前情提要

前情提要

- 命题逻辑的可靠性

$$\Sigma \vdash \tau \Rightarrow \Sigma \vDash \tau$$

- 命题逻辑的完全性

$$\Sigma \vDash \tau \Rightarrow \Sigma \vdash \tau$$

前情提要

- 命题逻辑的可靠性

$$\Sigma \vdash \tau \Rightarrow \Sigma \vDash \tau$$

- 命题逻辑的完全性

$$\Sigma \vDash \tau \Rightarrow \Sigma \vdash \tau$$

前情提要

- 命题逻辑的可靠性

$$\Sigma \text{ 可满足} \Rightarrow \Sigma \text{ 一致}$$

- 命题逻辑的完全性

$$\Sigma \text{ 一致} \Rightarrow \Sigma \text{ 可满足}$$

前情提要

完全性证明：

仍给 Σ，假设 Σ 一致。扩张 Σ 为极大一致集 $\Delta \supset \Sigma$。利用 Δ 提供的信息构造赋值 v。($v(A) = 1 \Leftrightarrow A \in \Delta$) 证明 v 满足 Δ 因而满足 Σ

前情提要

完全性证明：

仍给 Σ，假设 Σ 一致。扩张 Σ 为极大一致集 $\Delta \supset \Sigma$。利用 Δ 提供的信息构造赋值 v。($v(A) = 1 \Leftrightarrow A \in \Delta$) 证明 v 满足 Δ 因而满足 Σ

前情提要

完全性证明：

仍给 Σ，假设 Σ 一致。扩张 Σ 为极大一致集 $\Delta \supset \Sigma$。利用 Δ 提供的信息构造赋值 v。($v(A) = 1 \Leftrightarrow A \in \Delta$) 证明 v 满足 Δ 因而满足 Σ

前情提要

完全性证明：

仍给 Σ，假设 Σ 一致。扩张 Σ 为极大一致集 $\Delta \supset \Sigma$。利用 Δ 提供的信息构造赋值 v。（ $v(A) = 1 \Leftrightarrow A \in \Delta$ ）证明 v 满足 Δ 因而满足 Σ

关于命题逻辑

- 存在一个能行的方式（计算机程序），任给一个命题逻辑公式判断它是否是重言式

  真值表
- 根据可靠性与完全性定理，命题逻辑的内定理集是可判定的
- 命题逻辑重言式/内定理集的复杂度是 NP 完全的

关于命题逻辑

- 存在一个能行的方式（计算机程序），任给一个命题逻辑公式判断它是否是重言式

  真值表
- 根据可靠性与完全性定理，命题逻辑的内定理集是可判定的
- 命题逻辑重言式/内定理集的复杂度是 NP 完全的

关于命题逻辑

- 存在一个能行的方式（计算机程序），任给一个命题逻辑公式判断它是否是重言式

  真值表
- 根据可靠性与完全性定理，命题逻辑的内定理集是可判定的
- 命题逻辑重言式/内定理集的复杂度是 NP 完全的

关于命题逻辑

- 存在一个能行的方式（计算机程序），任给一个命题逻辑公式判断它是否是重言式

  真值表
- 根据可靠性与完全性定理，命题逻辑的内定理集是可判定的
- 命题逻辑重言式/内定理集的复杂度是 NP 完全的

一阶谓词逻辑

不是命题逻辑的逻辑

- 如果有一个人是所有人的王，那么所有人都有一个王
- 如果所有人都有一个王，那么有一个人是所有人的王

$$\exists x \forall y K x y \rightarrow \forall y \exists x K x y$$

$$\forall y \exists x K x y \rightarrow \exists x \forall y K x y$$

不是命题逻辑的逻辑

- 如果有一个人是所有人的王，那么所有人都有一个王
- 如果所有人都有一个王，那么有一个人是所有人的王

$$\exists x \forall y K x y \rightarrow \forall y \exists x K x y$$

$$\forall y \exists x K x y \rightarrow \exists x \forall y K x y$$

不是命题逻辑的逻辑

- 如果有一个人是所有人的王，那么所有人都有一个王
- 如果所有人都有一个王，那么有一个人是所有人的王

$$\exists x \forall y K x y \rightarrow \forall y \exists x K x y$$

$$\forall y \exists x K x y \rightarrow \exists x \forall y K x y$$

一阶逻辑的语言

符号：

- 逻辑符号
  - 括号：$(,)$
  - 命题联词：$\neg, \rightarrow$
  - 量词：$\forall$
  - 变元：$v_1, v_2, \ldots$

一阶逻辑的语言

符号：

- 参数符号
  - 等词：$\approx$（可有可无）
  - 常数符号：$c_1, c_2, \ldots$ (*)
  - n-元谓词符号：$P_1, P_2, \ldots$ (*)
  - n-元函数符号：$f_1, f_2, \ldots$ (*)

(*) 可以没有，也可以有无穷多

一阶逻辑的语言

各种一阶逻辑语言

- 集合论语言：$\{\approx, \in\}$
- 初等数论的语言：$\{\approx, 0, <, S, +, \cdot\}$
- 序关系的语言：$\{\approx, R\}$

我们说“给定一个一阶逻辑语言”就是规定各类参数符号的集合。

一阶逻辑的语言

各种一阶逻辑语言

- 集合论语言：$\{\approx, \in\}$
- 初等数论的语言：$\{\approx, 0, <, S, +, \cdot\}$
- 序关系的语言：$\{\approx, R\}$

我们说“给定一个一阶逻辑语言”就是规定各类参数符号的集合。

一阶逻辑的语言

各种一阶逻辑语言

- 集合论语言：$\{\approx, \in\}$
- 初等数论的语言：$\{\approx, 0, <, S, +, \cdot\}$
- 序关系的语言：$\{\approx, R\}$

我们说“给定一个一阶逻辑语言”就是规定各类参数符号的集合。

一阶逻辑的语言

各种一阶逻辑语言

- 集合论语言：$\{\approx, \in\}$
- 初等数论的语言：$\{\approx, 0, <, S, +, \cdot\}$
- 序关系的语言：$\{\approx, R\}$

我们说“给定一个一阶逻辑语言”就是规定各类参数符号的集合。

一阶逻辑的语言

各种一阶逻辑语言

- 集合论语言：$\{\approx, \in\}$
- 初等数论的语言：$\{\approx, 0, <, S, +, \cdot\}$
- 序关系的语言：$\{\approx, R\}$

我们说“给定一个一阶逻辑语言”就是规定各类参数符号的集合。

一阶逻辑的语言

项（term）

给定一个一阶逻辑语言 $\mathcal{L}$，定义 $\mathcal{L}$ 中的项为：

- 每个变元 v_i 是项
- 每个 $\mathcal{L}$ 中的常数符号是项
- 如果 $t_1, \ldots, t_n$ 是项并且 f 是 $\mathcal{L}$ 中 n 元函数符号，那么 $ft_1 \ldots t_n$ 也是项

一阶逻辑的语言

项（term）

给定一个一阶逻辑语言 $\mathcal{L}$，定义 $\mathcal{L}$ 中的项为：

- 每个变元 v_i 是项
- 每个 $\mathcal{L}$ 中的常数符号是项
- 如果 $t_1, \ldots, t_n$ 是项并且 f 是 $\mathcal{L}$ 中 n 元函数符号，那么 $ft_1 \ldots t_n$ 也是项

一阶逻辑的语言

项（term）

给定一个一阶逻辑语言 $\mathcal{L}$，定义 $\mathcal{L}$ 中的项为：

- 每个变元 v_i 是项
- 每个 $\mathcal{L}$ 中的常数符号是项
- 如果 $t_1, \ldots, t_n$ 是项并且 f 是 $\mathcal{L}$ 中 n 元函数符号，那么 $ft_1 \ldots t_n$ 也是项

一阶逻辑的语言

项（term）

给定一个一阶逻辑语言 $\mathcal{L}$，定义 $\mathcal{L}$ 中的项为：

- 每个变元 v_i 是项
- 每个 $\mathcal{L}$ 中的常数符号是项
- 如果 $t_1, \ldots, t_n$ 是项并且 f 是 $\mathcal{L}$ 中 n 元函数符号，那么 $ft_1 \ldots t_n$ 也是项

一阶逻辑的语言

例：

初等数论语言中的项

一阶逻辑的语言

合式公式（well-founded formula）

给定一个一阶逻辑语言 $\mathcal{L}$，定义 $\mathcal{L}$ 中的合式公式为：

- 如果 t_1, t_2 是项，那么$t_1 \approx t_2$是公式（若 $\mathcal{L}$ 中有等词）
- 如果 $t_1, \ldots, t_n$ 是项且 P 是一个 n 元谓词符号，那么$Pt_1 \ldots t_n$是公式

  称上述公式是原子公式
- 如果 α, β 是合式公式，那么 $(\neg\alpha), (\alpha \to \beta)$ 也是
- 如果 α 是合式公式，那么$\forall x\alpha$也是

一阶逻辑的语言

合式公式（well-founded formula）

给定一个一阶逻辑语言 $\mathcal{L}$，定义 $\mathcal{L}$ 中的合式公式为：

- 如果 t_1, t_2 是项，那么$t_1 \approx t_2$是公式（若 $\mathcal{L}$ 中有等词）
- 如果 $t_1, \ldots, t_n$ 是项且 P 是一个 n 元谓词符号，那么$Pt_1 \ldots t_n$是公式

  称上述公式是原子公式
- 如果 α, β 是合式公式，那么 $(\neg\alpha), (\alpha \to \beta)$ 也是
- 如果 α 是合式公式，那么$\forall x\alpha$也是

一阶逻辑的语言

合式公式 (well-founded formula)

给定一个一阶逻辑语言 $\mathcal{L}$，定义 $\mathcal{L}$ 中的合式公式为：

- 如果 t_1, t_2 是项，那么$t_1 \approx t_2$是公式 (若 $\mathcal{L}$ 中有等词)
- 如果 $t_1, \ldots, t_n$ 是项且 P 是一个 n 元谓词符号，那么$Pt_1 \ldots t_n$是公式

  称上述公式是原子公式
- 如果 α, β 是合式公式，那么 $(\neg\alpha), (\alpha \to \beta)$ 也是
- 如果 α 是合式公式，那么$\forall x\alpha$也是

一阶逻辑的语言

合式公式 (well-founded formula)

给定一个一阶逻辑语言 $\mathcal{L}$，定义 $\mathcal{L}$ 中的合式公式为：

- 如果 t_1, t_2 是项，那么$t_1 \approx t_2$是公式 (若 $\mathcal{L}$ 中有等词)
- 如果 $t_1, \ldots, t_n$ 是项且 P 是一个 n 元谓词符号，那么$Pt_1 \ldots t_n$是公式

  称上述公式是原子公式
- 如果 α, β 是合式公式，那么 $(\neg\alpha), (\alpha \to \beta)$ 也是
- 如果 α 是合式公式，那么$\forall x\alpha$也是

一阶逻辑的语言

合式公式 (well-founded formula)

给定一个一阶逻辑语言 $\mathcal{L}$，定义 $\mathcal{L}$ 中的合式公式为：

- 如果 t_1, t_2 是项，那么$t_1 \approx t_2$是公式 (若 $\mathcal{L}$ 中有等词)
- 如果 $t_1, \ldots, t_n$ 是项且 P 是一个 n 元谓词符号，那么$Pt_1 \ldots t_n$是公式

  称上述公式是原子公式
- 如果 α, β 是合式公式，那么 $(\neg\alpha), (\alpha \to \beta)$ 也是
- 如果 α 是合式公式，那么$\forall x\alpha$也是

一阶逻辑的语言

合式公式 (well-founded formula)

给定一个一阶逻辑语言 $\mathcal{L}$，定义 $\mathcal{L}$ 中的合式公式为：

- 如果 t_1, t_2 是项，那么$t_1 \approx t_2$是公式 (若 $\mathcal{L}$ 中有等词)
- 如果 $t_1, \ldots, t_n$ 是项且 P 是一个 n 元谓词符号，那么$Pt_1 \ldots t_n$是公式

  称上述公式是原子公式
- 如果 α, β 是合式公式，那么 $(\neg\alpha), (\alpha \to \beta)$ 也是
- 如果 α 是合式公式，那么$\forall x\alpha$也是

一阶逻辑的语言

合式公式 (well-founded formula)

给定一个一阶逻辑语言 $\mathcal{L}$，定义 $\mathcal{L}$ 中的合式公式为：

- 如果 t_1, t_2 是项，那么$t_1 \approx t_2$是公式 (若 $\mathcal{L}$ 中有等词)
- 如果 $t_1, \ldots, t_n$ 是项且 P 是一个 n 元谓词符号，那么$Pt_1 \ldots t_n$是公式

  称上述公式是原子公式
- 如果 α, β 是合式公式，那么 $(\neg\alpha), (\alpha \to \beta)$ 也是
- 如果 α 是合式公式，那么$\forall x\alpha$也是

一阶逻辑的语言

缩写规定：

- $\alpha \vee \beta =_{\text{abbr}} \neg\alpha \to \beta$
- $\alpha \wedge \beta =_{\text{abbr}} \neg(\alpha \to \neg\beta)$
- $\alpha \leftrightarrow \beta =_{\text{abbr}} (\alpha \to \beta) \wedge (\beta \to \alpha)$
- $\exists x\alpha =_{\text{abbr}} \neg\forall x\neg\alpha$

我们习惯称 $\forall x$ 为全称量词，称 $\exists x$ 为存在量词

一阶逻辑的语言

缩写规定：

- $\alpha \vee \beta =_{\text{abbr}} \neg\alpha \to \beta$
- $\alpha \wedge \beta =_{\text{abbr}} \neg(\alpha \to \neg\beta)$
- $\alpha \leftrightarrow \beta =_{\text{abbr}} (\alpha \to \beta) \wedge (\beta \to \alpha)$
- $\exists x\alpha =_{\text{abbr}} \neg\forall x\neg\alpha$

我们习惯称 $\forall x$ 为全称量词，称 $\exists x$ 为存在量词

一阶逻辑的语言

缩写规定：

- $\alpha \vee \beta =_{\text{abbr}} \neg\alpha \to \beta$
- $\alpha \wedge \beta =_{\text{abbr}} \neg(\alpha \to \neg\beta)$
- $\alpha \leftrightarrow \beta =_{\text{abbr}} (\alpha \to \beta) \wedge (\beta \to \alpha)$
- $\exists x\alpha =_{\text{abbr}} \neg\forall x\neg\alpha$

我们习惯称 $\forall x$ 为全称量词，称 $\exists x$ 为存在量词

一阶逻辑的语言

缩写规定：

- $\alpha \vee \beta =_{\text{abbr}} \neg\alpha \to \beta$
- $\alpha \wedge \beta =_{\text{abbr}} \neg(\alpha \to \neg\beta)$
- $\alpha \leftrightarrow \beta =_{\text{abbr}} (\alpha \to \beta) \wedge (\beta \to \alpha)$
- $\exists x\alpha =_{\text{abbr}} \neg\forall x\neg\alpha$

我们习惯称 $\forall x$ 为全称量词，称 $\exists x$ 为存在量词

一阶逻辑的语言

缩写规定：

- $\alpha \vee \beta =_{\text{abbr}} \neg\alpha \to \beta$
- $\alpha \wedge \beta =_{\text{abbr}} \neg(\alpha \to \neg\beta)$
- $\alpha \leftrightarrow \beta =_{\text{abbr}} (\alpha \to \beta) \wedge (\beta \to \alpha)$
- $\exists x\alpha =_{\text{abbr}} \neg\forall x\neg\alpha$

我们习惯称 $\forall x$ 为全称量词，称 $\exists x$ 为存在量词

自由出现与约束出现

$$\sum_{i=1}^{n} a_i$$

自由出现与约束出现

$$\sum_{i=1}^{n} a_i$$

自由出现与约束出现

$$\sum_{j=1}^{n} a_j$$

自由出现与约束出现

对公式 α 递归定义x 在 α 中自由出现：

- α 是原子公式：x 在 α 中出现
- $\alpha = \neg\beta$：x 在 β 中自由出现
- $\alpha = \beta \to \gamma$：$x$ 在 β 或 γ 中自由出现
- $\alpha = \forall y\beta$：x 在 β 中自由出现且 $x \neq y$

x 在 $\forall x\alpha$ 中的出现称作约束出现

自由出现与约束出现

对公式 α 递归定义x 在 α 中自由出现：

- α 是原子公式：x 在 α 中出现
- $\alpha = \neg\beta$：x 在 β 中自由出现
- $\alpha = \beta \to \gamma$：$x$ 在 β 或 γ 中自由出现
- $\alpha = \forall y\beta$：x 在 β 中自由出现且 $x \neq y$

x 在 $\forall x\alpha$ 中的出现称作约束出现

自由出现与约束出现

对公式 α 递归定义x 在 α 中自由出现：

- α 是原子公式：x 在 α 中出现
- $\alpha = \neg\beta$：x 在 β 中自由出现
- $\alpha = \beta \to \gamma$：$x$ 在 β 或 γ 中自由出现
- $\alpha = \forall y\beta$：x 在 β 中自由出现且 $x \neq y$

x 在 $\forall x\alpha$ 中的出现称作约束出现

自由出现与约束出现

对公式 α 递归定义x 在 α 中自由出现：

- α 是原子公式：x 在 α 中出现
- $\alpha = \neg\beta$：x 在 β 中自由出现
- $\alpha = \beta \to \gamma$：$x$ 在 β 或 γ 中自由出现
- $\alpha = \forall y\beta$：x 在 β 中自由出现且 $x \neq y$

x 在 $\forall x\alpha$ 中的出现称作约束出现

自由出现与约束出现

对公式 α 递归定义x 在 α 中自由出现：

- α 是原子公式：x 在 α 中出现
- $\alpha = \neg\beta$：x 在 β 中自由出现
- $\alpha = \beta \to \gamma$：$x$ 在 β 或 γ 中自由出现
- $\alpha = \forall y\beta$：x 在 β 中自由出现且 $x \neq y$

x 在 $\forall x\alpha$ 中的出现称作约束出现

自由出现与约束出现

对公式 α 递归定义x 在 α 中自由出现：

- α 是原子公式：x 在 α 中出现
- $\alpha = \neg\beta$：x 在 β 中自由出现
- $\alpha = \beta \to \gamma$：$x$ 在 β 或 γ 中自由出现
- $\alpha = \forall y\beta$：x 在 β 中自由出现且 $x \neq y$

x 在 $\forall x\alpha$ 中的出现称作约束出现

自由出现与约束出现

我们称一个合式公式 α 是语句（sentence），当且仅当没有变元在 α 中自由出现

代入

我们定义元语言中的一个表达方式α_t^x

首先对项 u 递归定义 u_t^x

- $u = y$：

$$u_t^x = \begin{cases} t & \text{若 } x = y \\ y & \text{否则} \end{cases}$$

- $u = ft_1 \dots t_n$：$u_t^x = f(t_1)_t^x \dots (t_n)_t^x$

代入

我们定义元语言中的一个表达方式α_t^x

首先对项 u 递归定义 u_t^x

- $u = y$：

$$u_t^x = \begin{cases} t & \text{若 } x = y \\ y & \text{否则} \end{cases}$$

- $u = ft_1 \dots t_n$：$u_t^x = f(t_1)_t^x \dots (t_n)_t^x$

代入

我们定义元语言中的一个表达方式α_t^x

首先对项 u 递归定义 u_t^x

- $u = y$:

$$u_t^x = \begin{cases} t & \text{若 } x = y \\ y & \text{否则} \end{cases}$$

- $u = ft_1 \dots t_n$: $u_t^x = f(t_1)_t^x \dots (t_n)_t^x$

代入

我们定义元语言中的一个表达方式α_t^x

首先对项 u 递归定义 u_t^x

- $u = y$:

$$u_t^x = \begin{cases} t & \text{若 } x = y \\ y & \text{否则} \end{cases}$$

- $u = ft_1 \dots t_n$: $u_t^x = f(t_1)_t^x \dots (t_n)_t^x$

代入

我们定义元语言中的一个表达方式α_t^x

首先对项 u 递归定义 u_t^x

- $u = y$:

$$u_t^x = \begin{cases} t & \text{若 } x = y \\ y & \text{否则} \end{cases}$$

- $u = ft_1 \dots t_n$: $u_t^x = f(t_1)_t^x \dots (t_n)_t^x$

代入

其次对公式 α 递归定义 α_t^x

- $\alpha = t_1 \approx t_2 : \alpha_t^x = (t_1)_t^x \approx (t_2)_t^x$
- $\alpha = Pt_1 \dots t_n : \alpha_t^x = P(t_1)_t^x \dots (t_n)_t^x$
- $\alpha = \neg\beta : \alpha_t^x = (\neg\beta)_t^x = \neg\beta_t^x$
- $\alpha = \beta \to \gamma : \alpha_t^x = (\beta \to \gamma)_t^x = \beta_t^x \to \gamma_t^x$
- $\alpha = \forall y\beta :$

$$\alpha_t^x = (\forall y\beta)_t^x = \begin{cases} \alpha & \text{若 } x = y \\ \forall y\beta_t^x & \text{否则} \end{cases}$$

代入

其次对公式 α 递归定义 α_t^x

- $\alpha = t_1 \approx t_2 : \alpha_t^x = (t_1)_t^x \approx (t_2)_t^x$
- $\alpha = Pt_1 \dots t_n : \alpha_t^x = P(t_1)_t^x \dots (t_n)_t^x$
- $\alpha = \neg\beta : \alpha_t^x = (\neg\beta)_t^x = \neg\beta_t^x$
- $\alpha = \beta \to \gamma : \alpha_t^x = (\beta \to \gamma)_t^x = \beta_t^x \to \gamma_t^x$
- $\alpha = \forall y\beta :$

$$\alpha_t^x = (\forall y\beta)_t^x = \begin{cases} \alpha & \text{若 } x = y \\ \forall y\beta_t^x & \text{否则} \end{cases}$$

代入

其次对公式 α 递归定义 α_t^x

- $\alpha = t_1 \approx t_2 : \alpha_t^x = (t_1)_t^x \approx (t_2)_t^x$
- $\alpha = Pt_1 \dots t_n : \alpha_t^x = P(t_1)_t^x \dots (t_n)_t^x$
- $\alpha = \neg\beta : \alpha_t^x = (\neg\beta)_t^x = \neg\beta_t^x$
- $\alpha = \beta \to \gamma : \alpha_t^x = (\beta \to \gamma)_t^x = \beta_t^x \to \gamma_t^x$
- $\alpha = \forall y\beta :$

$$\alpha_t^x = (\forall y\beta)_t^x = \begin{cases} \alpha & \text{若 } x = y \\ \forall y\beta_t^x & \text{否则} \end{cases}$$

代入

其次对公式 α 递归定义 α_t^x

- $\alpha = t_1 \approx t_2 : \alpha_t^x = (t_1)_t^x \approx (t_2)_t^x$
- $\alpha = Pt_1 \dots t_n : \alpha_t^x = P(t_1)_t^x \dots (t_n)_t^x$
- $\alpha = \neg\beta : \alpha_t^x = (\neg\beta)_t^x = \neg\beta_t^x$
- $\alpha = \beta \to \gamma : \alpha_t^x = (\beta \to \gamma)_t^x = \beta_t^x \to \gamma_t^x$
- $\alpha = \forall y\beta :$

$$\alpha_t^x = (\forall y\beta)_t^x = \begin{cases} \alpha & \text{若 } x = y \\ \forall y\beta_t^x & \text{否则} \end{cases}$$

代入

其次对公式 α 递归定义 α_t^x

- $\alpha = t_1 \approx t_2 : \alpha_t^x = (t_1)_t^x \approx (t_2)_t^x$
- $\alpha = Pt_1 \dots t_n : \alpha_t^x = P(t_1)_t^x \dots (t_n)_t^x$
- $\alpha = \neg\beta : \alpha_t^x = (\neg\beta)_t^x = \neg\beta_t^x$
- $\alpha = \beta \to \gamma : \alpha_t^x = (\beta \to \gamma)_t^x = \beta_t^x \to \gamma_t^x$
- $\alpha = \forall y\beta :$

$$\alpha_t^x = (\forall y\beta)_t^x = \begin{cases} \alpha & \text{若 } x = y \\ \forall y\beta_t^x & \text{否则} \end{cases}$$

一阶谓词逻辑的希尔伯特系统

一阶谓词逻辑的希尔伯特系统

定义 (全称概括)

称公式 φ 是公式 ψ 的全称概括，当且仅当存在自然数 n 和变元 $x_1, \ldots, x_n$ 使得

$$\varphi = \forall x_1 \ldots \forall x_n \psi$$

一阶谓词逻辑的希尔伯特系统

定义 (素公式)

我们称原子公式或形如 $\forall x\beta$ 的公式为素公式

令 $\{\beta_1, \beta_2, \ldots\}$ 是对所有素公式的枚举。我们定义一个一阶逻辑公式 α 的命题逻辑公式对应 α^P :

- $\alpha = \beta_i$ 是一个素公式 : $\alpha^P = \beta_i^P = A_i$
- $\alpha = \neg\beta$: $\alpha^P = \neg\beta^P$
- $\alpha = \beta \to \gamma$: $\alpha^P = \beta^P \to \gamma^P$

一阶谓词逻辑的希尔伯特系统

定义 (素公式)

我们称原子公式或形如 $\forall x\beta$ 的公式为素公式

令 $\{\beta_1, \beta_2, \ldots\}$ 是对所有素公式的枚举。我们定义一个一阶逻辑公式 α 的命题逻辑公式对应 α^P ：

- $\alpha = \beta_i$ 是一个素公式：$\alpha^P = \beta_i^P = A_i$
- $\alpha = \neg\beta$: $\alpha^P = \neg\beta^P$
- $\alpha = \beta \to \gamma$: $\alpha^P = \beta^P \to \gamma^P$

一阶谓词逻辑的希尔伯特系统

定义 (素公式)

我们称原子公式或形如 $\forall x\beta$ 的公式为素公式

令 $\{\beta_1, \beta_2, \ldots\}$ 是对所有素公式的枚举。我们定义一个一阶逻辑公式 α 的命题逻辑公式对应 α^P ：

- $\alpha = \beta_i$ 是一个素公式：$\alpha^P = \beta_i^P = A_i$
- $\alpha = \neg\beta$: $\alpha^P = \neg\beta^P$
- $\alpha = \beta \to \gamma$: $\alpha^P = \beta^P \to \gamma^P$

一阶谓词逻辑的希尔伯特系统

定义 (素公式)

我们称原子公式或形如 $\forall x\beta$ 的公式为素公式

令 $\{\beta_1, \beta_2, \ldots\}$ 是对所有素公式的枚举。我们定义一个一阶逻辑公式 α 的命题逻辑公式对应 α^P ：

- $\alpha = \beta_i$ 是一个素公式：$\alpha^P = \beta_i^P = A_i$
- $\alpha = \neg\beta$：$\alpha^P = \neg\beta^P$
- $\alpha = \beta \to \gamma$：$\alpha^P = \beta^P \to \gamma^P$

一阶谓词逻辑的希尔伯特系统

定义 (素公式)

我们称原子公式或形如 $\forall x\beta$ 的公式为素公式

令 $\{\beta_1, \beta_2, \ldots\}$ 是对所有素公式的枚举。我们定义一个一阶逻辑公式 α 的命题逻辑公式对应 α^P ：

- $\alpha = \beta_i$ 是一个素公式：$\alpha^P = \beta_i^P = A_i$
- $\alpha = \neg\beta$: $\alpha^P = \neg\beta^P$
- $\alpha = \beta \to \gamma$: $\alpha^P = \beta^P \to \gamma^P$

一阶谓词逻辑的希尔伯特系统

公理：一阶逻辑希尔伯特系统的公理是由下列公式的全称概括组成的

1. 对应的命题逻辑公式 α^P 是重言式的一阶逻辑公式 α
2. $\forall x\alpha \to \alpha_t^x$，其中项 t 可以在 α 中替代 x
3. $\forall x(\alpha \to \beta) \to (\forall x\alpha \to \forall x\beta)$
4. $\alpha \to \forall x\alpha$，其中 x 不在 α 中自由出现

一阶谓词逻辑的希尔伯特系统

公理：一阶逻辑希尔伯特系统的公理是由下列公式的全称概括组成的

1. 对应的命题逻辑公式 α^P 是重言式的一阶逻辑公式 α
2. $\forall x\alpha \to \alpha_t^x$，其中项 t 可以在 α 中替代 x
3. $\forall x(\alpha \to \beta) \to (\forall x\alpha \to \forall x\beta)$
4. $\alpha \to \forall x\alpha$，其中 x 不在 α 中自由出现

一阶谓词逻辑的希尔伯特系统

公理：一阶逻辑希尔伯特系统的公理是由下列公式的全称概括组成的

1. 对应的命题逻辑公式 α^P 是重言式的一阶逻辑公式 α
2. $\forall x\alpha \to \alpha_t^x$，其中项 t 可以在 α 中替代 x
3. $\forall x(\alpha \to \beta) \to (\forall x\alpha \to \forall x\beta)$
4. $\alpha \to \forall x\alpha$，其中 x 不在 α 中自由出现

一阶谓词逻辑的希尔伯特系统

公理：一阶逻辑希尔伯特系统的公理是由下列公式的全称概括组成的

1. 对应的命题逻辑公式 α^P 是重言式的一阶逻辑公式 α
2. $\forall x\alpha \to \alpha_t^x$，其中项 t 可以在 α 中替代 x
3. $\forall x(\alpha \to \beta) \to (\forall x\alpha \to \forall x\beta)$
4. $\alpha \to \forall x\alpha$，其中 x 不在 α 中自由出现

一阶谓词逻辑的希尔伯特系统

公理：一阶逻辑希尔伯特系统的公理是由下列公式的全称概括组成的

1. 对应的命题逻辑公式 α^P 是重言式的一阶逻辑公式 α
2. $\forall x\alpha \to \alpha_t^x$，其中项 t 可以在 α 中替代 x
3. $\forall x(\alpha \to \beta) \to (\forall x\alpha \to \forall x\beta)$
4. $\alpha \to \forall x\alpha$，其中 x 不在 α 中自由出现

一阶谓词逻辑的希尔伯特系统

公理：一阶逻辑希尔伯特系统的公理是由下列公式的全称概括组成的

若语言中包涵等词，则还有

5. $x \approx x$
6. $x \approx y \to \alpha \to \alpha'$，其中 α 为原子公式，且 α' 是将 α 中若干个 x 的出现替换为 y 所得到的公式

一阶谓词逻辑的希尔伯特系统

公理：一阶逻辑希尔伯特系统的公理是由下列公式的全称概括组成的

若语言中包涵等词，则还有

5. $x \approx x$
6. $x \approx y \to \alpha \to \alpha'$，其中 α 为原子公式，且 α' 是将 α 中若干个 x 的出现替换为 y 所得到的公式

一阶谓词逻辑的希尔伯特系统

公理：一阶逻辑希尔伯特系统的公理是由下列公式的全称概括组成的

若语言中包涵等词，则还有

5. $x \approx x$
6. $x \approx y \to \alpha \to \alpha'$，其中 α 为原子公式，且 α' 是将 α 中若干个 x 的出现替换为 y 所得到的公式

一阶谓词逻辑的希尔伯特系统

不可替代的例子

令 $\alpha = \exists y x \not\approx y$。分别考虑

- $\forall x \alpha \to \alpha_z^x$
- $\forall x \alpha \to \alpha_y^x$

如果我们希望定义 项 t 可以在 α 中替代 x 为“替换后 t 中的变元不会被 α 中已有的量词抓住”，我们该怎样严格地给出定义？

一阶谓词逻辑的希尔伯特系统

不可替代的例子

令 $\alpha = \exists y x \not\approx y$。分别考虑

- $\forall x \alpha \to \alpha^x_z$
- $\forall x \alpha \to \alpha^x_y$

如果我们希望定义 项 t 可以在 α 中替代 x 为“替换后 t 中的变元不会被 α 中已有的量词抓住”，我们该怎样严格地给出定义？

习题

4.1 (1) - (4)

5.1 (1)、(2)