第 5 章

基于一阶谓词的机器推理

基于一阶谓词(first-order predicate)的机器推理也称自动推理,它是早期人工智能的主要研究内容之一。一阶谓词是一种表达力很强的形式语言,而且这种语言很适合数字计算机处理,因而成为知识表示的首选。基于这种语言,不仅可以实现类似于人类推理的自然演绎法机器推理,而且也可实现不同于人类的归结(或称消解)法机器推理。本章主要介绍作为一种知识表示形式的一阶谓词和基于它的机器推理。

5.1 一阶谓词逻辑

5.1.1 谓词,函数,量词

定义 5-1 表达式

$$P(t_1,t_2,\cdots,t_n)$$

称为一个 n 元谓词,或简称谓词。其中,P 是谓词名或谓词符号,也称谓词,表示对象的属性、状态、关系、联系或行为;$t_1,t_2,\cdots,t_n$ 被称为谓词的项,一般代表对象。例如:

```
prime(2)
friend(张三,李四)
```

就是两个谓词。其中,prime(2)是一元谓词,表示“2 是个素数”;friend(张三,李四)是二元谓词,表示“张三和李四是朋友”。

谓词中的项 $t_1,t_2,\cdots,t_n$ 可以是代表具体事物的符号或数值,这样的项被称为**个体常元**;也可以是取不同的值的变元,这样的项被称为**个体变元**。显然,当项 $t_1,t_2,\cdots,t_n$ 全为常元时,$P(t_1,t_2,\cdots,t_n)$ 就表示一个命题。但当项 $t_1,t_2,\cdots,t_n$ 中含有变元时,$P(t_1,t_2,\cdots,t_n)$ 则是一个**命题形式**,称为(n 元)命题函数或谓词命名式。个体变元的取

值范围被称为**个体域**(或论述域),包揽一切事物的集合则称为**全总个体域**。

由于谓词有严格的语法格式,所以,谓词实际上就是一种形式语言。它可以表示自然语言命题。换句话说,谓词就是自然语言命题的一种形式化表示。

同一个个体域的个体之间往往存在某种对应关系。例如在人类集合中,每一个人都有一个母亲。又如在实数域中任何两个数都有它们的和与之对应。为了表达个体之间的对应关系,谓词逻辑中通常借用数学中函数的概念和记法,用如下形式

$$f(x_1,x_2,\cdots,x_n)$$

表示个体 $x_1,x_2,\cdots,x_n$ 所对应的个体 y,并称之为(n 元)个体函数,简称**函数**(或函词、函词命名式),其中 f 是函数符号。有了函数的概念和记法,谓词的表达能力就更强了。例如,可用 doctor(father(Li))表示"小李的父亲是医生",用 equa(sq(x),y)表示"x 的平方等于 y"。

下面约定用大写英文字母作为谓词符号,用小写字母 f、g、h 等表示函数符号,用小写字母 x、y、z 等作为个体变元符号,用小写字母 a、b、c 等作为个体常元符号。

在谓词逻辑中,将"所有""一切""任一""全体""凡是"等词统称为**全称量词**(universal quantifier),记为 $\forall$;"存在""一些""有些""至少有一个"等词统称为**存在量词**(existential quantifier),记为 $\exists$。

引入量词后,谓词的表达能力被进一步扩充。例如命题"凡是人都有名字",就可以表示为

$$\forall x(P(x)\rightarrow N(x))$$

其中,$P(x)$表示"x 是人",$N(x)$表示"x 有名字",该式可读作"对于任一 x,如果 x 是人,则 x 有名字"。这里的个体域取为全总个体域。如果把个体域取为人类集合,则该命题就可以表示为:

$$\forall xN(x)$$

同理,可以把命题"存在不是偶数的整数"表示为:

$$\exists x(I(x)\wedge\neg E(x))$$

其中 $I(x)$表示"x 是整数",$E(x)$表示"x 是偶数"。此式可读作"存在 x,x 是整数并且 x 不是偶数"。

紧接于量词之后被量词作用(即说明或限定)的命题函数式被称为该量词的**辖域**。例如:

(1) $\forall xP(x)$

(2) $\forall x(H(x)\rightarrow G(x,y))$

(3) $\exists xA(x)\wedge B(x)$

其中,(1)中的 $P(x)$为全称量词 $\forall$ 的辖域,(2)中的 $H(x)\rightarrow G(x,y)$为全称量词 $\forall$ 的辖域,(3)中的 $A(x)$为存在量词 $\exists$ 的辖域,但 $B(x)$并非 $\exists$ 的辖域。

量词后的变元被称为量词的**指导变元**(或作用变元),而在一个量词的辖域中与该量词的指导变元相同的变元被称为**约束变元**,其他变元(如果有的话)则被称为**自由变元**。

例如 $\forall x$、$\exists y$ 中的 x、y 分别是量词 $\forall$、$\exists$ 的指导变元;上面(2)式中的 x 为约束变

元，而 y 则为自由变元，(3)式中 $A(x)$ 中的 x 为约束变元，但 $B(x)$ 中的 x 为自由变元。一个变元在一个公式中既可约束出现，又可自由出现，但为了避免混淆，往往通过改名，使得一个公式中的变元仅以一种形式出现。例如上面的(3)式通过改名就变为

(3′) $\exists xA(x) \wedge B(y)$

约束变元的改名规则如下：

(1) 对需改名的变元，应同时更改该变元在量词及其辖域中的所有出现。

(2) 新变元符号必须是量词辖域内原先没有的，最好是公式中也未出现过的。

例如公式 $\exists xP(x) \wedge Q(x)$ 可改为 $\exists yP(y) \wedge Q(x)$，但二者的意义相同。

在谓词前加上量词，称作谓词中相应的个体变元被**量化**，例如 $\forall xA(x)$ 中的 x 被量化，$\exists yB(y)$ 中的 y 被量化。如果一个谓词中的所有个体变元都被量化，则这个谓词就变为一个命题了。例如，设 $P(x)$ 表示“x 是素数”，则 $\forall xP(x)$ 和 $\exists xP(x)$ 都是命题。这样就有两种从谓词(即命题函数)得到命题的方法：一种是给谓词中的个体变元代入个体常元，另一种就是把谓词中的个体变元全部量化。

需要说明的是，仅个体变元被量化的谓词称为**一阶谓词**。如果不仅个体变元被量化，而且函数符号和谓词符号也被量化，那样的谓词则称为**二阶谓词**。本书只涉及一阶谓词，所以，以后提及的谓词都是指一阶谓词。

如果一个命题形式中的所有个体变元都被量化，或者所有变元都是约束变元(或无自由变元)，则这个命题形式就是一个命题。特别地，称 $\forall xA(x)$ 为全称命题，$\exists xA(x)$ 为特称命题。对于这两种命题，当个体域为有限集时(设有 n 个元素)，有下面的等价式

$$\forall xA(x) \Leftrightarrow A(a_1) \wedge A(a_2) \wedge \cdots \wedge A(a_n)$$

$$\exists xA(x) \Leftrightarrow A(a_1) \vee A(a_2) \vee \cdots \vee A(a_n)$$

这两个式子也可以推广到个体域为可数无限集。

5.1.2　谓词公式

定义 5-2

(1) 个体常元和个体变元都是项。

(2) 设 f 是 n 元函数符号，若 $t_1, t_2, \cdots, t_n$ 是项，则 $f(t_1, t_2, \cdots, t_n)$ 也是项。

(3) 只有有限次使用(1)(2)得到的符号串才是项。

定义 5-3　设 P 为 n 元谓词符号，$t_1, t_2, \cdots, t_n$ 是项，则 $P(t_1, t_2, \cdots, t_n)$ 称为原子谓词公式，简称原子公式或者原子。

从原子谓词公式出发，通过命题联结词和量词，可以组成复合谓词公式。下面给出谓词公式的严格定义，即谓词公式的生成规则。

定义 5-4

(1) 原子公式是谓词公式。

(2) 若 P, Q 是谓词公式，则 $\neg P, P \wedge Q, P \vee Q, P \rightarrow Q, P \leftrightarrow Q, \forall xP, \exists xP$ 也是谓词公式。

(3) 只有有限步应用(1)、(2)生成的公式才是谓词公式。

由项的定义，当 $t_1, t_2, \cdots, t_n$ 全为个体常元时，所得的原子谓词公式就是原子命题公

式。所以,全体命题公式也都是谓词公式。谓词公式也称为谓词逻辑中的合适(式)公式,记为 Wff。

定义 5-5 设 G 为如下形式的谓词公式:

$$P_1 \wedge P_2 \wedge \cdots \wedge P_n$$

其中,$P_i(i=1,2,\cdots,n)$形如 $L_1 \vee L_2 \vee \cdots \vee L_m$,$L_j(j=1,2,\cdots,m)$为原子公式或其否定,则 G 称为合取范式。

例如:

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

就是一个合取范式。

应用逻辑等价式,任一谓词公式都可以化为与之等价的合取范式,这个合取范式就称为原公式的合取范式。一个谓词公式的合取范式一般不唯一。

定义 5-6 设 G 为如下形式的命题公式:

$$P_1 \vee P_2 \vee \cdots \vee P_n$$

其中,$P_i(i=1,2,\cdots,n)$形如 $L_1 \wedge L_2 \wedge \cdots \wedge L_m$,$L_j(j=1,2,\cdots,m)$为原子公式或其否定,则 G 称为析取范式。

例如:

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

就是一个析取范式。

应用逻辑等价式,任一谓词公式都可以化为与之等价的析取范式,这个析取范式被称为原公式的析取范式。同样,一个谓词公式的析取范式一般也不唯一。

定义 5-7 谓词公式 G 在个体域 D 中的一个解释 I 是指:

(1) 对 G 中每一个常元符号指定 D 中的一个元素。

(2) 对 G 中每一个 n 元函数符号指定一个函数,即 D^n 到 D 的一个映射。

(3) 对每个 n 元谓词符号指定一个谓词,即 D^n 到{T,F}的一个映射。

例 5-1 设谓词公式 $G=\exists x(P(f(x)) \wedge Q(x,f(a)))$,给出如下的一个解释 I:

$D=\{1,2\}$

$\dfrac{a}{1}$

$\dfrac{f(1)}{2}$, $\dfrac{f(2)}{1}$

$\dfrac{P(1)}{\mathrm{F}}$, $\dfrac{P(1)}{\mathrm{T}}$, $\dfrac{Q(1,1)}{\mathrm{T}}$, $\dfrac{Q(1,2)}{\mathrm{T}}$, $\dfrac{Q(2,1)}{\mathrm{F}}$, $\dfrac{Q(2,2)}{\mathrm{T}}$

因为 $x=1$ 时,

$$P(f(x)) \wedge Q(x,f(a)) = P(f(1)) \wedge Q(1,f(1)) = P(2) \wedge Q(1,2) = \mathrm{T} \wedge \mathrm{T} = \mathrm{T}$$

所以,谓词公式 G 在 I 下为真。

定义 5-8 设 G、H 是两个谓词公式,D 是它们的公共个体域,若对于 D 中的任一解释,G、H 有相同的真值,则称公式 G、H 在个体域 D 上逻辑等价。若 G、H 在所有个体域上等价,则称 G、H 逻辑等价,记为 $G \Leftrightarrow H$。

定义 5-9　设 G、H 是两个谓词公式，D 是它们的公共个体域，若对于 D 中的任一解释，当 G 真时 H 也真，则称在个体域 D 上公式 G 逻辑蕴涵公式 H。若在所有个体域上 G 都逻辑蕴涵 H，则称 G 逻辑蕴涵 H，或称 H 是 G 的逻辑结果，记为 $G \Rightarrow H$。

由上面的两个定义，可以证明下面的定理。

定理 5-1　设 G、H 是两个谓词公式，$G \Leftrightarrow H$ 的充分必要条件是 $G \Rightarrow H$ 且 $H \Rightarrow G$。

表 5-1 和表 5-2 所列的就是形式逻辑中常用的一些逻辑等价式和逻辑蕴涵式。

表 5-1　常用逻辑等价式

E_1	$\neg\neg P \Leftrightarrow P$	双重否定律
E_2	$P \wedge P \Leftrightarrow P$	等幂律
E_3	$P \vee P \Leftrightarrow P$	
E_4	$P \wedge Q \Leftrightarrow Q \wedge P$	交换律
E_5	$P \vee Q \Leftrightarrow Q \vee P$	
E_6	$(P \wedge Q) \wedge R \Leftrightarrow P \wedge (Q \wedge R)$	结合律
E_7	$(P \vee Q) \vee R \Leftrightarrow P \vee (Q \vee R)$	
E_8	$P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R)$	分配律
E_9	$P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \wedge (P \vee R)$	
E_{10}	$P \wedge (P \vee Q) \Leftrightarrow P$	吸收律
E_{11}	$P \vee (P \wedge Q) \Leftrightarrow P$	
E_{12}	$\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$	摩根律
E_{13}	$\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$	
E_{14}	$P \rightarrow Q \Leftrightarrow \neg P \vee Q$	蕴含表达式
E_{15}	$P \leftrightarrow Q \Leftrightarrow (P \rightarrow Q) \wedge (Q \rightarrow P)$	等价表达式
E_{16}	$P \wedge T \Leftrightarrow P$	
E_{17}	$P \wedge F \Leftrightarrow F$	
E_{18}	$P \vee T \Leftrightarrow T$	
E_{19}	$P \vee F \Leftrightarrow P$	
E_{20}	$P \wedge \neg P \Leftrightarrow F$	矛盾律
E_{21}	$P \vee \neg P \Leftrightarrow T$	排中律
E_{22}	$P \rightarrow (Q \rightarrow R) \Leftrightarrow P \wedge Q \rightarrow R$	输出律
E_{23}	$(P \rightarrow Q) \wedge (P \rightarrow \neg Q) \Leftrightarrow \neg P$	归谬律
E_{24}	$P \rightarrow Q \Leftrightarrow \neg Q \rightarrow \neg P$	逆否律
E_{25}	$\forall x P \Leftrightarrow P$	P 中不含约束变元 x
E_{26}	$\exists x P \Leftrightarrow P$	P 中不含约束变元 x
E_{27}	$\forall x(A(x) \wedge B(x)) \Leftrightarrow \forall x A(x) \wedge \forall x B(x)$	量词分配律
E_{28}	$\exists x(A(x) \vee B(x)) \Leftrightarrow \exists x A(x) \vee \exists x B(x)$	
E_{29}	$\neg \forall x A(x) \Leftrightarrow \exists x \neg A(x)$	量词转换律
E_{30}	$\neg \exists x A(x) \Leftrightarrow \forall x \neg A(x)$	
E_{31}	$\forall x A(x) \wedge P \Leftrightarrow \forall x(A(x) \wedge P)$	量词辖域扩张及收缩律
E_{32}	$\forall x A(x) \vee P \Leftrightarrow \forall x(A(x) \vee P)$	
E_{33}	$\exists x A(x) \wedge P \Leftrightarrow \exists x(A(x) \wedge P)$	
E_{34}	$\exists x A(x) \vee P \Leftrightarrow \exists x(A(x) \vee P)$	（P 为不含约束变元 x 的谓词公式）
E_{35}	$\forall x \forall y P(x,y) \Leftrightarrow \forall y \forall x P(x,y)$	

续表

E_{36}	$\exists x \exists y P(x,y) \Leftrightarrow \exists y \exists x P(x,y)$	
E_{37}	$\forall x A(x) \rightarrow P \Leftrightarrow \exists x (A(x) \rightarrow P)$	
E_{38}	$\exists x A(x) \rightarrow P \Leftrightarrow \forall x (A(x) \rightarrow P)$	
E_{39}	$P \rightarrow \forall x A(x) \Leftrightarrow \forall x (P \rightarrow A(x))$	
E_{40}	$P \rightarrow \exists x A(x) \Leftrightarrow \exists x (P \rightarrow A(x))$	(P 为不含约束变元 x 的谓词公式)

表 5-2　常用逻辑蕴涵式

I_1	$P \Rightarrow P \vee Q$	附加律
I_2	$P \wedge Q \Rightarrow P, P \wedge Q \Rightarrow Q$	简化律
I_3	$(P \rightarrow Q) \wedge P \Rightarrow Q$	假言推理(分离规则)
I_4	$(P \rightarrow Q) \wedge \neg Q \Rightarrow \neg P$	拒取式
I_5	$(P \vee Q) \wedge \neg P \Rightarrow Q$	析取三段论
I_6	$(P \rightarrow Q) \wedge (Q \rightarrow R) \Rightarrow P \rightarrow R$	假言三段论
I_7	$P \rightarrow Q \Rightarrow (Q \rightarrow R) \rightarrow (P \rightarrow R)$	
I_8	$(P \rightarrow Q) \wedge (R \rightarrow S) \Rightarrow P \wedge R \rightarrow Q \wedge S$	
I_9	$(P \leftrightarrow Q) \wedge (Q \leftrightarrow R) \Rightarrow P \leftrightarrow R$	
I_{10}	$P, Q \Rightarrow P \wedge Q$	合取式
I_{11}	$\forall x A(x) \Rightarrow A(y)$	y 是个体域中任一确定元素,全称指定规则(Universal Specification,US)
I_{12}	$\exists x A(x) \Rightarrow A(y)$	y 是个体域中某一确定元素,存在指定规则(Existential Specification,ES)
I_{13}	$A(y) \Rightarrow \forall x A(x)$	y 是个体域中任一确定元素,全称推广规则(Universal Generalization,UG)
I_{14}	$A(y) \Rightarrow \exists x A(x)$	y 是个体域中某一确定元素,存在推广规则(Existential Generalization,EG)
I_{15}	$\forall x A(x) \Rightarrow \exists x A(x)$	
I_{16}	$\forall x A(x) \vee \forall x B(x) \Rightarrow \forall x (A(x) \vee B(x))$	
I_{17}	$\exists x (A(x) \wedge B(x)) \Rightarrow \exists x A(x) \wedge \exists x B(x)$	
I_{18}	$\forall x \forall y P(x,y) \Rightarrow \exists y \exists x P(x,y)$	
I_{19}	$\exists y \forall x P(x,y) \Rightarrow \forall x \exists y P(x,y)$	
I_{20}	$\forall x \exists y P(x,y) \Rightarrow \exists y \exists x P(x,y)$	

5.1.3　永真式与推理规则

定义 5-10　设 P 为谓词公式,D 为其个体域,对于 D 中的任一解释 I:

(1) 若 P 恒为真,则称 P 在 D 上永真(或有效)或是 D 上的永真式。

(2) 若 P 恒为假,则称 P 在 D 上永假(或不可满足)或是 D 上的永假式。

(3) 若至少有一个解释,可使 P 为真,则称 P 在 D 上可满足或是 D 上的可满足式。

定义 5-11　设 P 为谓词公式,对于任何个体域:

(1) 若 P 都永真,则称 P 为永真式。

(2) 若 P 都永假,则称 P 为永假式。

(3) 若 P 都可满足,则称 P 为可满足式。

由于谓词公式的真值与个体域及解释有关,考虑到个体域的数目和个体域中元素数目无限的情形,所以要通过一个机械地执行的方法(即算法),判断一个谓词公式的永真性一般是不可能的,所以称一阶谓词逻辑是不可判定的(但它是半可判定的)。

定理 5-2　设 G、H 是两个谓词公式,则:

$$G \leftrightarrow H \text{ 永真的充分必要条件是} G \Leftrightarrow H$$

$$G \rightarrow H \text{ 永真的充分必要条件是} G \Rightarrow H$$

由篇幅所限,该定理的证明从略。

我们知道,推理是由一个或几个命题(称为前提)得到一个新命题(称为结论)的(思维)过程。可以看出,若把一个推理过程符号化,其形式就是一个蕴涵式。一个正确有效的推理首先要求其推理形式必须正确。那么,怎样判断一个推理的形式是否正确呢?下面的定理 5-3 给出了回答。

定理 5-3　一个推理形式正确,当且仅当其对应的蕴涵式永真。

由篇幅所限,该定理的证明从略。

正确的推理形式是推理中应该遵循的形式,所以,在逻辑学中,正确的推理形式被称为**推理规则**。这样,由定理 5-2 和定理 5-3 可得,上面表 5-2 中的逻辑蕴涵式都是推理规则,而表 5-1 中的一个逻辑等价式则可以分解为两个推理规则。

5.1.4　自然语言命题的谓词形式表示

由上所述,利用谓词公式这种形式语言可以将自然语言中的陈述语句严格地表示为一种符号表达式。将自然语言命题用谓词形式表示的一般方法是:

(1) 简单命题可以直接用原子公式来表示。

(2) 复合命题则需要先找出支命题,并将其符号化为原子公式,然后根据支命题之间的逻辑关系选用合适的连接词($\neg$,$\wedge$,$\vee$,$\rightarrow$,$\leftrightarrow$)和量词($\forall$,$\exists$)将这些原子公式连接起来。

一般来说,一个复合命题的形式化表示法并不唯一,即同一个复合命题可能符号化为不同形式的谓词公式。

由于不同的个体变元可能有不同的个体域,为了方便和统一起见,用谓词公式表示命题时,一般总取全总个体域,然后再采取使用限定谓词的办法来指出每个个体变元的个体域。具体来讲,就是:

(1) 对全称量词,把限定谓词作为蕴涵式之前件加入,即 $\forall x(P(x) \rightarrow \cdots)$。

(2) 对存在量词,把限定量词作为一个合取项加入,即 $\exists x(P(x) \wedge \cdots)$。

这里的 $P(x)$就是限定谓词。

例 5-2　用谓词公式表示命题:不存在最大的整数。

解　用 $I(x)$表示:x 是整数,用 $D(x,y)$表示:x 大于 y。则原命题就可形式化为:

$$\neg\, \exists x(I(x)) \wedge \forall y(I(y) \rightarrow D(x,y))$$

或

$$\forall x(I(x)) \rightarrow \exists y(I(y) \wedge D(y,x))$$

例 5-3 设有命题：对于所有的自然数 x、y，均有 $x+y>x$。用谓词公式表示之。

解 用 $N(x)$表示：x 是整数，$S(x,y)$表示函数：$s=x+y$，$D(x,y)$表示：x 大于 y，则原命题可形式化为谓词公式

$$\forall x \forall y(N(x) \wedge N(y) \rightarrow D(S(x,y),x))$$

例 5-4 将命题“某些人对某些食物过敏”用谓词公式表示。

解 用 $P(x)$表示：x 是人，用 $F(x)$表示：x 是食物，用 $A(x,y)$表示：x 对 y 过敏。则原命题可用谓词公式表示为

$$\exists x \exists y(P(x) \wedge F(y) \wedge A(x,y))$$

需注意的是，全称量词 $\forall$ 与存在量词 $\exists$ 不满足交换律。从而，

$$\forall x \exists yP(x,y) \neq \exists y \forall xP(x,y)$$

事实上，如果将 $P(x,y)$解释为：y 是 x 的母亲，则 $\forall x \exists yP(x,y)$的意思就是：任何人都有母亲，而 $\exists y \forall xP(x,y)$的意思则是：有一个人她是所有人的母亲。

5.1.5 基于谓词公式的形式演绎推理

上述从形式逻辑中抽象出来的推理规则也就是一些谓词公式的变换规则。这样，如果我们将待推理的前提命题表示成谓词公式，就可以利用推理规则把基于自然语言的逻辑推理转化为基于谓词公式的符号变换，即实现所说的形式推理。下面通过几个例子介绍基于谓词公式的形式演绎推理方法。

例 5-5 设有前提：

(1) 凡是大学生都学过计算机；

(2) 小王是大学生。

试问：小王学过计算机吗？

解 令 $S(x)$表示：x 是大学生；$M(x)$表示：x 学过计算机；a 表示：小王。则上面的两个命题可用谓词公式表示为

(1) $\forall x(S(x) \rightarrow M(x))$

(2) $S(a)$

下面遵循有关推理规则进行符号变换和推理：

(1) $\forall x(S(x) \rightarrow M(x))$ [前提]

(2) $S(a) \rightarrow M(a)$ [(1),US]

(3) $S(a)$ [前提]

(4) $M(a)$ [(2),(3),I_3]

得结果：$M(a)$，即“小王学过计算机”。

例 5-6 证明：$\neg P(a,b)$是 $\forall x \forall y(P(x,y) \rightarrow W(x,y))$和 $\neg W(a,b)$的逻辑结果。

证

(1) $\forall x \forall y(P(x,y) \rightarrow W(x,y))$[前提]

(2) $\forall y(P(a,y) \rightarrow W(a,y))$ [(1),US]

(3) $P(a,b) \rightarrow W(a,b)$ [(2),US]

(4) $\neg W(a,b)$ [前提]

(5) $\neg P(a,b)$　　[(3),(4),I_4]

例 5-7　证明：$\forall x(P(x)\rightarrow Q(x))\wedge\forall x(R(x)\rightarrow\neg Q(x))\Rightarrow\forall x(R(x)\rightarrow\neg P(x))$。

证

(1) $\forall x(P(x)\rightarrow Q(x))$　　[前提]

(2) $P(y)\rightarrow Q(y)$　　[(1),US]

(3) $\neg Q(y)\rightarrow\neg P(y)$　　[(2),E_{24}]

(4) $\forall x(R(x)\rightarrow\neg Q(x))$　　[前提]

(5) $R(y)\rightarrow\neg Q(y)$　　[(4),US]

(6) $R(y)\rightarrow\neg P(y)$　　[(3),(5),I_6]

(7) $\forall x(R(x)\rightarrow\neg P(x))$　　[(6),UG]

可以看出，上述的推理过程完全是一个符号变换过程。这种推理与人们用自然语言推理的思维过程十分相似，因而也被称为自然演绎推理。同时，这种推理实际上已几乎与谓词公式所表示的含义完全无关，而是一种纯形式的推理。这种形式推理是传统谓词逻辑中的基本推理方法。

由谓词公式的形式推理特点，人们自然想到将这种推理方法引入机器推理。但是，这种形式推理在机器中具体实施起来却存在许多困难。例如，推理规则太多、应用规则需要很强的模式识别能力、中间结论的指数递增等。所以，在机器推理中完全照搬谓词逻辑中的形式演绎推理方法，会有不少困难。于是，人们又开发了一些受限的自然演绎推理技术；或者另辟蹊径，发明了所谓的归结演绎推理技术。

5.2　归结演绎推理

5.2.1　子句与子句集

定义 5-12　原子谓词公式及其否定称为文字，若干个文字的一个析取式称为一个子句，由 r 个文字组成的子句叫 r-文字子句，1-文字子句叫单元子句，不含任何文字的子句称为空子句，记为□或 NIL。

例如，下面的析取式都是子句：

$$P\vee Q\vee\neg R$$

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

定义 5-13　对一个谓词公式 G，通过以下步骤所得的子句集合 S 称为 G 的子句集。

(1) 消去蕴涵词→和等值词←→。

可使用逻辑等价式：

① $P\rightarrow Q\Leftrightarrow\neg P\vee Q$

② $P\leftarrow\rightarrow Q\Leftrightarrow(\neg P\vee Q)\wedge(\neg Q\vee P)$

(2) 缩小否定词的作用范围，直到其仅作用于原子公式。

可使用逻辑等价式：

① $\neg(\neg P)\Leftrightarrow P$

② $\neg(P\wedge Q)\Leftrightarrow\neg P\vee\neg Q$

③ $\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$

④ $\neg \forall x P(x) \Leftrightarrow \exists x \neg P(x)$

⑤ $\neg \exists x P(x) \Leftrightarrow \forall x \neg P(x)$

(3) 适当改名,使量词间不含同名指导变元和约束变元。

(4) 消去存在量词。

消去存在量词时,同时还要进行变元替换。变元替换分两种情况:

① 若该存在量词在某些全称量词的辖域内,则用这些全称量词指导变元的一个函数代替该存在量词辖域中的相应约束变元,这样的函数被称为 Skolem 函数;

② 若该存在量词不在任何全称量词的辖域内,则用一个常量符号代替该存在量词辖域中的相应约束变元,这样的常量符号被称为 Skolem 常量。

(5) 消去所有全称量词。

(6) 化公式为合取范式。

可使用逻辑等价式:

① $P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \wedge (P \vee R)$

② $(P \wedge Q) \vee R \Leftrightarrow (P \vee R) \wedge (Q \vee R)$

(7) 适当改名,使子句间无同名变元。

(8) 消去合取词 $\wedge$,以子句为元素组成一个集合 S。

例 5-8 求下面谓词公式的子句集。

$$\forall x\{\forall y P(x,y) \rightarrow \neg \forall y[Q(x,y) \rightarrow R(x,y)]\}$$

解

由步骤(1)得 $\forall x\{\neg \forall y P(x,y) \vee \neg \forall y[\neg Q(x,y) \vee R(x,y)]\}$

由步骤(2)得 $\forall x\{\exists y \neg P(x,y) \vee \exists y[Q(x,y) \wedge \neg R(x,y)]\}$

由步骤(3)得 $\forall x\{\exists y \neg P(x,y) \vee \exists z[Q(x,z) \wedge \neg R(x,z)]\}$

由步骤(4)得 $\forall x\{\neg P(x,f(x)) \vee [Q(x,g(x)) \wedge \neg R(x,g(x))]\}$

由步骤(5)得 $\neg P(x,f(x)) \vee [Q(x,g(x)) \wedge \neg R(x,g(x))]$

由步骤(6)得 $[\neg P(x,f(x)) \vee Q(x,g(x))] \wedge [\neg P(x,f(x)) \vee \neg R(x,g(x))]$

由步骤(7)得 $[\neg P(x,f(x)) \vee Q(x,g(x))] \wedge [\neg P(y,f(y)) \vee \neg R(y,g(y))]$

由步骤(8)得 $\{\neg P(x,f(x)) \vee Q(x,g(x)), \neg P(y,f(y)) \vee \neg R(y,g(y))\}$

或写为

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

这就是原谓词公式的子句集。

需说明的是,在上述求子句集的过程中,当消去存在量词后,把所有全称量词都依次移到整个式子的最左边(或者先把所有量词都依次移到整个式子的最左边,再消去存在量词),再将右部的式子化为合取范式,这时所得的式子被称为原公式的 Skolem 标准型。例如,上例中谓词公式的 Skolem 标准型就是:

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

可以看出,消去 Skolem 标准型左部的全称量词和合取范式中的合取词,即得公式的

子句集。

例 5-9　设 $G=\exists x\forall y\forall z\exists u\forall v\exists w(P(x,y,z)\wedge Q(u,v,w))$，那么，用 a 代替 x，用 $f(y,z)$代替 u，用 $g(y,z,v)$代替 w，则得 G 的 Skolem 标准型

$$\forall y\forall z\forall v(P(a,y,z)\wedge\neg Q(f(y,z),v,g(y,z,v)))$$

进而得 G 的子句集为

$$\{P(a,x,y),\neg Q(f(u,v),w,g(u,v,w))\}$$

由此例还可看出，一个公式的子句集也可以通过先求前束范式，再求 Skolem 标准型而得到。

还需说明的是，引入 Skolem 函数，是由于存在量词在全称量词的辖域之内其约束变元的取值完全依赖于全称量词的取值。Skolem 函数就反映了这种依赖关系。但注意，Skolem 标准型与原公式一般并不等价，例如公式：

$$G=\exists xP(x)$$

它的 Skolem 标准型是

$$G'=P(a)$$

给出如下的一个解释 I

$$D=\{0,1\},\quad \frac{a}{0},\quad \frac{P(0)}{\mathrm{F}},\quad \frac{P(1)}{\mathrm{T}}$$

则在 I 下，$G=\mathrm{T}$，而 $G'=\mathrm{F}$。

由子句集的求法可以看出，一个子句集中的各子句间为合取关系，且每个个体变元都受全称量词的约束(假定公式中无自由变元，或将自由变元看作常元)。所以，一个公式的子句集也就是该公式的 Skolem 标准型的另一种表达形式。

有了子句集，就可以通过一个谓词公式的子句集来判断公式的不可满足性。

定理 5-4　谓词公式 G 不可满足当且仅当其子句集 S 不可满足。

定理 5-4 就把证明一个公式 G 的不可满足性，转化为证明其子句集 S 的不可满足性。

定义 5-14　子句集 S 是不可满足的，当且仅当其全部子句的合取式是不可满足的。

5.2.2　命题逻辑中的归结原理

归结演绎推理是基于一种称为归结原理(也称消解原理，principle of resolution)的推理规则的推理方法。归结原理是由鲁滨逊(J. A. Robinson)于 1965 年首先提出。它是谓词逻辑中一个相当有效的机械化推理方法。归结原理的出现，被认为是自动推理，特别是定理机器证明领域的重大突破。

定义 5-15　设 L 为一个文字，则称 L 与 $\neg L$ 为互补文字。

定义 5-16　设 C_1、C_2 是命题逻辑中的两个子句，C_1 中有文字 L_1、C_2 中有文字 L_2，且 L_1 与 L_2 互补，从 C_1、C_2 中分别删除 L_1、L_2，再将剩余部分析取起来，记构成的新子句为 C_{12}，则称 C_{12} 为 C_1、C_2 的归结式(或消解式)，C_1、C_2 称为其归结式的亲本子句，L_1、L_2 称为消解基。

例 5-10　设 $C_1=\neg P\vee Q\vee R$，$C_2=\neg Q\vee S$，于是 C_1、C_2 的归结式为

$$\neg P\vee R\vee S$$

定理 5-5 归结式是其亲本子句的逻辑结果。

证明 设 $C_1=L\vee C_1'$，$C_2=\neg L\vee C_2'$，C_1'、C_2'都是文字的析取式，则 C_1、C_2 的归结式为 $C_1'\vee C_2'$，因为

$$C_1=C_1'\vee L=\neg C_1'\rightarrow L,\quad C_2=\neg L\vee C_2'=L\rightarrow C_2'$$

所以

$$C_1\wedge C_2=(\neg C_1'\rightarrow L)\wedge(L\rightarrow C_2')\Rightarrow(\neg C_1'\rightarrow C_2')=C_1'\vee C_2'$$

证毕。

由该定理即得下面的推理规则

$$C_1\wedge C_2\Rightarrow(C_1-\{L_1\})\cup(C_2-\{L_2\})$$

其中 C_1、C_2 是两个子句，L_1、L_2 分别是 C_1、C_2 中的文字，且 L_1、L_2 互补。此规则就是**命题逻辑中的归结原理**。

例 5-11 用归结原理验证常用的推理规则假言推理(分离规则)：$P\wedge(P\rightarrow Q)\Rightarrow Q$ 和拒取式$(P\rightarrow Q)\wedge\neg Q\Rightarrow\neg P$。

证

$$P\wedge(P\rightarrow Q)=P\wedge(\neg P\vee Q)\Rightarrow Q$$

$$(P\rightarrow Q)\wedge\neg Q=(\neg P\vee Q)\wedge(\neg Q)\Rightarrow\neg P$$

类似地可以验证，其他推理规则也可以经消解原理推出。这就是说，用消解原理就可以代替其他所有的推理规则。再加上这个方法的推理步骤比较机械，这就为机器推理提供了方便。

由归结原理可知，如果两个互否的单元子句进行归结，则归结式为空子句，即 $L\wedge\neg L=\square$，而另一方面，$L\wedge\neg L=$F(假)。所以，空子句就是恒假子句，即

$$\square\Leftrightarrow\mathrm{F}$$

归结原理显然是一个很好的推理规则，但一般不使用它直接从前提推导结论，而是通过推导空子句来作间接证明。具体来讲，就是先求出要证的命题公式(谓词公式也一样)的否定式的子句集 S，然后对子句集 S(一次或多次)使用消解原理，若在某一步推出了空子句，即推出了矛盾，则说明子句集 S 是不可满足的，从而原否定式也是不可满足的，进而说明原公式是永真的。

为什么说，一旦推出了空子句就说明子句集 S 是不可满足的呢？这是因为空子句就是 F，推出了空子句就是推出了 F。但消解原理是推理规则，即正确的推理形式，那么由正确的推理形式推出了 F，则说明前提不真，即消解出空子句□的两个亲本子句中至少有一个为假。那么这两个亲本子句如果都是原子句集 S 中的子句，即说明原子句集 S 不可满足(因为子句集中各子句间为合取关系)。如果这两个亲本子句不是或不全是 S 中的子句，那么，它们必定是某次归结的结果，于是，用同样的道理再向上追溯，这样一定会推出原子句集 S 中至少有一个子句为假，从而说明 S 不可满足。

实际上，上述分析也可作为定理 5-5 的推论。

推论 设 C_1、C_2 是子句集 S 的两个子句，C_{12} 是它们的归结式，则：

(1) 若用 C_{12} 代替 C_1、C_2，得到新子句集 S_1，则由 S_1 的不可满足可推出原子句集 S 的不可满足。即

S_1 不可满足 $\Rightarrow S$ 不可满足

(2) 若把 C_{12} 加入 S 中，得到新子句集 S_2，则 S_2 与原 S 同不可满足。即

S_2 不可满足 $\Leftrightarrow S$ 不可满足

例 5-12　证明子句集$\{P \vee \neg Q, \neg P, Q\}$是不可满足的。

证

(1) $P \vee \neg Q$

(2) $\neg P$

(3) Q

(4) $\neg Q$　　由(1)(2)

(5) □　　由(3)(4)

例 5-13　用归结原理证明 R 是 P，$(P \wedge Q) \rightarrow R$，$(S \vee U) \rightarrow Q$，$U$ 的逻辑结果。

证　首先把前提条件化为子句形式，再把结论的非也化为子句，由所有子句得到子句集 $S=\{P, \neg P \vee \neg Q \vee R, \neg S \vee Q, \neg U \vee Q, U, \neg R\}$，然后对该子句集施行归结，归结过程用下面的归结演绎树表示(见图 5-1)。由于最后推出了空子句，所以子句集 S 不可满足，即命题公式 $P \wedge (\neg P \vee \neg Q \vee R) \wedge (\neg S \vee Q) \wedge (\neg U \vee Q) \wedge U \wedge \neg R$ 不可满足，从而得到 R 是题设前提的逻辑结果。

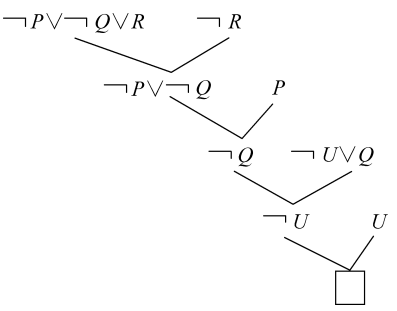

图 5-1　例 5-13 归结演绎树

5.2.3　替换与合一

在一阶谓词逻辑中应用消解原理，不像命题逻辑中那样简单，因为谓词逻辑中的子句含有个体变元，这就使寻找含互否文字的子句对的操作变得复杂。例如：

$$C_1 = P(x) \vee Q(x)$$

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

直接比较，似乎两者中不含互否文字，但如果用 a 替换 C_1 中的 x，则得到：

$$C_1' = P(a) \vee Q(a)$$

$$C_2' = \neg P(a) \vee R(y)$$

根据命题逻辑中的消解原理，得出 C_1' 和 C_2' 的消解式：

$$C_3' = Q(a) \vee R(y)$$

所以，要在谓词逻辑中应用消解原理，则一般需要对个体变元做适当的替换。

定义 5-17　一个替换(substitution)是形如$\{t_1/x_1, t_2/x_2, \cdots, t_n/x_n\}$的有限集合，其中 $t_1, t_2, \cdots, t_n$ 是项，称为替换的分子；$x_1, x_2, \cdots, x_n$ 是互不相同的个体变元，称为替换的分母；t_i 不同于 x_i，x_i 也不循环地出现在 t_j $(i, j=1,2,\cdots,n)$中；t_i/x_i 表示用 t_i 替换 x_i。若 $t_1, t_2, \cdots, t_n$ 都是不含变元的项(称为基项)时，该替换称为基替换；没有元素的替换称为空替换，记作 ε，它表示不做替换。

例如：$\{a/x, g(y)/y, f(g(b))/z\}$就是一个替换，而$\{g(y)/x, f(x)/y\}$则不是一个替换，因为 x 与 y 出现了循环替换。

下面将项、原子公式、文字、子句等统称为表达式，没有变元的表达式称为基表达式，出现在表达式 E 中的表达式称为 E 的子表达式。

定义 5-18 设 $\theta=\{t_1/x_1,\cdots,t_n/x_n\}$是一个替换，$E$ 是一个表达式，把对 E 施行替换 θ，即把 E 中出现的个体变元 x_j（$1\leqslant j\leqslant n$）都用 t_j 替换，记为 $E\theta$，所得的结果称为 E 在 θ 下的例(Instance)。

定义 5-19 设 $\theta=\{t_1/x_1,\cdots,t_n/x_n\}$，$\lambda=\{u_1/y_1,\cdots,u_m/y_m\}$是两个替换，则将集合$\{t_1\lambda/x_1,\cdots,t_n\lambda/x_n,u_1/y_1,\cdots,u_m/y_m\}$中凡符合下列条件的元素删除：

(1) $t_i\lambda/x_i \quad t_i\lambda=x_i$

(2) $u_i/y_i \quad y_i\in\{x_1,\cdots,x_n\}$

如此得到的集合仍然是一个替换，该替换称为 θ 与 λ 的复合或乘积，记为 $\theta\cdot\lambda$。

例 5-14 设 $\theta=\{f(y)/x, z/y\}$，$\lambda=\{a/x, b/y, y/z\}$，于是，

$$\{t_1\lambda/x_1, t_2\lambda/x_2, u_1/y_1, u_2/y_2, u_3/y_3\}=\{f(b)/x, y/y, a/x, b/y, y/z\}$$

从而

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

可以证明，替换的乘积满足结合律，即

$$(\theta\cdot\lambda)\cdot u=\theta\cdot(\lambda\cdot u)$$

定义 5-20 设 $S=\{F_1,F_2,\cdots,F_n\}$是一个原子谓词公式集，若存在一个替换 θ，可使 $F_1\theta=F_2\theta=\cdots=F_n\theta$，则称 θ 为 S 的一个合一(unifier)，称 S 为可合一的。

定义 5-21 设 σ 是原子公式集 S 的一个合一，如果对 S 的任何一个合一 θ，都存在一个替换 λ，使得

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

则称 σ 为 S 的最一般合一(most general unifier，MGU)。

例 5-15 设 $S=[P(u,y,g(y)),P(x,f(u),z)]$，$S$ 有一个最一般合一

$$\sigma=\{u/x, f(u)/y, g(f(u))/z\}$$

对 S 的任一合一，例如

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

存在一个替换

$$\lambda=\{a/u\}$$

使得

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

可以看出，如果能找到一个公式集的合一，特别是最一般合一，则可使互否文字的形式结构完全一致起来，进而达到消解的目的。如何求一个公式集的最一般合一？有一个算法，可以求任何可合一公式集的最一般合一。为了介绍这个算法，先引入差异集的概念。

定义 5-22 设 S 是一个非空的具有相同谓词名的原子公式集，从 S 中各公式的左边第一个项开始，同时向右比较，直到发现第一个不都相同的项为止，用这些项的差异部分

组成一个集合，这个集合就是原公式集 S 的一个差异集。

例 5-16　设 $S=\{P(x,y,z),P(x,f(a),h(b))\}$，则不难看出，$S$ 有两个差异集

$$D_1=\{y,f(a)\}$$

$$D_2=\{z,h(b)\}$$

设 S 为一非空有限具有相同谓词名的原子谓词公式集，下面给出求其最一般合一的算法。

合一算法（Unification algorithm）

(1) 置 $k=0$，$S_k=S$，$\sigma_k=\varepsilon$。

(2) 若 S_k 只含有一个谓词公式，则算法停止，σ_k 就是要求的最一般合一。

(3) 求 S_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 $S_{k+1}=S_k\{t_k/x_k\}$，$\sigma_{k+1}=\sigma_k\cdot\{t_k/x_k\}$，$k=k+1$，然后转步骤(2)。

(5) 算法停止，S 的最一般合一不存在。

5.2.4　谓词逻辑中的归结原理

定义 5-23　设 C_1、C_2 是两个无相同变元的子句，L_1、L_2 分别是 C_1、C_2 中的两个文字，如果 L_1 和 L_2 有最一般合一 σ，则子句 $(C_1\sigma-\{L_1\sigma\})\cup(C_2\sigma-\{L_2\sigma\})$ 称作 C_1 和 C_2 的二元归结式（或二元消解式），C_1 和 C_2 称作归结式的亲本子句，L_1 和 L_2 称作消解文字。

例 5-17　设 $C_1=P(x)\vee Q(x)$，$C_2=\neg P(a)\vee R(y)$，求 C_1、C_2 的归结式。

解　取 $L_1=P(x)$，$L_2=\neg P(a)$，则 L_1 与 $\neg L_2$ 的最一般合一 $\sigma=\{a/x\}$，于是，

$$\begin{aligned}&(C_1\sigma-\{L_1\sigma\})\cup(C_2\sigma-\{L_2\sigma\})\\&=(\{P(a),Q(a)\}-\{P(a)\})\cup(\{\neg P(a),R(y)\}-\{\neg P(a)\})\\&=\{Q(a),R(y)\}\\&=Q(a)\vee R(y)\end{aligned}$$

所以，$Q(a)\vee R(y)$ 是 C_1 和 C_2 的二元归结式。

例 5-18　设 $C_1=P(x,y)\vee\neg Q(a)$，$C_2=Q(x)\vee R(y)$，求 C_1、C_2 的归结式。

解　由于 C_1、C_2 中都含有变元 x、y，所以需先对其中一个进行改名，方可归结（归结过程是显然的，故从略）。还需说明的是，如果在参加归结的子句内部含有可合一的文字，则在进行归结之前，也应对这些文字进行合一，从而使子句达到最简。例如，设有两个子句：

$$C_1=P(x)\vee P(f(a))\vee Q(x)$$

$$C_2=\neg P(y)\vee R(b)$$

可见，在 C_1 中有可合一的文字 $P(x)$ 与 $P(f(a))$，那么，取替换 $\theta=\{f(a)/x\}$[这个替换也就是 $P(x)$ 和 $P(f(a))$ 的最一般合一]，则得

$$C_1\theta=P(f(a))\vee Q(f(a))$$

现在再用 $C_1\theta$ 与 C_2 进行归结，从而得到 C_1 与 C_2 的归结式

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

定义 5-24 如果子句 C 中，两个或两个以上的文字有一个最一般合一 σ，则 $C\sigma$ 称为 C 的因子，如果 $C\sigma$ 是单元子句，则 $C\sigma$ 称为 C 的单因子。

例 5-19 设 $C=P(x) \vee P(f(y)) \vee \neg Q(x)$，令 $\sigma=\{f(y)/x\}$，于是 $C\sigma=P(f(y)) \vee \neg Q(f(y))$ 是 C 的因子。

定义 5-25 子句 C_1、C_2 的消解式，是下列二元消解式之一：

(1) C_1 和 C_2 的二元消解式；

(2) C_1 和 C_2 的因子的二元消解式；

(3) C_1 的因子和 C_2 的二元消解式；

(4) C_1 的因子和 C_2 的因子的二元消解式。

定理 5-6 谓词逻辑中的消解式是它的亲本子句的逻辑结果。

由此定理即得谓词逻辑中的推理规则

$$C_1 \wedge C_2 \Rightarrow (C_1\sigma - \{L_1\sigma\}) \cup (C_2\sigma - \{L_2\sigma\})$$

其中，C_1、C_2 是两个无相同变元的子句，L_1、L_2 分别是 C_1、C_2 中的文字，σ 为 L_1 与 $\neg L_2$ 的最一般合一。此规则称为**谓词逻辑中的消解原理**(或**归结原理**)。

例 5-20 求证 G 是 A_1 和 A_2 的逻辑结果。

A_1：$\forall x(P(x) \rightarrow (Q(x) \wedge R(x)))$

A_2：$\exists x(P(x) \wedge S(x))$

G：$\exists x(S(x) \wedge R(x))$

证 用反证法，即证明 $A_1 \wedge A_2 \wedge \neg G$ 不可满足。首先求得子句集 S：

(1) $\neg P(x) \vee Q(x)$ } (A_1)

(2) $\neg P(y) \vee R(y)$ } (A_1)

(3) $P(a)$ } (A_2)

(4) $S(a)$ } (A_2)

(5) $\neg S(z) \vee \neg R(z)$ $(\neg G)$

} S

然后应用消解原理，得

(6) $R(a)$ $[(2),(3),\sigma_1=\{a/y\}]$

(7) $\neg R(a)$ $[(4),(5),\sigma_2=\{a/z\}]$

(8) $\square$ $[(6),(7)]$

所以，S 是不可满足的，从而 G 是 A_1 和 A_2 的逻辑结果。

例 5-21 设已知：

(1) 能阅读者是识字的。

(2) 海豚不识字。

(3) 有些海豚是很聪明的。

试证明：有些聪明者并不能阅读。

证 首先，定义如下谓词。

$R(x)$：x 能阅读。

$L(x)$：x 识字。

$I(x)$：x 是聪明的。

$D(x)$：x 是海豚。

然后把上述各语句翻译为谓词公式：

(1) $\forall x(R(x)\rightarrow L(x))$

(2) $\forall x(D(x)\rightarrow \neg L(x))$　} 已知条件

(3) $\exists x(D(x)\wedge I(x))$

(4) $\exists x(I(x)\wedge \neg R(x))$　需证结论

求题设与结论否定的子句集，得

(1) $\neg R(x)\vee L(x)$

(2) $\neg D(y)\vee \neg L(y)$

(3) $D(a)$

(4) $I(a)$

(5) $\neg I(z)\vee R(z)$

归结得

(6) $R(a)$　　由(5)(4)，$\{a/z\}$

(7) $L(a)$　　由(6)(1)，$\{a/x\}$

(8) $\neg D(a)$　　由(7)(2)，$\{a/y\}$

(9) □　　由(8)(3)

这个归结过程的演绎树如图 5-2 所示。

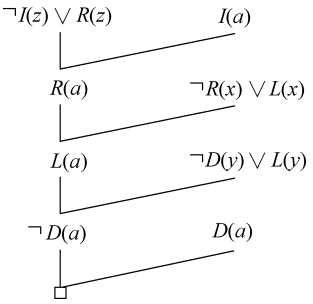

图 5-2　例 5-21 的归结演绎树

由以上例子可以看出，谓词逻辑中的消解原理也可以代替其他推理规则。

上面通过推导空子句证明了子句集的不可满足性，因此，存在问题：对于任一不可满足的子句集，是否都能通过归结原理推出空子句呢？回答是肯定的。

定理 5-7　(归结原理的完备性定理)如果子句集 S 是不可满足的，那么必存在一个由 S 推出空子句□的消解序列(该定理的证明要用到 Herbrand 定理，故从略)。

5.3　应用归结原理求取问题答案

归结原理除了能用于对已知结果的证明外，还能用于对未知结果的求解，即能求出问题的答案来。请看下例。

例 5-22　已知：

(1) 如果 x 和 y 是同班同学，则 x 的老师也是 y 的老师。

(2) 王先生是小李的老师。

(3) 小李和小张是同班同学。

问：小张的老师是谁？

解 设谓词 $T(x,y)$表示 x 是 y 的老师，$C(x,y)$表示 x 与 y 是同班同学，则已知可表示成如下的谓词公式

F_1：$\forall x\ \forall y\ \forall z(C(x,y)\wedge T(z,x)\rightarrow T(z,y))$

F_2：$T(\text{Wang},\text{Li})$

F_3：$C(\text{Li},\text{Zhang})$

为了得到问题的答案，先证明小张的老师是存在的，即证明公式

G：$\exists x T(x,\text{Zhang})$

于是，求 $F_1\wedge F_2\wedge F_3\wedge\rightarrow G$ 的子句集如下：

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

(2) $T(\text{Wang},\text{Li})$

(3) $C(\text{Li},\text{Zhang})$

(4) $\neg T(u,\text{Zhang})$

归结演绎，得

(5) $\neg C(\text{Li},y)\vee T(\text{Wang},y)$　　由(1)(2)，$\{\text{Wang}/z,\text{Li}/x\}$

(6) $\neg C(\text{Li},\text{Zhang})$　　由(4)(5)，$\{\text{Wang}/u,\text{Zhang}/y\}$

(7) $\square$　　由(3)(6)

这说明，小张的老师确实是存在的。那么，为了找到这位老师，我们给原来的求证谓词的子句再增加一个谓词 ANS(u)。于是，得到

(4)′ $\neg T(u,\text{Zhang})\vee\text{ANS}(u)$

现在，用(4)′代替(4)，重新进行归结，则得

(5)′ $\neg C(\text{Li},y)\vee T(\text{Wang},y)$　　由(1)(2)

(6)′ $\neg C(\text{Li},\text{Zhang})\vee\text{ANS}(\text{Wang})$　　由(4)′(5)′

(7)′ $\text{ANS}(\text{Wang})$　　由(3)(6)′

可以看出，归结到这一步，求证的目标谓词已被消去，即求证已成功，但还留下了谓词 ANS(Wang)。由于该谓词中原先的变元与目标谓词 $T(u,\text{Zhang})$中的一致，所以，其中的 Wang 也就是变元 u 的值。这样，就求得了小张的老师，也就是王老师。

上例虽然是一个很简单的问题，但它给了我们一个利用归结原理求取问题答案的方法，那就是：先为待求解的问题找一个合适的求证目标谓词；再给增配(以析取形式)一个辅助谓词，且该辅助谓词中的变元必须与对应目标谓词中的变元完全一致；然后进行归结，当某一步的归结式刚好只剩下辅助谓词时，辅助谓词中原变元位置上的项(一般是常量)就是所求的问题答案。

需说明的是，辅助谓词(如此题中的 ANS)是一个形式谓词，其作用仅是提取问题的答案，因而也可取其他谓词名。有些文献中就用需求证的目标谓词。如对上例，就取 $T(u,\text{Zhang})$为辅助谓词。

例 5-23 设有如下关系：

(1) 如果 x 是 y 的父亲，y 又是 z 的父亲，则 x 是 z 的祖父。

(2) 老李是大李的父亲。

(3) 大李是小李的父亲。

问：上述人员中谁和谁是祖孙关系？

解　先把上述前提中的3个命题符号化为谓词公式：

F_1：$\forall x\ \forall y\ \forall z(F(x,y)\wedge F(y,z)\rightarrow G(x,z))$

F_2：$F(\text{Lao},\text{Da})$

F_3：$F(\text{Da},\text{Xiao})$

并求其子句集如下：

(1) $\neg F(x,y)\vee\neg F(y,z)\vee G(x,z)$

(2) $F(\text{Lao},\text{Da})$

(3) $F(\text{Da},\text{Xiao})$

设求证的公式为

G：$\exists x\ \exists y G(x,y)$　（即存在 x 和 y，x 是 y 的祖父）

把其否定化为子句形式再析取一个辅助谓词 $\text{GA}(x,y)$，得

(4) $\neg G(u,v)\vee\text{GA}(u,v)$

对(1)～(4)进行归结，得

(5) $\neg F(\text{Da},z)\vee G(\text{Lao},z)$　　由(1)(2)，$\{\text{Lao}/x,\text{Da}/y\}$

(6) $G(\text{Lao},\text{Xiao})$　　由(3)(5)，$\{\text{Xiao}/z\}$

(7) $\text{GA}(\text{Lao},\text{Xiao})$　　由(4)(6)，$\{\text{Lao}/u,\text{Xiao}/v\}$

所以，上述人员中，老李是小李的祖父。

5.4　归结策略*

5.4.1　问题的提出

前面介绍了归结原理及其应用，但那些归结推理都是用人工实现的。而人们研究归结推理的目的主要是为了更好地实现机器推理，或者说自动推理。那么，现在就存在问题：归结原理如何在机器上实现？

把归结原理在机器上实现，就意味着用算法描述归结原理，然后编制程序，在计算机上运行。下面给出一个实现归结原理的一般性算法：

---

(1) 将子句集 S 置入 CLAUSES 表。

(2) 若空子句 NIL 在 CLAUSES 中，则归结成功，结束。

(3) 若 CLAUSES 表中存在可归结的子句对，则归结之，并将归结式并入 CLAUSES 表，转步骤(2)。

(4) 归结失败，退出。

可以看出，这个算法并不复杂，但问题是在其步骤(3)中应该以什么样的次序从已给的子句集 S 出发寻找可归结的子句对进行归结呢？

一种简单而直接的想法就是逐个考察 CLAUSES 表中的子句，穷举式地进行归结。可采用这样的具体做法：第一轮归结先让 CLAUSES 表(即原子句集 S)中的子句两两见面进行归结，将产生的归结式集合记为 S_1，再将 S_1 并入 CLAUSES 得 $\text{CLAUSES}=S\cup S_1$；下

一轮归结时，又让新的CLAUSES即$S \cup S_1$与S_1中的子句互相见面进行归结，并把产生的归结式集合记为S_2，再将S_2并入CLAUSES；在再一轮归结时，又让$S \cup S_1 \cup S_2$与S_2中的子句进行归结……如此进行，直到某一个S_k中出现空子句□为止。下面举例说明。

例 5-24 设有如下的子句集S，用上述的穷举算法归结如下。

S：

(1) $P \vee Q$
(2) $\neg P \vee Q$
(3) $P \vee \neg Q$
(4) $\neg P \vee \neg Q$

S_1：

(5) Q [(1),(2)]
(6) P [(1),(3)]
(7) $Q \vee \neg Q$ [(1),(4)]
(8) $P \vee \neg P$ [(1),(4)]
(9) $Q \vee \neg Q$ [(2),(3)]
(10) $P \vee \neg P$ [(2),(3)]
(11) $\neg P$ [(2),(4)]
(12) $\neg Q$ [(3),(4)]

S_2：

(13) $P \vee Q$ [(1),(7)]
(14) $P \vee Q$ [(1),(8)]
(15) $P \vee Q$ [(1),(9)]
(16) $P \vee Q$ [(1),(10)]
(17) Q [(1),(11)]
(18) P [(1),(12)]
(19) Q [(2),(6)]
(20) $\neg P \vee Q$ [(2),(7)]
(21) $\neg P \vee Q$ [(2),(8)]
(22) $\neg P \vee Q$ [(2),(9)]
(23) $\neg P \vee Q$ [(2),(10)]
(24) $\neg P$ [(2),(12)]
(25) P [(3),(5)]
(26) $P \vee \neg Q$ [(3),(7)]
(27) $P \vee \neg Q$ [(3),(8)]
(28) $P \vee \neg Q$ [(3),(9)]
(29) $P \vee \neg Q$ [(3),(10)]
(30) $\neg Q$ [(3),(11)]

(31) $\neg P$　　[(4),(5)]
(32) $\neg Q$　　[(4),(6)]
(33) $\neg P \vee \neg Q$　　[(4),(7)]
(34) $\neg P \vee \neg Q$　　[(4),(8)]
(35) $\neg P \vee \neg Q$　　[(4),(9)]
(36) $\neg P \vee \neg Q$　　[(4),(10)]
(37) Q　　[(5),(7)]
(38) Q　　[(5),(9)]
(39) □　　[(5),(12)]

可以看出,这个归结方法无任何技巧可言,只是一味地穷举式归结。因而对于如此简单的问题,计算机推导了35步,即产生35个归结式,才导出了空子句。那么,对于一个规模较大的实际问题,其时空开销就可想而知了。事实上,这种方法一般会产生许多无用的子句。这样,随着归结的进行,CLAUSES表将会越来越庞大,以至于机器不能容纳。同时,归结的时间消耗也是一个严重问题。

那么,怎样归结才能高效地推出空子句NIL呢?研究表明,要提高归结的效率,就必须运用一定的技巧,即所谓归结策略。例如,对于例5-24中的问题,若运用一定的策略,则仅用3步就可以解决问题。

事实上,归结反演的过程,就是一个在子句空间中搜索(空子句)的过程。因此,要用归结原理实现机器推理,一个重要的问题就是要赋予机器一定的搜索策略,即归结策略。这就是说,要让计算机进行归结演绎推理,仅有归结原理还不够,还必须研究归结策略。归结策略有哪些?下面将介绍几种。

需说明的是,上述的归结方法实际上是一种广度优先的按层次进行归结的方法。所以,一般也把它说成是一种归结策略,称之为广度优先策略,也可称为水平浸透法。

5.4.2　常用的归结策略

1. 删除策略

定义5-26　设C_1、C_2是两个子句,若存在替换θ,使得$C_1\theta \subseteq C_2$,则称子句C_1类含C_2。

例如:

$P(x)$类含$P(a) \vee Q(y)$　(只需取$\theta=\{a/x\}$)

$Q(y)$类含$P(x) \vee Q(y)$　($\theta=\varepsilon$)

$P(x)$类含$P(x)$,　$P(x)$类含$P(a)$,　P类含P,　P类含$P \vee R$

$P(a,x) \vee P(y,b)$类含$P(a,b)$　(取$\theta=\{b/x,a/y\}$)

删除策略:

在归结过程中可随时删除以下子句:

(1) 含有纯文字的子句。

(2) 含有永真式的子句。

(3) 被子句集中别的子句类含的子句。

所谓纯文字，是指那些在子句集中无补的文字。例如下面的子句集

$$\{P(x) \vee Q(x,y) \vee R(x),\quad \neg P(a) \vee Q(u,v),\quad \neg Q(b,z),\quad \neg P(w)\}$$

其中的文字 $R(x)$ 就是一个纯文字。

删除含有纯文字的子句，是因为在归结时纯文字永远不会被消去，因而用包含它的子句进行归结不可能得到空子句。删除永真式是因为永真式对子句集的不可满足性不起任何作用。删除被类包含的子句是因为它被类含它的子句所逻辑蕴涵，故它已是多余的。

例 5-25 对例 5-24 中的子句集使用删除策略。可以看出，这时原归结过程中产生的有些归结式是永真式（如(7)(8)(9)(10)），有些被前面已有的子句所类含（如(17)(18)等，重复出现可认为是一种类含），因此，它们可被立即删除。这样就导致它们的后裔将不可能出现。其归结步骤可简化为：

(1) $P \vee Q$

(2) $\neg P \vee Q$

(3) $P \vee \neg Q$

(4) $\neg P \vee \neg Q$

(5) Q [(1),(2)]

(6) P [(1),(3)]

(7) $\neg P$ [(2),(4)]

(8) $\neg Q$ [(3),(4)]

(9) $\square$ [(5),(8)]

其实，上述归结还可以进一步简化为：

(5) Q [(1),(2)]

(6) $\neg Q$ [(3),(4)]

(7) $\square$ [(5),(6)]

这是因为，(5)出现后，由于它就类含了(1)(2)，所以可将(1)和(2)删除。同理，当(6)出现时，可将(3)(4)删除。这样，下面也只能是(5)(6)归结了。

例 5-26 对下面的子句集 S，用宽度优先策略与删除策略相结合的方法进行消解。

S：(1) $P(x) \vee Q(x) \vee \neg R(x)$

(2) $\neg Q(a)$

(3) $\neg R(a) \vee Q(a)$

(4) $P(y)$

(5) $\neg P(z) \vee R(z)$

可以看出，(4)类含了(1)，所以先将(1)删除。由剩下的 4 个子句归结得：

S_1：(6) $\neg R(a)$ [(2),(3)]

(7) $\neg P(a) \vee Q(a)$ [(3),(5),$\{a/z\}$]

(8) $R(z)$ [(4),(5),$\{z/y\}$]

(6)出现后(3)可被删除，所以，第二轮归结在(2)、(4)、(5)、(6)、(7)、(8)间进行。其中(2)、(4)、(5)间的归结不必再重做，于是得

S_2：(9) $\neg P(a)$ [(2),(7)]

(10) $Q(a)$　　[(4),(7),$\{a/y\}$]
(11) $\neg P(a)$　　[(5),(6),$\{a/z\}$]
(12) □　　[(6),(8),$\{a/z\}$]

删除策略有如下特点：

(1) 删除策略的思想是及早删除无用子句，以避免无效归结，缩小搜索规模；尽量使归结式朝“小”(即元数少)方向发展，从而尽快导出空子句。

(2) 删除策略是完备的。即对于不可满足的子句集，使用删除策略进行归结，最终必导出空子句□。

定义 5-27　一个归结策略是完备的，如果对于不可满足的子句集，使用该策略进行归结，最终必导出空子句□。

2. 支持集策略

支持集策略：每次归结时，两个亲本子句中至少要有一个是目标公式否定的子句或其后裔。这里的目标公式否定的子句集即为支持集。

例 5-27　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，子句$\neg I(x)\vee R(x)$是目标公式否定的子句。

用支持集策略归结如下：

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) $R(a)$　　由(1)(2),$\{a/x\}$
(6) $\neg I(x)\vee \neg L(x)$　　由(1)(3),$\{x/y\}$

S_2：(7) $\neg L(a)$　　由(5)(3),$\{a/y\}$
(8) $\neg L(a)$　　由(6)(2),$\{a/x\}$
(9) $\neg I(a)$　　由(6)(4),$\{a/x\}$
(10) □　　由(7)(4)

支持集策略有如下特点：

(1) 这种策略的思想是尽量避免在可满足的子句集中做归结，因为从中导不出空子句。而求证公式的前提通常是一致的，所以，支持集策略要求在归结时从目标公式否定的子句出发进行归结。所以，支持集策略实际是一种目标制导的反向推理。

(2) 支持集策略是完备的。

3. 线性归结策略

线性归结策略：在归结过程中，除第一次归结可都用给定的子句集 S 中的子句外，其后的各次归结至少要有一个亲本子句是上次归结的结果。

例 5-28 对例 5-27 中的子句集，用线性归结策略归结。

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

(2) $I(a)$

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

(4) $L(a)$

(5) $R(a)$ 由(1)(2)，$\{a/x\}$

(6) $\neg L(a)$ 由(5)(3)，$\{a/y\}$

(7) $\square$ 由(6)(4)

线性归结策略的特点是：不仅它本身是完备的，高效的，而且还与许多别的策略兼容。例如在线性归结中可同时采用支持集策略或输入策略。

4. 输入归结策略

输入归结策略：每次参加归结的两个亲本子句，必须至少有一个是初始子句集 S 中的子句。

输入归结策略的特点是：

- 输入归结策略实际是一种自底向上的推理，它有相当高的效率。
- 输入归结是不完备的。例如子句集

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

是不可满足的。用输入归结都不能导出空子句，因为最后导出□的子句必定都是单文字子句，它们不可能在 S 中。

输入归结往往同线性归结配合使用，组成所谓线性输入归结策略。其进一步还可以与支持集策略结合。

5. 单元归结策略

单元归结策略：在归结过程中，每次参加归结的两个亲本子句中必须至少有一个是单元子句。

单元归结策略的特点：

- 单元归结的思想是，用单元子句归结可使归结式含有较少的文字，因而有利于尽快逼近空子句。
- 单元归结也是一种效率高但不完备的策略。

单元归结和输入归结虽都不完备，但应用它们可以证明相当广泛的一类定理，因此，它们不失为好的归结策略。另外，理论研究还表明：对不可满足的子句集 S，可用单元归结导出空子句当且仅当可用输入归结导出空子句。

单纯使用单元归结有时可能无法归结(当无单元子句时)。因此，一般是将单元归结的条件放宽，变为优先对单元子句进行归结。这种策略称为单元优先策略。

6. 祖先过滤形策略

祖先过滤形策略：参加归结的两个子句，要么至少有一个是初始子句集中的子句；

要么一个是另一个的祖先(或者说一个是另一个的后裔)。

5.4.3　归结策略的类型

除了上面介绍的一些常用的归结策略,人们还提出了许多别的策略,如锁归结、语义归结、加权策略、模型策略等。

锁归结的思想是:用数字1、2、3……对各子句中的文字进行编号,使互不相同的文字或相同文字的不同出现具有不同的编号,这种编号称为文字的锁,如 $1P \vee 3Q$ 和 $5P \vee 9P$ 中的1、3、5、9都是锁。这样,归结就可以用锁来控制。具体做法是:每次归结,参加归结的文字必须分别是所在子句中编号最小者。例如,有子句 $1P \vee 2Q$ 和 $3 \neg P \vee 4 \neg Q$,则只能对 P、$\neg P$ 作归结。

语义归结的基本思想是将子句集 S 中的子句分成两组,只考虑组间子句的归结。加权策略是对子句或其中的项赋予相应的权值,以反映子句或项在实际问题中的某种程度,这样,归结就可以用权值来控制,如给出某种顺序或限制。

归结策略很多,归纳起来,大致可以分为3类:

(1) 简化性策略。

这种策略的思想是尽量简化子句和子句集,以减少和避免无效归结。如删除策略就是简化策略。然而,简化策略在使用时,也要付出一定的开销,如要不断地做包含检验或真值计算。这又是它的缺点。

(2) 限制性策略。

前面所介绍的策略多数都是限制性策略。如支持集策略、线性策略、输入策略、单元策略、祖先过滤策略、语义归结等。限制性策略的思想是尽量缩小搜索范围,以提高搜索效率。

(3) 有序性策略。

有序性策略的思想是给子句安排一定的顺序,以便能尽快地推出空子句。单元优先策略、加权策略以及锁归结等都是有序性归结策略。

有了归结策略后,从本节开始所给的归结反演一般算法可改为:

---

(1) 将子句集 S 置入 CLAUSES 表。

(2) 若空子句 NIL 在 CLAUSES 中,则归结成功,结束。

(3) 按某种策略在 CLAUSES 表中寻找可归结的子句对,若存在则归结之,并将归结式并入 CLAUSES 表,转步骤(2)。

(4) 归结失败,退出。

5.5　归结反演程序举例*

下面给出一个可用于命题逻辑归结反演的 PROLOG 示例程序。

```
prove(F,S): - union(F,S,SY),proof(SY).
union([],Y,Y).
union([X|XR],Y,Z): - member(X,Y),!,union(XR,Y,Z).
```

```
union([X|XR],Y,[X|ZR]): - union(XR,Y,ZR).
proof([SH|ST]): - resolution(SH,ST,[]),!.
proof([SH|ST]): - resolution(SH,ST,NF),proof([NF,SH|ST]).
resolution(SH,[STH|ST],NF): - resolve(SH,STH,NF1),NF1 = SH,!,resolution(SH,ST,NF).
resolution(SH,[STH|ST],NF): - resolve(SH,STH,NF),print(SH,STH,NF).
resolve([],_,[]): - !.
resolve([F|FR],SF,FR): - not(F = no),invert(F,IF),IF = SF,!.
resolve([F|FR],SF,NF): - not(F = no),invert(F,IF),member(IF,SF),!,
                        pack(F,FR,SF,NF).
resolve([F|FR],SF,NF): - not(F = no),!,resolve(FR,SF,NF1),pack([[],[F],[NF1],NF).
resolve(F,SF,[]): - invert(F,IF),IF = SF,!.
resolve(F,SF,NF): - invert(F,IF),member(IF,SF),!,pack(F,[],SF,NF).
resolve(F,_,F).
invert(X,[no,X]): - atom(X).
invert([no,X],X): - atom(X).
member(X,[X|_]): - !.
member(X,[_|Y]): - member(X,Y).
pack(A,X,Y,Z): - combine(A,X,Y,[Z|[]]),!.
pack(A,X,Y,Z): - combine(A,X,Y,Z).
combine(A,X,Y,Z): - union(X,Y,Z1),delete(A,Z1,Z2),invert(A,IA),delete(IA,Z2,Z).
delete(_,[],[]).
delete(E,[E|ER],R): - !,delete(E,ER,R).
delete(E,[X|XR],[X|R]): - delete(E,XR,R).
print(F,S,R): - write(F),write(´,´),write(S),write("?"),write(R),nl.
```

该程序把子句用表来表示。例如：子句 $\neg P \vee Q$，表示为[[not,p],q]。子句集用子句表表示。例如：子句集$\{\neg P \vee Q, R \vee S, U\}$，则表示为

[[[not,p],q],[r,s],u]

该程序的目标子句是 prove(F,S)。其中，S 为前提，F 为要证明的结论的否定。程序运行时，谓词 union(F,S,FS)首先把待证结论的否定子句 F 与前提子句 S 合并为 FS。接着，谓词 proof(FS)对子句集 FS 进行归结反演，试图推出空子句[]。proof 又调用谓词 resolution 进行归结。proof 的第一个子句是归结反演的终结条件；第二个子句是归结反演的递归操作。

resolution(SH,ST,NF)谓词实现具体的归结操作。其中 SH 是从子句集 FS 中分离出的一个子句，它作为一个双亲子句；ST 为去掉 SH 后的子句集；NF 是 SH 与 ST 中子句产生的归结式。

resolution 的第一个子句处理 ST 子句集中的第一个子句 STH 不能与 SH 归结的情况，将引起 resolution 的递归操作。在这里，resolve 子句把 SH 放入 NF，于是 resolution 子句根据 NF1＝SH 知道 STH 不能与 SH 组成互补对，便对剩下部分(去掉 STH 以后的表尾)进行递归处理。resolution 的第二个子句处理 ST 子句集中的第一个子句 STH 能与 SH 归结的情况，这时将 SH 与 STH 的归结式放入 NF。接着由 print(SH,STH,NF)输出这一步的归结推理。

resolve(SH,STH,NF)谓词的作用是检查 SH 和 STH 是否为可归结的双亲子句。

resolve 共有 7 个子句。第一个子句是终止条件。第二至第四个子句处理 SH 为非单项析取式的情况，它们对 SH 从左到右依次查看每一个单项析取式，看是否在 STH 中存在它的否定，若存在则进行归结，若不存在则进行递归处理。其中，第二个子句处理 STH 为单项析取式的情况，第三个子句处理 STH 为非单项析取式的情况，第四个子句处理 STH 中不存在 SH 中的单项析取式的否定的情况。第五、第六个子句处理 SH 为单项析取式的情况，其中第五个子句处理 STH 也为单项析取式的情况，第六个子句处理 STH 为非单项析取式的情况。这两个子句直接判断 SH 能否与 STH 归结，子句 5 归结结果为[]，即产生矛盾；子句 6 产生归结式 NF。第七个子句处理 SH 不能与 STH 归结的情况。

pack(F,FR,SF,NF)谓词的功能是将子句[F|FR]和 SF 归结，产生归结式 NF，其中 F 是引起归结的项。pack 的第一个子句将删除归结式中可能多余的括号[]。如由[p,q]和[not,p]产生的归结式[q]便被改为 q。pack 的第二个子句调用 combine 来删除形如 L、$\neg L$ 的互补对，并确保在归结式中没有重复的元素。

invert(F,IF)谓词的功能是将 F 取否定放在 IF 中。

现在用这个程序证明 $\neg P \vee \neg U$ 是 $Q \vee \neg P$，$R \vee \neg Q$，$S \vee \neg R$ 和 $\neg U \vee \neg S$ 的逻辑结论。则有目标

```
?-prove([p,u],[[q,[no,p]],[r,[no,q]],[s,[no,r]],[[no,u],[no,s]]]).
```

对此目标，程序的运行结果为：

```
p,[q,[not,p]]⇒q
q,[r,[not,q]]⇒r
r,[s,[not,r]]⇒s
s,[[not,u],[not,s]⇒[not,u]
[not,u],u⇒[]
yes
```

5.6　Horn 子句逻辑与逻辑程序设计语言*

5.6.1　子句的蕴涵表示形式

我们知道，原子公式及其否定称为文字，现在把前者称为正文字，后者称为负文字。例如子句 $P(x) \vee \neg Q(x,y)$ 中 $P(x)$ 为正文字，$\neg Q(x,y)$ 为负文字。子句是若干文字的析取，析取词又满足交换律，所以对于任一个子句，总可以将其表示成如下形式

$$\neg Q_1 \vee \cdots \vee \neg Q_n \vee P_1 \vee \cdots \vee P_m \tag{5-1}$$

其中，P_i，$\neg Q_j$ 皆为文字。可以看出，式(5-1)进一步可变形为

$$Q_1 \wedge Q_2 \wedge \cdots \wedge Q_n \rightarrow P_1 \vee P_2 \vee \cdots \vee P_m \tag{5-2}$$

式(5-2)为一个蕴涵式，如果约定蕴涵式前件的文字之间恒为合取关系，而蕴涵式后件的文字恒为析取关系，那么式(5-2)又可以改写为

$$Q_1, Q_2, \cdots, Q_n \rightarrow P_1, P_2, \cdots, P_m \tag{5-3}$$

由于技术上的原因，又将式(5-3)改写为

$$P_1,P_2,\cdots,P_m \leftarrow Q_1,Q_2,\cdots,Q_n \tag{5-4}$$

作为特殊情形，当 $m=0$ 时式(5-4)变为

$$\leftarrow Q_1,Q_2,\cdots,Q_n \tag{5-4′}$$

它相当于 $\neg(Q_1 \wedge Q_2 \wedge \cdots \wedge Q_n)$；当 $n=0$ 时，式(5-4)变为

$$P_1,P_2,\cdots,P_m \leftarrow \tag{5-4″}$$

它相当于 $P_1 \vee P_2 \vee \cdots \vee P_m$。

这样，对于任一子句，总可以把它表示成式(5-4)的形式。子句的这种表示形式称为子句的蕴涵表示形式。例如，子句 $\neg P(x) \vee Q(x,y) \vee \neg R(y)$ 的蕴涵表示形式为：

$$Q(x,y) \leftarrow P(x),R(y)$$

可以看出，对于子句的蕴涵表示形式，消解过程变为：从其中一个子句的←号左侧与另一个子句的←号右侧(或从其中一个子句的←号右侧与另一个子句的←号左侧)的文字中寻找可合一文字对，然后消去它们，并把其余的左部(即←号的左侧)文字合并，作为消解式的左部，其余的右部文字合并，作为消解式的右部。

例如，子句 $Q_1(x),Q_2(x) \leftarrow P_1(x),P_2(x)$ 和 $P_1(y) \leftarrow R_1(y),R_2(y)$ 的归结式为

$$Q_1(x),Q_2(x) \leftarrow R_1(x),R_2(x),P_2(x)$$

一般地，这种蕴涵型子句的归结过程可表示如下：

设子句

$$C: P_1,\cdots,P_m \leftarrow Q_1,\cdots,Q_n$$

和

$$C': P'_1,\cdots,P'_s \leftarrow Q'_1,\cdots,Q'_t$$

中有 P_i 与 Q'_j(或 Q_i 与 P'_j)可合一，σ 为它们的 MGU，则 C 与 C' 的归结式为

$$P_1\sigma,\cdots,P_{i-1}\sigma,P_{i+1}\sigma,\cdots,P_m\sigma,P'_1\sigma,\cdots,P'_s\sigma \leftarrow$$
$$Q_1\sigma,\cdots,Q_n\sigma,Q'_1\sigma,\cdots,Q'_{j-1}\sigma,Q'_{j+1}\sigma,\cdots,Q'_t\sigma$$

或

$$P_1\sigma,\cdots,P_m\sigma,P'_1\sigma,\cdots,P'_{j-1}\sigma,P'_{j+1}\sigma,\cdots,P'_s\sigma \leftarrow$$
$$Q_1\sigma,\cdots,Q_{i-1}\sigma,Q_{i+1}\sigma,\cdots,Q_n\sigma,Q'_1\sigma,\cdots,Q'_t\sigma$$

5.6.2 Horn 子句逻辑与计算机程序语言

定义 5-28 至多含有一个正文字的子句称为 Horn(有些文献中译为“霍恩”)子句(因逻辑学家 Alfred Horn 首先研究它而得名)。

由定义，蕴涵型 Horn 子句有以下 3 种：

(1) $P \leftarrow Q_1,Q_2,\cdots,Q_m$　　称为条件子句，P 称为头部或结论

(2) $P \leftarrow$　　称为无条件。

(3) $\leftarrow Q_1,Q_2,\cdots,Q_m$　　称为目标子句，Q_i 称为子目标。

可以看出，Horn 子句形式简明，逻辑意义清晰。更重要的是 Horn 子句的消解过程可与计算机程序的执行过程统一起来。

例 5-29 证明 $P(a,c)$ 是下面子句集{(1),(2),(3),(4)}的逻辑结论。

证

(1) $P(x,z)\leftarrow P_1(x,y),P_2(y,z)$
(2) $P_1(u,v)\leftarrow P_{11}(u,v)$
(3) $P_{11}(a,b)\leftarrow$
(4) $P_2(b,c)\leftarrow$
（前提）

(5) $\leftarrow P(a,c)$　　（目标子句）

从目标子句出发，采用线性归结：

(6) $\leftarrow P_1(a,y),P_2(y,c)$　　[(5),(1),$\{a/x,c/z\}$]
(7) $\leftarrow P_{11}(a,y),P_2(y,c)$　　[(6),(2),$\{a/u,y/v\}$]
(8) $\leftarrow P_2(b,c)$　　[(7),(3),$\{b/y\}$]
(9) □　　[(8),(4)]

仔细考察以上归结过程，可以看出，上述归结过程中除最后一次外，每次产生的归结式都是目标子句；归结过程实际是对第一个目标的求解而导致了一连串目标求解；而目标求解的过程类似于计算机程序执行中的过程调用。事实上，如果用程序的眼光去看，则子句(1)和(2)就都是“过程”。例如(1)中 P 就是过程名，(5)和(1)消解就是对过程 P 的调用，而 P 的过程体为$\{P_1(x,y),P_2(y,z)\}$，从而又引起了对子过程 P_1、P_2 的调用，这样层层调用下去，子句(3)、(4)提供了过程出口。所以，子句(5)其实就相当于主程序，它包含一个过程调用。也就是说，Horn 子句与程序中的过程，基于 Horn 子句集的线性归结与程序的执行，不谋而合。

可见，完全可以把 Horn 子句逻辑作为一种计算机程序语言。这样，每一个 Horn 子句就是该语言中的语句，一个 Horn 子句的有限集合就是一个程序。这种用 Horn 子句组成的程序被称为**逻辑程序**。那么，用 Horn 子句实现的语言就是**逻辑程序设计语言**。显然，PROLOG 语言就是以 Horn 子句逻辑为基础的逻辑程序设计语言。这也就是称其为逻辑程序设计语言的原因。

PROLOG 程序的运行是一种从问题语句(目标语句)开始的线性归结过程。每次归结时，子目标的选择顺序是从左到右，新子目标的插入顺序是插入子目标队列的左端，匹配子句的顺序是自上而下，搜索空子句的策略是深度优先，推理方式是反向推理，且有回溯机制。PROLOG 程序的这种归结方法称为基于 Horn 子句的 SLD(Linear resolution with Selection function for Definite clause)归结，也称为 SLD 反驳-消解法。

上面介绍了谓词逻辑中的归结演绎推理。归结演绎虽然是一种有效的机器推理方法，但它仍存在不少问题。例如，归结策略仍然不能彻底解决大量无用归结式产生的问题。再从其本身来看，谓词公式的子句表达，掩盖了蕴涵词所表示的因果关系，使前提与结论混在一起，不便于在推理中使用启发式信息，知识表示的可读性也差。所以，这就导致人们又对非归结演绎推理进行研究。非归结演绎推理也取有不少成果，比较著名的有 Bledsoe 自然演绎法、基于规则的演绎推理、王浩算法等。由篇幅所限，这里不再介绍。

延伸学习导引

本章的内容属于自动推理的范畴，在本章的基础上可参阅自动推理方面的著作继续学习。具体来讲，延伸学习的内容包括各种非经典（或非标准）逻辑及其推理。如模态逻辑、时态逻辑、动态逻辑、真度逻辑、软语言真值逻辑、多值逻辑、多类逻辑和非单调逻辑等。除了基于各种逻辑的推理外，还有约束推理、定性推理、范例推理、并行推理、默认推理等。

习题 5

1. 将下列句子用一阶谓词形式表示。

(1) 雪是白的。

(2) 数 a 和数 b 之和大于数 c。

(3) 201 班的学生每人都有一台笔记本电脑。

(4) 如果明天天气晴朗且我们有时间，则我们去郊游。

(5) 一个三角形是等腰三角形，当且仅当其有两个角相等。

2. 什么是推理规则？举例说明一个推理形式可作为推理规则的充分必要条件是什么？

3. 求下列谓词公式的子句集。

(1) $\exists x \exists y(P(x,y) \wedge Q(x,y))$

(2) $\forall x \forall y(P(x,y) \rightarrow Q(x,y))$

(3) $\forall x \exists y((P(x,y) \vee Q(x,y)) \rightarrow R(x,y))$

(4) $\forall x(P(x) \rightarrow \exists y(P(y) \wedge R(x,y)))$

(5) $\exists x(P(x) \wedge \forall y(P(y) \rightarrow R(x,y)))$

4. 试判断下列子句集中哪些是不可满足的。

(1) $S=\{P(y) \vee \neg Q(y), \neg P(f(x)) \vee Q(y)\}$

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

(3) $S=\{\neg P(x) \vee \neg Q(y) \vee \neg L(x,y), P(a), \neg R(z) \vee L(a,z), R(b), Q(b)\}$

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

(5) $S=\{P(x) \vee Q(x), \neg Q(y) \vee R(y), \neg P(z) \vee Q(z), \neg R(u)\}$

5. 对下列各题分别证明，G 是否可肯定是 F、F_1、F_2……的逻辑结论。

(1) F：$(P \vee Q) \wedge (P \rightarrow R) \wedge (Q \rightarrow S)$

G：$R \vee S$

(2) F_1：$\forall x(P(x) \rightarrow \forall y(Q(y) \rightarrow L(x,y)))$

F_2：$\exists x(P(x) \wedge \forall y(R(y) \rightarrow L(x,y)))$

G：$\forall x(R(x) \rightarrow \neg Q(x))$

(3) F_1：$\forall x(P(x) \rightarrow Q(x) \wedge R(x))$

F_2：$\exists x(P(x) \wedge S(x))$

G：$\exists x(S(x) \wedge R(x))$

6. 设已知：

(1) 凡是清洁的东西就有人喜欢；

(2) 人们都不喜欢苍蝇。

试用谓词公式表示这两个命题，并用归结原理证明：苍蝇是不清洁的。

7. 某公司招聘工作人员，有 A、B、C 三人应聘，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。

试用谓词公式表示这三个命题，并用归结原理求证：公司一定录取 C。

8. 张某被盗，公安局派出 5 个侦查员去调查。研究案情时，侦查员 A 说“赵与钱中至少有一人作案”；侦查员 B 说“钱与孙中至少有一人作案”；侦查员 C 说“孙与李中至少有一人作案”；侦查员 D 说“赵与孙中至少有一人与此案无关”；侦查员 E 说“钱与李中至少有一人与此案无关”。假设这 5 个侦查员的话都是可信的，用谓词公式表示这 5 句话，并用归结原理推出谁是盗窃犯。

9. 试画出例 5-28 的(线性)归结演绎树。

10. Horn 子句逻辑与计算机程序语言有何关系？为什么称 PROLOG 语言为逻辑程序设计语言？