

# 人工智能：知识表示和推理 II

饶洋辉

计算机学院,

中山大学

[raoyangh@mail.sysu.edu.cn](mailto:raoyangh@mail.sysu.edu.cn)

<http://cse.sysu.edu.cn/node/2471>

课件来源：中山大学刘咏梅教授、王甲海教授；多伦多大学Hector Levesque教授和Sheila McIlraith教授；海军工程大学贲可荣教授等

# 知识表示和推理

- 1 谓词逻辑
- 2 归结推理
- 3 知识图谱

# 推理程序

- 我们希望找到一种自动的推理程序来判断“ $KB$ 逻辑上蕴涵 $\alpha$ ”是否成立。
- 对于一个推理程序：
- 它是合理的（Sound）是指：如果该推理程序认为答案为yes，那么“ $KB$ 逻辑上蕴涵 $\alpha$ ”是成立的；
- 它是完备的（Complete）是指：如果“ $KB$ 逻辑上蕴涵 $\alpha$ ”，那么该推理程序会认为答案为yes。

# 归结推理

- 1965年，由Robinson (鲁宾逊)提出归结法
- 归结法的基本思想
- 命题逻辑的归结原理和过程
- 谓词逻辑的归结原理和过程
- 应用归结原理求解问题
- 归结反演

# 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。

# 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) 。

# 归结式的定义及性质

- 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 $\{(P, C_1), (\neg P, C_2)\} \vdash (C_1, C_2)$

# 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。



# 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： $S \vdash C$

# 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  $C$ 。  
例如， $P \models (P, Q)$ ，但是  $P$  不能推导出  $(P, Q)$

# 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。

# 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 $P$ ，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 $P$ 的否定式 $\neg P$ 也转化成子句集表示，并将其加到 $S_0$ 中，得 $S = S_0 \cup S_{\neg P}$ ；
- (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

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

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

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

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

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

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

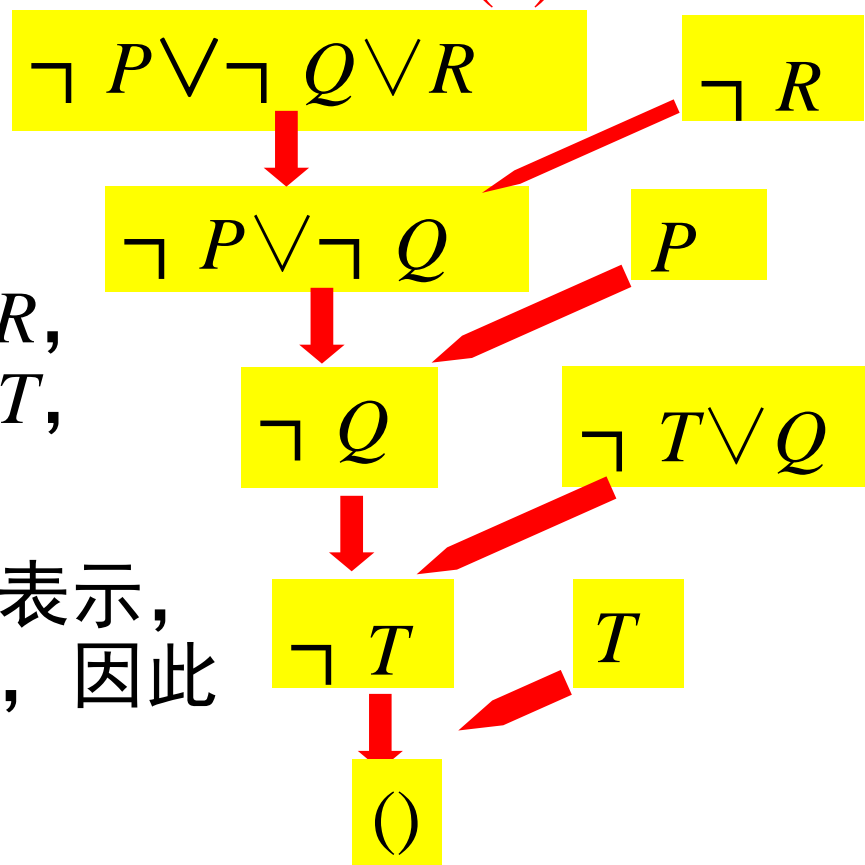
$T \dots\dots\dots(4)$

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。



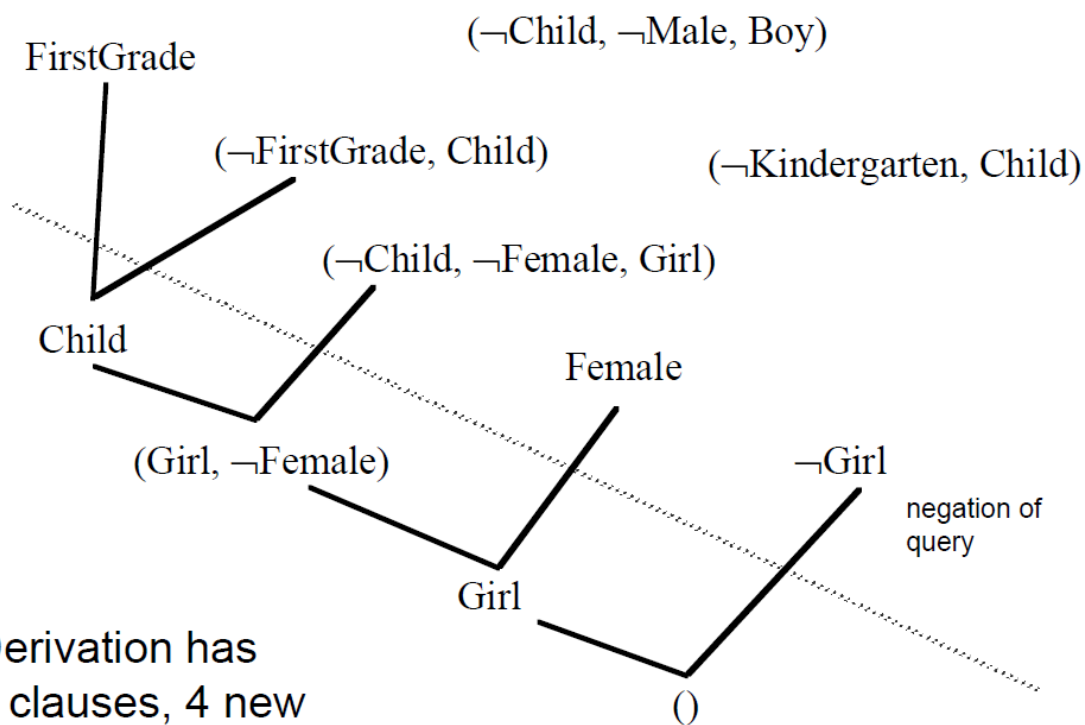
# 命题逻辑的归结原理和过程

例2:

KB

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

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




Derivation has  
9 clauses, 4 new

# 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


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

?



# 谓词逻辑的归结原理和过程

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

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)。

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

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

(2) **内移否定符号 $\neg$** ，将其移到紧靠谓词的位置上。

$$\text{双重否定律 } \neg(\neg P) \Leftrightarrow P$$

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

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

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

# 谓词逻辑的归结原理和过程

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

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。

# 谓词逻辑的归结原理和过程

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

Skolemize:

对于一般情况

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

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

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。

# 谓词逻辑的归结原理和过程

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

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。

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

# 谓词逻辑的归结原理和过程

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

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。

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

# 谓词逻辑的归结原理和过程

✱ 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

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

# 谓词逻辑的归结原理和过程

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

# 谓词逻辑的归结原理和过程

✿ 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

- 在证明定理的演绎过程中，经常要对量化的表达式进行匹配操作，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。

# 谓词逻辑的归结原理和过程

## 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  $\sigma = \{v_1/t_1, v_2/t_2, \dots, v_n/t_n\}$  来表示任一置换。 $v_i/t_i$  是指表达式中的变量  $v_i$  以项  $t_i$  来替换，且不允许  $v_i$  用与  $v_i$  有关的项  $t_i$  (但是  $t_i$  中可以包含其它变量) 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  $E\sigma$ 。

# 谓词逻辑的归结原理和过程

## 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  $E(\theta\sigma)$ 。

# 谓词逻辑的归结原理和过程

## 合一 (Unify) :

$$\theta = \{x_1 = s_1, x_2 = s_2, \dots, x_m = s_m\}, \sigma = \{y_1 = t_1, y_2 = t_2, \dots, y_k = t_k\}$$

- 合成置换 $\theta\sigma$ 的组成：1)  $\theta$ 的置换对，只是 $\theta$ 的项被 $\sigma$ 作了置换；2)  $\sigma$ 中与 $\theta$ 变量不同的那些变量对。
- 合成置换 $\theta\sigma$ 的步骤：
  - 1. Get  $S = \{x_1 = s_1\sigma, x_2 = s_2\sigma, \dots, x_m = s_m\sigma, y_1 = t_1, y_2 = t_2, \dots, y_k = t_k\}$
  - 2. Delete any equation  $y_i = s_i$  where  $y_i$  is equal to one of the  $x_j$  in  $\theta$
  - 3. Delete any identities, i.e., equations of the form  $v = v$

# 谓词逻辑的归结原理和过程

## 合一 (Unify) :

- 令  $\theta = \{x = f(y), y = z\}$ ,  $\sigma = \{x = a, y = b, z = y\}$ 
  - 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  $\theta\epsilon = \theta$ 。

# 谓词逻辑的归结原理和过程

## 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.

# 谓词逻辑的归结原理和过程

## 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.

# 谓词逻辑的归结原理和过程

## 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  $\lambda = \{x = a\}$



# 谓词逻辑的归结原理和过程

## 计算MGU:

- The MGU is the “least specialized” way of making atomic formulas with variables match.
- We can compute MGUs.
- Intuitively we line up the two formulas and find the first sub-expression where they disagree.
- The pair of subexpressions where they first disagree is called the disagreement set.
- The algorithm works by successively fixing disagreement sets until the two formulas become syntactically identical.

# 谓词逻辑的归结原理和过程

## 计算MGU:

Given two atomic formulas  $f$  and  $g$

- ①  $\sigma = \{\}; S = \{f, g\}$
- ② If  $S$  contains an identical pair of formulas, stop and return  $\sigma$  as the MGU of  $f$  and  $g$ .
- ③ Else find the disagreement set  $D = \{e_1, e_2\}$  of  $S$
- ④ If  $e_1 = V$  a variable, and  $e_2 = t$  a term not containing  $V$  (or vice-versa) then let  $\sigma = \sigma\{V = t\}$ ;  $S = S\{V = t\}$ ; Goto 2
- ⑤ Else stop,  $f$  and  $g$  cannot be unified.

Note: to update  $\sigma$ , we must compose  $\sigma$  with  $\{V = t\}$ .  
A common error is to just add  $V = t$  to  $\sigma$ .

# 谓词逻辑的归结原理和过程

示例:

- ①  $P(f(a), g(x))$  and  $P(y, y)$
- ②  $P(a, x, h(g(z)))$  and  $P(z, h(y), h(y))$
- ③  $P(x, x)$  and  $P(y, f(y))$

# 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.

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

# 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.  $(\neg R(x), L(x))$
  2.  $(\neg D(y), \neg L(y))$
  3.  $D(a)$
  4.  $I(a)$
  5.  $(\neg I(z), R(z))$

一个可行的证明过程：

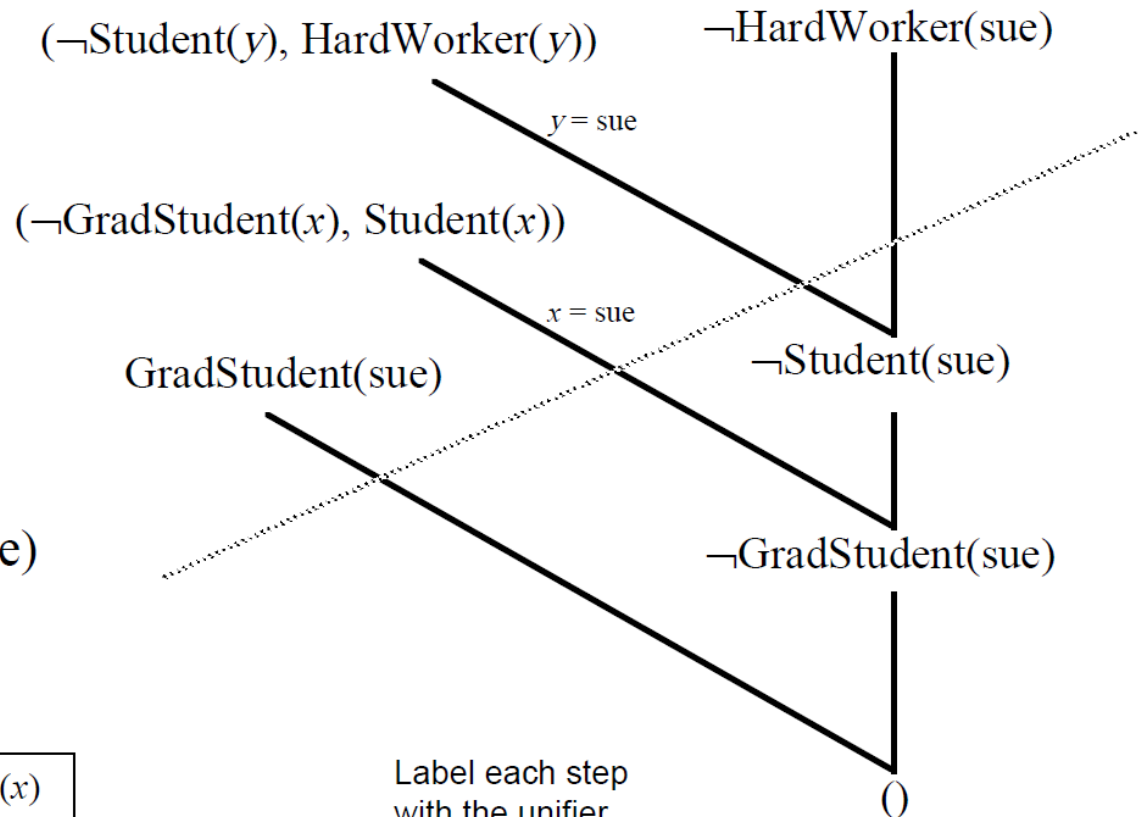
6.  $R[4, 5] \{z = a\} R(a)$
7.  $R[1, 6] \{x = a\} L(a)$
8.  $R[2, 7] \{y = a\} \neg D(a)$
9.  $R[3, 8] ()$

# 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

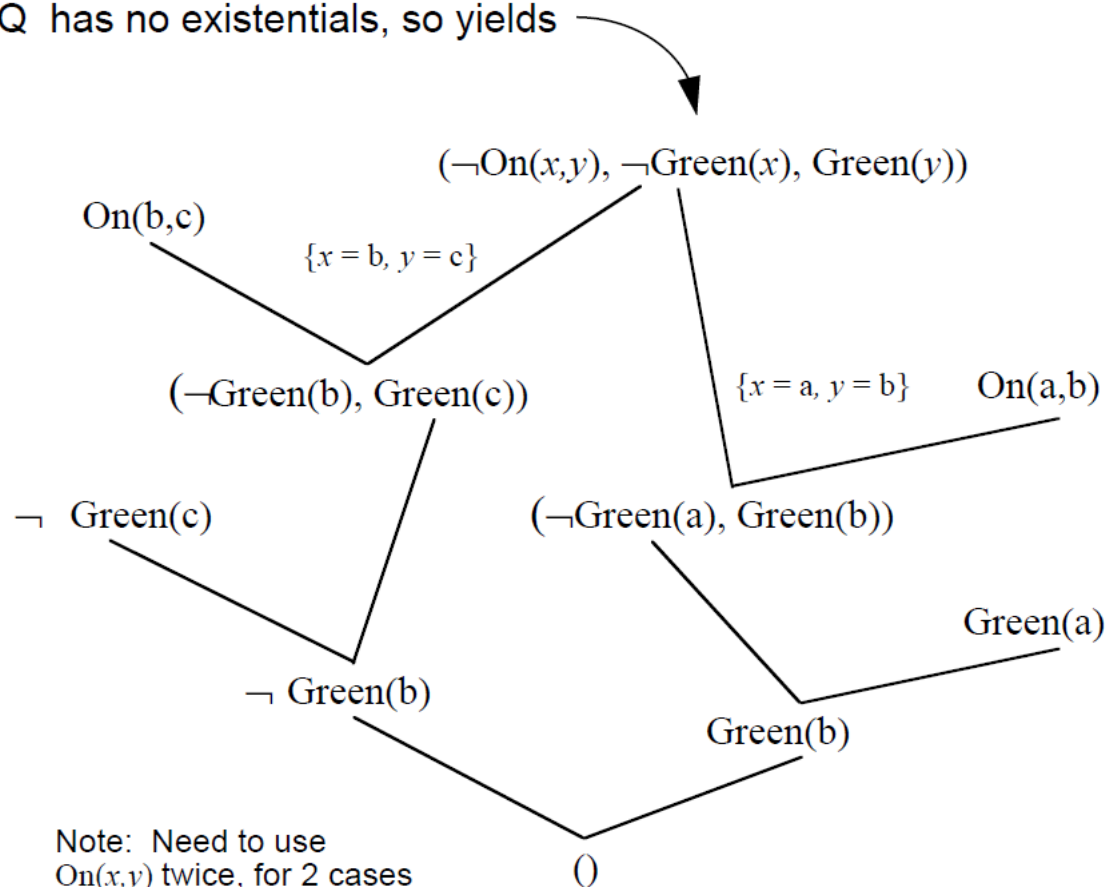
# 示例3

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





# 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

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

# 不可判定问题

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

KB:

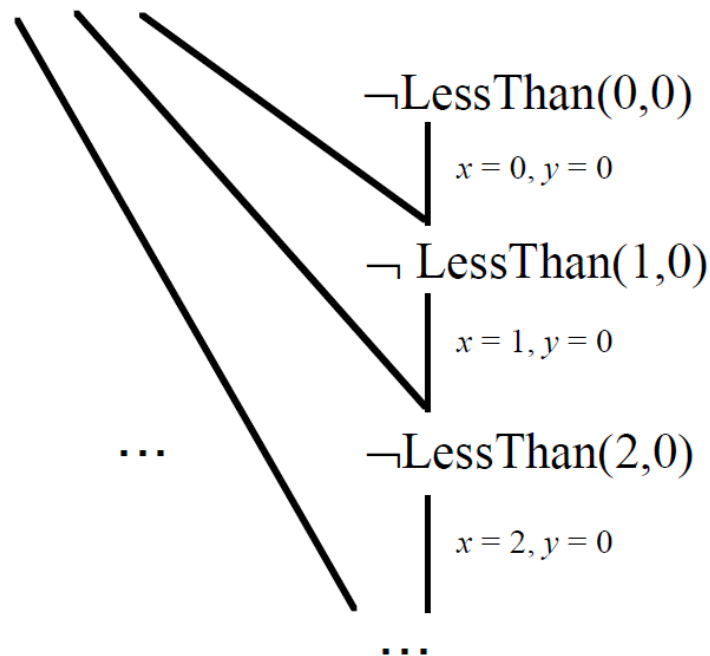
$\text{LessThan}(\text{succ}(x), y) \rightarrow \text{LessThan}(x, y)$

Query:

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

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

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



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。

# 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

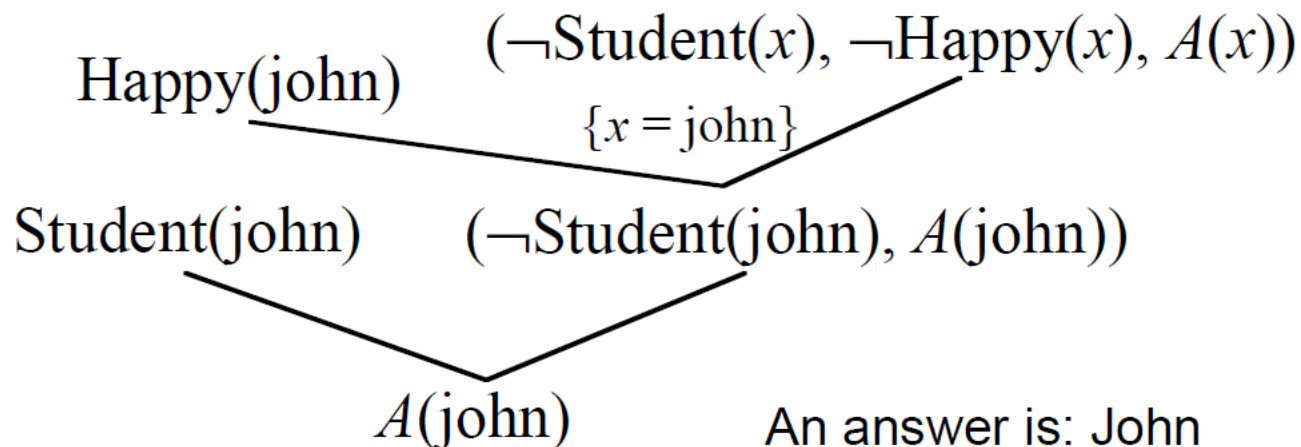
# 应用归结原理求解问题

- Replace query  $\exists x P(x)$  by  $\exists x [P(x) \wedge \neg answer(x)]$
- Instead of deriving  $()$ , derive any clause containing just the answer predicate
- 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。

# 示例1

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

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



# 示例2

KB:

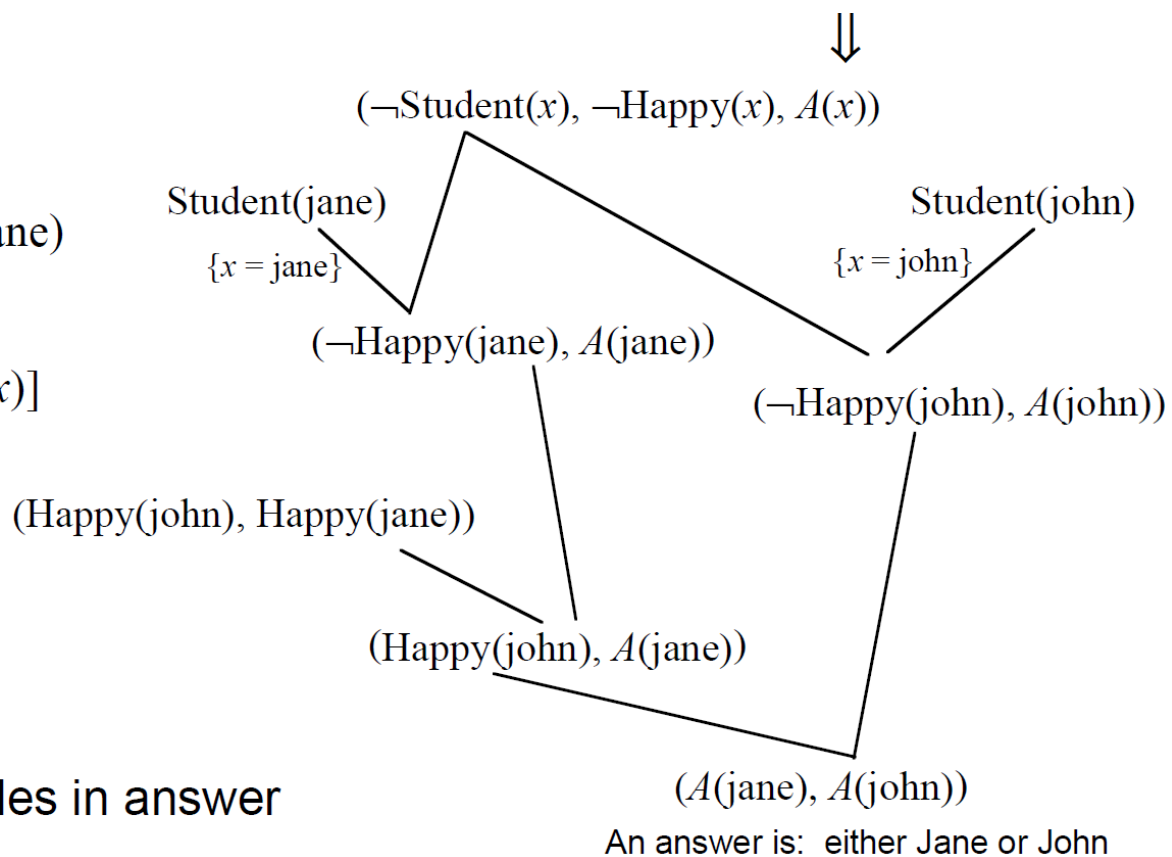
Student(john)

Student(jane)

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

Query:

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



Note: can have variables in answer

# 练习

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

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

# 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。



# 吴氏方法

吴文俊（1919年5月12日－2017年5月7日），1919年5月12日出生于上海，祖籍浙江嘉兴，数学家，中国科学院院士，中国科学院数学与系统科学研究院研究员，系统科学研究所名誉所长。

吴文俊先生的研究工作涉及数学的诸多领域，其主要成就表现在拓扑学和数学机械化两个领域。他为拓扑学做了奠基性的工作；他的示性类和示嵌类研究被国际数学界称为“吴公式”，“吴示性类”，“吴示嵌类”，至今仍被国际同行广泛引用。

# 吴氏方法

**吴方法进行几何定理机器证明的步骤如下：**

- 第一步是几何问题代数化，建立坐标系，并将命题涉及的几何图形的点选取适当的坐标；
- 然后把命题的条件和结论表示为坐标的多项式方程组；
- 最后判断条件方程组的解是否满足结论方程。

# 吴氏方法

- 通常的几何命题涉及的多项式方程组都是非线形的，一般无法将约束变元求出。吴氏方法是利用伪除法判定条件方程组的解是否是结论方程组的解。而且利用吴氏方法不仅可以判断定理的正确与否，还可以自动找出定理赖以成立的非退化条件，这是传统的做法无法做到的。
- 多项式的伪余除法可以通过计算机做符号计算进行。此外，单点例证法和数值并行法，这两种方法与吴方法进行大量符号计算不同，主要利用数值计算的方法进行定理的证明，所以有时也被单独列为一类方法，即几何定理证明的数值方法。数值方法与其它方法相比，具有效率高的优点。

# 王氏算法

王浩（1921年5月20日—1995年5月13日）数理逻辑学家。祖籍山东省德州市齐河县，生于山东省济南市。

20世纪50年代初被选为美国科学院院士，后又被选为不列颠科学院外国院士。1983年，被国际人工智能联合会授予第一届“数学定理机械证明里程碑奖”，以表彰他在数学定理机械证明研究领域中所作的开创性贡献。著有《数理逻辑概论》、《从数学到哲学》、《哥德尔》、《超越分析哲学》等专著。

# 王氏算法

1959年，王浩用他首创的“王氏算法”，在一台速度不高的IBM-704电脑上再次向《数学原理》发起挑战。不到9分钟，王浩的机器把这本数学史上视为里程碑的著作中全部（350条以上）的一阶逻辑定理，统统证明了一遍。

该书作者，数学大师罗素得知此事后感慨万端，他在信里写到：“我真希望，在怀海特和我浪费了10年的时间用手算来证明这些定理之前，就知道有这种可能。”王浩教授因此被国际上公认为机器定理证明的开拓者之一。

# 参考网址

- <https://baike.baidu.com/item/吴文俊/44938?fr=aladdin>
- <https://baike.baidu.com/item/几何定理机器证明/2197024?fr=aladdin>
- <https://baike.baidu.com/item/王浩/22564?fr=aladdin>
- [http://blog.sina.com.cn/s/blog\\_684b35950100n186.html](http://blog.sina.com.cn/s/blog_684b35950100n186.html)