数理逻辑复习笔记

课程资料整理

2025 年 5 月 14 日

目录

第一部分　预备知识

1　集合 (Set)

定义 1.1 (集合)**.** 一个**集合** *(Set)* 是一组对象的汇集。集合中的对象称为**元素** *(Element)*。

示例：

- $A =$ 我们班级的学生
- $B = \{1, 5, 10, 20\}$
- $\mathbb{Z} =$ 整数集

如果 Tom 是我们班级的学生，我们记作 Tom $\in A$。$1 \in B$，但 $30 \notin B$。

1.1　集合的内涵与外延 (Intension & Extension)

- **内涵 (Intension)**: 描述集合的定义性属性，即集合成员应满足的条件。
- **外延 (Extension)**: 集合的实际成员或内容。

示例：$\mathbb{P} =$ 素数集

- 内涵：大于 1 且不能被两个更小的自然数相乘得到的自然数。
- 外延：$\{2, 3, 5, 7, 11, \dots\}$

1.2　集合的表示与外延公理

如果 $\varphi(x)$ 代表一个性质，那么 $\{x|\varphi(x)\}$ 表示所有具有该性质的元素的集合。

- $\{x|x \in \mathbb{Z}$ 且x 能被 2 整除$\}$ 表示偶数集。
- $\{x|x \neq x\}$ 表示**空集** (Empty set)，记为 $\emptyset$ 或 $\{\}$。

定义 1.2 (外延公理 (Axiom of Extension))**.** 两个集合 A 和 B 相等 *($A = B$)*，当且仅当它们拥有相同的成员。

注意：集合中元素的顺序和重复无关。例如：$\{a, b\} = \{b, a\} = \{a, a, b\}$。

1.3　子集与幂集 (Subset & Power Set)

定义 1.3 (子集 (Subset))**.** 如果集合 A 的所有元素也是集合 B 的元素，则称 A 是 B 的**子集**，记作 $A \subseteq B$。形式化地，$A \subseteq B \iff \forall x(x \in A \to x \in B)$。

- **真子集 (Proper Subset)**: 如果 $A \subseteq B$ 且 $A \neq B$，则称 A 是 B 的真子集，记作 $A \subset B$。
- 对于任何集合 A，都有 $\emptyset \subseteq A$ 和 $A \subseteq A$。空集是任何集合（除自身外）的真子集。

定义 1.4 (幂集 (Power Set))**.** 集合 A 的**幂集** $\mathcal{P}(A)$ 是 A 的所有子集构成的集合。即 $\mathcal{P}(A) = \{X|X \subseteq A\}$。

示例：

- $\mathcal{P}(\{a, b, c\}) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$
- $\mathcal{P}(\emptyset) = \{\emptyset\}$

1.4 集合运算 (Set Operations)

- **并集 (Union)**: $A \cup B = \{x | x \in A \text{ 或} x \in B\}$
- **交集 (Intersection)**: $A \cap B = \{x | x \in A \text{ 且} x \in B\}$
- **差集 (Difference)**: $A - B = \{x | x \in A \text{ 且} x \notin B\}$

性质:

- $A \subseteq B \iff A \cap B = A$
- $A \subseteq B \iff A \cup B = B$

2 关系 (Relation)

关系用于描述对象之间的联系，是构建数学结构的基础。

2.1 n 元组 (n-tuples)

定义 2.1 (n 元组). 一个 ***n* 元组** 是 n 个元素的有限序列或有序列表，记作 $\langle x_1, x_2, \ldots, x_n \rangle$。

性质:

- **相等性**: $\langle x_1, \ldots, x_m \rangle = \langle y_1, \ldots, y_n \rangle \iff m = n$ 且$\forall i (1 \leq i \leq n), x_i = y_i$。
- 元组可以包含重复元素，例如 $\langle 1, 2, 2, 3 \rangle \neq \langle 1, 2, 3 \rangle$。
- 元组中元素是有序的，例如 $\langle 1, 2, 3 \rangle \neq \langle 3, 2, 1 \rangle$。
- 元组元素个数是有限的。

2.2 笛卡尔积与二元关系 (Cartesian Product & Binary Relation)

定义 2.2 (笛卡尔积). 集合 A 和 B 的**笛卡尔积** $A \times B = \{\langle x, y \rangle | x \in A \text{ 且} y \in B\}$。

定义 2.3 (二元关系). 集合 X 和 Y 上的一个**二元关系** R 是它们笛卡尔积 $X \times Y$ 的一个子集，即 $R \subseteq X \times Y$。如果 $\langle x, y \rangle \in R$，我们称 x 与 y 有关系 R，记作 xRy 或 $R(x, y)$。如果 $X = Y$，称 R 是 X 上的一个二元关系。

示例：

- $\{\langle x, y \rangle | x \text{ 是} y \text{ 的父亲}\}$ 是人类集合上的一个二元关系。
- $<$ 是自然数集 $\mathbb{N}$ 上的一个二元关系。

2.3 n 元关系 (n-ary Relation)

集合 $A_1, A_2, \ldots, A_n$ 的笛卡尔积为 $A_1 \times \cdots \times A_n = \{\langle x_1, \ldots, x_n \rangle | x_1 \in A_1, \ldots, x_n \in A_n\}$。如果 $A_1 = \cdots = A_n = A$，则记为 A^n。如果 $R \subseteq A^n$，则称 R 是 A 上的一个 **n 元关系**。

2.4 关系的性质

令 R 为集合 A 上的一个二元关系。

- **自反性 (Reflexive)**: 对所有 $x \in A$, 都有 xRx。
- **对称性 (Symmetric)**: 对所有 $x, y \in A$, 如果 xRy, 那么 yRx。
- **传递性 (Transitive)**: 对所有 $x, y, z \in A$, 如果 xRy 且 yRz, 那么 xRz。
- **反对称性 (Antisymmetric)**: 对所有 $x, y \in A$, 如果 xRy 且 yRx, 那么 $x = y$。

2.5 等价关系与等价类 (Equivalence Relation & Equivalence Class)

定义 2.4 (等价关系). 集合 A 上的一个二元关系 R 如果是自反的、对称的且传递的，则称 R 是一个**等价关系**。

示例：

- "=" 是 $\mathbb{N}$ 上的等价关系。
- "x 与 y 是同代人" 是人类集合上的等价关系。

定义 2.5 (等价类 (Equivalence Class)). 给定集合 A 上的等价关系 R，对于任何 $x \in A$，x 的**等价类**是 $[x]_R = \{y \in A | xRy\}$。等价类是 A 中所有与 x 等价的元素的集合。

定理 2.6. 如果 R 是集合 A 上的等价关系，则每个 $a \in A$ 恰好属于一个等价类。等价类的集合构成了集合 A 的一个**划分** *(Partition)*。

示例：在整数集 $\mathbb{Z}$ 上定义关系 R 为 $aRb \iff a \pmod 4 = b \pmod 4$。等价类为 $[0]_R, [1]_R, [2]_R, [3]_R$，它们划分了 $\mathbb{Z}$。**应用**：软件测试中的等价类划分。

2.6 偏序关系 (Partial Order Relation)

定义 2.7 (偏序关系). 集合 A 上的一个二元关系 R 如果是自反的、反对称的且传递的，则称 R 是一个**偏序关系**。

- **极小元 (Minimal Element)**: $x \in A$, 如果不存在其他 $y \in A$ 使得 yRx。
- **极大元 (Maximal Element)**: $x \in A$, 如果不存在其他 $y \in A$ 使得 xRy。

示例：

- $\leq$ 是 $\mathbb{N}$ 上的偏序关系，0 是极小元，没有极大元。
- "x 可被 y 整除" 在正整数集上是偏序关系。
- $\subseteq$ 在 $\mathcal{P}(\mathbb{N})$ 上是偏序关系，$\emptyset$ 是极小元，$\mathbb{N}$ 是极大元。

2.7 全序关系 (Total Order Relation)

定义 2.8 (全序关系 (或线性序)). 集合 A 上的一个偏序关系 R 如果对于任何 $x, y \in A$，都有 xRy 或 yRx，则称 R 是一个**全序关系**。直观地，全序关系中任意两个元素都是可比较的。

示例：$\leq$ 在 $\mathbb{N}$ 上是全序关系。但 "x 可被 y 整除" 在正整数集上不是全序关系 (例如 2 和 3 不可比较)。

3 函数 (Function)

函数是一种特殊的二元关系，它排除了“一对多”的情况。

3.1 定义与术语

定义 3.1 (函数). 从集合 X 到集合 Y 的一个**函数** f 是 X 和 Y 之间的一个二元关系 R，满足以下两个条件：

1. 对任意 $x \in X$，存在 $y \in Y$ 使得 xRy。
2. 如果 $y, z \in Y$ 使得 xRy 且 xRz，那么 $y = z$。

一个函数从集合 X 到集合 Y 为 X 中的每个元素恰好分配 Y 中的一个元素。我们通常用 $f : X \to Y$ 表示 f 是一个从 X 到 Y 的函数，用 $f(x) = y$ 表示 $\langle x, y\rangle \in f$ 。

- **定义域 (Domain)**: 输入值的集合 X。
- **陪域 (Codomain)**: 可能输出值的集合 Y。
- **值域 (Range)**: 实际输出值的集合，是 Y 的一个子集。

定义 3.2 (n 元函数). 如果函数 f 的定义域是笛卡尔积 $A_1 \times A_2 \times \cdots \times A_n$，则称 f 是一个 ***n* 元函数**。它将有序 n 元组映射到其陪域中的元素。

3.2 函数的类型 (Types of Functions)

给定函数 $f : X \to Y$：

- **单射 (Injective / one-to-one)**: 陪域中的每个元素最多被定义域中的一个元素映射到。即，对所有 $x_1, x_2 \in X$，如果 $f(x_1) = f(x_2)$，则 $x_1 = x_2$。
- **满射 (Surjective / onto)**: 陪域中的每个元素至少被定义域中的一个元素映射到。即，对任意 $y \in Y$，存在 $x \in X$ 使得 $f(x) = y$。
- **双射 (Bijective)**: 函数既是单射又是满射。陪域中的每个元素恰好被定义域中的一个元素映射到。

3.3 复合函数 (Composite Function)

给定 $f : X \to Y$ 和 $g : Y \to Z$，**复合函数**记作 $g \circ f : X \to Z$，定义为 $(g \circ f)(x) = g(f(x))$ 对所有 $x \in X$。

4 数学定义与证明 (Mathematical Definitions & Proof)

4.1 归纳定义 (Inductive Definition)

归纳定义通过指定基本元素和从现有元素构造新元素的规则来定义一个集合。示例：自然数集 $\mathbb{N}$ 的归纳定义：

(i) $0 \in \mathbb{N}$

(ii) 对任意 n，如果 $n \in \mathbb{N}$，那么 $n+1 \in \mathbb{N}$

(iii) 只有通过有限次迭代 (i) 和 (ii) 生成的 n 才属于 $\mathbb{N}$。

这等价于：$\mathbb{N}$ 是满足条件 (i) 和 (ii) 的最小归纳子集。归纳定义总是意味着我们寻找满足给定规则的最小集合。

4.2 归纳证明 (Proof by Induction)

对于归纳定义的集合，要证明其所有元素都具有某个性质 P，可以使用归纳证明法。令 P 是一个性质，$P(x)$ 表示 x 具有性质 P。如果：

(i) $P(0)$ （基础情况）

(ii) 对任意 $n \in \mathbb{N}$，如果 $P(n)$，那么 $P(n+1)$ （$n+1$ 是 n 的后继）

那么，对任意 $n \in \mathbb{N}$，$P(n)$ 都成立。**模板**:

- **基础情况 (Base case)**: 证明 $P(n)$ 对 n 的最小值成立，例如 $P(n_0)$ 成立。
- **归纳假设 (Induction Hypothesis)**: 假设对任意正整数 $k \geq n_0$，$P(k)$ 成立。
- **归纳步骤 (Inductive Step)**: 证明 $P(k+1)$ 成立。

4.3 递归定义 (Recursive Definition)

函数 f 的递归定义是根据函数在某些先前点的值来定义其在自然数 n 处的值。例如，令 g, h 是 $\mathbb{N}$ 上的已知函数，它们定义了 $\mathbb{N}$ 上的函数 f: $f(0) = g(0)$ $f(n+1) = h(f(n))$ 示例：斐波那契数列的递归定义：$F(0) = 0, F(1) = 1, F(n) = F(n-1) + F(n-2)$ 对 $n \geq 2$。

4.4 反证法 (Proof by Contradiction)

要证明一个命题，假设其否定为真，并由此导出一个矛盾。示例：证明 $\sqrt{2}$ 是无理数；证明素数有无限多个。

第二部分　命题逻辑 (Propositional Logic - PL)

命题逻辑是研究命题及其逻辑关系的逻辑分支。

5　命题逻辑的语法 (Syntax of PL)

语法定义了命题逻辑语言的“文法规则”。

5.1 命题与联结词 (Propositions & Connectives)

定义 5.1 (命题 (Proposition)). 一个**命题**是一个可以判断为真或假的陈述句。

- **原子命题 (Atomic Proposition)**: 不能分解为更小命题的命题 (例如，“雪是白的”)。通常用小写字母 $p, q, r, \ldots$ 表示。

- **复合命题 (Compound Proposition)**: 由多个原子命题通过逻辑联结词组合而成的命题。通常用大写字母 $A, B, C, \Phi, \Psi, \ldots$ 表示。

定义 5.2 (逻辑联结词 (Logical Connectives))**.** 连接多个命题以形成复合命题的词或符号。

- **否定** ***(Negation)****:* $\neg$ *(并非)*
- **合取** ***(Conjunction)****:* $\wedge$ *(并且)*
- **析取** ***(Disjunction)****:* $\vee$ *(或者)*
- **蕴含** ***(Implication)****:* $\rightarrow$ *(如果... 那么...)*
- **等价** ***(Equivalence)****:* $\leftrightarrow$ *(当且仅当)*

5.2 形式语言与命题逻辑语言 L^P

定义 5.3 (形式语言 (Formal Language))**.** 一个形式语言由一个**字母表** *(Alphabet,* 符号集合*)* 和一套称为**形式文法** *(Formal Grammar)* 的规则组成，这些规则规定了如何从字母表中的符号构成合法的**字符串** *(String,* 符号的有限序列*)*。

命题逻辑语言 L^P 的字母表包括：

- 原子命题符号：$p, q, r, \ldots$
- 逻辑联结词符号：$\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- 标点符号：(和)

定义 5.4 (表达式 (Expression))**.** 由 L^P 字母表中的符号组成的有限字符串。

5.3 合式公式 (Well-Formed Formulas - WFFs)

命题逻辑中的合式公式 *(WFFs)* 是通过以下构造规则有限次应用得到的表达式，并且只有这些表达式才是合式公式：

(i) **原子规则***:* 每个原子命题 $p, q, r, \ldots$ 都是一个合式公式。

(ii) **否定规则***:* 如果 ϕ 是一个合式公式，那么 $(\neg\phi)$ 也是一个合式公式。

(iii) **二元联结词规则***:* 如果 ϕ 和 ψ 是合式公式，那么 $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, 和 $(\phi \leftrightarrow \psi)$ 都是合式公式。

这可以用 **巴科斯范式** ***(BNF)*** 简洁地表示为：$\phi ::= p|(\neg\phi)|(\phi \wedge \phi)|(\phi \vee \phi)|(\phi \rightarrow \phi)|(\phi \leftrightarrow \phi)$

5.4 分析树 (Parse Tree)

定义 5.5 (分析树)**.** L^P 中公式的**分析树**是一个树，使得：

- 根节点是该公式。
- 叶节点是原子公式。
- 每个内部节点都是通过对其子节点应用某个构造规则形成的。

定理 5.6. L^P 的一个表达式是合式公式当且仅当它有一个分析树。每个合式公式都有唯一的分析树。

应用：编译器中的语法分析，自然语言处理。

5.5 子公式与主联结词 (Subformula & Leading Connective)

定义 5.7 (子公式 (Subformula))**.** 如果公式 G 出现在公式 F 内部（即 G 是 F 分析树中的一个节点），则 G 是 F 的**子公式**。如果 $G \neq F$，则 G 是 F 的**真子公式**。

定义 5.8 (直接子公式 (Immediate Subformula) 与主联结词 (Leading Connective))**.** 公式在其分析树中的子节点称为**直接子公式**，用于连接这些子节点的联结词称为**主联结词**。

5.6 公式的结构与唯一可读性

引理 5.9. L^P 的合式公式是非空表达式。每个 L^P 的合式公式都有相同数量的左括号和右括号。

引理 5.10. L^P 中合式公式的每个真前缀（*proper prefix*）的左括号多于右括号。每个合式公式的真后缀（*proper suffix*）的右括号多于左括号。因此，真前缀和真后缀都不是合式公式。

定理 5.11 (唯一可读性定理 (Unique Readability Theorem))**.** 每个合式公式都有唯一的构造方式。

5.7 约定与优先级 (Conventions & Precedence)

为了可读性，通常省略最外层括号。**联结词优先级** *(*从高到低，括号优先级最高*)*:

1. 括号
2. $\neg$
3. $\wedge$
4. $\vee$
5. $\rightarrow$
6. $\leftrightarrow$

联结词通常是**右结合**的，例如 $p \rightarrow q \rightarrow r$ 表示 $p \rightarrow (q \rightarrow r)$。

6 命题逻辑的语义 (Semantics of PL)

语义定义了命题逻辑公式的意义，即它们的真值。

6.1 真值表 (Truth Table)

原子命题的真值为真 *(1)* 或假 *(0)*。复合命题的真值取决于其原子命题和逻辑联结词。

- $\neg p$*:* 当 p 为假时为真，当 p 为真时为假。
- $p \wedge q$*:* 仅当 p 和 q 都为真时为真。

- $p \vee q$: 当 p 或 q (或两者) 为真时为真 (包含或)。
- $p \to q$: 仅当 p 为真且 q 为假时为假。换句话说，“并非 (p 为真且 q 为假)”。
- $p \leftrightarrow q$: 当 p 和 q 具有相同的真值时为真。

注意：自然语言中的联结词 (如“并且”、“或者”) 可能与形式逻辑中的对应词在含义上不完全相同。例如，自然语言中的“并且”可能暗示时间顺序，而逻辑联结词只关心真值。

6.2 真值指派 (Truth Valuation)

定义 6.1 (真值指派). 一个**真值指派** (或赋值) 是一个函数 $v : Atom(L^P) \to \{0, 1\}$，它为每个原子命题分配一个真值。公式 A 在指派 v 下的真值记为 A^v，它是根据联结词的真值表递归定义的。

定理 6.2. 在固定的真值指派 v 下，每个合式公式 $\alpha \in Form(L^P)$ 都有一个在 $\{0, 1\}$ 中的值 α^v。由于公式的唯一可读性，公式在任何真值指派 v 下只有一个真值。

6.3 公式的性质

令 $A \in Form(L^P)$：

- **重言式/永真式** ***(Tautology)***: 如果对每个真值指派 v，$A^v = 1$。(例如 $p \vee \neg p$)
- **矛盾式/永假式** ***(Contradiction)***: 如果对每个真值指派 v，$A^v = 0$。(例如 $p \wedge \neg p$)
- **可满足的** ***(Satisfiable)***: 如果存在某个真值指派 v 使得 $A^v = 1$。

可以使用推理、真值表或**赋值树** ***(Valuation Tree)*** 来确定公式的这些性质。

6.4 公式集合的可满足性

令 $\Sigma \subseteq Form(L^P)$ (一个合式公式的集合)。定义：如果对每个 $B \in \Sigma$，$B^v = 1$，则 $\Sigma^v = 1$，否则为 0。Σ 是**可满足的**当且仅当存在某个指派 v 使得 $\Sigma^v = 1$；我们称 v 满足 Σ。

6.5 逻辑等价 (Logical Equivalence)

定义 6.3 (逻辑等价). 两个公式 A 和 B 是**逻辑等价的**，记作 $A \equiv B$，当且仅当它们在任何指派下具有相同的值。这意味着：

- 对每个真值指派 v，$A^v = B^v$。
- A 和 B 的真值表最终列相同。
- $A \leftrightarrow B$ 是一个重言式。

重要的逻辑等价律:

- 同一律, 支配律, 幂等律, 双重否定律, 交换律, 结合律, 分配律, 德摩根定律, 吸收律, 否定律, 蕴含律, 逆否律, 等价律。

定理 6.4 (代换定理 (Substitution Theorem)). *1.* 如果 A 是一个重言式，B 是 A 的一个代换实例（将 A 中的原子命题统一替换为公式），那么 B 也是一个重言式。

2. 如果公式 A 包含子公式 C，且 $C \equiv D$，那么将 A 中 C 的某些（不必全部）出现替换为 D 得到公式 B，则 $A \equiv B$。

逻辑等价是 $Form(L^P)$ 上的一个等价关系（自反、对称、传递）。

应用：证明代码片段或电路的语义等价性。

7 命题逻辑的语义后承 (Semantic Entailment in PL)

语义后承处理逻辑推论的概念。

7.1 定义

定义 7.1 (语义后承/逻辑后承 (Semantic Entailment / Logical Consequence))**.** 令 Σ 为一个公式集合（前提），A 为一个公式（结论）。我们说 Σ **语义后承** A，或 A 是 Σ 的**逻辑后承**，记作 $\Sigma \models A$，当且仅当对所有真值指派 v，如果 $\Sigma^v = 1$（即 Σ 中的所有公式在 v 下都为真），那么 $A^v = 1$。

$\Sigma \not\models A$ 表示存在一个真值指派 v 使得 $\Sigma^v = 1$ 但 $A^v = 0$。

7.2 证明与反驳后承

证明 $\Sigma \models A$*:*

- **直接证明***:* 对于每个使得所有前提为真的真值指派，证明结论也为真。
- **使用真值表***:* 考虑真值表中所有前提都为真的行，验证结论在这些行中也为真。
- **反证法***:* 假设 $\Sigma \not\models A$，即存在一个指派 v 使得 $\Sigma^v = 1$ 且 $A^v = 0$，然后导出一个矛盾。

反驳 $\Sigma \models A$ *(*即证明 $\Sigma \not\models A$*):* 找到一个真值指派 v，使得 Σ 中的所有前提都为真，而结论 A 为假。

7.3 后承的性质

- $A \models B$ 当且仅当 $A \to B$ 是重言式。
- $A \equiv B$ 当且仅当 $A \models B$ 且 $B \models A$。
- $\emptyset \models A$ 意味着 A 是重言式。
- $A_1, \ldots, A_n \models A$ 当且仅当 $\emptyset \models (A_1 \wedge \cdots \wedge A_n) \to A$。

7.4 联结词的完备集 (Adequate Set of Connectives)

定义 7.2 (完备集)**.** 一个联结词集合被称为是**完备的**，当且仅当每个合式公式都逻辑等价于一个仅使用该集合中联结词的合式公式。或者说，每个 n 元联结词都可以用该完备集中的联结词来定义。

定理 7.3. $\{\neg, \wedge, \vee\}$ 是一个完备的联结词集合。$\{\neg, \wedge\}$, $\{\neg, \vee\}$ 和 $\{\neg, \to\}$ 也都是完备集。

例如，$\to$ 可以用 $\neg$ 和 $\vee$ 定义：$p \to q \equiv \neg p \vee q$。存在 2^{2^n} 个不同的 n 元联结词。例如，一元联结词有 *4* 个，二元联结词有 *16* 个。重要的二元联结词包括：

- 或非 ***(Joint denial / NOR)****:* $p \downarrow q \equiv \neg(p \vee q)$

- 与非 ***(Alternative denial / NAND)***: $p \uparrow q \equiv \neg(p \wedge q)$
- 异或 ***(Exclusive disjunction / XOR)***: $p \oplus q \equiv \neg(p \leftrightarrow q)$

$\{\downarrow\}$ 和 $\{\uparrow\}$ 各自都是完备集。

8 命题逻辑的证明系统 (Proof Systems in PL)

证明系统提供了一种纯句法的方式，通过公理和推理规则从前提推导出结论。

定义 8.1 (形式证明系统/演绎系统 (Formal Proof System / Deductive System))**.** 由**语言部分**（字母表、符号、公式）和**推理部分**（公理、推理规则）组成。一个**证明**是从前提集合和公理集合出发，通过有限次机械地应用推理规则得到新公式的过程。

记号 $\Sigma \vdash A$ 表示“存在一个以 Σ 为前提，以 A 为结论的证明”。这是句法概念，区别于语义后承 $\Sigma \models A$。

8.1 希尔伯特式系统 (Hilbert-style System)

该系统特点是公理多，推理规则少（通常只有分离规则）。证明是公式的线性序列。

8.1.1 语言

字母表通常只包含 $\neg, \rightarrow$ 和原子命题。其他联结词可以通过这两个定义。公式的定义与之前相同（见定义 *5.3*，但只考虑这两个联结词）。

8.1.2 公理与推理规则

A, B, C 为任意合式公式。

公理 8.1 (公理模式)**.**

- $A_1 : A \rightarrow (B \rightarrow A)$
- $A_2 : (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
- $A_3 : (\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A)$

规则 8.1 (分离规则 (Modus Ponens - MP))**.** 从 A 和 $A \rightarrow B$ 可以推出 B。记作：$\dfrac{A, A \rightarrow B}{B}$。

8.1.3 形式证明 (Formal Proof)

定义 8.2 (希尔伯特系统中的证明)**.** H 中公式 A 的一个**证明**是一个有限的公式序列 $A_1, A_2, \ldots, A_n$，使得 $A_n = A$，并且对于任意 $i \leq n$，A_i

- 是 H 中的一个公理，或者
- 是前提集合 Σ 中的一个前提，或者
- 是由 A_j, A_k $(j, k < i)$ 通过 *MP* 规则推导出来的 *(*例如 $A_k = A_j \rightarrow A_i$*)*。

如果公式 A 有一个前提为 Σ 的证明，我们称“A 可以从 Σ 证明”，记作 $\Sigma \vdash_H A$ 或简称 $\Sigma \vdash A$。如果 $\Sigma = \emptyset$，则 $\Sigma \vdash A$ 简化为 $\vdash A$，表示 A 是 H 中的一个**定理** *(Theorem)*。如果 $A \in \Sigma$，则 $\Sigma \vdash A$ *(*称为**假定规则** ***Assumption****)*。

定理 8.3 (H 中的定理)**.** $\vdash_H A \to A$。

证明. (1) $(A \to ((A \to A) \to A)) \to ((A \to (A \to A)) \to (A \to A))$ (Axiom A_2)

(2) $A \to ((A \to A) \to A)$ (Axiom A_1)

(3) $(A \to (A \to A)) \to (A \to A)$ (MP: (1), (2))

(4) $A \to (A \to A)$ (Axiom A_1)

(5) $A \to A$ (MP: (3), (4))

□

证明通常比较复杂，为了简化，引入了导出规则。

8.1.4 导出规则 (Derived Rules)

规则 8.2 (演绎定理 (Deduction Rule/Theorem))**.** 对任意公式集合 Γ 和公式 A, B，$\Gamma \cup \{A\} \vdash B$ 当且仅当 $\Gamma \vdash A \to B$。特别地，$\{A\} \vdash B$ 当且仅当 $\vdash A \to B$。

规则 8.3 (逆否规则 (Contrapositive Rule))**.** 如果 $\Gamma \vdash \neg B \to \neg A$，则 $\Gamma \vdash A \to B$。*(此规则基于 A_3) 以及*如果 $\Gamma \vdash A \to B$，则 $\Gamma \vdash \neg B \to \neg A$。

定理 8.4. $\vdash (A \to B) \to ((B \to C) \to (A \to C))$

证明. 使用演绎定理：

(1) $A \to B$ (Assumption 1)

(2) $B \to C$ (Assumption 2)

(3) A (Assumption 3)

(4) B (MP from (1) and (3))

(5) C (MP from (2) and (4))

(6) $\{A \to B, B \to C\} \vdash A \to C$ (Deduction Thm on (3)-(5))

(7) $\{A \to B\} \vdash (B \to C) \to (A \to C)$ (Deduction Thm on (2)-(6))

(8) $\vdash (A \to B) \to ((B \to C) \to (A \to C))$ (Deduction Thm on (1)-(7))

□

规则 8.4 (传递规则 (Transitivity Rule))**.** 如果 $\Gamma \vdash A \to B$ 且 $\Gamma \vdash B \to C$，则 $\Gamma \vdash A \to C$。

定理 8.5. $\vdash (A \to (B \to C)) \to (B \to (A \to C))$

规则 8.5 (前提交换规则 (Exchange of Antecedent Rule))**.** 如果 $\Gamma \vdash A \to (B \to C)$，则 $\Gamma \vdash B \to (A \to C)$。

定理 8.6. $\vdash \neg A \to (A \to B)$

证明. 使用演绎定理：

(1) $\neg A$ (Assumption 1)

(2) A (Assumption 2)

(3) $\vdash \neg A \to (\neg B \to \neg A)$ (Axiom A_1)

(4) $\neg B \to \neg A$ (MP from (1) and (3))

(5) $\vdash (\neg B \to \neg A) \to (A \to B)$ (Axiom A_3)

(6) $A \to B$ (MP from (4) and (5))

(7) B (MP from (2) and (6))

(8) $\{\neg A\} \vdash A \to B$ (Deduction Thm on (2)-(7))

(9) $\vdash \neg A \to (A \to B)$ (Deduction Thm on (1)-(8))

□

定理 8.7 (双重否定律)**.** $\vdash \neg\neg A \to A$ 和 $\vdash A \to \neg\neg A$。

证明. **Part 1:** $\vdash \neg\neg A \to A$

(1) $\neg\neg A \to (\neg A \to \neg\neg\neg A)$ (Axiom A_1)

(2) $(\neg A \to \neg\neg\neg A) \to (\neg\neg A \to A)$ (Axiom A_3, substituting $\neg A$ for A_{inA_3} and A for B_{inA_3})

(3) $\neg\neg A \to (\neg\neg A \to A)$ (Derived by Hypothetical Syllogism (HS) from (1) and (2). HS states: if $\vdash P \to Q$ and $\vdash Q \to R$, then $\vdash P \to R$. HS is derived using A_1, A_2, MP. More explicitly: (a) $(Q \to R) \to (P \to (Q \to R))$ (Axiom A_1) (b) $P \to (Q \to R)$ (MP using (2) and (a)) (c) $(P \to (Q \to R)) \to ((P \to Q) \to (P \to R))$ (Axiom A_2) (d) $(P \to Q) \to (P \to R)$ (MP using (b) and (c)) (e) $P \to R$ (MP using (1) and (d)). Here $P = \neg\neg A$, $Q = (\neg A \to \neg\neg\neg A)$, $R = (\neg\neg A \to A)$.)

(4) $(\neg\neg A \to (\neg\neg A \to A)) \to ((\neg\neg A \to \neg\neg A) \to (\neg\neg A \to A))$ (Axiom A_2)

(5) $(\neg\neg A \to \neg\neg A) \to (\neg\neg A \to A)$ (MP from (3), (4))

(6) $\neg\neg A \to \neg\neg A$ (Theorem $\vdash X \to X$)

(7) $\neg\neg A \to A$ (MP from (6), (5))

Part 2: $\vdash A \to \neg\neg A$

(1) $\vdash \neg\neg\neg A \to \neg A$ (Instance of Part 1 just proved: $\neg\neg X \to X$ with $X/\neg A$)

(2) $\vdash (\neg\neg\neg A \to \neg A) \to (A \to \neg\neg A)$ (Axiom A_3, substituting $\neg\neg A$ for A_{inA_3} and A for B_{inA_3})

(3) $\vdash A \to \neg\neg A$ (MP from (1), (2))

□

规则 8.6 (双重否定规则 (Double Negation Rule))**.** *From Theorem 8.7, we can establish the derived rules:* $\dfrac{\Sigma \vdash A}{\Sigma \vdash \neg\neg A}$ *and* $\dfrac{\Sigma \vdash \neg\neg A}{\Sigma \vdash A}$.

定理 8.8. $\vdash (\neg A \to \mathit{false}) \to A$ *(其中 false 通常指某个特定的矛盾式, e.g.,* $\neg(P \to P)$*).*

证明. Let false be defined as $\neg(P \to P)$ for some formula P. We know $\vdash P \to P$ (Theorem $\vdash X \to X$). So, $\vdash \neg\neg(P \to P)$ (by Theorem 8.7 Part 2, applied to $P \to P$). This means $\vdash \neg(\text{false})$. We want to prove $(\neg A \to \text{false}) \to A$. By Deduction Theorem, assume $\neg A \to \text{false}$.

(1) $\neg A \to$ false (Assumption)

(2) $\vdash (\neg A \to \text{false}) \to (\neg(\text{false}) \to \neg\neg A)$ (Instance of A_3: $(\neg X \to \neg Y) \to (Y \to X)$, with X/A and $Y/$false. Wait, this is $(\neg A \to \text{false}) \implies (\neg\text{false} \implies \neg\neg A)$ if false means $\neg Y$. Let $X = A$, $Y = \neg$false. $A_3 : (\neg A \to \neg(\neg\text{false})) \to (\neg\text{false} \to A)$. Let's use A_3 as $(\neg X \to \neg Y) \to (Y \to X)$. Let $X = A$ and $Y = T$ where $T = P \to P$. So false $= \neg T$. The assumption is $\neg A \to \neg T$. This is an instance of $\neg X \to \neg Y$. So, by A_3, we can conclude $T \to A$.

(3) $T \to A$ (From (1) being $\neg A \to \neg T$, and A_3)

(4) $\vdash T$ (Since $T = P \to P$ is a theorem)

(5) A (MP from (3), (4))

So, $\{\neg A \to \text{false}\} \vdash A$. By Deduction Theorem, $\vdash (\neg A \to \text{false}) \to A$. □

规则 8.7 (归谬法 (Reductio ad Absurdum))**.** 如果 $\Sigma \vdash \neg A \to$ *false (i.e.* $\neg A$ *leads to a contradiction),* 则 $\Sigma \vdash A$。

定理 8.9. $\vdash (A \to \neg A) \to \neg A$。

证明. We use the Deduction Theorem.

(1) $A \to \neg A$ (Assumption for outer $\to I$)

(2) $\neg\neg A$ (Assumption for inner $\to I$ or RAA reasoning)

(3) $\vdash \neg\neg A \to A$ (Theorem 8.7, Part 1)

(4) A (MP from (2), (3))

(5) $\neg A$ (MP from (1), (4))

Steps (2)-(5) show that $\{A \to \neg A, \neg\neg A\} \vdash \neg A$. By Deduction Theorem, $\{A \to \neg A\} \vdash \neg\neg A \to \neg A$. (Let this be line (6)) Now we need to show that if $\neg\neg A \to \neg A$, then $\neg A$.

(6) $\{A \to \neg A\} \vdash \neg\neg A \to \neg A$ (Derived above)

(7) $\vdash (\neg\neg A \to \neg A) \to ((\neg A \to \neg\neg A) \to ((\neg A \to \neg A) \to \ldots))$ (This is not direct.) A frequently used theorem schema is $(X \to Y) \to ((Y \to \neg X) \to \neg X)$, or related forms for RAA. However, a simpler continuation from line (6) using properties of implication and negation: We have shown $\{A \to \neg A\} \vdash \neg\neg A \to \neg A$. Let $P_1 = A \to \neg A$. Let $P_2 = \neg\neg A \to \neg A$. So $P_1 \vdash P_2$. We also have $\vdash (\neg\neg A \to \neg A) \to ((\neg A \to A) \to \neg\neg A)$ (Instance of A_2 and A_3 used carefully). No. The RAA pattern can be used: from $\{A \to \neg A\}$, assume $\neg\neg A$. This leads to $\neg A$ (as shown in steps 2-5). We have both $\neg\neg A$ and $\neg A$. This is a contradiction. Let F_0 represent a

contradiction (e.g. $\neg(P \to P)$). $\{A \to \neg A, \neg\neg A\} \vdash F_0$. So $\{A \to \neg A\} \vdash \neg\neg A \to F_0$ (by DT). Using $\vdash (\neg X \to \text{false}) \to X$ with $X/\neg A$: We have $\neg\neg A \to F_0$. This is $\neg(\neg A) \to F_0$. So this gives $\neg A$. Thus, $\{A \to \neg A\} \vdash \neg A$.

Finally, by Deduction Theorem on the outermost assumption: $\vdash (A \to \neg A) \to \neg A$. □

8.2 自然演绎系统 (Natural Deduction System - ND)

该系统公理少（甚至没有），推理规则多。证明通常是树状的，并使用子证明（在有限范围内做出的假设）。

8.2.1 语言

通常包含所有标准联结词：$\neg, \wedge, \vee, \to, \leftrightarrow$。公式定义同 *5.3*。

8.2.2 推理规则

规则通常成对出现：引入规则 *(增加联结词)* 和排除规则 *(移除联结词)*。

规则 8.8 (自反性/前提 (Reflexivity/Premise)). $\Sigma \cup \{\alpha\} \vdash_{ND} \alpha$。*(一个前提可以直接写出)*

合取 *(∧)* 规则:

- **∧-引入** *(∧I)*: 从 $\Sigma\alpha$ 和 $\Sigma\beta$，可以推出 $\Sigma(\alpha \wedge \beta)$。记作：$\dfrac{\alpha \quad \beta}{\alpha \wedge \beta}$
- **∧-排除** *(∧E)*: 从 $\Sigma(\alpha \wedge \beta)$，可以推出 $\Sigma\alpha$ 和 $\Sigma\beta$。记作：$\dfrac{\alpha \wedge \beta}{\alpha}$ 和 $\dfrac{\alpha \wedge \beta}{\beta}$

蕴含 *(→)* 规则:

- **→-引入** *(→ I)*: 如果通过假设 α *(在子证明中)* 能推出 β *(即 $\Sigma, \alpha\beta$)*，那么可以推出 $\Sigma(\alpha \to \beta)$。子证明通常用方框表示，方框内的内容不能直接用于方框外，只能作为一个整体使用。记作：

$$\begin{array}{|l} \alpha \quad \textit{(Assumption)} \\ \vdots \\ \beta \\ \hline \alpha \to \beta \end{array}$$

- **→-排除** *(→ E / **Modus Ponens**)*: 从 $\Sigma(\alpha \to \beta)$ 和 $\Sigma\alpha$，可以推出 $\Sigma\beta$。记作：$\dfrac{\alpha \to \beta \quad \alpha}{\beta}$

析取 *(∨)* 规则:

- **∨-引入** *(∨I)*: 从 $\Sigma\alpha$，可以推出 $\Sigma(\alpha \vee \beta)$ *(和 $\Sigma(\beta \vee \alpha)$)*。记作：$\dfrac{\alpha}{\alpha \vee \beta}$ 和 $\dfrac{\alpha}{\beta \vee \alpha}$
- **∨-排除** *(∨E / **Proof by Cases**)*: 如果 $\Sigma(\alpha_1 \vee \alpha_2)$，并且 $\Sigma, \alpha_1\beta$ 且 $\Sigma, \alpha_2\beta$，那么可以推出 $\Sigma\beta$。记作：

$$\dfrac{\begin{array}{ccc} & [\alpha_1] & [\alpha_2] \\ & \vdots & \vdots \\ \alpha_1 \vee \alpha_2 & \beta & \beta \end{array}}{\beta}$$

否定 *(¬)* 规则: 用 $\bot$ *(读作"bottom"或"falsum")* 代表任意矛盾。

- **¬-引入** *(¬I)*: 如果假设 α 能推出矛盾 $\bot$ (即 $\Sigma, \alpha \vdash \bot$),那么可以推出 $\Sigma(\neg\alpha)$。记作：
$$\begin{array}{|l} \alpha \quad \textit{(Assumption)} \\ \vdots \\ \bot \\ \hline \neg\alpha \end{array}$$
- **¬-排除** *(¬E)*: 从 $\Sigma\alpha$ 和 $\Sigma(\neg\alpha)$，可以推出 $\Sigma \bot$。记作：$\dfrac{\alpha \quad \neg\alpha}{\bot}$
- **双重否定排除** *(¬¬E)*: 从 $\Sigma(\neg\neg\alpha)$，可以推出 $\Sigma\alpha$。记作：$\dfrac{\neg\neg\alpha}{\alpha}$
- **矛盾排除** *($\bot$ E / **Ex Falso Quodlibet**)*: 从 $\Sigma \bot$，可以推出 $\Sigma\alpha$ *(对任意 α)*。记作：$\dfrac{\bot}{\alpha}$ *(此规则有时被认为是多余的，可以用其他规则导出)*。

8.2.3 导出规则 (Derived Rules)

如果有一个证明 $\Gamma \vdash_{ND} \alpha$，我们可以将其视为一个导出规则 $\dfrac{\Gamma}{\alpha}$。使用导出规则可以简化证明，但不会增加可证明的内容。

- **取拒式** ***(Modus Tollens - MT)***: 从 $A \to B$ 和 $\neg B$，可以推出 $\neg A$。记作：$\dfrac{\alpha \to \beta \quad \neg\beta}{\neg\alpha}$ *(MT)*

  Proof of Modus Tollens. We want to show $\{\alpha \to \beta, \neg\beta\}\neg\alpha$.

  1. $\alpha \to \beta$ (Premise)
  2. $\neg\beta$ (Premise)
  3. $\quad\alpha$ (Assumption for $\neg I$)
  4. $\quad\beta$ ($\to E$: 1, 3)
  5. $\quad\bot$ ($\neg E$: 4, 2)
  6. $\neg\alpha$ ($\neg I$: 3-5)

  □

- **双重否定引入** *(¬¬I)*: 从 α，可以推出 $\neg\neg\alpha$。记作：$\dfrac{\alpha}{\neg\neg\alpha}$ *(¬¬I)*

  Proof of Double Negation Introduction. We want to show $\{\alpha\}\neg\neg\alpha$.

  1. α (Premise)
  2. $\quad\neg\alpha$ (Assumption for $\neg I$)
  3. $\quad\bot$ ($\neg E$: 1, 2)
  4. $\neg\neg\alpha$ ($\neg I$: 2-3)

  □

- **排中律** ***(Law of Excluded Middle - LEM)***: $\vdash_{ND} (\alpha \vee \neg\alpha)$ 是一个定理。记作：$\overline{\alpha \vee \neg\alpha}$ *(LEM)*

  Proof of Law of Excluded Middle. We want to show $\alpha \vee \neg\alpha$.

1. $\neg(\alpha \vee \neg\alpha)$ (Assumption for $\neg I$ or PBC)
2. $\quad \alpha$ (Assumption for $\neg I$)
3. $\quad \alpha \vee \neg\alpha$ ($\vee I$: 2)
4. $\quad \bot$ ($\neg E$: 3, 1)
5. $\neg\alpha$ ($\neg I$: 2-4)
6. $\alpha \vee \neg\alpha$ ($\vee I$: 5)
7. $\bot$ ($\neg E$: 6, 1)
8. $\neg\neg(\alpha \vee \neg\alpha)$ ($\neg I$: 1-7)
9. $\alpha \vee \neg\alpha$ ($\neg\neg E$: 8)

□

备注 8.10. *This proof of LEM uses $\neg\neg E$, making it a classical logic proof. In intuitionistic logic, LEM is not generally provable.*

- **反证法/归谬法 *(Proof by Contradiction / Reductio ad Absurdum - PBC)*:** 如果假设 $\neg\alpha$ 导致矛盾 $\bot$，则可以推出 α。记作：

$$\begin{array}{|l} \neg\alpha \quad \textit{(Assumption)} \\ \vdots \\ \bot \\ \hline \alpha \end{array} \quad \textit{(PBC)}$$

Derivation of PBC rule. Assume we have a subproof from $\neg\alpha$ to $\bot$.

1. $\neg\alpha$ (Assumption)
2. $\quad \vdots$ (Steps leading to contradiction)
3. $\bot$ (Derived Contradiction)
4. $\neg\neg\alpha$ ($\neg I$: 1-3, assuming the subproof from 1 to 3 exists)
5. α ($\neg\neg E$: 4)

This shows that PBC is derivable from $\neg I$ and $\neg\neg E$. □

8.2.4 命题逻辑自然演绎示例证明 (Example Proofs in PL Natural Deduction)

[蕴含前提的交换] 证明: $A \to (B \to C) B \to (A \to C)$

证明. 1. $A \to (B \to C)$ (Premise)
2. $\quad B$ (Assumption for $\to I$)
3. $\quad\quad A$ (Assumption for $\to I$)
4. $\quad\quad B \to C$ ($\to E$: 1, 3)
5. $\quad\quad C$ ($\to E$: 4, 2)
6. $\quad A \to C$ ($\to I$: 3-5)

7. $B \to (A \to C)$ ($\to I$: 2-6)

□

[析取三段论 (Disjunctive Syllogism)] 证明: $\{A \vee B, \neg A\}B$

证明. 1. $A \vee B$ (Premise)

2. $\neg A$ (Premise)

3. A (Assumption for $\vee E$, Case 1)

4. $\bot$ ($\neg E$: 3, 2)

5. B ($\bot$ E: 4)

6. B (Assumption for $\vee E$, Case 2)

7. B ($\vee E$: 1, 3-5, 6-6)

□

[蕴含与析取的交互] 证明: $\{A \to B\}(C \vee A) \to (C \vee B)$

证明. 1. $A \to B$ (Premise)

2. $C \vee A$ (Assumption for $\to I$)

3. C (Assumption for $\vee E$, Case 1)

4. $C \vee B$ ($\vee I$: 3)

5. A (Assumption for $\vee E$, Case 2)

6. B ($\to E$: 1, 5)

7. $C \vee B$ ($\vee I$: 6)

8. $C \vee B$ ($\vee E$: 2, 3-4, 5-7)

9. $(C \vee A) \to (C \vee B)$ ($\to I$: 2-8)

□

8.3 归结系统 (Resolution System)

该系统主要用于自动定理证明，通过反驳法工作，适用于合取范式 *(CNF)* 的公式。

8.3.1 文字与子句 (Literals & Clauses)

定义 8.11 (文字 (Literal)). 一个**文字**是一个原子命题或其否定 *(例如 $p, \neg q$)*。

定义 8.12 (子句 (Clause)).

- **析取子句 *(Disjunctive Clause)*:** 文字的析取 *(例如 $p \vee \neg q \vee r$)*。
- **合取子句 *(Conjunctive Clause)*:** 文字的合取 *(在 DNF 中出现)*。

在归结中，我们通常处理析取子句。

8.3.2 范式 (Normal Forms)

定义 8.13 (合取范式 (Conjunctive Normal Form - CNF))**.** 一个公式如果是一系列析取子句的合取，则称其处于**合取范式**。形式：$(L_{11} \vee \cdots \vee L_{1n_1}) \wedge \cdots \wedge (L_{k1} \vee \cdots \vee L_{kn_k})$。

定义 8.14 (析取范式 (Disjunctive Normal Form - DNF))**.** 一个公式如果是一系列合取子句的析取，则称其处于**析取范式**。形式：$(L_{11} \wedge \cdots \wedge L_{1n_1}) \vee \cdots \vee (L_{k1} \wedge \cdots \vee L_{kn_k})$。

定理 8.15. 每个命题逻辑公式都逻辑等价于某个 *CNF* 公式和某个 *DNF* 公式。存在将任意公式转换为 *CNF/DNF* 的步骤：

1. 去除 $\rightarrow$ 和 $\leftrightarrow$。
2. 内移否定：使用双重否定律和德摩根定律将 $\neg$ 作用于原子命题。
3. 应用分配律：将 $\vee$ 分配到 $\wedge$ 上（得到 *CNF*）或将 $\wedge$ 分配到 $\vee$ 上（得到 *DNF*）。

定义 8.16 (主合取/析取范式 (Principal CNF/DNF))**.** 一种特殊的 *CNF/DNF*，其中每个子句都包含公式中所有原子命题恰好一次，且没有重复的子句。

8.3.3 归结推理规则 (Resolution Inference Rule)

对于任意原子命题 p 和析取子句的其余部分 A, B：从 $(A \vee p)$ 和 $(\neg p \vee B)$，可以推出 $(A \vee B)$ *(称为归结式 Resolvent)*。记作：$\dfrac{A \vee p \quad \neg p \vee B}{A \vee B}$ **特殊情况**:

- **单元归结** ***(Unit Resolution)***: $\dfrac{A \vee p \quad \neg p}{A}$
- **矛盾** ***(Contradiction)***: $\dfrac{p \quad \neg p}{\bot}$ *(空子句)*

空子句 $\bot$ *(*或写作 {} 或 $\square$*)* 代表矛盾，因为它不包含任何为真的文字，所以总是假的。

8.3.4 归结证明过程 (Resolution Proof Procedure - 反驳法)

要通过归结反驳法证明 $\Sigma \vdash_{Res} \phi$:

1. **设置反驳**: 目标是证明 $\Sigma \cup \{\neg\phi\} \vdash_{Res} \bot$。即，假设结论的否定为真，并推导出矛盾。
2. **转换为 *CNF***: 将 $\Sigma \cup \{\neg\phi\}$ 中的每个公式都转换为 *CNF*。
3. **子句集**: 将 *CNF* 公式看作一个析取子句的集合 *(*在 $\wedge$ 处分割*)*。例如，$(p \vee q) \wedge (\neg q \vee r) \wedge s$ 对应子句集 $\{\{p, q\}, \{\neg q, r\}, \{s\}\}$。
4. **应用归结规则**: 从得到的子句集中，不断对子句对应用归结规则，直到：
   - 推导出空子句 $\bot$。此时，证明了 $\Sigma \vdash_{Res} \phi$。
   - 无法再应用规则产生新的子句。此时，ϕ 不能从 Σ 证明。

应用：可满足性问题 *(SAT)* 求解器，广泛用于软硬件验证、规划、调度等。也可用于软件依赖解析、符号执行和测试用例生成。

9 命题逻辑的可靠性与完备性 (Soundness & Completeness of PL)

这些是评估证明系统质量的关键元性质，连接了句法可证明性 ($\vdash$) 和语义后承 ($\models$)。

定义 9.1 (可靠性 (Soundness))**.** 一个证明系统是**可靠的**，如果 $\Sigma \vdash A$ 那么 $\Sigma \models A$。这意味着证明系统只能证明那些实际上是逻辑后承的东西（即“真的”结论）。不可靠的系统基本无用。

定义 9.2 (完备性 (Completeness))**.** 一个证明系统是**完备的**，如果 $\Sigma \models A$ 那么 $\Sigma \vdash A$。这意味着所有逻辑上为真的后承都可以在该系统中被证明出来。

定理 9.3. 我们介绍的希尔伯特式系统、自然演绎系统和归结系统对于命题逻辑都是**既可靠又完备的**。

示例：

- **直觉主义自然演绎系统** (没有 $\neg\neg e$ 规则) 是可靠但**不完备**的，因为它无法证明排中律 $p \vee \neg p$。
- 如果一个自然演绎系统额外加入 $p \wedge \neg p$ 作为公理，那么它是完备但**不可靠**的 (因为它可以证明任何命题)。

第三部分 一阶逻辑 (First-Order Logic - FOL)

一阶逻辑（也称谓词逻辑）扩展了命题逻辑，允许对对象、其属性和它们之间的关系进行推理，并使用量词。

10 一阶逻辑的语法 (Syntax of FOL)

10.1 命题逻辑的局限性与一阶逻辑的基本概念

命题逻辑无法表达个体间的关系 (如“爱丽丝嫁给了杰伊”)、概括模式 (如“每个学生都比某个老师年轻”) 或处理无限论域。**一阶逻辑的基本概念**:

- **论域** ***(Domain)***: 一个非空的对象集合，陈述所针对的世界 (例如自然数、人)。
- **个体常元** ***(Constant Symbols)***: 表示论域中特定对象的符号 (例如 *'Alice'*, *'0'*)。
- **个体变元** ***(Variable Symbols)***: 符号 $(x, y, z, \dots)$，作为论域中对象的占位符。
- **谓词符号** ***(Predicate Symbols)***: 表示对象属性 (一元谓词，如 $S(x)$ 表示“x 是学生”) 或对象间关系 (n 元谓词，如 $Y(x, y)$ 表示“x 比 y 年轻”) 的符号。
- **函词符号** ***(Function Symbols)***: 符号 $(f, g, h, \dots)$，表示将对象 (或对象的元组) 映射到对象的函数 (例如 $m(y)$ 表示“y 的母亲”)。每个函词有其**元数** *(Arity)*。
- **量词** ***(Quantifiers)***:
  - **全称量词** ***(Universal Quantifier)***: $\forall$ (对于所有/每一个)。
  - **存在量词** ***(Existential Quantifier)***: $\exists$ (存在/对于某个)。

10.2 一阶逻辑语言 L 的字母表

- **非逻辑符号** ***(Non-logical Symbols)****: (其含义随解释而变)*
  1. 个体常元符号：通常用 $c_1, c_2, \ldots, a, b, \ldots$ 表示。
  2. 谓词符号：通常用大写字母 $P, Q, R, \ldots$ 表示，上标表示元数，如 $P^{(n)}$ 是 n 元谓词。
  3. 函词符号：通常用小写字母 $f, g, h, \ldots$ 表示，上标表示元数，如 $f^{(n)}$ 是 n 元函词。
- **逻辑符号** ***(Logical Symbols)****: (其含义由语法和语义固定)*
  1. 量词符号：$\forall, \exists$。
  2. 个体变元符号：通常用 $x, y, z, x_1, x_2, \ldots$ 表示。
  3. 逻辑联结词符号：$\neg, \wedge, \vee, \rightarrow, \leftrightarrow$。
  4. 标点符号：*(,)* 和逗号。
  5. 等词符号 *(*特殊二元谓词*)*：$=$ *(*可选*)*。

所有一阶语言共享相同的逻辑符号，它们的不同之处仅在于非逻辑符号。

10.3 项 (Terms)

定义 10.1 (项)**.** **项**是表示“对象”的表达式，递归定义如下：

(i) 个体常元符号和个体变元符号是 *(*原子*)* 项。

(ii) 如果 $f^{(n)}$ 是一个 n 元函词符号，$t_1, t_2, \ldots, t_n$ 是项，那么 $f^{(n)}(t_1, t_2, \ldots, t_n)$ 是一个项。

(iii) 没有其他东西是项。

示例：若 *'0'* 是常元，x, y 是变元，$s^{(1)}$ 是一元函词，$+^{(2)}$ 是二元函词，则 $0, x, s(x), +(x, s(y))$ *(*或 $x + s(y)$*)* 都是项。

10.4 原子公式 (Atomic Formulas)

定义 10.2 (原子公式)**.** **原子公式**是表示对象属性或关系的最基本公式，定义如下：

(i) 如果 $P^{(n)}$ 是一个 n 元谓词符号，$t_1, \ldots, t_n$ 是项，那么 $P^{(n)}(t_1, \ldots, t_n)$ 是一个原子公式。

(ii) 如果 t_1, t_2 是项，那么 $t_1 = t_2$ *(*或 $=(t_1, t_2)$*)* 是一个原子公式。

10.5 合式公式 (Well-Formed Formulas - WFFs)

一阶逻辑的合式公式 *(WFFs)* 递归定义如下：

(i) 每个原子公式都是合式公式。

(ii) 如果 α 是一个合式公式，那么 $(\neg\alpha)$ 是一个合式公式。

(iii) 如果 α 和 β 是合式公式，那么 $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$, 和 $(\alpha \leftrightarrow \beta)$ 都是合式公式。

(iv) 如果 α 是一个合式公式，x 是一个体变元，那么 $(\forall x\alpha)$ 和 $(\exists x\alpha)$ 都是合式公式。

10.6 优先级、约定与分析树 (Precedence, Conventions & Parse Trees)

括号决定运算顺序。$\forall x, \exists x$ 和 $\neg$ 具有最高优先级，然后是二元联结词（与命题逻辑中类似）。可以省略最外层括号和量词周围的括号。一阶逻辑公式的**分析树**构造方式与命题逻辑类似，增加了量词节点、谓词节点和函词节点。

10.7 量词的辖域、自由变元与约束变元 (Scope, Free & Bound Variables)

定义 10.3 (量词的辖域 (Scope of Quantifier))**.** 在公式 $\forall x\alpha$ 或 $\exists x\alpha$ 中，x 是被量化的变元，其**辖域**是公式 α。

定义 10.4 (自由变元与约束变元 (Free and Bound Variables))**.**

- 变元 x 在公式中的一次出现是**自由的** ***(Free)***，当且仅当这次出现不在某个以 x 为量化变元的量词的辖域内。
- 否则，变元的这次出现是**约束的** ***(Bound)***。

紧跟在 $\forall$ 或 $\exists$ 后的变元符号既不是自由的也不是约束的 *(它是量词的一部分)*。一个变元在同一个公式中可能既有自由出现也有约束出现，但每次具体的出现要么是自由的，要么是约束的。

定义 10.5 (语句/闭公式 (Sentence / Closed Formula))**.** 一个没有自由变元的公式称为**语句**或**闭公式**。

定义 10.6 (闭包 (Closure))**.** 如果 $\{x_1, \ldots, x_n\}$ 是公式 α 的所有自由变元，那么：

- **全称闭包** ***(Universal Closure)****:* $\forall x_1 \ldots \forall x_n \alpha$
- **存在闭包** ***(Existential Closure)****:* $\exists x_1 \ldots \exists x_n \alpha$

量词等价式*:*

- $\forall x P(x) \leftrightarrow \neg\exists x \neg P(x)$
- $\exists x P(x) \leftrightarrow \neg\forall x \neg P(x)$
- $\neg\forall x P(x) \leftrightarrow \exists x \neg P(x)$
- $\neg\exists x P(x) \leftrightarrow \forall x \neg P(x)$

在有限论域 $D = \{a_1, \ldots, a_n\}$ 中：$\forall x P(x) \leftrightarrow P(a_1) \wedge \cdots \wedge P(a_n)$ $\exists x P(x) \leftrightarrow P(a_1) \vee \cdots \vee P(a_n)$

10.8 自然语言形式化 (Formalization of Natural Language)

示例：

- “每个学生都知道数学”：$\forall x(S(x) \to K(x, Math))$ *(S(x): x 是学生, K(x,y): x 知道 y, Math 是个体常元)*
- “有些学生知道数学”：$\exists x(S(x) \wedge K(x, Math))$
- “每个学生都比某个老师年轻”：$\forall x(S(x) \to \exists y(I(y) \wedge Y(x, y)))$ *(I(y): y 是老师, Y(x,y): x 比 y 年轻)*

选择合适的论域可以简化公式。

11 一阶逻辑的语义 (Semantics of FOL)

语义为一阶逻辑公式赋予意义和真值。这需要一个解释，并且对于带有自由变元的公式，还需要一个环境。

11.1 解释 (Interpretation)

定义 11.1 (解释 $\mathcal{I}$). 一个**解释** $\mathcal{I}$ (或结构) 包含：

1. 一个非空集合 D，称为 $\mathcal{I}$ 的**论域 *(Domain)*** 或全集。
2. 对每个个体常元符号 c，$\mathcal{I}$ 指定 D 中的一个元素 $c^{\mathcal{I}}$。
3. 对每个 n 元函词符号 $f^{(n)}$，$\mathcal{I}$ 指定一个 n 元函数 $f^{\mathcal{I}} : D^n \to D$。
4. 对每个 n 元谓词符号 $P^{(n)}$，$\mathcal{I}$ 指定一个 n 元关系 $P^{\mathcal{I}} \subseteq D^n$ (即一个 n 元组的集合，对这些元组谓词为真)。

公式的真值依赖于解释。

11.2 环境 (Environment)

定义 11.2 (环境 E). 对于带有自由变元的公式，一个**环境** E (或变元赋值) 是一个函数，它将语言中的每个个体变元符号映射到论域 D 中的一个元素。

解释和环境的组合为每个项、原子公式和公式提供了一个值。

11.3 项的真值 (Value of Terms)

在解释 $\mathcal{I}$ 和环境 E 下，项 t 的值 $t^{(\mathcal{I},E)}$ 定义如下：

- 如果 t 是常元 c，则 $t^{(\mathcal{I},E)} = c^{\mathcal{I}}$。
- 如果 t 是变元 x，则 $t^{(\mathcal{I},E)} = x^E$ (或 $E(x)$)。
- 如果 t 是 $f(t_1, \ldots, t_n)$，则 $t^{(\mathcal{I},E)} = f^{\mathcal{I}}(t_1^{(\mathcal{I},E)}, \ldots, t_n^{(\mathcal{I},E)})$。(函数 $f^{\mathcal{I}}$ 应用于项 t_i 的值)。

项的值总是论域 D 的一个成员。

11.4 原子公式的真值 (Value of Atomic Formulas)

固定解释 $\mathcal{I}$ 和环境 E。原子公式 $P(t_1, \ldots, t_n)$ 的真值 $P(t_1, \ldots, t_n)^{(\mathcal{I},E)}$ 为 $P^{\mathcal{I}}(t_1^{(\mathcal{I},E)}, \ldots, t_n^{(\mathcal{I},E)})$ 的真值 (即，如果由项的值构成的元组属于关系 $P^{\mathcal{I}}$，则为 *1*，否则为 *0*)。原子公式 $(t_1 = t_2)^{(\mathcal{I},E)}$ 为 *1* 当且仅当 $t_1^{(\mathcal{I},E)}$ 和 $t_2^{(\mathcal{I},E)}$ 是 D 中的同一个元素。

11.5 合式公式的真值 (Value of Well-Formed Formulas)

固定解释 $\mathcal{I}$ 和环境 E。合式公式的真值递归定义如下：

(i) 原子公式的真值如上定义。

(ii) $(\neg\alpha)^{(\mathcal{I},E)} = 1$ 当且仅当 $\alpha^{(\mathcal{I},E)} = 0$。

(iii) $(\alpha \wedge \beta)^{(\mathcal{I},E)} = 1$ 当且仅当 $\alpha^{(\mathcal{I},E)} = 1$ 且 $\beta^{(\mathcal{I},E)} = 1$。

(iv) $(\alpha \vee \beta)^{(\mathcal{I},E)} = 1$ 当且仅当 $\alpha^{(\mathcal{I},E)} = 1$ 或 $\beta^{(\mathcal{I},E)} = 1$。

(v) $(\alpha \to \beta)^{(\mathcal{I},E)} = 1$ 当且仅当 $\alpha^{(\mathcal{I},E)} = 0$ 或 $\beta^{(\mathcal{I},E)} = 1$。

(vi) $(\alpha \leftrightarrow \beta)^{(\mathcal{I},E)} = 1$ 当且仅当 $\alpha^{(\mathcal{I},E)} = \beta^{(\mathcal{I},E)}$。

(vii) 为了计算 $(\forall x\alpha)^{(\mathcal{I},E)}$ 和 $(\exists x\alpha)^{(\mathcal{I},E)}$，我们定义一个新的环境 $E[x \mapsto d]$，它与 E 相同，只是将变元 x 赋值为论域元素 d：$E[x \mapsto d](y) = \begin{cases} d & \text{如果} y \text{ 是} x \\ E(y) & \text{如果} y \text{ 不是} x \end{cases}$

(viii) $(\forall x\alpha)^{(\mathcal{I},E)} = 1$ 当且仅当对论域 D 中的**每个**元素 d，都有 $\alpha^{(\mathcal{I},E[x\mapsto d])} = 1$。

(ix) $(\exists x\alpha)^{(\mathcal{I},E)} = 1$ 当且仅当在论域 D 中**存在某个**元素 d，使得 $\alpha^{(\mathcal{I},E[x\mapsto d])} = 1$。

如果一个公式 α 是一个语句（没有自由变元），它的真值仅依赖于解释 $\mathcal{I}$，而与环境 E 无关，此时可以写作 $\alpha^{\mathcal{I}}$。

12 一阶逻辑中的逻辑等价 (Logical Equivalence in FOL)

在一阶逻辑中，我们同样关心公式之间的逻辑等价性。许多命题逻辑中的等价关系在适当调整后依然适用，同时量词也引入了新的等价关系。

12.1 对偶性 (Duality)

量词的对偶性（也称为量词否定等价式）是非常重要的一组等价关系：

- $\neg\forall x A(x) \leftrightarrow \exists x \neg A(x)$
- $\neg\exists x A(x) \leftrightarrow \forall x \neg A(x)$

这意味着，全称量词可以通过存在量词和否定来表达（例如 $\forall x A(x) \leftrightarrow \neg\exists x \neg A(x)$），反之亦然（例如 $\exists x A(x) \leftrightarrow \neg\forall x \neg A(x)$）。因此，从表达能力上讲，一个量词和否定就足够了，另一个量词可以视为简写。

12.2 命题逻辑等价的代换 (Substitution from PL Equivalences)

命题逻辑中的逻辑等价律在一阶逻辑中仍然有效，条件是我们将命题逻辑中的同一个原子命题统一替换为同一个 *FOL* 公式。

示例 12.1. 例如：

- 双重否定律：$\forall x P(x) \leftrightarrow \neg\neg\forall x P(x)$
- 蕴含等价式：$(\forall x P(x) \to \exists y Q(y)) \leftrightarrow (\neg\forall x P(x) \vee \exists y Q(y))$

12.3 量词的交换律 (Commutativity of Quantifiers)

同类型的量词可以交换顺序：

- $\forall x \forall y A(x,y) \leftrightarrow \forall y \forall x A(x,y)$
- $\exists x \exists y A(x,y) \leftrightarrow \exists y \exists x A(x,y)$

注意：不同类型的量词一般不能随意交换顺序，例如 $\forall x \exists y A(x,y)$ 通常不等价于 $\exists y \forall x A(x,y)$。

12.4 量词的分配律 (Distributivity of Quantifiers)

- 全称量词对于合取 *(∧)* 满足分配律：$\forall x(A(x) \land B(x)) \leftrightarrow (\forall x A(x) \land \forall x B(x))$
- 存在量词对于析取 *(∨)* 满足分配律：$\exists x(A(x) \lor B(x)) \leftrightarrow (\exists x A(x) \lor \exists x B(x))$

注意：

- 全称量词对于析取 *(∨)* 一般不满足分配律，即 $\forall x(A(x) \lor B(x))$ 通常不等价于 $(\forall x A(x) \lor \forall x B(x))$。我们只有单向的蕴含：$(\forall x A(x) \lor \forall x B(x)) \to \forall x(A(x) \lor B(x))$。
- 存在量词对于合取 *(∧)* 一般不满足分配律，即 $\exists x(A(x) \land B(x))$ 通常不等价于 $(\exists x A(x) \land \exists x B(x))$。我们只有单向的蕴含：$\exists x(A(x) \land B(x)) \to (\exists x A(x) \land \exists x B(x))$。

12.5 等价示例推导

一个有用的等价关系推导：

$$
\begin{aligned}
\exists x(A(x) \to B(x)) &\leftrightarrow \exists x(\neg A(x) \lor B(x)) && \textit{(蕴含等价式)} \\
&\leftrightarrow \exists x \neg A(x) \lor \exists x B(x) && \textit{(存在量词对析取的分配律)} \\
&\leftrightarrow \neg(\forall x A(x)) \lor \exists x B(x) && \textit{(量词对偶性)} \\
&\leftrightarrow (\forall x A(x)) \to (\exists x B(x)) && \textit{(蕴含等价式)}
\end{aligned}
$$

所以，我们有 $\exists x(A(x) \to B(x)) \leftrightarrow ((\forall x A(x)) \to (\exists x B(x)))$。

12.6 有限论域中的量词消除 (Quantifier Removal in Finite Domains)

如果论域 D 是有限的，例如 $D = \{a_1, a_2, \ldots, a_n\}$，那么量词可以被消除：

- $\forall x P(x) \leftrightarrow P(a_1) \land P(a_2) \land \cdots \land P(a_n)$
- $\exists x P(x) \leftrightarrow P(a_1) \lor P(a_2) \lor \cdots \lor P(a_n)$

12.7 解释示例 (Interpretation Example)

令语言包含一元函词符号 $f^{(1)}$，一元谓词符号 $P^{(1)}$，二元谓词符号 $Q^{(2)}, R^{(2)}$，以及个体常元符号 a。定义一个解释 $\mathcal{I}$ 如下：

- 论域 *(Domain)*: $D = \{2, 3\}$
- 常元解释: $a^{\mathcal{I}} = 2$

- 函词解释: $f^{\mathcal{I}}(2) = 3, f^{\mathcal{I}}(3) = 2$
- 谓词解释: $P^{\mathcal{I}} = \{3\}$, $Q^{\mathcal{I}} = \{\langle 2,2\rangle, \langle 2,3\rangle, \langle 3,2\rangle\}$, $R^{\mathcal{I}} = \{\langle 2,2\rangle, \langle 3,3\rangle\}$

计算公式 $\forall x(P(x) \land Q(x,a))$ 在解释 $\mathcal{I}$ 下的真值:

$$
\begin{aligned}
(\forall x(P(x) \land Q(x,a)))^{\mathcal{I}} &\leftrightarrow (P(2) \land Q(2,a))^{\mathcal{I}} \land (P(3) \land Q(3,a))^{\mathcal{I}} \\
&\leftrightarrow (P(2)^{\mathcal{I}} \land Q(2,a^{\mathcal{I}})^{\mathcal{I}}) \land (P(3)^{\mathcal{I}} \land Q(3,a^{\mathcal{I}})^{\mathcal{I}}) \\
&\leftrightarrow (P(2)^{\mathcal{I}} \land Q(2,2)^{\mathcal{I}}) \land (P(3)^{\mathcal{I}} \land Q(3,2)^{\mathcal{I}}) \\
&\leftrightarrow (0 \land 1) \land (1 \land 1) \qquad (\text{因为} 2 \notin P^{\mathcal{I}}, \langle 2,2\rangle \in Q^{\mathcal{I}}, 3 \in P^{\mathcal{I}}, \langle 3,2\rangle \in Q^{\mathcal{I}}) \\
&\leftrightarrow 0 \land 1 \\
&\leftrightarrow 0
\end{aligned}
$$

因此，公式 $\forall x(P(x) \land Q(x,a))$ 在该解释下为假。

13 一阶逻辑中的可满足性 (Satisfiability in FOL)

13.1 公式的满足性 (Satisfiability of a formula)

定义 13.1 (满足 (Satisfaction)). 一个解释 $\mathcal{I}$ 和环境 E **满足** ***(satisfy)*** 公式 α，记作 $\mathcal{I} \models_E \alpha$，当且仅当 $\alpha^{(\mathcal{I},E)} = 1$ (即公式 α 在解释 $\mathcal{I}$ 和环境 E 下的真值为真)。如果不满足，则记作 $\mathcal{I} \not\models_E \alpha$ (即 $\alpha^{(\mathcal{I},E)} = 0$)。

如果对**所有**环境 E 都有 $\mathcal{I} \models_E \alpha$（这通常只对语句或闭公式有意义，因为语句的真值不依赖于环境 E），则称解释 $\mathcal{I}$ 满足 α，记作 $\mathcal{I} \models \alpha$。

下表总结了各种类型公式的满足条件（即 $\mathcal{I} \models_E \alpha$ 的条件）:

公式类型 α	使得 $\mathcal{I} \models_E \alpha$ 的条件
$P(t_1, \ldots, t_k)$	$\langle t_1^{(\mathcal{I},E)}, \ldots, t_k^{(\mathcal{I},E)}\rangle \in P^{\mathcal{I}}$
$t_1 = t_2$	$t_1^{(\mathcal{I},E)} = t_2^{(\mathcal{I},E)}$
$\neg\beta$	$\mathcal{I} \not\models_E \beta$
$\beta \land \gamma$	$\mathcal{I} \models_E \beta$ 并且 $\mathcal{I} \models_E \gamma$
$\beta \lor \gamma$	$\mathcal{I} \models_E \beta$ 或者 $\mathcal{I} \models_E \gamma$ (或两者都满足)
$\beta \to \gamma$	$\mathcal{I} \not\models_E \beta$ 或者 $\mathcal{I} \models_E \gamma$ (或两者都满足)
$\forall x\beta$	对论域 $D = dom(\mathcal{I})$ 中的**每个**元素 d，都有 $\mathcal{I} \models_{E[x\mapsto d]} \beta$
$\exists x\beta$	在论域 $D = dom(\mathcal{I})$ 中**存在某个**元素 d，使得 $\mathcal{I} \models_{E[x\mapsto d]} \beta$

其中 $E[x \mapsto d]$ 表示一个与环境 E 几乎相同，仅将变元 x 映射到论域元素 d 的新环境。

13.2 可满足性示例 (Satisfiability Examples)

(此处省略详细示例的 *LaTeX* 代码，因其通常涉及复杂的解释构造和多步的真值计算。可参考 *PDF* 中 *Example 1, 2, 3 (pages 12-15)* 的思路进行构造和分析。)

14 一阶逻辑中的语义后承 (Semantic Entailment in FOL)

14.1 公式集满足性的定义 (Satisfiability of a set of formulas)

定义 14.1 (公式集的满足). 令 Σ 为一个 *FOL* 公式的集合。对于解释 $\mathcal{I}$ 和环境 E，我们说 $\mathcal{I}$ 和 E **满足** Σ，记作 $\mathcal{I} \models_E \Sigma$，当且仅当对于 Σ 中的每一个公式 $\phi \in \Sigma$，都有 $\mathcal{I} \models_E \phi$。

14.2 语义后承的定义 (Definition of Semantic Entailment $\Sigma \models \alpha$)

定义 14.2 (语义后承 (Semantic Entailment)). 令 Σ 为一个 *FOL* 公式的集合（前提集），α 为一个 *FOL* 公式（结论）。我们说 Σ **语义后承** α，或称 α 是 Σ 的**逻辑后承** ***(Logical Consequence)***，记作 $\Sigma \models \alpha$，当且仅当：对于任何解释 $\mathcal{I}$ 和任何环境 E，如果 $\mathcal{I} \models_E \Sigma$ *(即 $\mathcal{I}$ 和 E 满足 Σ 中的所有公式)*，那么一定有 $\mathcal{I} \models_E \alpha$ *(即 $\mathcal{I}$ 和 E 也满足 α)*。

直观解释：任何使所有前提 Σ 都为真的模型（解释与环境的组合），也必须使结论 α 为真。

14.3 语义后承示例 (Semantic Entailment Examples)

示例 14.3 (证明 $\forall x(\neg\gamma) \models \neg(\exists x\gamma)$). 我们要证明：对于任意解释 $\mathcal{I}$ 和环境 E，如果 $\mathcal{I} \models_E \forall x(\neg\gamma)$，那么 $\mathcal{I} \models_E \neg(\exists x\gamma)$。

假设 $\mathcal{I} \models_E \forall x(\neg\gamma)$。根据 $\forall$ 的语义，这意味着对于论域 $D = dom(\mathcal{I})$ 中的每一个元素 $d \in D$，都有 $\mathcal{I} \models_{E[x\mapsto d]} (\neg\gamma)$。这等价于，对于每一个 $d \in D$，$\mathcal{I} \not\models_{E[x\mapsto d]} \gamma$。这意味着，在论域 D 中不存在任何元素 d 使得 $\mathcal{I} \models_{E[x\mapsto d]} \gamma$。根据 $\exists$ 的语义，如果不存在这样的 d，那么 $\mathcal{I} \not\models_E (\exists x\gamma)$。最后，根据 $\neg$ 的语义，如果 $\mathcal{I} \not\models_E (\exists x\gamma)$，那么 $\mathcal{I} \models_E \neg(\exists x\gamma)$。证明完毕。

示例 14.4 (证明 $\forall x(P(x) \to Q(x)) \not\models (\forall xP(x)) \to (\forall xQ(x))$ 是错误的,应为证明 $\forall x(P(x) \to Q(x)) \models (\forall xP(x)) \to (\forall xQ(x))$ 但 PDF 第 20 页似乎在构造一个反例，说明 $(\forall xP(x)) \to (\forall xQ(x)) \not\models \forall x(P(x) \to Q(x))$，或者说两者不等价。PDF 原文目标是证明 $\forall x(P(x) \to Q(x)) \not\models (\forall xP(x)) \wedge \neg(\forall xQ(x))$ 不是一个有效的证明方式，而是要找到一个解释使得前提为真结论为假。). 更正：我们来证明 $\forall x(P(x) \to Q(x)), \forall xP(x) \models \forall xQ(x)$ （这在自然演绎中已证）。或者，证明一个更简单的有效公式的后承，例如：$\forall xA(x) \models \exists xA(x)$ *(如果论域非空)*。假设 $\mathcal{I} \models_E \forall xA(x)$。论域 D 非空，取 $d \in D$。则 $\mathcal{I} \models_{E[x\mapsto d]} A(x)$。因此存在 $d \in D$ 使得 $\mathcal{I} \models_{E[x\mapsto d]} A(x)$。所以 $\mathcal{I} \models_E \exists xA(x)$。

14.4 有效性与可满足性再讨论 (Validity and Satisfiability Revisited)

一个 *FOL* 公式 α 是：

- **有效的** ***(Valid)*** **/ 逻辑真** ***(Logically True)***: 如果对于**每一个**解释 $\mathcal{I}$ 和**每一个**环境 E，都有 $\mathcal{I} \models_E \alpha$。这等价于 $\emptyset \models \alpha$ *(即 α 是空前提集的逻辑后承)*。有效的公式类似于命题逻辑中的重言式。
- **可满足的** ***(Satisfiable)***: 如果**存在某个**解释 $\mathcal{I}$ 和某个环境 E，使得 $\mathcal{I} \models_E \alpha$。
- **不可满足的** ***(Unsatisfiable)*** **/ 矛盾的** ***(Contradictory)***: 如果对于**每一个**解释 $\mathcal{I}$ 和**每一个**环境 E，都有 $\mathcal{I} \not\models_E \alpha$。即不存在任何解释和环境使其为真。这类似于命题逻辑中的矛盾式。

示例 14.5 (K-公理的有效性). 证明公式 $(\forall x(\alpha \to \beta)) \to ((\forall x\alpha) \to (\forall x\beta))$ 是有效的。采用反证法。假设该公式无效，则存在某个解释 $\mathcal{I}$ 和环境 E 使得：$\mathcal{I} \models_E \forall x(\alpha \to \beta)$ *(1)* $\mathcal{I} \models_E \forall x\alpha$ *(2)* $\mathcal{I} \not\models_E \forall x\beta$ *(3)*

由 *(3)* 可知，存在某个 $d_0 \in D = dom(\mathcal{I})$ 使得 $\mathcal{I} \not\models_{E[x\mapsto d_0]} \beta$。*(4)* 由 *(1)* 可知，对于所有的 $d \in D$，$\mathcal{I} \models_{E[x\mapsto d]} (\alpha \to \beta)$。特别地，对于 d_0，有 $\mathcal{I} \models_{E[x\mapsto d_0]} (\alpha \to \beta)$。*(5)* 由 *(2)* 可知，对于所有的 $d \in D$，$\mathcal{I} \models_{E[x\mapsto d]} \alpha$。特别地，对于 d_0，有 $\mathcal{I} \models_{E[x\mapsto d_0]} \alpha$。*(6)*

从 *(5)* 和 *(6)*，根据蕴含的定义，由于前提 α 在 $E[x \mapsto d_0]$ 下为真，则后件 β 在 $E[x \mapsto d_0]$ 下也必须为真，即 $\mathcal{I} \models_{E[x\mapsto d_0]} \beta$。但这与 *(4)* $\mathcal{I} \not\models_{E[x\mapsto d_0]} \beta$ 矛盾。因此，初始假设（该公式无效）错误。所以该公式是有效的。

示例 14.6 (其他有效性判断).

- $\exists y \forall x R(x,y) \to \forall x \exists y R(x,y)$：**有效**。*(如果存在一个 y_0 对所有 x 都有 $R(x,y_0)$，那么对每个 x 都存在一个 y (即 y_0) 使得 $R(x,y)$。)*
- $\forall x \exists y R(x,y) \to \exists y \forall x R(x,y)$：**无效**。*(反例：论域为整数，$R(x,y)$ 为 $y = x+1$。对每个整数 x，都存在一个 y (即 $x+1$) 满足关系，但不存在一个 y 对所有整数 x 都满足 $y = x+1$。)*
- $\exists x (P(x) \to \forall x P(x))$：**有效**。*(考虑两种情况：(1) $\forall x P(x)$ 为真。那么 $P(x) \to \forall x P(x)$ 的后件为真，所以蕴含式为真，整个公式为真。(2) $\forall x P(x)$ 为假。这意味着存在某个 a 使得 $P(a)$ 为假。那么对于这个 a，$P(a) \to \forall x P(x)$ 的前件为假，所以蕴含式为真。因此，总能找到一个 x 使 $P(x) \to \forall x P(x)$ 为真。)*

15 一阶逻辑的判定性 (Decidability of FOL)

15.1 一阶逻辑的不可判定性 (Undecidability of First-Order Logic)

对于一个给定的 *FOL* 公式 α，判断其是否有效（即 $\emptyset \models \alpha$ 是否成立）的问题，称为 *FOL* 的**有效性问题** ***(Validity Problem)***。

定理 15.1 (邱奇-图灵定理 Church-Turing Theorem). 一阶逻辑的有效性问题是**不可判定的** ***(Undecidable)***。

这意味着不存在一个通用算法（如图灵机或等价的计算机程序），该算法对于输入的任意一个 *FOL* 公式，都能在有限时间内停机并正确回答该公式是否有效。与此相对，命题逻辑是**可判定的** ***(Decidable)***，因为真值表方法提供了一个判定任意命题逻辑公式是否为重言式（即有效）的算法。

尽管一般情况下 *FOL* 是不可判定的，但对于 *FOL* 的某些特定片段或满足特定条件的公式集，有效性问题可能是可判定的。例如，只包含一元谓词符号且不含函词符号的一阶逻辑（称为一元一阶逻辑，*Monadic First-Order Logic*）是可判定的。

16 一阶逻辑的自然演绎系统 (Natural Deduction for FOL)

FOL 的自然演绎系统扩展了命题逻辑的 *ND* 系统，增加了处理量词的规则。其他证明技巧（如假设、子证明）和命题逻辑的规则（如 $\wedge i, \wedge e, \to i, \to e$ 等）保持不变。

16.1 代换 (Substitution)

在谓词逻辑证明中，常常需要将公式中的变元替换为项。

定义 16.1 (代换 (Substitution)). 对于变元 x，项 t，和公式 α，我们将把 α 中 x 的**每个自由出现** ***(free occurrence)*** 都替换为项 t 后得到的公式记为 $\alpha[t/x]$。**重要限制**：代换**不应影响**约束出现 *(bound occurrences)* 的变元。直观地，$\alpha[t/x]$ 回答了“如果 x 具有由项 t 指定的值，α 会发生什么?”

示例 16.2 (代换示例).

- 令 $\alpha \equiv E(f(x))$。那么 $\alpha[(y+y)/x]$ 就是 $E(f((y+y)))$。
- 令 $\beta \equiv P(x) \land (\exists x Q(x))$。那么 $\beta[y/x]$ 就是 $P(y) \land (\exists x Q(x))$。注意：只有 $P(x)$ 中的 x *(自由出现)* 被替换为 y；而 $\exists x Q(x)$ 中的 x 是被存在量词 $\exists x$ 绑定的，因此不受此替换影响。
- 令 $\gamma \equiv \forall x(E(f(x)) \land S(x,y))$。那么 $\gamma[g(x,y)/x]$ 仍然是 γ 本身，因为 γ 中没有 x 的自由出现。*(要替换 x，必须是自由的 x。)*

16.1.1 避免捕获 (Avoid Capture)

当用一个包含自由变元 v 的项 t 去替换公式 α 中的自由变元 x 时，如果 x 在 α 中出现的某些位置位于一个约束 v 的量词 *($\forall v$ 或 $\exists v$)* 的作用域内，那么 t 中的 v 在被代入后就会从自由变元“意外地”变成了约束变元。这称为“捕获”，它会改变原公式的逻辑含义，是必须避免的。

解决方法: 在进行代换之前，如果预见到可能发生捕获，就需要对 α 中可能引起捕获的量词进行**变元重命名** ***(renaming)***。即将该量词约束的变元（比如上例中的 y）换成一个全新的、在 α 和 t 中都未作为自由变元出现的变元（比如 w）。重命名被约束的变元不会改变原公式的语义。

示例 16.3 (避免捕获示例).

- 设 $\alpha \equiv \forall z(\exists y((z+y)=x))$，我们要计算 $\alpha[(y-1)/x]$。
- 若直接代换，会得到 $\forall z(\exists y((z+y)=(y-1)))$，其中 $(y-1)$ 中的 y 被 $\exists y$ 捕获。
- 正确做法：先将 α 中的 $\exists y$ 重命名为 $\exists w$ *(假设 w 是新变元)*，得到 $\alpha' \equiv \forall z(\exists w((z+w)=x))$。
- 然后再进行代换 $\alpha'[(y-1)/x]$，得到 $\forall z(\exists w((z+w)=(y-1)))$。此时 $(y-1)$ 中的 y 仍然是自由的。

16.2 等词规则 (Equality Rules)

(此规则通常作为 FOL 自然演绎的一部分)

规则 16.1 (等词引入 $(=i)$). 对任意项 t，$\vdash t=t$。记作：$\dfrac{}{t=t}\quad (=i)$

规则 16.2 (等词排除 $(=e)$). 如果 $\Sigma \vdash t_1=t_2$ 且 $\Sigma \vdash \alpha[t_1/x]$，则 $\Sigma \vdash \alpha[t_2/x]$。记作：$\dfrac{t_1=t_2 \quad \alpha[t_1/x]}{\alpha[t_2/x]}\quad (=e)$

16.3 量词规则 (Quantifier Rules)

16.3.1 全称量词 ($\forall$) 规则

规则 16.3 ($\forall$-排除 $(\forall e)$ (Universal Elimination)). 如果 $\Sigma \vdash_{ND} (\forall x\alpha)$，那么对任意项 t，$\Sigma \vdash_{ND} \alpha[t/x]$。

$$\frac{\forall x\alpha}{\alpha[t/x]}\quad (\forall e)$$

直观: 如果一个性质 α 对所有 x 都为真，那么它对由某个具体项 t 所代表的特定值也为真。

规则 16.4 ($\forall$-引入 $(\forall i)$ (Universal Introduction)). 如果 $\Sigma \vdash_{ND} \alpha[y/x]$，并且 y 是一个**新颖** ***(fresh)*** 的变元，那么 $\Sigma \vdash_{ND} (\forall x\alpha)$。

$$\boxed{\begin{array}{l} y \quad \textit{fresh (新颖变元)} \\ \quad \vdots \\ \alpha[y/x] \end{array}} \Longrightarrow \forall x\alpha \quad (\forall i)$$

直观: 为了证明 $\forall x\alpha$，我们选择一个任意的个体（用新颖变元 y 表示），并证明 $\alpha[y/x]$ 成立。因为 y 是任意选取的，所以这个证明对所有个体都有效。新颖变元 y 只应在引入 $\forall$ 的子证明内部使用。

16.3.2 存在量词 ($\exists$) 规则

规则 16.5 ($\exists$-引入 ($\exists i$) (Existential Introduction)). 如果 $\Sigma \vdash_{ND} \alpha[t/x]$ 对某个项 t 成立，那么 $\Sigma \vdash_{ND} (\exists x\alpha)$。

$$\frac{\alpha[t/x]}{\exists x\alpha} \quad (\exists i)$$

直观: 如果我们已经证明了性质 α 对于某个特定的项 t 成立，那么我们就可以推断出存在至少一个 x 使得 α 为真。

规则 16.6 ($\exists$-排除 ($\exists e$) (Existential Elimination)). 如果 *(1)* $\Sigma \vdash_{ND} (\exists x\alpha)$，并且 *(2)* 通过从 Σ 增加一个临时假设 $\alpha[u/x]$ *(*其中 u 是一个新颖变元*)* 可以推导出结论 β *(*即 $\Sigma, \alpha[u/x] \vdash_{ND} \beta$*)*，并且新颖变元 u 不在 Σ, α *(*除去 x*)*, 或 β 中自由出现，那么从 Σ 可以推导出 β。

$$\frac{\exists x\alpha \qquad \boxed{\begin{array}{ll} u & \textit{fresh, } \alpha[u/x] \quad \textit{(假设)} \\ & \vdots \\ \beta & \end{array}}}{\beta \quad (\exists e) \textit{ (条件: } u \textit{ 不在} \beta, \Sigma, \alpha \textit{ 中自由出现)}}$$

直观: 我们知道“存在某个 x 满足 α”。我们给它一个临时的名字 u（新颖变元），并假设 $\alpha[u/x]$ 成立。如果基于此能推导出结论 β，且 β 本身不依赖于 u 的具体身份，则 β 成立。

16.4 示例证明

以下证明使用编号列表，子证明通过嵌套的编号列表和缩进表示。

16.4.1 证明: $\forall xP(x) \vdash_{\mathbf{ND}} \exists xP(x)$

1. $\forall xP(x)$ *(前提)*
2. $P(u)$ *(∀e: 1, u 是一个项)*
3. $\exists xP(x)$ *(∃i: 2)*

16.4.2 证明: $\{P(t), \forall x(P(x) \to \neg Q(x))\} \vdash_{\mathbf{ND}} \neg Q(t)$

1. $P(t)$ *(前提)*
2. $\forall x(P(x) \to \neg Q(x))$ *(前提)*
3. $P(t) \to \neg Q(t)$ *(∀e: 2, 用 t 代换 x)*
4. $\neg Q(t)$ *(→ e: 1, 3)*

16.4.3 证明: $\forall x P(x) \vdash_{\mathbf{ND}} \forall y P(y)$

1. $\forall x P(x)$ *(前提)*

2. *引入 u fresh 开始子证明 (为 $\forall i$):*

   2..1. $P(u)$ *($\forall e$: 1, 用 u 代换 x)*

3. $\forall y P(y)$ *($\forall i$: 2, 将 $P(u)$ 中的 u 泛化为 y)*

16.4.4 证明: $\emptyset \vdash_{\mathbf{ND}} \forall x(P(x) \to P(x))$

1. *引入 u fresh 开始子证明 (为 $\forall i$):*

   1..1. *假设 $P(u)$ 开始子证明 (为 $\to i$):*

      1..1..1. $P(u)$ *(重复假设 1.1)*

   1..2. $P(u) \to P(u)$ *($\to i$: 1.1)*

2. $\forall x(P(x) \to P(x))$ *($\forall i$: 1)*

16.4.5 证明: $\{\forall x(P(x) \to Q(x)), \forall x P(x)\} \vdash_{\mathbf{ND}} \forall x Q(x)$

1. $\forall x(P(x) \to Q(x))$ *(前提)*

2. $\forall x P(x)$ *(前提)*

3. *引入 u fresh 开始子证明 (为 $\forall i$):*

   3..1. $P(u) \to Q(u)$ *($\forall e$: 1)*

   3..2. $P(u)$ *($\forall e$: 2)*

   3..3. $Q(u)$ *($\to e$: 3.1, 3.2)*

4. $\forall x Q(x)$ *($\forall i$: 3)*

16.4.6 证明: $\exists x P(x) \vdash_{\mathbf{ND}} \exists y P(y)$

1. $\exists x P(x)$ *(前提)*

2. *假设 $P(u)$ (u fresh) 开始子证明 (为 $\exists e$):*

   2..1. $\exists y P(y)$ *($\exists i$: 2, u 是 y 的一个实例)*

3. $\exists y P(y)$ *($\exists e$: 1, 2)*

16.4.7 证明: $\exists y(\forall x P(x, y)) \vdash_{\mathbf{ND}} \forall x(\exists y P(x, y))$

1. $\exists y(\forall x P(x, y))$ *(前提)*

2. *假设 $\forall x P(x, w)$ (w fresh) 开始子证明 (为 $\exists e$):*

   2..1. *引入 u fresh 开始子证明 (为 $\forall i$):*

      2..1..1. $P(u, w)$ *($\forall e$: 2, 即 $\forall x P(x, w)$)*

      2..1..2. $\exists y P(u, y)$ *($\exists i$: 2.1.1, w 是 y 的一个实例)*

   2..2. $\forall x(\exists y P(x, y))$ *($\forall i$: 2.1)*

3. $\forall x(\exists y P(x,y))$ *(∃e: 1, 2)*

16.4.8 证明: $\{\exists x P(x), \forall x(P(x) \to Q(x))\} \vdash_{\mathbf{ND}} \exists x Q(x)$

1. $\exists x P(x)$ *(前提)*

2. $\forall x(P(x) \to Q(x))$ *(前提)*

3. 假设 $P(u)$ *(u fresh) 开始子证明 (为* $\exists e$*):*

3..1. $P(u) \to Q(u)$ *(∀e: 2)*

3..2. $Q(u)$ *(→ e: 3 (即* $P(u)$*), 3.1)*

3..3. $\exists x Q(x)$ *(∃i: 3.2)*

4. $\exists x Q(x)$ *(∃e: 1, 3)*

16.4.9 证明: $\exists x(\neg P(x)) \vdash_{\mathbf{ND}} \neg(\forall x P(x))$

1. $\exists x(\neg P(x))$ *(前提)*

2. 假设 $\neg P(u)$ *(u fresh) 开始子证明 (为* $\exists e$*):*

2..1. 假设 $\forall x P(x)$ *开始子证明 (为* $\neg i$*):*

2..1..1. $P(u)$ *(∀e: 2.1, 即* $\forall x P(x)$*)*

2..1..2. $\bot$ *(¬e: 2 (即* $\neg P(u)$*), 2.1.1)*

2..2. $\neg(\forall x P(x))$ *(¬i: 2.1)*

3. $\neg(\forall x P(x))$ *(∃e: 1, 2)*

16.4.10 证明: $\neg(\forall x P(x)) \vdash_{\mathbf{ND}} \exists x(\neg P(x))$ **(经典逻辑)**

1. $\neg(\forall x P(x))$ *(前提)*

2. 假设 $\neg(\exists x \neg P(x))$ *开始子证明 (为 RAA/PBC 或* $\neg i$*):*

2..1. 引入 u fresh 开始子证明 (为 $\forall i$*):*

2..1..1. 假设 $\neg P(u)$ *开始子证明 (为 RAA/PBC 或* $\neg i$ *以得到* $P(u)$*):*

2..1..1..1. $\exists x \neg P(x)$ *(∃i: 2.1.1, 即* $\neg P(u)$*)*

2..1..1..2. $\bot$ *(¬e: 2 (即* $\neg(\exists x \neg P(x))$*), 2.1.1.1)*

2..1..2. $P(u)$ *(RAA/PBC 或* $\neg i$*: 2.1.1;* $\neg\neg e$ *隐含使用)*

2..2. $\forall x P(x)$ *(∀i: 2.1)*

2..3. $\bot$ *(¬e: 1, 2.2)*

3. $\exists x(\neg P(x))$ *(RAA/PBC 或* $\neg i$*: 2; 经典规则)*

备注: 最后一个证明 $\neg(\forall x P(x)) \vdash_{ND} \exists x(\neg P(x))$ 通常需要经典逻辑规则。在直觉主义逻辑中此方向不成立。

17 程序验证概述

17.1 动机

传统的代码审查和测试方法难以保证程序的完全正确性，尤其对于安全关键、商业关键或任务关键的系统。历史上有很多因软件错误导致的严重事故和损失。形式化验证提供了一种数学上严格的方法来证明程序的正确性。

17.2 验证过程

1. 将程序的非形式化需求 R 转换为逻辑公式 ϕ_R（形式规约）。
2. 编写旨在满足需求 R 的程序 P。
3. 证明程序 P 满足形式规约 ϕ_R。

本部分主要关注第 3 步。

17.3 命令式程序与状态 (Imperative Programs & State)

我们将使用一种包含整数/布尔表达式、赋值、条件语句和 *while* 循环的核心命令式语言。执行命令式程序会改变**程序状态** *(State)*，即程序变量的值。程序执行可以看作是从一个**初始状态** *(Initial state)* 转换到一个**最终状态** *(Final state)* 的过程。示例：计算阶乘的程序在执行过程中会经历一系列状态变化。

17.4 形式规约与霍尔三元组 (Formal Specification & Hoare Triples)

程序需求通常用自然语言描述，可能不一致或模糊。**形式规约** *(Formal Specification)* 使用数学符号精确描述程序应满足的属性，如同合同。关键组成部分：

- **前条件** ***(Precondition)***：程序执行**前**状态必须满足的条件。
- **后条件** ***(Postcondition)***：程序执行**后**状态必须满足的条件。

霍尔三元组 ***(Hoare Triple)*** *(由 C.A.R. Hoare 提出)*：$\{P\}\,C\,\{Q\}$

- P 是前条件 *(FOL 公式)*。
- C 是程序代码。
- Q 是后条件 *(FOL 公式)*。

$\{P\}\,C\,\{Q\}$ 的含义是：如果程序 C 的执行在满足前条件 P 的状态下开始，**并且如果执行能够终止**，那么执行结束时的状态将满足后条件 Q。**注意**：写出正确的规约本身可能很棘手。错误的规约可能导致证明了一个“正确”但实际行为错误的程序。

18 部分正确性与完全正确性 (Partial and Total Correctness)

18.1 部分正确性 (Partial Correctness)

定义 18.1 (部分正确性). 霍尔三元组 $\{P\}\,C\,\{Q\}$ 在**部分正确性**意义下成立，记作 $\models_{par} \{P\}\,C\,\{Q\}$，当且仅当：对每个满足 P 的初始状态 s_1，如果 C 从 s_1 开始执行**能够终止**于状态 s_2，那么状态 s_2 满足 Q。

部分正确性不要求程序必须终止。如果程序从不终止（例如无限循环），那么任何关于它的部分正确性规约都是（空洞地）成立的。它只保证如果程序停下来，结果是对的。

18.2 完全正确性 (Total Correctness)

定义 18.2 (完全正确性). 霍尔三元组 $\{P\}\,C\,\{Q\}$ 在**完全正确性**意义下成立，记作 $\models_{tot} \{P\}\,C\,\{Q\}$，当且仅当：对每个满足 P 的初始状态 s_1：

- C 从 s_1 开始执行**必须终止**于某个状态 s_2，
- 并且状态 s_2 满足 Q。

直观地，完全正确性 = 部分正确性 + 终止性。

示例辨析：

- $\models_{par} \{x = 1\}\, y = x\, \{y = 1\}$ *(且 $\models_{tot}$)*
- $\not\models_{par} \{x = 1\}\, y = x\, \{y = 2\}$
- $\models_{par} \{x = 1\}\, \mathit{while\ True\ do}\ x = 0\, \{x > 0\}$ *(因为不终止)*
- $\models_{par} \{x \geq 0\}\, \mathit{Factorial\ Program}\, \{y = x!\}$ *(假设输入 x 不变)*
- $\not\models_{par} \{\mathit{True}\}\, \mathit{Factorial\ Program}\, \{y = x!\}$ *(如果 $x<0$，不满足前条件，但前条件是 True)*

19 程序变量与逻辑变量 (Program Variables and Logical Variables)

在规约中，除了程序代码中出现的**程序变量** ***(Program variables)***，有时还需要引入不出现在程序中的**逻辑变量** ***(Logical variables)*** *(也叫辅助变量 Auxiliary variables 或幽灵变量 Ghost variables)*。这对于描述程序执行前后变量值的关系特别有用，尤其是当程序变量在执行过程中被修改时。示例：对于一个计算阶乘并将结果存入 y，同时修改了输入 x 的程序 C: $y = 1;\ \mathit{while}\ (x! = 0)\{y = y * x; x = x - 1;\}$ 规约 $\{x \geq 0\}\, C\, \{y = x!\}$ 是不正确的，因为程序结束时 x 的值变成了 *0*。正确的规约应该使用逻辑变量 x_0 来保存 x 的初始值：$\{x = x_0 \wedge x \geq 0\}\, C\, \{y = x_0!\}$ 逻辑变量 x_0 不会被程序 C 修改，因为它不出现在 C 中。程序状态只为程序变量赋值，不为逻辑变量赋值。

20 霍尔逻辑：公理与推理规则 (Hoare Logic: Axioms and Inference Rules)

霍尔逻辑提供了一个形式化的证明系统，用于推导形如 $\vdash_{par} \{P\}\, C\, \{Q\}$ 的部分正确性断言。证明过程通常是在代码语句之间插入中间断言（*FOL* 公式），并为每个霍尔三元组提供理由。

20.1 赋值公理 (Assignment Axiom)

公理 20.1 (赋值公理). $\vdash_{par} \{Q[E/x]\}\, x = E\, \{Q\}$

其中 $Q[E/x]$ 表示将后条件 Q 中所有变量 x 的自由出现替换为表达式 E 得到的新公式。直观含义：要确保赋值语句 $x = E$ 执行后状态满足 Q，必须要求执行前的状态满足将 Q 中的 x 替换为 E 后的条件。在实际证明中，通常从后条件 Q 出发，反向计算（或“上推”）得到赋值语句之前所需满足的最弱前条件 $Q[E/x]$。

20.2 隐含规则 (Rule of Consequence / Implied Rule)

该规则允许我们在证明中加强前条件或减弱后条件。

规则 20.1 (前条件加强 (Precondition Strengthening)). $$\frac{P \to P' \quad \vdash_{par} \{P'\}\, C\, \{Q\}}{\vdash_{par} \{P\}\, C\, \{Q\}}$$

规则 20.2 (后条件减弱 (Postcondition Weakening)). $$\frac{\vdash_{par} \{P\}\, C\, \{Q'\} \quad Q' \to Q}{\vdash_{par} \{P\}\, C\, \{Q\}}$$

这个规则连接了程序逻辑的证明和 *FOL*（加上算术公理）的证明。要应用此规则，需要证明 $P \to P'$ 或 $Q' \to Q$ 在 *FOL*（及算术理论）中是有效的。

20.3 复合规则 (Rule of Composition / Sequencing Rule)

用于证明顺序执行的程序的正确性。

规则 20.3 (复合规则). $$\frac{\vdash_{par} \{P\}\, C_1\, \{R\} \quad \vdash_{par} \{R\}\, C_2\, \{Q\}}{\vdash_{par} \{P\}\, C_1; C_2\, \{Q\}}$$

要证明 $\{P\}\, C_1; C_2\, \{Q\}$，需要找到一个合适的**中间条件** ***(Midcondition)*** R，并分别证明 $\{P\}\, C_1\, \{R\}$ 和 $\{R\}\, C_2\, \{Q\}$。

20.4 条件规则 (Conditional Rule / If Rule)

用于证明 *if-then-else* 语句的正确性。

规则 20.4 (条件规则). $$\frac{\vdash_{par} \{P \wedge B\}\, C_1\, \{Q\} \quad \vdash_{par} \{P \wedge \neg B\}\, C_2\, \{Q\}}{\vdash_{par} \{P\}\, \mathit{if}\ B\{C_1\}\ \mathit{else}\ \{C_2\}\, \{Q\}}$$

证明分解为两种情况：条件 B 为真（执行 C_1）和条件 B 为假（执行 C_2）。对于简单的 *if-then* 语句 *(if B { C })*，可以看作 *if B { C } else { skip }*。

20.5 循环规则 (While Rule / Iteration Rule)

用于证明 *while* 循环的部分正确性。

规则 20.5 (部分循环规则 (Partial While Rule)). $$\frac{\vdash_{par} \{I \wedge B\}\, C\, \{I\}}{\vdash_{par} \{I\}\, \mathit{while}\ B\{C\}\, \{I \wedge \neg B\}}$$

这里的 I 是一个 *FOL* 公式，称为**循环不变式 *(Loop Invariant)***。**循环不变式** I 是一个在循环执行过程中保持其真值的性质：

- 在循环开始前必须为真。
- 如果在某次循环迭代开始时为真，并且循环条件 B 也为真，那么执行一次循环体 C 后，I 必须仍然为真。*(*这是规则的前提 $\vdash_{par} \{I \wedge B\}\, C\, \{I\}$ 所要求的*)*
- 因此，无论循环执行多少次（包括零次），只要 I 初始为真，那么在循环的每次迭代开始时（如果进入循环）和结束时（如果循环终止）都为真。

循环规则的结论 $\{I\}\, while\ B\{C\}\, \{I \wedge \neg B\}$ 表明：如果循环以满足不变式 I 的状态开始，并且循环最终终止，那么终止时的状态将满足 I 并且循环条件 B 为假。

证明 *while* 循环的部分正确性步骤：

1. **找到循环不变式** I：这是最关键也最具创造性的一步，通常需要对循环行为的理解。可以通过观察循环执行过程中的变量关系来寻找（例如，检查每次循环测试时的变量值）。
2. **证明不变式初始成立**：证明在循环第一次执行前，I 是成立的（通常需要结合循环前的代码和隐含规则）。
3. **证明不变式保持**：证明循环规则的前提 $\vdash_{par} \{I \wedge B\}\, C\, \{I\}$ 成立。即假设在某次迭代开始时 I 和 B 都为真，证明执行一次循环体 C 后 I 仍然为真。
4. **证明最终结果**：证明当循环终止时（即 $I \wedge \neg B$ 成立），可以推导出期望的最终后条件（通常需要使用隐含规则）。

选择一个“足够强”的不变式很重要，它不仅要在循环中保持，还要在循环结束后能帮助推导出最终的后条件。

21 证明表 (Proof Tableaux)

证明表是一种系统地组织霍尔逻辑证明的方法，特别是对于顺序执行的代码。

定义 21.1 (最弱前条件 (Weakest Precondition))**.** 给定程序（片段）C 和后条件 ψ，使得 $\{\phi\}\, C\, \{\psi\}$ 成立的**最弱前条件** *(wp)* ϕ 是逻辑上最弱的公式，其在 C 执行前的真值足以保证 C 执行后 ψ 为真（假设 C 终止）。

构造证明表的一般过程：对于程序 $C_1; C_2; \ldots; C_n$ 和给定的前条件 ϕ 和后条件 ψ：

1. 从最终的后条件 ψ 开始。
2. 反向“上推 *(push upwards)*”后条件，依次通过 $C_n, C_{n-1}, \ldots, C_1$，计算每个语句执行前的（最弱）前条件。
   - 对于赋值 $x = E$，如果之后需要满足 Q，之前需要满足 $Q[E/x]$。
   - 对于复合 $C_1; C_2$，如果之后需要满足 Q，那么在 C_2 之前需要满足 $R = wp(C_2, Q)$，在 C_1 之前需要满足 $P = wp(C_1, R)$。
   - 对于条件 *if* $B\{C_1\}$ *else* $\{C_2\}$，如果之后需要满足 Q，设 $P_1 = wp(C_1, Q)$ 和 $P_2 = wp(C_2, Q)$，则在 *if* 语句之前需要满足 $(B \to P_1) \wedge (\neg B \to P_2)$。

- 对于循环 *while* $B\{C\}$，需要找到一个循环不变式 I。证明 $\{I \wedge B\}\, C\, \{I\}$，然后使用 $\{I\}$ *while* $B\{C\}$ $\{I \wedge \neg B\}$。

3. 当计算到程序最开始时，得到一个前条件 ϕ'。

4. 最后，使用隐含规则证明给定的初始前条件 ϕ 能够蕴含 ϕ' (即 $\phi \to \phi'$)。

虽然证明表通常是自底向上（从后条件到前条件）构建的，但其理由（*justification*）是自顶向下阅读时成立的。