻辑

形式化验证 ~ Formal Verification

推理形式化（莱布尼茨）

“哲学家间的交流将变得同会计一样无需争辩；它们只需拿起笔和石板并对彼此说：让我们计算吧。”

形式化验证 ~ Formal Verification

推理形式化（莱布尼茨）

“哲学家间的交流将变得同会计一样无需争辩；它们只需拿起笔和石板并对彼此说：让我们计算吧。”

定理证明（Theorem Proving）

- 自动定理证明（Automated Theorem Proving）
- 交互式定理证明（Interactive Theorem Proving）
  - 基于类型系统的交互式定理证明
  - “证明助手”

程序 ~ Program

类型系统（Type System）

函数 $f(x) = x + 1$，不同的类型系统可能会给出：

1. `func` 类型，表示一个函数
2. `int` $\to$ `int` 类型，表示整数到整数的函数
3. $\forall\alpha[\texttt{hasone,hasadd}]\ \alpha \to \alpha$，任何定义了 1 和 + 的类型上的函数（多态）

一个观察

类型是对程序性质的抽象，正如命题是对证明（即推理过程）的抽象。

形式化验证的例子 ~ Examples of Formal Verification

形式化的数学证明

1. 四色定理的形式化证明 [Gon07]
2. 可计算性理论、停机问题 [Car18]
3. 连续统假设独立于 ZFC [HvD20]

形式化验证的例子 ~ Examples of Formal Verification

形式化的数学证明

1. 四色定理的形式化证明 [Gon07]
2. 可计算性理论、停机问题 [Car18]
3. 连续统假设独立于 ZFC [HvD20]

形式化验证的软件

1. 经过验证的 C 编译器 CompCert
2. 经过验证的操作系统内核 seL4

主题 ~ Topic

1. 非常基本的逻辑系统：命题逻辑
2. 非常基本的程序形式：简单类型 λ 演算
3. 柯里-霍华德对应：描述证明和程序的关系
4. 实现一个命题逻辑的交互式定理证明器

主要内容 ~ Outline

直观解释 ~ Intuition

例子

1. “如果 X2 第一题挂了，那么 C 一定能进省队”
2. “LCR 是存在的，并且我爱她”
3. “人被杀就会死，现在人被杀了，所以他死了”

直观解释 ~ Intuition

例子

1. “如果 X2 第一题挂了，那么 C 一定能进省队”
2. “LCR 是存在的，并且我爱她”
3. “人被杀就会死，现在人被杀了，所以他死了”

直观理解

- 描述原子命题之间的联系；
- 原子命题用连接词（与、或、非等）连接；
- 给出原子命题的真假，就能判定句子的真假。
- 推理：给出假设，推出结论。

语法 ~ Syntax

令 $\mathcal{A}$ 为原子命题的集合

定义

句子（sentence）按照以下方式归纳地定义。

- 原子命题 $p \in \mathcal{A}$ 是句子；
- 若 φ 和 ψ 是句子，那么

$$\neg\varphi, \quad \varphi \wedge \psi, \quad \varphi \vee \psi, \quad \varphi \to \psi$$

  都是句子，$\{\neg, \wedge, \vee, \to\}$ 称为连接词（connectives）。
- 仅由以上两条规则经有限步生成的是句子。

归纳定义 ~ Inductive Definition

归纳定义的例子

定义出现集合 $\mathcal{O}(\varphi)$ 如下：

1. $\mathcal{O}(p) := \{p\}$
2. $\mathcal{O}(\neg\varphi) := \mathcal{O}(\varphi)$
3. $\mathcal{O}(\varphi \wedge \psi) := \mathcal{O}(\varphi) \cup \mathcal{O}(\psi)$
4. $\mathcal{O}(\varphi \vee \psi) := \mathcal{O}(\varphi) \cup \mathcal{O}(\psi)$
5. $\mathcal{O}(\varphi \to \psi) := \mathcal{O}(\varphi) \cup \mathcal{O}(\psi)$

语义 ~ Semantics

模型与取值

任一函数 $M: \mathcal{A} \to \{0,1\}$ 称为一个模型（Model）。给定一个模型 M，一个句子 φ 的取值 $M^*(\varphi)$ 归纳定义为：

$$
\begin{aligned}
p.&\quad M^*(\varphi) := M(p);\\
\neg\varphi_1.&\quad M^*(\varphi) := 1 \text{ 当且仅当 } M^*(\varphi_1) = 0;\\
\varphi_1 \wedge \varphi_2.&\quad M^*(\varphi) := 1 \text{ 当且仅当 } M^*(\varphi_1) = 1 \text{ 且 } M^*(\varphi_2) = 1;\\
\varphi_1 \vee \varphi_2.&\quad M^*(\varphi) := 1 \text{ 当且仅当 } M^*(\varphi_1) = 1 \text{ 或 } M^*(\varphi_2) = 1;\\
\varphi_1 \to \varphi_2.&\quad M^*(\varphi) := 0 \text{ 当且仅当 } M^*(\varphi_1) = 1 \text{ 且 } M^*(\varphi_2) = 0。
\end{aligned}
$$

记号：$M \models \varphi$ 如果 $M^*(\varphi) = 1$

基本概念 ~ Basic Concepts

定义

1. 重言式（Tautology）：在所有模型中都为真
2. 矛盾式（Contradictory）：在所有模型中都为假
3. 可满足（Satisfiable）：存在一个模型中为真
4. 句子集合可满足：存在一个模型使得每个句子都为真

真值表 ~ Truth Table

对于句子 φ，$\mathcal{O}(\varphi) = \{x_1, x_2, \ldots, x_n\}$，定义其真值表（Truth Table）为所有模型下的取值。

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

表: $p \wedge q$ 的真值表.

- 真值表表示了一个 n 变量布尔函数 $f\colon \{0,1\}^n \to \{0,1\}$;
- 真值表行数、列数、所有可能的真值表个数？
- 重言式、矛盾式和可满足式的真值表形态？

连接词完备集 ~ Complete Set of Connectives

定理 ($\{\neg, \wedge, \vee\}$ 是连接词完备集)

任一布尔函数 $f\colon \{0,1\}^n \to \{0,1\}$ 都是某个仅包含 $\{\neg, \vee, \wedge\}$ 的句子的真值表。

连接词完备集 ~ Complete Set of Connectives

定理 ($\{\neg, \wedge, \vee\}$ 是连接词完备集)

任一布尔函数 $f\colon \{0,1\}^n \to \{0,1\}$ 都是某个仅包含 $\{\neg, \vee, \wedge\}$ 的句子的真值表。

p	q	$p \wedge q$	
0	0	1	$\leftarrow$ - - - - $\varphi_{00} = \neg p \wedge \neg q$
0	1	0	
1	0	0	
1	1	1	$\leftarrow$ - - - - $\varphi_{11} = p \wedge q$

图: $\varphi = (\neg p \wedge \neg q) \vee (p \wedge q)$

连接词完备集 ~ Complete Set of Connectives

定理 ($\{\neg, \wedge, \vee\}$ 是连接词完备集)

任一布尔函数 $f\colon \{0,1\}^n \to \{0,1\}$ 都是某个仅包含 $\{\neg, \vee, \wedge\}$ 的句子的真值表。

证明.

令原子命题为 $x_1, x_2, \ldots, x_n$。

- 对于任何 $a = (a_1, a_2, \ldots, a_n)$，定义 φ_a 使得 φ_a 仅在 $M(x_i) = a_i$ 这一模型下为真。
- $$\varphi := \bigvee_{a:f(a)=1} \varphi_a.$$
- 析取范式（Disjunctive Normal Form）

□

连接词完备集 ~ Complete Set of Connectives

定理 ($\{\neg, \wedge, \vee\}$ 是连接词完备集)

任一布尔函数 $f\colon \{0,1\}^n \to \{0,1\}$ 都是某个仅包含 $\{\neg, \vee, \wedge\}$ 的句子的真值表。

p	q	$p \wedge q$	
0	0	1	
0	1	0	$\leftarrow$ - - - - $\overline{\varphi}_{01} = p \vee \neg q$
1	0	0	$\leftarrow$ - - - - $\overline{\varphi}_{10} = \neg p \vee q$
1	1	1	

图: $\varphi = (p \vee \neg q) \wedge (\neg p \vee q)$

连接词完备集 ~ Complete Set of Connectives

定理 ($\{\neg, \wedge, \vee\}$ 是连接词完备集)

任一布尔函数 $f\colon \{0,1\}^n \to \{0,1\}$ 都是某个仅包含 $\{\neg, \vee, \wedge\}$ 的句子的真值表。

证明.

令原子命题为 $x_1, x_2, \ldots, x_n$。

- 对于任何 $a = (a_1, a_2, \ldots, a_n)$，定义 $\overline{\varphi}_a$ 使得 $\overline{\varphi}_a$ 仅在 $M(x_i) = a_i$ 这一模型下为假。
- $$\varphi := \bigwedge_{a:f(a)=0} \overline{\varphi}_a.$$
- 合取范式（Conjunctive Normal Form）

□

逻辑等价 ~ Logical Equivalence

定义（逻辑等价，Logical Equivalence）

称 φ 和 ψ 等价 $(\varphi \equiv \psi)$，如果它们在任何模型的取值都相等（具有相同的真值表）。

1. $\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$
2. $\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$
3. $\varphi \to \psi \equiv \neg\varphi \vee \psi$
4. $\neg\neg\varphi \equiv \varphi$
5. $\varphi \wedge (\psi \vee \gamma) \equiv (\varphi \wedge \psi) \vee (\varphi \wedge \gamma)$
6. $\varphi \vee (\psi \wedge \gamma) \equiv (\varphi \vee \psi) \wedge (\varphi \vee \gamma)$

逻辑推论 ~ Logical Consequence

定义

令 Γ 为一句子的集合，φ 为一句子，称 φ 是 Γ 的逻辑推论，记作 $\Gamma \models \varphi$，如果 Γ 成立的每一个模型中 φ 都成立。等价地，$\Gamma \cup \{\neg\varphi\}$ 是不可满足的。

1. $\varphi, \varphi \to \psi \models \psi$
2. $\varphi \to \psi, \psi \to \gamma \models \varphi \to \gamma$
3. 如果 $\Gamma, \varphi \models \psi$，那么 $\Gamma \models \varphi \to \psi$

证明系统 ~ Proof System

“逻辑推理”

给定一个 Γ 和 φ，是否有一形式化的方法论证 $\Gamma \models \varphi$？

- 令 S 为一个形式系统，S 中可定义关系 $\Gamma \vdash_S \varphi$。
- 可靠的（Soundness）：如果 $\Gamma \vdash_S \varphi$，那么 $\Gamma \models \varphi$
- 完备性（Completeness）：如果 $\Gamma \models \varphi$，那么 $\Gamma \vdash_S \varphi$

在无歧义的前提下，用 $\Gamma \vdash \varphi$ 代替 $\Gamma \vdash_S \varphi$。

简化语法 ~ Simplification of Syntax

为了证明系统表述的方便，我们简化命题逻辑的语法如下：

$$\varphi := p \mid \bot \mid \varphi_1 \to \varphi_2.$$

1. $\bot$ 表示矛盾，在任何模型 M 中都有 $M(\bot) = 0$；
2. $\neg\varphi$ 用 $\varphi \to \bot$ 定义；
3. $\{\neg, \to\}$ 是一个连接词完备集

$$\varphi \vee \psi \equiv \neg\varphi \to \psi$$
$$\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$$

4. $\to$ 是右结合的：$\varphi \to \psi \to \gamma := \varphi \to (\psi \to \gamma)$.

希尔伯特系统 ~ Hilbert-style System

定义（希尔伯特系统）

证明由若干行构成，每一行要么是假设 $\varphi \in \Gamma$，要么是公理，要么由证明规则得到。

公理模式 1 $\varphi \to \psi \to \varphi$

公理模式 2 $(\varphi \to \psi \to \gamma) \to (\varphi \to \psi) \to \varphi \to \gamma$

公理模式 3 $\neg\neg\varphi \to \varphi$

规则 1 (Modus Ponens) $\dfrac{\Gamma \vdash \varphi \to \psi \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$

希尔伯特系统 ~ Hilbert-style System

定义（希尔伯特系统）

证明由若干行构成，每一行要么是假设 $\varphi \in \Gamma$，要么是公理，要么由证明规则得到。

公理模式 1 $\varphi \to \psi \to \varphi$

公理模式 2 $(\varphi \to \psi \to \gamma) \to (\varphi \to \psi) \to \varphi \to \gamma$

公理模式 3 $\neg\neg\varphi \to \varphi$

规则 1 (Modus Ponens) $\dfrac{\Gamma \vdash \varphi \to \psi \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$

定理

希尔伯特系统是可靠的。

例子 ~ Examples

例 1: $\psi \vdash \varphi \to \psi$

行	公式	模式
1	ψ	假设
2	$\psi \to \varphi \to \psi$	公理 1
3	$\varphi \to \psi$	MP 1,2

表: $\psi \vdash \varphi \to \psi$

例子 ~ Examples

例 2: $\vdash \varphi \to \varphi$

行	公式	模式
1	$\varphi \to (\varphi \to \varphi) \to \varphi$	公理 1
2	$(\varphi \to (\varphi \to \varphi) \to \varphi) \to (\varphi \to \varphi \to \varphi) \to \varphi \to \varphi$	公理 2
3	$(\varphi \to \varphi \to \varphi) \to \varphi \to \varphi$	MP 1,2
4	$\varphi \to \varphi \to \varphi$	公理 1
5	$\varphi \to \varphi$	MP 3,4

表: $\vdash \varphi \to \varphi$

演绎定理 ~ Deduction Theorem

演绎定理（Deduction Theorem）

$\Gamma \vdash \varphi \to \psi$ 当且仅当 $\Gamma, \varphi \vdash \psi$。

演绎定理 ~ Deduction Theorem

演绎定理（Deduction Theorem）

$\Gamma \vdash \varphi \to \psi$ 当且仅当 $\Gamma, \varphi \vdash \psi$。

证明.

($\Rightarrow$) 如果 $\Gamma \vdash \varphi \to \psi$，下面构造 $\Gamma, \varphi \vdash \psi$。

行	公式	模式
1	$\varphi \to \psi$	条件
2	φ	假设
3	ψ	MP 1,2

表: $\Gamma, \varphi \vdash \psi$

演绎定理 ~ Deduction Theorem

演绎定理（Deduction Theorem）

$\Gamma \vdash \varphi \to \psi$ 当且仅当 $\Gamma, \varphi \vdash \psi$。

证明.

($\Leftarrow$) 将 $\Gamma, \varphi \vdash \psi$ 的证明逐行翻译为 $\Gamma \vdash \varphi \to \psi$ 的证明。

- 施归纳于 i：如果前者第 i 行证明了 γ_i，则 $\Gamma \vdash \varphi \to \gamma$。

1. $\gamma_i \in \Gamma$：则 $\Gamma \vdash \gamma_i$，由例 1 有 $\Gamma \vdash \varphi \to \gamma_i$；
2. $\gamma_i = \varphi$：由例 2 有 $\Gamma \vdash \varphi \to \varphi$；
3. 如果 γ_i 是公理，显然也有 $\Gamma \vdash \varphi \to \gamma_i$

演绎定理 ~ Deduction Theorem

演绎定理（Deduction Theorem）

$\Gamma \vdash \varphi \to \psi$ 当且仅当 $\Gamma, \varphi \vdash \psi$。

证明.

4. 如果 γ_i 是由 MP 得到，那么存在 $j, k < i$ 和 σ 使得
   - $\gamma_j = \sigma \to \gamma_i, \gamma_k = \sigma$

行	公式	模式
1	$\varphi \to \sigma \to \gamma_i$	归纳假设
2	$\varphi \to \sigma$	归纳假设
3	$(\varphi \to \sigma \to \gamma_i) \to (\varphi \to \sigma) \to \varphi \to \gamma_i$	公理 2
4	$\varphi \to \gamma_i$	两次使用 MP

表: $\Gamma \vdash \varphi \to \gamma_i$

演绎定理的应用 ~ Application of Deduction Theorem

应用：三段论

如果 $\Gamma \vdash \varphi \to \psi$，$\Gamma \vdash \psi \to \gamma$，那么 $\Gamma \vdash \varphi \to \gamma$。

证明.

根据演绎定理，只需证明 $\Gamma, \varphi \vdash \gamma$。

演绎定理的应用 ~ Application of Deduction Theorem

应用：三段论

如果 $\Gamma \vdash \varphi \to \psi$，$\Gamma \vdash \psi \to \gamma$，那么 $\Gamma \vdash \varphi \to \gamma$。

证明.

根据演绎定理，只需证明 $\Gamma, \varphi \vdash \gamma$。

行	公式	模式
1	$\varphi \to \psi$	$\Gamma \vdash \varphi \to \psi$
2	φ	假设
3	ψ	MP 1,2
4	$\psi \to \gamma$	$\Gamma \vdash \psi \to \gamma$
5	γ	MP 4,3

表: $\Gamma, \varphi \vdash \gamma$

反证法 ~ Proof By Contradiction

反证法（Proof By Contradiction）

如果 $\Gamma, \neg\varphi \vdash \bot$，那么 $\Gamma \vdash \varphi$。

证明.

根据演绎定理，有 $\Gamma \vdash \neg\varphi \to \bot$（$\neg\varphi \to \bot = \neg\neg\varphi$）。

反证法 ~ Proof By Contradiction

反证法（Proof By Contradiction）

如果 $\Gamma, \neg\varphi \vdash \bot$，那么 $\Gamma \vdash \varphi$。

证明.

根据演绎定理，有 $\Gamma \vdash \neg\varphi \to \bot$（$\neg\varphi \to \bot = \neg\neg\varphi$）。

行	公式	模式
1	$\neg\neg\varphi$	条件
2	$\neg\neg\varphi \to \varphi$	公理 3
3	φ	MP 2,1

表: $\Gamma \vdash \varphi$

□

完备性定理 ~ Completeness Theorem

模型存在定理（Model Existence Theorem）

如果 $\Gamma \nvdash \bot$，那么 Γ 有一个模型。

完备性定理 ~ Completeness Theorem

模型存在定理（Model Existence Theorem）

如果 $\Gamma \nvdash \bot$，那么 Γ 有一个模型。

完备性定理（Completeness Theorem）

如果 $\Gamma \models \varphi$，那么 $\Gamma \vdash \varphi$；

完备性定理 ~ Completeness Theorem

模型存在定理（Model Existence Theorem）

如果 $\Gamma \nvdash \bot$，那么 Γ 有一个模型。

完备性定理（Completeness Theorem）

如果 $\Gamma \models \varphi$，那么 $\Gamma \vdash \varphi$；

证明.

用反证法：如果 $\Gamma \nvdash \varphi$，那么 $\Gamma, \neg\varphi \nvdash \bot$。根据模型存在定理，$\Gamma \cup \{\neg\varphi\}$ 有一个模型，这意味着 $\Gamma \not\models \varphi$，矛盾。 □

模型存在定理 ~ Model Existence Theorem

模型存在定理（Model Existence Theorem）

如果 Γ 是无矛盾的（$\Gamma \nvdash \bot$），那么它是可满足的（存在模型）。

证明概要.

1. 设所有的句子为 $\varphi_1, \varphi_2, \ldots$，$\Gamma_0 = \Gamma$
2. 容易说明：$\Gamma_{i-1} \cup \{\varphi_i\}$ 和 $\Gamma_{i-1} \cup \{\neg\varphi_i\}$ 至少有一个是无矛盾的。令 Γ_i 为其中任一无矛盾者。
3. $\Gamma = \Gamma_0 \subseteq \Gamma_1 \subseteq \Gamma_2 \subseteq \ldots$ 都是无矛盾的
4. 定义模型 $M_{\Gamma^*}(p) = 1$ 当且仅当存在 i 使得 $p \in \Gamma_i$。
5. 容易验证 M_{Γ^*} 是 Γ 的模型。

□

小结 ~ Summary

1. 命题逻辑的语法和语义
2. 真值表、连接词完备集
3. 逻辑等价与逻辑推论
4. 希尔伯特证明系统
5. 演绎定理、完备性定理

主要内容 ~ Outline

自然演绎系统 ~ Natural Deduction System

直观解释

自然演绎系统的证明是一棵树 T

- 每个结点标有一个句子
- 根节点为证明的命题，叶节点为证明的假设
- 证明树从叶节点向上按规则构造
- 一些内部结点标有子树中某些叶节点的编号，表明它们在此处被“关闭”

令 $\mathcal{H}(T)$ 为所有未被关闭的假设，φ 为根节点，T 构成证明 $\mathcal{H}(T) \vdash \varphi$。

自然演绎系统-续 ~ Natural Deduction System, Cont'd

令 $\mathcal{H}(T)$ 为所有未被关闭的假设，φ 为根节点，T 构成证明 $\mathcal{H}(T) \vdash \varphi$。

定义

一棵证明树 T 和其开放假设集合 $\mathcal{H}(T)$ 归纳定义如下：

1. 句子 φ 构成一棵树 $[\varphi]$，定义 $\mathcal{H}([\varphi]) = \{\varphi\}$
2. 如果有以 $\varphi \to \psi$ 为根的 T_1，以 φ 为根的 T_2，那么可以构造一棵以 ψ 为根的树 $T = \dfrac{\begin{matrix} T_1 & T_2 \end{matrix}}{\psi}$，定义 $\mathcal{H}(T) = \mathcal{H}(T_1) \cup \mathcal{H}(T_2)$

$$(\to \mathrm{E}) \dfrac{\begin{matrix} \vdots & \vdots \\ \varphi \to \psi & \varphi \end{matrix}}{\psi}$$

自然演绎系统-续 ~ Natural Deduction System, Cont'd

令 $\mathcal{H}(T)$ 为所有未被关闭的假设，φ 为根节点，T 构成证明 $\mathcal{H}(T) \vdash \varphi$。

定义

一棵证明树 T 和其开放假设集合 $\mathcal{H}(T)$ 归纳定义如下：

3. 如果有以 ψ 为根的树 T_1，可以构造以 $\varphi \to \psi$ 为根的 $T = \frac{T_1}{\varphi \to \psi}$，定义 $\mathcal{H}(T) = \mathcal{H}(T_1) - \{\varphi\}$

$$(\to \mathrm{I})\frac{\begin{array}{c}[\varphi]^i \\ \vdots \\ \psi\end{array}}{\varphi \to \psi^i}$$

自然演绎系统-续 ~ Natural Deduction System, Cont'd

令 $\mathcal{H}(T)$ 为所有未被关闭的假设，φ 为根节点，T 构成证明 $\mathcal{H}(T) \vdash \varphi$。

定义

一棵证明树 T 和其开放假设集合 $\mathcal{H}(T)$ 归纳定义如下：

4. 如果有以 $\bot$ 为根的树 T_1，可以构造以 φ 为根的树 $T = \frac{T_1}{\varphi}$，$\mathcal{H}(T) = \mathcal{H}(T_1) - \{\neg\varphi\}$。

$$(\text{PBC})\frac{\begin{array}{c}[\neg\varphi]^i \\ \vdots \\ \bot\end{array}}{\varphi^i}$$

自然演绎系统-续 ~ Natural Deduction System, Cont'd

一棵证明树 T 称为 $\Gamma \vdash \varphi$ 的证明，如果

1. T 以 φ 为根
2. $\mathcal{H}(T) \subseteq \Gamma$

自然演绎系统-续 ~ Natural Deduction System, Cont'd

一棵证明树 T 称为 $\Gamma \vdash \varphi$ 的证明，如果

1. T 以 φ 为根
2. $\mathcal{H}(T) \subseteq \Gamma$

$$\dfrac{\dfrac{\dfrac{\dfrac{[\varphi \to \psi \to \gamma]_1 \quad [\varphi]_3}{\psi \to \gamma} \quad \dfrac{[\varphi \to \psi]_2 \quad [\varphi]_3}{\psi}}{\gamma}}{\varphi \to \gamma^3}}{(\varphi \to \psi) \to \varphi \to \gamma^2}}{(\varphi \to \psi \to \gamma) \to (\varphi \to \psi) \to \varphi \to \gamma^1}$$

可靠性 ~ Soundness

定理：自然演绎系统是可靠的

如果 $\Gamma \vdash \varphi$ 存在一棵证明树 T，那么 $\Gamma \models \varphi$

可靠性 ~ Soundness

定理：自然演绎系统是可靠的

如果 $\Gamma \vdash \varphi$ 存在一棵证明树 T，那么 $\Gamma \models \varphi$

证明.

施归纳于树 T 的高度，证明如果存在一棵以 φ 为根的树 T，那么 $\mathcal{H}(T) \models \varphi$。

$$(\mathrm{HYP})[\varphi] \quad (\to \mathrm{E})\frac{\begin{array}{cc}\vdots & \vdots \\ \varphi \to \psi & \varphi\end{array}}{\psi} \quad (\to \mathrm{I})\frac{\begin{array}{c}[\varphi]^i \\ \vdots \\ \psi\end{array}}{\varphi \to \psi^i} \quad (\mathrm{PBC})\frac{\begin{array}{c}[\neg\varphi]^i \\ \vdots \\ \bot\end{array}}{\varphi^i}$$

□

完备性 ~ Completeness

定理：自然演绎系统是完备的

如果 $\Gamma \models \varphi$，那么 $\Gamma \vdash \varphi$ 存在一棵证明树 T。

完备性 ~ Completeness

定理：自然演绎系统是完备的

如果 $\Gamma \models \varphi$，那么 $\Gamma \vdash \varphi$ 存在一棵证明树 T。

证明.

证明任一希尔伯特系统可证的命题在自然演绎系统也是可证的。

- 施归纳于证明的行数
- 验证公理是可证明的
- 说明如何考虑 MP

□

交互式证明协议 ~ Interactive Proof Protocol

定义

Prover 向 Verifier（一个算法 $\mathcal{A}$）证明 $\Gamma \models \varphi$

1. 进行若干轮通信：Prover 发送 x_1，Verifier 发送 y_1，Prover 发送 x_2，等等
2. Verifier 在某个时刻选择接受或拒绝

协议称为

- 可靠的（Sound）：如果 Verifier 接受了，$\Gamma \models \varphi$
- 完备的（Complete）：对于任何 $\Gamma \models \varphi$，都存在一个 Prover 使得 Verifier 接受

一个“自然的”协议 ~ A Natural Protocol

Verifier $\mathcal{A}$ 要求 Prover 从根出发 DFS 构建证明树。
证明系统的状态是目标 φ 和假设集合 Γ，有证明策略：

1. Assumption：如果 $\varphi \in \Gamma$ 则接受，否则拒绝；
2. Apply ψ：递归地证明 $\Gamma \vdash \psi \to \varphi$ 和 $\Gamma \vdash \psi$，两者均接受时接受，否则拒绝；
3. Intro：若 $\varphi \neq \psi \to \gamma$，拒绝；否则递归证明 $\Gamma \cup \{\psi\} \vdash \gamma$；
4. Contra：递归地证明 $\Gamma \cup \{\neg\varphi\} \vdash \bot$；

一个“自然的”协议 ~ A Natural Protocol

Verifier $\mathcal{A}$ 要求 Prover 从根出发 DFS 构建证明树。
证明系统的状态是目标 φ 和假设集合 Γ，有证明策略：

1. Assumption：如果 $\varphi \in \Gamma$ 则接受，否则拒绝；
2. Apply ψ：递归地证明 $\Gamma \vdash \psi \to \varphi$ 和 $\Gamma \vdash \psi$，两者均接受时接受，否则拒绝；
3. Intro：若 $\varphi \neq \psi \to \gamma$，拒绝；否则递归证明 $\Gamma \cup \{\psi\} \vdash \gamma$；
4. Contra：递归地证明 $\Gamma \cup \{\neg\varphi\} \vdash \bot$；

定理

存在 Prover 可证明 $\Gamma \vdash \varphi$ 当且仅当其存在一棵证明树。因此，这一协议是可靠且完备的。

改进的协议 ~ Improved Protocol

1. Assumption：如果 $\varphi \in \Gamma$ 则接受，否则拒绝；
2. Apply $\psi_1, \psi_2, \ldots, \psi_n$：如果

$$\psi_1 \to \psi_2 \to \cdots \to \psi_n \to \varphi \notin \Gamma$$

   则拒绝；否则递归地证明 $\Gamma \vdash \psi_i$，对于 $i = 1, 2, \ldots, n$；
3. Intro：若 $\varphi \neq \psi \to \gamma$，拒绝；否则递归证明 $\Gamma \cup \{\psi\} \vdash \gamma$；
4. Contra：递归地证明 $\Gamma \cup \{\neg\varphi\} \vdash \bot$；

改进的协议 ~ Improved Protocol

1. Assumption：如果 $\varphi \in \Gamma$ 则接受，否则拒绝；
2. Apply $\psi_1, \psi_2, \ldots, \psi_n$：如果

$$\psi_1 \to \psi_2 \to \cdots \to \psi_n \to \varphi \notin \Gamma$$

   则拒绝；否则递归地证明 $\Gamma \vdash \psi_i$，对于 $i = 1, 2, \ldots, n$；
3. Intro：若 $\varphi \neq \psi \to \gamma$，拒绝；否则递归证明 $\Gamma \cup \{\psi\} \vdash \gamma$；
4. Contra：递归地证明 $\Gamma \cup \{\neg\varphi\} \vdash \bot$；

定理：“无需灵感”

以上协议是完备的。

小结 ~ Summary

1. 自然演绎系统的定义，证明树
2. 可靠性和完备性
3. 基于自然演绎的交互式证明协议

主要内容 ~ Outline

λ 演算 $\sim$ λ Calculus

语法：$e := \lambda x.e \mid e_1\ e_2 \mid x$

1. 变量（Variable）：x（为方便，不考虑重名问题）
2. 抽象（Abstraction）：$\lambda x.e$，可理解为定义函数 $f(x) = e$；
3. 应用（Application）：$e_1\ e_2$，可理解为函数 e_1 取值 e_2；

λ 演算 $\sim$ λ Calculus

语法：$e := \lambda x.e \mid e_1\ e_2 \mid x$

1. 变量（Variable）：x（为方便，不考虑重名问题）
2. 抽象（Abstraction）：$\lambda x.e$，可理解为定义函数 $f(x) = e$；
3. 应用（Application）：$e_1\ e_2$，可理解为函数 e_1 取值 e_2；
4. 自由变量 $\mathrm{FV}(*)$，定义为

$$
\begin{aligned}
\mathrm{FV}(x) &:= x \\
\mathrm{FV}(\lambda x.e) &:= \mathrm{FV}(e) - \{x\} \\
\mathrm{FV}(e_1\ e_2) &:= \mathrm{FV}(e_1) \cup \mathrm{FV}(e_2)
\end{aligned}
$$

λ 演算 $\sim$ λ Calculus

语法：$e := \lambda x.e \mid e_1\ e_2 \mid x$

1. 变量（Variable）：x（为方便，不考虑重名问题）
2. 抽象（Abstraction）：$\lambda x.e$，可理解为定义函数 $f(x) = e$；
3. 应用（Application）：$e_1\ e_2$，可理解为函数 e_1 取值 e_2；
4. 自由变量 $\mathrm{FV}(*)$，定义为

$$\begin{aligned}\mathrm{FV}(x) &:= x\\ \mathrm{FV}(\lambda x.e) &:= \mathrm{FV}(e) - \{x\}\\ \mathrm{FV}(e_1\ e_2) &:= \mathrm{FV}(e_1) \cup \mathrm{FV}(e_2)\end{aligned}$$

5. 结合性：抽象向右延伸，应用左结合

$$\lambda x.\lambda y.z\ x\ y := \lambda x.(\lambda y.(z\ x)\ y).$$

替换和规约 ~ Substitution and Reduction

替换（Substitution）

令 e 是一个项，定义 $e[x \mapsto t]$ 为

1. $x[x \mapsto t] := t,\ y[x \mapsto t] := y$;
2. $(\lambda x.e)[x \mapsto t] := \lambda x.e,\ (\lambda y.e)[x \mapsto t] := \lambda y.e[x \mapsto t]$;
3. $(e_1\ e_2)[x \mapsto t] := (e_1[x \mapsto t]\ e_2[x \mapsto t])$

替换和规约 ~ Substitution and Reduction

替换（Substitution）

令 e 是一个项，定义 $e[x \mapsto t]$ 为

1. $x[x \mapsto t] := t,\ \ y[x \mapsto t] := y$;
2. $(\lambda x.e)[x \mapsto t] := \lambda x.e,\ \ (\lambda y.e)[x \mapsto t] := \lambda y.e[x \mapsto t]$;
3. $(e_1\ e_2)[x \mapsto t] := (e_1[x \mapsto t]\ e_2[x \mapsto t])$

β-规约（β-reduction）

$$(\lambda x.e_1)\ e_2 \to_\beta e_1[x \mapsto e_2]$$

此外：

$$\frac{e_1 \to_\beta e_1'}{\lambda x.e_1 \to_\beta \lambda x.e_1'} \qquad \frac{e_1 \to_\beta e_1'}{(e_1\ e_2) \to_\beta (e_1'\ e_2)} \qquad \frac{e_2 \to_\beta e_2'}{(e_1\ e_2) \to_\beta (e_1\ e_2')}$$

正则项 ~ Normalized Term

定义（多步规约）

称 $e \to_\beta^* e'$，如果存在从 e 到 e' 的零步、一步或多步规约。

定义（正则项）

一个项 e 称为正则的（Normalized），如果其不能进一步进行 β 规约。

正则项 ~ Normalized Term

定义（多步规约）

称 $e \to_\beta^* e'$，如果存在从 e 到 e' 的零步、一步或多步规约。

定义（正则项）

一个项 e 称为正则的（Normalized），如果其不能进一步进行 β 规约。

定理（Church-Rosser，1936）

若 $e \to_\beta^* e_1$ 且 $e \to_\beta^* e_2$，则存在 e' 使得 $e_1 \to_\beta^* e'$ 且 $e_2 \to_\beta^* e'$。

推论

一个项可规约得到的正则项是唯一的。

自然数 ~ Natural Number

自然数可以如下定义（丘奇数，Church Number）。

$$
\begin{aligned}
0 &:= \lambda f.\lambda x.x \\
1 &:= \lambda f.\lambda x.f\ x \\
2 &:= \lambda f.\lambda x.f\ (f\ x)
\end{aligned}
$$

- $n\ f \to_\beta^* f\ (f\ (\ldots f\ x))$ 即函数 f 的 n 次迭代。

自然数 ~ Natural Number

自然数可以如下定义（丘奇数，Church Number）。

$$
\begin{aligned}
0 &:= \lambda f.\lambda x.x \\
1 &:= \lambda f.\lambda x.f\ x \\
2 &:= \lambda f.\lambda x.f\ (f\ x)
\end{aligned}
$$

- $n\ f \to_\beta^* f\ (f\ (\ldots f\ x))$ 即函数 f 的 n 次迭代。
- **SUCC** : $n \mapsto n+1$: $\lambda m.\lambda f.\lambda x.f\ (n\ f\ x)$
- **MUL** : $n, m \mapsto n \times m$: $\lambda n.\lambda m.\lambda f.\lambda x.(m\ (n\ f)\ x)$
- **ADD** : $n, m \mapsto n+m$: $\lambda n.\lambda m.\lambda f.\lambda x.n\ f\ (m\ f\ x)$

丘奇-图灵论题 ~ Church-Turing Thesis

定义

令 $[n]$ 为 n 的丘奇数编码。称 λ 项 e 计算函数 $f\colon \mathbb{N} \to \mathbb{N}$，如果对于 $n \in \mathbb{N}$，都有 $e\,[n] \to_\beta^* [f(n)]$。

定理

一个函数是图灵可计算的，当且仅当其是 λ 可计算的，当且仅当存在一个递归函数（Recursive Function）计算它。

丘奇-图灵论题 ~ Church-Turing Thesis

定义

令 $[n]$ 为 n 的丘奇数编码。称 λ 项 e 计算函数 $f\colon \mathbb{N} \to \mathbb{N}$，如果对于 $n \in \mathbb{N}$，都有 $e\,[n] \to_\beta^* [f(n)]$。

定理

一个函数是图灵可计算的，当且仅当其是 λ 可计算的，当且仅当存在一个递归函数（Recursive Function）计算它。

丘奇-图灵论题（Church-Turing Thesis）

图灵可计算代表了对“可计算”这一概念的精确刻画。

类型 ∼ Type

不终止的程序

$$
\begin{aligned}
&(\lambda x.x\ x)\ (\lambda x.x\ x)\\
\rightarrow_\beta &(\lambda x.x\ x)\ (\lambda x.x\ x)\\
\rightarrow_\beta &(\lambda x.x\ x)\ (\lambda x.x\ x)\\
&\cdots
\end{aligned}
$$

类型 ∼ Type

不终止的程序

$$
\begin{aligned}
&(\lambda x.x\ x)\ (\lambda x.x\ x)\\
\rightarrow_\beta &(\lambda x.x\ x)\ (\lambda x.x\ x)\\
\rightarrow_\beta &(\lambda x.x\ x)\ (\lambda x.x\ x)\\
&\cdots
\end{aligned}
$$

类型系统（Type System）

为了保证程序拥有好的性质，可用类型系统确定一个项 e 的类型 α，记作 $(e:\alpha)$。有类型的项称为类型良好的（Well-typed）。

简单类型 λ 演算 ~ Simply-typed λ-calculus

类型（Type）

- 原子类型的集合 $\mathcal{A}$
- $\alpha := t \mid \alpha_1 \to \alpha_2$

简单类型 λ 演算 ~ Simply-typed λ-calculus

类型（Type）

- 原子类型的集合 $\mathcal{A}$
- $\alpha := t \mid \alpha_1 \to \alpha_2$

简单类型 λ 演算（Simply-typed λ-calculus）

- 抽象变量需要附类型 $\lambda x : \varphi . e$
- 令 $\Gamma = \{(x_i : \varphi_i)\}|_{i=1}^{n}$ 为环境
- 根据如下类型规则（Typing Rules）确定项的类型

简单类型 λ 演算 ~ Simply-typed λ-calculus

类型（Type）

- 原子类型的集合 $\mathcal{A}$
- $\alpha := t \mid \alpha_1 \to \alpha_2$

简单类型 λ 演算（Simply-typed λ-calculus）

- 抽象变量需要附类型 $\lambda x : \varphi . e$
- 令 $\Gamma = \{(x_i : \varphi_i)\}|_{i=1}^{n}$ 为环境
- 根据如下类型规则（Typing Rules）确定项的类型

$$\frac{(x:\varphi) \in \Gamma}{\Gamma \vdash x : \varphi} \qquad \frac{\Gamma, (x:\varphi) \vdash e : \psi}{\Gamma \vdash (\lambda x : \varphi . e) : \varphi \to \psi} \qquad \frac{\Gamma \vdash e_1 : \varphi \to \psi \quad \Gamma \vdash e_2 : \varphi}{\Gamma \vdash (e_1\ e_2) : \psi}$$

自然数的类型 ~ Type of Natural Numbers

丘奇数的定义：$n\ f$ 表示函数 f 的 n 次迭代

- 丘奇数是函数到函数的映射：$n : (t \to t) \to (t \to t)$
- 令 $\mathbb{N} := (t \to t) \to (t \to t)$

后继函数的类型：

$$\mathbf{SUCC} = \lambda n : \mathbb{N}.\lambda f : t \to t.\lambda x : t.f\ (n\ f\ x)$$

柯里-霍华德对应 ~ Curry-Howard Correspondence

类型 $\Leftrightarrow$ 命题，程序 $\Leftrightarrow$ 证明，假设 $\Leftrightarrow$ 变量

- 直观：类型是程序性质的抽象，命题是证明性质的抽象

$$\frac{(x:\varphi)\in\Gamma}{\Gamma\vdash x:\varphi} \qquad \frac{\Gamma,(x:\varphi)\vdash e:\psi}{\Gamma\vdash(\lambda x:\varphi.e):\varphi\to\psi} \qquad \frac{\Gamma\vdash e_1:\varphi\to\psi \quad \Gamma\vdash e_2:\varphi}{\Gamma\vdash(e_1\ e_2):\psi}$$

$$\Updownarrow \qquad\qquad \Updownarrow \qquad\qquad \Updownarrow$$

$$(\mathrm{HYP})[\varphi] \qquad (\to \mathrm{I})\frac{\begin{array}{c}[\varphi]^i\\ \vdots\\ \psi\end{array}}{\varphi\to\psi^i} \qquad (\to \mathrm{E})\frac{\begin{array}{cc}\vdots & \vdots\\ \varphi\to\psi & \varphi\end{array}}{\psi}$$

反证法 ~ Proof by Contradiction

如何处理反证法：

$$\text{(PBC)}\frac{\begin{array}{c}[\neg\varphi]^i\\ \vdots\\ \bot\end{array}}{\varphi^i}.$$

反证法 ~ Proof by Contradiction

如何处理反证法：

$$(\mathrm{PBC})\dfrac{\begin{array}{c}[\neg\varphi]^i\\ \vdots\\ \bot\end{array}}{\varphi^i}.$$

$\lambda\mu$-演算（$\lambda\mu$-calculus）

1. 在语言中增加新抽象规则 $\mu x:\neg\alpha.e$
2. 增加新的类型 $\bot$ 和类型规则

$$\frac{\Gamma, x:\neg\alpha \vdash e:\bot}{\Gamma \vdash (\mu x:\neg\alpha.e):\alpha}$$

柯里-霍华德对应 ~ Curry-Howard Correspondence

定理（命题逻辑的柯里-霍华德对应）

一个命题逻辑句子 φ 存在一棵证明树 T，当且仅当类型 φ 存在一个 $\lambda\mu$ 项 e。也就是说，$\lambda\mu$ 项是证明树的一种表示，其中

1. 命题对应于类型；
2. 证明对应于程序；
3. 假设对应于变量；
4. 假设集合 Γ 对应于类型上下文。

小结 ~ Summary

1. λ 演算的语法
2. β 规约，图灵-丘奇论题
3. 简单类型 $\lambda\mu$ 演算
4. 柯里-霍华德对应

主要内容 ~ Outline

改进的协议 ~ Improved Protocol

1. Assumption：如果 $\varphi \in \Gamma$ 则接受，否则拒绝；
2. Apply $\psi_1, \psi_2, \ldots, \psi_n$：如果

$$\psi_1 \to \psi_2 \to \cdots \to \psi_n \to \varphi \notin \Gamma$$

   则拒绝；否则递归地证明 $\Gamma \vdash \psi_i$，对于 $i = 1, 2, \ldots, n$；
3. Intro：若 $\varphi \neq \psi \to \gamma$，拒绝；否则递归证明 $\Gamma \cup \{\psi\} \vdash \gamma$；
4. Contra：递归地证明 $\Gamma \cup \{\neg\varphi\} \vdash \bot$；

定理：“无需灵感”

以上协议是完备的。

证明思路 ~ Proof Sketch

“自然的协议”

2. Apply ψ：递归地证明 $\Gamma \vdash \psi \to \varphi$ 和 $\Gamma \vdash \psi$，两者均接受时接受，否则拒绝；

改进后的协议

2. Apply $\psi_1, \psi_2, \ldots, \psi_n$：如果 $\psi_1 \to \psi_2 \to \cdots \to \psi_n \to \varphi \notin \Gamma$ 则拒绝；否则递归地证明 $\Gamma \vdash \psi_i$，对于 $i = 1, 2, \ldots, n$；

证明思路 ~ Proof Sketch

“自然的协议”

2. Apply ψ：递归地证明 $\Gamma \vdash \psi \to \varphi$ 和 $\Gamma \vdash \psi$，两者均接受时接受，否则拒绝；

改进后的协议

2. Apply $\psi_1, \psi_2, \ldots, \psi_n$：如果 $\psi_1 \to \psi_2 \to \cdots \to \psi_n \to \varphi \notin \Gamma$ 则拒绝；否则递归地证明 $\Gamma \vdash \psi_i$，对于 $i = 1, 2, \ldots, n$；

$$\dfrac{\dfrac{\dfrac{\vdots \quad \vdots}{\psi_1 \to \psi_2 \to \varphi \quad \psi_1}}{\psi_2 \to \varphi} \quad \vdots}{\varphi} \quad \psi_2 \quad \overset{?}{\to} \quad \dfrac{\vdots \quad \vdots \quad \vdots}{\psi_1 \to \psi_2 \to \varphi \quad \psi_1 \quad \psi_2}{\varphi}$$

证明思路-续 ~ Proof Sketch, Cont'd

$$\dfrac{\dfrac{\vdots \quad \vdots}{\psi_1 \to \psi_2 \to \varphi \quad \psi_1}}{\dfrac{\psi_2 \to \varphi \qquad \psi_2}{\varphi}} \quad \to \quad \dfrac{\vdots \quad \vdots \quad \vdots}{\psi_1 \to \psi_2 \to \varphi \quad \psi_1 \quad \psi_2}{\varphi} \quad \overset{?}{\to} \quad \dfrac{\vdots \quad \vdots \quad \vdots}{[\psi_1 \to \psi_2 \to \varphi] \quad \psi_1 \quad \psi_2}{\varphi}$$

$$\Updownarrow \qquad\qquad \Updownarrow \qquad\qquad \Updownarrow$$

$$(e_1^{\psi_1 \to \psi_2 \to \varphi} \; e_2^{\psi_1}) \; e_3^{\psi_2} \quad \to \quad e_1^{\psi_1 \to \psi_2 \to \varphi} \; e_2^{\psi_1} \; e_3^{\psi_2} \quad \overset{?}{\to} \quad x^{\psi_1 \to \psi_2 \to \varphi} \; e_2^{\psi_1} \; e_3^{\psi_2}$$

证明思路-续 ~ Proof Sketch, Cont'd

证明思路

- 我们称应用 $(e_1\ e_2)$ 是极左的，如果 $e_1 \neq (e_1'\ e_1'')$

证明思路-续 ~ Proof Sketch, Cont'd

证明思路

- 我们称应用 $(e_1\ e_2)$ 是极左的，如果 $e_1 \neq (e_1'\ e_1'')$

引理

如果一个 $\lambda\mu$ 项任何极左的应用 $(e_1\ e_2)$ 都满足 e_1 为变量，那么其对应的证明可以用改进的协议表示。

证明思路-续 ~ Proof Sketch, Cont'd

证明思路

- 我们称应用 $(e_1\ e_2)$ 是极左的，如果 $e_1 \neq (e_1'\ e_1'')$

引理

如果一个 $\lambda\mu$ 项任何极左的应用 $(e_1\ e_2)$ 都满足 e_1 为变量，那么其对应的证明可以用改进的协议表示。

证明.

施归纳于 $\lambda\mu$ 项的结构，证明其一定为

$$e := \lambda x : \varphi . e_1 \mid \mu x : \neg\varphi . e_1 \mid x\ e_1\ e_2\ \ldots\ e_n$$

而这自然地对应于改进的协议的一个证明。 □

证明思路-续 ~ Proof Sketch, Cont'd

定义

$\lambda\mu$ 项中的一个 $(\lambda x.e_1)\ e_2$ 或 $(\mu x.e_1)\ e_2$ 称为一个可规约项（Redex）；不存在可规约项的 $\lambda\mu$ 项是正则的（Normalized）。

证明思路-续 ~ Proof Sketch, Cont'd

定义

$\lambda\mu$ 项中的一个 $(\lambda x.e_1)\ e_2$ 或 $(\mu x.e_1)\ e_2$ 称为一个可规约项（Redex）；不存在可规约项的 $\lambda\mu$ 项是正则的（Normalized）。

引理：正则项的结构

正则项任一极左的应用 $(e_1\ e_2)$ 都满足 e_1 为变量。

证明思路-续 ~ Proof Sketch, Cont'd

引理

如果一个 $\lambda\mu$ 项任何极左的应用 $(e_1\ e_2)$ 都满足 e_1 为变量，那么其对应的证明可以用改进的协议表示。

引理：正则项的结构

正则项任一极左的应用 $(e_1\ e_2)$ 都满足 e_1 为变量。

证明思路-续 ~ Proof Sketch, Cont'd

引理

如果一个 $\lambda\mu$ 项任何极左的应用 $(e_1\ e_2)$ 都满足 e_1 为变量，那么其对应的证明可以用改进的协议表示。

引理：正则项的结构

正则项任一极左的应用 $(e_1\ e_2)$ 都满足 e_1 为变量。

弱正则定理

任一类型良好的 $\lambda\mu$ 项 e 都有同类型的正则项 e_n。

证明思路-续 ~ Proof Sketch, Cont'd

引理

如果一个 $\lambda\mu$ 项任何极左的应用 $(e_1\ e_2)$ 都满足 e_1 为变量，那么其对应的证明可以用改进的协议表示。

引理：正则项的结构

正则项任一极左的应用 $(e_1\ e_2)$ 都满足 e_1 为变量。

弱正则定理

任一类型良好的 $\lambda\mu$ 项 e 都有同类型的正则项 e_n。

推论

改进的协议是完备的。

类型与 β 规约 ~ Type and β-reduction

引理：β 规约

类型良好的项在 β 规约后仍然是类型良好的，且类型不变。

证明.

证明如果 $\Gamma \vdash e : \alpha$，并且 $e \to_\beta e'$，那么 $\Gamma \vdash e' : \alpha$。

$$(\lambda x : \varphi . e_1^\psi)\ e_2^\varphi \to_\beta e_1^\psi [x \mapsto e_2]$$

- 类型规则保证了 x 和 e_2 类型的匹配

□

β-规约与证明正则化 $\sim$ β-reduction and Proof Normalization

$$
\begin{array}{ccc}
(\lambda x.e_1)\ e_2 & \to_\beta & e_1[x \mapsto e_2] \\
\Updownarrow & & \Updownarrow \\
\dfrac{\dfrac{\begin{array}{c}[\varphi]^i \\ \vdots \\ \psi\end{array}}{\varphi \to \psi^i} \quad \dfrac{\vdots}{\varphi}}{\psi} & \to &
\end{array}
$$

μ 的规约 ~ Reduction for μ

类似 β 规约，我们可以为 $\mu x.e$ 设计规约规则

$$
\begin{array}{c}
(\mu x : \neg(\varphi \to \psi).e_1)\ (e_2 : \varphi) \\
\downarrow \\
\mu y : \neg\psi.e_1[x \mapsto (\lambda z : \varphi \to \psi.y\ (z\ e_2))]
\end{array}
$$

μ 的规约 ~ Reduction for μ

类似 β 规约，我们可以为 $\mu x.e$ 设计规约规则

$$(\mu x : \neg(\varphi \to \psi).e_1)\ (e_2 : \varphi)$$
$$\downarrow$$
$$\mu y : \neg\psi.e_1[x \mapsto (\lambda z : \varphi \to \psi.y\ (z\ e_2))]$$

这对应于证明正则化规则

$$\dfrac{\dfrac{\begin{array}{c}[\neg(\varphi \to \psi)]^i \\ \vdots \\ \bot\end{array}}{\varphi \to \psi^i} \qquad \dfrac{\vdots}{\varphi}}{\psi} \to$$

弱正则定理 ~ Weak Normalization Theorem

定理：弱正则定理（Weak Normalization Theorem）

任何类型良好的 $\lambda\mu$ 项 e 都可归约到一个正则的 $\lambda\mu$ 项 e_n。

定义（可规约项的高度）

- 定义类型 α 的高度 $[\alpha]_h$ 为对应树的高度。也就是说，$[t]_h = 1$，$[\alpha \to \beta]_h = 1 + \max\{[\alpha]_h, [\beta]_h\}$。
- 定义可规约项 $\Delta = (e_1\ e_2)$ 的高度为 e_1 类型的高度。

弱正则定理 ~ Weak Normalization Theorem

定义（可规约项的高度）

- 定义类型 α 的高度 $[\alpha]_h$ 为对应树的高度。也就是说，$[t]_h = 1$，$[\alpha \to \beta] = 1 + \max\{[\alpha]_h, [\beta]_h\}$。
- 定义可规约项 $\Delta = (e_1\ e_2)$ 的高度为 e_1 类型的高度。

证明思路.

令 h_0 为最高可规约项的高度，n_0 为高度为 h_0 的可规约项个数，$|e|$ 为项的长度。

1. 称 $(h_0, n_0, |e|)$ 为项的字典序
2. 施归纳于项的字典序：仅需证明若字典序小于此的项均可归约到正则项，那么此项也可归约到正则项

□

弱正则定理 ~ Weak Normalization Theorem

施归纳于 $(h_0, n_0, |e|)$。

1. 取 $\lambda\mu$ 项最右侧的最高可规约项 Δ，$[\Delta]_h = h_0$

弱正则定理 ~ Weak Normalization Theorem

施归纳于 $(h_0, n_0, |e|)$。

1. 取 $\lambda\mu$ 项最右侧的最高可规约项 Δ，$[\Delta]_h = h_0$
2. 如果 $\Delta = (\lambda x.e_1)\ e_2$，证明做此规约后不会引入同样高度的可规约项。

$$(\lambda x : \varphi.e_1^{\psi})\ e_2^{\varphi} \quad \to \quad e_1^{\psi}[x \mapsto e_2]$$

2.1 e_2 内部没有高度为 h_0 的可规约项（最右侧）
2.2 由 Δ 和外部产生的新可规约项 $e_1[x \mapsto e_2]\ e_2'$，高度 $[\psi]_h$
2.3 由 $x \mapsto e_2$ 产生的新可规约项 $(e_2\ e_3)$，高度 $[\varphi]_h$

弱正则定理 ~ Weak Normalization Theorem

施归纳于 $(h_0, n_0, |e|)$。

1. 取 $\lambda\mu$ 项最右侧的可规约项 Δ，$[\Delta]_h = h_0$
2. 如果 $\Delta = (\lambda x.e_1)\ e_2$，证明做此规约后不会引入同样高度的可规约项。

弱正则定理 ~ Weak Normalization Theorem

施归纳于 $(h_0, n_0, |e|)$。

1. 取 $\lambda\mu$ 项最右侧的可规约项 Δ，$[\Delta]_h = h_0$
2. 如果 $\Delta = (\lambda x.e_1)\ e_2$，证明做此规约后不会引入同样高度的可规约项。
3. 如果 $\Delta = (\mu x.e_1)\ e_2$，证明做此规约后进一步规约若干步，使得要么 h_0 减小，要么 h_0 不变且 n_0 减小。

$$(\mu x : \neg(\varphi \to \psi).e_1^{\perp}\ e_2^{\varphi}) \quad \to \quad \mu y : \neg\psi.e_1^{\perp}[x \mapsto \lambda z : \varphi \to \psi.y\ (z\ e_2)]$$

3.1 e_2 内部没有高度为 h_0 的可规约项（最右侧）

3.2 由 Δ 和外部产生的新可规约项 $(\mu y : \neg\psi.e_1')\ e_2'$，高度 $[\psi]_h$

3.3 由 $x \mapsto \lambda z.y\ (z\ e_2)$ 构成的可规约项 $(\lambda z : \varphi \to \psi.(y\ (z\ e_2))^{\perp})\ e_2'$，将其进一步规约为 $(y\ (e_2'\ e_2))$。易验证 $(e_2'\ e_2)$ 满足字典序小于原 $\lambda\mu$ 项；根据归纳假设，$(e_2'\ e_2)$ 可规约到一个类型相同的正则项。

例子 ~ An Example

试用改进的系统证明 $(\varphi \to \psi) \to (\neg\varphi \to \psi) \to \psi$.

- 先证明 $(\varphi \to \psi) \to (\neg\psi \to \neg\varphi)$

$$\mathbf{CP}_{\varphi,\psi} := \lambda x : \varphi \to \psi.\lambda y : \neg\psi.\lambda z : \varphi.y\ (x\ z).$$

- 再证明原命题

$$\lambda x : \varphi \to \psi.\lambda y : \neg\varphi \to \psi.\mu z : \neg\psi.(\mathbf{CP}_{\neg\varphi,\psi}\ y\ z)\ (\mathbf{CP}_{\varphi,\psi}\ x\ z)$$

- 正则化

$$\lambda x : \varphi \to \psi.\lambda y : \neg\varphi \to \psi.\mu z : \neg\psi.z\ (y\ \lambda w : \varphi.z\ (x\ w))$$

计算能力 ~ Computation Ability

回到没有 μ 的情景：简单类型 λ 演算

推论

存在一个可计算函数函数 f 不可用简单类型 λ 演算计算。

计算能力 ~ Computation Ability

回到没有 μ 的情景：简单类型 λ 演算

推论

存在一个可计算函数函数 f 不可用简单类型 λ 演算计算。

证明.

容易将所有的简单类型 λ 演算项用自然数编码为 $\{e_0, e_1, \ldots, e_n, \ldots\}$。定义函数

$$f(n) := e_n(n) + 1.$$

很明显 $f(n)$ 不能用任何简单类型 λ 演算项计算。由于弱正则定理给出了模拟 e_n 的一般算法，$f(n)$ 是可计算的。 □

小结 ~ Summary

传统逻辑方法：给出证明与模型的关系
证明论方法：考虑证明的结构变换

- 程序计算与证明正则化的关系

主要内容 ~ Outline

再谈希尔伯特系统 ~ An Eagle Eye on Hilbert-style System

简单类型组合子逻辑

程序的语法：$e := \mathbf{N} \mid (e_1\ e_2) \mid (x : \varphi)$，其中 $\mathbf{N}$ 为以下三种组合子

$$\mathbf{K}_{\varphi,\psi}\ \ \varphi \to \psi \to \varphi$$

$$\mathbf{S}_{\varphi,\psi,\gamma}\ \ (\varphi \to \psi \to \gamma) \to (\varphi \to \psi) \to \varphi \to \gamma$$

$$\mathbf{D}_{\varphi}\ \ \neg\neg\varphi \to \varphi$$

唯一的类型规则 $\dfrac{e_1 : \varphi \to \psi \quad e_2 : \varphi}{(e_1\ e_2) : \psi}$

再谈希尔伯特系统 ~ An Eagle Eye on Hilbert-style System

简单类型组合子逻辑

程序的语法：$e := \mathbf{N} \mid (e_1\ e_2) \mid (x : \varphi)$，其中 $\mathbf{N}$ 为以下三种组合子

$$\mathbf{K}_{\varphi,\psi}\ \ \varphi \to \psi \to \varphi$$

$$\mathbf{S}_{\varphi,\psi,\gamma}\ \ (\varphi \to \psi \to \gamma) \to (\varphi \to \psi) \to \varphi \to \gamma$$

$$\mathbf{D}_{\varphi}\ \ \neg\neg\varphi \to \varphi$$

唯一的类型规则 $\dfrac{e_1 : \varphi \to \psi \quad e_2 : \varphi}{(e_1\ e_2) : \psi}$

例子：$\mathbf{I}$ 组合子

$$\mathbf{I}_{\varphi} := \mathbf{S}_{\varphi,\varphi\to\varphi,\varphi}\ \mathbf{K}_{\varphi,\varphi\to\varphi}\ \mathbf{K}_{\varphi,\varphi}$$

- 容易验证 $\mathbf{I}_{\varphi} : \varphi \to \varphi$
- $\mathbf{I} := \mathbf{S}\ \mathbf{K}\ \mathbf{K}$

用 $\lambda\mu$ 项表示组合子 ~ Represent Combinators with $\lambda\mu$-terms

定理

如果类型 φ 存在一个组合子项 c，那么其也存在一个 $\lambda\mu$ 项 e。

用 $\lambda\mu$ 项表示组合子 ~ Represent Combinators with $\lambda\mu$-terms

定理

如果类型 φ 存在一个组合子项 c，那么其也存在一个 $\lambda\mu$ 项 e。

证明.

$$\begin{aligned}\mathbf{K} &:= \lambda x.\lambda y.x\\ \mathbf{S} &:= \lambda x.\lambda y.\lambda z.(x\ z)\ (y\ z)\\ \mathbf{D} &:= \lambda x.\mu y.x\ y\end{aligned}$$

□

完备性 ~ Completeness

定理：组合子逻辑的完备性

如果类型 φ 存在一个 $\lambda\mu$ 项 e，那么其也存在一个组合子项 c。

完备性 ~ Completeness

定理：组合子逻辑的完备性

如果类型 φ 存在一个 $\lambda\mu$ 项 e，那么其也存在一个组合子项 c。

证明.

由希尔伯特系统和自然演绎系统的完备性、可靠性立刻得到。 □

完备性 ~ Completeness

定理：组合子逻辑的完备性

如果类型 φ 存在一个 $\lambda\mu$ 项 e，那么其也存在一个组合子项 c。

证明.

由希尔伯特系统和自然演绎系统的完备性、可靠性立刻得到。 □

定理：证明的变换

存在算法将一个 $\lambda\mu$ 项转化成同类型的组合子项。这也说明，存在算法将一个自然演绎证明转化为一个希尔伯特系统的证明。

转化算法 ~ Translation Algorithm

定义函数 $\mathrm{ElimVar}(x, e)$，其中 $(x : \varphi)$ 为变量，$(e : \psi)$ 为组合子项

$$
\begin{aligned}
\mathrm{ElimVar}(x, (x : \varphi)) &:= \mathbf{I}_{\varphi} \\
\mathrm{ElimVar}(x, (y : \psi)) &:= \mathbf{K}_{\psi,\varphi}\ (y : \psi) \\
\mathrm{ElimVar}(x, \mathbf{N}^{\psi}) &:= \mathbf{K}_{\psi,\varphi}\ \mathbf{N}^{\psi} \\
\mathrm{ElimVar}(x, (e_1^{\gamma \to \psi}\ e_2^{\gamma})) &:= \mathbf{S}_{\varphi,\gamma,\psi}\ \mathrm{ElimVar}(x, e_1)\ \mathrm{ElimVar}(y, e_2)
\end{aligned}
$$

转化算法 ~ Translation Algorithm

定义函数 $\mathrm{ElimVar}(x, e)$，其中 $(x : \varphi)$ 为变量，$(e : \psi)$ 为组合子项

$$
\begin{aligned}
\mathrm{ElimVar}(x, (x : \varphi)) &:= \mathbf{I}_{\varphi} \\
\mathrm{ElimVar}(x, (y : \psi)) &:= \mathbf{K}_{\psi,\varphi}\ (y : \psi) \\
\mathrm{ElimVar}(x, \mathbf{N}^{\psi}) &:= \mathbf{K}_{\psi,\varphi}\ \mathbf{N}^{\psi} \\
\mathrm{ElimVar}(x, (e_1^{\gamma \to \psi}\ e_2^{\gamma})) &:= \mathbf{S}_{\varphi,\gamma,\psi}\ \mathrm{ElimVar}(x, e_1)\ \mathrm{ElimVar}(y, e_2)
\end{aligned}
$$

引理

$\mathrm{ElimVar}(x, e)$ 是一个不包含变量 x 的组合子项，且类型为 $\varphi \to \psi$

转化算法 ~ Translation Algorithm

引理

$\mathrm{ElimVar}(x, e)$ 是一个不包含变量 x 的组合子项，且类型为 $\varphi \to \psi$

转化算法 ~ Translation Algorithm

引理

$\mathrm{ElimVar}(x, e)$ 是一个不包含变量 x 的组合子项，且类型为 $\varphi \to \psi$

转化算法

定义函数 $\mathrm{Translate}(e, \Gamma)$，其中 e 是 $\lambda\mu$ 项，Γ 为自由变量的类型表，算法将其转化为等价的组合子项

$$
\begin{aligned}
\mathrm{Translate}(e_1\ e_2, \Gamma) &:= \mathrm{Translate}(e_1, \Gamma)\ \mathrm{Translate}(e_2, \Gamma) \\
\mathrm{Translate}(\lambda x : \varphi.e, \Gamma) &:= \mathrm{ElimVar}(x, \mathrm{Translate}(e, \Gamma \cup \{x : \varphi\})) \\
\mathrm{Translate}(\mu x : \neg\varphi.e, \Gamma) &:= \mathbf{D}_\varphi\ \mathrm{ElimVar}(x, e) \\
\mathrm{Translate}(x, \Gamma) &:= (x : \Gamma(x))
\end{aligned}
$$

容易验证 $\mathrm{Translate}(e, \Gamma)$ 生成一个相同类型的组合子项，且自由变量被 Γ 包含。

无类型组合子逻辑 ~ Untyped Combinator Logic

无类型组合子逻辑：$e := x \mid \mathbf{K} \mid \mathbf{S} \mid (e_1\ e_2)$，其中

$$\mathbf{K} := \lambda x.\lambda y.x$$
$$\mathbf{S} := \lambda x.\lambda y.\lambda z.(x\ z)\ (y\ z)$$

用 $(e)_\lambda$ 表示无类型组合子项 e 对应的 λ 项。

无类型组合子逻辑 ~ Untyped Combinator Logic

无类型组合子逻辑：$e := x \mid \mathbf{K} \mid \mathbf{S} \mid (e_1\ e_2)$，其中

$$\mathbf{K} := \lambda x.\lambda y.x$$
$$\mathbf{S} := \lambda x.\lambda y.\lambda z.(x\ z)\ (y\ z)$$

用 $(e)_\lambda$ 表示无类型组合子项 e 对应的 λ 项。

例子：$\mathbf{I} := \mathbf{S}\ \mathbf{K}\ \mathbf{K}$

$$\begin{aligned}\mathbf{I}\ e &:= \mathbf{S}\ \mathbf{K}\ \mathbf{K}\ e \\ &\to_\beta^* \mathbf{K}\ e\ (\mathbf{K}\ e) \\ &\to_\beta^* e.\end{aligned}$$

作为基的组合子 ~ Combinators as Basis

定理

组合子逻辑是图灵完备的。换言之，任何 λ 可计算的函数 $f\colon \mathbb{N} \to \mathbb{N}$ 都被某个组合子项计算。进一步，存在一个算法 $(*)_c$，如果 e 是一个计算 f 的 λ 项，那么 $(e)_c$ 是计算 f 的组合子项。

作为基的组合子 ~ Combinators as Basis

定理

组合子逻辑是图灵完备的。换言之，任何 λ 可计算的函数 $f\colon \mathbb{N} \to \mathbb{N}$ 都被某个组合子项计算。进一步，存在一个算法 $(*)_c$，如果 e 是一个计算 f 的 λ 项，那么 $(e)_c$ 是计算 f 的组合子项。

类似上面，令 e 为组合子项，定义 $\mathrm{ElimVar}(x, e)$ 为

$$
\begin{aligned}
\mathrm{ElimVar}(x, x) &:= \mathbf{I} \\
\mathrm{ElimVar}(x, y) &:= \mathbf{K}\ y \\
\mathrm{ElimVar}(x, \mathbf{N}) &:= \mathbf{K}\ \mathbf{N} \\
\mathrm{ElimVar}(x, (e_1\ e_2)) &:= \mathbf{S}\ \mathrm{ElimVar}(x, e_1)\ \mathrm{ElimVar}(x, e_2)
\end{aligned}
$$

可以看出 $\mathrm{ElimVar}(x, e)$ 和 $\lambda x.e$ 是等价的：对于同样的输入，给出相同的输出。

作为基的组合子 ~ Combinators as Basis

定理

组合子逻辑是图灵完备的。换言之，任何 λ 可计算的函数 $f\colon \mathbb{N} \to \mathbb{N}$ 都被某个组合子项计算。进一步，存在一个算法 $(*)_c$，如果 e 是一个计算 f 的 λ 项，那么 $(e)_c$ 是计算 f 的组合子项。

- $\mathrm{ElimVar}(x, e)$ 和 $\lambda x.e$ 是等价的

作为基的组合子 ~ Combinators as Basis

定理

组合子逻辑是图灵完备的。换言之，任何 λ 可计算的函数 $f\colon \mathbb{N} \to \mathbb{N}$ 都被某个组合子项计算。进一步，存在一个算法 $(*)_c$，如果 e 是一个计算 f 的 λ 项，那么 $(e)_c$ 是计算 f 的组合子项。

- $\mathrm{ElimVar}(x, e)$ 和 $\lambda x.e$ 是等价的
- 可递归地定义转化算法

$$
\begin{aligned}
(x)_c &:= x \\
(e_1\ e_2)_c &:= (e_1)_c\ (e_2)_c \\
(\lambda x.e)_c &:= \mathrm{ElimVar}(x, (e)_c)
\end{aligned}
$$

简单的图灵完备语言 ~ Simple Turing-complete Language

无类型组合子逻辑：$e := \mathbf{K} \mid \mathbf{S} \mid (e_1\ e_2)$，并有以下计算规则：

$$\mathbf{K}\ A\ B \to A$$
$$\mathbf{S}\ A\ B\ C \to A\ C\ (B\ C)$$

- “无需变量的程序设计语言”

小结 ~ Summary

1. 简单类型组合子逻辑：与希尔伯特系统对应
2. 无类型组合子逻辑：简单的图灵完备语言
3. 组合子逻辑与 λ 演算（$\lambda\mu$ 演算）的互译

主要内容 ~ Outline

致谢

参考的课程材料

1. 魏达格（Dag Westerståhl），逻辑学基础理论，清华大学 2020 年春
2. 段然，计算理论，清华大学 2020 年春

致谢

- 感谢 CCF 组织这次交流活动
- 感谢吴梦迪为 slide 审稿，并提出许多有价值的意见


Mario M. Carneiro.
Formalizing computability theory via partial recursive functions.
CoRR, abs/1810.08380, 2018.


Georges Gonthier.
The four colour theorem: Engineering of a formal proof.
In *Computer Mathematics, 8th Asian Symposium, ASCM 2007, Singapore, December 15-17, 2007. Revised and Invited Papers*, page 333, 2007.


Jesse Michael Han and Floris van Doorn.
A formal proof of the independence of the continuum hypothesis.
In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, page 353–366, New York, NY, USA, 2020. Association for Computing Machinery.


Morten Sørensen and Paweł Urzyczyn.
Lectures on the curry-howard isomorphism.
Studies in Logic and the Foundations of Mathematics, 149, 10 2010.

Thanks!