

第 3 章 确定性推理方法



确定性推理方法

■ 3.1 推理的基本概念

■ 3.2 自然演绎推理

■ 3.3 谓词公式化为子句集的方法

■ 3.4 鲁宾逊归结原理

■ 3.5 归结反演

■ 3.6 应用归结反演求解问题

归
结
演
绎
推
理

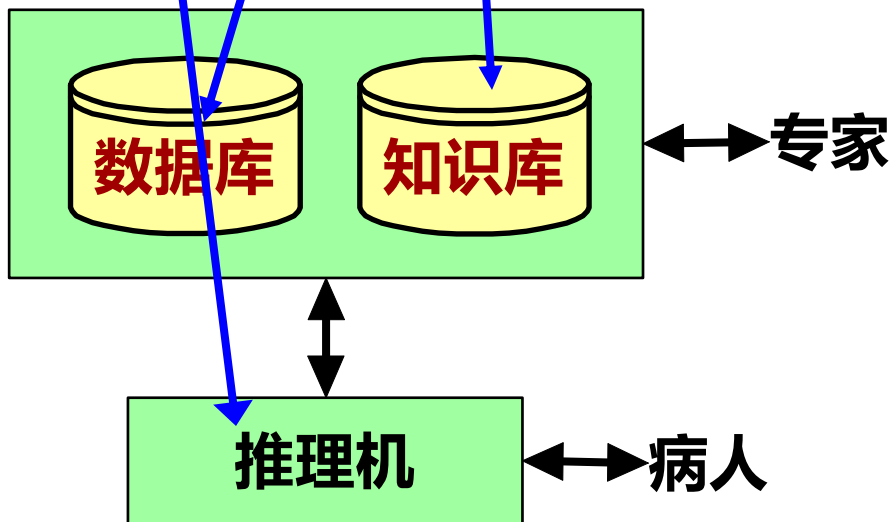
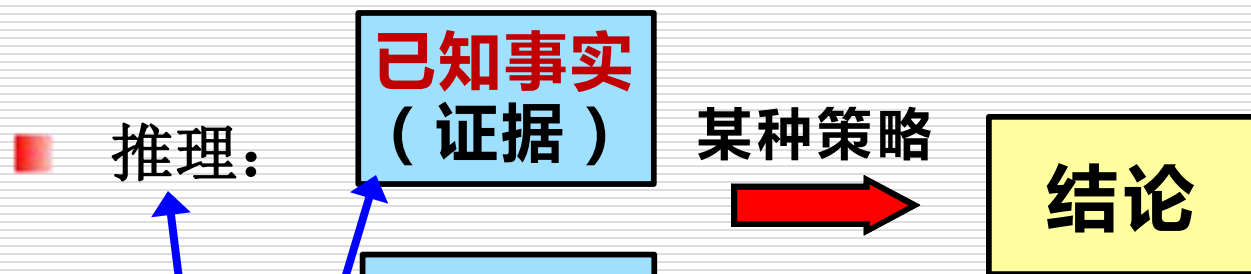
要求：了解推理的分类和自然演绎推理的特点，掌握谓词公式化为子句集的方法，熟练掌握归结原理、方法，并能灵活应用，掌握知识图谱中的**RDF**模型及归纳推理

重点：归结原理及应用

难点：知识图谱中的归纳推理

□ 补充：知识图谱中的知识表示、知识推理

3.1 推理的基本概念



医疗专家系统

知识	专家的经验、医学常识
初始证据	病人的症状、化验结果
证据	中间结论

3.1.2推理方式及其分类

1. 演绎推理、归纳推理、默认推理

1) **演绎推理** (deductive reasoning) : 一般 \rightarrow 个别

■ **三段论式** (三段论法)

① 所有的推理系统都是智能系统 ; (**大前提**)

② 专家系统是推理系统 ; (**小前提**)

③ 所以, 专家系统是智能系统。 (**结 论**)

特点: 没有增加新知识: 由大前提推出的
适合于小前提的新判断

3.1.2推理方式及其分类


1. 演绎推理、归纳推理、默认推理

2) **归纳推理** (inductive reasoning): 个别 \rightarrow 一般

{ 完全归纳推理 (必然性推理)
不完全归纳推理 (非必然性推理)

3) **默认推理** (default reasoning, 缺省推理)

- 知识不完全的情况下假设某些条件已经具备所进行的推理。

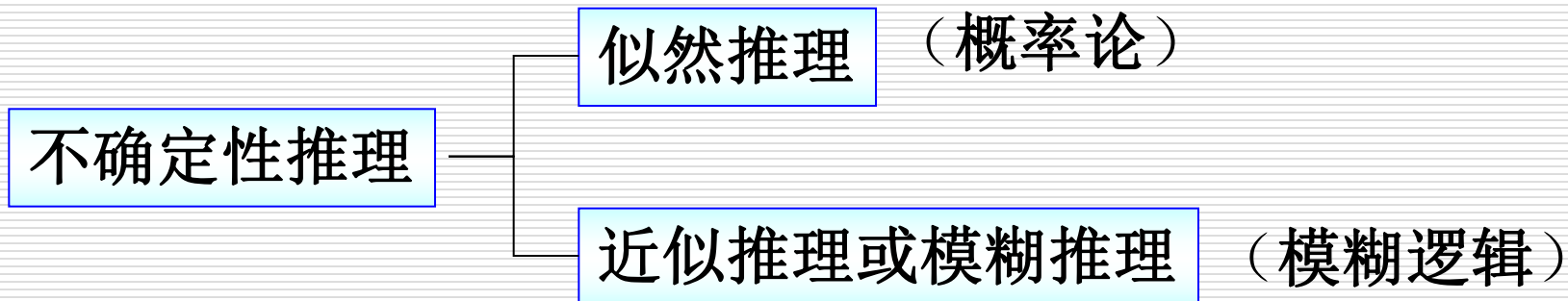
A 成立
B 成立?  结 论
(默认B成立)

3.1.2推理方式及其分类

2. 确定性推理、不确定性推理

1) **确定性推理**（精确推理）：推理时所用的知识与证据都是确定的，推出的结论也是确定的，其真值或者为真或者为假。

2) **不确定性推理**（不精确推理）：推理时所用的知识与证据不都是确定的，推出的结论也是不确定的。



3.1.2 推理方式及其分类

3. 单调推理、非单调推理

基于经典逻辑的演绎推理

- 1) **单调推理**: 随着推理向前推进及新知识的加入, 推出的结论越来越接近最终目标。
- 2) **非单调推理**: 由于新知识的加入, 不仅没有加强已推出的结论, 反而要否定它, 使推理退回到前面的某一步, 重新开始。

默认推理是非单调推理

X: 鸟 \rightarrow X: 不会飞

X: 企鹅 \rightarrow

3. 1. 2推理方式及其分类

4. 启发式推理、非启发式推理

- 启发性 (**heuristic**) 知识：与问题有关且能加快推理过程、提高搜索效率的知识。

- 推理目标：在脑膜炎、肺炎、流感中选择一个

- 产生式规则：

r_1 : 脑膜炎

r_2 : 肺炎

r_3 : 流感

- 启发式知识：“目前正在盛行流感”、“脑膜炎危险”

确定性推理方法（第3章）

■ 3.1 推理的基本概念

■ 3.2 自然演绎推理

■ 3.3 谓词公式化为子句集的方法

■ 3.4 鲁宾逊归结原理

■ 3.5 归结反演

■ 3.6 应用归结反演求解问题

归结
演绎
推理

要求：了解推理的分类和自然演绎推理的特点，掌握谓词公式化为子句集的方法，熟练掌握归结原理、方法，并能灵活应用，掌握知识图谱中的**RDF**模型及归纳推理

重点：归结原理及应用

难点：知识图谱中的归纳推理

□ 补充：知识图谱中的知识表示、知识推理

3.2 自然演绎推理



- ❁ 自然演绎推理：从一组已知为真的事实出发，运用**经典逻辑的推理规则**推出结论的过程。
- ❁ 推理规则：P规则、T规则、假言推理、拒取式推理等。


■ **假言推理**： $P \rightarrow Q, P \Rightarrow Q$ (肯定前件式)

■ “如果 x 是金属，则 x 能导电”，“铜是金属” 推出 “**铜能导电**”

■ **拒取式推理**： $P \rightarrow Q, \neg Q \Rightarrow \neg P$ (否定后件式)

■ “如果下雨，则地下就湿”，“地上不湿” 推出 “**没有下雨**”

3.2 自然演绎推理

 **错误1**——否定前件: $P \rightarrow Q, \neg P \not\Rightarrow \neg Q$

- (1) 如果下雨, 则地上是湿的 ($P \rightarrow Q$);
- (2) 没有下雨 ($\neg P$);
- (3) 所以, 地上不湿 ($\neg Q$)。

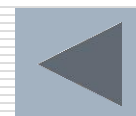
 **错误2**——肯定后件: $P \rightarrow Q, Q \not\Rightarrow P$

- (1) 如果行星系统是以太阳为中心的, 则金星会显示出位相变化 ($P \rightarrow Q$);
- (2) 金星显示出位相变化 (Q);
- (3) 所以, 行星系统是以太阳为中心 (P)。

3.2 自然演绎推理

■ 谓词逻辑的推理规则：

- ① **P规则**：在推理的任何步骤上都可引入前提。
- ② **T规则**：在推理过程中，如果前面步骤中有一个或多个谓词公式永真蕴含谓词公式 S ，则可将 S 引入推理过程中。



3.2 自然演绎推理

❖ 例 3.1

■ 已知事实:

- (1) 凡是容易的课程小王(wang)都喜欢;
- (2) c 班的课程都是容易的;
- (3) ds 是 c班的一门课程。

■ 求证: 小王喜欢 ds 这门课程。

3.2 自然演绎推理



证明:

■ 定义谓词:

$\text{easy}(X)$: X 是容易的

$\text{like}(X, Y)$: X 喜欢 Y

$\text{c}(X)$: X 是 c 班的一门课程

■ 已知事实和结论用谓词公式表示:

$(\forall x) (\text{easy}(x) \rightarrow \text{like}(\text{wang}, x))$

$(\forall x) (\text{c}(x) \rightarrow \text{easy}(x))$

$\text{c}(\text{ds})$

结论: $\text{like}(\text{wang}, \text{ds})$

■ 已知事实:

(1) 凡是容易的课程小王(wang)都喜欢;

(2) c 班的课程都是容易的;

(3) ds 是 c 班的一门课程。

■ 求证: 小王喜欢 ds 这门课程。

3.2 自然演绎推理

- 应用推理规则进行推理：

$(\forall x) (\text{easy} (x) \rightarrow \text{like} (\text{wang}, x))$

$(\forall x) (c (x) \rightarrow \text{easy} (x))$

$c (\text{ds})$

$\text{like} (\text{wang}, \text{ds})$

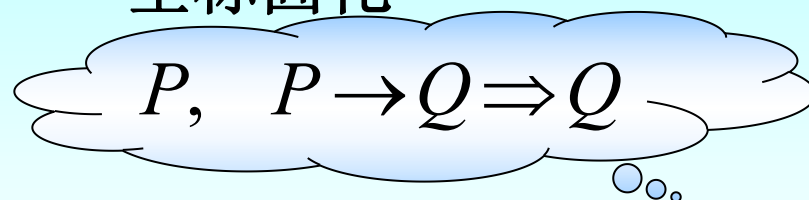
$(\forall x) (\text{easy} (x) \rightarrow \text{like} (\text{wang}, x))$
 $\hookrightarrow \text{easy} (Z) \rightarrow \text{like} (\text{wang}, Z)$ 全称固化

$(\forall x) (c (x) \rightarrow \text{easy} (x))$
 $\hookrightarrow c (Y) \rightarrow \text{easy} (Y)$

$\therefore c (\text{ds}), c (Y) \rightarrow \text{easy} (Y)$
 $\Rightarrow \text{easy} (\text{ds})$

$\therefore \text{easy} (\text{ds}), \text{easy} (Z) \rightarrow \text{like} (\text{wang}, Z)$
 $\Rightarrow \text{like} (\text{wang}, \text{ds})$


全称固化




P规则及假言推理

T规则及假言推理

3.2 自然演绎推理

 优点：

- 表达定理证明过程自然，易理解，
- 拥有较丰富的推理规则，推理过程灵活，
- 便于嵌入领域启发式知识。

 缺点：易产生组合爆炸，得到的中间结论一般呈指数形式递增。

确定性推理

- **Datalog**: 面向知识库和数据库设计的逻辑语言, 表达能力与**OWL**相当, 支持递归; 便于撰写规则, 实现推理
- **RDFox** (牛津大学开发的**RDF**三元组存储系统) 支持并行**Datalog**推理

■ 3.1 推理的基本概念

■ 3.2 自然演绎推理 (例如**Datalog**推理)

■ 3.3 谓词公式化为子句集的方法

■ 3.4 鲁宾逊归结原理

■ 3.5 归结反演

■ 3.6 应用归结反演求解问题

归
结
演
绎
推
理

要求: 了解推理的分类和自然演绎推理的特点, 掌握谓词公式化为子句集的方法, 熟练掌握归结原理、方法, 并能灵活应用, 掌握知识图谱中的**RDF**模型及归纳推理

重点: 归结原理及应用

难点: 知识图谱中的归纳推理

□ 补充: 知识图谱中的知识表示、知识推理

归结演绎推理

■ 反证法: $P \Rightarrow Q$, 当且仅当 $P \wedge \neg Q \Leftrightarrow F$,
即 Q 为 P 的逻辑结论, 当且仅当 $P \wedge \neg Q$ 是不可满足的。

■ 定理: Q 为 P_1, P_2, \dots, P_n 的逻辑结论, 当且仅当
 $(P_1 \wedge P_2 \wedge \dots \wedge P_n) \wedge \neg Q$ 是不可满足的。

归结演绎推理

■ 定理: Q 为 P_1, P_2, \dots, P_n 的逻辑结论, 当且仅当 $(P_1 \wedge P_2 \wedge \dots \wedge P_n) \wedge \neg Q$ 是不可满足的。

(1)能阅读者是识字的;

(2)海豚不识字;

(3)有些海豚是聪明的;

归结原理: 一种基于一阶谓词逻辑知识表示的演绎推理方法

请用归结原理证明: 有些聪明者并不能阅读。

已知, 谓词 $R(x)$ 表示 x 能阅读, $L(x)$ 表示识字,

$D(x)$ 表示 x 是海豚, $I(x)$ 表示聪明的

a. 把已知事实和结论的否定用谓词公式表示

b. 要证已知事实和结论的否定谓词公式集是不可满足的!
如何证明?

确定性推理方法（第3章）

- 3.1 推理的基本概念
- 3.2 自然演绎推理
- 3.3 谓词公式化为子句集的方法
- 3.4 鲁宾逊归结原理
- 3.5 归结反演
- 3.6 应用归结反演求解问题

归
结
演
绎
推
理

3.3 谓词公式化为子句集的方法

■ 原子谓词公式: 不含连接词的单个谓词。

■ 文字: 原子谓词公式及其否定。

正负文字
互补!

$R(x)$: 正文字, $\neg R(x)$: 负文字。

■ 子句: 任何文字的析取式。

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

单文字子句: 只含有一个文字的子句。

空子句 (NIL): 不包含任何文字的子句。

■ 子句集: 由子句构成的集合。

空子句是永假的,
不可满足的。

谓词公式 \rightarrow 子句集 ?

谓词公式化为子句集的九个步骤：

(1) 消去谓词公式中的 “ \rightarrow ” 和 “ \leftrightarrow ” 符号

(2) 把否定符号 \neg 移到紧靠谓词的位置上

(3) 变量标准化 ○ ○ ○

保证每个量词变元都不同名！

(4) 消去存在量词：存在固化 或 Skolem 化

(5) 化为前束形

(6) 化为 Skolem 标准形（子句的合取式）

(7) 略去全称量词

(8) 消去合取符号把母式用子句集表示

(9) 子句变量标准化 ○ ○ ○

保证每个子句的变量都不同名！

课堂测试题

已知:

(1) Paul喜欢酒 (wine)。

(2) Paul不喜欢奶酪 (cheese)

(3) 如果Paul喜欢某物则John也喜欢某物。

(4) 如果某人是贼, 而且他喜欢某物, 则他可能会偷窃某物。

(5) John是贼。

求证John可能会偷酒(wine)。

现定义如下谓词:

$\text{thief}(x)$: 某人 x 是贼;

$\text{likes}(x,y)$: 某人 x 喜欢某物 y ;

$\text{may-steal}(x,y)$: 某人 x 可能偷窃某物 y 。

(1) $\text{likes}(\text{Paul}, \text{wine})$

(2) $\neg \text{likes}(\text{Paul}, \text{cheese})$

(3) $\exists x(\text{likes}(\text{Paul}, x) \rightarrow \text{likes}(\text{John}, x))$

(4) $\exists x \exists y(\text{thief}(x) \wedge \text{likes}(x, y) \rightarrow \text{may_steal}(x, y))$

(5) $\text{thief}(\text{John})$

(6) $\neg \text{may_steal}(\text{John}, \text{wine})$

即要证公式集 $\{(1), (2), (3), (4), (5), (6)\}$ 是不可满足的!
如何证明?

3.3 谓词公式化为子句集的方法

■ 例，将下列谓词公式化为子句集。

$$(4) \exists x \exists y (thief(x) \wedge likes(x, y) \rightarrow may_steal(x, y))$$

■ 解：（1）消去谓词公式中的 “ \rightarrow ” 和 “ \leftrightarrow ” 符号

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

$$\Leftrightarrow \exists x \exists y (\neg(thief(x) \wedge likes(x, y)) \vee may_steal(x, y))$$

（2）把否定符号 \neg 移到紧靠谓词的位置上

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

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

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

$$\Leftrightarrow \exists x \exists y (\neg thief(x) \vee \neg likes(x, y) \vee may_steal(x, y))$$

3.3 谓词公式化为子句集的方法

■ 例 将下列谓词公式化为子句集。

$$\longleftrightarrow \exists x \exists y (\neg thief(x) \vee \neg likes(x, y) \vee may_steal(x, y))$$

(3) 变量标准化 $(\exists x)P(x) \Leftrightarrow (\exists y)P(y), (\forall x)P(x) \Leftrightarrow (\forall y)P(y)$

$$\longleftrightarrow \forall x \exists y (\neg thief(x) \vee \neg likes(x, y) \vee may_steal(x, y))$$

(4) 消去存在量词

- a. 存在量词不出现在全称量词的辖域内。 $(\exists x)P(x) \Rightarrow P(a)$
- b. 存在量词出现在一个或者多个全称量词的辖域内。

$$\longleftrightarrow \neg thief(a) \vee \neg likes(a, b) \vee may_steal(a, b)$$

3.3 谓词公式化为子句集的方法

(4) 消去存在量词

对于一般情况

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

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

Skolem化: 用Skolem函数代替每个存在量词量化的变量的过程。

$$\begin{array}{l} y=f(x) \\ \longleftrightarrow (\forall x)((\exists y)\neg P(x, y) \vee Q(x, y)) \\ (\forall x)(\neg P(x, f(x)) \vee Q(x, y)) \end{array}$$

这里的 y 需要Skolem化吗?

3.3 谓词公式化为子句集的方法

$$\longleftrightarrow \neg thief(a) \vee \neg likes(a,b) \vee may_steal(a,b)$$

(5) 化为前束形

(前缀) : 全称量词串;
{母式}: 不含量词的谓词公式。

■ 前束形 = (前缀) {母式}

$$\longleftrightarrow \neg thief(a) \vee \neg likes(a,b) \vee may_steal(a,b)$$

(6) 化为 **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。

$$P \vee (Q \wedge R) \Leftrightarrow \underline{(P \vee Q) \wedge (P \vee R)}$$

$$\underline{P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge R)}$$

哪边是Skolem
标准形?

$$\longleftrightarrow \neg thief(a) \vee \neg likes(a,b) \vee may_steal(a,b)$$

3.3 谓词公式化为子句集的方法

$$\longleftrightarrow \neg thief(a) \vee \neg likes(a,b) \vee may_steal(a,b)$$

(7) 略去全称量词

$$\longleftrightarrow \neg thief(a) \vee \neg likes(a,b) \vee may_steal(a,b)$$

(8) 消去合取符号把母式用子句集表示

$$\longleftrightarrow \neg thief(a) \vee \neg likes(a,b) \vee may_steal(a,b)$$

(9) 子句变量标准化

保证每个子句的变量都不同名!

$$\longleftrightarrow \{\neg thief(a) \vee \neg likes(a,b) \vee may_steal(a,b)\}$$

课堂测试题

1. 已知:

1) Paul喜欢酒 (wine)

2) Paul不喜欢奶酪 (cheese)。

3) 如果Paul喜欢某物则John也喜欢某物。

4) 如果某人是贼, 而且他喜欢某物, 则他可能会偷窃某物。

5) John是贼。

求证John可能会偷酒(wine)。

(1) $likes(\text{Paul}, \text{wine})$

(2) $\neg likes(\text{Paul}, \text{cheese})$

(3) $\exists x(likes(\text{Paul}, x) \rightarrow likes(\text{John}, x))$

(4) $\forall x \exists y(thief(x) \wedge likes(x, y) \rightarrow may_steal(x, y))$

(5) $thief(\text{John})$

(6) $\neg may_steal(\text{John}, \text{wine})$

子句集:

(1) $likes(\text{Paul}, \text{wine})$

(2) $\neg likes(\text{Paul}, \text{cheese})$

(3) $\neg likes(\text{Paul}, c) \vee likes(\text{John}, c)$

(4) $\neg thief(a) \vee \neg likes(a, b) \vee may_steal(a, b)$

(5) $thief(\text{John})$

(6) $\neg may_steal(\text{John}, \text{wine})$

注意: 子句集中每个子句的变量都不同名!

定理 3.1: 谓词公式不可满足的充要条件是其子句集不可满足。

确定性推理方法（第3章）

■ 3.1 推理的基本概念

■ 3.2 自然演绎推理

■ 3.3 谓词公式化为子句集的方法

✓ 3.4 鲁宾逊归结原理

■ 3.5 归结反演

■ 3.6 应用归结原理求解问题

网络教学平台
作业W3-1

归
结
演
绎
推
理

3.4 鲁宾逊归结原理

◆ **子句**: 任何文字的**析取式**。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。

如何归结？

子句集：

- (1) $likes$ (Paul, wine)
- (2) $\neg likes$ (Paul, cheese)
- (3) $\neg likes$ (Paul, c) $\vee likes$ (John, c)
- (4) $\neg thief$ (a) $\vee \neg likes$ (a, b) $\vee may_steal$ (a, b)
- (5) $thief$ (John)
- (6) $\neg may_steal$ (John, wine)

- **空子句 (NIL)** : 不包含任何文字的子句。
- 如果一个子句中含有 $R(x) \vee \neg R(x)$ ，则可以消去。

3.4 鲁宾逊归结原理

1. 命题逻辑中的归结原理（**基子句的归结**）

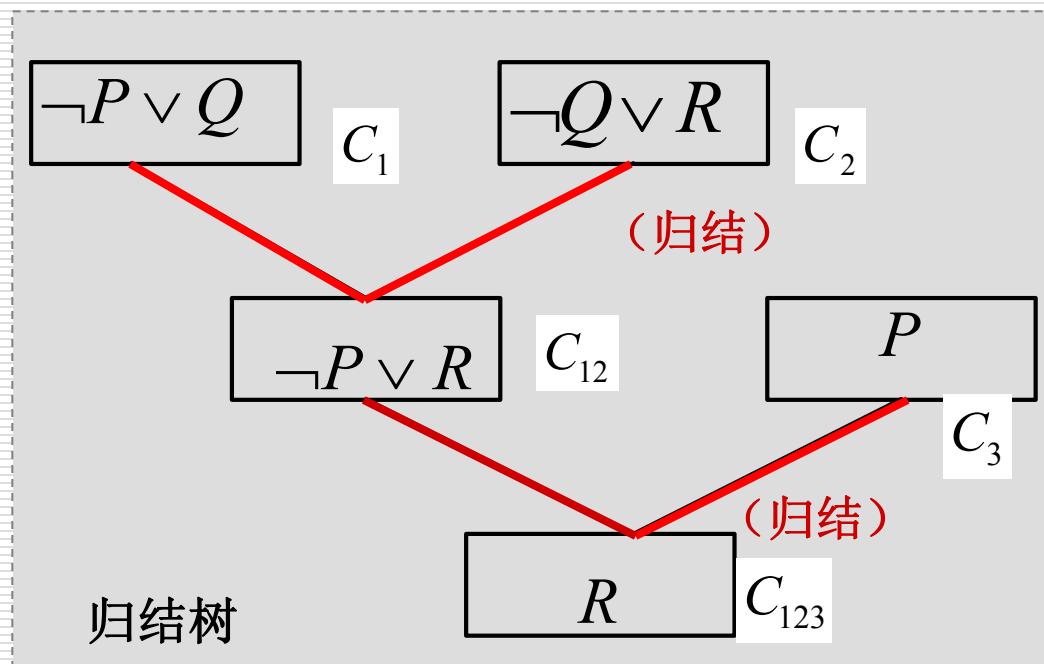
$Q \vee \neg Q$ 永真

- 定义3.1（**归结**）：设 C_1 与 C_2 是子句集中的任意两个子句，如果 C_1 中的文字 L_1 与 C_2 中的文字 L_2 **互补**，那么从 C_1 和 C_2 中分别消去 L_1 和 L_2 ，并将二个子句中余下的部分**析取**，构成一个**新子句** C_{12} 。

C_{12} : C_1 和 C_2 的归结式

C_1 、 C_2 : C_{12} 的亲本子句

例，设 $C_1 = \neg P \vee Q$ ，
 $C_2 = \neg Q \vee R$ ， $C_3 = P$



3.4 鲁宾逊归结原理

◆ 定理3.2: 归结式 C_{12} 是其亲本子句 C_1 与 C_2 的逻辑结论。即如果 C_1 与 C_2 为真, 则 C_{12} 为真。

◆ 推论1: 设 C_1 与 C_2 是子句集 S 中的两个子句, C_{12} 是它们的归结式, 若用 C_{12} 代替 C_1 与 C_2 后得到新子句集 S_1 , 则由 S_1 不可满足性可推出原子句集 S 的不可满足性, 即:

S_1 的不可满足性 $\Rightarrow S$ 的不可满足性

归结方法!

◆ 推论2: 设 C_1 与 C_2 是子句集 S 中的两个子句, C_{12} 是它们的归结式, 若 C_{12} 加入原子句集 S , 得到新子句集 S_2 , 则 S 与 S_1 在不可满足的意义上是等价的, 即:

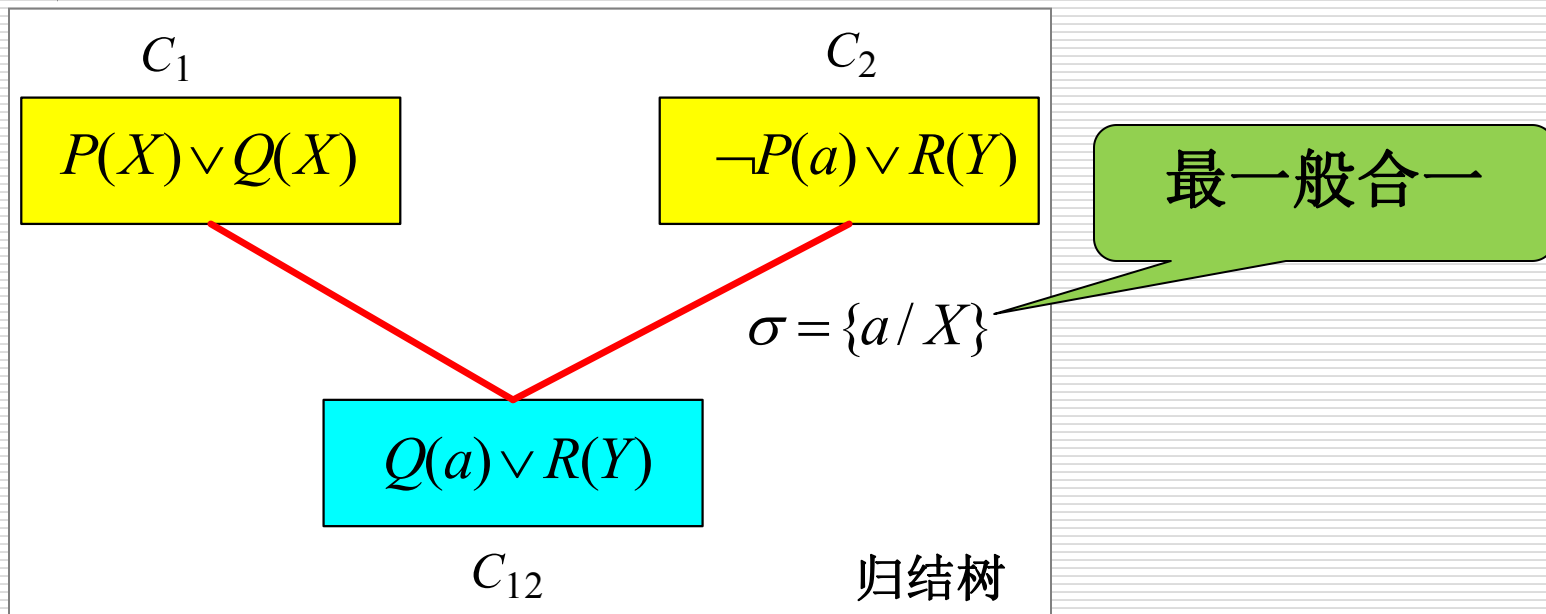
S_2 的不可满足性 $\Leftrightarrow S$ 的不可满足性

3.4 鲁宾逊归结原理

2. 谓词逻辑中的归结原理（含有变量的子句的归结）

例: $C_1 = P(X) \vee Q(X)$ $\sigma = \{a/X\}$ $C_1\sigma = P(a) \vee Q(a)$
 $C_2 = \neg P(a) \vee R(Y)$ $C_2\sigma = \neg P(a) \vee R(Y)$
 $C_{12} = Q(a) \vee R(Y)$

?



3.4 鲁宾逊归结原理

■ 已知两个子句如下：

$$C_1 = R(c) \vee P(f(a)) \vee Q(X),$$

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

求其二元归结式。

注意：子句集中每个子句的变量都不同名！

得：

$$C_{12} = R(c) \vee Q(X) \vee R(b)$$

$$C_1\sigma = R(c) \vee P(f(a)) \vee Q(f(a))$$

$$C_2\sigma = \neg P(f(a)) \vee R(b)$$

$$\therefore C_{12} = R(c) \vee Q(f(a)) \vee R(b)$$



3.4 鲁宾逊归结原理

- 对于命题逻辑、谓词逻辑中的归结原理：
 - ◆ 归结式是其亲本子句的逻辑结论。
 - ◆ 若从子句集存在一个到空子句的归结演绎，则该子句集是不可满足的。
 - ◆ 若子句集是不可满足的，则必存在一个从该子句集到空子句的归结演绎；

归结原理是完备的

注意：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。

一阶谓词逻辑的确定性推理方法（第3章）

■ 3.1 推理的基本概念

■ 3.2 自然演绎推理

■ 3.3 谓词公式化为子句集的方法

■ 3.4 鲁宾逊归结原理

✓ 3.5 归结反演

■ 3.6 应用归结原理求解问题

归
结
演
绎
推
理

翻页

3.5 归结反演

反证法: $P_1, P_2, \dots, P_n \rightarrow Q \Leftrightarrow (P_1 \wedge P_2 \wedge \dots P_n) \wedge \neg Q$ 是不可满足的

■ **归结反演**: 应用归结原理证明定理的过程。

- 设 **F** : 已知前提的谓词公式集 $\{P_1, P_2, \dots, P_n\}$, **Q** : 目标 (结论) 的谓词公式。
- 用归结反演证明 **Q** 为真的步骤:
 - 1) 把谓词公式集 **$\{F, \neg Q\}$** 化为子句集 **S** ;
 - 2) 对 **S** 中的子句进行归结, 把归结式都并入到 **S** 中。
 - 3) 重复2), **若出现空子句, 则停止归结**, 即证明 **Q** 为真。

课堂测试题

1.已知:

1) Paul喜欢酒 (wine) 。

2) Paul不喜欢奶酪 (cheese) 。

3) 如果Paul喜欢某物则John也喜欢某物。

4)如果某人是贼，而且他喜欢某物，则他可能会偷窃某物。

5) John是贼。

求证John可能会偷酒(wine)。

■ 现定义如下谓词:

thief(x): 某人x是贼;

likes(x,y): 某人x喜欢某物y;

may-steal(x,y):某人x可能偷窃某物y。

3.3 谓词公式化为子句集的方法

■ 把已知事实和结论的否定用谓词公式集表示

- (1) $likes(Paul, wine)$
- (2) $\neg likes(Paul, cheese)$
- (3) $\exists x(likes(Paul, x) \rightarrow likes(John, x))$
- (4) $\exists x \exists y(thief(x) \wedge likes(x, y) \rightarrow may_steal(x, y))$
- (5) $thief(John)$
- (6) $\neg may_steal(John, wine)$

■ 把上述谓词公式集{(1), (2), (3), (4), (5), (6)}化成子句集:

子句集:

- (1) $likes(Paul, wine)$
- (2) $\neg likes(Paul, cheese)$
- (3) $\neg likes(Paul, c) \vee likes(John, c)$
- (4) $\neg thief(a) \vee \neg likes(a, b) \vee may_steal(a, b)$
- (5) $thief(John)$
- (6) $\neg may_steal(John, wine)$

1.已知:

- 1) Paul喜欢酒 (wine) 。
- 2) Paul不喜欢奶酪 (cheese) 。
- 3) 如果Paul喜欢某物则John也喜欢某物。
- 4) 如果某人是贼, 而且他喜欢某物, 则他可能会偷窃某物。
- 5) John是贼。



归结?

3.5 归结反演

子句集:

- (1) *likes* (Paul, wine)
- (2) $\neg \textit{likes}$ (Paul, cheese)
- (3) $\neg \textit{likes}$ (Paul, *c*) \vee *likes* (John, *c*)
- (4) $\neg \textit{thief}$ (*a*) \vee $\neg \textit{likes}$ (*a*, *b*) \vee *may _ steal* (*a*, *b*)
- (5) *thief* (John)
- (6) $\neg \textit{may _ steal}$ (John , wine)

■ 应用归结原理进行归结:

(7) *likes*(John, wine)

(8) $\neg \textit{likes}$ (John, *b*) \vee *may _ steal*(John, *b*)

(9) *may _ steal*(John, wine)

(10) *NIL*

即证, **John可能会偷酒(wine)** 。

1.已知:

1) Paul喜欢酒 (wine) 。

2) Paul不喜欢奶酪 (cheese) 。

3) 如果Paul喜欢某物则John也喜欢某物。

4) 如果某人是贼, 而且他喜欢某物, 则他可能会偷窃某物。

5) John是贼。

(1) 与 (3) 归结

(4) 与 (5) 归结

(7) 与 (8) 归结

(9) 与 (6) 归结

归结策略

- 对子句集进行归结时，关键的一步是**从子句集中找出可进行归结的子句**。
- **穷举式归结**：对子句集中的所有子句逐对地进行比较，对任何一对可归结的子句对都进行归结。
- 采用**穷举式归结**对于一个规模较大的实际问题来说，其时空开销就很大。
- 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**。

归结策略

解决问题的关键在于选择有利于导致快速产生空子句的子句对进行归结。

归结策略

- 穷举式策略：穷尽子句比较的复杂搜索方法，例如宽度优先策略。
- 限制性策略：尽量缩小归结范围，例如支撑集策略、线性输入策略、祖先过滤策略等
- 有序性策略：给子句安排一定的顺序进行归结，例如单文字优先策略等
- 简化性策略：尽量简化子句集，例如删除策略（在归结过程中随时删除无用的子句）

1.宽度优先策略

设初始子句集为 S_0 ,归结过程如下:

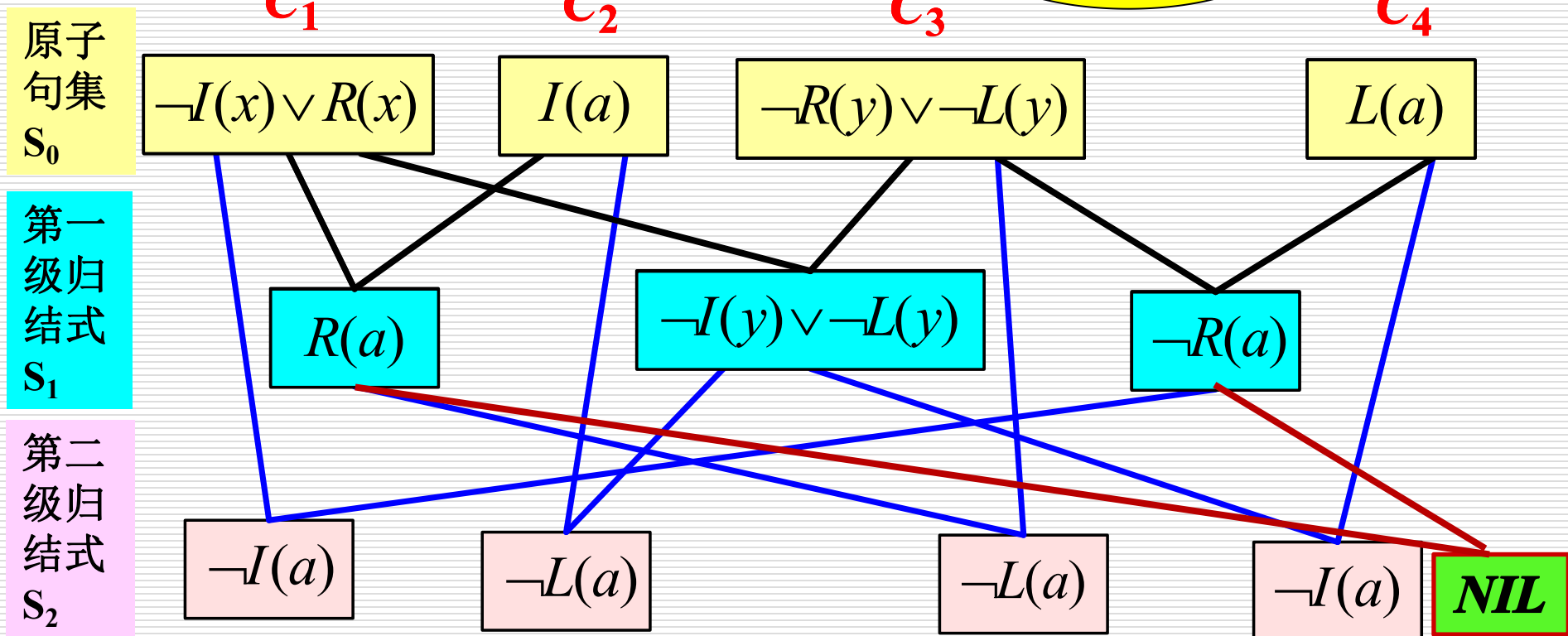
1. 从 S_0 出发,对 S_0 中的全部子句作所有可能的归结,得到第一层归结式,把这些归结式的集合记为 S_1 。
2. 对 S_0 中的子句与 S_1 中的子句作所有可能的归结,得到第二层归结式,把这些归结式的集合记为 S_2 。
3. 用 S_0 和 S_1 中的子句与 S_2 中的子句作所有可能的归结,得到第三层归结式,把这些归结式的集合记为 S_3 。

如此继续,直到得到空子句。

归结策略

当问题有解时，用广度优先策略归结能保证找到最短路径。因此，它是一种完备的归结策略。

1. 宽度优先策略

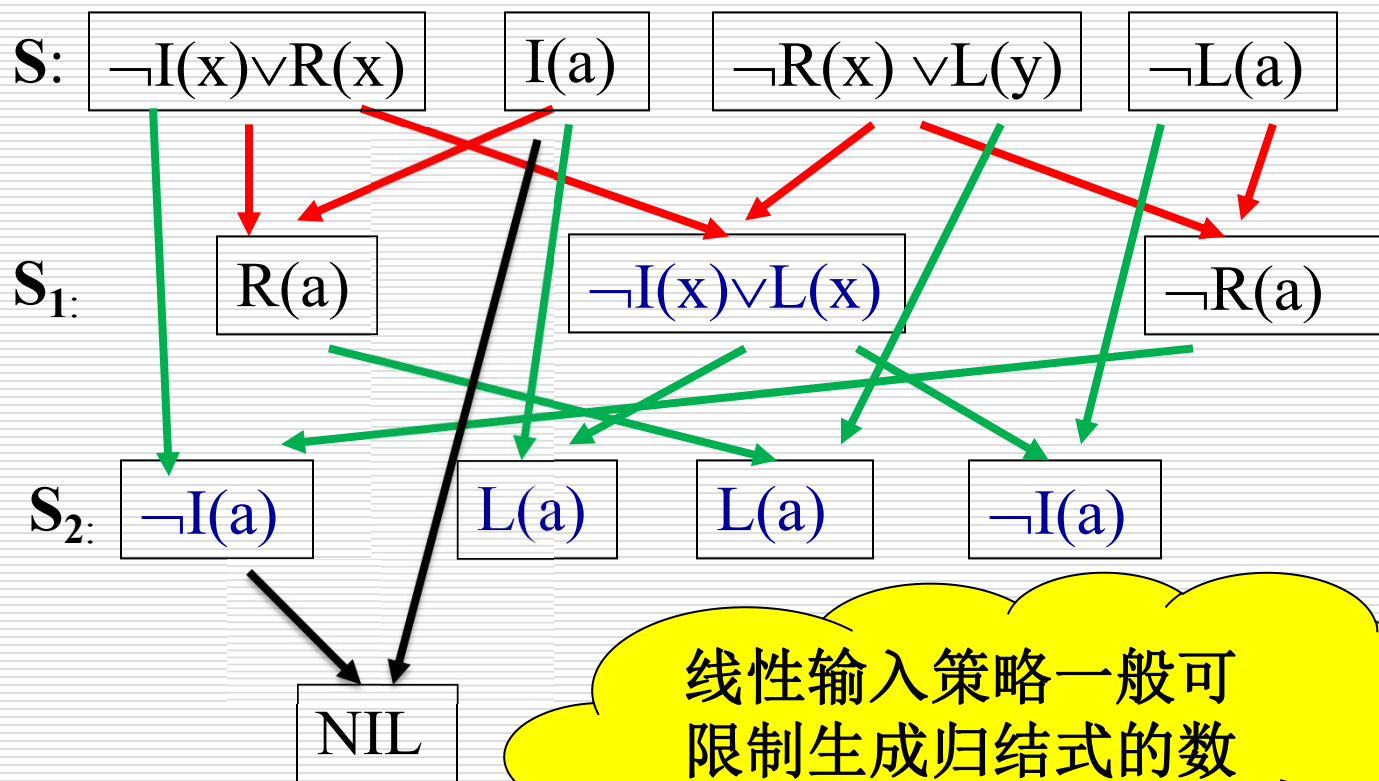


归结策略

2. 线性输入策略

每次参加归结的各个亲本子句中，至少应该有一个是初始子句集中的子句。

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



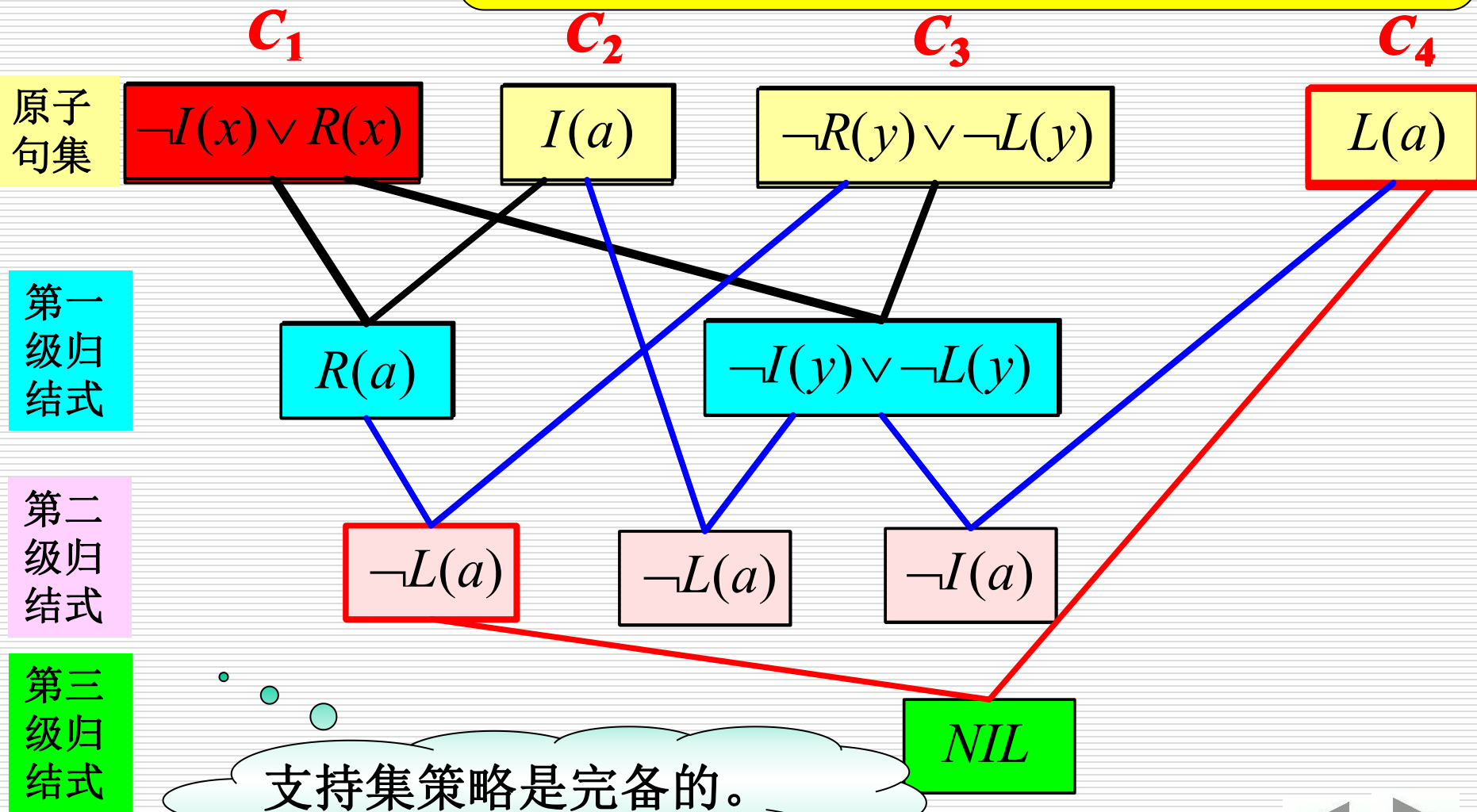
线性输入策略一般可限制生成归结式的数目，但不是一种不完备的策略。



归结策略

3. 支持集策略

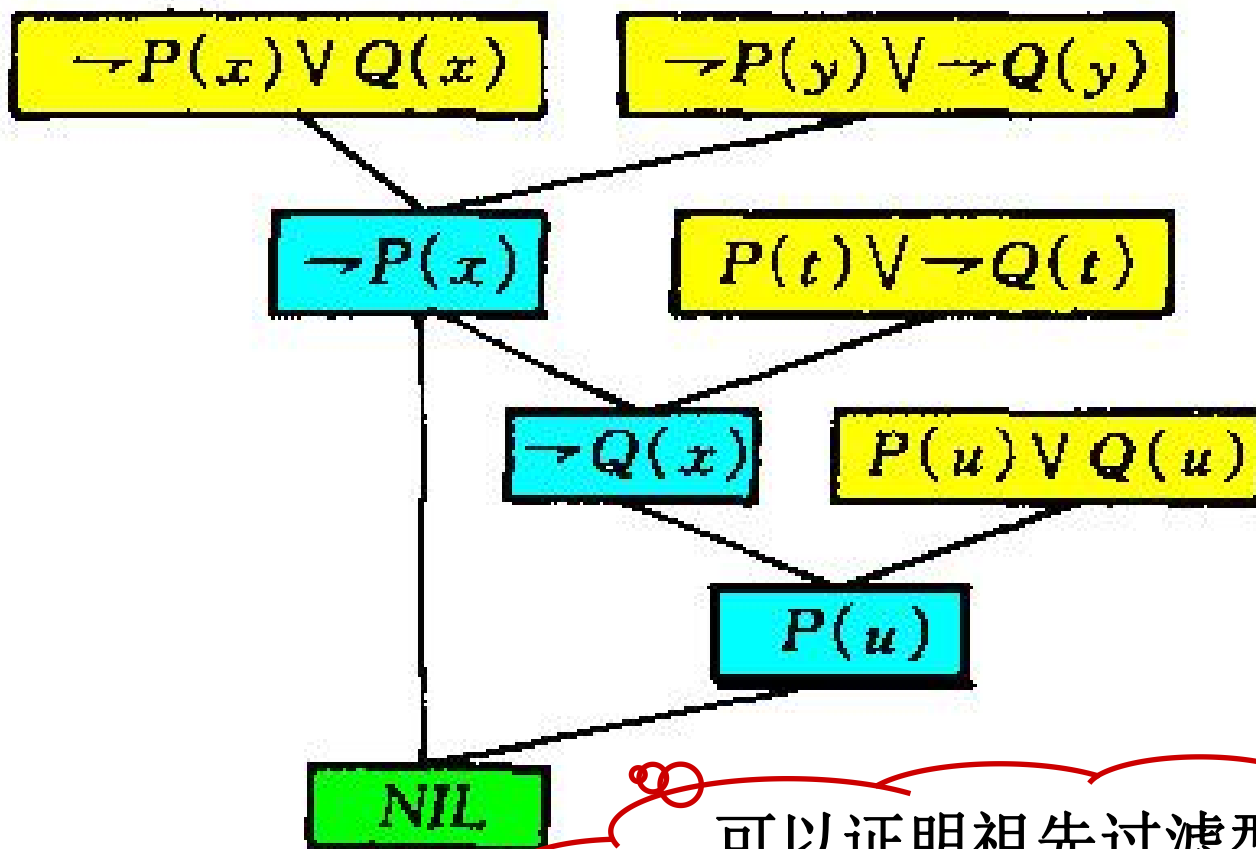
每次归结时父辈子句中至少有一个目标公式否定所得到的子句或其后代。



归结策略

■ 4. 祖先过滤型策略

每次归结时父辈子句中至少有一个是原子句集的子句或一个是另一个的祖先。

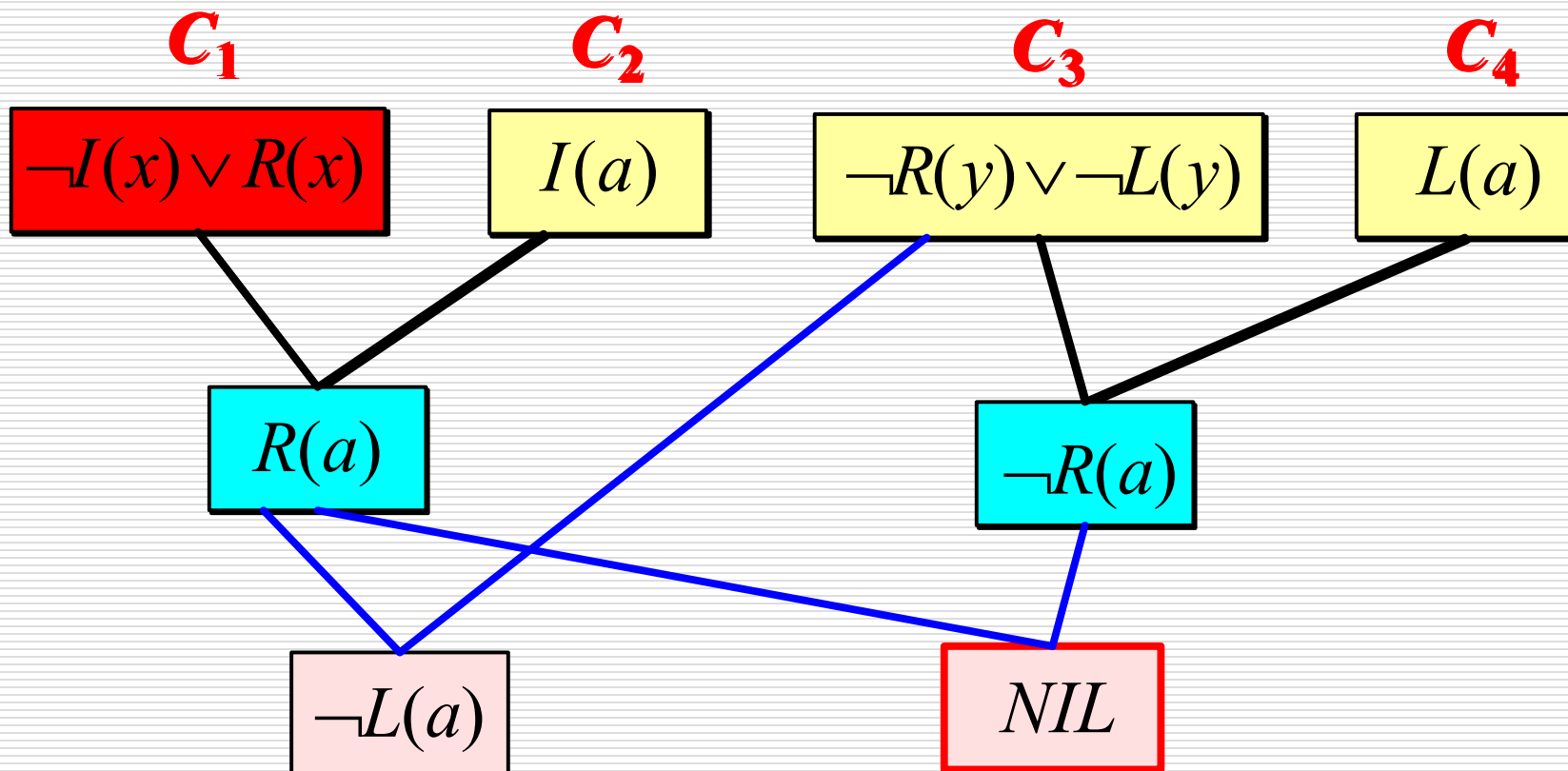


可以证明祖先过滤型策略是完备的。

归结策略

5. 单文字子句优先策略

每次归结时父辈子句中至少有一个是单文字子句。



单文字子句优先策略是不完备的。



6. 删除策略

归结时将**无用**的子句**删除**掉，缩小搜索范围，减少比较次数，从而提高归结效率。

常用的删除方法：

(1) 纯文字删除法

纯文字：如果文字L在子句集中不存在与其互补的文字 $\neg L$ ，则称该文字为纯文字。

例：对于子句集 $S = \{P \vee Q \vee R, \neg Q \vee R, Q, \neg R\}$

其中P为纯文字，因此， $P \vee Q \vee R$ 可从S中删除。

(2) 重言式删除法

重言式： 如果一个子句中包含有互补的文字对，则称该子句为重言式。

例： $P(x) \vee Q(x) \vee \neg P(x)$

删除重言式后： $Q(x)$

归结演绎推理的归结策略

- 宽度优先策略
- 支持集策略
- 线性输入策略
- 祖先过滤策略
- 单文字子句策略
- 删除策略

在选择归结策略时，主要应考虑其完备性和效率问题。

确定性推理方法（第3章）

- 3.1 推理的基本概念
- 3.2 自然演绎推理
- 3.3 谓词公式化为子句集的方法
- 3.4 鲁宾逊归结原理
- 3.5 归结反演
- ✓ 3.6 应用归结原理求解问题

归
结
演
绎
推
理

课堂测试题

1.用子句集表示下列刑侦知识。

1) Paul喜欢酒 (wine) 。

2) Paul不喜欢奶酪 (cheese) 。

3) 如果Paul喜欢某物则John也喜欢某物。

4)如果某人是贼，而且他喜欢某物，则他可能会偷窃某物。

5) John是贼。

2. 问John可能会偷窃什么？（给出归结策略和归结树）

现定义如下谓词：

thief(x): 某人x是贼；

likes(x,y): 某人x喜欢某物y；

may-steal(x,y):某人x可能偷窃某物y。

3.6 应用归结原理求解问题

■ 应用归结原理求解问题的步骤:

- (1) 已知前提用谓词公式 F 表示;
- (2) 把待求解的问题用谓词公式 Q 表示, 并否定 Q , 再与 answer 构成析取式 $(\neg Q \vee \text{answer})$;
- (3) 把公式集 $\{F, \neg Q \vee \text{answer}\}$ 化为子句集 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。

answer 是专设的谓词, 其个体变元与 Q 的个体变元一样

课堂测试题

1. 已知:

1) Paul喜欢酒 (wine) 。

2) Paul不喜欢奶酪 (cheese) 。

3) 如果Paul喜欢某物则John也喜欢某物。

4) 如果某人是贼, 而且他喜欢某物, 则他可能会偷窃某物。

5) John是贼。

2. 问John可能会偷窃什么?
(给出归结策略和归结树)

(1) *likes* (Paul, wine)

(2) \neg *likes* (Paul, cheese)

(3) $\exists x(\text{likes}(\text{Paul}, x) \rightarrow \text{likes}(\text{John}, x))$

(4) $\exists x \exists y(\text{thief}(x) \wedge \text{likes}(x, y) \rightarrow \text{may_steal}(x, y))$

(5) *thief* (John)

(6) $\neg \text{may_steal}(\text{John}, x) \vee \text{answer}(x)$

子句集:

(1) *likes* (Paul, wine)

(2) \neg *likes* (Paul, cheese)

(3) \neg *likes* (Paul, *c*) \vee *likes* (John, *c*)

(4) \neg *thief* (*a*) \vee \neg *likes* (*a*, *b*) \vee *may_steal* (*a*, *b*)

(5) *thief* (John)

(6) $\neg \text{may_steal}(\text{John}, x) \vee \text{answer}(x)$

子句集:

(1) *likes* (Paul, wine)

(2) \neg *likes* (Paul, cheese)

(3) \neg *likes* (Paul, *c*) \vee *likes* (John, *c*)

(4) \neg *thief* (*a*) \vee \neg *likes* (*a*, *b*) \vee *may* _ *steal* (*a*, *b*)

(5) *thief* (John)

(6) \neg *may* _ *steal* (John , *x*) \vee *answer* (*x*)

- 按支持集策略和单文字优先相结合，将子句集进行归结：

$$C_{64} = \neg \textit{thief}(\textit{John}) \vee \neg \textit{likes}(\textit{John}, x) \vee \textit{answer}(x)$$

$$C_{645} = \neg \textit{likes}(\textit{John}, x) \vee \textit{answer}(x)$$

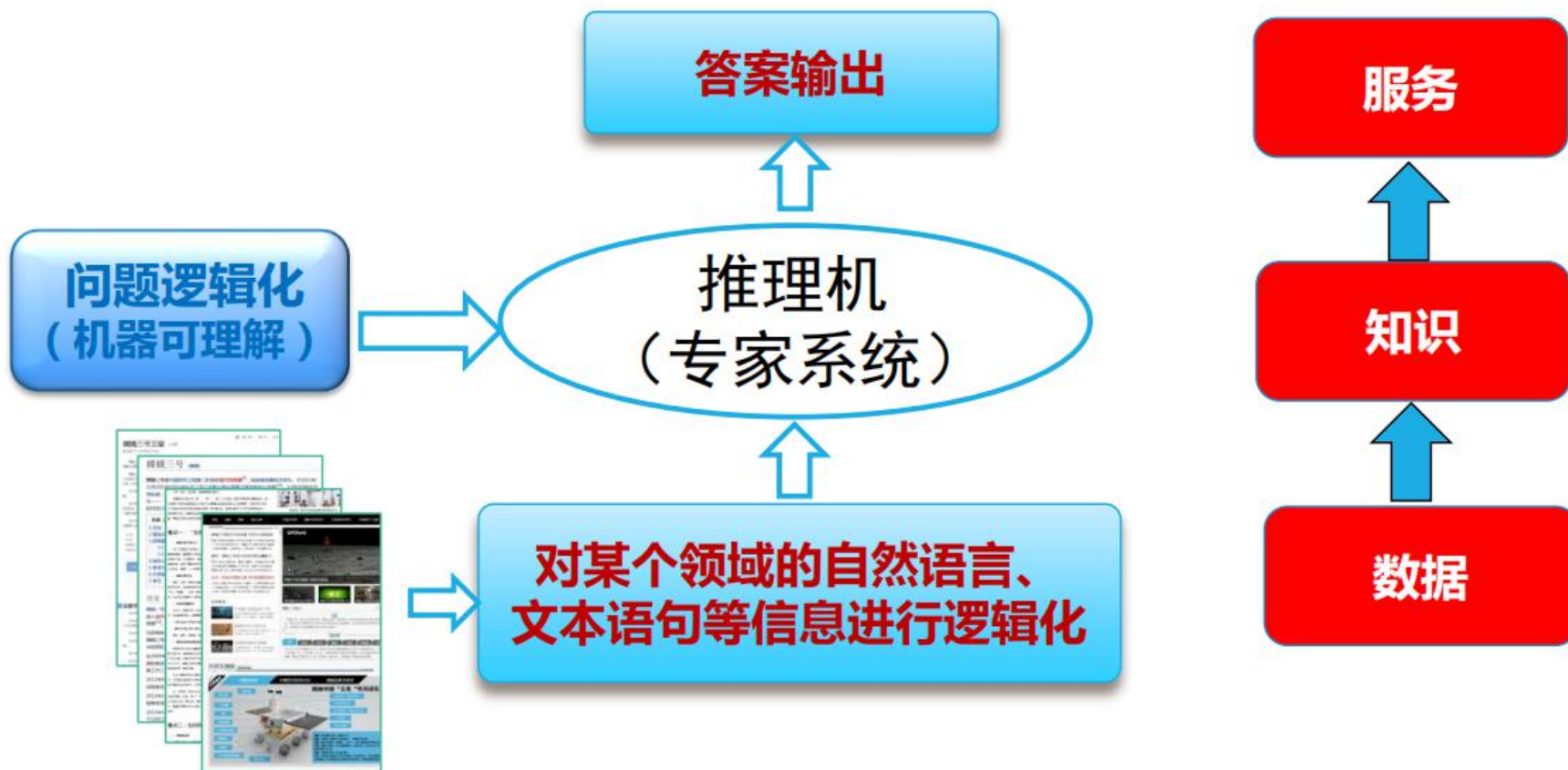
$$C_{6453} = \neg \textit{likes}(\textit{Paul}, x) \vee \textit{answer}(x)$$

$$C_{64531} = \textit{answer}(\textit{wine})$$

3.6 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 用谓词公式 F 表示;
- (2) 把待求解的问题 用谓词公式 Q 表示, 并否定 Q , 再与 answer 构成析取式 $(\neg Q \vee \text{answer})$;
- (3) 把谓词公式集 $\{F, \neg Q \vee \text{answer}\}$ 化为子句集 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。



确定性推理方法（第3章）

- 3.1 推理的基本概念
- 3.2 自然演绎推理（例如**Datalog**推理）
- 3.3 谓词公式化为子句集的方法
- 3.4 鲁宾逊归结原理
- 3.5 归结反演
- 3.6 应用归结反演求解问题

MOOC:
第四讲单元测试
及作业

归
结
演
绎
推
理

要求：了解推理的分类和自然演绎推理的特点，掌握谓词公式化为子句集的方法，熟练掌握归结原理、方法，并能灵活应用，掌握知识图谱中的**RDF**模型及归纳推理

重点：归结原理及应用

难点：知识图谱中的归纳推理

□ 补充：知识图谱中的
知识表示、知识推理