第4章：确定性推理

Certainty Reasoning

李磊

第四章 确定性推理

- **推理和判断能力是人类智能行为的重要特征之一，所以研究人工智能就必须对推理加以深入研究，研究推理的机理、推理的方法和怎样实现机器推理。**

第四章 确定性推理

- **本章主要内容**
- **4.1 推理的基本概念**
- **4.2 置换与合一**
- **4.3 推理的逻辑基础**
- **4.4 自然演绎推理**
- **4.5 归结推理**
- **4.6 产生式推理**

4.1 推理的基本概念

- **本节主要内容**
  - **4.1.1 什么是推理**
  - **4.1.2 推理方法及分类**
  - **4.1.3 推理的控制策略**
  - **4.1.4 推理的方向控制**
  - **4.1.5 冲突消解策略**

4.1.1 什么是推理

- **1.什么是推理**
  - **所谓推理 (Reasoning) 就是由已知的判断推导出新的判断的思维过程。**
  - **或，按照某种策略从已知事实出发去推出结论的过程。**
    - **如我们证明或求解一个数学题，首先使用问题给出的已知条件（证据），运用相关的知识（已知定律、定理等），对问题进行运算和推导，最后得出结论，这个过程就是一个推理的过程。**

4.1.1 什么是推理

- **2. 推理中的证据**
  - **事实 -- 初始证据，推理前由用户提供（已知条件）**
  - **中间结论 -- 推理过程中得出的中间结果**
- **3. 推理机**
  - **智能系统中负责推理控制的部分，目前大多由计算机程序实现。**
- **4. 推理的基本问题**
  - **(1) 推理的方法 -- 解决前提和结论的逻辑关系，确定性传递**
  - **(2) 推理的控制策略 -- 解决推理方向，冲突消解策略**

4.1.2 推理方法及分类

- 存在多种分类方法。

1. 按推理的逻辑基础分类

- 可分为：演绎推理、归纳推理、缺省推理
- (1) 演绎推理 (Deduction Reasoning)
  - 演绎推理是从一般知识到具体判断的推理过程。即从问题领域的一般知识和具体问题的已知事实、判断出发，推导出这个具体问题的一个新的判断。
  - 演绎推理是一般到具体（抽象到具体）的推理过程。

4.1.2 推理方法及分类

- **演绎推理有多种推理形式，其中最著名的是三段论推理，三段论推理组成如下：**
  - **①大前提 – 问题领域的一般性知识或判断**
  - **②小前提 – 具体问题的已知事实或判断**
  - **③结　论 – 满足大前提和小前提情况下，得出的关于具体问题的新判断**
- **举一个三段论推理的例子：1）人都是要死的；2）苏格拉底是人；3）苏格拉底是要死的。其中，“人都是要死的”是推理的大前提，“苏格拉底是人”是小前提，“苏格拉底是要死的”是结论。**

4.1.2 推理方法及分类

- **由上面的例子可见，演绎推理中，结论蕴含在前提的一般性知识之中，是一般到具体的过程，只要大前提、小前提为真，结论就一定为真。**
- **演绎推理早在人工智能诞生以前就被人类认知，并广泛使用，是经典逻辑中最重要的推理方法。在人工智能中，演绎推理是许多智能系统的基本推理形式之一，特别是在问题求解、定理证明等领域。**

4.1.2 推理方法及分类

- **(2) 归纳推理 （Induction Reasoning）**
  - **归纳推理是从个别到一般的过程，是从足够多的具体事例中归纳出一般性知识的推理过程。**
  - **归纳推理是人类思维活动中最基本、最常用的一种推理形式，人类的所有知识，特别是理论知识都是不断进行归纳的结果。人工智能中，特别是机器学习领域，归纳推理是重要的一种推理形式。**
  - **归纳推理是从具体到一般（具体到抽象）的过程。**

4.1.2 推理方法及分类

- 从选择归纳的事例的广泛性划分，归纳推理分为：
- ① 完全归纳推理
  - 完全归纳推理要求归纳时包括论域内的每一个事例，由所有事例的共性，推导出一个新的判断。
- ② 不完全归纳推理
  - 不完全归纳推理时，用抽样方法选择部分事例，由这部分事例的共性，推导出新的判断，并认为这个新判断适用论域内的所有事例。
  - 如：“天下乌鸦一般黑”

4.1.2 推理方法及分类

- **完全归纳推理的结论，由于考察了论域内的所有个体，必然是正确的，是必然性推理；**
- **非完全归纳推理的结论，由于是抽样考察论域内部分个体得出的，其正确与否有一定的偶然性，需要进一步的验证和归纳，是非必然性推理。**
- **由此可见完全归纳推理的条件要求非常严格，人类的归纳推理通常是非完全的、非必然的，人工智能中也是这样。**

4.1.2 推理方法及分类

- 按照推理所使用的方法可分为**枚举、类比、统计和差异归纳推理**等
- ① 枚举归纳推理
  - 是指在进行归纳时，如果已知某类事物的有限可数个具体事物都具有某种属性，则可推出该类事物都具有此种属性。
- 例如，设有如下事例：
  - 王强是计算机系学生，他会编程序；
  - 高华是计算机系学生，她会编程序；
  - …… ……
- 当这些具体事例足够多时，就可归纳出一个一般性的知识：
  - 凡是计算机系的学生，就一定会编程序。

4.1.2 推理方法及分类

- ② **类比归纳推理**
  - **是指在两个或两类事物有许多属性都相同或相似的基础上，推出它们在其他属性上也相同或相似的一种归纳推理。**
  - **设A、B分别是两类事物的集合：**
  - **A={a1,a2,……}**
  - **B={b1,b2,……}**
  - **并设ai与bi总是成对出现，且当ai有属性P时，bi就有属性Q与此对应，即**
  - **P(ai)→Q(bi)　i=1,2,…..**
  - **则当A与B中有一新的元素对出现时，若已知a'有属性P，b'有属性Q，即**
  - **P(a')→Q(b')**
- **类比归纳推理的基础是相似原理，其可靠程度取决于两个或两类事物的相似程度以及这两个或两类事物的相同属性与推出的那个属性之间的相关程度。**
  - **推荐系统**

4.1.2 推理方法及分类

- **演绎推理与归纳推理的区别**
  - **演绎推理是在已知领域内的一般性知识的前提下，通过演绎求解一个具体问题或者证明一个结论的正确性。它所得出的结论实际上早已蕴含在一般性知识的前提中，演绎推理只不过是将已有事实揭露出来，因此它不能增殖新知识。**
  - **归纳推理所推出的结论是没有包含在前提内容中的。这种由个别事物或现象推出一般性知识的过程，是增殖新知识的过程。**
  - **例如，一位计算机维修员，通过大量实例积累经验，是一种归纳推理方式。运用这些一般性知识去维修计算机的过程则是演绎推理。**

4.1.2 推理方法及分类

- **(3) 缺省推理 (Default Reasoning)**
  - **缺省推理，也叫做默认推理，指推理时缺少部分前提条件、或部分前提条件没有证据证明为真，在我们假设这部分前提条件为真的情况下，推导出结论的过程。**
  - **这些缺少的、或没有证据证明为真的部分前提条件，通常是当前推理相关领域的一些常识性知识、事实，并且根据经验其存在、且为真的可能性极大，因此我们有理由默认这部分前提条件存在、且为真。**

4.1.2 推理方法及分类

- **缺省推理是在不完全前提条件下进行的，得出的结论判断往往是不准确的，甚至是完全错误的，需要随着情况的发展不断进行修正，因而是一种非单调推理方式。**
- **缺省推理是人类日常生活中经常采用的推理形式之一。**
  - **例如，我们去拜访一位陌生的客人，路过鲜花店时想起是否要送客人一束鲜花，我们不知道这位客人是否真的喜欢鲜花，是否有花粉过敏症，但我们通常会根据大部分人都喜欢鲜花的常识，带上一束鲜花去会见客人。**
  - **类似的例子很多很多，告诉你一只鸟，你立刻会想到“会飞翔”，……，这些结论就是默认推理得出的。缺省推理又可细分为常识推理和约束推理。**

4.1.2 推理方法及分类

- **2. 按推理的确定性分类**
  - **(1) 确定性推理**
    - **推理的证据、知识、结论都是确定的。**
    - **本章主要介绍确定性推理。**
  - **(2) 不确定推理**
    - **推理的证据、知识、结论都是不确定的。**

4.1.2 推理方法及分类

- **3. 按推理的单调性分类**
  - **(1) 单调推理 (Monotonic Reasoning)**
    - **单调推理，指系统中已知为真的判断、命题或知识随时间严格增加，呈现单调性，即推导产生的新的为真的判断、命题或知识加入到系统知识库中，不会和系统原有的知识不相容、或矛盾。基于经典命题逻辑和一阶谓词逻辑的推理是单调推理。**
  - **(2) 非单调推理 (Nonmonotonic Reasoning)**
    - **非单调推理，指推理产生的结论有可能与系统知识库中原有的知识不相容、或矛盾，甚至完全否定原有的某些判断。出现这种情况时，必须使用某种正确性维持机制，解除前后推理的不相容和矛盾，系统知识库中知识的增长呈现非单调性。**
  - **在不完全知识的前提下的推理都是非单调推理，**
  - **前面讨论的归纳推理、缺省推理、不确定性推理等都是非单调推理。**

4.1.2 推理方法及分类

- **4. 按推理中是否使用启发信息**
  - **(1) 启发式推理**
  - **(2) 非启发式推理**
- **5. 其它分类方法**
  - **基于知识的推理、统计推理、直觉推理 等。**

4.1.3 推理的控制策略

- **1. 推理的控制策略**
  - **推理过程不仅依赖于所用的推理方法，同时也依赖于推理的控制策略。**
  - **推理的控制策略是指如何使用领域知识使推理过程尽快达到目标的策略。**
- **2. 推理控制策略的分类**
  - **推理策略**
  - **搜索策略**
    - **主要解决推理线路、推理效果、推理效率等问题。即第3章讨论的各种搜索方法和策略。**

4.1.3 推理的控制策略

- **2. 推理控制策略的分类**
  - **推理策略**
    - **主要解决求解策略、限制策略、推理方向、冲突消解等问题。**
    - **求解策略是指仅求一个解，还是求所有解或最优解等。**
    - **限制策略是指对推理的深度、宽度、时间、空间等进行的限制。**
    - **推理方向控制策略用于确定推理的控制方向，可分为正向推理、逆向推理、混合推理及双向推理。**
    - **冲突消解策略是指当推理过程有多条知识可用时，如何从这多条可用知识中选出一条最佳知识用于推理的策略。**

4.1.4 推理的方向控制

- **1.正向推理（Forward Reasoning)**
  - **正向推理是从事实出发往目标方向进行的推理，又称为数据驱动推理、前向链推理、模式制导推理等。**
  - **正向推理控制的基本思想：**
    - **从用户给出的问题的事实开始，扫描系统知识库，得到适用于当前事实的所有知识的集合；运用某种冲突消解策略，从得到的知识集合中选择一条知识；将这条知识与当前事实结合进行推理，推导出新的事实或判断；将推导的结论加入原有的事实集合；重复上述过程，直到到达目标、或知识库中再没有知识可用。**

4.1.4 推理的方向控制

- **正向推理的主要优点**
  - **比较直观，允许用户主动提供有用的事实信息，适合于诊断、设计、预测、监控等领域的问题求解。**
- **正向推理的主要缺点**
  - **推理无明确目标，求解问题时可能会执行许多与解无关的操作，导致推理效率较低。**

4.1.4 推理的方向控制

- **2. 逆向推理 (Backward Reasoning)**
  - **逆向推理是从假设目标开始往事实方向进行的推理，又称为目标驱动推理、反向链推理、目标制导推理等。**
  - **逆向推理的基本思想：**
    - **用户假设一个目标，找寻这个假设目标成立的证据，首先寻找假设目标的事实证据、或用户交互提供的证据；如果找不到事实证据，接着扫描系统知识库，找寻适用于这个假设的所有知识，使用某种冲突消解策略选择一条知识应用于假设目标，产生新的假设，即子目标；对每个子目标重复前面的过程，直到由假设目标产生的所有子目标都得到事实证据、或用户交互提供的证据。**

4.1.4 推理的方向控制

- **逆向推理的主要优点**
  - **不必寻找和使用那些与假设目标无关的信息和知识**
  - **推理过程的目标明确**
  - **也有利于向用户提供解释，在诊断性专家系统中较为有效。**
- **逆向推理的主要缺点**
  - **用户对解的情况认识不清时，由系统自主选择假设目标的盲目性比较大，若选择不好，可能需要多次提出假设，会影响系统效率。**

4.1.4 推理的方向控制

- **3. 混合推理的概念**
  - **把正向推理和逆向推理结合起来所进行的推理称为混合推理。是一种解决较复杂问题的方法。**
  - **混合推理的方法:**
  - **(1) 先正向后逆向**
    - **这种方法先进行正向推理，从已知事实出发推出部分结果，然后再用逆向推理对这些结果进行证实或提高它们的可信度。**
  - **(2) 先逆向后正向**
    - **这种方法先进行逆向推理，从假设目标出发推出一些中间假设，然后再用正向推理对这些中间假设进行证实。**
  - **(3) 双向混合**
    - **是指正向推理和逆向推理同时进行，使推理过程在中间的某一步结合起来。**

4.1.5 冲突消解策略

- **1. 推理冲突**
  - **在正向推理中出现一条事实能与系统知识库中的多条知识匹配，或者多个事实能与同一条知识匹配；在逆向推理中出现同一个假设目标能与系统知识库中多条知识匹配，或多个假设目标能与同一条知识匹配，叫做推理冲突。**
- **2. 冲突消解**
  - **冲突消解策略就是推理出现冲突时，如何在多条冲突的知识中有效地选择知识进行推理的控制策略。**

4.1.5 冲突消解策略

- **3. 冲突消解策略的重要性**
  - 冲突消解最简单的方法是顺次选择、或随机选择知识进行推理，这样的消解策略效率低下甚至不能容忍或组合爆炸。
  - 一方面随着问题复杂性增加，冲突加剧，为了尝试每一种可能的推理路径，浪费大量的时空资源，甚至出现组合爆炸。
  - 另一方面有些实际的系统，比如疾病诊断、实时控制系统，实时性要求非常高，不允许系统花费大量时间进行尝试。
  - 由此可见冲突消解的策略选择对于提高系统的推理效率至关重要。
  - 同时在实际的推理系统中实施冲突消解策略是复杂而又困难的工作。系统采用不同的知识表示方法，要求相应的消解策略；不同的应用领域，消解策略也可能不同；推理是要求得到一个解、还是最优解、还是全部解，相应的消解策略也会不同。

4.1.5 冲突消解策略

- **4. 冲突消解策略的总体原则**
  - **(1) 按针对性排序**
    - **假设系统知识库中有以下两条知识：**
    - **R_1: if ($a_1 \wedge a_2 \wedge \ldots \wedge a_n$) then h_1**
    - **R_2: if ($a_1 \wedge a_2 \wedge \ldots \wedge a_n \wedge b_1 \wedge b_2 \wedge \ldots \wedge b_m$) then h_2**
    - **R_2包含了R_1的全部条件，我们说R_2比R_1具有更大的针对性， R_1比R_2具有更大的通用性。按针对性排序，即优先选择针对性大的知识。如本例选R_2**

4.1.5 冲突消解策略

- **(2) 根据问题领域的特点排序**
  - **根据问题领域的特点，在组织系统知识库时预先给每条知识赋予优先级，出现推理冲突时，按知识的优先级选择知识。缺点是每往知识库中增加一条知识，都需要确定其优先级别参数。**
- **(3) 按匹配程度排序**
  - **在不确定匹配中，只要达到一定的匹配程度就可以认为两个知识模式是可匹配的。在这种情况下出现推理冲突，优先选择匹配度较大的知识进行推理。**
- **(4) 按上下文限制排序**
  - **系统知识库根据问题的状态、属性等，往往采用分块、或分组的方式进行组织。推理中根据推理的进程，即上下文线索，往往只能在某一块或某一组内选择知识。**

4.1.5 冲突消解策略

- **(5) 按数据冗余限制排序**
  - **当应用一条知识产生上下文冗余项时，这条知识的优先级降低。产生的冗余项越多，其优先级越低。**
- **(6) 并行处理**
  - **并行处理的思想是当出现推理冲突时，对全部的知识，或按某种策略选择的部分知识，并行进行推理。实现并行处理一方面要求计算机有强大的处理能力，另一方面要有一套好的并行处理算法。**

4.2 置换与合一

- **本节主要内容**
  - **4.2.1 置换**
  - **4.2.2 合一 (Unification)**
  - **4.2.3 求mgu算法**

4.2.1 置换

- **1. 置换 (Substitution)**
  - **又称为替换，即在一个谓词公式中用项去替换变量。**
  - **置换是形如 $\{ t_1/x_1, t_2/x_2, \ldots, t_n/x_n \}$ 的有限集合。**
  - **其中：**
    - **$t_1, t_2, \ldots, t_n$是项；**
    - **$x_1, x_2, \ldots, x_n$是互不相同的变元；**
    - **t_i/x_i表示用t_i替换x_i。**
    - **并且要求t_i与x_i不能相同，x_i不能循环地出现在另一个t_i中。**

4.2.1 置换

- **例：**
  - **{a/x, c/y, f(b)/z} 是一个置换**
  - **{g(a)/x, f(x)/y} 是一个置换**
  - **{g(y)/x, f(x)/y} 不是一个置换**
    - **因为置换的目的是用其它的变量、常量、函数替换谓词公式中的某个变量，使其不再出现在公式中，{ f(y)/x, g(x)/y } 中出现了 x，y 的循环替换，结果既不能消去x也不能消去 y，所以不是一个置换。不符合定义的要求。**
- **置换常用罗马字符表示：如 θ、σ、 α、 λ等**

4.2.1 置换

- **2. 谓词公式的置换**
  - **设θ={$t_1/x_1,t_2/x_2,\ldots,t_n/x_n$}是一个置换，F是一个谓词公式，把公式F中出现的所有 x_i 换成 t_i (i=1,2,…,n)，得到一个新的公式G，称G为F在置换θ下的例示，记作G=Fθ。**
  - **一个谓词公式的任何例示都是该公式的逻辑结论。**

4.2.1 置换

- **3. 置换的合成**
  - **设**

    $\theta=\{ t_1/x_1, t_2/x_2, \ldots, t_n/x_n \}$

    $\lambda=\{ u_1/y_1, u_2/y_2, \ldots , u_m/y_m \}$
  - **是两个置换。则θ与λ的合成也是一个置换，记作θλ。**
  - **对谓词公式F来说，即先做Fθ，再做置换(Fθ)λ**
  - **但是，通常 $\theta\lambda \neq \lambda\theta$**

4.2.1 置换

- **例：对公式 P(x, f(y), z)，有2个置换**
  - **θ = { g(w)/x, c/z } , λ = { a/x, b/y }，合成置换：**
  - **α=θλ= { g(w)/x, c/z, b/y }**
  - **β=λθ= { a/x, b/y, c/z }**
  - **Pθ=P(g(w), f(y), c);**
  - **(Pθ)λ=P(g(w), f(b), c)**
  - **Pα =P(g(w), f(b), c)**
  - **可见：(Pθ)λ= Pα**
  - **Pβ =P(a, f(b), c)，可见Pα ≠ Pβ**

4.2.2 合一（Unification）

- **简单地说，合一是寻找置换，使得2个谓词或谓词公式一致。**
- **1. 合一和可合一公式：**
  - **设有公式集$F=\{F_1, F_2,\ldots,F_n\}$，若存在一个置换$\theta$，可使：　$F_1\theta=F_2\theta=\ldots=F_n\theta$，**
  - **则称θ是F的一个合一置换。**
  - **称$F_1,F_2,\ldots,F_n$是可合一的公式。**

4.2.2 合一 (Unification)

- **例：公式集：F={ P(x, f(y), B), P(x, f(B), B) }**
  - **置换 θ={ A/x, B/y } 是一个合一置换。因为：**
    - **P(x, f(y), B) θ = P(A, f(B), B)**
    - **P(x, f(B), B) θ = P(A, f(B), B)**
  - **置换 λ={ g(w)/x, B/y } 是另一个合一置换。**
- **可合一公式集合的合一置换往往不止一个。**

4.2.2 合一 (Unification)

- **2. 最一般（通用）合一者**
  - **mgu – most general unification**
  - **如果 $F=\{F_1, F_2,\ldots,F_n\}$是一个可合一公式集合，设σ是公式集F的一个合一置换，如果对F的任一个合一者θ都存在一个置换λ，使得$\theta=\sigma\lambda$，则称σ是 F 的最一般合一者。**
- **一个公式集的最一般合一者是唯一的。**
- **例：公式集 F={ P(x, f(y), B), P(x, f(B), B) } 的 mgu 为 σ= {B/y}**

4.2.3 合一算法

- **1. 差异集合（Disagreement Set）**
  - **分歧集合，**
  - **设有一非空有限公式集合 $F=\{F_1, F_2,\ldots,F_n\}$， 从F中各个公式的第一个符号同时向右比较， 直到发现第一个不尽相同的符号为止， 从F的各个公式中取出那些以第一个不一致符号开始的最大的子表达式为元素，组成一个集合D， 称为F的差异集合。**
  - **实际操作时，总是2个公式进行扫描，所以差异集中只有2个元素，这样便于置换、合一。**
  - **例：F = { P(x, y, z), P(x, f(a), h(b)) }，从两个表达式的第一个符号从左往右扫描，得到第一个差异集合 D1 = { y,f(a) }，接着扫描得到第二个差异集合 D2 = { z,h(b) }。**

4.2.3 合一算法

- **2. 合一算法**

  即寻找mgu，设F为非空有限公式集合，求mgu:

  - **①置 i=0， F_i=F， σ_i= ε， 其中ε为空集合，是不含任何元素的空置换；**
  - **② 若 F_i 只含有一个表达式， 算法停止， σ_i 就是mgu；**
  - **③ 否则，找出 F_i 的差异集合 D_i ；**
  - **④ 若 D_i 中存在元素 t_i 和 x_i ，其中： t_i 是项， x_i 是变元，且 x_i 不在 t_i 中出现，则置：**
    - $\sigma_{i+1}= \sigma_i \{t_i / x_i\}$
    - $F_{i+1}=F_i \{t_i / x_i\}$
    - i=i+1
    - GOTO ②
  - **⑤ 算法停止，F的mgu不存在。**

4.2.3 合一算法

- **例4.2.1：F ={ P(a, x, f(g(y))), P(z, h(z,u), f(u)) }，求mgu 。**
- **解:**
- **①i=0, F_0=F, σ_0= ε**
  - **F_0未合一，求D_0={a, z}，则:**
    - **σ_1= σ_0{a/z} ={a/z}**
    - **F_1 =F_0{a/z}={ P(a, x, f(g(y))), P(a, h(a,u), f(u)) }**
    - **i=1;**

4.2.3 合一算法

- ②i=1
  - **F_1未合一，求D_1={x, h(a,u)}，则:**
    - **$\sigma_2 = \sigma_1${h(a,u)/x} ={a/z, h(a,u)/x}**
    - **$F_2 = F_1${h(a,u)/x}={ P(a, h(a,u), f(g(y))), P(a, h(a,u), f(u)) }**
    - **i=2;**

4.2.3 合一算法

- ③i=2
  - **F_2未合一，求D_2={g(y), u}，则:**
    - **$\sigma_3 = \sigma_2\{g(y)/u\}$ = ={a/z, h(a,g(y))/x, g(y)/u}**
    - **$F_3 = F_2\{g(u)/u\}$={ P(a, h(a,u), f(g(y))), P(a, h(a,u), f(g(y))) }**
    - **i=3;**
- ④i=3
  - **F_3 已经合一， σ_3 即为mgu。即:**
  - **mgu={ a/z, h(a,g(y))/x, g(y)/u }**

4.2.3 合一算法

- **注意**
  - 用 t_i 去置换 x_i 时，要把一个公式中所有的 x_i 都替换成 t_i 。
  - 置换的合成
- **其它例子:**
  - { P(x),P(A) }, mgu={ A/x }
  - { P(f(x),y,g(y)), P(f(x),b,g(b)) }, mgu={ b/y }
  - { P(f(x,g(A,y)), g(A,y)), P(f(x,z),z) }, mgu={ g(A,y)/z }

4.3 推理的逻辑基础

- **本节基本内容**
  - **4.3.1 谓词公式的解释**
  - **4.3.2 谓词公式的永真性和可满足性**
  - **4.3.3 谓词公式的等价和永真蕴含**
  - **4.3.4 谓词公式的范式**

4.3.1 谓词公式的解释

- **1. 命题公式的解释**
  - 在命题逻辑中，**命题公式的一个解释就是对该命题公式中各个命题变元的一次真值指派。**
  - 有了命题公式的解释，就可据此求出该命题公式的真值。
- **2. 谓词公式的解释**
  - 由于谓词公式中可能包含由**个体常量、变元或函数**，因此，必须先考虑这些个体常量、变量和函数在个体域（论域）上的取值，然后才能根据它们的具体取值为谓词分别指派真值。
  - **定义：** 设D是谓词公式P的非空个体域（论域），若对P中的个体常量、变量、函数和谓词按如下规定赋值：

    (1) 为每个个体常量和变量指派D中的一个元素；

    (2) 为每个n元函数指派一个从D^n到D的一个映射，其中

    $$D^n = \{(x_1, x_2, \ldots, x_n) \mid x_1, x_2, \ldots, x_n \in D\}$$

    (3) 为每个n元谓词指派一个从D^n到{F，T}的映射。

  则称这些指派为**P在D上的一个解释I**

4.3.1 谓词公式的解释

- **例4.3.2 设个体域D={1, 2}，求公式A=(∀x)(∀ y)P(x, y)在D上的解释，并指出在每一种解释下公式A的真值。**
- **解：由于公式A中没有包含个体常量和函数，故可直接根据个体变量为谓词指派真值，设有**

P(1,1)	P(1,2)	P(2,1)	P(2,2)
T	F	T	F

  - **这就是公式A 在D 上的一个解释。从这个解释可以看出：**
    当x=1、y=1时，有P(x,y)的真值为T;
    当x=2、y=1时，有P(x,y)的真值为T;
  - **即对x在D上的任意取值，都存在y=1使P(x,y)的真值为T。因此，在此解释下公式A的真值为T。**

4.3.1 谓词公式的解释

- **说明：一个谓词公式在其个体域上的解释不是唯一的。例如，对公式A，若给出另一组真值指派**

P(1,1)	P(1,2)	P(2,1)	P(2,2)
T	T	F	F

  - **这也是公式A 在D 上的一个解释。从这个解释可以看出：**
    当x=1、y=1时，有P(x,y)的真值为T;
    当x=2、y=1时，有P(x,y)的真值为F;
  - **即对x在D上的任意取值，不存在一个y使得P(x,y)的真值为T。因此，在此解释下公式A的真值为F。**
- **实际上本例中，A在D上共有16种解释，这里不在一一列举。**

4.3.1 谓词公式的解释

- **例3.3 设个体域D={1, 2}，求公式B=(∀x)P(f(x), a)在D上的解释，并指出在该解释下公式B的真值。**
- **解：设对个体常量 a 和函数f(x)的值指派如左图；对谓词的真值指派如右图：**

a	f(1)	f(2)
1	1	2

P(1,1)	P(1,2)	P(2,1)	P(2,2)
T	x	T	x

  - **由于已指派a=1，因此P(1,2)和P(2,2)不可能出现，故没给它们指派真值。**
  - **上述指派是公式B在D上的一个解释。在此解释下有**

    当x=1时，a=1使P(1,1)=T

    当x=2时，a=1使P(2,1)=T
  - **即对x在D上的任意取值，都存在a=1使P(f(x), a)的真值为T。因此，在此解释下公式B的真值为T。**
- **由上面的例子可以看出，谓词公式的真值都是针对某一个解释而言的，它可能在某一个解释下真值为T，而在另一个解释下为F。**

4.3.2 谓词公式的永真性和可满足性

- **1. 谓词公式的永真性**
  - **如果谓词公式 P对非空个体域D上的任一解释都取得真值T，则称P在D上是永真的；**
  - **如果P在任何非空个体域上均是永真的，则称P永真。**
  - **可见，要判定一谓词公式为永真，必须对每个非空个体域上的每个解释逐一进行判断。当解释的个数为有限时，尽管工作量大，公式的永真性毕竟还可以判定，但当解释个数为无限时，其永真性就很难判定了。**

4.3.2 谓词公式的永真性和可满足性

- **2. 谓词公式的可满足性**
  - **对于谓词公式P，如果至少存在D上的一个解释，使公式P在此解释下的真值为T，则称公式P在D上是可满足的。**
  - **谓词公式的可满足性也称为相容性。**

4.3.2 谓词公式的永真性和可满足性

- **3. 谓词公式的不可满足性（永假性）**
  - **如果谓词公式P对非空个体域D上的任一解释都取真值F，则称P在D上是永假的；**
  - **如果P在任何非空个体域上均是永假的，则称P永假。**
  - **谓词公式的永假性又称不可满足性或不相容。**
- **后面我们要给出的归结推理，就是采用一种逻辑上的反证法，将永真性转换为不可满足性的证明。**

4.3.3 谓词公式的等价和永真蕴含

- **谓词公式的等价性和永真蕴含性可分别用相应的等价式和永真蕴含式来表示，这些等价式和永真蕴含式都是演绎推理的主要依据，因此也称它们为推理规则。**
- **1. 等价式**
  - **设P与Q是D上的两个谓词公式，若对D上的任意解释，P与Q都有相同的真值，则称P与Q在D 上是等价的。**
  - **如果D是任意非空个体域，则称P与Q是等价的，记作P⇔Q。**
  - **常用等价式在第二章已经给出。**

4.3.3 谓词公式的等价和永真蕴含

- **2. 永真蕴含式**
  - **对谓词公式P和Q，如果P→Q永真，则称P 永真蕴含Q，且称Q为P的逻辑结论，P为Q的前提，记作 P ⇒ Q。**

4.3.3 谓词公式的等价和永真蕴含

- **常用的永真蕴含式如下：**
  - (1) 化简式　$P \wedge Q \Rightarrow P$，　$P \wedge Q \Rightarrow Q$
  - (2) 附加式　$P \Rightarrow P \vee Q$，　$Q \Rightarrow P \vee Q$
  - (3) 析取三段论　$\neg P \wedge (P \vee Q) \Rightarrow Q$
  - (4) 假言推理　$P \wedge (P \rightarrow Q) \Rightarrow Q$
  - (5) 拒取式　$\neg Q \wedge (P \rightarrow Q) \Rightarrow \neg P$
  - (6) 假言三段论　$(P \rightarrow Q) \wedge (Q \rightarrow R) \Rightarrow P \rightarrow R$
  - (7) 二难推理　$(P \vee Q) \wedge (P \rightarrow R) \wedge (Q \rightarrow R) \Rightarrow R$
  - (8) 全称固化　$(\forall x)P(x) \Rightarrow P(y)$

    其中，y是个体域中的任一个体，依此可消去谓词公式中的全称量词。
  - (9) 存在固化　$(\exists x)P(x) \Rightarrow P(y)$

    其中，y是个体域中某一个可以使P(y)为真的个体，依此可消去谓词公式中的存在量词。

4.3.4 谓词公式的范式

范式是谓词公式的标准形式。在谓词逻辑中，范式分为两种：

- **1. 前束范式**
  - **设F为一谓词公式，如果其中的所有量词均非否定地出现在公式的最前面，且它们的辖域为整个公式，则称F为前束范式。一般形式：**

    $$(Q_1x_1)\ldots\ldots(Q_nx_n)M(x_1,x_2,\ldots\ldots,x_n)$$

    其中，$Q_i(i=1,2,\ldots\ldots,n)$为前缀，它是一个由全称量词或存在量词组成的量词串； $M(x_1,x_2,\ldots\ldots,x_n)$为母式，它是一个不含任何量词的谓词公式。
  - **例如，$(\forall x)\ (\forall y)\ (\exists z)(P(x)\wedge Q(y,z)\vee R(x,z))$ 是前束范式。**
  - **任一谓词公式均可化为与其对应的前束范式，其化简方法将在后面子句集的化简中讨论。**

4.3.4 谓词公式的范式

- **2. Skolem范式**
  - **如果前束范式中所有的存在量词都在全称量词之前，则称这种形式的谓词公式为Skolem范式。**
  - **例如，$(\exists x)(\exists z)(\forall y)(P(x) \vee Q(y,z) \wedge R(x,z))$ 是Skolem范式。**
  - **任一谓词公式均可化为与其对应的Skolem范式，其化简方法也将在后面子句集的化简中讨论。**

第四章 确定性推理

- **本章主要内容**
- **4.1 推理的基本概念**
- **4.2 置换与合一**
- **4.3 推理的逻辑基础**
- **4.4 自然演绎推理**
- **4.5 归结推理**
- **4.6 产生式推理**

4.4 自然演绎推理

- **自然演绎推理**
  - **从一组已知为真的事实出发，直接运用经典逻辑中的推理规则推出结论的过程称为自然演绎推理。**
- **自然演绎推理最基本的推理规则是三段论推理，它包括：**
  - **假言推理　　$P, P \rightarrow Q \Rightarrow Q$**
  - **拒取式　　$\neg Q, P \rightarrow Q \Rightarrow \neg P$**
  - **链式规则　　$P \rightarrow Q, Q \rightarrow R \Rightarrow P \rightarrow R$**

4.4 自然演绎推理

- **例4.4.1 设已知如下事实和知识：**

  A，B，A→C，B∧C→D，D→Q

  求证：Q为真。

- **证明：因为**
  - **A, A→C⇒ C　　假言推理**
  - **B，C⇒ B∧C　　引入合取词**
  - **B∧C，B∧C→D ⇒ D　假言推理**
  - **D，D→Q ⇒ Q　　假言推理**
  - **因此，Q为真**

4.4 自然演绎推理

- **例4.4.2 设已知如下事实和知识：**
- **(1) 只要是需要编程序的课，李都喜欢。**
- **(2) 所有的程序设计语言课都是需要编程序的课。**
- **(3) Python是一门程序设计语言课。**
- **求证：李喜欢Python这门课。**
- **证明：首先定义谓词**
  - **Prog(x)　　x是需要编程序的课。**
  - **Like(x, y)　　x喜欢y。**
  - **Lang(x)　　x是一门程序设计语言课**

4.4 自然演绎推理

- **把已知事实及待求解问题用谓词公式表示如下：**

  (∀x)(Prog(x)→Like(Li , x))

  (∀x)(Lang(x)→Prog(x))

  Lang(Python)

- **应用推理规则进行推理：**

  Prog(y)→Like(Li , y)　　全称固化

  Lang(y)→Prog(y)　　全称固化

  Lang(Python)，Lang(y)→Prog(y) ⇒ Prog(Python)　假言推理　　{Python/y}

  Prog(Python)，Prog(x)→Like(Li , x) ⇒ Like(Li , Python)　假言推理　{Python/x}

- **因此，李喜欢Python这门课。**

4.4 自然演绎推理

- **自然演绎推理的优点：**
  - **定理证明过程自然，易于理解，并且有丰富的推理规则可用。**
- **自然演绎推理的缺点：**
  - **容易产生知识爆炸，推理过程中得到的中间结论一般按指数规律递增，对于复杂问题的推理不利，甚至难以实现。**

4.5 归结推理

- **归结演绎推理是一种基于鲁宾逊（Robinson）归结原理的机器推理技术。鲁宾逊归结原理亦称为消解原理，是鲁宾逊于1965年在海伯伦（Herbrand）理论的基础上提出的一种基于逻辑的“反证法”。**
- **归结原理将永真性的证明转化为关于不可满足性的证明。**
- **鲁宾逊归结原理使定理证明的机械化成为现实。**

4.5.1 子句及子句集

- **1. 文字**
  - **原子谓词公式及其否定统称为文字。**
  - **例如：P(x)、Q(x)、¬ P(x)、 ¬ Q(x)等都是文字。**
- **2.子句 (Clause)**
  - **单个文字或任何文字的析取公式称为子句。**
  - **例如，**
    - **P(x)**
    - **P(x)∨Q(x)**
    - **P(x，f(x))∨Q(x，g(x)) 都是子句。**

4.5.1 子句及子句集

- **3. 空子句**
  - 不含任何文字的子句称为空子句。
  - 由于空子句不含有任何文字，也就不能被任何解释所满足，因此空子句是永假的，不可满足的。
  - 空子句一般被记为□或NIL。
- **4. 子句集**
  - 由子句构成的集合称为子句集。
  - 在谓词逻辑中，任何一个谓词公式都可以通过应用等价关系及推理规则化成相应的子句集。

4.5.1 子句及子句集

- **5. 谓词公式的子句集表示**
  - **任何一个谓词公式都可以化为合取范式，合取的每一个部分就是一个子句，所有子句组成这个谓词公式的子句集合，我们将这个子句集合叫做谓词公式的子句集表示。**
  - **下面我们结合实例介绍将谓词公式化为子句集表示的一般步骤。设有公式：**

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

4.5.1 子句及子句集

- **(1)消去蕴涵符号**
  - **使用等价公式 $P \rightarrow Q \Leftrightarrow \neg P \vee Q$ 消去谓词公式中所有的蕴含连接词。**
  - **例如公式：**

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

消去蕴含后等价变化为：

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

4.5.1 子句及子句集

- **(2) 缩小否定符号的作用范围 （辖域）**
  - **反复使用以下推理规则，将每个否定符号“¬”移到紧靠谓词的位置，即使得每个否定符号只作用于一个谓词上。**
  - **双重否定律**

    $\neg(\neg P) \Leftrightarrow P$
  - **德摩根定律**

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

    $\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$
  - **量词转换律**

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

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

4.5.1 子句及子句集

- **例如，**

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

- **上式经等价变换后为：**

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

4.5.1 子句及子句集

- **(3) 变量命名标准化**
  - **在一个量词的辖域内，把谓词公式中受该量词约束的变元全部用另外一个没有出现过的任意变元代替，使不同量词约束的变元有不同的名字。**
  - **例如，**

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

  - **上式经变换后为**

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

4.5.1 子句及子句集

- **(4) 消去存在量词(Skolem化)**
  - **引入Skolem函数，消去存在量词，分为两种情况。**
  - **①形如 (∀x) [(∃y)P(x,y)]，存在量词在全称量词的辖域内**
    - **存在量词y可能依赖于x，令这种依赖关系为y=f(x)，由f(x)把每个x值映射到存在的那个y，f(x)叫做Skolem函数。**
    - **引入Skolem函数f(x)后，上面的公式变为：**

      (∀x) [P(x, f(x)]

4.5.1 子句及子句集

- **② 形如 (∃y)(∀x)P(x,y)，存在量词前没有全称量词约束**
  - **这种情况比较容易理解，即在变量y的论域内，存在一些特定的取值，使得公式为真，那么我们就可以用y论域上使公式为真的一个特定取值--常量，来替代变量y，从而消去存在量词。所以这种情况的Skolem函数为一常量，即用变量y论域内的，使得公式为真的特定取值的一个常量符号替代原来的变量y，**
  - **这个常量符号不能与公式中的其它符号同名，也就是没有在公式中出现过。**
  - **例上面的公式，令y=A，Skolem化后，变为：**

    (∀x) [P(x, A)]

4.5.1 子句及子句集

- **前例：**

(∀x){(∃y)¬P(x, y)∨(∃z)[Q(x, z) ∧¬R(x, z)]}

- **Skolem化，令y=f(x)，z=g(x)**
- **公式变换为：**

(∀x){¬P(x, f(x))∨[Q(x,g(x))∧¬R(x,g(x))]}

4.5.1 子句及子句集

- **(5) 化为前束范式**
  - **将公式中的所有约束量词移到公式最前面，形成前束公式，形式如下：**

    前束式=（约束量词前缀）（母式）
  - **在移动时不能改变其相对顺序。**
  - **由于第(3)步已对变元进行了标准化，每个量词都有自己的变元，这就消除了任何由变元引起冲突的可能，即：移动不改变量词辖域。**
  - **例：前例已经是前束范式，**

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

4.5.1 子句及子句集

- **(6) 消去全称量词**
  - **由于母式中的全部变元均受全称量词的约束，并且全称量词的次序已无关紧要，因此可以省掉全称量词。**
  - **剩下的母式，默认所有变元受全称量词约束。**
  - **例如，上式消去全称量词后为**

    {¬P(x, f(x))∨[Q(x,g(x))∧¬R(x,g(x))]}

4.5.1 子句及子句集

- **(7) 化为合取范式**
  - **反复运用谓词公式的结合律和分配律，将公式化为合取范式。**
  - **分配率**

    P∧(Q∨R) <=> (P∧Q) ∨ (P∧R)

    P∨(Q∧R) <=> (P∨Q) ∧ (P∨R)
  - **结合率**

    (P∨Q)∨R <=> P∨(Q∨R)

    (P∧Q)∧R <=> P∧(Q∧R)

4.5.1 子句及子句集

- **例公式：**

{¬P(x, f(x))∨[Q(x,g(x))∧¬R(x,g(x))]}

- **只用到分配律，化为合取范式：**

[¬P(x, f(x))∨Q(x,g(x))] ∧ [¬P(x, f(x))∨¬R(x,g(x))]

4.5.1 子句及子句集

- **(8) 将公式用子句集合表示**
  - **取出合取范式中的每个子句；**
  - **对子句变量换名，使任意两个子句中不出现相同的变量名，便于推理中的置换、合一。**
  - **例如，对前面的公式，**

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

  - **可把第二个子句中的变元名x更换为y，得到如下子句集：**

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

4.5.1 子句及子句集

- **例4.5.1 将下面公式化为子句集表示。**

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

- **解：**

(1) 去蕴含

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

(2) 处理否定连接词

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

(3) 变量命名标准化 ---- 令后面y=w

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

4.5.1 子句及子句集

(4) 消去存在量词 -- 引入Skolem函数 w=g(x)

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

(5) 前束化

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

(6) 隐去全称量词

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

4.5.1 子句及子句集

(7) 化为合取范式

- 用集合律，去除括号：

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

- 用分配律，得合取范式：

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

(8) 变量更名、将公式用子句集合表示

$\{ [\neg P(x) \vee \neg P(y) \vee P(f(x,y))],$

$[\neg P(x1) \vee Q(x1, g(x1))],$

$[\neg P(x2) \vee \neg P(g(x2))] \}$

4.5.1 子句及子句集

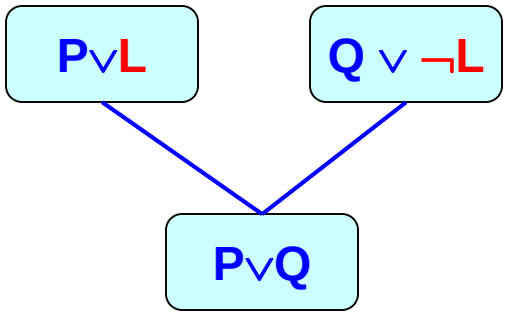
- **6. 子句集的不可满足性**
  - **子句集中的子句之间是合取关系。因此，子句集中只要有一个子句为不可满足，则整个子句集就是不可满足的；**
  - **即：在解释I下，S中只要有一个子句的取值为假，则S不可满足；**
  - **空子句是不可满足的。因此，一个子句集中如果包含有空子句，则此子句集就一定是不可满足的。**

4.5.1 子句及子句集

- **7.定理4.1：（谓词公式不可满足性与子句集不可满足性的关系）**
  - **设有谓词公式F，其标准子句集为S，则F为不可满足的充要条件是S为不可满足的。**
  - **即：如果一个谓词公式是不可满足的，则其子句集也一定是不可满足的，反之亦然。**
    - **这样我们要证明一个谓词公式是不可满足的，只要能证明其相应的子句集是不可满足的就可以了。**
    - **但是谓词公式和其子句集表示并不一定是等值的。我们只是利用子句集不可满足时，原谓词公式一定不可满足的特点来帮助我们证明一个谓词公式是不可满足的。**

4.5.2 归结式（消解式）

- **1. 互补对（互补文字）**
  - 若P是原子谓词公式，则称P与¬P为互补对(文字)。
- **2. 归结式（消解式）**
  - 设有C_1和C_2两个子句，其中C_1 =P∨L，C_2=Q∨¬L，消去两个子句中的互补对（L 和¬L），两个子句剩下部分组成一个新的子句C_{12} =P∨Q，称子句C_{12} =P∨Q为C_1和C_2的归结式（消解式）。
  - 求归结式的归结树表示：

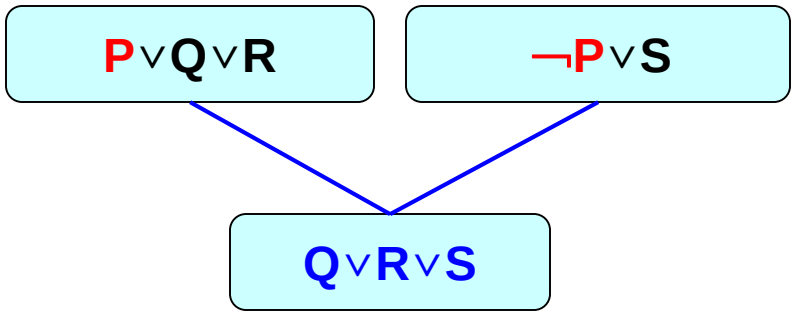


4.5.2 归结式（消解式）

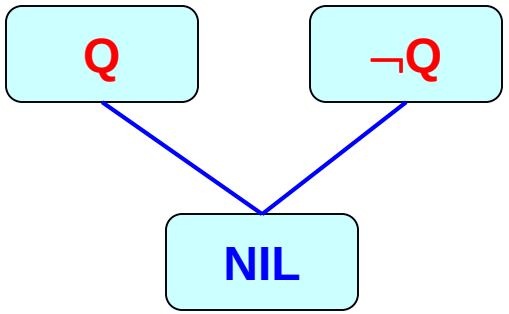
- **3. 求归结式注意事项**
  - **两个子句能够消解的先绝条件是两个子句中存在互补的文字对。**
  - **两个子句中若存在多个互补对，可根据需要先任选一对进行消解。**
  - **谓词逻辑子句中常含有变量，需要先进行置换、合一处理，然后消去两个子句中的互补对，得到谓词逻辑的归结式或消解式。**
  - **两个谓词逻辑子句消解时，要求两个子句中不能有相同的变量名称，如果出现相同变量名称，则要将其中一个子句的变量更名，然后做置换、合一处理，再进行消解。**

4.5.2 归结式（消解式）

- **【例4.5.2】 设C_1=P∨Q∨R，C_2=¬P∨S，求C_1和C_2的归结式C_{12}。**
  - **解：消去互补对P和¬P，归结式为：**
  - **C_{12}= Q∨R∨S**
- **【例4.5.3】 设C_1=¬Q， C_2 =Q，求C_1和C_2的归结式C_{12} 。**
  - **解：消去互补对¬Q和Q，得到归结式：C_{12}= NIL**



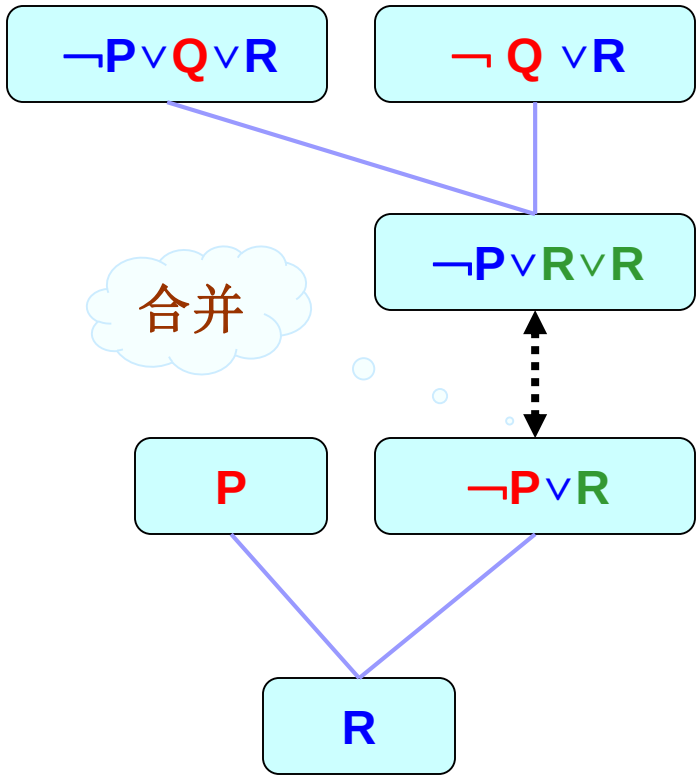
例4.5.2 归结树



例4.5.3 归结树

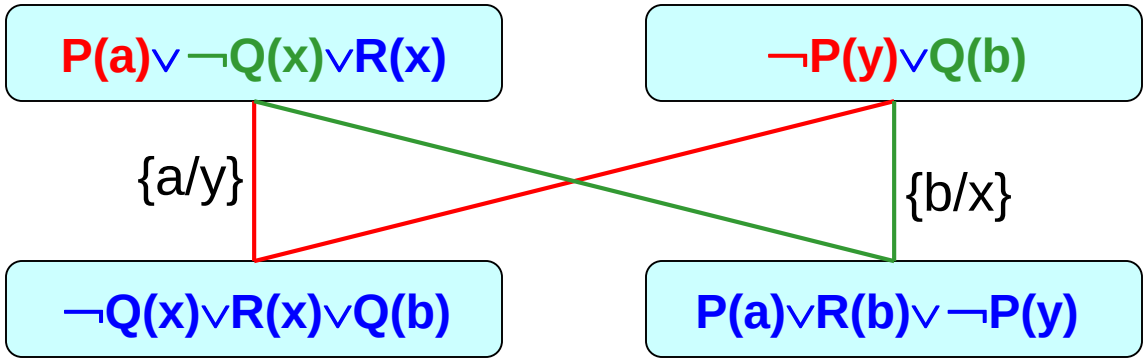
4.5.2 归结式（消解式）

- **【例4.5.4】 设C_1=¬P∨Q∨R，C_2=¬ Q ∨R， C_3=P，求归结式C_{123}。**
  - **解：先对C_1和C_2进行归结得归结式C_{12}； 最对C_{12}和C_3归结得归结式C_{123}=R。**
  - **归结树如图；**
  - **本例中也可以先对C_1和C_3 进行归结，再对C_{13} 和C_2 进行归结，结果相同。**



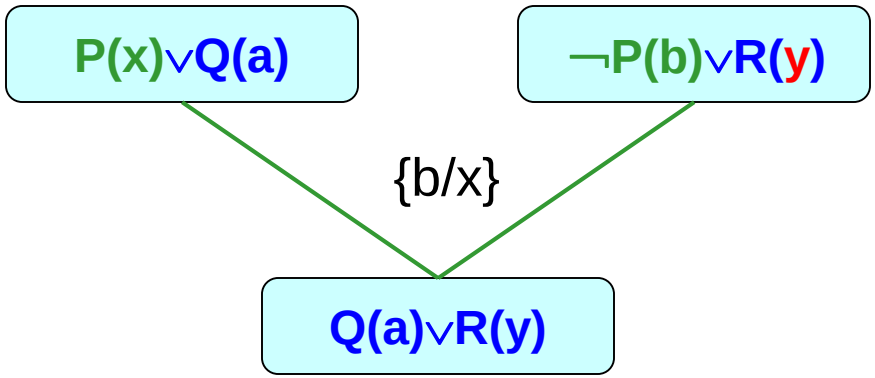
4.5.2 归结式（消解式）

- **【例4.5.5】 设C_1=P(a)∨¬Q(x)∨R(x)， C_2 =¬P(y)∨Q(b)，求归结式C_{12}。**
  - **解：先进行置换、合一；再归结。**
  - **归结树如图；**
  - **本例有2种归结方式。**



4.5.2 归结式（消解式）

- **【例4.5.6】 设C_1=P(x)∨Q(a)， C_2 =¬P(b)∨R(x)，假定a、b为常量，x为变量，求归结式C_{12}。**
  - **解：本题中两个子句有同名变量，归结之前需要先进行变量更名，比如把后一子句变量名变换为y。**
  - **其它同前。**



4.5.3 鲁滨逊归结原理

- **1. 定理4.2：**
  - **设子句C_1、C_2的归结式为C_{12}，则归结式C_{12}是C_1、C_2的逻辑结论。**
  - **即: 如果C_1、C_2同时为真， C_{12}一定为真，表示为:**

    C_1，C_2 => C_{12} 。
- **证明：**
  - **不妨设C_1、C_2的互补对为L和¬L ，则有： $C_1=C'_1 \vee L$，$C_2=C'_2 \vee \neg L$**
  - **归结式： $C_{12}=C'_1 \vee C'_2$**
  - **将C_1、C_2用等价的蕴含式表示：**
    - $C'_1 \vee L$ <=> $\neg C'_1 \rightarrow L$
    - $C'_2 \vee \neg L$ <=> $L \rightarrow C'_2$

4.5.3 鲁滨逊归结原理

- 根据假言三段式推理规则（链式）：
  - $\neg C'_1 \rightarrow L$，$L \rightarrow C'_2 \Rightarrow \neg C'_1 \rightarrow C'_2 \Leftrightarrow C'_1 \vee C'_2$
- 所以：C_1，$C_2 \Rightarrow C_{12}$，定理成立。
- 进一步论证：
  - 设$C_1=C'_1 \vee L$，$C_2=C'_2 \vee \neg L$关于解释I为真，则只需证明$C_{12}=C'_1 \vee C'_2$关于解释I也为真。
  - 对于解释I，L和¬L中必有一个为假。
  - 若L为假，则必有C'_1为真，不然就会使C_1为假，这将与前提假设C_1为真矛盾，因此只能有C'_1为真。
  - 同理，L为真，则¬L为假，则必有C'_2为真。
  - 因此，必有$C_{12}=C'_1 \vee C'_2$关于解释I也为真。即C_{12}是C_1和C_2的逻辑结论。

4.5.3 鲁滨逊归结原理

- ‘定理4.2’是归结原理中一个非常重要的定理，不仅对命题逻辑成立，在一阶谓词逻辑中也是适用的。
- 由这个定理得到下面两个重要的推论：

■ **2. 推论4.2.1：**

- C_1、C_2 是子句集S中的两个子句，C_{12}是C_1、C_2的归结式，用C_{12}取代S中的C_1、C_2，得到一个新的子句集S_1，如果S_1不可满足，则S一定不可满足。
- 即：S_1的不可满足性 ⇒ S的不可满足性

4.5.3 鲁滨逊归结原理

- **证明：**
  - 设S={ C_1，C_2，C_3，……，C_n}，C_{12}是C_1和C_2的归结式，则用C_{12}代替C_1和C_2后可得到一个新的子句集

    S_1={ C_{12}，C_3，…, C_n}
  - 设S_1是不可满足的，则对不满足S_1的任一解释I，都可能有以下两种情况：
    - ① 解释I使C_{12}为真，则C_3，……，C_n中必有一个为假，即S是不可满足的。
    - ② 解释I使C_{12}为假，根据‘定理4.2’，在解释I下，C_1和C_2为中必有一个为假。即S也是不可满足的。
  - 因此可以得出

    S_1的不可满足性⇒S的不可满足性

4.5.3 鲁滨逊归结原理

- **3. 推论4.2.2：**
  - **设C_1和C_2是子句集S中的两个子句，C_{12}是C_1和C_2的归结式，若把C_{12}加入S中得到新的子句集S_2，则S与S_2的不可满足性是等价的。即：**

    S_2的不可满足性 ⇔ S的不可满足性

- **证明：**
  - **(1) S不可满足则S_2不可满足**
    - **设S={ C_1，C_2，C_3，…，C_n}是不可满足的，则C_1，C_2，C_3，…，C_n中必有一子句为假，**
    - **因而S_2={ C_{12}，C_1，C_2，C3，……，C_n}必为不可满足。**

4.5.3 鲁滨逊归结原理

- **(2) S_2 不可满足则S不可满足**
  - **设S_2是不可满足的,则对不满足S_2的任一解释I,都可能有以下两种情况:**
  - **① 解释I使C_{12}为真，则C_1，C_2，C_3，…，C_n中必有一个为假，即S是不可满足的。**
  - **② 解释I使C_{12}为假，根据‘定理4.2’，C_1和C_2中必有一个为假，即S也是不可满足的。**
- **由此可见S与S_2的不可满足性是等价的。即**

S的不可满足性 ⇔ S_2的不可满足性

4.5.3 鲁滨逊归结原理

- **根据这两个推论，要证明子句集S的不可满足性，可以通过归结的方法，对S中能够归结的子句进行归结。**
- **用产生的归结式取代原来的子句得到新的子句集S_1，或将产生的归结式加入到S中得到新的子句集S_2，只要能证明S_1或S_2是不可满足的，那么S就是不可满足的。**

4.5.3 鲁滨逊归结原理

- **要证明S_1或S_2是不可满足的，对其继续使用上述方法进行归结处理，如果能够归结产生出空子句，NIL，由于空子句是不可满足的、矛盾的，也就证明了S是不可满足的。**
- **所以要证明子句集S的不可满足性，可通过归结的方法试图归结出空子句，一旦归结出了空子句，也就证明了S是不可满足的，这就是归结原理的基本思想。**

4.5.3 鲁滨逊归结原理

- **4. 鲁滨逊归结原理**
  - **原命题：在给定前提P为真下，证明结论Q为真。即：P=>Q**
  - **(1) 假设原命题成立 — 即：P为真则Q为真；**
  - **(2) 反证法构造公式：P∧¬Q，则此公式一定不可满足。**
  - **(3) 证明P∧¬Q不可满足**

    通过证明P∧¬Q不可满足，而证明原命题成立。---- 反证（反演）
  - **(4) 将P∧¬Q化为子句集S表示；**

4.5.3 鲁滨逊归结原理

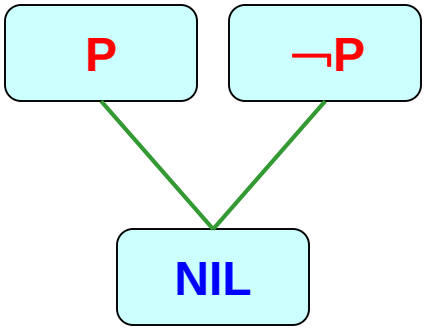
- (5) 对S反复进行归结，产生的新子句加入S：
  - 如果S中出现空子句（NIL），则子句集S不可满足，公式P∧¬Q 亦不可满足，原命题成立。
  - 如果S中没有新的归结，又没有出现空子句，则原命题不成立。

- 说明：
  - 为叙述方便，以上的前提条件是以单个谓词公式P给出的，实际问题中前提条件可能有多个，即给定前提条件为P_1，P_2，P_3，…，P_n同时为真，证明结论Q为真。即原命题为：$P_1 \wedge P_2 \wedge P_3 \wedge \ldots \wedge P_n \Rightarrow Q$
  - 处理过程同前，构造的公式为：

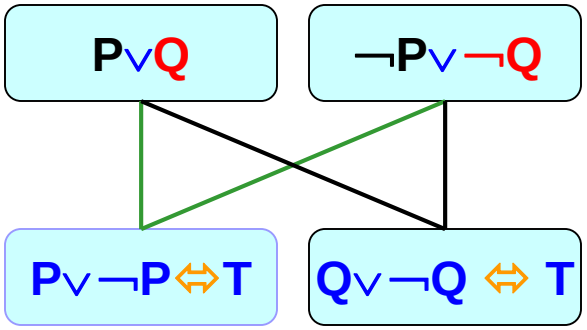
$$P_1 \wedge P_2 \wedge P_3 \wedge \ldots \wedge P_n \wedge \neg Q$$

4.5.3 鲁滨逊归结原理

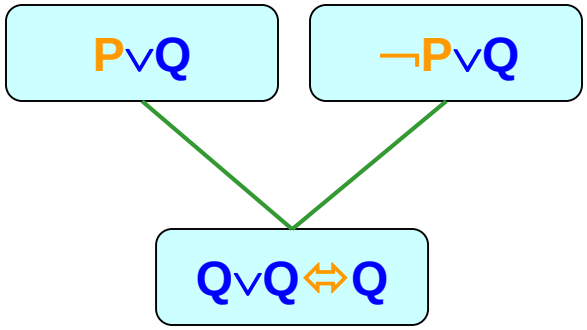
5. 常用推理规则的归结式表示



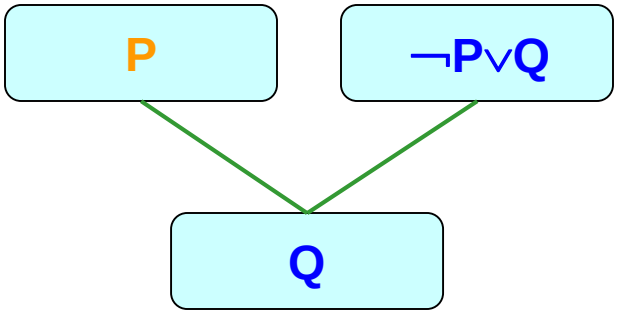
(1)空子句



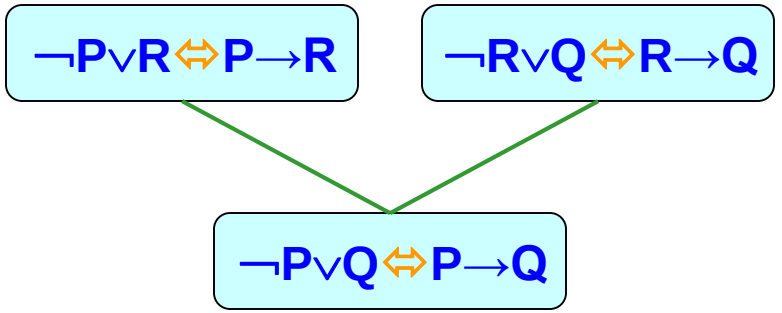
(2)永真式空子句



(3)合并



(4)假言推理



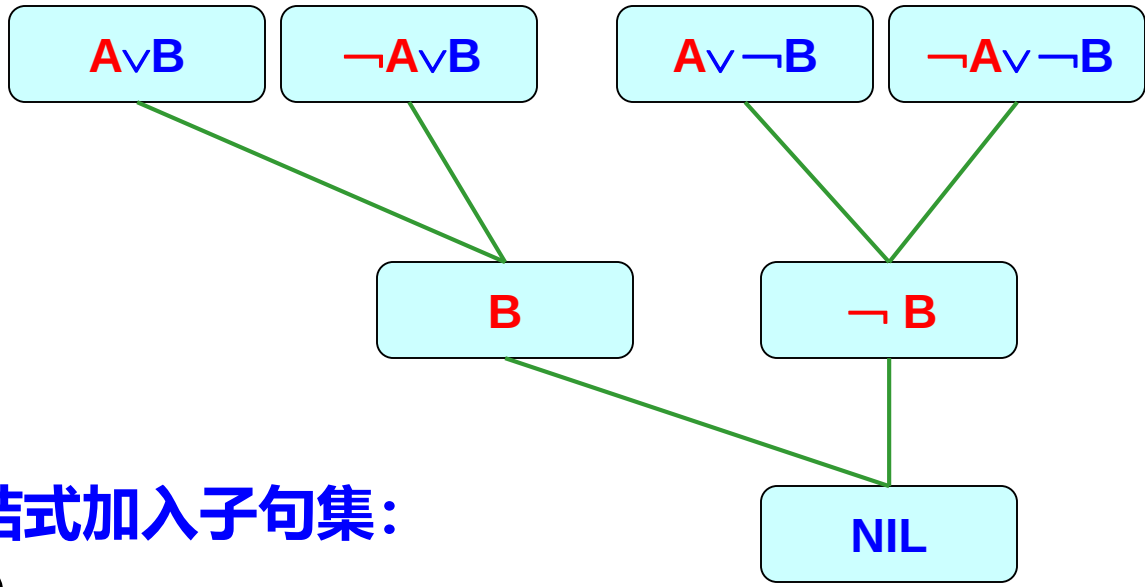
(5)假言三段式(链式)

4.5.4 归结反演证明

- **归结反演证明过程**
  - **命题：在给定前提P为真下，证明结论Q为真。即：P=>Q**
  - **(1) 构造公式： $P \wedge \neg Q$**
  - **(2) 将公式$P \wedge \neg Q$化为子句集S表示；**
  - **(3) 反复对子句集S进行归结处理，并把新产生的归结式加入到S中：**
    - **如果出现空子句，则停止归结，原命题得证，即证明了P为真时，Q为真。**
    - **否则，没有新的归结，原命题不成立。**

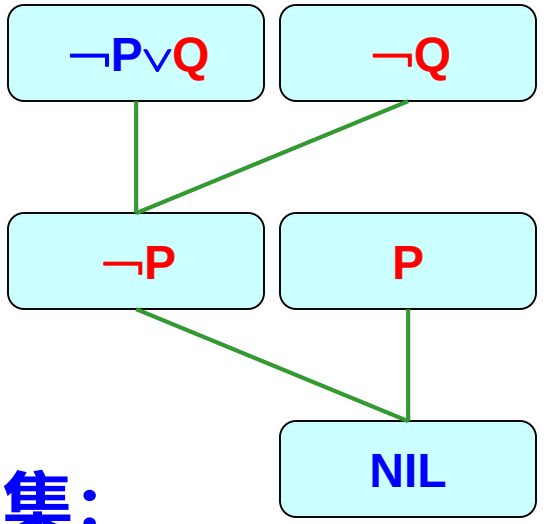
4.5.4 归结反演证明

- **例4.5.7： 证明子句集S={A∨B, ¬A∨B, A∨¬B,¬A∨¬B}是不可满足的。**
- **证明：**
  - **原有子句：**
    - ①A∨B
    - ②¬A∨B
    - ③A∨¬B
    - ④¬A∨¬B
  - **进行归结处理，归结式加入子句集：**
    - ⑤B -- 归结①、②。
    - ⑥¬B -- 归结③、④。
    - ⑦NIL -- 归结⑤、⑥。
  - **出现空子句，S不可满足。归结树如图 。**



4.5.4 归结反演证明

- **例4.5.8 证明 (¬P∨Q)∧¬Q => ¬P。**
- **证明：**
  - **否定目标得 P，并化为子句集：**
    - **①¬P∨Q**
    - **②¬Q**
    - **③ P**
  - **进行归结处理，归结式加入子句集：**
    - **④¬P -- 归结①、②。**
    - **⑤ NIL -- 归结③、④。**
  - **出现空子句，原命题成立。归结树如图**

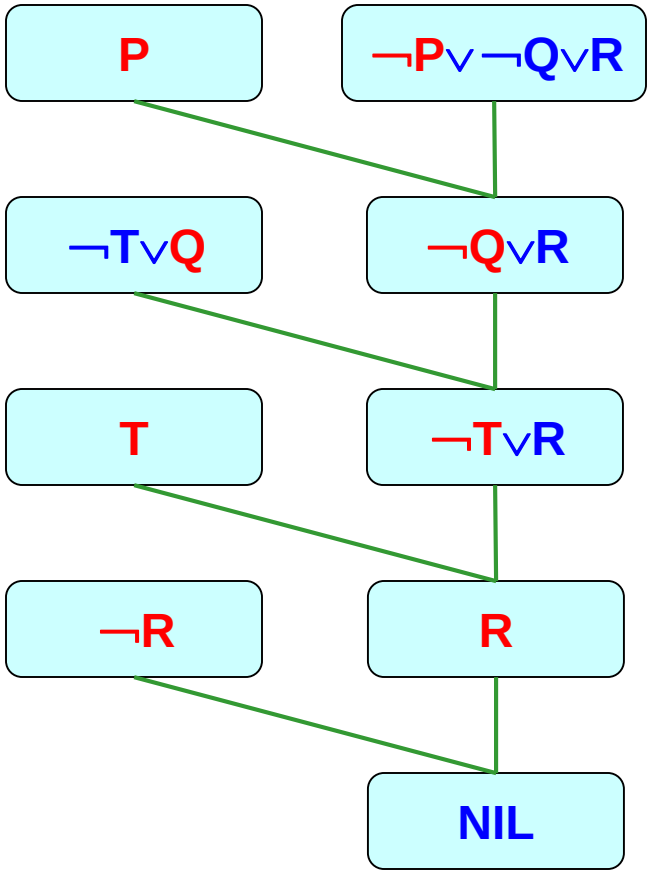


4.5.4 归结反演证明

- **例4.5.9 设已知的公式集为{ P, (P∧Q)→R, (S∨T)→Q, T }，求证结论R。**
- **证明：**
  - **分析：**
  - **本题给出的前提条件公式有4个，即在此4个条件公式同时为真时，证明结论R为真。**
  - **将4个条件公式中的每一个公式单独化为子句集表示；**
  - **再将目标公式否定，即¬R，化为子句集表示；**
  - **由全部子句构成证明的子句集S。**

4.5.4 归结反演证明

- **原有子句：**
  - ①P
  - ②¬P∨¬Q∨R
  - ③¬S∨Q – 未用冗余子句
  - ④¬T∨Q
  - ⑤T
  - ⑥¬R
- **进行归结处理，归结式加入子句集：**
  - ⑦¬Q∨R -- 归结①、②
  - ⑧¬T∨R -- 归结④、⑦
  - ⑨R -- 归结⑤、⑧
  - ⑩ NIL -- 归结⑥、⑨
  - **出现空子句，S不可满足。归结树如图 。**



4.5.4 归结反演证明

- **例4.5.10 已知:**
  - **F: $(\forall x)\{[(\exists y)(A(x, y) \wedge B(y))] \rightarrow (\exists y)[C(y) \wedge D(x, y]\}$**
  - **G: $\neg(\exists x)C(x) \rightarrow (\forall x)(\forall y)[A(x, y) \rightarrow \neg B(y)]$**

  求证G是F的逻辑结论。
- **证明:**
  - **分析:**
    - **否定G，构造公式$F \wedge \neg G$;**
    - **将公式$F \wedge \neg G$化为子句集S;**
    - **注意：化子句集时可对F和$\neg G$单独处理，将所有子句放入S中。**

4.5.4 归结反演证明

- **(1) F化为子句集表示**
  - $(\forall x)\{\neg[(\exists y)(A(x, y)\wedge B(y))]\vee(\exists y)[C(y)\wedge D(x, y]\}$
  - $(\forall x)\{(\forall y)[\neg A(x, y)\vee\neg B(y)]\vee(\exists y)[C(y)\wedge D(x, y]\}$
  - **变量更名，另后面的y=w；**
  - $(\forall x)\{(\forall y)[\neg A(x, y)\vee\neg B(y)]\vee(\exists w)[C(w)\wedge D(x, w)]\}$
  - **Skolem 化，另w=f(x)**
  - $(\forall x)\{(\forall y)[\neg A(x, y)\vee\neg B(y)]\vee[C(f(x))\wedge D(x, f(x))]\}$
  - **前束化、消去全称量词**
  - $[\neg A(x, y)\vee\neg B(y)]\vee[C(f(x))\wedge D(x, f(x))]$

4.5.4 归结反演证明

- F化为2个子句
  - ①¬A(x, y)∨¬B(y)∨C(f(x))
  - ②¬A(x1, y1)∨¬B(y1)∨D(x1, f(x1))

- (2) ¬G化为子句集表示
  - ¬{¬(∃x)C(x)→(∀x)(∀y)[A(x, y)→¬B(y)]}
  - ¬{(∃x)C(x)∨(∀x)(∀y)[¬A(x, y)∨¬B(y)]}
  - (∀x)¬C(x)∧(∃x)(∃y)[A(x, y)∧B(y)]
  - 变量更名，另后面x=w；
  - 消去存在量词：另w=a，y=b，a、b为常量符号。
  - (∀x)¬C(x)∧A(a, b)∧B(b)
  - 前束化，消去全称量词：
  - ¬C(x)∧A(a, b)∧B(b)

4.5.4 归结反演证明

- **¬G化为3个子句**

③¬C(x2)

④A(a, b)

⑤B(b)

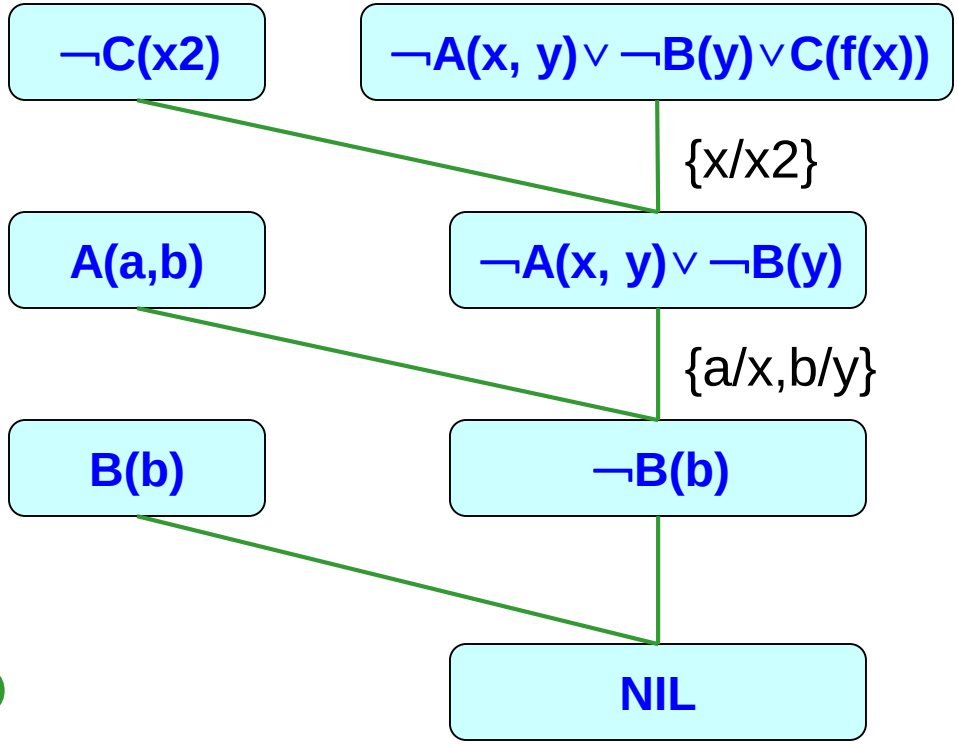
- **归结处理过程**

⑥¬A(x, y)∨¬B(y)

-- 归结①、③

⑦¬B(b) -- 归结④、⑥

⑧NIL -- 归结⑤、⑦



4.5.4 归结反演证明

- **例4.12 已知：**
  - F_1：$(\forall x)\{P(x)\rightarrow(\forall y)(Q(y)\rightarrow\neg L(x,y))\}$
  - F_2：$(\exists x)\{P(x)\wedge(\forall y)(R(y)\rightarrow L(x,y))\}$
  - G：$(\forall x)(R(x)\rightarrow\neg Q(x))$

  求证：G是F_1和F_2的逻辑结论。
- **证明：**
  - **分析：**
    - **否定G，构造公式 $F_1\wedge F_1\wedge\neg G$；**
    - **将公式$F_1\wedge F_1\wedge\neg G$化为子句集S；**
    - **注意：化子句集时可对F_1、 F_2和$\neg G$单独处理，将所有子句放入S中**

4.5.4 归结反演证明

- **(1) F_1化为子句集表示**
  - ① ¬P(x)∨¬Q(y)∨¬L(x,y)
- **(2) F_2化为子句集表示**
  - ② P(a)；a为Skolem化常量。
  - ③ ¬R(y1)∨L(a,y1)
- **(3) ¬G化为子句集表示**
  - ④ R(b)；b为Skolem化常量。
  - ⑤ Q(b)

4.5.4 归结反演证明

- **归结处理过程**

⑥ ¬Q(y)∨¬L(a,y) -- 归结①、②

⑦ ¬R(y)∨¬Q(y) -- 归结③、⑥

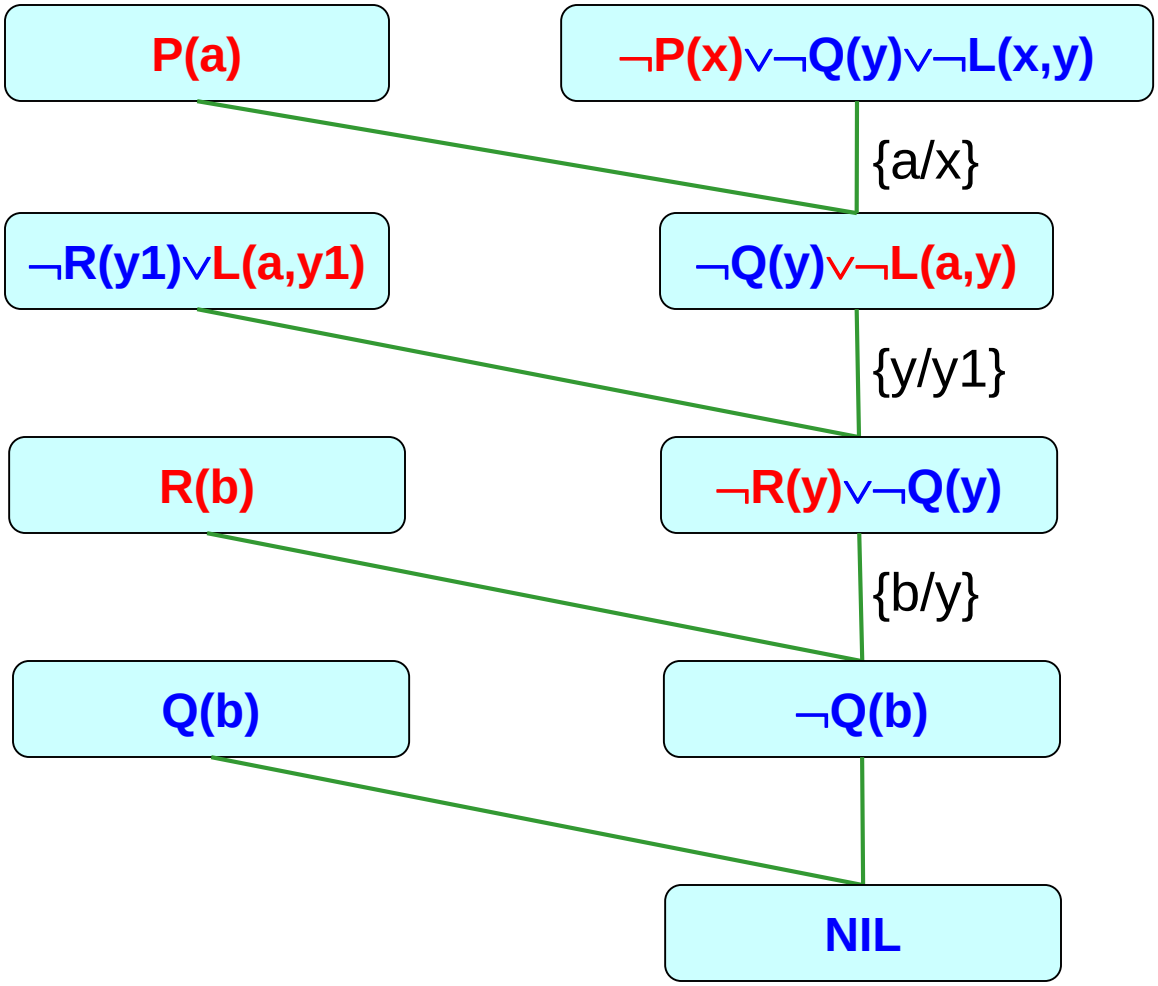
⑧ ¬Q(b) -- 归结④、⑦

⑨ NIL -- 归结⑤、⑧

- **归结树见下页图示。**

- **提示：**
  - **归结次序不是唯一的，归结树也不是唯一的；**
  - **归结的最终结果是一致的。**

4.5.4 归结反演证明



4.5.4 归结反演证明

- **例4.5.13：设a、b、c三人中有人从不说真话，有人从不说假话，某人分别向这三人提出同一个问题：谁是说谎者？a答：‘b和c都是说谎者’；b答：‘a和c都是说谎者’；c答：‘a和b中至少一个是说谎者’。证明c是说真话者。**

- **证明：设T(x)表示x是说真话者，那么¬T(x)表示说谎者。**
  - 根据题意，可以列出下列条件公式：
  - 如果a说真话，则有：$T(a) \rightarrow (\neg T(b) \wedge \neg T(c))$；
  - 如果a说假话，则有：$\neg T(a) \rightarrow (T(b) \vee T(c))$；
  - 如果b说真话，则有：$T(b) \rightarrow (\neg T(a) \wedge \neg T(c))$；
  - 如果b说假话，则有：$\neg T(b) \rightarrow (T(a) \vee T(c))$；
  - 如果c说真话，则有：$T(c) \rightarrow (\neg T(a) \vee \neg T(b))$；
  - 如果c说假话，则有：$\neg T(c) \rightarrow (T(a) \wedge T(b))$。
  - 目标公式为：T(c)。

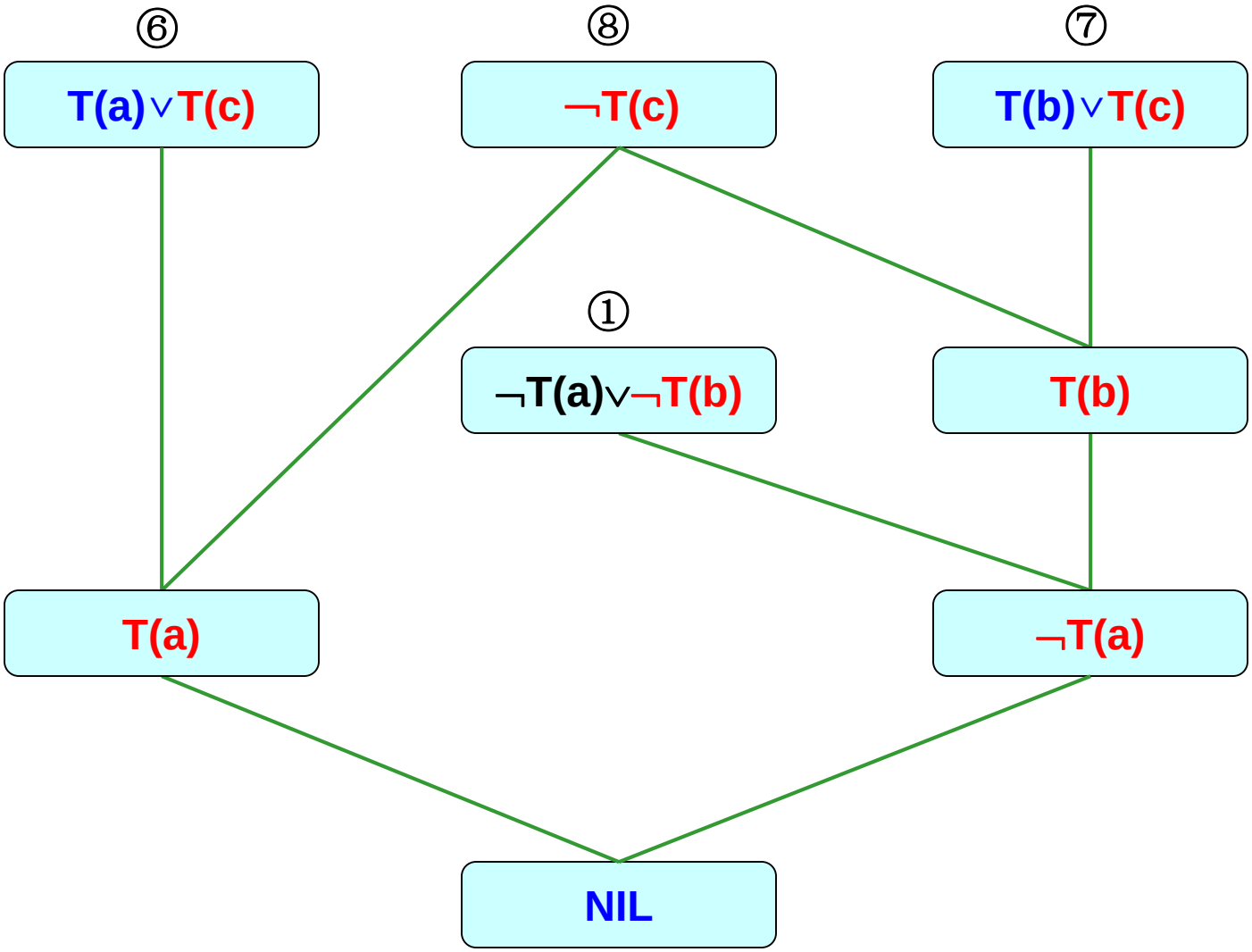
4.5.4 归结反演证明

- **将条件公式和目标的否定化为子句形式：**
  - ① ¬T(a)∨¬T(b)
  - ② ¬T(a)∨¬T(c)
  - ③ T(a)∨T(b)∨T(c)
  - ④ ¬T(b)∨¬T(c)
  - ⑤ ¬T(a)∨¬T(b)∨¬T(c)
  - ⑥ T(a)∨T(c)
  - ⑦ T(b)∨T(c)
  - ⑧ ¬T(c)

4.5.4 归结反演证明

- 将⑧分别与⑥、⑦进行归结，得到两个归结式T(a)和T(b)。将T(a)与①归结，归结式为¬T(b)。再将T(b)与¬T(b)归结，得到空子句。
- 所以c是说真话者，同理可以证明a、b都是说谎话者。
- 本例中只使用了①、 ⑥、⑦、 ⑧4个原始子句。其它4个子句在本归结过程中冗余。当然本例有多种归结过程。
- 归结树如下图。

4.5.4 归结反演证明



4.5.4 归结反演证明

- **例4.5.14：假设：所有不贫穷并且聪明的人是快乐的。喜欢读书的人是聪明的。快乐的人过着激动人心的生活。li喜欢读书并且不贫穷。证明：li过着激动人心的生活。**
- **证明：**
  - **定义谓词：**
    - **Poor(x) – 表示x是贫穷的；**
    - **Smart(x) – 表示x是聪明的；**
    - **Happy(x) – 表示x是快乐的；**
    - **Read(x) – 表示x喜欢读书；**
    - **Exciting(x) – 表示x过着激动人新的生活。**

4.5.4 归结反演证明

- **根据题意，可列出以下条件公式：**
  - **(∀x){ (¬Poor(x)∧Smart(x))→Happy(x) }**
  - **(∀x){ Read(x)→Smart(x) }**
  - **(∀x){ Happy(x)→Exciting(x) }**
  - **Read(li)∧¬Poor(li) -- 事实(小前提)**
- **目标公式为：**
  - **Exciting(li)**

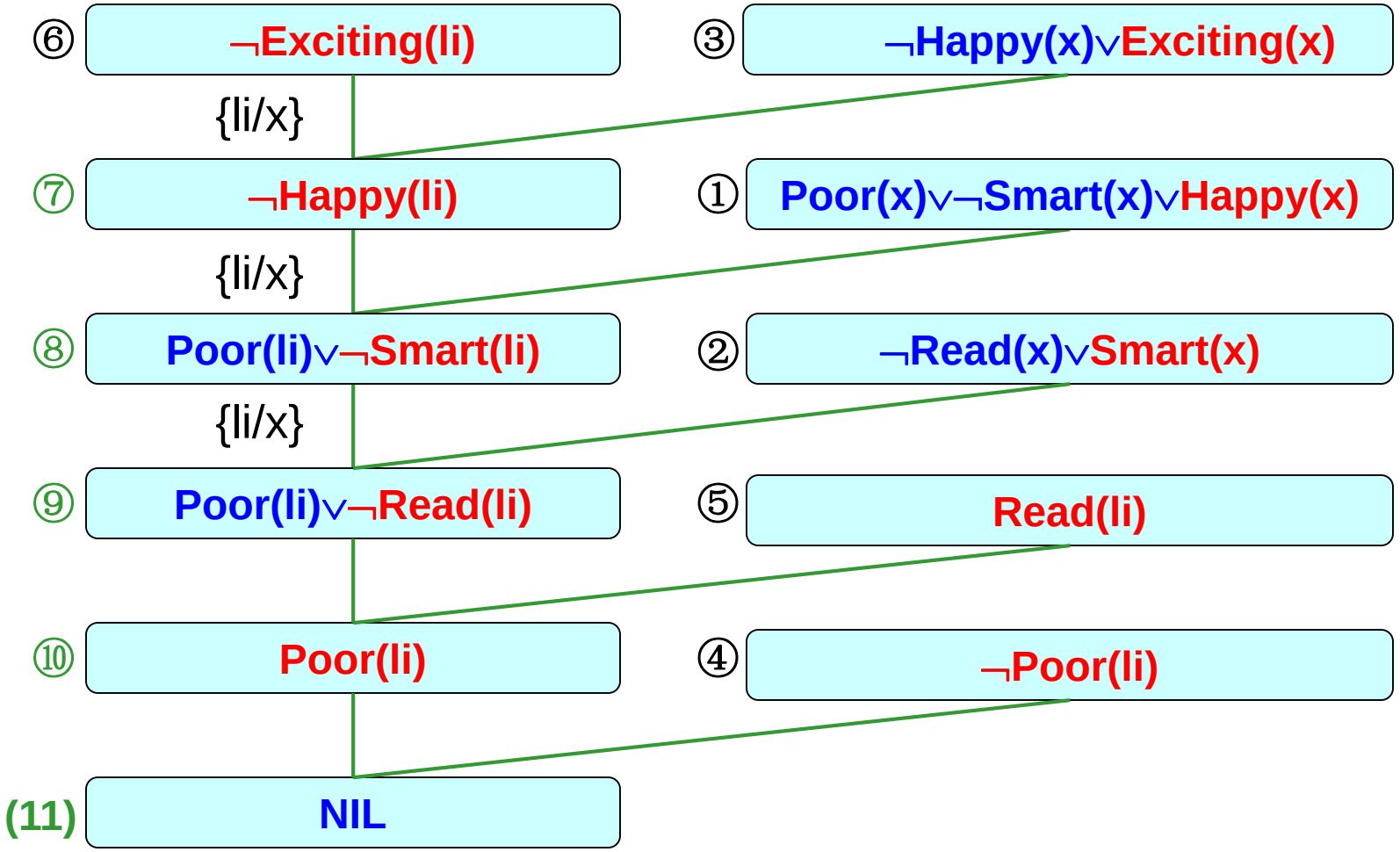
4.5.4 归结反演证明

- **将条件公式和目标的否定化为子句形式：**
  - ① Poor(x)∨¬Smart(x)∨Happy(x)
  - ② ¬Read(x)∨Smart(x)
  - ③ ¬Happy(x)∨Exciting(x)
  - ④ ¬Poor(li)
  - ⑤ Read(li)
  - ⑥ ¬Exciting(li)

4.5.4 归结反演证明

- **归结处理：**
  - ⑦ ¬Happy(li) -- 归结③、⑥，作置换{li/x}。
  - ⑧ Poor(li)∨¬Smart(li) -- 归结① 、 ⑦ ， 作置换{Bill/x}。
  - ⑨ Poor(li)∨¬Read(li) -- 归结② 、 ⑧ ， 作置换{Bill/x}。
  - ⑩ Poor(li) -- 归结⑤ 、 ⑨。
  - (11)NIL -- 归结④ 、 ⑩，
  - 得到空子句，所以li过着激动人心的生活，归结树如下页图。

4.5.4 归结反演证明


⑥ ¬Exciting(li)
③ ¬Happy(x)∨Exciting(x)
{li/x}
⑦ ¬Happy(li)
① Poor(x)∨¬Smart(x)∨Happy(x)
{li/x}
⑧ Poor(li)∨¬Smart(li)
② ¬Read(x)∨Smart(x)
{li/x}
⑨ Poor(li)∨¬Read(li)
⑤ Read(li)
⑩ Poor(li)
④ ¬Poor(li)
(11) NIL

4.5.4 归结反演证明

- **例4.5.15：假设：任何通过计算机考试并获奖的人都是快乐的，任何肯学习或幸运的人都可以通过所有考试，李不肯学习但他是幸运的，任何幸运的人都能获奖。**

  求证：李是快乐的。

- **解：**
  - **定义谓词：**
    - **Pass(x, y)　x可以通过y考试**
    - **Win(x, prize)　x能获得奖励**
    - **Study(x)　x肯学习**
    - **Happy(x)　x是快乐的**
    - **Lucky(x)　x是幸运的**

4.5.4 归结反演证明

- **再将问题用谓词表示如下：**
  - **“任何通过计算机考试并获奖的人都是快乐的”**

$(\forall x)(Pass(x, computer) \wedge Win(x, prize) \rightarrow Happy(x))$

  - **“任何肯学习或幸运的人都可以通过所有考试”**

$(\forall x)\ (\forall\ y)\ (Study(x) \vee Lucky(x) \rightarrow Pass(x, y))$

  - **“李不肯学习但他是幸运的”**

$\neg Study(li) \wedge Lucky(li)$

  - **“任何幸运的人都能获奖”**

$(\forall x)\ (Lucky(x) \rightarrow Win(x, prize))$

  - **结论“李是快乐的”的否定**

$\neg Happy(li)$

4.5.4 归结反演证明

- **将上述谓词公式转化为子句集如下：**

① ¬Pass(x, computer)∨¬Win(x, prize)∨Happy(x)

② ¬Study(y)∨Pass(y, z)

③ ¬Lucky(u)∨Pass(u, v)

④ ¬Study(li)

⑤ Lucky(li)

⑥ ¬Lucky(w)∨Win(w, prize)

⑦ ¬ Happy(li)　(结论的否定)

4.5.5 归结反演求解

- 归结反演的方法不仅可以向上面讨论的，用来证明问题，稍加改进就可以用来求取问题的解答。
- 求取问题解答时，命题总是成立的，如果P为前提条件公式集，Q为目标公式，也就是要求取得解答。由于原命题成立，总能将{ P, ¬Q }对应的子句集归结到一个空子句。
- 构造公式集{P, ¬Q∨Q }，由于{ P, ¬Q }能归结到空子句，NIL，所以{P, ¬Q∨Q }沿着{ P, ¬Q }归结到空子句的路径进行归结，最后会得到NIL∨Q=Q的结果，即得到问题的解答。

4.5.5 归结反演求解

- **其一般步骤为：**
  - **(1) 把问题的已知条件和目标用谓词公式表示出来；**
  - **(2) 构造公式： {P, ¬Q∨Q }；**
  - **(3) 将{P, ¬Q∨Q }化为子句集S**
  - **(4)对子句集S应用归结原理求出其归结树，这时归结树的根子句不为空，称这个归结树为修改的证明树；**
  - **(5) 用修改证明树的根子句作为回答语句，则答案就在此根子句中。**

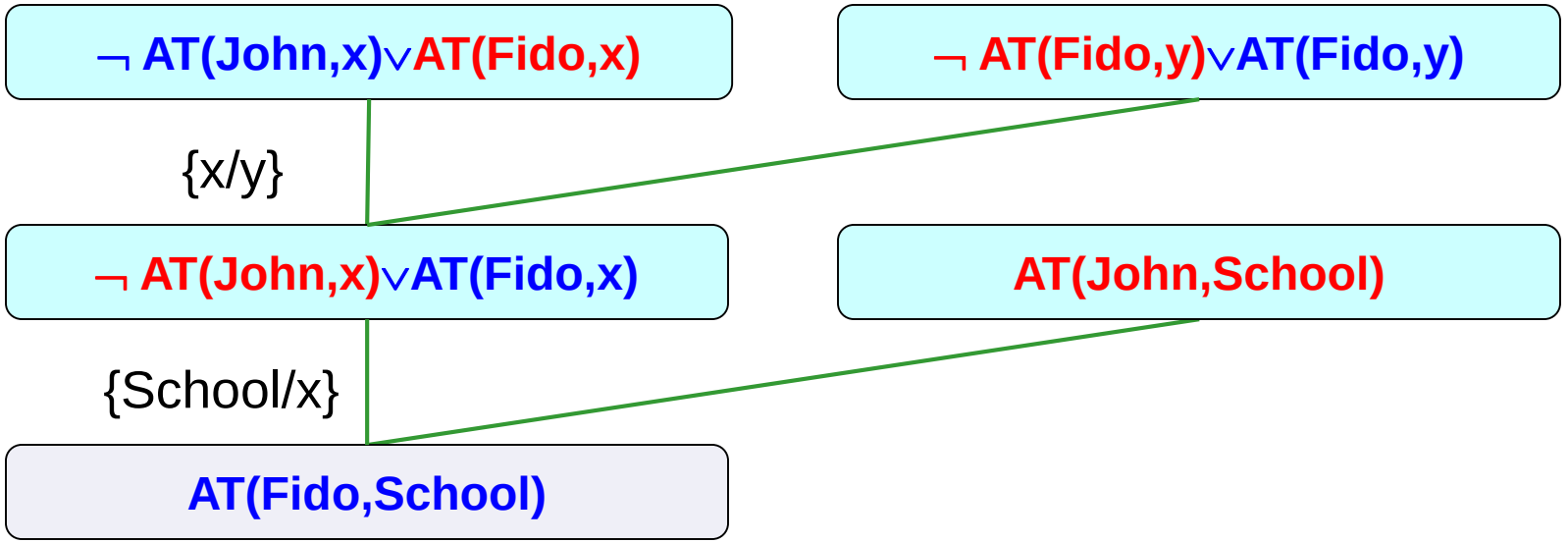
4.5.5 归结反演求解

- **例4.5.16：　Fido是John的爱犬，无论John到那里，Fido也去那里；如果John在学校，Fido在那里?**
- **解：引入谓词**
  - **AT(x,y)—表示x在y地方。**
  - **根据题意，可列出以下条件公式：**
    - **(∀x){ AT(John,x) → AT(Fido,x) }**
    - **AT(John,School)**
  - **目标公式：**
    - **(∃y)AT(Fido,y)**

4.5.5 归结反演求解

- **条件公式化为子句形式：**
  - **① ¬ AT(John,x)∨AT(Fido,x)**
  - **② AT(John,School)**
- **目标析取目标的否定公式化为子句形式：**
  - **③ ¬AT(Fido,y) ∨ AT(Fido,y)**
- **归结处理：**
  - **④¬AT(John,x)∨AT(Fido,x) --归结①、③，{x/y}。**
  - **⑤AT(Fido,School) -- 归结②、④，{ School/x }**
  - **AT(Fido,School)即为问题的解答。**
  - **归结树如下图。**

4.5.5 归结反演求解


¬ AT(John,x)∨AT(Fido,x)
¬ AT(Fido,y)∨AT(Fido,y)
{x/y}
¬ AT(John,x)∨AT(Fido,x)
AT(John,School)
{School/x}
AT(Fido,School)

4.5.5 归结反演求解

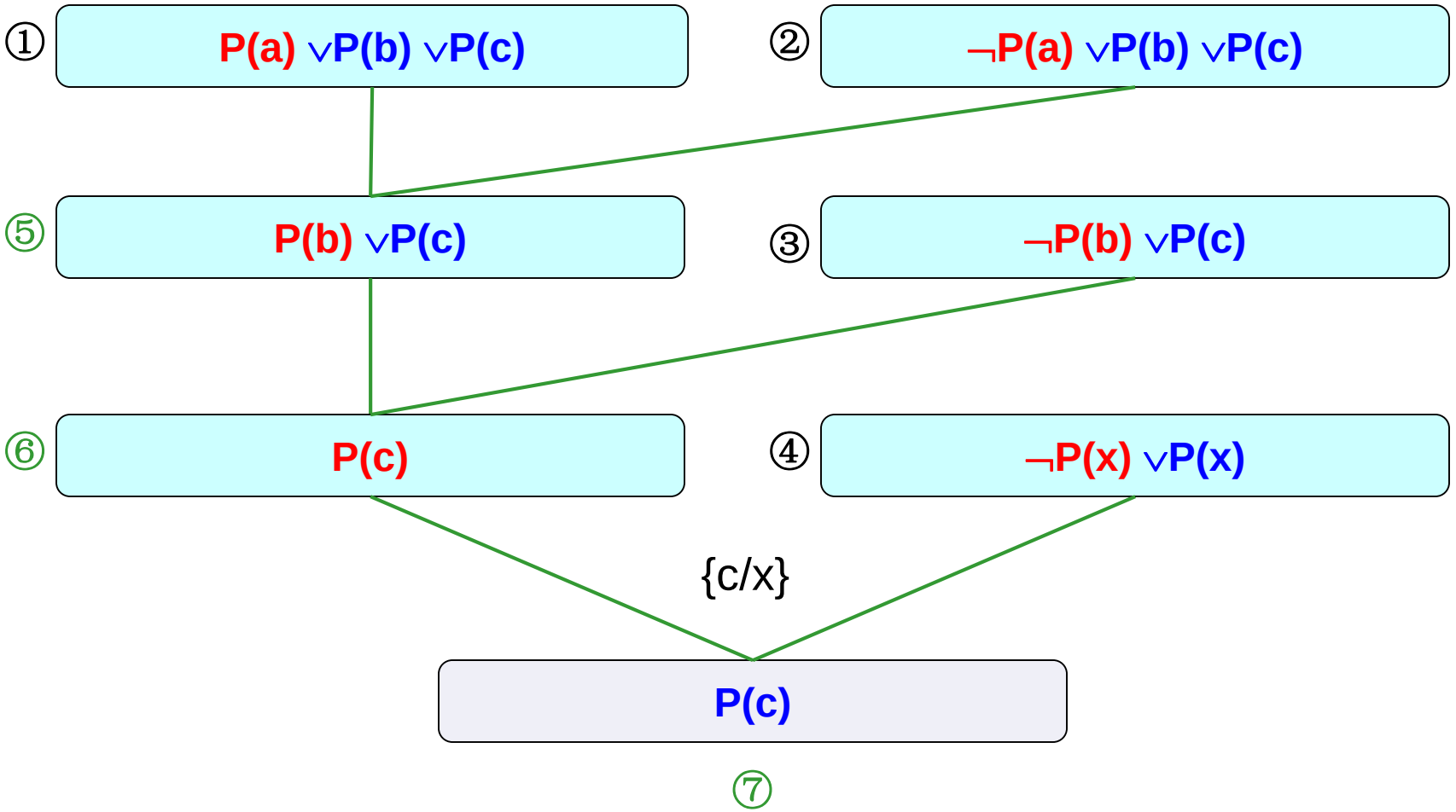
- **例4.5.17：某公司招聘工作人员，a、b、c三人应试，面试后公司表示如下想法：**
  - **① 三人中至少一人录取；**
  - **② 如果录取a而不录取b，则一定录取c；**
  - **③ 如果录取b，则一定录取c。**

  请问公司至少录取了谁?
- **解：定义谓词P(x)，表示录取了x。**
  - **条件公式为：**
    - **$P(a) \vee P(b) \vee P(c)$**
    - **$(P(a) \wedge \neg P(b)) \rightarrow P(c)$**
    - **$P(b) \rightarrow P(c)$**
  - **目标公式为：**
    - **$(\exists x)P(x)$**

4.5.5 归结反演求解

- **条件公式化为子句形：**
  - **① P(a) ∨P(b) ∨P(c)**
  - **② ¬P(a) ∨P(b) ∨P(c)**
  - **③ ¬P(b) ∨P(c)**
- **目标析取目标的否定，化为子句形：**
  - **④ ¬P(x) ∨P(x)**
- **归结处理：**
  - **⑤ P(b) ∨P(c)；归结①、②。**
  - **⑥ P(c)；归结③、⑤。**
  - **⑦ P(c)；归结④、⑥。**
- **结果为P(c)，即公司录用了c。归结树如下图。**

4.5.5 归结反演求解

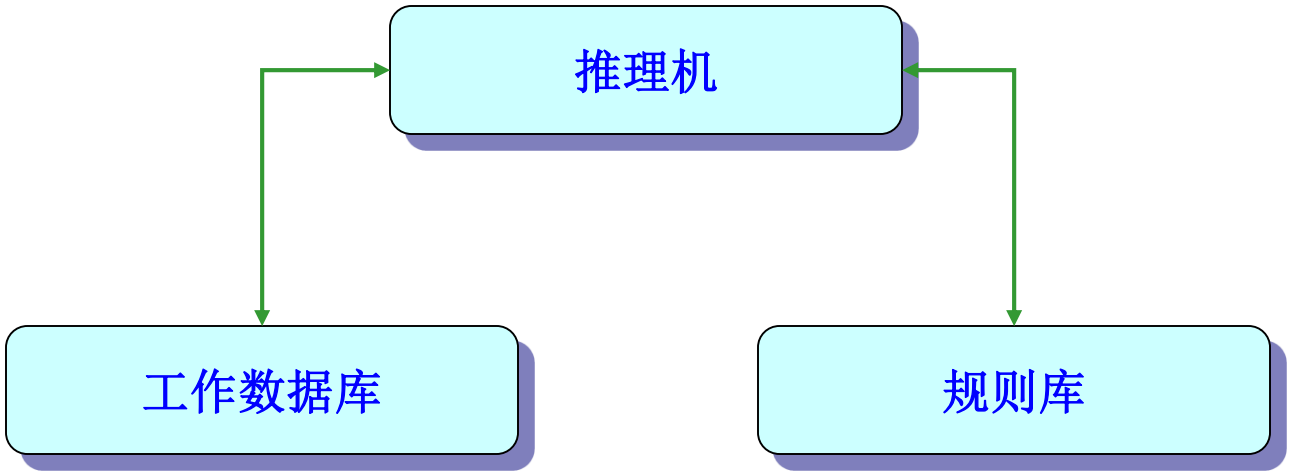


4.6 产生式推理系统

- **4.6.1 产生式系统的构成**
- **4.6.2 产生式正向推理**
- **4.6.3 产生式逆向推理**
- **4.6.4 混合推理**

4.6.1 产生式系统的构成

- **产生式系统由规则库、工作数据库和推理机三个部分构成，结构组成如图：**



4.6.1 产生式系统的构成

- **1. 规则库**
  - **又叫知识库**
  - **Production Rule Base，Knowledge Base**
  - **规则库用来存放系统的所有产生式规则；**
  - **用来表示问题领域的一般知识（背景知识）。**

4.6.1 产生式系统的构成

- **2. 工作数据库**
  - 又叫事实库、总数据库、工作区、工作内存、综合数据库、上下文、黑板等；
  - 用来存放**事实知识、中间结论、最终结果**
    - **事实知识**：即给定的推理前提条件。可以在推理前全部输入；或在推理过程中交互输入；也可以从其它系统导入。
    - **中间结论**：正向推理中是已经证明为真的证据；逆向推理中是匹配产生的中间目标。
    - **最终结果**：推理的最终目标或结论。
  - 总数据库中的内容在系统工作时是动态变化的。

4.6.1 产生式系统的构成

- **规则库是产生式系统问题求解的基础。**
- **比如，正向推理时：**
  - **每次从总数据库中选择一条事实、或中间结果（证据）；**
  - **到规则库中检索前件匹配的所有规则；**
  - **选择一条规则匹配，此规则激活（triggered）；**
  - **其后件若为结论直接写入总数据库，若为动作则执行该动作，执行的结果写入总数据库。**

4.6.1 产生式系统的构成

- **3. 推理机（控制策略）**
  - **Reasoning Machine**
  - **完成推理控制策略。负责整个产生式系统的运行，决定问题求解过程的推理线路。**
  - **控制系统的主要任务：**
  - **(1) 选择匹配：**
    - **按一定策略从规则库种选择规则与综合数据库中的已知事实进行匹配。**
    - **正向推理匹配：匹配是指把所选规则的前提与综合数据库中的已知事实进行比较，若事实库中的事实与所选规则前提一致，则称匹配成功，该规则为可用；否则，称匹配失败，该规则不可用。**

4.6.1 产生式系统的构成

- **逆向推理匹配：匹配是指选择中间目标，按先事实、后规则次序匹配；**
- **规则匹配指用所选中间目标与规则后件进行比较；若中间目标与规则后件一致，则称匹配成功，该规则为可用；否则，称匹配失败，该规则不可用。**

☞ **(2)冲突消解：**

- **对匹配成功的规则，按照某种策略从中选出一条规则执行。**

☞ **(3) 执行操作：**

- **对所执行的规则，若其后件为一个或多个结论，则把这些结论加入综合数据库；若其后件为一个或多个操作时，执行这些操作。**

4.6.1 产生式系统的构成

- **(4) 终止推理：**
  - **检查综合数据库中是否包含有目标，若有，则停止推理。**
- **(5)路径解释：**
  - **在问题求解过程中，记住应用过的规则序列，以便最终能够给出问题的解的路径。**

4.6.2 产生式正向推理

- **也称数据驱动方式，它是从已知事实出发，正向匹配规则，向着结论前进的一种推理方法。**
- **正向推理过程：**
  - **①初始化综合数据库，即把欲解决问题的已知事实、目标输入综合数据库中；**
  - **②检查规则库中是否有未使用过的规则，若没有，失败，退出；**
  - **③检查规则库的未使用规则中是否有其前提可与综合数据库中已知事实相匹配的规则，若有，形成当前可用（候选）规则集；否则转⑥；**

4.6.2 产生式正向推理

- ④ 按照冲突消解策略，从当前可用（候选）规则集中选择一个规则执行，并对该规则作上标记。把执行该规则后所得到的结论作为新的证据加入综合数据库；

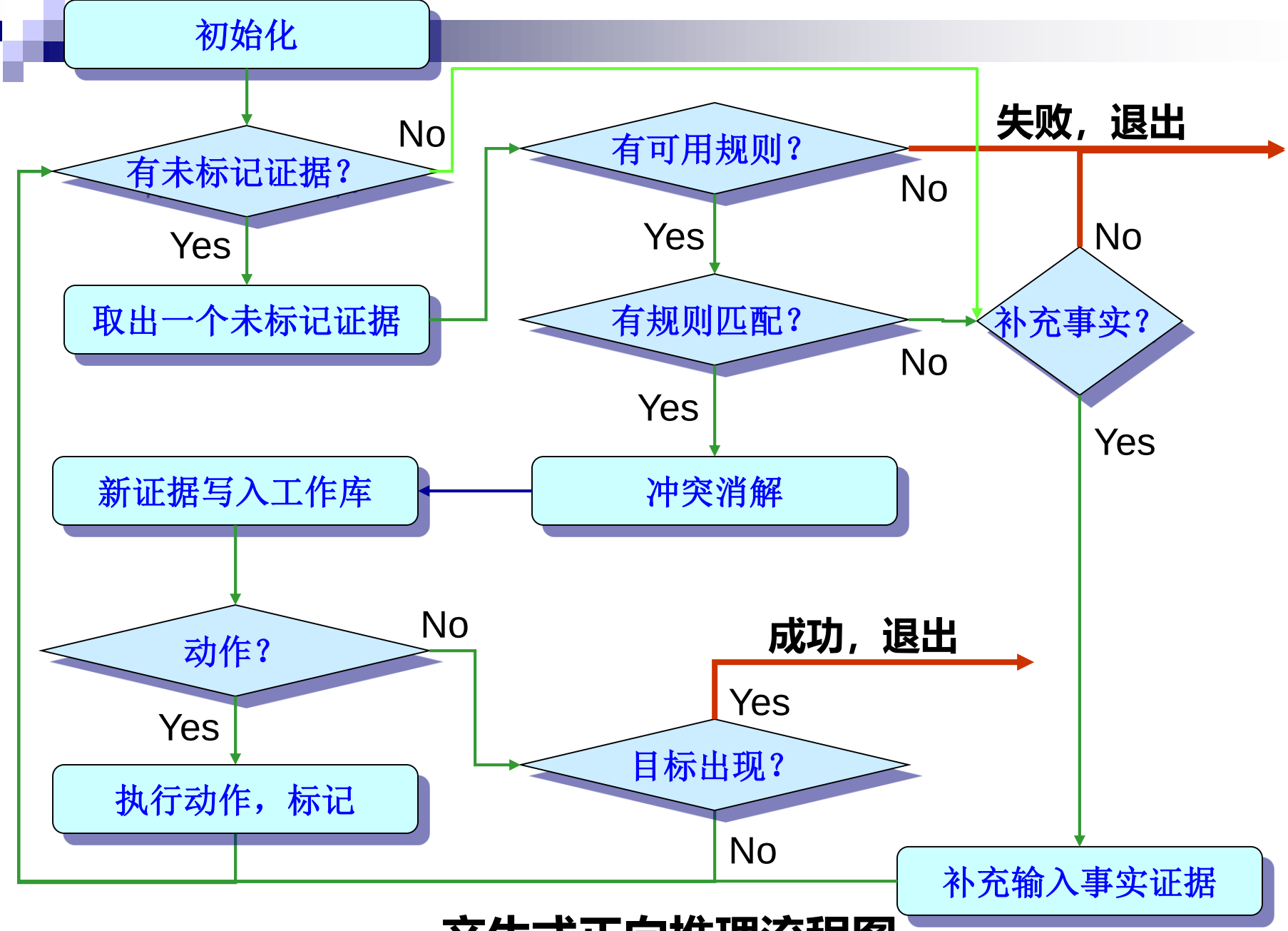
  如果该规则的结论是一些操作，则执行这些操作；

- ⑤检查综合数据库中是否包含了该问题的解（结论），若已包含，说明解已求出，成功，退出；否则，转②；

4.6.2 产生式正向推理

- ☞ **⑥ 当规则库中还有未使用规则，但均不能与综合数据库中的已有事实相匹配时，要求用户进一步提供关于该问题的已知事实，若能提供，则转②；否则，失败，退出；**

- **说明：从第③步到第⑤步的循环过程实际上就是一个搜索过程。**


初始化
有未标记证据？
No
Yes
取出一个未标记证据
有可用规则？
No
Yes
失败，退出
有规则匹配？
No
Yes
补充事实？
No
Yes
新证据写入工作库
冲突消解
动作？
No
Yes
执行动作，标记
成功，退出
目标出现？
Yes
No
补充输入事实证据

产生式正向推理流程图

4.6.2 产生式正向推理

- **例4.6.1：　事实库中有已知事实 P。**

  规则库中有如下规则：

  - **R_1　P ->P_1**
  - **R_2　P_1->P_2**
  - **R_3　P_2->P_3**
  - **R_4　P_3->Q**

  求证结论Q。

4.6.2 产生式正向推理

- **解：使用产生式正向推理。**
  - **(1)数据库中选择已知事实P，到规则库中检索‘前件’匹配的规则，找到R_1规则，且唯一，使用R1，产生（证明）P_1中间证据，P_1加入数据库；检测结论Q不在数据库中；**
    - **规则库中标记R_1已使用；数据库中P没有其它匹配规则，标记已使用。**
  - **(2)事实库中取出唯一可用证据P_1，检索规则库，找到R_2，且唯一，使用R_2，产生P_2中间证据，P_2加入数据库；结论Q不在数据库中；**
    - **标记R_2、P_1已经使用。**

4.6.2 产生式正向推理

- **(3)事实库中取出唯一可用证据P_2，检索规则库，找到R_3，且唯一，使用R_3，产生P_3中间证据，P_3加入数据库；结论Q不在数据库中；**
  - **标记R_3、P_2已经使用。**
- **(4)事实库中取出唯一可用证据P_3，检索规则库，找到R_4，且唯一，使用R_4，产生Q中间证据，Q加入数据库；Q即为结论，成功退出。**

4.6.2 产生式正向推理

工作数据库	候选规则	被触发规则
P	R_1	R_1
P, P_1	R_2	R_2
P, P_1, P_2	R_3	R_3
P, P_1, P_2, P_3	R_4	R_4
P, P_1, P_2, P_3, Q		

规则库：

R_1 P ->P_1

R_2 P_1->P_2

R_3 P_2->P_3

R_4 P_3->Q

4.6.3 产生式逆向推理

规则库：

R_1 $P \rightarrow P_1$

R_2 $P_1 \rightarrow P_2$

R_3 $P_2 \rightarrow P_3$

R_4 $P_3 \rightarrow Q$

- **实例分析：对 ‘例4.6.1’进行逆向推理。**
  - **假设目标Q成立，去寻找目标Q成立的证据。**
  - **Q在工作库中没有直接的事实证据；利用规则匹配找Q成立的间接证据；**
  - **到系统知识库中按 ‘规则后件’ 匹配检索匹配规则，如果能匹配，则此规则的前件即为Q成立的间接证据（倒用假言推理规则）；**
  - **本例Q与规则R_4的后件匹配，其前件P_3即为Q成立的证据，只要能证明P_3为真，则能证明Q为真；**
  - **接下来的工作就是证明P_3为真，所以P_3成为证明的子目标。**

4.6.3 产生式逆向推理

- **证明成立的过程与Q一样，直到：**
- **当由Q匹配产生的所有子目标都找到事实证据（匹配到事实）时，则Q成立。否则Q不成立。**

4.6.3 产生式逆向推理

工作数据库		候选规则	被触发规则
事实证据	假设目标		
P	Q	R_4	R_4
P	Q, P_3	R_3	R_3
P	Q, P_3, P_2	R_2	R_2
P	Q, P_3, P_2, P_1	R_1	R_1
P	Q, P_3, P_2, P_1, P		

规则库：

R_1 P ->P_1
R_2 P_1->P_2
R_3 P_2->P_3
R_4 P_3->Q

4.6.3 产生式逆向推理

- **也称目标（模式）驱动推理，它是从目标出发，逆向匹配规则，向着事实前进的一种推理方法。‘自顶向下’推理。**
- **逆向推理过程：**
  - **①初始化综合数据库：即把欲解决问题的已知事实输入综合数据库中，并标记；假设目标输入数据库，并标记；**
    - **此处标记是为了区分‘事实证据’、‘假设目标’。**
  - **②从工作库中取出一个尚未匹配的‘假设目标’，到工作库中检索‘事实证据’匹配，若找到事实匹配，标记匹配。否则，找不到事实证据，转④。**

4.6.3 产生式逆向推理

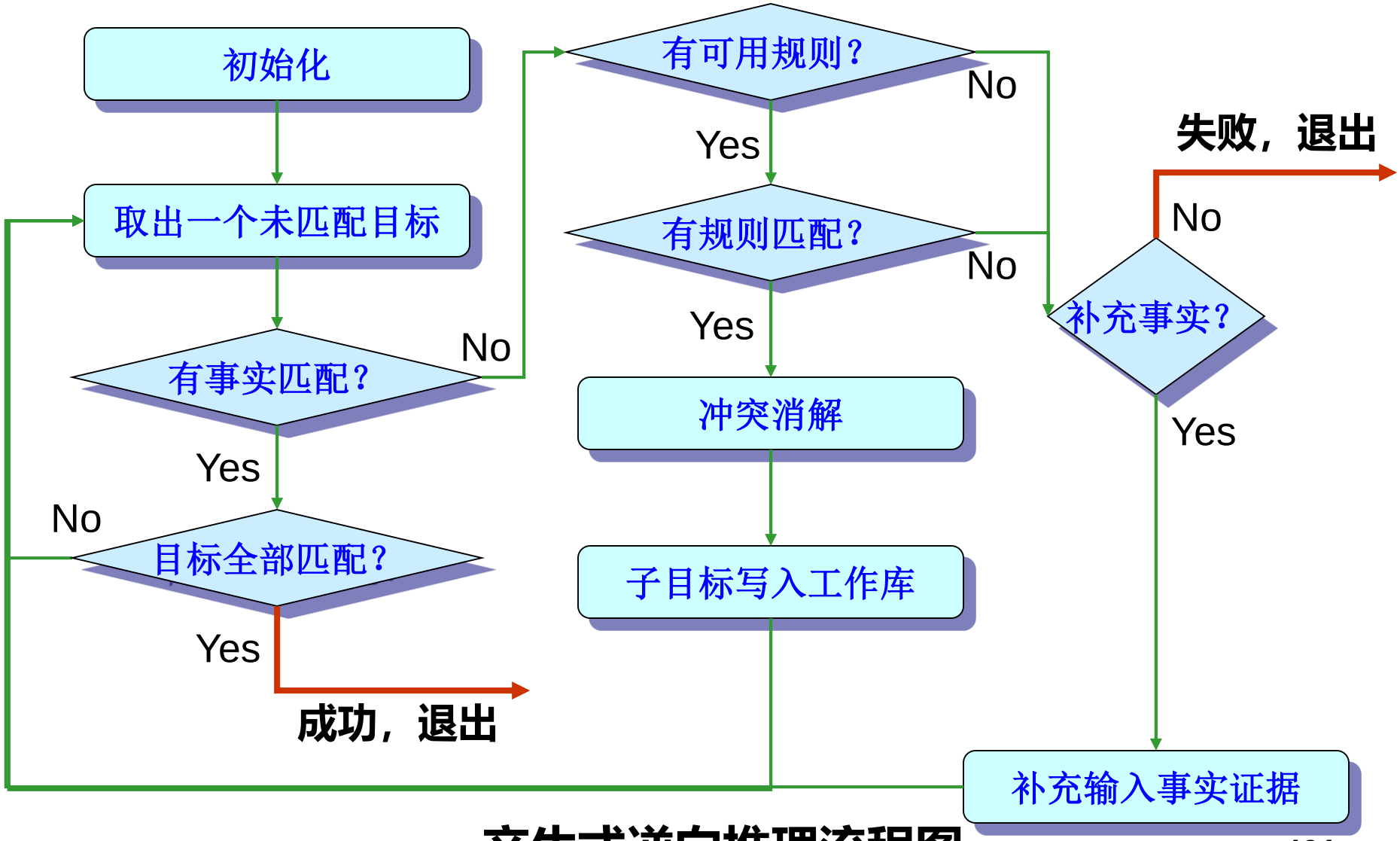
- ③ 检查综合数据库中是否所有‘假设目标’都已标记匹配，若全部匹配，则总目标目标成立，成功，退出；否则，转②；
- ④检查规则库中是否有未使用过的规则，若没有，转⑦；
- ⑤检索规则库中未使用规则中是否有其后件可与当前‘假设目标’相匹配的规则，若有，形成当前可用（候选）规则集；否则转⑦；

4.6.3 产生式逆向推理

- ⑥ 按照冲突消解策略，从当前可用（候选）规则集中选择一个规则执行，并对该规则作上标记。把此规则的**前件作为新的子目标**加入综合数据库，转②；
- ⑦当工作数据库中还有未匹配的子目标，但**既不能找到事实证据，又没有规则匹配**，要求用户进一步补充提供关于该假设目标的已知事实，若能提供，则转②；否则，失败，退出；

- **子目标匹配次序：先事实、后规则。**

4.6.3 产生式逆向推理



产生式逆向推理流程图

4.6.4 混合推理

- **混合推理是把正向推理和反向推理结合起来使用的一种推理方式**
- **它需要把问题的初始状态和目标状态合并到一起构成综合数据库**
- **1. 先正向推理，再逆向推理**
- **2. 先逆向推理，再正向推理**
- **3. 正向和逆向推理同时进行，在中间某处完全匹配—双向推理。**

- **例4.6.2：设已知事实A，B；字符转换规则**

  R_1:A∧B→C

  R_2:A∧C→D

  R_3:B∧C→G

  R_4:B∧E→F

  R_5:D→E

  求：F

- **解：本题可用正向推理或逆向推理。**
  - **(1) 正向推理**
  - **(2) 逆向推理**

工作数据库	候选规则	被触发规则
A,B	R_1	R_1
A,B,C	R_2 ,R_3	R_2
A,B,C,D	R_3 ,R_5	R_3
A,B,C,D,G	R_5	R_5
A,B,C,D,G,E	R_4	R_4
A,B,C,D,G,E,F		

规则库：

R_1:A∧B→C
R_2:A∧C→D
R_3:B∧C→G
R_4:B∧E→F
R_5:D→E

工作数据库		候选规则	被触发规则
事实证据	假设目标		
A,B	F	R_4	R_4
A,B	F, E	R_5	R_5
A,B	F,E,D	R_2	R_2
A,B	F,E,D,C	R_1	R_1
A,B	F,E,D,C		

规则库：

R_1:A∧B→C
R_2:A∧C→D
R_3:B∧C→G
R_4:B∧E→F
R_5:D→E

谢谢大家！！！