

# 人工智能 知识表示与推理

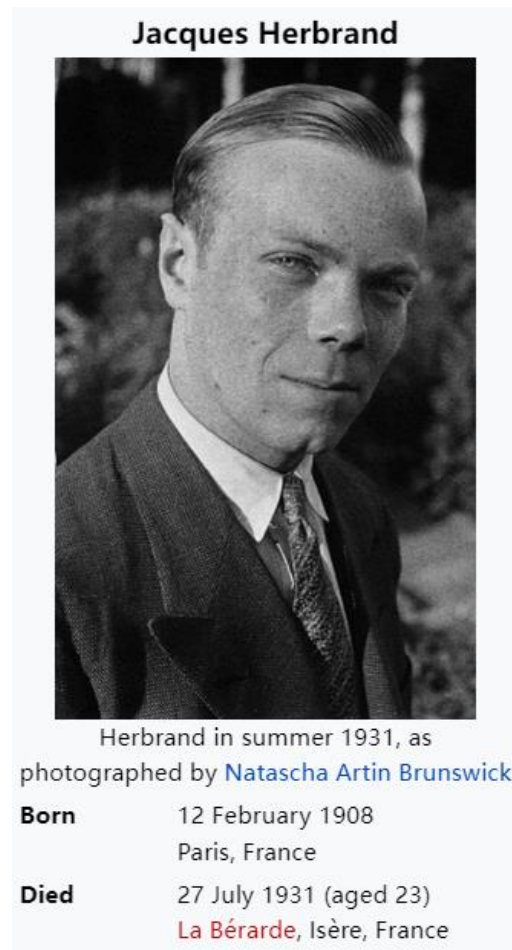
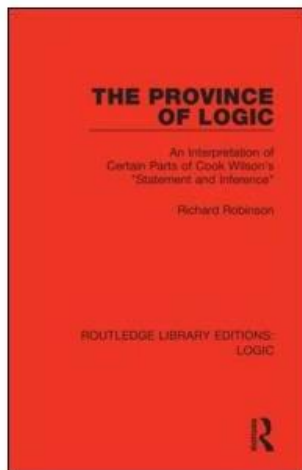
课件来源：陈川老师  
中山大学 计算机学院



中山大學  
SUN YAT-SEN UNIVERSITY

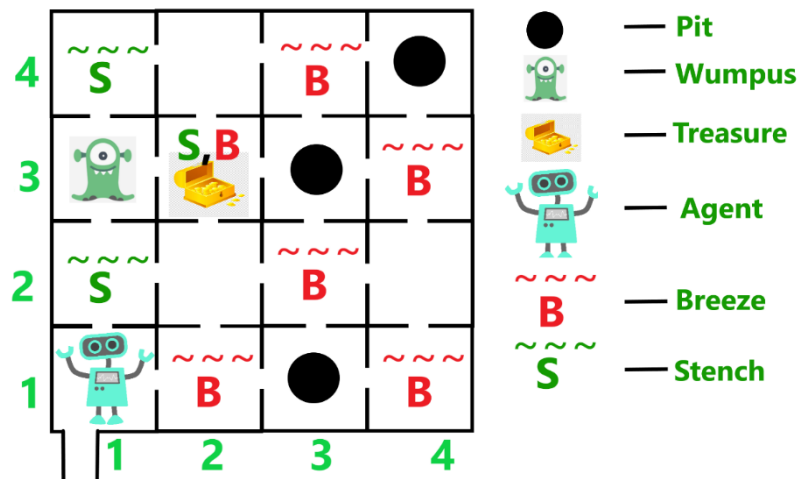
## 2.4 归结推理

- 定理证明即证明 $P \rightarrow Q (\neg P \vee Q)$ 的永真性。根据反证法，只要证明其否定 $(P \wedge \neg Q)$ 不可满足性即可。
- 海伯伦(Herbrand)定理**(1930)为自动定理证明奠定了理论基础；**鲁滨逊(Robinson)归结原理**(1965)年提出的使机器定理证明成为现实。



## 2.4 归结推理

- 归结法的基本思想
- 命题逻辑归结原理和过程
- 谓词逻辑归结原理和过程
- 应用归结原理求解问题
- 归结反演
- 归结策略



The Wumpus World

## 2.4 归结推理

### 什么是归结原理？

- 在定理证明系统中，已知一个公式集  $F_1, F_2, \dots, F_n$ ，要证明一个公式  $W$  (定理) 是否成立，即要证明  $W$  是公式集的逻辑推论时，一种证明法就是要证明  $F_1 \wedge F_2 \wedge \dots \wedge F_n \rightarrow W$  为永真式。
- **反证法**：证明  $F = F_1 \wedge F_2 \wedge \dots \wedge F_n \wedge \neg W$  为永假，这等价于证明  $F$  对应的**子句集**  $S = \{F_1, F_2, \dots, F_n, \neg W\}$  为不可满足的。

## 2.4 归结推理

### 子句集

- 文字（**literal**）：原子公式及其否定。例如， $P$ ：正文字， $\neg P$ ：负文字。
- 子句（**clause**）：任何文字的析取。某个文字本身也都是子句。

$$P \vee Q \vee \neg R, \text{ 记作 } (P, Q, \neg R)$$

- 空子句（**NIL**）：不包含任何文字的子句。

空子句是永假的，不可满足的。

- 子句集：由子句构成的集合（子句的合取）。

## 2.4 归结推理

### 归结式的定义与性质

- 对于任意两个子句 $C_1$ 和 $C_2$ ，若 $C_1$ 中有一个文字 $L$ ，而 $C_2$ 中有一个与 $L$ 成互补的文字 $\neg L$ ，则分别从 $C_1$ 和 $C_2$ 中删去 $L$ 和 $\neg L$ ，并将其剩余部分组成新的析取式。这个新的子句被称为 $C_1$ 和 $C_2$ 关于 $L$ 的**归结式**， $C_1$ 和 $C_2$ 则是该归结式的亲本子句
- 例如， $P$ 和 $\neg P$ 的归结式为空子句，记作 $()$ 、 $\square$ 或 $NIL$ ； $(W, R, Q)$ 和 $(W, S, \neg R)$ 关于 $R$ 的归结式为 $(W, Q, S)$

**定理：**两个子句的归结式是这两个子句的逻辑推论，如 $\{(L, C_1), (\neg L, C_2)\} \vdash (C_1, C_2)$

**定理:**两个子句 $C_1$ 和 $C_2$ 的归结式 $C$ 是 $C_1$ 和 $C_2$ 的逻辑推论。

**证:** 设 $C_1 = L \vee C_1'$ ,  $C_2 = (\neg L) \vee C_2'$  关于解释 $I$ 为真, 则需证明  
 $C = C_1' \vee C_2'$  关于解释 $I$ 也为真。

关于解释 $I$ ,  $L$ 和 $\neg L$ 二者中必有一个为假。若 $L$ 为假, 则 $C_1'$ 必为真, 否则 $C_1$ 为假, 这与前提假设矛盾, 所以只能是 $C_1'$ 为真。

同理, 若 $\neg L$ 为假, 则 $C_2'$ 必为真。最后有 $C = C_1' \vee C_2'$  关于解释 $I$ 为真, 即 $C$ 是 $C_1$ 和 $C_2$ 的逻辑推论。

**推论：**子句集 $S=\{C_1, C_2, \dots, C_n\}$ 与子句集 $S_1=\{C, C_1, C_2, \dots, C_n\}$ 的不可满足性是等价的 (其中 $C$ 是 $C_1$ 和 $C_2$ 的归结式)。

**证：**设 $S$ 是不可满足的，则 $C_1, C_2, \dots, C_n$ 中必有一为假，因而 $S_1$ 必为不可满足的。

设 $S_1$ 是不可满足的，则对于不满足 $S_1$ 的任一解释 $I$ ，可能有两种情况：

(1)  $I$ 使 $C$ 为真，则 $C_1, C_2, \dots, C_n$ 中必有一子句为假，因而 $S$ 是不可满足的。

(2)  $I$ 使 $C$ 为假，则根据定理有归结式 $C=(C_1, C_2)$ 为假，即 $I$ 或使 $C_1$ 为假，或使 $C_2$ 为假，因而 $S$ 也是不可满足的。

由此可见 $S$ 和 $S_1$ 的不可满足性是等价的。



- 同理可证 $S_i$ 和 $S_{i+1}$  (由 $S_i$ 导出的扩大的子句集) 的不可满足性也是等价的, 其中 $i=1, 2, \dots$ 。
- **归结原理**就是从子句集 $S$ 出发, 应用归结推理规则导出子句集 $S_1$ 。再从 $S_1$ 出发导出 $S_2$ , 依此类推, 直到某一个子句集 $S_n$ 出现空子句为止。
- 根据不可满足性等价原理, 已知**若 $S_n$ 为不可满足的, 则可逆向依次推得 $S$ 必为不可满足的。**
- **用归结法, 过程比较单钝, 只涉及归结推理规则的应用问题, 因而便于实现机器证明。**

## 2.4 归结推理

### 归结推导

- 从一个子句集 $S$ （如 $KB$ ）推导出一个子句 $C$ 的过程中会产生一系列子句 $C_1, C_2, \dots, C_n$ ，其中 $C_n = C$ ，且对于 $C_i$  ( $i = 1, 2, \dots, n-1$ )均有：
  - $C_i \in S$
  - 或者 $C_i$ 是推导过程中产生的某两个子句的归结式
- 从 $S$ 推导出 $C$ 记为： $S \vdash C$

## 2.4 归结推理

### 归结推导的合理性

■ **定理：** 如果  $S \vdash C$ ，那么  $S \models C$

■ **证明：**

令  $S$  推导出  $C$  产生的子句序列为  $C_1, C_2, \dots, C_n$

通过数学归纳法证明对于  $i \in [1, n]$ ， $S \models C_i$  均成立

■ 反之，若  $S \models C$ ，则从  $S$  中不一定能够推导出  $C$ 。

## 2.4 归结推理

### 归结法的合理性和完备性

- 定理： $S \vdash ()$ ，当且仅当 $S \models ()$ ，当且仅当 $S$ 不可满足
- 由前文可知， $KB \models \alpha$ ，当且仅当 $KB \wedge \neg \alpha$ 不可满足。结合上述定理，我们通过下述过程来判断 $KB \models \alpha$ 是否成立：
  - 记 $KB \wedge \neg \alpha$ 的子句集为 $S$
  - 判断 $S \vdash ()$ 是否成立，即从 $S$ 中能否推导出空子句
- 归结法的过程比较单纯，只涉及归结推理规则的应用问题，因而便于实现机器证明。

## 2.4 归结推理

### 鲁宾逊归结原理

◆ 子句集中子句之间是合取关系，只要有一个子句不可满足，则子句集就不可满足。

- ◆ 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。

## 2.4 归结推理

### 命题逻辑的归结原理和过程

命题逻辑中，若给定前提集 $F$ 和命题 $R$ ，则归结证明过程可归纳如下：

- (1) 把 $F$ 转化成子句集表示，得到子句集 $S_0$ ；
- (2) 把命题 $R$ 的否定式 $\neg R$ 也转化成子句集表示，并将其加到 $S_0$ 中，得 $S = S_0 \cup S_{\neg R}$ ，
- (3) 对子句集 $S$ 反复应用归结推理规则（推导），直至导出含有空子句的扩大子句集为止。即出现归结式为空子句时，表明已找到矛盾，证明过程结束。

## 命题逻辑的归结原理和过程

例1： 设已知前提集为

$$P \dots\dots\dots(1) \quad (P \wedge Q) \rightarrow R \dots\dots(2)$$

$$(S \vee T) \rightarrow Q \dots(3) \quad T \dots\dots\dots(4)$$

求证 $R$ 。

证明：化成子句集

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

- 归结可用图的演绎树表示，  
由于根部出现空子句，因此  
命题 $R$ 得证。

## 命题逻辑的归结原理和过程

例1： 设已知前提集为

$P \dots\dots\dots(1)$

$(P \wedge Q) \rightarrow R \dots\dots(2)$

$(S \vee T) \rightarrow Q \dots(3)$

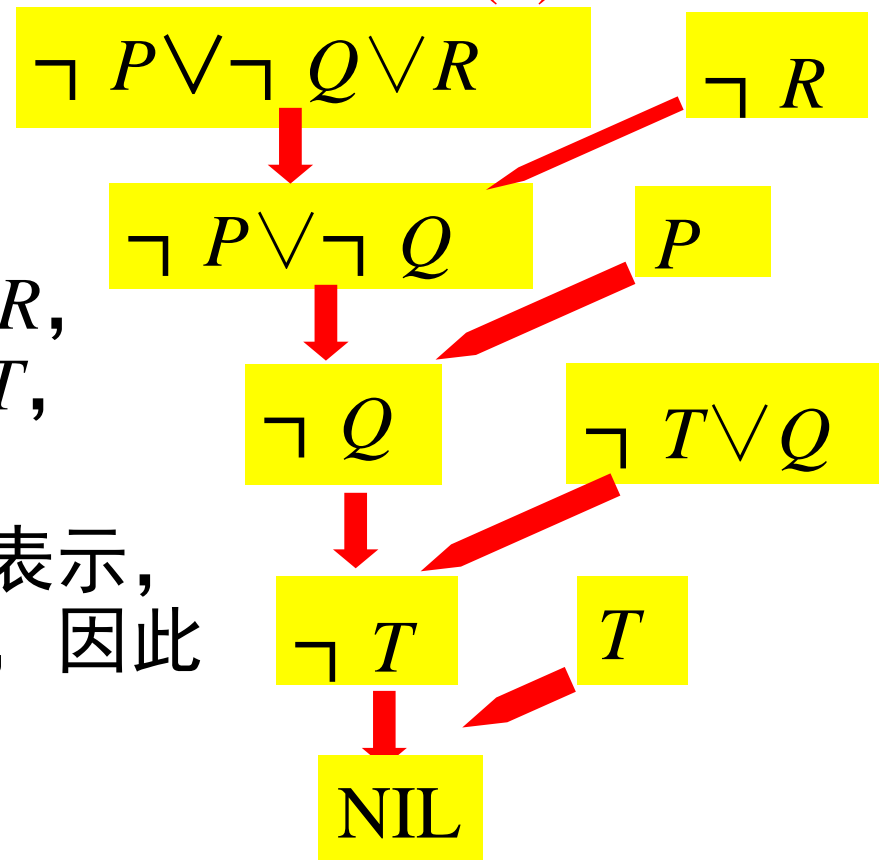
$T \dots\dots\dots(4)$

求证 $R$ 。

证明： 化成子句集

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

- 归结可用图的演绎树表示，由于根部出现空子句，因此命题 $R$ 得证。





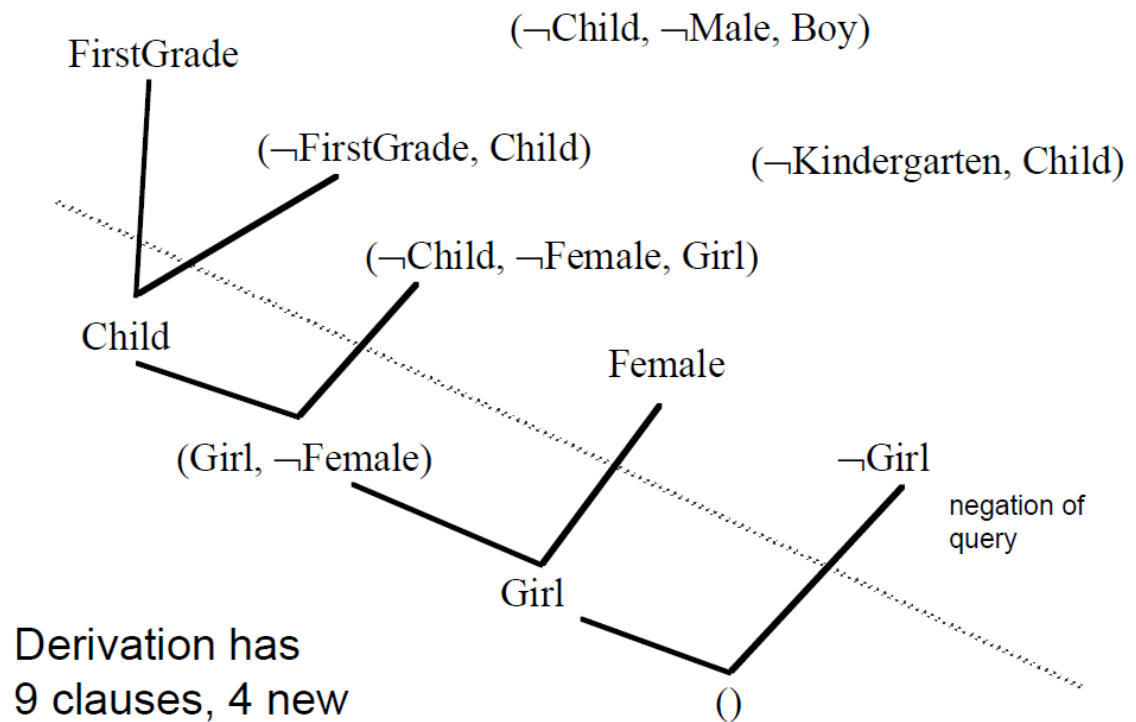
# 命题逻辑的归结原理和过程

## 例2:

KB

FirstGrade  
 FirstGrade  $\rightarrow$  Child  
 Child  $\wedge$  Male  $\rightarrow$  Boy  
 Kindergarten  $\rightarrow$  Child  
 Child  $\wedge$  Female  $\rightarrow$  Girl  
 Female

Show that  $KB \models \text{Girl}$



## 谓词逻辑的归结原理和过程

- 在谓词逻辑中，由于子句中含有变元，所以不能像命题逻辑那样直接消去互补文字，而需要先对变元进行代换，然后才能进行归结。

例如，设有两个子句

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

由于 $P(x)$ 与 $P(a)$ 不同，所以 $C_1$ 与 $C_2$ 不能直接进行归结。但是若引入替换

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

对两个子句分别进行代换：

$$\begin{aligned} C_1 \sigma &= P(a) \vee Q(a) \\ C_2 \sigma &= \neg P(a) \vee R(y) \end{aligned}$$

就可对它们进行归结，得到归结式：

$$Q(a) \vee R(y)$$

## 前束范式

**定义** 设F为一谓词公式，如果其中的所有量词均不以否定形式出现在公式之中，而他们的辖域为整个公式，则称F为**前束范式**，一般写成

$$(Q_1x_1) \dots (Q_nx_n)M(x_1, \dots, x_n)$$

其中 $Q_i (i = 1, 2, \dots, n)$ 为前缀， $(Q_1x_1) \dots (Q_nx_n)$ 是一个由全称量词或存在量词组成的量词串， $M(x_1, \dots, x_n)$ 为母式，它是一个不含任何量词的谓词公式

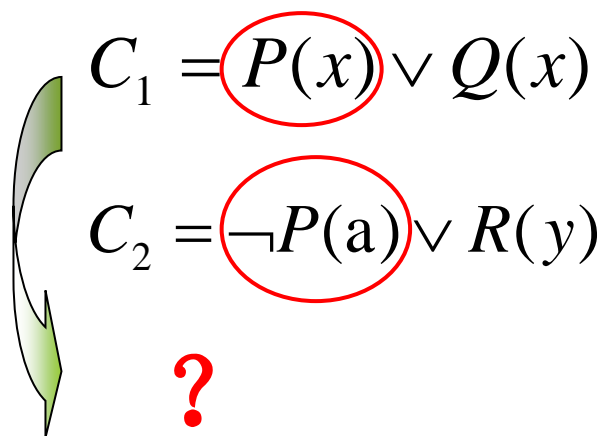
## 公式化为前束范式

- (1) **消去蕴涵符号** ( $\leftrightarrow$   $\rightarrow$  联结词)。
- (2) **内移否定词**  $\neg$  的辖域范围。
- (3) **变量标准化**。对变量作必要的换名，使每一量词只约束一个唯一的变量名。
- (4) **把所有量词都集中到公式左面**，移动时不要改变其相对顺序。

## 谓词逻辑的归结原理和过程

在谓词逻辑中应用归结法时，首先需要：

- (1) 将所有谓词公式（包括知识库 $KB$ 和查询 $\alpha$ ）化为子句集
- (2) 通过合一，对含有变量的子句进行归结


$$\begin{aligned} C_1 &= P(x) \vee Q(x) \\ C_2 &= \neg P(a) \vee R(y) \end{aligned}$$

?

## A. 谓词公式转化子句集

$$\forall x(\forall yP(x, y) \rightarrow \neg \forall y(Q(x, y) \rightarrow R(x, y)))$$

谓词公式化为子句集的步骤：

(1) **消去蕴涵和等价符号** ( $\rightarrow$ 和 $\leftrightarrow$ 联结词)。

$$P \rightarrow Q \Leftrightarrow \neg P \vee Q, \quad P \leftrightarrow Q \Leftrightarrow (P \wedge Q) \vee (\neg P \wedge \neg Q)$$

$$\longleftrightarrow \forall x(\neg \forall yP(x, y) \vee \neg \forall y(\neg Q(x, y) \vee R(x, y)))$$

(2) **内移否定符号 $\neg$** ，将其移到紧靠谓词的位置上。

双重否定律  $\neg(\neg P) \Leftrightarrow P$

德.摩根律  $\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q, \quad \neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$

量词转换律  $\neg \exists xP \Leftrightarrow \forall x\neg P, \quad \neg \forall xP \Leftrightarrow \exists x\neg P$

$$\longleftrightarrow \forall x(\exists y\neg P(x, y) \vee \exists y(Q(x, y) \wedge \neg R(x, y)))$$

## A. 谓词公式转化子句集

$$\forall x(\exists y\neg P(x, y) \vee \exists y(Q(x, y) \wedge \neg R(x, y)))$$

谓词公式化为子句集的步骤：

- (3) **变量标准化**。对变量作必要的换名，使每一量词只约束一个唯一的变量名。

$$\exists xP(x) \equiv \exists yP(y), \quad \forall xP(x) \equiv \forall yP(y)$$

$$\longleftrightarrow \forall x(\exists y\neg P(x, y) \vee \exists z(Q(x, z) \wedge \neg R(x, z)))$$

- (4) **消去存在量词 (Skolemize)**。对于待消去的存在量词，若不在任何全称量词辖域之内，则用Skolem常量替代公式中存在量词约束的变量；若受全称量词约束，则要用Skolem函数替代存在量词约束的变量，然后就可消去存在量词。

## A. 谓词公式转化子句集

- a. 存在量词不出现在全称量词的辖域内，则只要用一个新的个体常量替换受该量词约束的变元。
- b. 存在量词位于一个或者多个全称量词的辖域内，此时要用Skolem函数 $f(x_1, x_2, \dots, x_n)$ 替换受该存在量词约束的变元。

$$\forall x(\exists y \neg P(x, y) \vee \exists z(Q(x, z) \wedge \neg R(x, z)))$$

Skolemize:

对于一般情况

$$\forall x_1 \forall x_2 \cdots \forall x_n \exists y P(x_1, x_2, \dots, x_n, y)$$

存在量词 $y$ 的Skolem函数为 $y = f(x_1, x_2, \dots, x_n)$

Skolem化：用Skolem函数替代存在量词约束的变量的过程。

$$\begin{array}{l} y = f(x), \\ z = g(x) \end{array} \iff \forall x(\neg P(x, f(x)) \vee (Q(x, g(x)) \wedge \neg R(x, g(x))))$$

(5) **化为前束型**。前束型=（前缀）{母式}。其中，前缀为全称量词串，母式为不含量词的谓词公式。



## A. 谓词公式转化子句集

$$\forall x(\neg P(x, f(x)) \vee (Q(x, g(x)) \wedge \neg R(x, g(x))))$$

谓词公式化为子句集的步骤：

- (6) **把母式化成合取范式**。反复使用结合律和分配律，将母式表达成合取范式的Skolem标准形。

**Skolem 标准形：**  $\forall x_1 \forall x_2 \cdots \forall x_n M$

**$M$ ：**子句的合取式，称为Skolem标准形的母式。

$$P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R)$$

$$P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \wedge (P \vee R)$$

$$\longleftrightarrow \forall x((\neg P(x, f(x)) \vee Q(x, g(x))) \wedge (\neg P(x, f(x)) \vee \neg R(x, g(x))))$$

- (7) **略去全称量词**。由于母式的变量均受全称量词的约束，因此可省略掉全称量词。

$$\longleftrightarrow (\neg P(x, f(x)) \vee Q(x, g(x))) \wedge (\neg P(x, f(x)) \vee \neg R(x, g(x)))$$

## A. 谓词公式转化子句集

$$(\neg P(x, f(x)) \vee Q(x, g(x))) \wedge (\neg P(x, f(x)) \vee \neg R(x, g(x)))$$

谓词公式化为子句集的步骤：

(8) **把母式用子句集表示**。把母式中每一个合取元称为一个子句，省去合取联结词，这样就可把母式写成集合的形式表示，每一个元素就是一个子句。

$$\longleftrightarrow \{(\neg P(x, f(x)), Q(x, g(x))), (\neg P(x, f(x)), \neg R(x, g(x)))\}$$

(9) **子句变量标准化**。对某些变量重新命名，使任意两个子句不会有相同的变量出现。这是因为在使用子句集进行证明推理的过程中，有时需要例化某一个全称量词约束的变量，该步骤可以使公式尽量保持其一般化形式，增加了应用过程的灵活性。

$$\longleftrightarrow \{(\neg P(x, f(x)), Q(x, g(x))), (\neg P(y, f(y)), \neg R(y, g(y)))\}$$

# 子句集的性质与意义

## 性质

- (1) 子句集中子句之间是合取关系。
- (2) 子句集中的变元受全称量词的约束。

## 意义

子句集S的不可满足性：对于任意论域中的任意一个解释，S中的子句不能同时取得真值T。

**定理** 设有谓词公式F，其子句集为S，则F不可满足的充要条件是S不可满足。

要证明 $P \rightarrow Q$ 永真，只需证明公式 $F = (P \wedge \neg Q)$ 永假，即S不可满足。

## 示例1

✱ 例1 将下列谓词公式化为子句集。

$$\forall x\{[\neg P(x) \vee \neg Q(x)] \rightarrow \exists y[S(x, y) \wedge Q(x)]\} \wedge \forall x[P(x) \vee B(x)]$$

### ■ (1) 消去蕴涵符号

$$\forall x\{\neg[\neg P(x) \vee \neg Q(x)] \vee \exists y[S(x, y) \wedge Q(x)]\} \wedge \forall x[P(x) \vee B(x)]$$

### ■ (2) 把否定符号移到每个谓词前面

$$\forall x\{[P(x) \wedge Q(x)] \vee \exists y[S(x, y) \wedge Q(x)]\} \wedge \forall x[P(x) \vee B(x)]$$

### ■ (3) 变量标准化

$$\forall x\{[P(x) \wedge Q(x)] \vee \exists y[S(x, y) \wedge Q(x)]\} \wedge \forall w[P(w) \vee B(w)]$$

### ■ (4) 消去存在量词，设y的Skolem函数是 $f(x)$ ，则

$$\forall x\{[P(x) \wedge Q(x)] \vee [S(x, f(x)) \wedge Q(x)]\} \wedge \forall w[P(w) \vee B(w)]$$

## 示例1

✱ 例1 将下列谓词公式化为子句集（续）。

### (5) 化为前束型

$$\forall x \forall w \{ \{ [P(x) \wedge Q(x)] \vee [S(x, f(x)) \wedge Q(x)] \} \wedge [P(w) \vee B(w)] \}$$

### (6) 化为标准形

$$\forall x \forall w \{ \{ [Q(x) \wedge P(x)] \vee [Q(x) \wedge S(x, f(x))] \} \wedge [P(w) \vee B(w)] \}$$

$$\forall x \forall w \{ Q(x) \wedge [P(x) \vee S(x, f(x))] \wedge [P(w) \vee B(w)] \}$$

### (7) 略去全称量词

$$Q(x) \wedge [P(x) \vee S(x, f(x))] \wedge [P(w) \vee B(w)]$$

### (8) 消去合取词，把母式用子句集表示

$$\{Q(x), (P(x), S(x, f(x))), (P(w), B(w))\}$$

### (9) 子句变量标准化 $\{Q(x), (P(y), S(y, f(y))), (P(w), B(w))\}$

## 练习

- ✱ 例2 将下列谓词公式化为不含存在量词的前束型。

$$\exists x \forall y (\forall z (P(z) \wedge \neg Q(x, z)) \rightarrow R(x, y, f(a)))$$

- (1) 消去存在量词

$$\forall y (\forall z (P(z) \wedge \neg Q(b, z)) \rightarrow R(b, y, f(a)))$$

- (2) 消去蕴涵符号

$$\forall y (\neg \forall z (P(z) \wedge \neg Q(b, z)) \vee R(b, y, f(a)))$$

$$\forall y (\exists z (\neg P(z) \vee Q(b, z)) \vee R(b, y, f(a)))$$

- (3) 设 $z$ 的Skolem函数是 $g(y)$ , 则

$$\forall y (\neg P(g(y)) \vee Q(b, g(y)) \vee R(b, y, f(a)))$$

## 谓词逻辑的归结原理和过程

- 在获得子句集后，证明定理演绎过程中，经常要对量化的表达式（**不同子句**）进行匹配操作，因而需要对项作变量置换使表达式一致起来。

### 归结过程：

- ◆ 若 $S$ 中两个子句间有相同互补文字的谓词，但它们的项不同，则必须找出对应的不一致项；
- ◆ 进行变量置换，使它们的对应项一致；
- ◆ 求归结式看能否推导出空子句。

## B. 置换与合一

例 设

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

若选  $L_1 = P(a)$ ,  $L_2 = \neg P(y)$ , 则有:  $L_1 = P(a)$ ,  $\neg L_2 = P(y)$ ,  $\sigma = \{a/y\}$  就是  $L_1$  与  $\neg L_2$  的最一般合一。可得:

$$\begin{aligned} C_{12} &= \neg Q(x) \vee R(x) \vee Q(b) \\ &= (C_1 \sigma - \{L_1 \sigma\}) \cup (C_2 \sigma - \{L_2 \sigma\}) \end{aligned}$$

**定义** 设  $C_1$  与  $C_2$  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  $\sigma$  是  $L_1$  和  $\neg L_2$  的最一般合一, 则称

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

为  $C_1$  和  $C_2$  的二元归结式,  $L_1$  和  $L_2$  称为归结式上的文字。



## B. 置换与合一

- **定义** 若子句C含有可合一的文字，则在进行归结之前应先对这些文字进行合一。记其最一般的合一为  $\sigma$ ，称  $C\sigma$  子句C的因子。

$$C1 = P(x) \vee P(f(a)) \vee Q(x),$$

$$C2 = \neg P(y) \vee R(b)$$

$$\sigma = \{ f(a)/x \}$$

$$C1\sigma = P(f(a)) \vee Q(f(a))$$

$$C12 = Q(f(a)) \vee R(b)$$

## B. 置换与合一

- **定义** 子句 $C_1$ 和 $C_2$ 的归结式是下列二元归结式之一：
  1.  $C_1$ 与 $C_2$ 的二元归结式；
  2.  $C_1$ 与 $C_2$ 的因子 $C_2\sigma_2$ 的二元归结式；
  3.  $C_1$ 的因子 $C_1\sigma_1$ 与 $C_2$ 的二元归结式；
  4.  $C_1$ 的因子 $C_1\sigma_1$ 与 $C_2$ 的因子 $C_2\sigma_2$ 的二元归结式。
- **对于一阶谓词逻辑归结原理也是完备的。**即，若子句集 $S$ 不可满足，则必然存在一个从 $S$ 到空子句的归结演绎；若存在一个从 $S$ 到空子句的归结演绎，则 $S$ 一定是不可满足的。

## B. 置换与合一

### 合一 (Unify) :

在谓词逻辑的归结过程中，寻找项之间合适的变量置换使表达式一致，这个过程称为合一。

- 一个表达式的项可以是常量符号、变量符号或函数式。
- 表达式的例 (instance) 是指在表达式中用置换项置换变量后而得到的一个特定的表达式。
- 用  $\sigma = \{t_1/v_1, t_2/v_2, \dots, t_n/v_n\}$  来表示任一置换。 $t_i/v_i$  是指表达式中的用变量  $t_i$  来替换  $v_i$ ，但不允许用与  $v_i$  有关的项  $t_i$  (但是  $t_i$  中可以包含其它变量) 也不允许变元  $v_i$  循环地出现在另一个  $t_i$  中作置换。为了便于理解，后续记  $\sigma = \{v_1 = t_1, v_2 = t_2, \dots, v_n = t_n\}$ 。
- 用  $\sigma$  对表达式  $E$  作置换后的例简记为  $E\sigma$ 。

## B. 置换与合一

例如：

$\{a/x, f(b)/y, w/z\}$  是代换  
 $\{g(y)/x, f(x)/y\}$  不是代换  
 $\{g(a)/x, f(x)/y\}$  是代换  
 $\{g(x)/x, f(y)/y\}$  不是代换

令  $\theta = \{t_1/x_1, t_2/x_2, \dots, t_n/x_n\}$  为一个代换， $F$  为表达式，则  $F\theta$  表示对  $F$  用  $t_i$  代换  $x_i$  后得到的表达式。 $F\theta$  称为  $F$  的特例。

**规则：** IF father( $x, y$ ) and man( $y$ ) THEN son( $y, x$ )

**事实：** father(李四, 李小四) and man(李小四)

$F = \text{father}(x, y) \wedge \text{man}(y)$

$\theta = \{\text{李四}/x, \text{李小四}/y\}$

$F\theta = \text{father}(\text{李四}, \text{李小四}) \wedge \text{man}(\text{李小四})$

结论： son(李小四, 李四)

## B. 置换与合一

**例** 表达式  $P[x, f(y), B]$ ，对应于不同的变换  $s_i$ ，可得到不同的例：

### 置换

$$s_1 = \{z/x, w/y\},$$

$$s_2 = \{A/y\},$$

$$s_3 = \{g(z)/x, A/y\},$$

$$s_4 = \{C/x, A/y\},$$

### 置换的例

$$P[x, f(y), B] s_1 = P[z, f(w), B]$$

$$P[x, f(y), B] s_2 = P[x, f(A), B]$$

$$P[x, f(y), B] s_3 = P[g(z), f(A), B]$$

$$P[x, f(y), B] s_4 = P[C, f(A), B]$$

第一个例叫做原始文字的初等变式，实际上置换后只是对变量作了换名。  
第四个例称作基例，即置换后项中不再含有变量。

## B. 置换与合一

### 合一 (Unify) :

- 例如,  $P(x, g(y, z))\{x = y, y = f(a)\} \Rightarrow P(y, g(f(a), z))$
- 注意: 置换是同时进行的, 而不是先后进行的。
- 可以对表达式多次置换, 如用 $\theta$ 和 $\sigma$ 依次对 $E$ 进行置换, 记为 $(E\theta)\sigma$ 。其结果等价于先将这两个置换合成 (组合) 为一个置换, 即 $\theta\sigma$ , 再用合成置换对 $E$ 进行置换, 即 $E(\theta\sigma)$ 。

## B. 置换与合一

定义 设

$$\theta = \{t_1/x_1, t_2/x_2, \dots, t_n/x_n\}$$

$$\lambda = \{u_1/y_1, u_2/y_2, \dots, u_m/y_m\}$$

是两个代换，则这两个代换的复合也是一个代换，它是从

$$\{t_1 \lambda / x_1, t_2 \lambda / x_2, \dots, t_n \lambda / x_n, u_1/y_1, u_2/y_2, \dots, u_m/y_m\}$$

中删去如下两种元素：

$$t_i \lambda / x_i \quad \text{当 } t_i \lambda = x_i$$

$$u_i/y_i \quad \text{当 } y_i \in \{x_1, x_2, \dots, x_n\}$$

后剩下的元素所构成的集合，记为  $\theta \circ \lambda$ 。

- 注：  $t_i \lambda$  表示对  $t_i$  运用  $\lambda$  进行代换。
- $\theta \circ \lambda$  就是对一个公式  $F$  先运用  $\theta$  进行代换，然后再运用  $\lambda$  进行代换： $F(\theta \circ \lambda) = (F\theta)\lambda$

## B. 置换与合一

例如，设有代换

$$\theta = \{ f(y)/x, \quad z/y \}$$
$$\lambda = \{ a/x, \quad b/y, \quad y/z \}$$

则

$$\begin{aligned}\theta^\circ\lambda &= \{ f(y)\lambda/x, \quad z\lambda/y, \quad a/x, \quad b/y, \quad y/z \} \\ &= \{ f(b)/x, \quad y/y, \quad a/x, \quad b/y, \quad y/z \} \\ &= \{ f(b)/x, \quad y/z \}\end{aligned}$$

定义 设

$$\theta = \{ t_1/x_1, t_2/x_2, \dots, t_n/x_n \}$$

$$\lambda = \{ u_1/y_1, u_2/y_2, \dots, u_m/y_m \}$$

是两个代换，则这两个代换的复合也是一个代换，它是从

$\{ t_1 \lambda / x_1, t_2 \lambda / x_2, \dots, t_n \lambda / x_n, u_1/y_1, u_2/y_2, \dots, u_m/y_m \}$   
中删去如下两种元素：

$$\begin{array}{ll} t_i \lambda / x_i & \text{当 } t_i \lambda = x_i \\ u_i / y_i & \text{当 } y_i \in \{ x_1, x_2, \dots, x_n \} \end{array}$$

后剩下的元素所构成的集合，记为  $\theta^\circ\lambda$ 。

- 注：  $t_i \lambda$  表示对  $t_i$  运用  $\lambda$  进行代换。
- $\theta^\circ\lambda$  就是对一个公式  $F$  先运用  $\theta$  进行代换，然后再运用  $\lambda$  进行代换： $F(\theta^\circ\lambda) = (F\theta)\lambda$



## B. 置换与合一

- 例  $\theta = \{x = f(y), y = z\}$ ,  $\sigma = \{x = a, y = b, z = y\}$ 
  - 1. Get  $S = \{x = f(b), y = y, x = a, y = b, z = y\}$
  - 2. Delete  $x = a$ ; Delete  $y = b$
  - 3. Delete  $y = y$

$$\theta\sigma = S = \{x = f(b), z = y\}$$

这样的合成法可使  $(E\theta)\sigma = E(\theta\sigma)$ ，即可结合。

但置换是不可交换的，即  $\theta\sigma \neq \sigma\theta$ 。

空置换  $\epsilon = \{\}$  也是一个置换，且  $\theta\epsilon = \theta$ 。

## B. 置换与合一：公式集的合一

**定义** 设有公式集  $F = \{F_1, F_2, \dots, F_n\}$ ，若存在一个代换  $\lambda$  使得

$$F_1 \lambda = F_2 \lambda = \dots = F_n \lambda$$

则称  $\lambda$  为公式集  $F$  的一个合一，且称  $F_1, F_2, \dots, F_n$  是可合一的。

- 例如，设有公式集

$$F = \{P(x, y, f(y)), P(a, g(x), z)\}$$

则下式是它的一个合一：

$$\lambda = \{a/x, g(a)/y, f(g(a))/z\}$$

- 一个公式集的合一一般不唯一。

## B. 置换与合一：公式集的合一

### 合一项 (Unifier) :

- A unifier (合一项) of two formulas  $f$  and  $g$  is a substitution  $\sigma$  that makes  $f$  and  $g$  syntactically identical.
- Note that not all formulas can be unified – substitutions only affect variables.
- e.g.,  $P(f(x), a)$  and  $P(y, f(w))$  cannot be unified, as there is no way of making  $a = f(w)$  with a substitution.

## B. 置换与合一：最一般合一

**定义** 设  $\sigma$  是公式集  $F$  的一个合一，如果对任一个合一  $\theta$  都存在一个置换  $\lambda$ ，使得  $\theta = \sigma \circ \lambda$  则称  $\sigma$  是一个**最一般合一 (MGU)**。

- (1) 置换过程是一个用项代替变元的过程，因此是一个从一般到特殊的过程。
- (2) 最一般合一是唯一的。

## B. 置换与合一：最一般合一

### 最一般的合一项 (Most General Unifier) :

A substitution  $\sigma$  of two formulas  $f$  and  $g$  is a Most General Unifier (MGU) if

- $\sigma$  is a unifier.
- For every other unifier  $\theta$  of  $f$  and  $g$  there must exist a third substitution  $\lambda$  such that  $\theta = \sigma\lambda$ .

This says that every other unifier is “more specialized” than  $\sigma$ .

The MGU of a pair of formulas  $f$  and  $g$  is unique up to renaming.

## B. 置换与合一：最一般合一

### MGU示例：

- $P(f(x), z)$  and  $P(y, a)$
- $\sigma = \{y = f(a), x = a, z = a\}$  is a unifier, but not an MGU
- $\theta = \{y = f(x), z = a\}$  is an MGU
- $\sigma = \theta\lambda$ , where  $\lambda = \{x = a\}$

## B. 置换与合一：求解最一般合一

- **差异集**：两个公式中相同位置处不同符号的集合。

例：  $F = \{P(x, y, z), P(x, f(a), h(b))\}$ ，则  $D1 = \{y, f(a)\}$ ，  $D2 = \{z, h(b)\}$  是差异集。

求取最一般合一的算法：

1. 令  $k=0$ ,  $F_k = F$ ,  $\sigma_k = \varepsilon$ 。  $\varepsilon$  是空代换。
2. 若  $F_k$  只含一个表达式，则算法停止，  $\sigma_k$  就是最一般合一。
3. 找出  $F_k$  的差异集  $D_k$ 。
4. 若  $D_k$  中存在元素  $x_k$  和  $t_k$ ，其中  $x_k$  是变元，  $t_k$  是项，且  $x_k$  不在  $t_k$  中出现，则置：

$$\begin{aligned} F_{k+1} &= F_k \{t_k / x_k\} \\ \sigma_{k+1} &= \sigma_k \circ \{t_k / x_k\} \\ k &= k+1 \end{aligned}$$

然后转(2)。若不存在这样的  $x_k$  和  $t_k$  则算法停止。

5. 算法终止，  $F$  的最一般合一不存在。

## B. 置换与合一：求解最一般合一示例

例如，设  $F = \{P(a, x, f(g(y))), P(z, f(z), f(u))\}$

求其最一般合一。

1. 令  $F_0 = F$ ,  $\sigma_0 = \varepsilon$ 。  $F_0$  中有两个表达式，所以  $\sigma_0$  不是最一般合一。
2. 差异集： $D_0 = \{a, z\}$ 。代换： $\{a/z\}$
3.  $F_1 = F_0 \{a/z\} = \{P(a, x, f(g(y))), P(a, f(a), f(u))\}$ 。  
 $\sigma_1 = \sigma_0 \circ \{a/z\} = \{a/z\}$
4.  $D_1 = \{x, f(a)\}$ 。代换： $\{f(a)/x\}$
5.  $F_2 = F_1 \{f(a)/x\} = \{P(a, f(a), f(g(y))), P(a, f(a), f(u))\}$ 。  
 $\sigma_2 = \sigma_1 \circ \{f(a)/x\} = \{a/z, f(a)/x\}$   
 $D_2 = \{g(y), u\}$ 。代换： $\{g(y)/u\}$
6.  $F_3 = F_2 \{g(y)/u\} = \{P(a, f(a), f(g(y))), \underline{P(a, f(a), f(g(y)))}\}$ 。  
 $\sigma_3 = \sigma_2 \circ \{g(y)/u\} = \{a/z, f(a)/x, g(y)/u\}$



# 谓词逻辑的归结原理和过程

## 归结原理和过程：

From the two clauses  $\{\rho_1\} \cup c_1$  and  $\{\neg\rho_2\} \cup c_2$ , where there exists a MGU  $\sigma$  for  $\rho_1$  and  $\rho_2$ , infer the clause  $(c_1 \cup c_2)\sigma$

**Theorem.**  $S \vdash ()$  iff  $S$  is unsatisfiable

1.  $(P(x), Q(g(x)))$
2.  $(R(a), Q(z), \neg P(a))$
3.  $R[1a, 2c]\{X=a\} (Q(g(a)), R(a), Q(z))$

- “R” means resolution step.
- “1a” means the 1st (a-th) literal in the first clause:  $P(x)$ .
- “2c” means the 3rd (c-th) literal in the second clause:  $\neg P(a)$ .
- 1a and 2c are the “clashing” literals.
- $\{X = a\}$  is the MGU applied.

## 2.4 归结推理

### 示例1

已知：

- (1) 会朗读的人是识字的，
- (2) 海豚都不识字，
- (3) 有些海豚是很机灵的。

证明：有些很机灵的东西不会朗读。

解：把问题用谓词逻辑描述如下，

已知：

- (1)  $\forall x (R(x) \rightarrow L(x))$
- (2)  $\forall x (D(x) \rightarrow \neg L(x))$
- (3)  $\exists x (D(x) \wedge I(x))$

求证：  $\exists x (I(x) \wedge \neg R(x))$

## 2.4 归结推理

### 示例1

- 前提化简，待证结论取反并化成子句形，求得子句集：

$$1. (\neg R(x), L(x)) = \neg R(x) \vee L(x)$$

$$2. (\neg D(y), \neg L(y)) = \neg D(y) \vee \neg L(y)$$

$$3. D(a)$$

$$4. I(a)$$

$$5. (\neg I(z), R(z)) = \neg I(z) \vee R(z)$$

一个可行的证明过程：

$$6. R[4, 5] \{z = a\} R(a)$$

$$7. R[1, 6] \{x = a\} L(a)$$

$$8. R[2, 7] \{y = a\} \neg D(a)$$

$$9. R[3, 8] ()$$

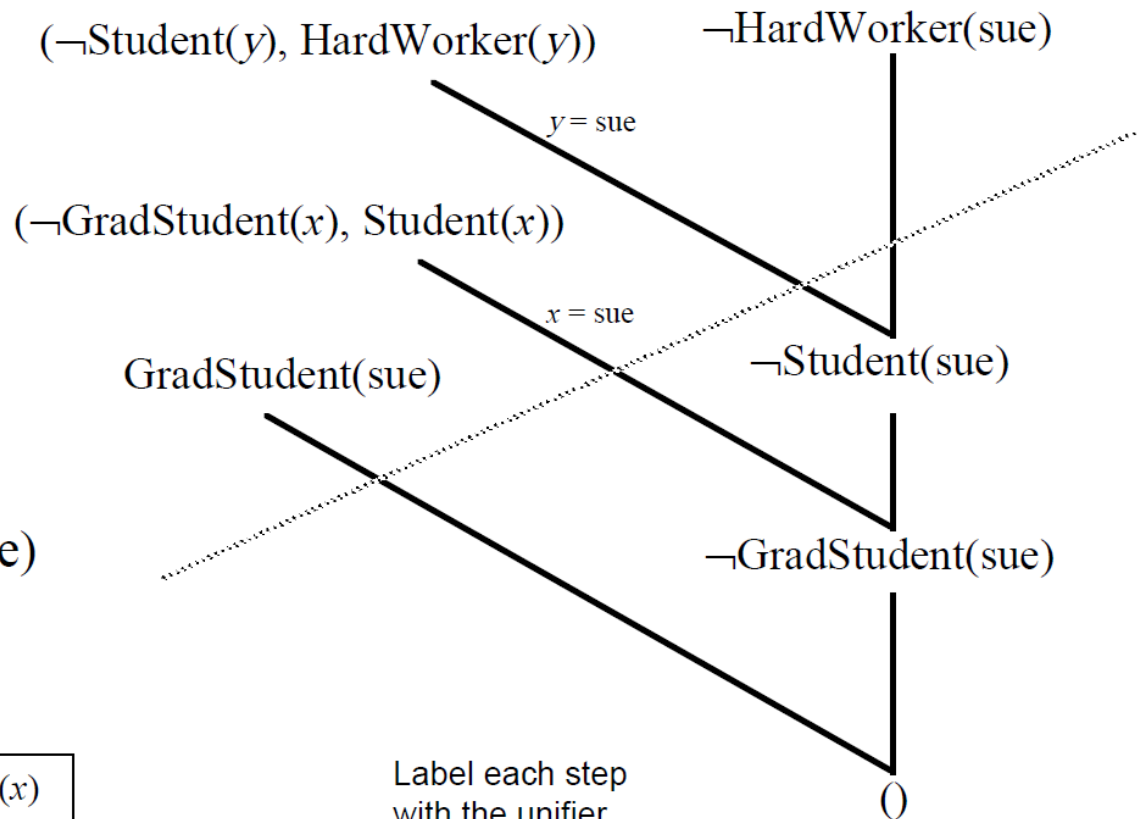
## 2.4 归结推理

### 示例2

?  
KB  $\models$  HardWorker(sue)

KB

$\forall x \text{ GradStudent}(x) \rightarrow \text{Student}(x)$ $\forall x \text{ Student}(x) \rightarrow \text{HardWorker}(x)$ $\text{GradStudent}(\text{sue})$
---



Label each step  
with the unifier

Point to relevant  
literals in clauses

## 2.4 归结推理

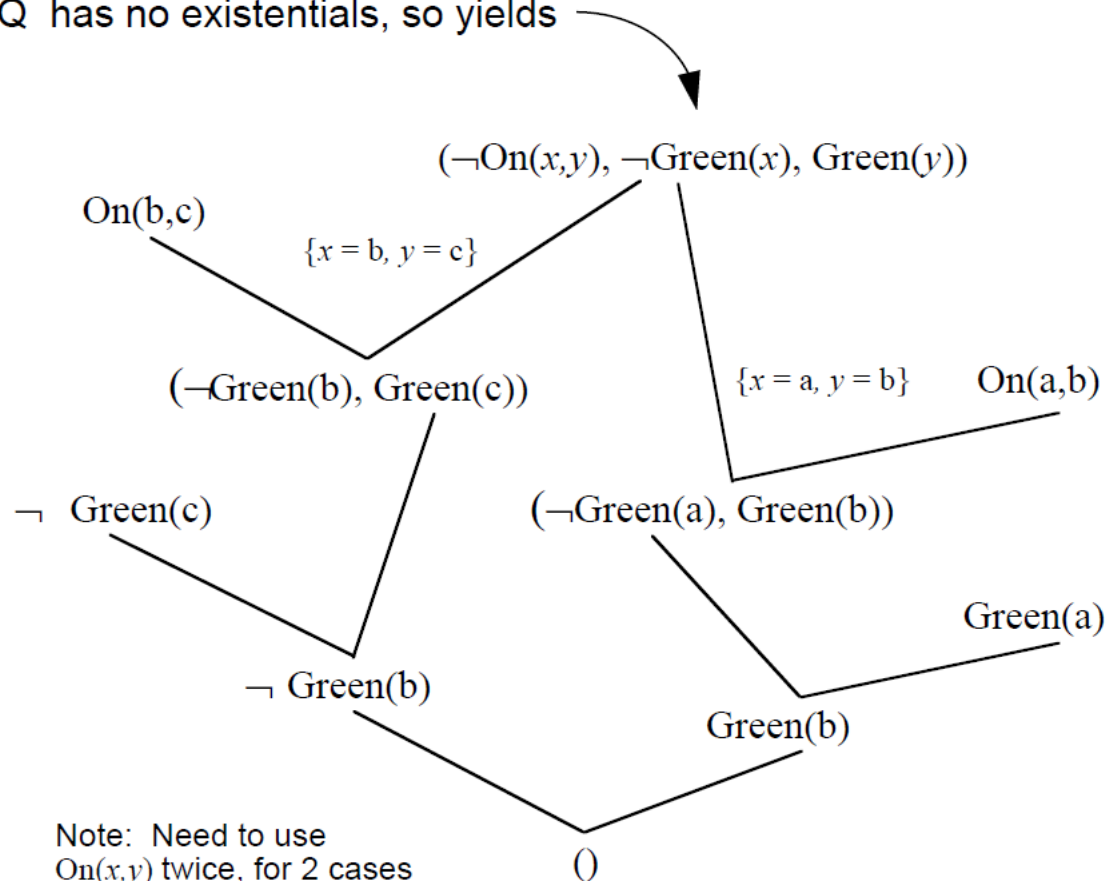
### 示例3

KB = {On(a,b), On(b,c), Green(a),  $\neg$ Green(c)}

already in CNF

Query =  $\exists x \exists y [\text{On}(x,y) \wedge \text{Green}(x) \wedge \neg \text{Green}(y)]$

Note:  $\neg Q$  has no existentials, so yields



## 2.4 归结推理

### 练习

Prove that  $\exists y \forall x P(x, y) \models \forall x \exists y P(x, y)$

- $\exists y \forall x P(x, y) \Rightarrow 1. P(x, a)$
- $\neg \forall x \exists y P(x, y) \Leftrightarrow \exists x \forall y \neg P(x, y) \Rightarrow 2. \neg P(b, y)$
- $R[1,2]\{x = b, y = a\}()$

Exercises: Prove

- $\forall x P(x) \vee \forall x Q(x) \models \forall x (P(x) \vee Q(x))$
- $\exists x (P(x) \wedge Q(x)) \models \exists x P(x) \wedge \exists x Q(x)$

## 2.4 归结推理

### 不可判定问题

We use 1 for  $\text{succ}(0)$ , 2 for  $\text{succ}(\text{succ}(0))$ , ...

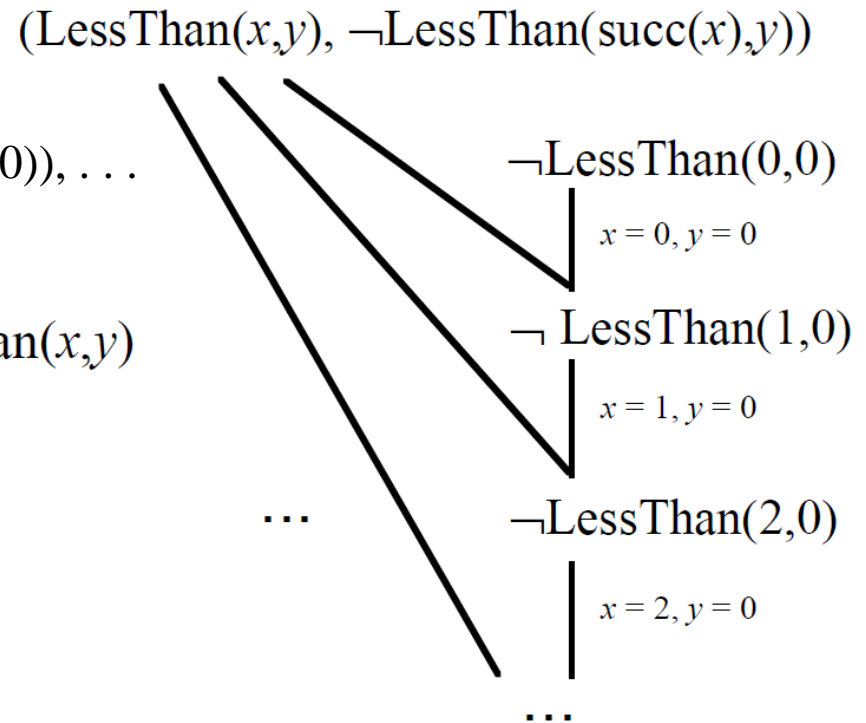
KB:

$\text{LessThan}(\text{succ}(x), y) \rightarrow \text{LessThan}(x, y)$

Query:

$\text{LessThan}(0, 0)$

Should fail since  $\text{KB} \not\models Q$



Infinite branch of resolvents

对于谓词逻辑，若子句集不可满足，则必存在一个从该子句集到空子句的推导；若从子句集存在一个到空子句的推导，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。

## 2.4 归结推理

### 不可判定问题

- 可判定的问题：如果存在一个算法或过程，该算法用于求解该类问题时，可在有限步内停止，并给出正确的解答。
- 如果不存在这样的算法或过程则称这类问题是**不可判定的**。例如，There can be no procedure to decide if a set of clauses is satisfiable.

**Theorem.**  $S \vdash ()$  iff  $S$  is unsatisfiable

However, there is no procedure to check if  $S \vdash ()$ , because

When  $S$  is satisfiable, the search for  $()$  may not terminate



## 2.4 归结推理

### 应用归结原理求解问题

- 应用归结原理求解问题的步骤：

- (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  $P$  用谓词公式表示，并否定  $P$ ，再与  $answer$  构成析取式  $(\neg P \vee answer)$ ；
- (3) 把  $(\neg P \vee answer)$  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。

## 用归结原理求解问题的例子

**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同一个问题：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中至少有一个是说谎者”。求谁是老实人，谁是说谎者？

**解：** 设用 $T(x)$ 表示 $x$ 说真话。

$$\begin{aligned} & T(C) \vee T(A) \vee T(B) \\ & \neg T(C) \vee \neg T(A) \vee \neg T(B) \\ & T(A) \rightarrow \neg T(B) \wedge \neg T(C) \\ & \neg T(A) \rightarrow T(B) \vee T(C) \\ & T(B) \rightarrow \neg T(A) \wedge \neg T(C) \\ & \neg T(B) \rightarrow T(A) \vee T(C) \\ & T(C) \rightarrow \neg T(A) \vee \neg T(B) \\ & \neg T(C) \rightarrow T(A) \wedge T(B) \end{aligned}$$

## 用归结原理求解问题的例子

把上述公式化成子句集，得到S：

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

下面先求谁是老实人。把  $\neg T(x) \vee \text{Answer}(x)$  并入S得到  $S_1$ 。即多一个子句：

- (8)  $\neg T(x) \vee \text{Answer}(x)$

应用归结原理对  $S_1$  进行归结：

- (9)  $\neg T(A) \vee T(C)$  (1) 和 (7) 归结
- (10)  $T(C)$  (6) 和 (9) 归结
- (11)  $\text{Answer}(C)$  (8) 和 (10) 归结

所以C是老实人，即C从不说假话。

## 用归结原理求解问题的例子

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

下面证明A不是老实人，即证明 $\neg T(A)$ 。

对 $\neg T(A)$ 进行否定，并入S中，得到子句集S2，即S2比S多如下子句：

- (8)  $\neg(\neg T(A))$ ，即 $T(A)$

应用归结原理对S2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

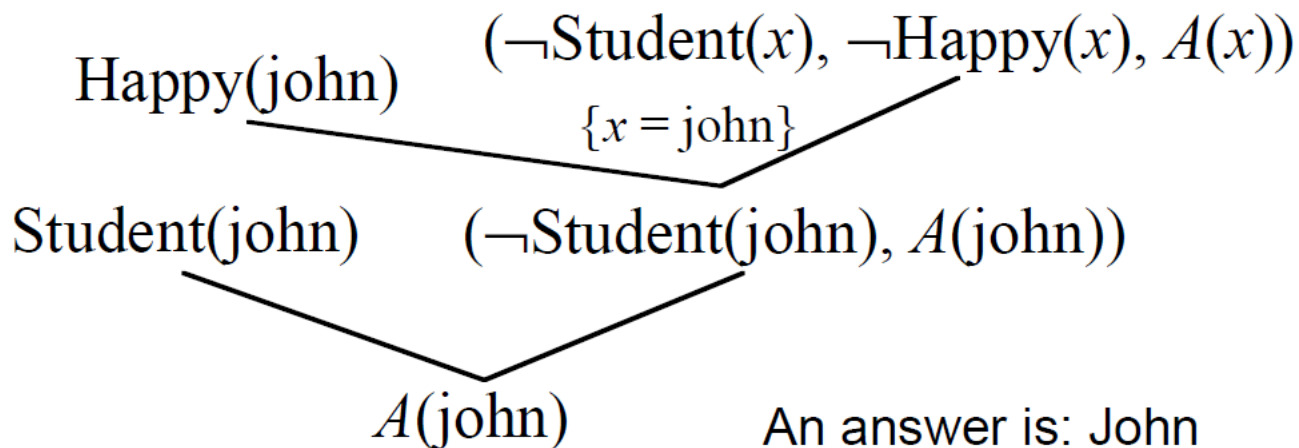
所以A不是老实人。同样可以证明B也不是老实人。

## 2.4 归结推理

### 示例1

KB: Student(john)  
 Student(jane)  
 Happy(john)

Q:  $\exists x[\text{Student}(x) \wedge \text{Happy}(x)]$



## 2.4 归结推理

### 示例2

KB:

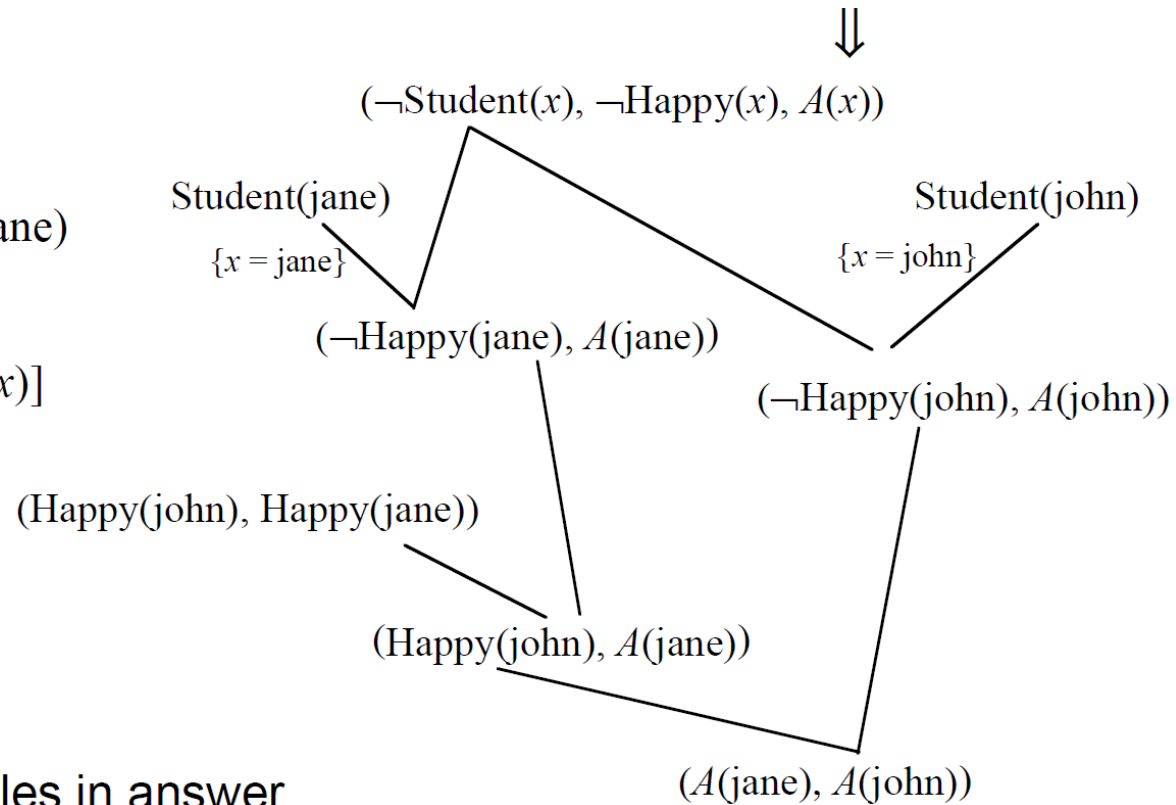
Student(john)

Student(jane)

Happy(john)  $\vee$  Happy(jane)

Query:

$\exists x[\text{Student}(x) \wedge \text{Happy}(x)]$



Note: can have variables in answer

An answer is: either Jane or John

## 2.4 归结推理

### 练习

- Whoever can read is literate.
- Dolphins are not literate.
- Flipper is an intelligent dolphin.
- Who is intelligent but cannot read.

Use predicates:  $R(x)$ ,  $L(x)$ ,  $D(x)$ ,  $I(x)$

## 2.4 归结推理

### 归结反演

- ✿ 应用归结原理证明定理的过程称为归结反演。
- ✿ 用归结反演证明的步骤是：
  - (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，若出现了空子句，则停止归结，此时就证明了 $Q$ 为真。



## 用归结反演证明结论的例子

例 已知

$$F: (\forall x)((\exists y)(A(x, y) \wedge B(y)) \rightarrow (\exists y)(C(y) \wedge D(x, y)))$$

$$G: \neg(\exists x)C(x) \rightarrow (\forall x)(\forall y)(A(x, y) \rightarrow \neg B(y))$$

求证：G是F的逻辑结论。

证明：首先把F和 $\neg G$ 化为子句集：

$$F = \{\neg A(x, y) \vee \neg B(y) \vee C(f(x)), \neg A(x, y) \vee \neg B(y) \vee D(x, f(x))\}$$

$$\neg G = \{\neg C(z), A(a, b), B(b)\}$$

然后进行归结：

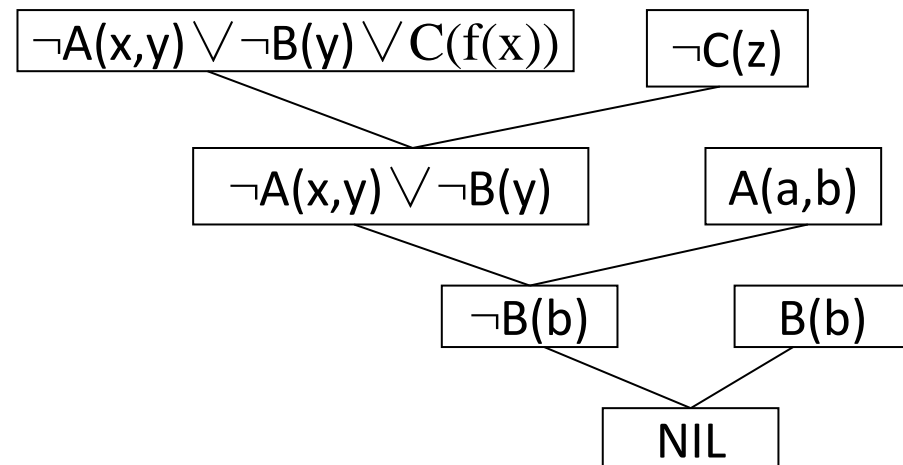
(6)  $\neg A(x, y) \vee \neg B(y)$       由(1)与(3)归结， $\{f(x)/z\}$

(7)  $\neg B(b)$       由(4)与(6)归结， $\{a/x, b/y\}$

(8) NIL      由(5)与(7)归结

所以G是F的逻辑结论。

上述归结过程如右图归结树所示。



## 2.4 归结推理

### 归结策略

归结的一般过程（宽度优先策略）

设有子句集 $S=\{C_1, C_2, C_3, C_4\}$ ，则对此子句集归结的一般过程是：

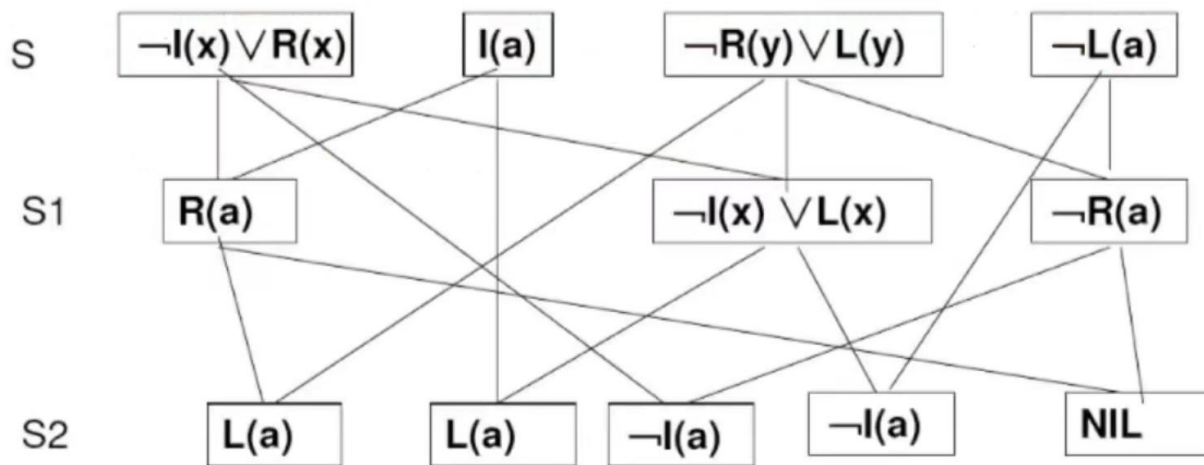
1.  $S$ 内任意子句两两逐一进行归结，得到一组归结式，称为第一级归结式，记为 $S_1$ 。
2. 把 $S$ 与 $S_1$ 内的任意子句两两逐一进行归结，得到一组归结式，称为第二级归结式，记为 $S_2$ 。
3.  $S$ 和 $S_1$ 内的子句与 $S_2$ 内的任意子句两两逐一进行归结，得到一组归结式，称为第三级归结式，记为 $S_3$ 。
4. 如此继续，直到出现了空子句或者不能再继续归结为止。

# 宽度优先策略

设有如下子句集:

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

用宽度优先策略证明S为不可满足。



从这个例子可以看出，宽度优先策略归结出了许多无用的子句，既浪费时间，又浪费空间。但是，当问题有解时，这种策略保证能找到最短归结路径。

因此，它是一种完备的归结策略。

宽度优先对大问题的归结容易产生组合爆炸，但对小问题却仍是一种比较好的归结策略。

# 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。包含纯文字的子句可以删除。

- 重言式删除法

如果一个子句中同时包含互补文字对，则该子句称为重言式。重言式是永远为真的子句，可以删除。

- 包孕删除法

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

## 支持集策略

对参加归结的子句提出如下限制：每一次归结时，**亲本子句中至少有一个是由目标公式的否定所得到的子句**，或者是它的后裔。可以证明，支持集策略是**完备的**。

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

(2)  $I(a)$

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

(4)  $L(a)$

$S_1$ : (5)  $R(a)$

(1) 与 (2) 归结

(6)  $\neg I(x) \vee \neg L(x)$

(1) 与 (3) 归结

$S_2$ : (7)  $\neg L(a)$

(2) 与 (6) 归结

(8)  $\neg L(a)$

(3) 与 (5) 归结

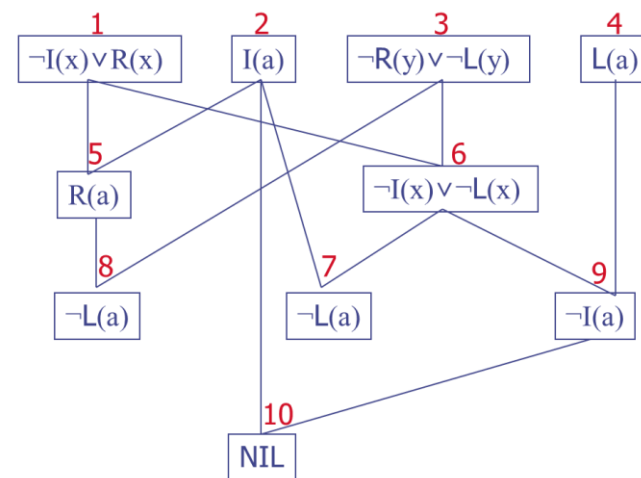
(9)  $\neg I(a)$

(4) 与 (6) 归结

$S_3$ : (10) NIL

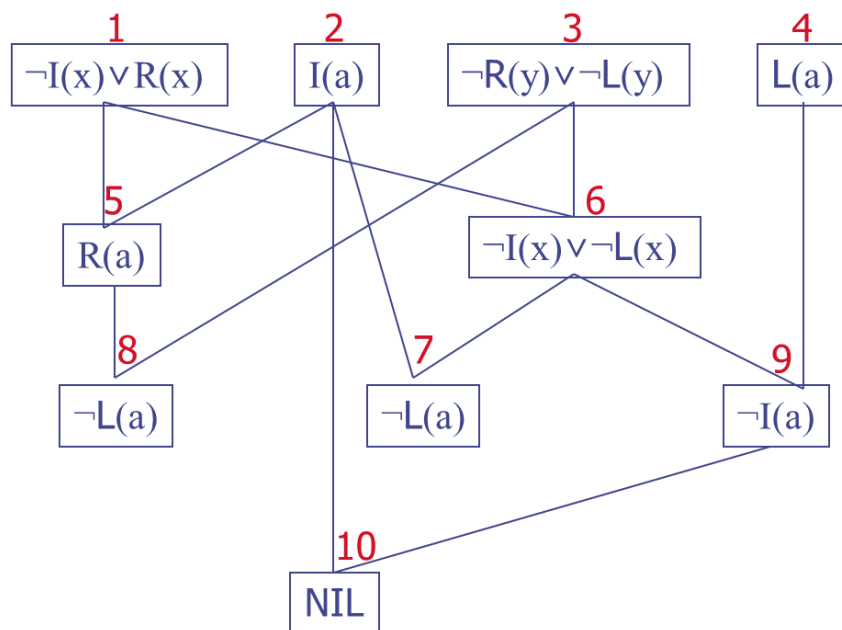
(2) 与 (9) 归结

### 支持集策略示例



# 支持集策略

## 支持集策略示例



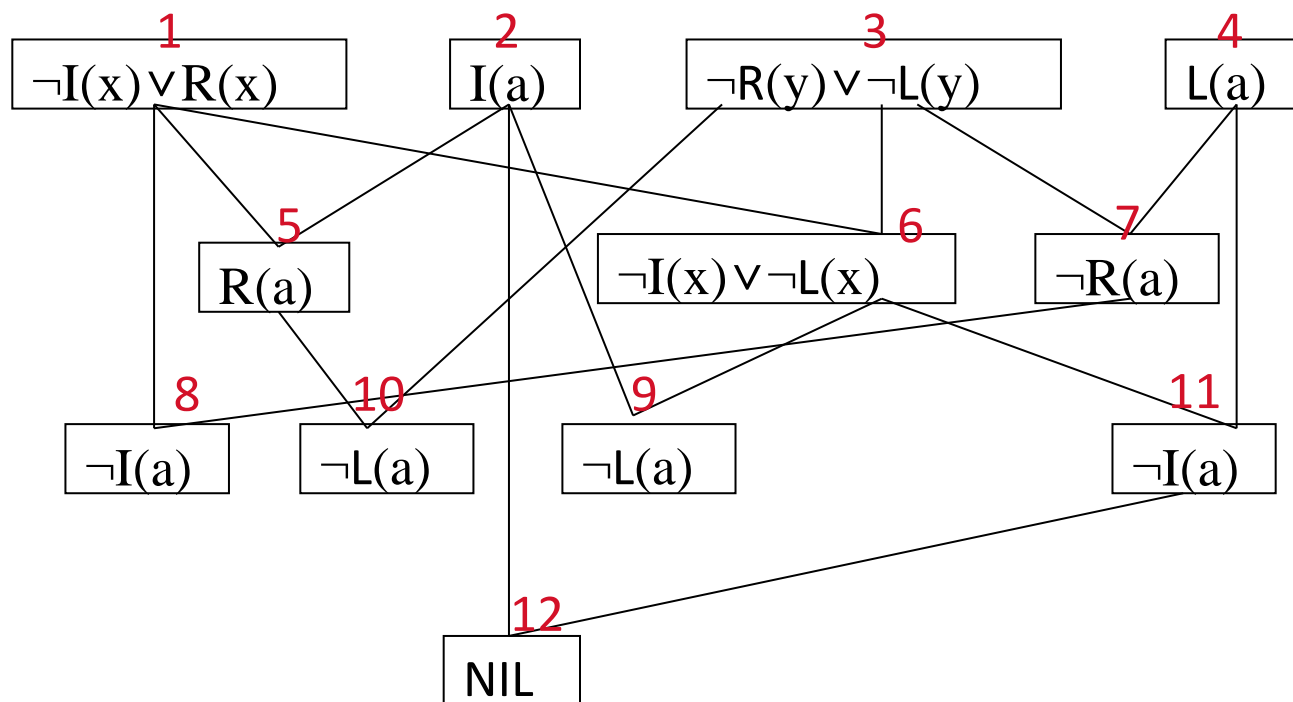
从上述归结过程可以看出，各级归结式数目要比宽度优先策略生成的少，但在第二级还没有空子句。

就是说这种策略限制了子句集元素的剧增，但会增加空子句所在的深度。

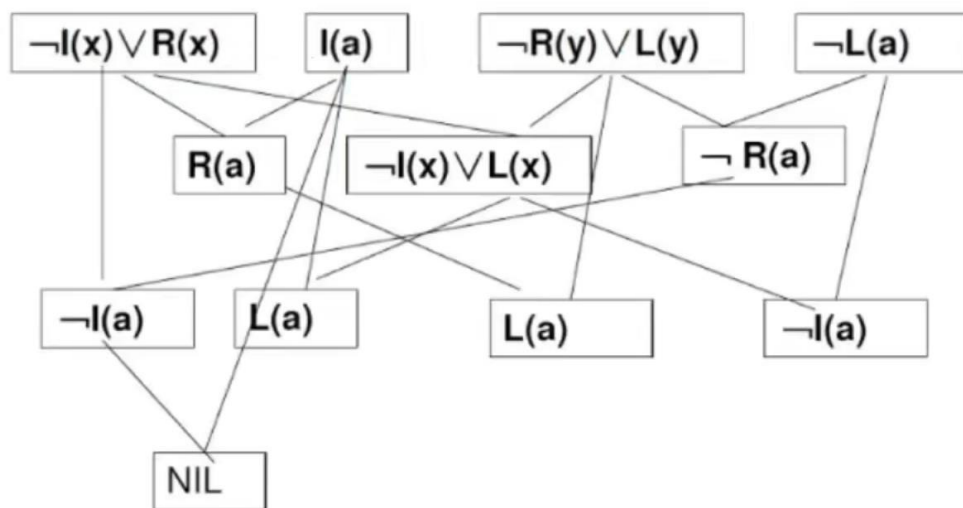
此外支持集策略具有逆向推理的含义，由于进行归结的亲本子句中至少有一个与目标子句有关，因此推理过程可以看作是沿目标、子目标的方向前进的。

## 线性输入策略

- 限制：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。
- 线性输入策略可限制生成归结式的数量，具有简单、高效的优点。但是它是**不完备的**。



## 线性输入策略



线性输入策略可限制生成归结式的数目，具有简单和高效的优点。但是，这种策略也是一种不完备的策略。

例如

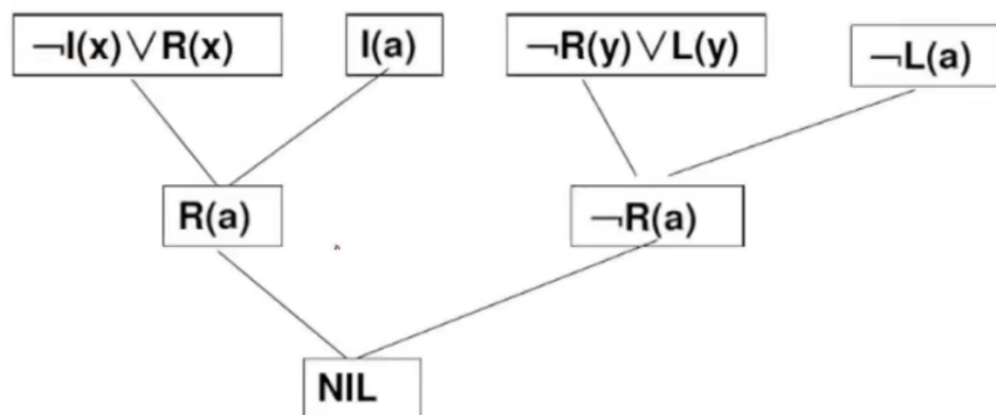
$S = \{ Q(u) \vee P(a), \neg Q(w) \vee P(w), \neg Q(x) \vee \neg P(x), Q(y) \vee \neg P(y) \}$   
从S出发很容易找到一棵归结反演树，但却不存在线性输入策略的归结反演树。



## 单文字策略

- 如果一个子句只包含一个文字，则称它为单文字子句。
- 限制：参加归结的两个子句中必须至少有一个是单文字子句。
- 用单文字子句策略归结时，归结式比亲本子句含有较少的文字，这有利于朝着空子句的方向前进，因此它有较高的归结效率。但是，这种归结策略是不完备的。当初法子句集中不包含单文字子句时，归结就无法进行。

## 单文字策略



采用单文字子句策略，归结式包含的文字数将少于其亲本子句中的文字数，这将有利于向空子句的方向发展，因此会有较高的归结效率。

但这种策略是不完备的，即当子句集为不可满足时，用这种策略不一定能归结出空子句。

## 祖先过滤策略

该策略与线性策略比较相似，但放宽了限制。当对两个子句 $C_1$ 和 $C_2$ 进行归结时，只要它们满足下述任一个条件就可以归结。

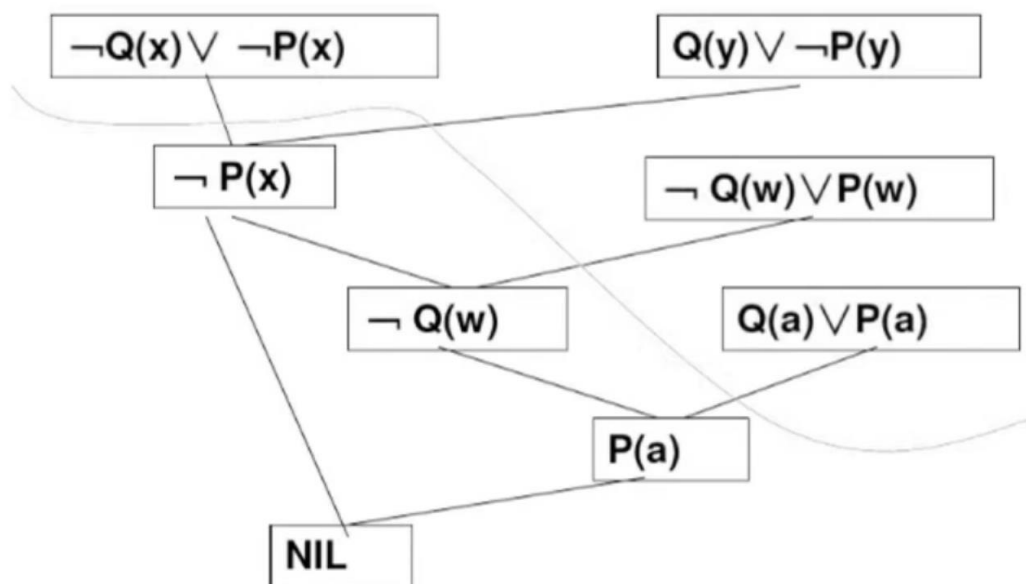
1.  $C_1$ 和 $C_2$ 中至少有一个是初始子句集中的子句。
  2.  $C_1$ 和 $C_2$ 中一个是另外一个的祖先子句。
- 祖先过滤策略是完备的。

## 祖先过滤策略

例 设有如下子句集：

$S = \{\neg Q(x) \vee \neg P(x), Q(y) \vee \neg P(y), \neg Q(w) \vee \neg P(w), Q(a) \vee \neg P(a)\}$

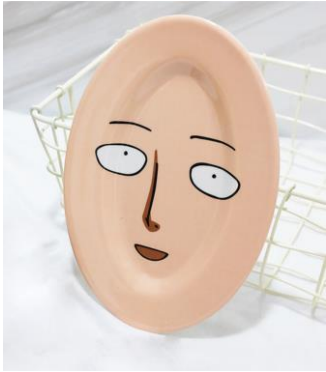
用祖先过滤策略证明 $S$ 为不可满足。



可以证明祖先过滤策略也是完备的。

在选择归结反演策略时,主要应考虑其完备性和效率问题。

## 练习



- If Superman were able and willing to prevent evil, he would do so.
  - If Superman were unable to prevent evil, he would be impotent;
  - if he were unwilling to prevent evil, he would be malevolent.
  - Superman does not prevent evil.
  - If Superman exists, he is neither impotent nor malevolent.
- ◆ Therefore, Superman does not exist

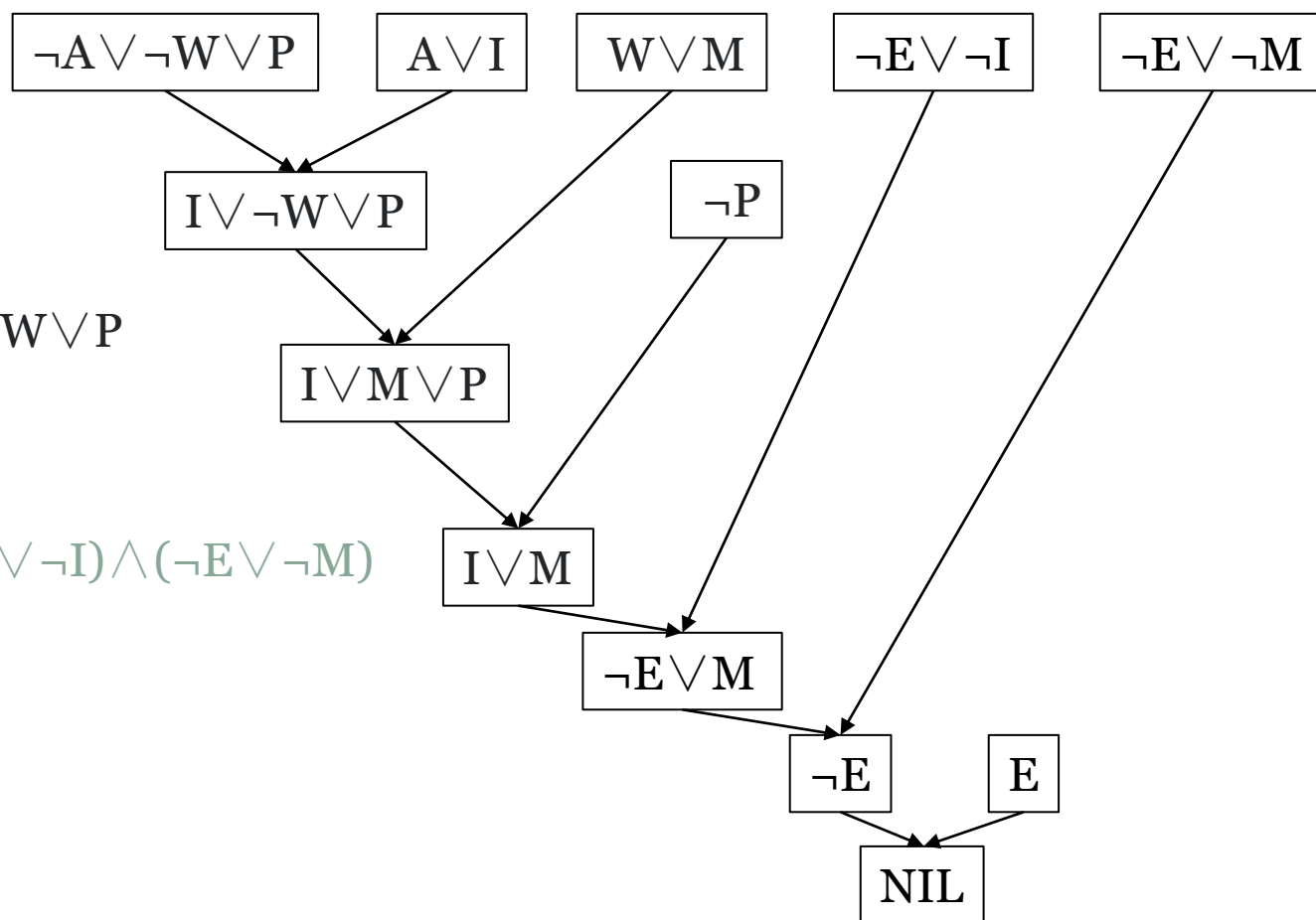
W: Superman is willing to prevent evil.    A: Superman is able to prevent evil.  
P: Superman prevents evil.                      I: Superman is impotent.  
M: Superman is malevolent.                      E: Superman exists

- Superman were able and willing to prevent evil, he would do so  $(A \wedge W) \rightarrow P$
- If Superman were unable to prevent evil he would be impotent  $\neg A \rightarrow I$
- if he were unwilling to prevent evil, he would be malevolent  $\neg W \rightarrow M$
- Superman does not prevent evil  $\neg P$
- If Superman exists, he is neither impotent nor malevolent  $E \rightarrow (\neg I \wedge \neg M)$
- Superman does not exist  $\neg E$

# 练习

- Superman were able and willing to prevent evil, he would do so  $(A \wedge W) \rightarrow P$
- If Superman were unable to prevent evil he would be impotent  $\neg A \rightarrow I$
- if he were unwilling to prevent evil, he would be malevolent  $\neg W \rightarrow M$
- Superman does not prevent evil  $\neg P$
- If Superman exists, he is neither impotent nor malevolent  $E \rightarrow (\neg I \wedge \neg M)$
- Superman does not exist  $\neg E$

- 1)  $\neg(A \wedge W) \vee P \equiv \neg A \vee \neg W \vee P$
- 2)  $A \vee I$
- 3)  $W \vee M$
- 4)  $\neg P$
- 5)  $\neg E \vee (\neg I \wedge \neg M) \equiv (\neg E \vee \neg I) \wedge (\neg E \vee \neg M)$
- 5#)  $\neg E \vee \neg I$
- 5##)  $\neg E \vee \neg M$
- 6)  $E$



# Thanks

Thanks the support from Prof. YongMei Liu (SYSU) ,Prof. Jiahai Wang(SYSU), Prof. Hector Levesque (University of Toronto), and Prof. Sheila McIlraith (University of Toronto), and the authors of the courseware, for the refer to their slides.