第四章 确定性推理——
规则演绎系统

规则演绎系统

- 消解反演方法的特点是简单，易于程序实现。
- 其不足是效率低，不直观，人难于理解其“证明”过程。其原因是消解反演方法将所有的谓词公式均化简为子句，致使很多隐含在原来的谓词公式中的、对推理有利的信息得不到充分的利用。
- 比如蕴涵关系P→Q，除了其逻辑含义外，还隐含了”由P推出Q”这样的信息。如果有效的利用这些信息，会使得推理进行的更加合理、自然。基于规则的演绎系统将类似于P→Q这样的蕴涵关系作为规则使用，直接用于推理。这类系统主要强调使用规则进行演绎，故称为规则演绎系统。

规则演绎系统

- 基于规则的演绎推理是一种直接的推理方法，它不像消解反演把知识转化为子句集，而是把有关问题的知识和信息划分为规则和事实两种类型。
- 规则由包含蕴含形式的表达式表示，事实由无蕴含形式的表达式表示，并画出相应的与或图，然后通过规则进行演绎推理。
- 规则演绎系统可以分为规则正向演绎推理、规则逆向演绎系统和规则双向演绎系统。

148

规则演绎系统

基于规则的问题求解系统运用下述规则来建立：

- If→Then

其中，If部分可能由几个if组成，而Then部分可能由一个或一个以上的then组成。

在这种系统中，通常称每个if部分为前项(antecedent)，称每个then部分为后项(consequent)。

规则正向演绎系统

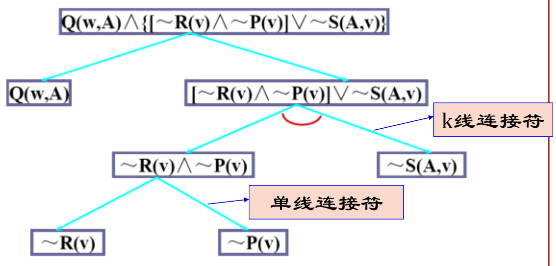
- 1. 定义
- 规则正向演绎系统是从事实到目标进行操作的，即从状况条件到动作进行推理的，也就是从if到then的方向进行推理的。
- 2. 正向推理过程
- （1） 事实表达式的与或形变换
- 把事实表示为非蕴涵形式的与或形，作为系统的总数据库。具体变换步骤与前述化为子句形类似。
- 注意：我们不把这些事实化为子句形，而是把它们表示为谓词演算公式，并把这些公式变换为非蕴涵形式的与或形。

- 与或形表达式是由符号∧和∨连接的一些文字的子表达式组成的。呈与或形的表达式并不是子句形，与子句集比起来，与或形更多地保留了公式的原始形式。

把事实表达式变换为与或形

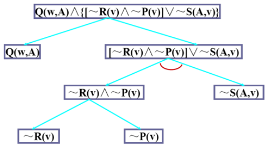
- 事实表达式
- (∃u)(∀v){Q(v，u)∧～[(R(v)∨P(v))∧S(u，v)]}
- 把它化为
- Q(v,A)∧{[～R(v)∧～P(v)]∨～S(A,v)}
- 对变量更名标准化，使得同一变量不出现在事实表达式的不同主要合取式中。更名后得表达式：
- Q(w,A)∧{[～R(v)∧～P(v)]∨～S(A,v)}
- 注意：Q(v，A)中的变量v可用新变量w代替，而合取式[～R(v)∧～P(v)]中的变量v却不可更名，因为后者也出现在析取式～S(A,v)中。

(2)事实表达式的与或图表示



(2)事实表达式的与或图表示

公式的与或图表示有个有趣的性质，即由变换该公式得到的子句集可作为此与或图的解图的集合(终止于叶节点)读出；也就是说，所得到的每个子句是作为解图的各个叶节点上文字的析取。



这样，由表达式
Q(w, A)∧{[～R(v)∧～P(v)]∨～S(A, v)}
得到的子句为
Q(w, A), ～S(A, v)∨～R(v),
～S(A, v)∨～P(v)

我们一般把事实表达式的与或图表示倒过来画，即把根节点画在最下面，而把其后继节点往上画。

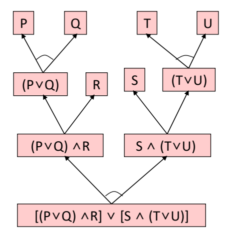
规则正向演绎系统

- (3)与或图的F规则变换
- 这些规则是建立在某个问题辖域中普通陈述性知识的蕴涵公式基础上的。我们把允许用作规则的公式类型限制为下列形式：
- L→W

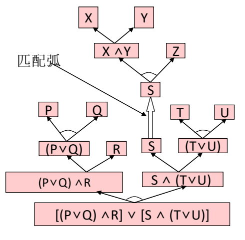
  式中：L是单文字；W为与或形的公式。

(3)与或图的F规则变换

- 公式 (∀x){[(∃y)(∀ z)P(x,y,z)]→(∀u)Q(x,u)}
  可以通过下列步骤加以变换：
- (1) 暂时消去蕴涵符号
  (∀x){～[(∃y)(∀z)P(x,y,z)]∨(∀u)Q(x,u)}
- (2) 把否定符号移进第一个析取式内，调换变量的量词
  (∀x){(∀y)(∃z)[～P(x,y,z)]∨(∀u)Q(x,u)}
- (3) 进行Skolem化
  (∀x){(∀y)[～P(x,y,f(x,y))]∨(∀u)Q(x,u)}
- (4) 把所有全称量词移至前面然后消去
  ～P(x,y,f(x,y))∨Q(x,u)
- (5) 恢复蕴涵式P(x,y,f(x,y))→Q(x,u)



不含变量的与或图

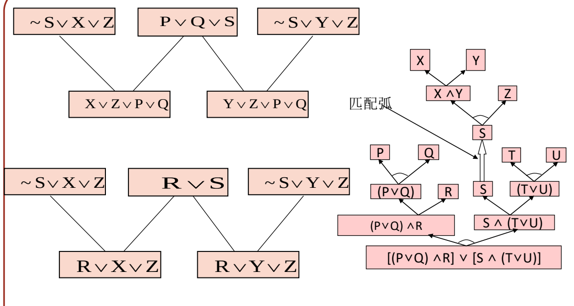


应用一条L→W规则得到的与或图

把形式为L→W的规则应用到任一个具有叶节点n并由文字L标记的与或图上，可以得到一个新的与或图。在新的图上，节点n由一个单线连接符接到后继节点(也由L标记)，它是表示为W的一个与或图结构的根节点。

作为例子，考虑把规则S→(X∧Y)∨Z应用到左图所示的与或图中标有S的叶节点上。所得到的新与或图结构表示于右图，图中标记S的两个节点由一条叫做匹配弧的弧线连接起来。

- 规则S → [(X∧Y)∨Z]的子句形是：
  ~S∨X∨Z，~S∨Y∨Z
  事实表达式[(P∨Q)∧R]∨[S∧(T∨U)]的子句形解图集为：
  P∨Q∨S，R∨S，P∨Q∨T∨U，R∨T∨U
  应用两个规则子句中任一个对上述子句形中的S进行消解：于是我们得到4个子句对S进行消解的消解式的完备集为：
  X∨Z∨P∨Q，Y∨Z∨P∨Q，R∨X∨Z，R∨Y∨Z



X∨Z∨P∨Q，Y∨Z∨P∨Q，R∨X∨Z，R∨Y∨Z

这些消解式全部包含在右图的解图所表示的子句之中。

- 从上述讨论我们可以得出结论：应用一条规则到与或图的过程，以极其经济的方式达到了用其它方法要进行多次消解才能达到的目的。

（4）作为终止条件的目标公式

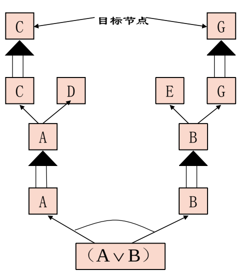
- 基于规则的正向演绎推理的基本原理是：应用F规则作用于表示事实的与或图，改变与或图的结构，从而产生新的事实，直到推出目标公式，则推理成功结束。
- 其推理过程为：
- （1）首先用与或图把已知事实表示出来。
- （2）用F规则的左部和与或图的叶节点进行匹配，并将匹配成功的F规则加入到与或图中，即利用F规则转换与或图。
- （3）重复第（2）步，直到产生一个含有以目标节点作为终止节点的解图为止。
- 若事实表达式、规则和目标表达式中有变量，则在推理中需要用最一般的合一进行变量的置换。

（4）作为终止条件的目标公式

证明过程如下：
事实：A∨B
规则：A → C∧D，B → E∧G
目标：C∨G
把规则化为子句形，得子句集：
~A∨C，~A∨D
~B∨E，~B∨G
目标的否定为：

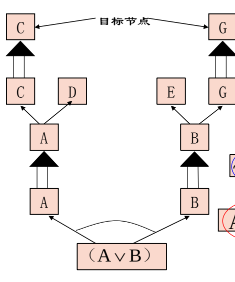
~(C∨G)
其子句形为：

~C，~G

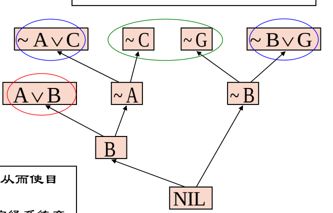


满足终止条件的与或图

正向演绎系统限制目标表达式为由文字析取组成的表达式



目标的否定为：~(C∨G)
其子句形为：~C，~G



用消解反演求证目标公式的图解

从右图我们推得一个空子句NIL，从而使目标公式(C∨G)得到证明。
我们得到的结论是：当正向演绎系统产生一个含有以目标节点作为终止的解图时，此系统就成功地终止。

规则逆向演绎系统

- ◇定义
- 规则逆向演绎系统是从then向if进行推理的，即从目标或动作向事实或状况条件进行推理的。
- ◇求解过程
- 目标表达式的与或形式
- 与或图的B规则变换
- 作为终止条件的事实节点的一致解图

规则逆向演绎系统

- 逆向演绎系统能够处理任意形式的目标表达式。
- 采用和变换事实表达式类似的过程，把目标公式化成与或形：
- （1）消去蕴涵符号
- （2）把否定符号移到每个谓词符号前面
- （3）变量标准化
- （4）引入Skolem函数消去全称量词
- （5）将公式化为前束形
- （6）删去存在量词。留在目标表达式与或形中的变量假定都已被存在量词量化。
- （7）重新命名变量，使同一变量不出现在不同的主要析取式中。

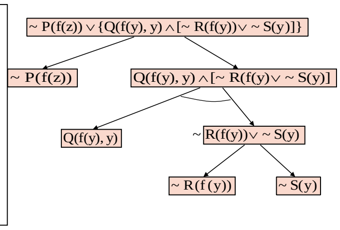
用对偶形式对目标表达式进行变换

把目标公式化成与或形

- 举例如下：
  目标表达式
  $(\exists y)(\forall x)\{P(x)\rightarrow[Q(x,y)\wedge\sim[R(x)\wedge S(y)]]\}$
  被化成与或形：
  $\sim P(f(y))\vee\{Q(f(y),y)\wedge[\sim R(f(y))\vee\sim S(y)]\}$
  式中，f(y)为一Skolem函数。
- 对目标的主要析取式中的变量分离标准化可得：
- $\sim P(f(z))\vee\{Q(f(y),y)\wedge[\sim R(f(y))\vee\sim S(y)]\}$

把目标公式化成与或形

与或形的目标公式也可以表示为与或图。不过，与事实表达式的与或图不同的是，对于目标表达式，与或图中的k线连接符用来分开合取关系的子表达式。上例所用的目标公式的与或图如右图所示。在目标公式的与或图中，我们把根节点的任一后裔叫做子目标节点，而标在这些后裔节点中的表达式叫做子目标。



这个目标公式的子句形表示中的子句集可从终止在叶节点上的解图集读出：$\sim P(f(z))$，$Q(f(y),y)\wedge\sim R(f(y))$，$Q(f(y),y)\wedge\sim S(y)$

可见目标子句是文字的合取，而这些子句的析取是目标公式的子句形（析取范式）。

与或图的B规则变换

- 现在让我们应用B规则即逆向推理规则来变换逆向演绎系统的与或图结构。
- 这个B规则是建立在确定的蕴涵式基础上的，正如正向系统的F规则一样。不过，我们现在把这些B规则限制为：W→L形式的表达式。
- 其中，W为任一与或形公式，L为文字，而且蕴涵式中任何变量的量词辖域为整个蕴涵式。其次，把B规则限制为这种形式的蕴涵式还可以简化匹配，使之不会引起重大的实际困难。
- 此外，可以把像$W\rightarrow(L1\wedge L2)$这样的蕴涵式化为两个规则$W\rightarrow L1$和$W\rightarrow L2$。

作为终止条件的事实节点的一致解图

- 逆向系统中的事实表达式均限制为文字合取形，它可以表示为一个文字集。当一个事实文字和标在该图文字节点上的文字相匹配时，就可把相应的后裔事实节点添加到该与或图中去。这个事实节点通过标有mgu的匹配弧与匹配的子目标文字节点连接起来。同一个事实文字可以多次重复使用（每次用不同变量），以便建立多重事实节点。

  逆向系统成功的终止条件是与或图包含有某个终止在事实节点上的一致解图。

举例：

- 这个例子的事实、应用规则和问题分别表示于下：

  事实：

  F1: DOG(FIDO)；狗的名字叫 Fido

  F2: ～BARKS(FIDO)；Fido是不叫的

  F3: WAGS-TAIL(FIDO)；Fido摇尾巴

  F4: MEOWS(MYRTLE)；猫咪的名字叫Myrtle

  规则：

  R1: [WAGS-TAIL(x1)∧DOG(x1)] → FRIENDLY(x1)；
- 摇尾巴的狗是温顺的狗

  R2: [FRIENDLY(x2)∧～BARKS(x2)] → ～AFRAID(y2,x2)；
- 温顺而又不叫的东西是不值得害怕的

  R3: DOG(x3) → ANIMAL(x3)；狗为动物

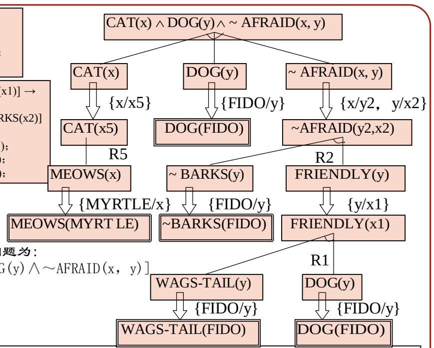
  R4: CAT(x4) → ANIMAL(x4)；猫为动物

  R5: MEOWS(x5) → CAT(x5)；猫咪是猫

  问题：是否存在这样的一只猫和一条狗，使得这只猫不怕这条狗？

F1：DOG(FIDO)；
F2：～BARKS(FIDO)；
F3：WAGS-TAIL(FIDO)；
F4：MEOWS(MYRTLE)；

R1: [WAGS-TAIL(x1)∧DOG(x1)] → FRIENDLY(x1)；
R2: [FRIENDLY(x2)∧～BARKS(x2)] → ～AFRAID(y2,x2)；
R3: DOG(x3) → ANIMAL(x3)；
R4: CAT(x4) → ANIMAL(x4)；
R5: MEOWS(x5) → CAT(x5)；



用目标表达式表示此问题为：
(∃x) (∃y) [CAT(x) ∧DOG(y) ∧～AFRAID(x, y)]

图中，用双线框表示事实节点，用规则编号R1、R2和R5等来标记所应用的规则。此解图中有八条匹配弧，每条匹配弧上都有一个置换。这些置换为{x/x5}，{MYRTLE/x}，{FIDO/y}，{x/y2, y/x2}，{FIDO/y}。由图可见，终止在事实节点前的置换为{MYRTLE/x}和{FIDO/y}。把它应用到目标表达式，我们就得到该问题的回答语句如下：
[CAT(MYRTLE) ∧DOG(FIDO) ∧～AFRAID(MYRTLE, FIDO)]

规则双向演绎系统

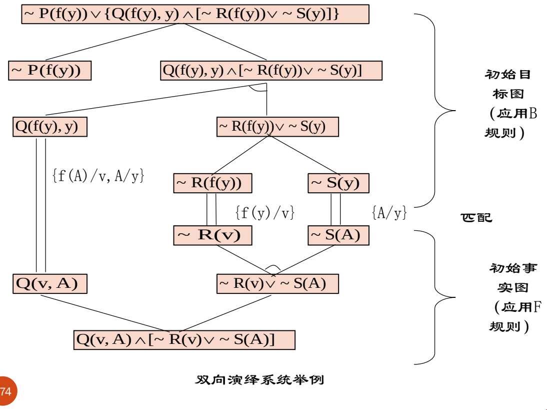
1．基于规则的正向演绎系统和逆向演绎系统的特点和局限性

正向演绎系统能够处理任意形式的if表达式，但被限制在then表达式为由文字析取组成的一些表达式。逆向演绎系统能够处理任意形式的then表达式，但被限制在if表达式为文字合取组成的一些表达式。双向(正向和逆向)组合演绎系统具有正向和逆向两系统的优点，克服各自的缺点。

2．双向(正向和逆向)组合演绎系统的构成

正向和逆向组合系统是建立在两个系统相结合的基础上的。此组合系统的总数据库由表示目标和表示事实的两个与或图结构组成，并分别用F规则和B规则来修正.

- 双向演绎系统的主要复杂之处在于其终止条件，终止涉及两个图结构之间的适当交接处。这些结构可由标有合一文字的节点上的匹配棱线来连接。



双向演绎系统举例

174

小结

规则演绎系统：

- **规则正向演绎系统**
- **规则逆向演绎系统**

- **练习**
- 目标表达式

$(\exists y)(\forall x)\{P(x) \rightarrow [Q(x,y) \wedge \sim[R(x) \wedge S(y)]]\}$

被化成与或形：

$\sim P(f(y)) \vee \{Q(f(y),y) \wedge [\sim R(f(y)) \vee \sim S(y)]\}$

作业

设某事实表达式的与或形为：

$(P \wedge Q) \vee (R \wedge S) \vee T$

请画出上式对应的与或图，并根据该与或图读出上述表达式对应的所有子句。