离散数学
Discrete Mathematics
for Computer Science

计算机学院计科系
薛思清 xuesiqing@cug.edu.cn

第3讲 谓词逻辑 Predicate Logic(2)

Can a mortal ask questions which God finds unanswerable?
Quite easily, I should think. All nonsense questions are unanswerable.

-C. S. Lewis

Logical Equivalence（逻辑等价，公式α和β在论域D上等值？）

Two wffs A and B are said to be logically equivalent if they both have the same truth value with respect to every interpretation of both A and B. By an interpretation of both A and B, we mean that all free variables, constants, functions, and predicates that occur in either A or B are interpreted with respect to a single domain.

If A and B are equivalent, then we writeA ≡ B or A⇔B.

1 从命题逻辑中"移植"的等值式

2 与量词有关，一阶逻辑特有的一些等值式

代入定理？置换定理？---代入/置换 谓词公式

证明公式等价基本方法？

Equivalence

A ≡ B(or A⇔B) if and only if (A → B) ∧ (B → A) is valid.

if and only if A → B and B → A are both valid.

谓词逻辑中特有的等值式

[1].在有限个体域D = $\{a_1,a_2,\ldots,a_n\}$中消除量词：

(1). $\forall xA(x) \Leftrightarrow A(a_1) \wedge A(a_2) \wedge \ldots \wedge A(a_n)$

(2). $\exists xA(x) \Leftrightarrow A(a_1) \vee A(a_2) \vee \ldots \vee A(a_n)$

[2].量词否定等值式：

(1). $\neg(\forall xA(x)) \Leftrightarrow \exists x(\neg A(x))$

(2). $\neg(\exists xA(x)) \Leftrightarrow \forall x(\neg A(x))$

[3].量词分配等值式：

(1).$\forall x(A(x) \wedge B(x)) \Leftrightarrow (\forall xA(x)) \wedge (\forall xB(x))$

(2).$\exists x(A(x) \vee B(x)) \Leftrightarrow (\exists xA(x)) \vee (\exists xB(x))$

[4].量词顺序变换等值式：

(1).$\forall x\forall y(A(x, y)) \Leftrightarrow \forall y\forall x(A(x, y))$

(2).$\exists x\exists y(A(x, y)) \Leftrightarrow \exists y\exists x(A(x, y))$

示例 Quantifiers and Negation

¬ (∀ x W) ≡ □ x ¬ W and ¬ (□ x W) ≡ ∀ x ¬ W.

Let I be an interpretation with domain D for the two wffs.

I is a model for ¬(∀ xW)

iff ¬(∀ xW) is true for I

iff ∀ xW is false for I

iff W(x/d) is false for some d ∈ D

iff ¬W(x/d) is true for some d ∈ D

iff □ x¬W is true for I

iff I is a model for □ x¬W.

¬ (∀ x ¬ W) ≡ □ x ¬ ¬ W, ∀ x ¬ W ≡ ¬ (□ x W).

示例 证明 ∃x(A(x)∨B(x))⇔∃xA(x)∨∃xB(x)

证明 令 I 为上述公式的一个解释，其个体域为D.

在解释 I 下，∃x(A(x)∨B(x))⇔T，即存在c∈D，A(c)∨B(c)⇔T，即A(c)⇔T或B(c)⇔T，因此有∃xA(x)⇔T或∃xB(x)⇔T，所以，在解释 I 下，∃xA(x)∨∃xB(x)⇔T.

反之，在解释 I 下，∃xA(x)∨∃xB(x)⇔T，即∃xA(x)⇔T或∃xB(x)⇔T，即存在c∈D，使得A(c)⇔T，或存在d∈D，使得B(d)⇔T，从而在D中存在e∈D(e=c或e=d）使得A(e)∨B(e)⇔T，所以，在解释 I 下，∃x(A(x)∨B(x))⇔T.

由以上两方面知等值式成立.

[5].量词辖域的收缩与扩张等值式：下述等值式中，变元x不在B中出现

(1). ∀x(A(x)∨B) ⇔ (∀xA(x)) ∨ B

(2). ∀x(A(x)∧B) ⇔ (∀xA(x)) ∧ B

(3). ∀x(A(x)→B) ⇔ (∃xA(x)) → B

(4). ∀x(B→A(x)) ⇔ B → (∀xA(x))

(5). ∃x(A(x)∨B) ⇔ (∃xA(x)) ∨ B

(6). ∃x(A(x)∧B) ⇔ (∃xA(x)) ∧ B

(7). ∃x(A(x)→B) ⇔ (∀xA(x)) → B

(8). ∃x(B→A(x)) ⇔ B → (∃xA(x))

注意，应用这些公式时，公式中，合取（析取）的两个分支，和蕴涵的前件与后件选取的约束变元不同，当约束变元相同时，且又不能运用∀对∧的分配律和∃对∨的分配律时则可使用换名规则，使得约束变元不同。

示例 证明等值式.

∃x(A(x)→B(x))⇔∀xA(x)→∃xB(x)

证明 ∃x(A(x)→B(x))

⇔∃x(¬A(x)∨B(x))

⇔∃x¬A(x)∨∃xB(x)

⇔¬∀xA(x)∨∃xB(x)

⇔∀xA(x)→∃xB(x).

基于逻辑等价定理的证明方法

The following is a proof of the above equivalences consisting of two subproofs showing that each side implies the other.

First we'll prove the validity of □ x (A(X) → B(X)) → (∀ x A(X) → □ x B(X)).

Proof: Let I be a model for □ x (A(X) → B(X)) with domain D. Then □ x (A(X) → B(X)) is true for I, which means that A(d) →B(d) is true for some d ∈ D. Therefore, either A(d) is false or both A(d) and B(d) are true for some d ∈ D. If A(d) is false, then ∀ x A(X) is false for I; if both A(d) and B(d) are true, then□ xB(X) is true for I. In either case,∀ xA(X)→□ x B(X) is true for I. Therefore, I is a model for ∀ x A(X) → □ x B(X). QED.

基于逻辑等价定理的证明方法

Now we'll prove the validity of (∀ x A(X) → □ x B(X)) → □ x (A(X) → B(X)).

Proof: Let I be a model for ∀ x A(X) → □ x B(X) with domain D. Then ∀ x A(X) → □ x B(X) is true for I.

Therefore, either ∀ x A(X) is false for I or both ∀ x A(X) and □ x B(X) are true for I. If ∀ x A(X) is false for I, then A(d) is false for some d ∈ D. Therefore, A(d) → B(d) is true. If both ∀ x A(X) and □ x B(X) are true for I, then there is some c ∈ D such that both A(c) and B(c) are true. Thus A(c) → B(c) is true. So in either case, □ x (A(X) → B(X)) is true for I.

Thus I is a model for □ x (A(X) → B(X)). QED.

3. 等值演算

练习

证明 a. $\exists x\,(A(x) \vee B(x)) \Leftrightarrow \exists x\, A(x) \vee \exists x\, B(x)$.

b. $\forall x\,(A(X) \wedge B(X)) \Leftrightarrow \forall x\, A(X) \wedge \forall x\, B(X)$.

证明 (a) (b) ?

$\square\ x\,(A(x) \vee B(x))$

$\Leftrightarrow \square\ x\,(\neg A(x) \rightarrow B(x))$

$\Leftrightarrow \forall x \neg A(x) \rightarrow \square\ x\, B(x)$

$\Leftrightarrow \neg \square\ x\, A(x) \rightarrow \square\ x\, B(x)$

$\Leftrightarrow \square\ x\, A(x) \vee \square\ x\, B(x)$.

$\forall x(A(x) \wedge B(x)) \equiv \neg \square\ x(\neg A(x) \vee \neg B(x))$

$\forall x\, A(X) \wedge \forall x B(X) \Leftrightarrow \neg(\exists x \neg A(x) \vee \exists x \neg B(x))$

3. 等值演算

练习 判断以下式子是否成立?

(1) $\exists x A(x) \wedge \exists x B(x) \Leftrightarrow \exists x(A(x) \wedge B(x))$

(2) $\forall x A(x) \vee \forall x B(x) \Leftrightarrow \forall x(A(x) \vee B(x))$

(3) $\forall x \exists y P(x,y) \Leftrightarrow \exists y \forall x P(x,y)$

3. 等值演算

设D = {*a*, *b*, *c*}，则:

$\forall x(A(x) \vee B(x)) \Leftrightarrow$

(化为析取公式)

而 $(\forall x A(x)) \vee (\forall x B(x)) \Leftrightarrow$

(化为析取公式)

3. 等值演算

$\forall x(A(x) \vee B(x)) \Leftrightarrow (A(a) \vee B(a)) \wedge (A(b) \vee B(b)) \wedge (A(c) \vee B(c))$

$\Leftrightarrow (A(a) \wedge A(b) \wedge A(c)) \vee (B(a) \wedge A(b) \wedge A(c)) \vee (A(a) \wedge B(b) \wedge A(c)) \vee$
$(B(a) \wedge B(b) \wedge A(c)) \vee (A(a) \wedge A(b) \wedge B(c)) \vee (B(a) \wedge A(b) \wedge B(c)) \vee$
$(A(a) \wedge B(b) \wedge B(c)) \vee (B(a) \wedge B(b) \wedge B(c))$

而 $(\forall x A(x)) \vee (\forall x B(x)) \Leftrightarrow (A(a) \wedge A(b) \wedge A(c)) \vee (B(a) \wedge B(b) \wedge B(c))$

3. 等值演算

设D = {a, b, c}，则:

(i). $\forall x \forall y A(x, y)$,(ii). $\forall x \exists y A(x, y)$, (iii). $\exists x \forall y A(x, y)$,(iv). $\exists x \exists y A(x, y)$ =?

(i). $\forall x \forall y A(x, y) \Leftrightarrow \forall y A(a, y) \wedge \forall y A(b, y) \wedge \forall y A(c, y)$

$\Leftrightarrow (A(a, a) \wedge A(a, b) \wedge A(a, c)) \wedge (A(b, a) \wedge A(b, b) \wedge A(b, c)) \wedge (A(c, a) \wedge A(c, b) \wedge A(c, c))$

(ii). $\forall x \exists y A(x, y) \Leftrightarrow \exists y A(a, y) \wedge \exists y A(b, y) \wedge \exists y A(c, y)$

$\Leftrightarrow (A(a, a) \vee A(a, b) \vee A(a, c)) \wedge (A(b, a) \vee A(b, b) \vee A(b, c)) \wedge (A(c, a) \vee A(c, b) \vee A(c, c))$

(iii). $\exists x \forall y A(x, y) \Leftrightarrow \forall y A(a, y) \vee \forall y A(b, y) \vee \forall y A(c, y)$

$\Leftrightarrow (A(a, a) \wedge A(a, b) \wedge A(a, c)) \vee (A(b, a) \wedge A(b, b) \wedge A(b, c)) \vee (A(c, a) \wedge A(c, b) \wedge A(c, c))$

(iv). $\exists x \exists y A(x, y) \Leftrightarrow \exists y A(a, y) \vee \exists y A(b, y) \vee \exists y A(c, y)$

$\Leftrightarrow (A(a, a) \vee A(a, b) \vee A(a, c)) \vee (A(b, a) \vee A(b, b) \vee A(b, c)) \vee (A(c, a) \vee A(c, b) \vee A(c, c))$

4 范式

前束范式

设A为谓词逻辑公式，若A具有如下形式:

$Q_1 x_1 Q_2 x_2 \ldots Q_n x_n B$

则称A为前束范式(Prenex Normal Form)。其中$Q_i (1 \leq i \leq n)$

是$\forall$或$\exists$，B为不含量词的公式。

前束合取范式、前束析取范式

示例

$\forall y \exists u \exists v (A(x,y) \wedge B(u,y) \rightarrow R(u,v))$

如何求解前束范式?

(1)消去联结词→，↔及多余的量词;

(2)将联结词¬向内深入，使之只作用于原子谓词公式;

(3)利用改名或代入规则使所有约束变元的符号均不同，并且自由变元与约束变元的符号也不同;

(4)利用量词辖域的扩张与收缩律，扩大量词的辖域至整个公式。

前束范式一定存在吗?

任意一个谓词公式都存在着与之等值的前束范式。

示例 将公式$\forall x\forall y(\exists z(P(x,z)\wedge P(y,z))\rightarrow\exists zQ(x,y,z))$化为前束范式。

解 $\forall x\forall y(\exists z(P(x,z)\wedge P(y,z))\rightarrow\exists zQ(x,y,z))$

$\Leftrightarrow\forall x\forall y(\neg\exists z(P(x,z)\wedge P(y,z))\vee\exists zQ(x,y,z))$ (消去→)

$\Leftrightarrow\forall x\forall y(\forall z(\neg P(x,z)\vee\neg P(y,z))\vee\exists zQ(x,y,z))$ (¬深入)

$\Leftrightarrow\forall x\forall y(\forall z(\neg P(x,z)\vee\neg P(y,z))\vee\exists uQ(x,y,u))$ (改名)

$\Leftrightarrow\forall x\forall y\forall z\exists u(\neg P(x,z)\vee\neg P(y,z)\vee Q(x,y,u))$ (量词前移)

示例 将公式$\neg\forall x(\exists yA(x,y)\rightarrow\exists x\forall y(B(x,y)\wedge\forall y(A(y,x)\rightarrow B(x,y))))$化为前束合取范式和前束析取范式。

解 $\neg\forall x(\exists yA(x,y)\rightarrow\exists x\forall y(B(x,y)\wedge\forall y(A(y,x)\rightarrow B(x,y))))$

$\Leftrightarrow\neg\forall x(\neg\exists yA(x,y)\vee(\exists x\forall y(B(x,y)\wedge\underline{\forall u}(\neg A(\underline{u},x)\vee B(x,\underline{u})))))$

$\Leftrightarrow\exists x(\exists yA(x,y)\wedge\neg(\exists x\forall y(B(x,y)\wedge\underline{\forall u}(\neg A(u,x)\vee B(x,u)))))$

$\Leftrightarrow\exists x(\exists yA(x,y)\wedge\neg(\exists x\forall y\underline{\forall u}(B(x,y)\wedge(\neg A(u,x)\vee B(x,u)))))$

$\Leftrightarrow\exists x(\exists yA(x,y)\wedge\forall v\exists w\exists u(\neg B(v,w)\vee(A(u,v)\wedge\neg B(v,u))))$

$\Leftrightarrow\underline{\exists x\exists y\forall v\exists w\exists u}(A(x,y)\wedge\neg B(v,w))\vee(A(x,y)\wedge A(u,v)\wedge\neg B(v,u)))$

示例 求下列公式的前束范式:

[1]. $\forall xF(x,y)\wedge\forall yF(x,y)$　[2]. $\forall x\forall yF(x,y)\rightarrow\forall x\forall yF(x,y)$

[3]. $\forall xF(x,y)\rightarrow(\forall xG(x)\rightarrow\exists yF(y,z))$

解 [1]. $\forall xF(x,y)\wedge\forall yF(x,y)\Leftrightarrow\forall xF(x,u)\wedge\forall yF(v,y)\Leftrightarrow\forall x\forall y(F(x,u)\wedge F(v,y))$，显然不能变换为：$\forall x\forall y(F(x,y)\wedge F(x,y))$。

[2]. $\forall x\forall yF(x,y)\rightarrow\forall x\forall yF(x,y)\Leftrightarrow\forall x\forall yF(x,y)\rightarrow\forall u\forall vF(u,v)$

$\Leftrightarrow\exists x(\forall yF(x,y)\rightarrow\forall u\forall vF(u,v))\Leftrightarrow\exists x\exists y(F(x,y)\rightarrow\forall u\forall vF(u,v))$

$\Leftrightarrow\exists x\exists y\forall u\forall v(F(x,y)\rightarrow F(u,v))\Leftrightarrow\ldots$

[3]. $\forall xF(x,y)\rightarrow(\forall xG(x)\rightarrow\exists yF(y,z))\Leftrightarrow\forall xF(x,u)\rightarrow(\forall vG(v)\rightarrow\exists yF(y,z))\Leftrightarrow\forall xF(x,u)\rightarrow(\exists v\exists y(G(v)\rightarrow F(y,z)))\Leftrightarrow\exists x\exists v\exists y(F(x,u)\rightarrow(G(v)\rightarrow F(y,z)))\Leftrightarrow\ldots$

► 归结原理

由J.A.Robinson由1965年提出。

■ 与演绎法(deductive inference)完全不同的一种逻辑演算(inductive inference)方法。

■ 一阶逻辑中，至今为止的最有效的半可判定的算法。即，一阶逻辑中任意恒真公式，使用归结原理，总可以在有限步内给以判定。

■ 语义网络、框架表示、产生式规则等等都是以推理方法为前提的。即，有了规则已知条件，顺藤摸瓜找到结果。而归结方法是自动推理、自动推导证明用的（"数学定理机器证明"）。

自然演绎法：该方法依据推理规则从前提和公理中可以推出许多定理，如果待证明的定理在其中则定理得证。典型代表：LT程序、证明平面几何的程序。

判定法：该方法是对一类问题找出统一的计算机上可实现的算法。典型代表：数学家吴文俊教授——吴氏方法。

人机交互进行定理证明：计算机作为数学家的辅助工具，用计算机帮助人完成手工证明中的难以完成的烦杂的大量计算推理和穷举。典型代表：四色定理的证明。

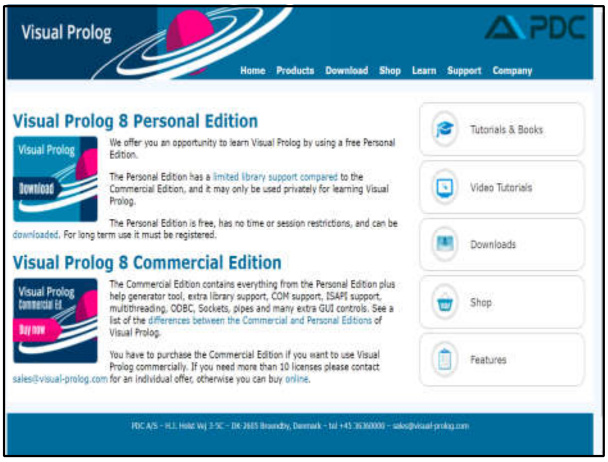
定理证明器：它是研究一切可判定问题的证明方法。鲁滨逊的归结原理。

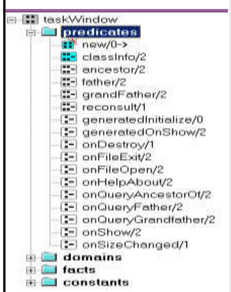
谓词逻辑是一种表达力很强的形式语言，谓词逻辑及其推理方法是人工智能中知识表示方法,机器推理,定理证明的基本方法。

谓词逻辑中的替换合一技术，也是符号推理中模式匹配的基本技术。

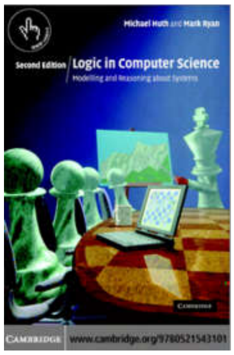
归结原理（谓词逻辑的归结演绎推理）——PROLOG（PROgramming in LOGic）

- designed for natural language processing
- has been used in a variety of other areas as well, including:
  - theorem proving
  - expert systems
  - games
  - automated answering systems
  - ontologies
  - sophisticated control systems
- "...modern Prolog environments support creation of grgraphical user interfaces, as well as administrative and networked applications."






```
domains
   gender = female(); male().

class facts - familyDB
   person : (string Name, gender Gender).
   parent : (string Person, string Parent).

class predicates
   father : (string Person, string Father) nondeterm anyflow.
clauses
   father(Person, Father) :-
      parent(Person, Father),
      person(Father, male()).

class predicates
   grandFather : (string Person, string Grandfather) nondeterm anyflow.
clauses
   grandfather(Person, Grandfather) :-
      parent(Person, Parent),
      father(Parent, Grandfather).

class predicates
   ancestor : (string Person, string Ancestor) nondeterm anyflow.
clauses
   ancestor(Person, Ancestor) :-
      parent(Person, Ancestor).
   ancestor(Person, Ancestor) :-
      parent(Person, P1),
      ancestor(P1, Ancestor).

class predicates
   reconsult : (string FileName).
clauses
```



归结方法

- 命题逻辑中，归结过程
  - 将命题写成合取范式
  - 求出子句集
  - 对子句集使用归结推理规则
  - 归结式作为新子句参加归结
  - 归结式为空子句□，S是不可满足的（矛盾），原命题成立。
- 谓词逻辑中：除了有量词和函数以外，其余和命题逻辑中归结过程一样。

归结方法

示例 证明公式：$(P \to Q) \Rightarrow (\sim Q \to \sim P)$

证明：

（1）根据归结原理，将待证明公式转化成待归结命题公式：

$(P \to Q) \wedge \sim(\sim Q \to \sim P)$

（2）分别将公式前题项化为合取范式：

$P \to Q = \sim P \vee Q$

结论求～后的后项化为合取范式：

$\sim(\sim Q \to \sim P) = \sim(Q \vee \sim P) = \sim Q \wedge P$

两项合并后化为合取范式：

$(\sim P \vee Q) \wedge \sim Q \wedge P$

（3）则子句集为：

$\{\sim P \vee Q, \sim Q, P\}$

子句集为： { ~P∨Q， ~Q，P}

（4）对子句集中的子句进行归结可得：

1. ~P∨Q
2. ~Q
3. P
4. Q， （1，3归结）
5. □， （2，4归结）

由上可得原公式成立。

示例

设公理集：	化子句集：
P,	(P∧Q) →R
(P∧Q) →R,	=> ~(P∧Q)∨R
(S∨T) →Q,	=> ~P∨~Q∨R
T	
求证：R	(S∨T) →Q
子句集：	=> ~ (S∨T)∨Q
(1) P	=> (~S∧~T)∨Q
(2) ~P∨~Q∨R	=> (~S∨Q) ∧(~T∨Q)
(3) ~S∨Q	=> {~S∨Q, ~T∨Q}
(4) ~T∨Q	
(5) T	
(6) ~R（目标求反）	

子句集：	归结：	
(1) P	(7) ~P∨~Q	(2, 6)
(2) ~P∨~Q∨R	(8) ~Q	(1, 7)
(3) ~S∨Q	(9) ~T	(4, 8)
(4) ~T∨Q	(10) nil	(5, 9)
(5) T		
(6) ~R（目标求反）		

归结原理（谓词逻辑的归结演绎推理）

Skolem范式

1 化成前束合取范式；

2 化成Skolem标准型：消去存在量词，略去全称量词

消去存在量词时，需要进行变元替换。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相应约束变元，这样的常量符号称为Skolem常量

示例 求公式的SKOLEM标准形

∃x(X(x)∧(∃yY(x，y)→∃xZ(x)))

解：①先把公式化为前束范式

原式=∃x(X(x)∧(¬∃yY(x，y)∨∃xZ(x)))

=∃x(X(x)∧(∀y¬Y(x，y)∨∃xZ(x)))

=∃x(X(x)∧(∀y¬Y(x，y)∨∃uZ(u)))

=∃x∀y∃u(X(x)∧(¬Y(x，y)∨ Z(u)))

②化为SKOLEM标准形

原式=∀y∃u(X(a)∧(¬Y(a，y)∨ Z(u)))

=∀y(X(a)∧(¬Y(a，y)∨ Z(f(y))))

= (X(a)∧(¬Y(a，y)∨ Z(f(y))))

应用归结原理的具体步骤：

1. 将定理或问题用逻辑形式表示。
2. 消去存在量词，使公式中出现的所有个体变元只受全称量词约束。
3. 构造子句集，包括将所有前提表示为子句形式；将结论否定也表示为子句形式。
4. 证明子句集S的不可满足性,即应用归结原理和置换合一算法，反复推求两子句的归结式（对命题逻辑情形无需采用合一算法）,直到最终推导出空子句□,即表明定理得证或问题有解。

推理过程便由计算机自动执行。

归结方法

首先假定要证明的结论不成立，然后通过推导出存在矛盾的方法，反证出结论成立。在归结法中首先对结论求反，然后将已知条件和结论的否定合在一起用子句集表达。如果该子句集存在矛盾，则证明了结论的正确性。其思路如下：

设S是已知条件和结论的否定合并后所对应的子句集。假定有一种变换方法，可以对S实施一系列的变换。而且该变换能够保证变换前后的子句集，在不可满足的意义下是等价的。这样，如果最终得到的子句集是不可满足的，就证明了子句集S是不可满足的，从而证明结论成立。

由前面合适公式转化为子句集的过程可知，子句集中的子句是"与"的关系，如果在子句集中出现了空子句，则说明该子句集是不可满足的。因此，归结过程就是"寻找"空子句的过程。如果把这一过程看作是搜索的话，初始状态就是已知条件和结论的否定对应的子句集，而目标就是空子句，规则就是归结。

归结方法

在定理证明系统中，已知一公式集F_1，F_2，…，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$F_1 \wedge F_2 \wedge \ldots \wedge F_n \rightarrow W$为永真式。如果直接应用推理规则进行推导，则由于演绎技巧等因素的影响，给建立机器证明系统带来困难。

另一种证明法是采用间接法（反证法），即不去证明$F_1 \wedge F_2 \wedge \ldots \wedge F_n \rightarrow W$为永真，而是去证明$F = F_1 \wedge F_2 \wedge \ldots \wedge F_n \wedge \sim W$为永假，这等价于证明F对应的子句集$S = S_0 \cup \{\sim W\}$为不可满足的。这时如果用归结作为推理规则使用时，就可以使机器证明大为简化。

归结方法

归结原理的基本思路是设法检验扩充的子句集S_i是否含有空子句，若S集中存在空子句，则表明S为不可满足的。

若没有空子句，则就进一步用归结法从S导出S_1，然后再检验S_1是否有空子句。可以证明用归结法导出的扩大子句集S_1，其不可满足性是不变的，所以若S_1中有空子句，也就证得了S的不可满足性。

归结过程可以一直进行下去，这就是要通过归结过程演绎出S的不可满足性来，从而使定理得到证明。

归结方法

定理：二个子句C_1和C_2的归结式C是C_1和C_2的逻辑推论。

证：

设C1＝L∨P，C2＝（~L）∨Q关于解释I为真，则需证明C＝P∨Q关于解释I也为真。

关于解释I，L和~L二者中必有一个为假。若L为假，则P必为真，否则C1为假，这与前提假设矛盾，所以只能是P为真。

同理，若~L为假，则Q必为真。最后有C＝P∨Q关于解释I为真，即C是C1和C2的逻辑推论。[证毕]

推论：

子句集$S = \{C_1, C_2, \ldots, C_n\}$与子句集$S_1 = \{C, C_1, C_2, \ldots, C_n\}$的不可满足性是等价的（$S_1$中C是$C_1$和$C_2$的归结式，即$S_1$是对S应用归结法后导出的子句集）。

归结方法

证明：

设S是不可满足的，则C1，C2，…，Cn中必有一为假，因而S1必为不可满足的。

设S1是不可满足的，则对于不满足S1的任一解释I，可能有两种情况：

①I使C为真，则C1，C2，…，Cn中必有一子句为假，因而S是不可满足的。

注意这里假定C是由C1、C2归结得到的。

②I使C为假，则根据定理有C1∧C2为假，即I或使C1为假，或使C2为假，因而S也是不可满足的。

由此可见S和S1的不可满足性是等价的。[证毕]

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，…。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。由此可以看出，用归结法证明定理，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。

示例

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。
   ∀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。
   ∀x Roman(x)→Loyalto(x, Caesar)∨Hate(x, Caesar)
6. 每个人都忠于某个人。
   ∀x∃y Loyalto(x, y)
7. 人们只想暗杀他们不忠于的统治者。
   ∀x∀y man(x)∧Ruler(y)∧Tryassassinate(x, y)→~Loyalto(x, y)
8. 马科斯试图谋杀恺撒。
   Tryassassinate(Marcus, Caesar)

证明：马科斯仇恨恺撒。Hate(Marcus, Caesar)

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)∨～Rule(y1) ∨ ～ Tryassassinate(x4,y1) ∨ ～ Loyalto((x4，y1))
8. Tryassassinate(Marcus, Caesar)

9. ~Hate(Marcus, Caesar)