



经典逻辑推理

Outline

❖ 自然演绎推理

❖ **SKOLEM**标准形及子句集

❖ 海伯伦定理

❖ 鲁宾逊归结原理

❖ 归结反演

❖ 应用归结反演求解问题

❖ 归结策略



归结
演绎
推理

自然演绎推理

❖ 自然演绎推理：从一组已知为真的事实出发，运用**经典逻辑的推理规则**推出结论的过程。

❖ 推理规则： P 规则、 T 规则、假言推理、拒取式推理

■ 假言推理： $P, P \rightarrow Q \Rightarrow Q$

“如果 x 是金属，则 x 能导电”，“铜是金属”推出“铜能导电”

■ 拒取式推理： $P \rightarrow Q, \neg Q \Rightarrow \neg P$

“如果下雨，则地湿”，“地不湿”推出“没有下雨”

❁ 例 已知事实:

(1) 凡是容易的课程小王(**Wang**)都喜欢;

(2) C 班的课程都是容易的;

(3) **ds** 是 C 班的一门课程。

■ 求证: 小王喜欢 **ds** 这门课程。

证明:

■ 定义谓词:

EASY(*x*): *x* 是容易的

LIKE(*x*, *y*): *x* 喜欢 *y*

C(*x*): *x* 是 C 班的一门课程

- 已知事实和结论用谓词公式表示:


$(\forall x) (EASY(x) \rightarrow LIKE(Wang, x))$


$(\forall x) (C(x) \rightarrow EASY(x))$

$C(ds)$

$LIKE(Wang, ds)$

- 应用推理规则进行推理:

 $(\forall x) (EASY(x) \rightarrow LIKE(Wang, x))$
 $EASY(z) \rightarrow LIKE(Wang, z)$ 全称固化

 $(\forall x) (C(x) \rightarrow EASY(x))$
 $C(y) \rightarrow EASY(y)$ 全称固化

所以 $C(ds), C(y) \rightarrow EASY(y)$

$\Rightarrow EASY(ds)$ P 规则及假言推理

所以 $EASY(ds), EASY(z) \rightarrow LIKE(Wang, z)$

$\Rightarrow LIKE(Wang, ds)$ T 规则及假言推理

自然演绎推理的特点

优点:

- 表达定理证明过程自然，易理解。
- 拥有丰富的推理规则，推理过程灵活。
- 便于嵌入领域启发式知识。

➤ 缺点:

- 易产生组合爆炸，得到的中间结论一般呈指数形式递增。

Outline

- ❖ 自然演绎推理

- ❖ **SKOLEM**标准形及子句集

- ❖ 海伯伦定理

- ❖ 鲁宾逊归结原理

- ❖ 归结反演

- ❖ 应用归结反演求解问题

- ❖ 归结策略

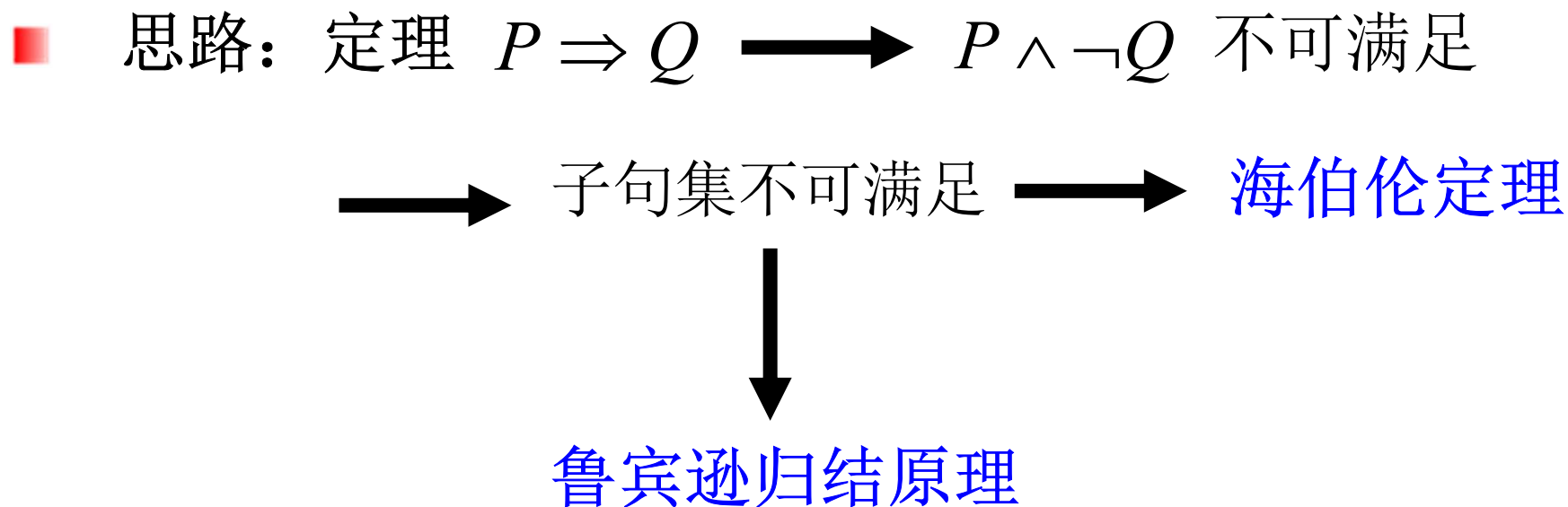


归结
演绎
推理

归结演绎推理

- 反证法: $P \Rightarrow Q$, 当且仅当 $P \wedge \neg Q \Leftrightarrow F$,
即 Q 为 P 的逻辑结论, 当且仅当 $P \wedge \neg Q$ 是不可满足的。
- 定理: Q 为 P_1, P_2, \dots, P_n 的逻辑结论, 当且仅当
 $(P_1 \wedge P_2 \wedge \dots \wedge P_n) \wedge \neg Q$ 是不可满足的。

归结演绎推理



SKOLEM标准形:

只有 \wedge , \vee , 谓词（原子），前有“非”符号（ \sim ）的谓词（负原子），以及看不见的全称量词（ \forall ）组成的合适公式，也称“与或句”。

SKOLEM标准形求取

❖ 任何合适公式都可化成与或句的形式

◆ 化成前束范式

所有量词都在合适公式的最前面，每个量词的辖域（适用范围）都是整个公式。

◆ 将合适公式化成等值的合取范式

◆ 消去存在量词

SKOLEM标准形求取

❖ SKOLEM标准形求取步骤:

1) “If A then B else C”化成

$$(A \rightarrow B) \wedge (\sim A \rightarrow C)$$

或 $(A \supset B) \wedge (\sim A \supset C)$

2) $A \equiv B$ (或 $A \Leftrightarrow B$) 化成

$$(A \rightarrow B) \wedge (B \rightarrow A)$$

或 $(A \supset B) \wedge (B \supset A)$

3) $A \rightarrow B$ (或 $A \supset B$) 化成

$$\sim A \vee B$$

4) 消去或移入“非”符号

- ✓ $\sim \sim A$ 化成 A
- ✓ $\sim(A \vee B)$ 化成 $\sim A \wedge \sim B$
- ✓ $\sim(A \wedge B)$ 化成 $\sim A \vee \sim B$
- ✓ $\sim \forall x A(x)$ 化成 $\exists x (\sim A(x))$
- ✓ $\sim \exists x A(x)$ 化成 $\forall x (\sim A(x))$

5) 所有的量词变量全部换成不同的名字

$\exists x A(x) \vee \exists x B(x)$ 化为 $\exists x A(x) \vee \exists y B(y)$

6) 所有的量词按原来次序移至最前边。

7) 消去存在量词:

- ◆ 存在量词未出现在全称量词的辖域内，用常量代替。
- ◆ 存在量词位于 n 个全称量词 x_1, x_2, \dots, x_n 的辖域内，需用skolem函数 $f(x_1, x_2, \dots, x_n)$ 替换受该存在量词约束的变元。

合适公式转**SKOLEM**标准形例

例1. 试将 $G=(\forall x)(\exists y)(\exists z) ((\sim P(x,y) \wedge Q(x,z)) \vee R(x,y,z))$ 化成**SKOLEM**标准形（即“与或式”）。

解： 1) 令 $M(x, y, z) = (\sim P(x,y) \wedge Q(x,z)) \vee R(x,y,z)$

则 $G = (\forall x)(\exists y)(\exists z) M(x, y, z)$

2) $M(x, y, z)$ 化成合取范式：

$$M(x, y, z) = (\sim P(x,y) \vee R(x,y,z)) \wedge (Q(x,z) \vee R(x,y,z))$$

$$\text{则 } G = (\forall x)(\exists y)(\exists z) ((\sim P(x,y) \vee R(x,y,z)) \wedge (Q(x,z) \vee R(x,y,z)))$$

3) 消去 $(\exists y)$, 令 $y = f(x)$, 有

$$G = (\forall x)(\exists z) ((\sim P(x, f(x)) \vee R(x, f(x), z)) \wedge (Q(x, z) \vee R(x, f(x), z)))$$

4) 消去 $(\exists z)$, 令 $z = g(x)$, 有

$$G = (\forall x) ((\sim P(x, f(x)) \vee R(x, f(x), g(x))) \wedge (Q(x, g(x)) \vee R(x, f(x), g(x))))$$

上式即为G的SKOLEM标准形。

例2. 试将 $G=(\exists x)(\forall y)(\forall z)(\exists u) P(x, y, z, u)$ 化成 SKOLEM 标准形。

解： 令 $x = a$ （某个常量）

则 $G = (\forall y)(\forall z)(\exists u) P(a, y, z, u)$

再令 $u = f(y, z)$ ，得

G 的 SKOLEM 标准形为：

$$G = (\forall y)(\forall z) P(a, y, z, f(y, z))$$

二、子句集

- ❖ 原子 (**atom**) 谓词公式： 一个不能再分解的命题
- ❖ 文字 (**literal**)： 原子谓词公式及其否定
 - ◆ P ： 正文字， $\sim P$ ： 负文字
- ❖ 子句 (**clause**)： 任何文字的析取式。任何文字本身也都是子句。
 $P(x) \vee Q(x), \neg P(x, f(x)) \vee Q(x, g(x))$
- ❖ 空子句 (**NIL**)： 不包含任何文字的子句
- ❖ 子句集： 由子句构成的集合

子句集的求取（共9步）

❖ 子句集S的求取：

将G的已消去存在量词的SKOLEM标准形，再略去全称量词，最后以“,”代替合取符号“ \wedge ”

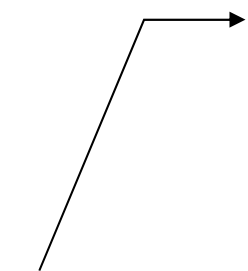
例：将下列谓词演算公式化为一个子句集

$$(\forall x) \{ P(x) \Rightarrow \{ (\forall y) [P(y) \Rightarrow P(f(x,y))] \wedge \sim (\forall y) [Q(x,y) \Rightarrow P(y)] \} \}$$

解：（1）消去蕴涵符号

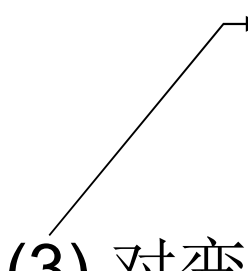
只应用 \vee 和 \sim 符号，以 $\sim P \vee Q$ 替换 $P \rightarrow Q$ 。

$$(1) (\forall x) \{ \sim P(x) \vee \{ (\forall y) [\sim P(y) \vee P(f(x,y))] \wedge \sim (\forall y) [\sim Q(x,y) \vee P(y)] \} \}$$


$$(2) (\forall x) \{ \sim P(x) \vee \{ (\forall y) [\sim P(y) \vee P(f(x,y))] \wedge (\exists y) [Q(x,y) \wedge \sim P(y)] \} \}$$

(2) 减少否定符号的辖域

每个否定符号 \sim 最多只用到一个谓词符号上，
并反复应用狄·摩根定律。


$$(3) (\forall x) \{ \sim P(x) \vee \{ (\forall y) [\sim P(y) \vee P(f(x,y))] \wedge (\exists w) [Q(x,w) \wedge \sim P(w)] \} \}$$

(3) 对变量标准化

重新命名变元名，使不同量词约束的变元有不同的名字。

$$(4) (\forall x) \{ \sim P(x) \vee \{ (\forall y) [\sim P(y) \vee P(f(x,y))] \} \wedge [Q(x,g(x)) \wedge \sim P(g(x))] \}$$

式中, $w=g(x)$ 为一Skolem函数。

(4) 消去存在量词

以Skolem函数代替存在量词内的约束变量, 然后消去存在量词

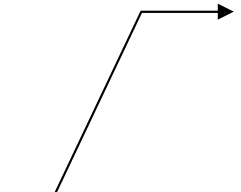
(5) 化为前束形

$$(5) (\forall x)(\forall y) \{ \sim P(x) \vee \{ [\sim P(y) \vee P(f(x,y))] \wedge [Q(x,g(x)) \wedge \sim P(g(x))] \}$$

把所有全称量词移到公式的左边, 并使每个量词的辖域包括这个量词后面公式的整个部分。

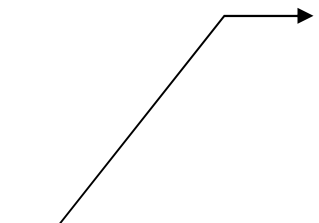
前束形 = {前缀} {母式}

全称量词串 无量词公式


$$(6) (\forall x)(\forall y) \{ [\sim P(x) \vee \sim P(y) \vee P(f(x,y))] \wedge [\sim P(x) \vee Q(x,g(x))] \wedge [\sim P(x) \vee \sim P(g(x))] \}$$

(6) 把母式化为合取范式

任何母式都可写成由一些谓词公式和(或)谓词公式的否定的析取的有限集组成的合取。

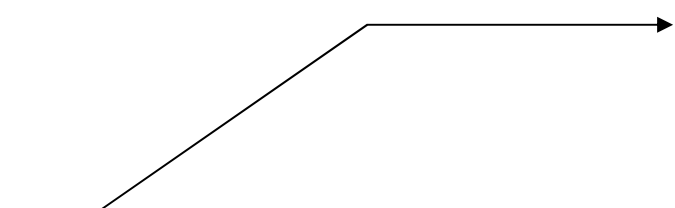

$$(7) \{ [\sim P(x) \vee \sim P(y) \vee P(f(x,y))] \wedge [\sim P(x) \vee Q(x,g(x))] \wedge [\sim P(x) \vee \sim P(g(x))] \}$$

(7) 消去全称量词

所有余下的量词均被全称量词量化了。消去前缀，即消去明显出现的全称量词。

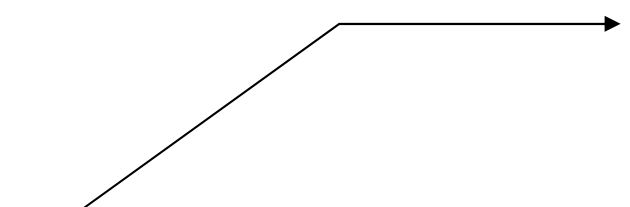
(8) 消去连词符号 \wedge

用 $\{\mathbf{A}, \mathbf{B}\}$ 代替 $(\mathbf{A} \wedge \mathbf{B})$ ，消去符号 \wedge 。最后得到一个有限集，其中每个公式是文字的析取。


$$(8) \quad \{ \sim P(x) \vee \sim P(y) \vee P(f(x,y)), \\ \sim P(x) \vee Q(x,g(x)), \\ \sim P(x) \vee \sim P(g(x)) \}$$

(9) 更换变量名称

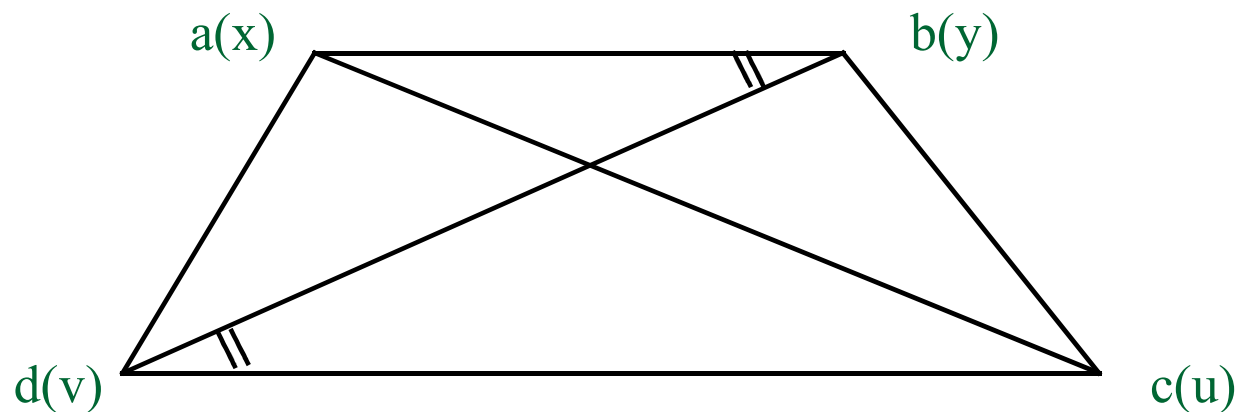
可以更换变量符号的名称，使一个变量符号不出现在一个以上的子句中。


$$(9) \quad \{ \sim P(x_1) \vee \sim P(y) \vee P[f(x_1,y)], \\ \sim P(x_2) \vee Q[x_2,g(x_2)], \\ \sim P(x_3) \vee \sim P[g(x_3)] \}$$

三、子句集建立举例

第一类：代数、几何证明（定理证明）

例1. 证明梯形的对角线与上下底构成的内错角相等



证明：①设梯形的顶点依次为a,b,c,d.引入谓词：

$T(x,y,u,v)$ 表示以xy为上底，uv为下底的梯形

$P(x,y,u,v)$ 表示 $xy \parallel uv$

$E(x,y,z,u,v,w)$ 表示 $\angle xyz = \angle uvw$

②问题的逻辑描述和相应的子句集为

i. 梯形上下底平行：

$$A_1 : (\forall x)(\forall y)(\forall u)(\forall v)(T(x, y, u, v) \rightarrow P(x, y, u, v))$$

$$SA_1 : \sim T(x, y, u, v) \vee P(x, y, u, v)$$

ii. 平行内错角相等

$$A_2 : (\forall x)(\forall y)(\forall u)(\forall v)(P(x, y, u, v) \rightarrow E(x, y, v, u, v, y))$$

$$SA_2 : \sim P(x, y, u, v) \vee E(x, y, v, u, v, y)$$

iii. 已知条件

$$A_3 : T(a, b, c, d)$$

$$SA_3 : T(a, b, c, d)$$

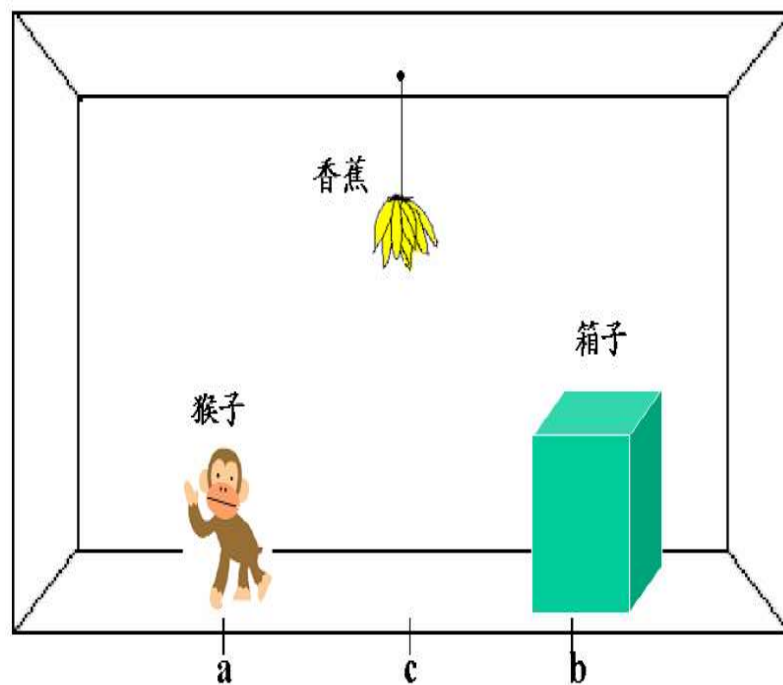
iv. 要证明的结论：B: $E(a,b,d,c,d,b)$

结论的“非”： $S \sim B : \sim E(a,b,d,c,d,b)$

从而 $S = \{SA_1, SA_2, SA_3, S \sim B\}$

第二类 机器人动作问题

例2. 猴子摘香蕉问题



初始状态 S_0

解:

1) 引入谓词

$P(x,y,z,s)$: 表示猴子位于 x 处, 香蕉位于 y 处, 台子位于 z 处, 状态为 s

$R(s)$: 表示 s 状态下猴子拿到香蕉

$ANS(s)$: 表示形式谓词, 只是为求得回答的动作序列而虚设的。

2) 引入状态转移函数

$Walk(y, z, s)$: 表示原状态 s 下, 在 $walk$ 作用下, 猴子从 y 走到 z 处所建立的新状态。

$Carry(y,z,s)$: 表示原状态 s 下, 在 $Carry$ 作用下, 猴子从 y 搬台子到 z 处所建立的新状态。

$Climb(s)$: 表示原状态 s 下, 在 $Climb$ 作用下, 猴子爬上台子所建立的新状态。

3) 初始状态为S0, 猴子位于a, 香蕉位于b, 台子位于c, 问题描述如下:

a) 猴子走到台子处 (从x $\xrightarrow{\text{walk}}$ z)

$$A_1 : (\forall x)(\forall y)(\forall z)(\forall s)(P(x, y, z, s) \rightarrow P(z, y, z, \text{walk}(x, z, s)))$$

$$S_{A1} : \sim P(x, y, z, s) \vee P(z, y, z, \text{walk}(x, z, s))$$

b) 猴子搬着台子到y处

$$A_2 : (\forall x)(\forall y)(\forall s)(P(x, y, x, s) \rightarrow P(y, y, y, \text{carry}(x, y, s)))$$

$$S_{A2} : \sim P(x, y, x, s) \vee P(y, y, y, \text{carry}(x, y, s))$$

c) 猴子爬上台子拿到香蕉

$$A_3 : (\forall s)(P(b, b, b, s) \rightarrow R(\text{Climb}(s)))$$

$$S_{A3} : \sim P(b, b, b, s) \vee R(\text{Climb}(s))$$

d) 初始条件

$$A_4 : (P(a, b, c, s_0))$$

$$S_{A4} : P(a, b, c, s_0)$$

e) 结论

$$B : (\exists s)R(s)$$

$$S \sim B : \sim R(s) \vee \text{ANS}(s)$$

第三类 程序设计自动化问题

例3：简单的程序集合问题

若一台计算机有寄存器a,b,c和累加器A，
要求自动设计实现

$$(a) + (b) \rightarrow c$$

的程序。

解:

1) 引入谓词

$P(u,x,y,z,s)$:表示累加器A,寄存器a,b,c分别放入u,x,y,z时的状态为s

$Load(x,s)$:表示状态s下,对任一寄存器x来说,实现 $(x) \rightarrow A$ 后的新状态

$Add(x,s)$:表示状态s下,对任一寄存器x来说,实现 $(x)+(A) \rightarrow A$ 后的新状态

$Store(x,s)$:表示状态s下,对任一寄存器x来说,实现 $(A) \rightarrow x$ 后的新状态

2) 问题描述

a. $((a) \rightarrow A)$:寄存器a中的值放入寄存器A中

$$A1 : (\forall u)(\forall x)(\forall y)(\forall z)(\forall s)(P(u, x, y, z, s) \rightarrow P(x, x, y, z, load(a, s)))$$

$$S_{A1} : \sim P(u, x, y, z, s) \vee P(x, x, y, z, load(a, s))$$

b. $((b)+(A) \rightarrow A)$

$$A2 : (\forall u)(\forall x)(\forall y)(\forall z)(\forall s)(P(u, x, y, z, s) \rightarrow P(u + y, x, y, z, add(b, s)))$$

$$S_{A2} : \sim P(u, x, y, z, s) \vee P(u + y, x, y, z, add(b, s))$$

c. $((A) \rightarrow C)$

$A3 : (\forall u)(\forall x)(\forall y)(\forall z)(\forall s)(P(u, x, y, z, s) \rightarrow P(u, x, y, u, \text{store}(c, s)))$

$S_{A3} : \sim P(u, x, y, z, s) \vee P(u, x, y, u, \text{store}(c, s))$

d. 初始状态d下，累加器A与寄存器a,b,c中的数值

$A4 : P(1,2,3,4, d)$

$S_{A4} : P(1,2,3,4, d)$

e. 结论

$B : (\exists u)(\exists x)(\exists y)(\exists s)(P(u, x, y, x + y, s))$

$S_{\sim B} : \sim P(u, x, y, x + y, s) \vee \text{ANS}(s)$

\Rightarrow

$\sim B = \sim (\exists u)(\exists x)(\exists y)(\exists s)(P(u, x, y, x + y, s))$

$= (\forall u)(\forall x)(\forall y)(\forall s)(\sim (P(u, x, y, x + y, s)))$

$S_{\sim B} : \sim P(u, x, y, x + y, s) \vee \text{ANS}(s)$

子句集 $S = \{SA1, SA2, SA3, SA4, S_{\sim B}\}$

谓词公式
不可满足性 \longleftrightarrow 子句集
不可满足性 ?

定理:

谓词公式不可满足的充要条件是其子句集不可满足。

Outline

- ❖ 自然演绎推理
- ❖ **SKOLEM**标准形及子句集
- ❖ 海伯伦定理
- ❖ 鲁宾逊归结原理
- ❖ 归结反演
- ❖ 应用归结反演求解问题
- ❖ 归结策略

H 域

定义（ H 域）设 S 为子句集，则按下述方法构造的域 H_∞ 称为海伯伦域，简记为 H 域。

(1) 令 H_0 是 S 中所有个体常量的集合，若 S 中不包含个体常量，则令 $H_0 = \{a\}$ ，其中 a 为任意指定的一个个体常量。

(2) 令 $H_{i+1} = H_i \cup \{ S \text{ 中所有 } n \text{ 元函数 } f(x_1, \dots, x_n) \mid x_j (j=1, 2, \dots, n) \text{ 是 } H_i \text{ 中的元素} \}$ ，其中 $i=1, 2, \dots$

□ 例求子句集 $S = \{P(x) \vee Q(x), R(f(y))\}$ 的 H 域。

解：指定一个常量 a 作为个体常量，则得：

$$H_0 = \{a\}$$

$$H_1 = H_0 \cup \{f(a)\} = \{a\} \cup \{f(a)\} = \{a, f(a)\}$$

$$H_2 = H_1 \cup \{f(a), f(f(a))\} = \{a, f(a), f(f(a))\}$$

$$H_3 = \{a, f(a), f(f(a)), f(f(f(a)))\}$$

⋮

$$H_\infty = \{a, f(a), f(f(a)), f(f(f(a))), \dots\}$$

海伯伦（Herbrand）定理

- ❖ **基子句**：用 H 域中的元素代换子句中的变元后所得的子句，其中的谓词称为**基原子**。
- ❖ **原子集**：子句集中所有基原子构成的集合。
- ❖ **子句集在 H 域上的解释**：对子句集中出现的常量、函数及谓词取值，一次取值就是一个解释。
- ❖ **海伯伦定理**：

子句集不可满足的**充要条件**是存在一个有限的不可满足的基子句集。

Outline

- ❖ 自然演绎推理
- ❖ **SKOLEM**标准形及子句集
- ❖ 海伯伦定理
- ❖ 鲁宾逊归结原理
- ❖ 归结反演
- ❖ 应用归结反演求解问题
- ❖ 归结策略

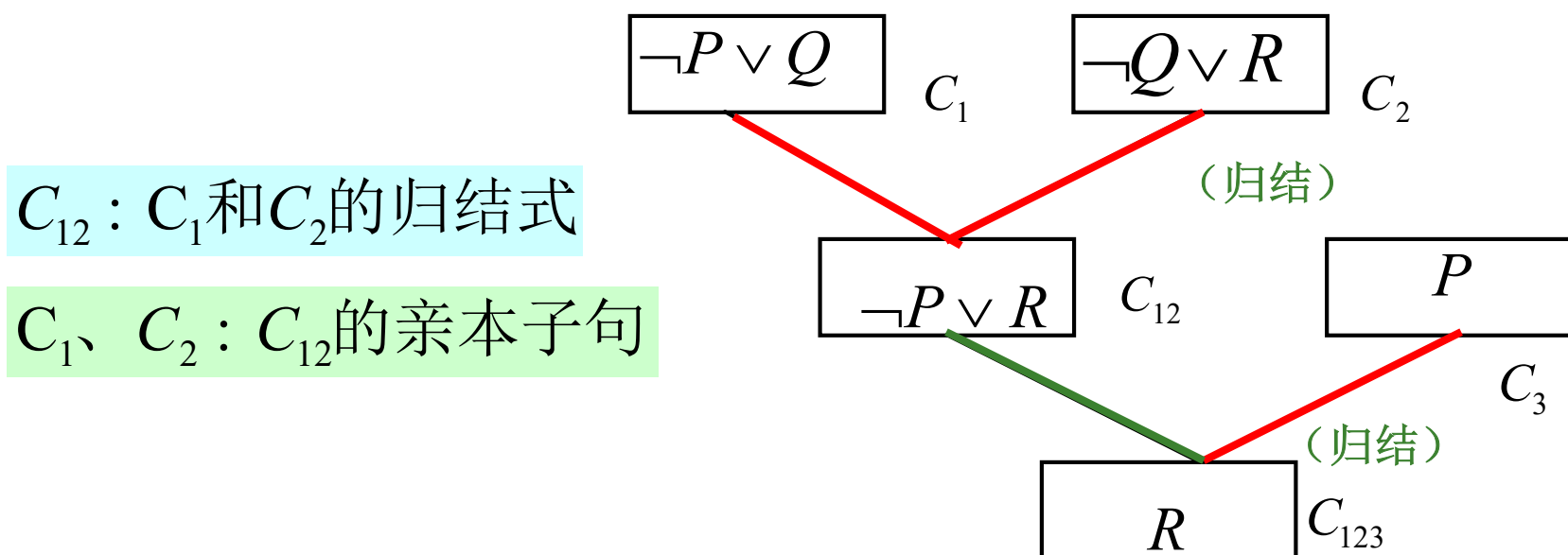
鲁宾逊归结原理

一、 鲁宾逊归结原理（消解原理）的基本思想：

- ▣ 检查子句集 S 中是否包含空子句，若包含，则 S 不可满足。
- ▣ 若不包含，在 S 中选择合适的子句进行归结，一旦归结出空子句，就说明 S 是不可满足的。

二、命题逻辑中的归结原理(基子句的归结)

- ❖ 定义：若 p 是原子谓词公式，则称 p 与 $\neg p$ 为**互补文字**。
- ❖ 定义：设 C_1 与 C_2 是子句集中的任意两个子句，如果 C_1 中的文字 L_1 与 C_2 中的文字 L_2 互补，那么从 C_1 和 C_2 中分别消去 L_1 和 L_2 ，并将两个子句中余下的部分析取，构成一个新子句 C_{12} ，则称这个过程为**归结**。



◆ 定理：归结式 C_{12} 是其亲本子句 C_1 与 C_2 的逻辑结论。即如果 C_1 与 C_2 为真，则 C_{12} 为真。

◆ 推论1：设 C_1 与 C_2 是子句集 S 中的两个子句， C_{12} 是它们的归结式，若用 C_{12} 代替 C_1 与 C_2 后得到新子句集 S_1 ，则由 S_1 不可满足性可推出原子句集 S 的不可满足性，即：

$$S_1 \text{ 的不可满足性} \Rightarrow S \text{ 的不可满足性}$$

◆ 推论2：设 C_1 与 C_2 是子句集 S 中的两个子句， C_{12} 是它们的归结式，若 C_{12} 加入原子句集 S ，得到新子句集 S_1 ，则 S 与 S_1 在不可满足的意义上是等价的，即：

$$S_1 \text{ 的不可满足性} \Leftrightarrow S \text{ 的不可满足性}$$

三、谓词逻辑中的归结原理

(含有变量的子句的归结)

$C_1 = P(x) \vee Q(x)$
 $C_2 = \neg P(a) \vee R(y)$

$\sigma = \{a/x\}$

$C_1\sigma = P(a) \vee Q(a)$
 $C_2\sigma = \neg P(a) \vee R(y)$
 $C_{12} = Q(a) \vee R(y)$

合一置换

- ❖ 设有公式集 $\{E_1, \dots, E_k\}$ 和置换 θ , 使

$$E_1 \theta = E_2 \theta = \dots E_k \theta$$

则称 E_1, \dots, E_k 是**可合一的**,

称 θ 为**合一置换** (union replacement) 。

- ❖ 若 E_1, \dots, E_k 有合一置换, 且对 E_1, \dots, E_k 的任一合一置换 σ 都有置换 λ 存在, 使得

$$\theta = \sigma \cdot \lambda$$

则称 σ 是 E_1, \dots, E_k 的**最一般合一置换**, 记作 ***mgu*** (most general Unifier)

定义：设 C_1 和 C_2 是两个没有相同变元的子句， L_1 和 L_2 分别是 C_1 和 C_2 中的文字，若 σ 是 L_1 和 $\neg L_2$ 的最一般合一，则称

$$C_{12} = (C_1\sigma - \{L_1\sigma\}) \vee (C_2\sigma - \{L_2\sigma\})$$

为 C_1 和 C_2 的二元归结式。

-
- ❖ 对于谓词逻辑，归结式是其亲本子句的逻辑结论。
 - ❖ 对于一阶谓词逻辑，若子句集是不可满足的，则必存在一个从该子句集到空子句的归结演绎；若从子句集存在一个到空子句的演绎，则该子句集是不可满足的。
 - ❖ 如果没有归结出空子句，则既不能说 S 不可满足，也不能说 S 是可满足的。
-

Outline

- ❖ 自然演绎推理
- ❖ **SKOLEM**标准形及子句集
- ❖ 海伯伦定理
- ❖ 鲁宾逊归结原理
- ❖ 归结反演
- ❖ 应用归结反演求解问题
- ❖ 归结策略

归结反演

- ❖ 应用归结原理**证明**定理的过程称为**归结反演**。
- ❖ 为证明 $A \rightarrow B$ 成立，其中 A, B 是谓词公式，使用反演过程，先建立

$$G = A \wedge \sim B$$

进而做出相应的子句集 S ，证明 S 是不可满足的。

- ❖ 过程：对 S 中的可归结的子句作归结，求得归结式，并将这归结式（新子句）仍放入 S 中，反复进行这个归结过程直至产生**空子句**为止。这时 S 必是不可满足的，从而证明 $A \rightarrow B$ 是成立的。

归结反演

❖ 步骤:

- (1) 将已知前提表示为谓词公式 F 。
- (2) 将待证明的**结论**表示为谓词公式 Q ，并**否定**得到 $\neg Q$ 。
- (3) 把谓词公式集 $\{F, \neg Q\}$ 化为子句集 S 。
- (4) 应用归结原理对子句集 S 中的子句进行归结，并把每次归结得到的归结式都并入到 S 中。如此反复进行，若出现**空子句**，则停止归结，证明了 Q 为真。

❖ 例：某公司招聘工作人员， A ， B ， C 三人应试，经面试后公司表示如下想法：

(1) 三人中至少录取一人。

(2) 如果录取 A 而不录取 B ，则一定录取 C 。

(3) 如果录取 B ，则一定录取 C 。

求证：公司一定录取 C 。

❖ 证明：公司的想法用谓词公式表示： $P(x)$: 录取 x

(1) $P(A) \vee P(B) \vee P(C)$

(2) $P(A) \wedge \neg P(B) \rightarrow P(C)$

(3) $P(B) \rightarrow P(C)$

■ 把要求证的结论用谓词公式表示出来并否定，得：

(4) $\neg P(C)$

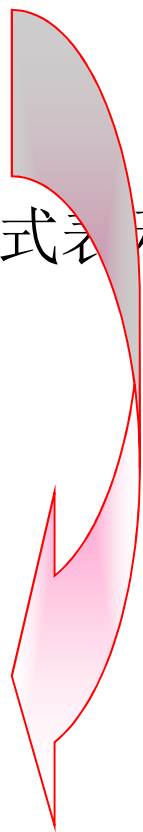
■ 把上述公式化成子句集：

(1) $P(A) \vee P(B) \vee P(C)$

(2) $\neg P(A) \vee P(B) \vee P(C)$

(3) $\neg P(B) \vee P(C)$

(4) $\neg P(C)$



应用归结原理进行归结：

(5) $P(B) \vee P(C)$

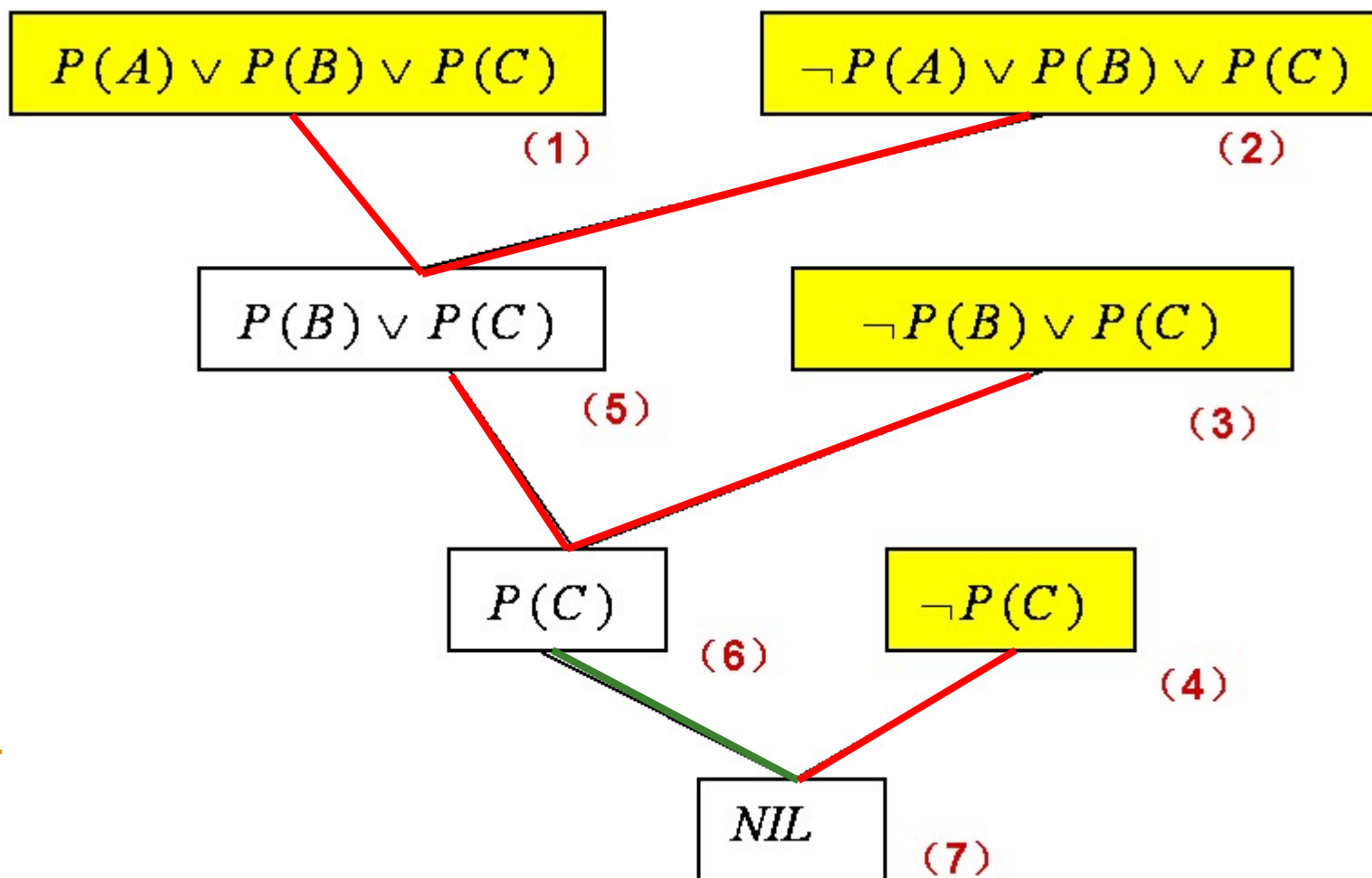
(6) $P(C)$

(7) NIL

(1) 与 (2) 归结

(3) 与 (5) 归结

(4) 与 (6) 归结



Outline

- ❖ 自然演绎推理
- ❖ **SKOLEM**标准形及子句集
- ❖ 海伯伦定理
- ❖ 鲁宾逊归结原理
- ❖ 归结反演
- ❖ 应用归结反演求解问题
- ❖ 归结策略

应用归结反演求解问题

❖ 步骤:

- (1) 已知前提 F 用谓词公式表示, 并化为子句集 S ;
- (2) 把待求解的问题 Q 用谓词公式表示, 并否定 Q , 再与 $ANSWER$ 构成析取式 ($\neg Q \vee ANSWER$);
- (3) 把 ($\neg Q \vee ANSWER$) 化为子句集, 并入到子句集 S 中, 得到子句集 S' ;
- (4) 对 S' 应用归结原理进行归结;
- (5) 若得到归结式 $ANSWER$, 则答案就在 $ANSWER$ 中。

例：设A, B, C三人中有人从不说真话，也有人从不说假话，某人向这三人分别提出一个问题：谁是说谎者？A答：“B和C都是说谎者”；B答：“A和C都是说谎者”；C答：“A和B至少有一个是说谎者”。求谁是老实人，谁是说谎者？

解：已知前提用谓词表示，设用 $T(x)$ 表示x说真话。

如果A说的是真话，则有 $T(A) \rightarrow \neg T(B) \wedge \neg T(C)$

如果A说的是假话，则有 $\neg T(A) \rightarrow T(B) \vee T(C)$

如果B说的是真话，则有 $T(B) \rightarrow \neg T(A) \wedge \neg T(C)$

如果B说的是假话，则有 $\neg T(B) \rightarrow T(A) \vee T(C)$

如果C说的是真话，则有 $T(C) \rightarrow \neg T(A) \vee \neg T(B)$

如果C说的是假话，则有 $\neg T(C) \rightarrow T(A) \wedge T(B)$

❖ 化成子句集，得到 S

$$(1) \neg T(A) \vee \neg T(B)$$

$$(2) \neg T(A) \vee \neg T(C)$$

$$(3) T(A) \vee T(B) \vee T(C)$$

$$(4) \neg T(B) \vee \neg T(C)$$

$$(5) \neg T(C) \vee \neg T(A) \vee \neg T(B)$$

$$(6) T(C) \vee T(A)$$

$$(7) T(C) \vee T(B)$$

前提：

$$T(A) \rightarrow \neg T(B) \wedge \neg T(C)$$

$$\neg T(A) \rightarrow T(B) \vee T(C)$$

$$T(B) \rightarrow \neg T(A) \wedge \neg T(C)$$

$$\neg T(B) \rightarrow T(A) \vee T(C)$$

$$T(C) \rightarrow \neg T(A) \vee \neg T(B)$$

$$\neg T(C) \rightarrow T(A) \wedge T(B)$$

❖ 求谁是老实人？

把 $\neg T(X) \vee ANSWER(X)$ 并入S得到S1。则S1比S多如下子句

(8) $\neg T(X) \vee ANSWER(X)$

(1) $\neg T(A) \vee \neg T(B)$
(2) $\neg T(A) \vee \neg T(C)$
(3) $T(A) \vee T(B) \vee T(C)$
(4) $\neg T(B) \vee \neg T(C)$
(5) $\neg T(C) \vee \neg T(A) \vee \neg T(B)$
(6) $T(C) \vee T(A)$
(7) $T(C) \vee T(B)$

❖ 应用归结原理对S1进行归结：

(9) $\neg T(A) \vee T(C)$ (1)和(7)进行归结

(10) $T(C)$ (6)和(9)进行归结

(11) $ANSWER(C)$ (8)和(10)进行归结

$\therefore C$ 是老实人。

此题中无论如何对S1进行归结，都推不出 $ANSWER(B)$ 与 $ANSWER(A)$ 。

❖ 证明A与B不是老实人:

设A不是老实人, 则有 $\neg T(A)$, 把它否定并入S中, 得到子句集 S_2 , 即子句集 S_2 比S多如下一个子句:

(8) $\neg(\neg T(A))$, 即 $T(A)$

❖ 应用归结原理对 S_2 进行归结:

(9) $\neg T(A) \vee T(C)$ (1) 和 (7) 归结

(10) $\neg T(A)$ (2) 和 (9) 归结

(11) NIL (8) 和 (10) 归结

A不是老实人。同理可证明B也不是老实人

(1) $\neg T(A) \vee \neg T(B)$
(2) $\neg T(A) \vee \neg T(C)$
(3) $T(A) \vee T(B) \vee T(C)$
(4) $\neg T(B) \vee \neg T(C)$
(5) $\neg T(C) \vee \neg T(A) \vee \neg T(B)$
(6) $T(C) \vee T(A)$
(7) $T(C) \vee T(B)$

Outline

- ❖ 自然演绎推理
- ❖ **SKOLEM**标准形及子句集
- ❖ 海伯伦定理
- ❖ 鲁宾逊归结原理
- ❖ 归结反演
- ❖ 应用归结反演求解问题
- ❖ 归结策略

归结的一般过程

设有子句集 $S=\{C_1, C_2, C_3, C_4\}$ ，其中 C_1, C_2, C_3, C_4 是 S 中的子句，
则归结的一般过程为：

- ①从子句 C_1 开始，逐个与 C_2, C_3, C_4 进行比较，归结。然后用 C_2 与 C_3 进行比较，归结。最后用 C_3 和 C_4 比较，归结。得到第一级归结式。
- ②再从 C_1 开始，用 S 中的子句分别与第一级归结式中的子句逐个地进行比较，归结，得第二级归结式。
- ③仍从 C_1 开始，用 S 中的子句集第一级归结式中的子句逐个地与第二级归结式中的子句进行比较，得第三级归结式。
继续直到出现空子句或不能再继续归结为止。

例 设有子句集 $S = \{P, \neg R, \neg P \vee Q, \neg Q \vee R\}$ **归结过程为：**

S: (1) P

(2) $\neg R$

(3) $\neg P \vee Q$

(4) $\neg Q \vee R$

S1: (5) Q

(1) 与 (3) 归结

(6) $\neg Q$

(2) 与 (4) 归结

(7) $\neg P \vee R$

(3) 与 (4) 归结

S2: (8) R

(1) 与 (7)

(9) $\neg P$

(2) 与 (7)

(10) $\neg P$

(3) 与 (6)

(11) R

(4) 与 (5)

S3: (12) NIL

(1) 与 (9)

归结策略

- ❖ 归结策略大致可分为两大类：
 - ◆ 删除策略：通过删除某些无用的子句来缩小归结的范围；
 - ◆ 限制策略：通过对参加归结的子句进行种种限制，尽可能减小归结的盲目性，使其尽快地归结出空子句。

(1) 删除策略

❖ 纯文字删除法:

如果某文字 L 在子句集中不存在可与之互补的文字 $\neg L$ ，则称该文字为纯文字。

例 $S=\{P\vee Q\vee R, \neg Q\vee R, Q, \neg R\}$

❖ 重言式删除法:

如果一个子句中同时包含互补文字对，则称该子句为重言式。

例 $P(X)\vee\neg P(X)$ ， $P(X)\vee Q(X)\vee\neg P(X)$

❖ 包孕删除法:

设有子句 C_1 和 C_2 ，如果存在一个代换 σ ，使得 $C_1\sigma\subseteq C_2$ ，则称 C_1 包孕于 C_2

例 $P(X)$ 包孕于 $P(y)\vee Q(z)$ $\sigma=\{y/x\}$

(2) 支持集策略

- ❖ 1965年沃斯等人提出的一种归结策略。
- ❖ **核心**：每一次归结时，亲本子句中至少应有一个是由目标公式的否定所得到的子句，或者是它们的后裔。
- ❖ 支持集策略是**完备**的，若子句集是不可满足的，归结为空。

例

设有子集

$S = \{\neg I(X) \vee R(X), I(a), \neg R(y) \vee \neg L(y), L(a)\}$, 其中
 $\neg I(X) \vee R(X)$ 是目标公式否定后得到的子句。

用支持集策略进行归结的过程是

S: (1) $\neg I(x) \vee R(x)$	S1: (5) $R(a)$	(1) 与 (2)
(2) $I(a)$	(6) $\neg I(x) \vee \neg L(x)$	(1) 与 (3)
(3) $\neg R(y) \vee \neg L(y)$	S2: (7) $\neg L(a)$	(2) 与 (6)
(4) $L(a)$	(8) $\neg L(a)$	(3) 与 (5)
	(9) $\neg I(a)$	(4) 与 (6)
	S3: (10) NIL	(2) 与 (9)

(3) 线性输入策略

- ❖ 参加归结的两个子句中必须至少有一个是初始子句集中的子句。

例： 有子句集

$$S = \{\neg I(x) \vee R(x), I(a), \neg R(y) \vee L(y), L(a)\}$$

用线性输入策略对子句集进行归结：

S: (1) $\neg I(x) \vee R(x)$

(2) $I(a)$

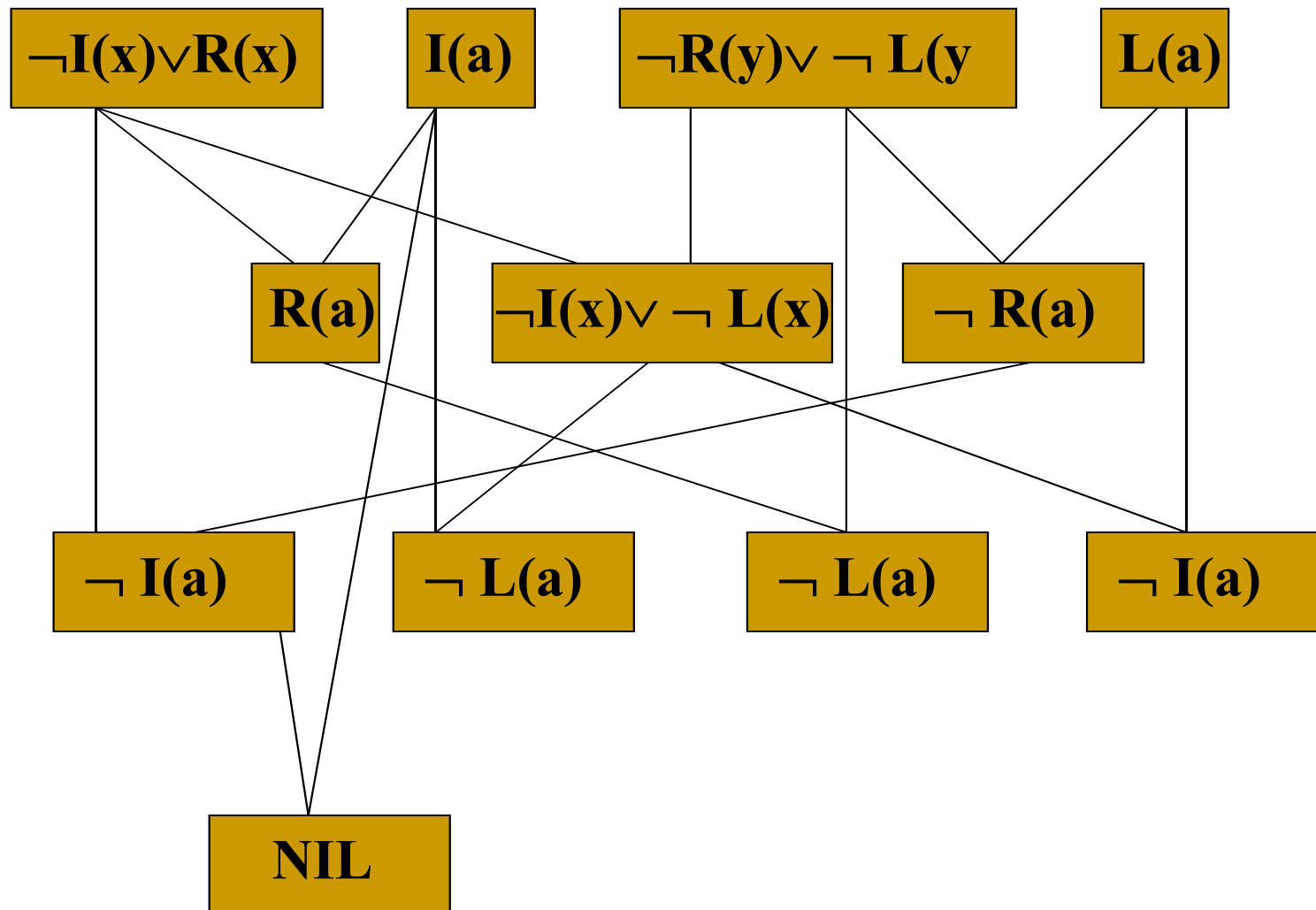
(3) $\neg R(y) \vee \neg L(y)$

(4) $L(a)$

S_1 : (5) $R(a)$ (1)与(2)

(6) $\neg I(x) \vee \neg L(x)$ (1)与(3)

(7) $\neg R(a)$ (3)与(4)



(4) 单文字子句策略

- ❖ 单文字子句策略要求参加归结的两个子句中必须至少有一个是单文字子句
- ❖ 单文字子句：一个子句只包含一个文字

例 有子句集 $S = \{\neg I(x) \vee R(x), I(a), \neg R(y) \vee \neg L(y), L(a)\}$

归结过程:

S: (1) $\neg I(x) \vee R(x)$

(2) $I(a)$

(3) $\neg R(y) \vee \neg L(y)$

(4) $L(a)$

S_1 : (5) $R(a)$ (1)与(2)

(6) $\neg R(a)$ (3)与(4)

S_2 : (7) $\neg I(a)$ (1)与(6)

(8) $\neg L(a)$ (3)与(5)

S_3 : (9) NIL (2)与(7)

(5) 祖先过滤形策略

祖先过滤形策略的可归结条件：

- ① C_1 与 C_2 中至少有一个是初始子句集中的子句。
- ② 如果两个子句都不是初始子句集中的子句，则一个应是另一个的祖先， C_2 是由 C_1 与别的子句归结后得到的归结式。

例 设有子句集

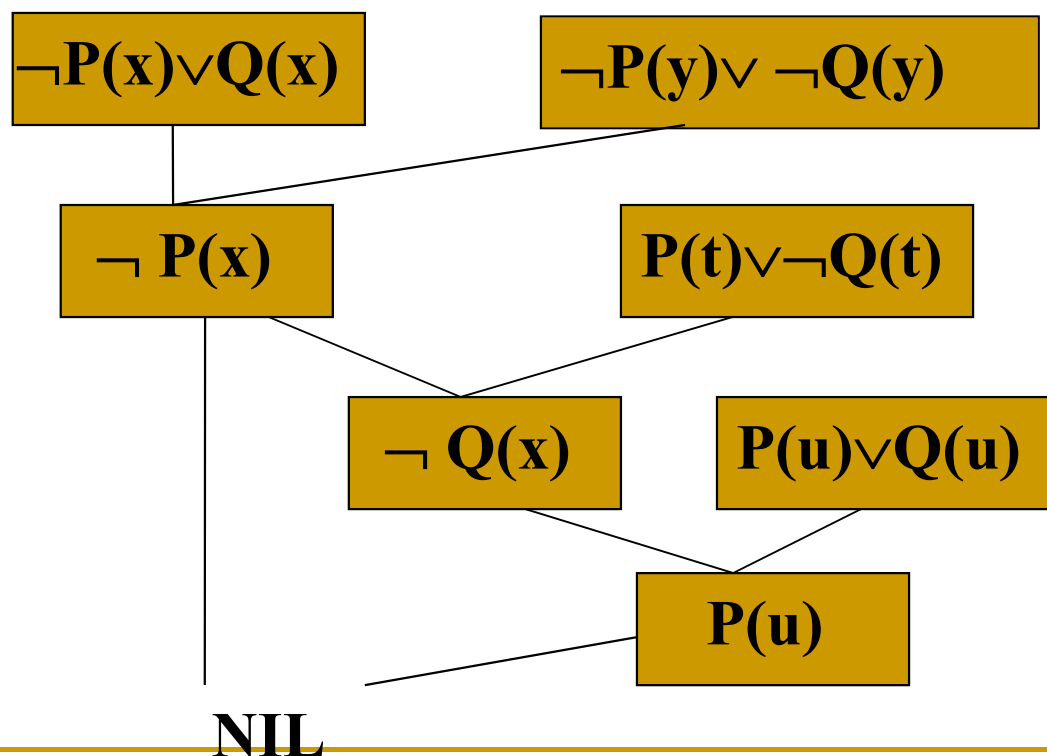
$S = \{ \neg P(x) \vee Q(x),$

$\neg P(y) \vee \neg Q(y),$

$P(u) \vee Q(u),$

$P(t) \vee \neg Q(t) \}$

归结如下



小结

归结反演推理是在自动证明领域影响较大的一种推理方法。

优点： 比较简单；
便于在计算机上实现。

缺点： 主要因为它要求将逻辑公式转换为子句集，因而带来一些问题：

(1) 不便于阅读和理解

$(\forall x) (\text{Bird}(x) \rightarrow \text{Canfly}(x))$ “鸟能飞”
用子句表示 $\neg \text{Bird}(x) \vee \text{Canfly}(x)$ 则不好理解。

(2) 在转化成子句的过程中有可能丢失一些重要的控制信息。

谢谢！