第11章 谓词逻辑的归结原理及其应用

归结推理来源于这样的一种想法：我们能否找到一种标准方法来证明谓词逻辑中的定理。

- 20世纪60年代初，美籍华人王浩首先总结了十条推理规则
- 1965年Robinson 提出了归结原理(resolution)，从而使谓词演算的定理机械证明有了统一的标准方法。

这种方法是反证法，即通过证明该命题的否定与一个已知命题或一个推导出来的命题相矛盾。

11.1 命题演算的归结方法

要求证明在A∧B∧C成立的条件下有*D*成立，也即**A∧B∧C→*D*是重言式**。

由于，A∧B∧C→*D*是重言式等价于～(A∧B∧C→*D*) 是永假式，也即

～(～(A∧B∧C)∨*D*)= A∧B∧C∧～D

是永假式。

归结推理方法就是从A∧B∧C∧～D出发，使用推理规则来找出矛盾，达到证明：

A∧B∧C→D

这种方法就称为**归结推理方法**。

11.1.1 基本概念

与归结相关的定义：

文字：指任一原子公式或原子公式的非。

例如：～*P*，Q，R 都是文字。

子句：文字的析取范式。

例如：～*P*∨Q∨R 就是一个子句。

由子句构成的集合称为**子句集**。

不包含任何文字的子句称为**空子句**，表示为**NIL**。

由于空子句不含有文字，它不能被任何解释满足，所以**空子句是永假的、不可满足的**。

在谓词逻辑中，任何一个谓词公式都可以通过应用等价关系及推理规则化成相应的子句集，从而能够比较容易地判定谓词公式的不可满足性。

11.1.1 基本概念

与归结相关的定义：

亲本子句(parent clauses)：是指这样的两个子句，一个子句中含文字L，另一个子句含文字～L，L与～L又称互补文字。亲本子句又称为母子句。

例如：

C1=～*P*∨Q∨R，

C2=*P*∨S

C1，C2就可作为亲本子句，～*P*与*P* 就是互补文字。

11.1.1 基本概念

归结式(resolvent)：从亲本子句中去掉一对互补文字后，剩余的两个部分的析取范式。

例如：$P \vee Q$ 与 $\sim Q \vee R$ 归结后，归结式为 $P \vee R$。

11.1.2 命题演算的归结方法

设给定的已知条件为公式集F，求证的命题为G，进行归结的步骤为：

（1）将公式集F中的所有命题改写成子句。

（2）将命题～G改写成子句或多个子句。

（3）将（1）、（2）所得到的子句合并成子句集S，并在出现一个矛盾或无任何进展之前执行：

（3.1） 从子句集中选一对亲本子句；

（3.2） 将亲本子句对归结成一个归结式；

（3.3） 若归结式为非空子句，将其加入子句集S；

若归结式为空子句，则归结结束。

11.1.2 命题演算的归结方法

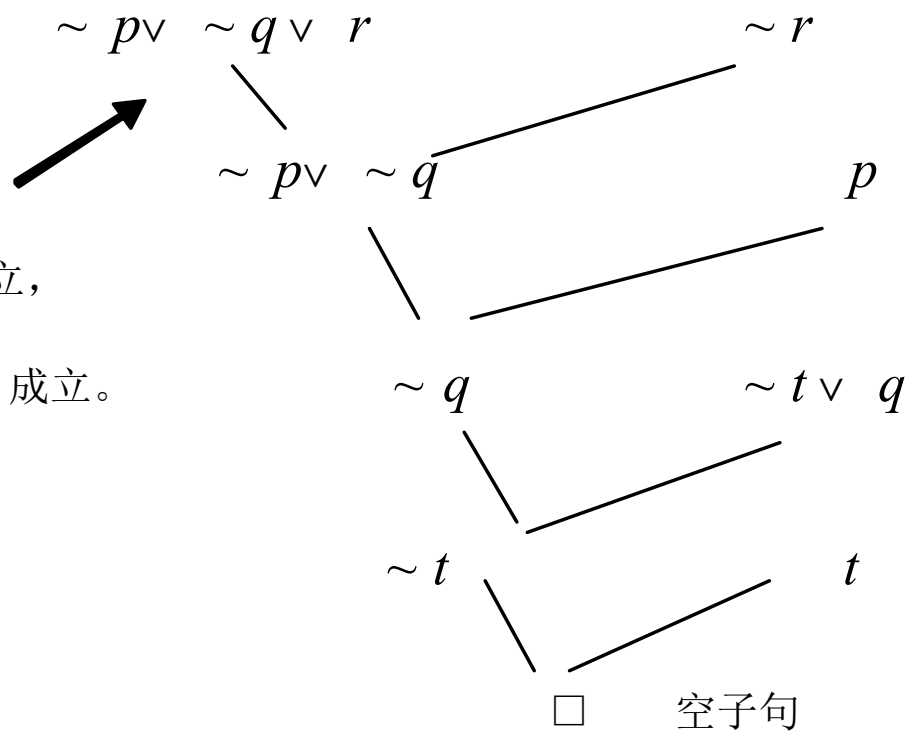
例11.1 已知，命题公式集 $s=\{p, p\wedge q\rightarrow r, u\vee t\rightarrow q, t\}$，求证 r 。

首先，将每个命题化为子句形式。其过程为，将s中的公式化为子句集S’；其对应关系为：

p	p
$p\wedge q\rightarrow r$	$\sim p\vee\sim q\vee r$
$u\vee t\rightarrow q$	$\sim u\vee q$ $\sim t\vee q$
t	t
目标 r	目标的否定：$\sim r$

11.1.2 命题演算的归结方法

用反证法求证目标，即求证 $\sim r$ 与子句集S产生矛盾，其过程如下图所示



这一步归结直观上看，

是 r 与 $\sim r$ 不能同时成立，

所以只能是 $\sim p \vee \sim q$ 成立。

p

$\sim p \vee \sim q \vee r$

$\sim u \vee q$

$\sim t \vee q$

t

目标的否定： $\sim r$

11.2 谓词演算的归结

一、 谓词演算的基本问题

由于谓词含有变量，谓词演算逻辑归结需要解决如下3个问题：

(1) 将任一表达式变成标准子句形式。

(2) 如何确定哪两个子句作为亲本子句。合一算法来解决

(3) 如何挑选亲本子句才更有效。归结的控制策略来解决

11.2 谓词演算的归结

二、 将公式化成标准子句形式的步骤

1 用 $\neg A \vee B$ 取代 $A \to B$，消去“→”符号。

2 降低 $\neg$ 号的辖域，直到原子公式之前或消去 $\neg$ 号。

例如，可用 $\neg A \vee \neg B$ 代替 $\neg(A \wedge B)$；

$\neg A \wedge \neg B$ 代替 $\neg(A \vee B)$；

A 代替 $\neg\neg A$；

$(\exists x)(\neg A(x))$ 代替 $\neg \forall x A(x)$；

$(\forall x)(\neg A(x))$ 代替 $\neg(\exists x) A(x)$；

11.2 谓词演算的归结

二、 将公式化成标准子句形式的步骤

3 变量标准化。重新命名哑变量(Dummy variable)，以保证每个量词有自己唯一的变量名。

例如，对 $(\forall x)P(x)\vee(\exists x)Q(x)$ 进行标准化变量时，可将其改为：

$(\forall x)P(x)\vee(\exists y)Q(y)$

4 将公式变为前束范式(Prefix)。即将所有量词移到公式的前部，后面的一部分变成无量词的公式，即母式(Matrix)

将上述第3步的 $(\forall x)P(x)\vee(\exists y)Q(y)$ 改为 $\forall x\exists y(P(x)\vee Q(y))$

11.2 谓词演算的归结

二、 将公式化成标准子句形式的步骤

5 消去存在量词。

a. 存在量词不出现在全称量词的辖域内。

$(\exists x)P(x,y)$ 可以化为$P(A，y)$ 其中， A称为 Skolem常量。

b. 存在量词出现在一个或者多个全称量词的辖域内。用Skolem函数代替存在量词所量化变量的每个出现。Skolem函数的变量是存在量词之前的所有全称量词中的变量。

$$(\forall x)(\forall y)(\exists z)(P(x,y)\vee\ Q(y,z)\vee\ W(z))$$

可以化为：

$$\forall x\forall y(P(x,y)\vee\ Q(y,f(x,y))\vee\ W(f(x,y))$$

其中f(x, y)称为Skolem函数，它是由于出现在辖域之内，所以去掉量词时要用f(x, y)代替z的出现。

11.2 谓词演算的归结

二、 将公式化成标准子句形式的步骤

6 消去全称量词，因为全称量词的次序无关紧要，只要简单消去就行了，这样公式变成无量词公式。

7 重复利用分配律，变公式为析取式的合取式。例如用(A∨B) ∧(A∨C) 代替A∨(B ∧ C) 。

8 消去“∧”连词，使公式成为若干子句。例如(A∨B) ∧(A∨C)就成为:

A∨B ，A∨C 两个子句。

9 将变量换名，使一个变量符不会出现在二个和二个以上的子句中。该步骤称为变量分离标准化。

11.2 谓词演算的归结

二、 将公式化成标准子句形式的步骤

例11.3 考虑如下命题所组成的集合。

1. 马科斯是人。

Man(Marcus)

2. 马科斯是庞贝人。

Pompeian(Marcus)

3. 所有庞贝人都是罗马人。

$\forall x$ (Pompeian(x)→Roman(x))

4. 恺撒是一位统治者。

Ruler(Caesar)

5. 所有罗马人或忠于或仇恨恺撒。

$\forall x$(Roman(x)→Loyalto(x, Caesar)∨Hate(x, Caesar))

6. 每个人都忠于某个人。

$\forall x \exists y$Loyalto(x, y)

7. 人们只想暗杀他们不忠于的统治者。

$\forall x \forall y$(man(x)∧Ruler(y)∧Tryassassinate(x, y) →～Loyalto(x, y))

8. 马科斯试图谋杀恺撒。

Tryassassinate(Marcus, Caesar)

11.2 谓词演算的归结

二、 将公式化成标准子句形式的步骤

例11.3 考虑如下命题所组成的集合。

1. 马科斯是人。

   Man(Marcus)

2. 马科斯是庞贝人。

   Pompeian(Marcus)

3. 所有庞贝人都是罗马人。

$\forall x$ (Pompeian(x)→Roman(x))　　～Pompeian(x1)∨Roman(x1)

4. 恺撒是一位统治者。

   Ruler(Caesar)

11.2 谓词演算的归结

二、 将公式化成标准子句形式的步骤

例11.3 考虑如下命题所组成的集合。

5. 所有罗马人或忠于或仇恨恺撒。 ～Roman(x2)∨Loyalto(x2，Caesar) ∨Hate(x2，Caesar)

$\forall x$ (Roman(x)→Loyalto(x, Caesar)∨Hate(x, Caesar))

6. 每个人都忠于某个人。

$\forall x \exists y$ Loyalto(x, y)　　Loyalto(x3，f(x3))

7. 人们只想暗杀他们不忠于的统治者。

～Man(x4)∨～ Ruler(y1) ∨ ～ Tryassassinate(x4,y1) ∨ ～Loyalto(x4，y1)

$\forall x \forall y$ (man(x)∧Ruler(y)∧Tryassassinate(x, y) →～Loyalto(x, y))

8. 马科斯试图谋杀恺撒。

Tryassassinate(Marcus, Caesar)

11.2 谓词演算的归结

将上述逻辑公式转换成如下的子句形式：

1. Man(Marcus)
2. Pompeian(Marcus)
3. ～Pompeian(x1)∨Roman(x1)
4. Ruler(Caesar)
5. ～Roman(x2)∨Loyalto(x2，Caesar) ∨Hate(x2，Caesar)
6. Loyalto(x3，f(x3))
7. ～Man(x4)∨～ Ruler(y1) ∨ ～ Tryassassinate(x4,y1) ∨ ～Loyalto(x4，y1)
8. Tryassassinate(Marcus, Caesar)

11.2 谓词演算的归结

三　合一算法

- 决定哪二个子句为亲本子句。

  例如，L(f(x))∨L(A)与~L(B)是否能够成为母子句呢？

- 在谓词逻辑中，一个表达式的项是常量符号、变量符号或函数式。

- 表达式的例示(instance)是指在表达式中用项来置换变量而得到特定的表达式，用来置换的项称为置换项。

- 在归结过程中，寻找项之间合适的变量置换使表达式一致的过程，称为合一过程，简称合一(Unify)。

11.2 谓词演算的归结

三　合一算法

定义11.1 若存在一个代换s，使得二个文字L_1与L_2进行代换后，有$L_{1s}=L_{2s}$，即 L_1 与~L_2为互补文字，则L_1与L_2为可合一的。这个代换s称为**合一元(unifier)**。

代换的使用：

(1）只能用项(常量，变量或函数符号) t 去代换变量 x在公式中的一切出现，代换记为s= t/x，对一个公式F作s代换记为F_s。

显然，f(x)、A、B之间不存在代换。

(2）任一被代换的变量不能出现在用作代换的表达式中。{g(x)/x}不是代换。

11.2 谓词演算的归结

三　合一算法

(3) 代换并非唯一。

例11.4 对于F=$P(x, f(y), B)$，存在四种代换：

$s_1=\{z/x, w/y\}$，　则$F_{s_1}=P(z, f(w), B)$ (较一般的代换)

$s_2=\{A/y\}$，　　则$F_{s_2}=P(x, f(A), B)$

$s_3=\{g(z)/x, A/y\}$，则$F_{s_3}=P(g(z), f(A), B)$

$s_4=\{C/x, A/y\}$，　则$F_{s_4}=P(C, f(A), B)$ (限制最严的代换)

S_4称为基例示，因为作替换后不再含有变量。

11.2 谓词演算的归结

三　合一算法

(4) 复合置换

$$Es_1s_2=(Es_1)s_2$$

通常要求用尽可能一般的代换，其置换项的限制最少，所产生的例示更一般化，因而有利于产生新的置换。

11.2 谓词演算的归结

三　合一算法

定义11.2 表达式集合{E1, ... , Er}的合一元σ称为是最一般合一元(most general unifier 简写为mgu)，当且仅当对集合的每一个合一θ都存在代换λ使得θ=σ·λ，即任何代换都可以由σ再次代换而来。

例11.5 表达式集合{$P(x)$, $P(f(y))$}是可合一的，其最一般合一元为σ={f(y)/x}。因为对此集合的合一元θ ={f(a)/x, a/y}，有代换 λ ={a/y}，使　$\theta=\sigma\cdot\lambda$={f(y)/x}.{a/y}

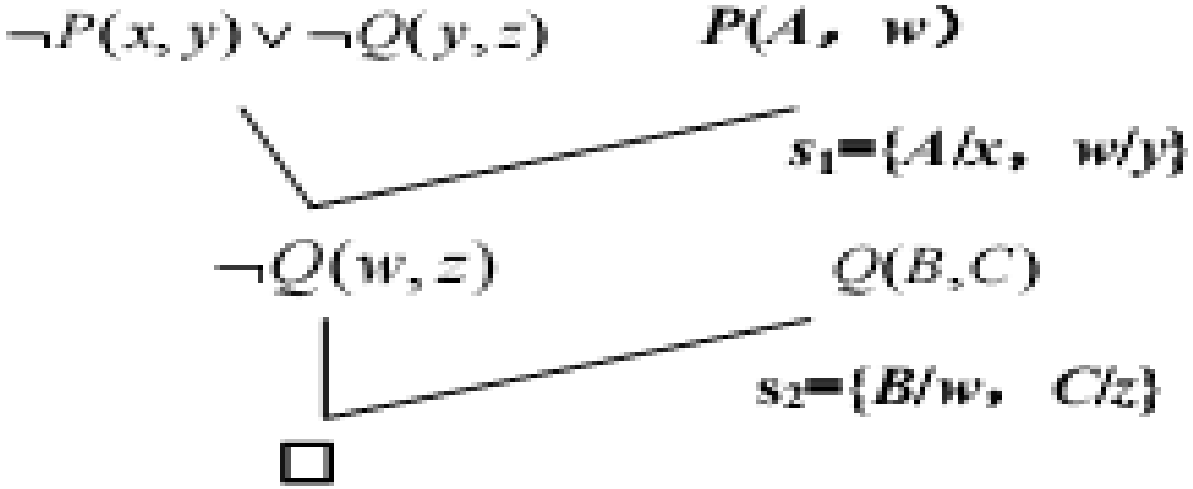
11.2 谓词演算的归结

三　合一算法

从下例就可以看到在合一时，尽可能使用最一般代换的重要性。

例11.6 求证 $\{\neg P(x,y) \vee \neg Q(y,z), P(A,w)\} \vdash \neg Q(B,C)$

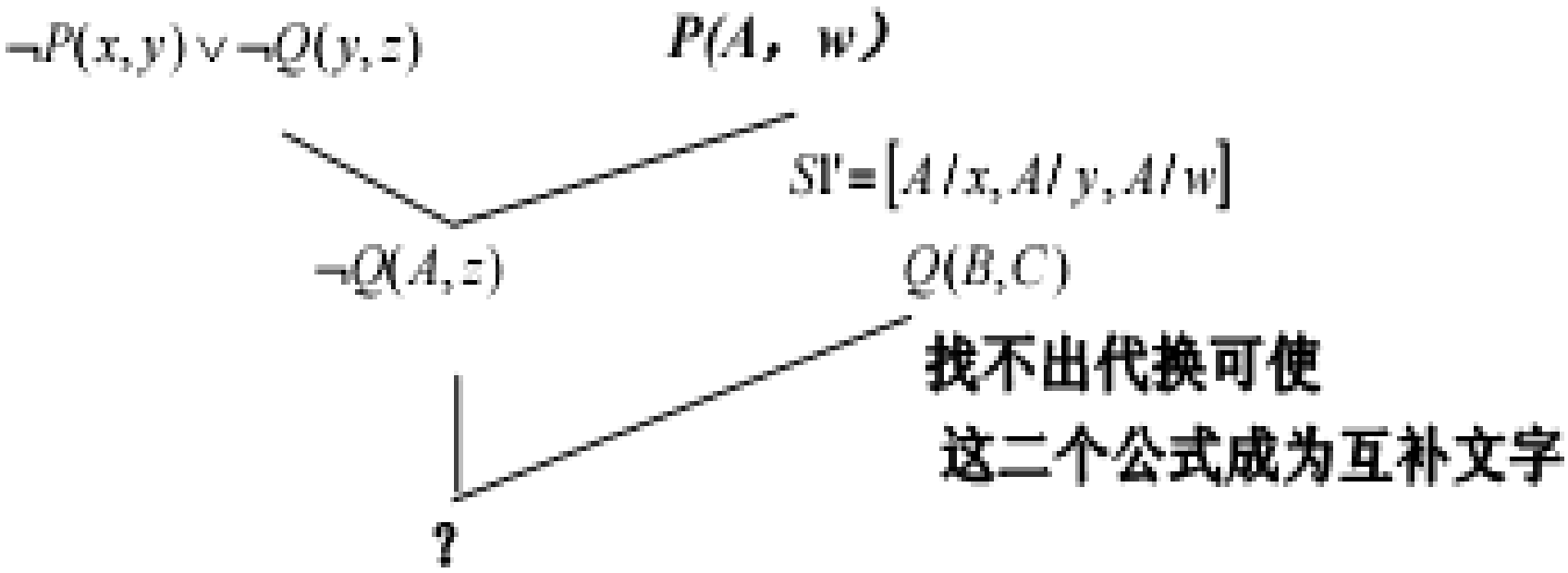
首先将Q(B, C)加入到子句集，再使用最一般的代换进行归结：



11.2 谓词演算的归结

三　合一算法

- 若使用限制较严格的代换进行归结：



11.2 谓词演算的归结

三　合一算法

对于合一运算算法的实现，需要作如下说明：

(1) 规定将每个文字和函数符表达成一个表，表中第一元素为谓词名，其余元素为变元。如：$P(x, y)$ 表示成 $(P\ x\ y)$。$P(\mathrm{f}(x), y)$ 表示成 $(P\ (f\ x)\ y)$，这样谓词和函数都统一表示成表。

(2) 判断两个文字能否合一，则要判断它们所对应的表项是否能够匹配，判断两个表项匹配的规则为：

- 变量可与常量、函数或变量匹配；
- 常量与常量，函数与函数，谓词与谓词相等才可以匹配。

11.2 谓词演算的归结

三　合一算法

(3) 判断两个文字能否合一，还必须判断表示文字的表的长度必须相等，并且谓词是否相同。

例如：$P(x, y)$化成表形式为$(P\ x\ y)$，其长度为3，

$Q(x, y, g(z))$化成表形式为(Q x y (g z))，其长度为4。

合一过程Unify(L1, L2)用一张表作为其返回的值。算法中的空表NIL表示可以匹配，但无需任何代换。返回由F值组成的表，则说明合一过程失败。

11.2 谓词演算的归结

三　合一算法

合一过程Unify(*L1*, *L2*)

1 .若*L1*或*L2*为一原子，则执行

1.1 若*L1*和*L2*恒等，则返回NIL

1.2 若*L1*为一变量，则执行：

若*L1*出现在*L2*中，则返回F；

否则返回（*L2*/ *L1*）

1.3 若*L2*为一变量，则执行：

若*L2*中出现在*L1*中，则返回F；

否则返回（*L1*/ *L2*）

1.4 返回*F*

2. 若length（*L1*）不等于length（*L2*），则返回F

3. 置*SUBST*为NIL，在结束本过程时，*SUBST*将包含用来合一*L1*和*L2*的所有代换。

4. 对于i：=1 到*L1*的元素数|*L1*|，执行：

4.1 对合一*L1*的第 i 个元素和*L2*的第 i 个元素调用UNIFY(*L1*.i, *L2*.i)，并将结果放在*S*中。

4.2 若*S*=F，则返回F 。

4.3 若*S*不等于NIL，则执行：

把S应用到L1和*L2*的剩余部分；

SUBST：=APPEND(*S*,*SUBST*)。

返回*SUBST*。

11.2 谓词演算的归结

三　合一算法

例11.7 设L1=(f Marcus)，L2=(f Caesar)。对合一算法进行跟踪。

1. L1，L2都不为原子
2. Length(L1)=Length(L2)
3. 量*SUBST*为NIL
4. 分别对(f f)、(Marcus Caesar)进行合一
5. f与f合一返回NIL
6. Marcus与Caesar不能合一，返回F
7. *SUBST* ={F}
8. 所以不能合一。

11.2 谓词演算的归结

四 变量分离标准化

例11.8 设知识库中有如下知识：

(1）若 x 是 y的父亲，则 x 不是女人;

father(x,y) →~woman(x)

(2）若 x 是 y的母亲，则 x 是女人;

mother(x,y) →woman(x)

(3）Chris是Mary的母亲;

mother(Chris, Mary)

(4）Chris是Bill的父亲;

father(Chris , Bill)

求证这些断言包含有矛盾。将上述公式化成子句集为：

（1） ~father(x,y) ∨~woman(x)

（2） ~mother(x,y) ∨ woman(x)

（3） mother(*Chris, Mary*)

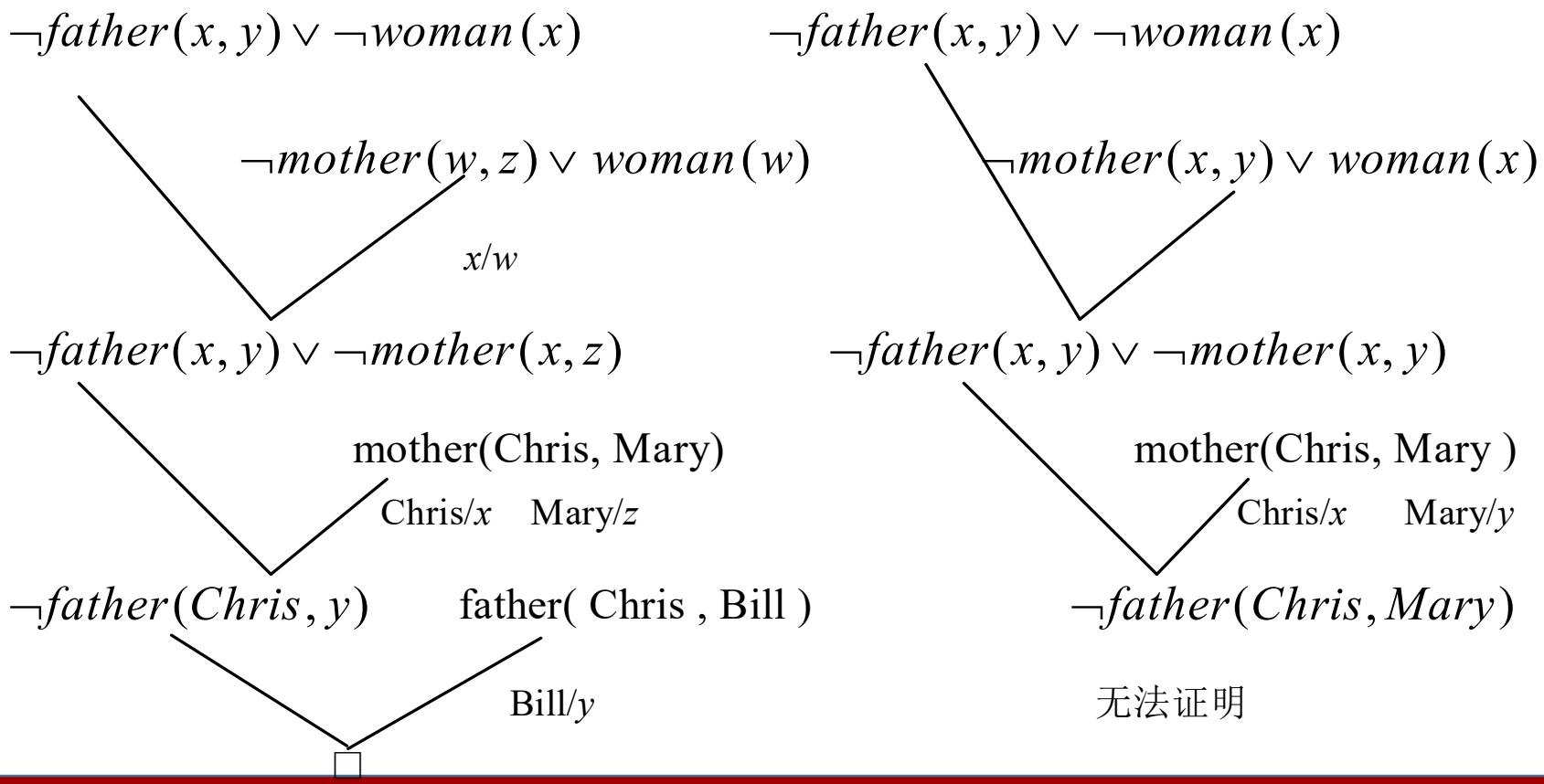
（4） father(*Chris , Bill*)

其中将(2)分离标准化后为：

(2）~mother(w,z) ∨ woman(w)

11.2 谓词演算的归结

四 变量分离标准化



11.2 谓词演算的归结

五 谓词演算的归结算法

设$F=\{F1,..., Fn\}$为给定的公理集，G为要求证的定理，则自动归结证明的过程为：

1. 将F中的一切公式变成子句形式。
2. 将~G变成子句形式，并加入F的子句构成子句集

   $S^* =\{C1, C2,..., Cn\}$。
3. 在出现一个矛盾(空子句)或无任何进展之前执行：

11.2 谓词演算的归结

五 谓词演算的归结算法

3.1 从S*中挑选一对子句C_1与C_2，并找出一个最一般合一元s使得：

当 $C_1s=(L_1 \vee P)s = L_1s \vee Ps$

$C_2s=(\sim L_2 \vee Q)s = \sim L_2s \vee Qs$时，

有 $L_1s = L_2s$成立。

3.2 令$C_{12}= Ps \vee Qs$作为C_1与C_2的归结式

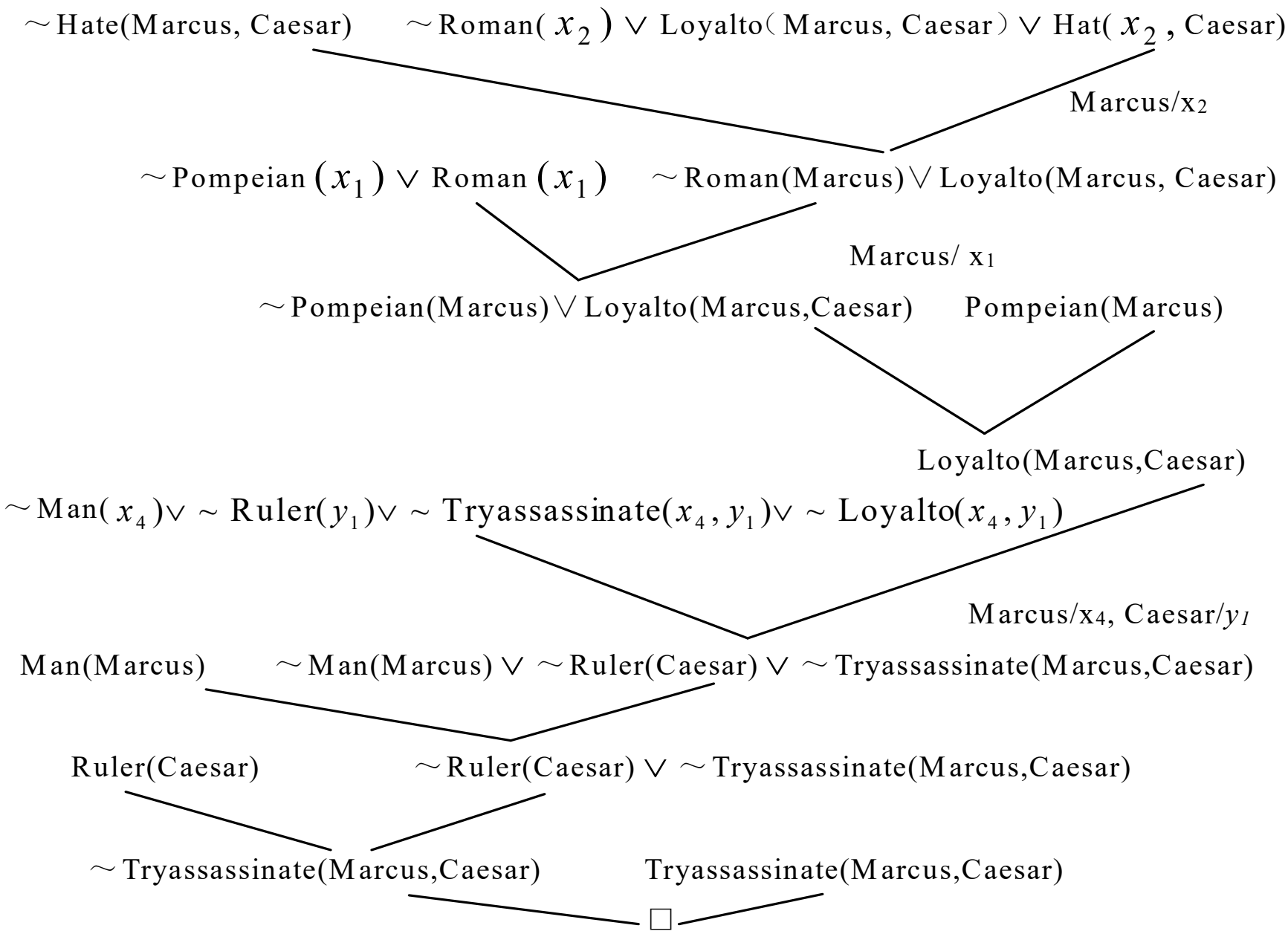
3.3 若C_{12}为空子句，即找到矛盾，G得证；

若C_{12}为非空子句，则将C_{12}加入S*。

11.2 谓词演算的归结

五 谓词演算的归结算法

例11.9 归结算法来对例11.3中用子句表达的知识进行归结, 得到的是如图11.2所示的一棵归结树。


~Hate(Marcus, Caesar)
~Roman(x_2) ∨ Loyalto（Marcus, Caesar）∨ Hat(x_2 , Caesar)
Marcus/x2
~Pompeian (x_1) ∨ Roman (x_1)
~Roman(Marcus)∨ Loyalto(Marcus, Caesar)
Marcus/ x1
~Pompeian(Marcus)∨ Loyalto(Marcus,Caesar)
Pompeian(Marcus)
Loyalto(Marcus,Caesar)
~Man(x_4)∨ ~ Ruler(y_1)∨ ~ Tryassassinate(x_4 , y_1)∨ ~ Loyalto(x_4 , y_1)
Marcus/x4, Caesar/y_1
Man(Marcus)
~Man(Marcus) ∨ ~Ruler(Caesar) ∨ ~Tryassassinate(Marcus,Caesar)
Ruler(Caesar)
~Ruler(Caesar) ∨ ~Tryassassinate(Marcus,Caesar)
~Tryassassinate(Marcus,Caesar)
Tryassassinate(Marcus,Caesar)
□

11.4 归结过程的控制策略

简化策略

支撑集策略

线性输入

11.4 归结过程的控制策略

主要思想：归结过程在寻找可归结子句时，子句集中的子句越多，需要付出的代价就会越大。如果在归结时能把子句集中无用的子句删除掉，就会缩小搜索范围，减少比较次数，从而提高归结效率。

简化策略对阻止不必要的归结式的产生来缩短归结过程是有效的。

尽管使用删除策略的归结，少做了归结但不影响产生空子句，就是说删除策略的归结推理是完备的。

11.4 归结过程的控制策略

简化策略

（1）**消去重言式：**若对S使用归结推理过程中, 当归结式C_j是重言式(永真式)，可将C_j删除。

（2）**归类：**设有两个子句C和D，若有置换σ使得$C\sigma \subset D$成立，则称子句C把子句D归类。

直观上：由于”小的”子句可以代表”大的”子句, 所以小的替代了大的了。

当C_j被S中子句和子句集的归结式$C_i(i<j)$所归类时, 可将C_j删除。

11.4 归结过程的控制策略

采用支撑集⇔完备

支撑集：设有不可满足子句集S的子集T，如果S-T是可满足的，则T是支持集。

采用支撑集策略时，从开始到得到空子句的整个归结过程中, 只选取不同时属于S-T的子句，在其间进行归结。就是说，至少有一个子句来自于支撑集T或由T导出的归结式。

对于待证公式 H1∧H2 ∧ H3 ⇒ C，则结论 C 的否定～C为支撑集。

11.4 归结过程的控制策略

例如：$A_1 \wedge A_2 \wedge A_3 \wedge \sim B$中，$\sim B$可以作为初始支撑集。要求每一次参加归结的亲本子句中，只要应该有一个是有目标公式的否定($\sim B$)所得到的子句或者它们的后裔。

支撑集策略的归结是完备的，对于不可满足的子句集使用支撑集策略，一定可以归结出空子句。

同样，所有可归结的谓词公式都可以用采用支撑集策略达到加快归结速度的目的。

11.4 归结过程的控制策略

例：S＝{P∨Q, ～P∨R, ～Q∨R, ～R}

取 T＝{～R}

支持集归结过程：

(1)	P∨Q	
(2)	～P∨R	
(3)	～Q∨R	
(4)	～R	
(5)	～P	(2) (4)
(6)	～Q	(3) (4)
(7)	Q	(1) (5)
(8)	P	(1) (6)
(9)	R	(3) (7)
(10)	□	(6) (7)

11.4 归结过程的控制策略

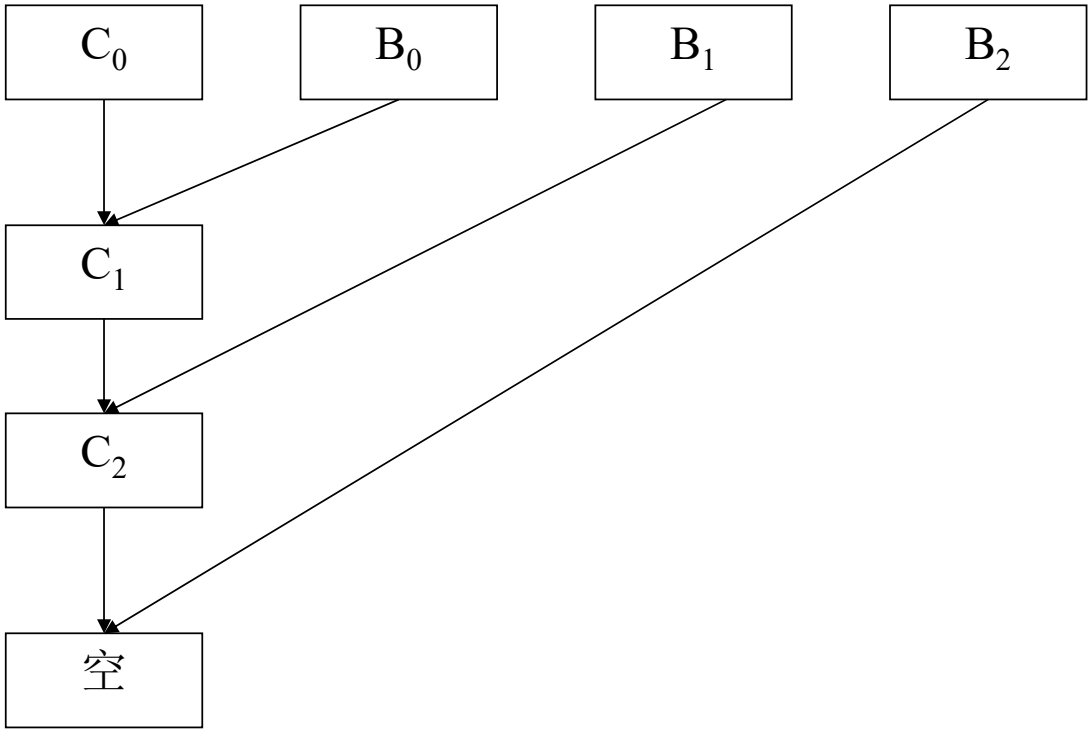
线性归结

线性归结策略首先从子句集中选取一个称作顶子句的子句 C_0 开始作归结。

归结过程中所得到的归结式C_i立即同另一子句 B_i 进行归结得归结式C_{i+1}。

而B_i属于S或是已出现的归结式 $C_j(j<i)$。即, 如下图所示归结得到的新子句立即参加归结。

11.4 归结过程的控制策略



线性归结策略示意图

11.4 归结过程的控制策略

例: $S=\{ P \vee Q , \sim P \vee Q, P \vee \sim Q, \sim P \vee \sim Q \}$

选取顶子句$C_0=P \vee Q$。

线性归结过程:

(1)	$P \vee Q$	前提
(2)	$\sim P \vee Q$	前提
(3)	$P \vee \sim Q$	前提
(4)	$\sim P \vee \sim Q$	前提
(5)	Q	(1)(2)
(6)	P	(5)(3)
(7)	$\sim Q$	(6)(4)
(8)	□	(7)(5)

顶子句的选择直接影响着归结的效率。如可选得C_0使$S-\{C_0\}$是可满足的。

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。

归结反演

- 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。**

归结反演

- 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）$P(A) \vee P(B) \vee P(C)$

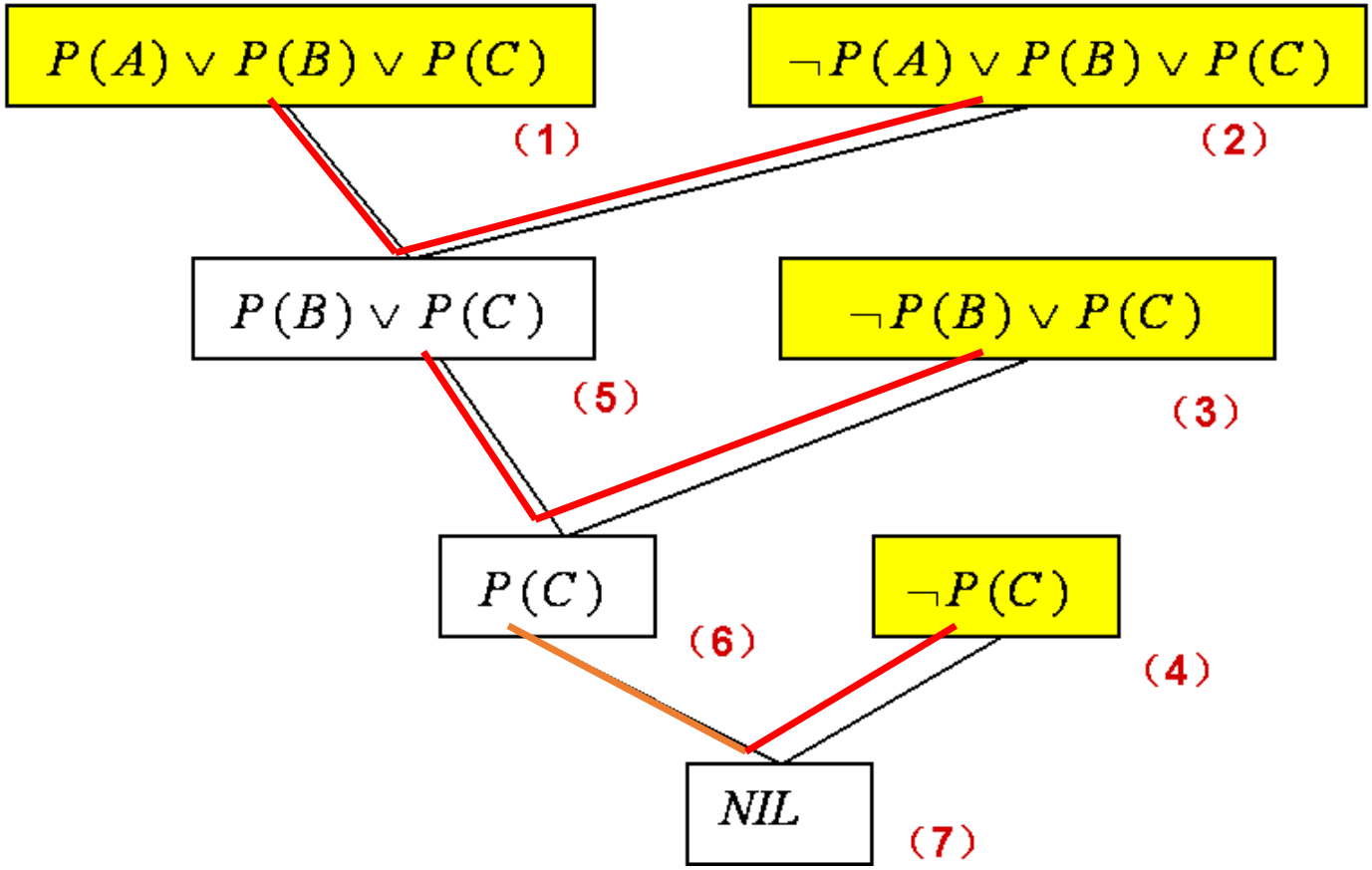
  （2）$\neg P(A) \vee P(B) \vee P(C)$

  （3）$\neg P(B) \vee P(C)$

  （4）$\neg P(C)$

归结反演

- 应用归结原理进行归结：



归结反演

- 例 已知：

  规则1：任何人的兄弟不是女性；

  规则2：任何人的姐妹必是女性。

  事 实：*Mary* 是 *Bill* 的姐妹。

  求证： *Mary* 不是 *Tom* 的兄弟。

- 证明：定义谓词

  brother (x, y)：x 是 y 的兄弟

  sister (x, y)： x 是 y 的姐妹

  woman (x)：x 是女性

归结反演

- 证明：将规则与事实用谓词公式表示：

（1）$(\forall x)(\forall y)(brother(x,y) \rightarrow \neg woman(x))$

（2）$(\forall x)(\forall y)(sister(x,y) \rightarrow woman(x))$

（3）$sister(Mary, Bill)$

- 把要求证的结论用谓词公式表示出来并否定，得：

（4）$brother(Mary, Tom)$

- 把上述公式化成子句集：

$C_1 = \neg brother(x,y) \vee \neg woman(x)$

$C_2 = \neg sister(x,y) \vee woman(x)$

$C_3 = sister(Mary, Bill)$

$C_4 = brother(Mary, Tom)$

- 将子句集进行归结：

$C_{23} = woman(Mary)$

$C_{123} = \neg brother(Mary, y)$

$C_{1234} = NIL$

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。

应用归结原理求解问题

- 例14 已知：

  F_1：王（Wang）先生是小李（Li）的老师。

  F_2：小李与小张（Zhang）是同班同学。

  F_3：如果x与y是同班同学，则x的老师也是y的老师。

  求：小张的老师是谁？

应用归结原理求解问题

F_1：王（Wang）先生是小李（Li）的老师。
F_2：小李与小张（Zhang）是同班同学。
F_3：如果 x与 y 是同班同学，则 x的老师也是 y的老师。
求：小张的老师是谁？

◆ **解：**

- **定义谓词：**

  $T(x,y)$： x 是 y 的老师。

  $C(x,y)$： x 与 y 是同班同学。

- 把已知前提表示成谓词公式：

  F_1： $T(Wang,Li)$

  F_2： $C(Li,Zhang)$

  F_3： $(\forall x)(\forall y)(\forall z)(C(x,y)\wedge T(z,x)\rightarrow T(z,y))$

- 把目标表示成谓词公式，并把它否定后与 *ANSWER* 析取：

  G： $\neg(\exists x)T(x,Zhang)\vee ANSWER(x)$

应用归结原理求解问题

- 把上述公式化为子句集：

（1） $T(Wang, Li)$

（2） $C(Li, Zhang)$

（3） $\neg C(x, y) \vee \neg T(z, x) \vee T(z, y)$

（4） $\neg T(u, Zhang) \vee ANSWER(u)$

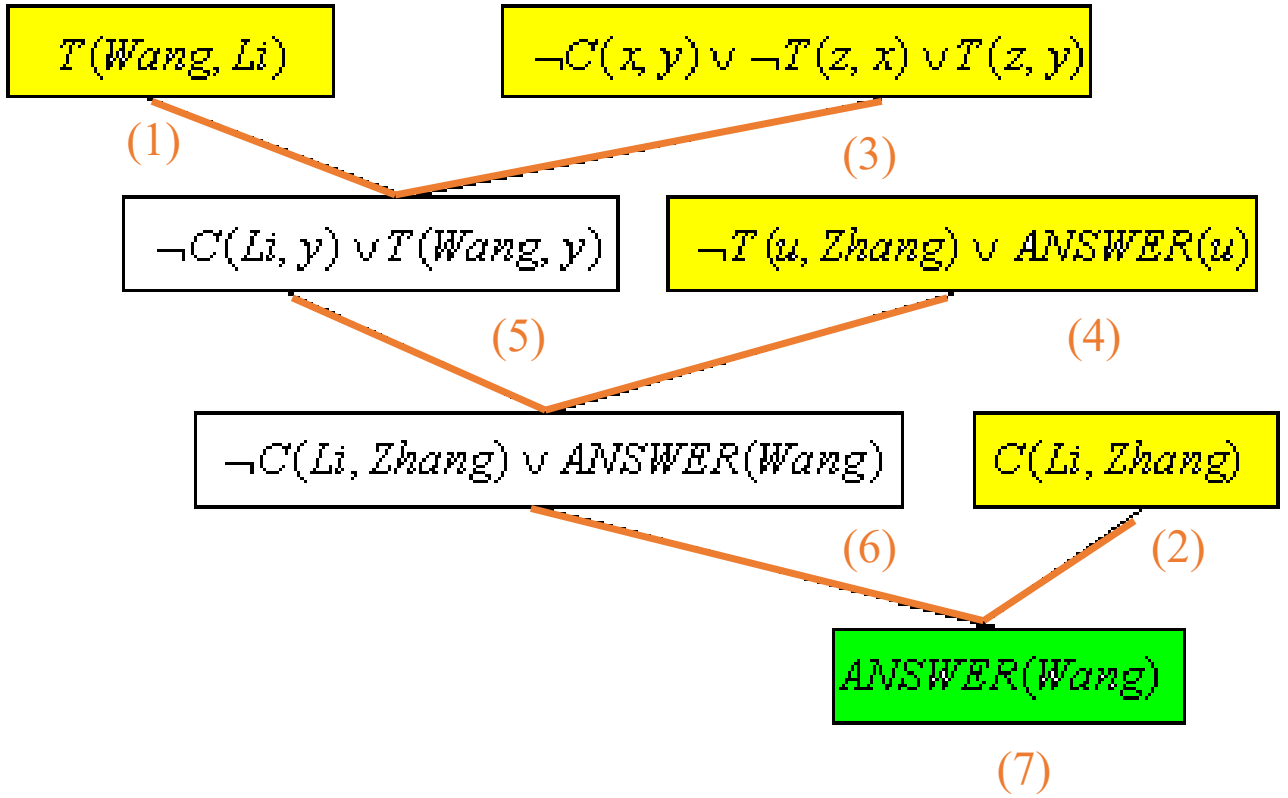
- 应用归结原理进行归结：

（5） $\neg C(Li, y) \vee T(Wang, y)$ （1）与（3）归结

（6） $\neg C(Li, Zhang) \vee ANSWER(Wang)$ （4）与（5）归结

（7） $ANSWER(Wang)$ （2）与（6）归结

应用归结原理求解问题


T(Wang, Li)
¬C(x, y) ∨ ¬T(z, x) ∨ T(z, y)
(1)
(3)
¬C(Li, y) ∨ T(Wang, y)
¬T(u, Zhang) ∨ ANSWER(u)
(5)
(4)
¬C(Li, Zhang) ∨ ANSWER(Wang)
C(Li, Zhang)
(6)
(2)
ANSWER(Wang)
(7)

利用推理破案的实例

在上述逻辑公式中，谓词 $SK(x,y)$ 表示怀疑 x 杀了 y； $L(x)$ 表示 x 生活在这栋房子里； $H(x,y)$ 表示 x 恨 y； $R(x,y)$ 表示 x 比 y 富有。

例 2.13 破案问题。在一栋房子里发生了一件神秘的谋杀案，现在可以肯定以下几点事实：

1. 在这栋房子里仅住有 A，B，C 三人 $\forall x L(x) \rightarrow x = A \vee x = B \vee x = C$
2. 是住在这栋房子里的人杀了 A $\exists y SK(y, A) \wedge L(y)$
3. 谋杀者非常恨受害者 $\forall x SK(x, A) \rightarrow H(x, A)$
4. A 所恨的人，C 一定不恨 $\forall x H(A, x) \rightarrow \sim H(C, x)$
5. 除了 B 之外，A 恨所有的人 $\forall x \sim EQ(x, B) \rightarrow H(A, x)$
6. B 恨所有不比 A 富有的人 $\forall y \sim R(y, A) \rightarrow H(B, y)$
7. A 所恨的人，B 也恨 $\forall y H(A, y) \rightarrow H(B, y)$
8. 没有一个恨所有的人 $\forall x \exists y \sim H(x, y)$
9. 杀人嫌疑犯一定不会比受害者富有 $\forall x SK(x, A) \rightarrow \sim R(x, A)$

为了推理需要，增加如下常识：

10. A 不等于 B $\sim EQ(A, B)$

根据这些事实，读者可以试图推理一下，谋杀者究竟是谁呢？

1. $\sim L(x) \vee EQ(x, A) \vee EQ(x, B) \vee EQ(x, C)$
2. $SK(K_0, A)$ 3. $L(K_0)$
4. $\sim SK(x, A) \vee H(x, A)$
5. $\sim H(A, x) \vee \sim H(C, x)$
6. $EQ(x, B) \vee H(A, x)$
7. $R(y, A) \vee H(B, y)$
8. $\sim H(A, y) \vee H(B, y)$
9. $\sim H(x, f(x))$
10. $\sim SK(x, A) \vee \sim R(x, A)$
11. $\sim EQ(A, B)$

现在要回答“谁杀 A”，因此设

12. 为 $\sim SK(x, A) \vee SK(x, A)$

利用推理破案的实例

13. $EQ(K_0, A) \vee EQ(K_0, B) \vee EQ(K_0, C)$ 由 1，3
14. $H(A, A)$ 由 11，6
15. $\sim H(C, A)$ 由 14，5
16. $\sim SK(C, A)$ 由 15，4
17. $EQ(y, B) \vee H(B, y)$ 由 6，8
18. $\sim SK(x, A) \vee H(B, x)$ 由 7，10
19. $EQ(f(B), B)$ 由 17，9
20. $\sim H(B, B)$ 由 19，9 等式归结
21. $\sim SK(B, A)$ 由 20，18
22. $SK(A, A) \vee EQ(K_0, B) \vee EQ(K_0, C)$ 由 13，2
23. $SK(A, A) \vee SK(B, A) \vee EQ(K_0, C)$ 由 22，2
24. $SK(A, A) \vee SK(B, A) \vee SK(C, A)$ 由 23，2
25. $SK(A, A) \vee SK(B, A)$ 由 24，16
26. $SK(A, A)$ 由 21，25
27. $\square \vee SK(A, A)$ 由 26，12

推导表明 A 是自杀。