第14章　命题演算中的归结

14.1　一种新的推理规则：归结

14.1.1　作为合式公式的子句

在上一章中提到了几种推理规则，包括假言推理。许多这样的规则可以组合成一个规则，叫做归结（*resolution*）。本书中使用的归结应用于代表合式公式的一个特殊的形式，叫做子句（*clause*），现在来定义它。

首先，回想一下，一个文字或者是一个原子（在这种情况下，它叫做肯定文字（ *positive literal*）），或者是一个原子的否定（在这种情况下，它叫做否定文字（ *negative literal*））。一个子句是一个文字的集合。这个集合是对这个集合中的所有文字的析取式的一个缩写。因此，一个子句是一种特殊的合式公式。通常把子句写为析取式，但是当它们用集合概念来表达时，有些包含归结的定义就会比较简单。例如，子句{P，Q，¬R}（相当于P ∨ Q ∨ ¬R）是一个合式公式。空子句{}（有时候写为Nil）相当于F（它的值是假）。

14.1.2　子句上的归结

命题演算的归结规则可以陈述如下：从 $\{\lambda\} \cup \Sigma_1$ 和 $\{\neg\lambda\} \cup \Sigma_2$（其中 Σ_1 和 Σ_2 是文字的集合，而 λ 是一个原子），我们可以推出 $\Sigma_1 \cup \Sigma_2$，它被称为两个子句的归结式（*resolvent*）。原子 λ 是被归结的原子（*atom resolved upon*），这个过程叫做归结。

下面是一些例子：

- 归结R ∨ P和 ¬P ∨ Q 产生R ∨ Q。这两个被归结的子句可以写成隐含式 ¬R　P和P　Q。一条应用于这些隐含的、叫做链式（ *chaining*）的推理规则产生 ¬R　Q，它等价于归结式R ∨ Q。因此我们知道链是归结的一个特例。
- 归结R和 ¬R ∨ P 产生P。既然第二个子句与R　P等价，我们知道假言推理也是归结的一个特例。
- 在P上归结 ¬P ∨ Q ∨ W 和P ∨ Q ∨ R ∨ S 产生 Q ∨ R ∨ S ∨ W。注意只有一个Q的个例出现在归结式中——这个毕竟被定义为一个集合。
- 在Q上归结P ∨ W ∨ ¬Q ∨ R 和P ∨ Q ∨ ¬R 产生P ∨ ¬R ∨ R ∨ W。在R上归结它们产生 P ∨ Q ∨ 入 Q ∨ W。在这种情况下，既然 ¬R ∨ R和Q ∨ ¬Q有真值，那么这些归结式中的每一个的值也是真的。在这个例子中，我们必须在 Q上或者在R上归结，但不能两者同时！也就是说，P ∨ W不是这两个子句的归结式！

用 ¬λ 来归结一个肯定文字λ 产生空子句。因为λ 和 ¬λ 是互补的，从 λ 和 ¬λ 可以推出F。任何包含 λ 和 ¬λ 的合式公式的集合都是不可满足的。另一方面，一个包含一个原子和它的否定的子句（如 λ ∨ ¬λ），不管 λ 的真假值，总是为真值。

14.1.3 归结的合理性

刚才描述的归结规则是一种合理的推理规则。就是说，假如子句 $\{\lambda\}\cup\Sigma_1$ 和 $\{\neg\lambda\}\cup\Sigma_2$ 都有真值，那么它们的归结式 $\Sigma_1\cup\Sigma_2$ 也有真值。验证这一事实的一种方法是“用实例来推论”。我们知道原子 λ 或者有真值或者为假值。假如（情形1）λ 为真值，那么 ¬λ 就有假值，故要使子句 $\{\neg\lambda\}\cup\Sigma_2$ 为真，子句 Σ_2 必须为真值。假如（情形2）λ 为假值，那么要使子句 $\{\lambda\}\cup\Sigma_1$ 为真，子句 Σ_1 必须有真值。结合这两种情形，可以看到或者 Σ_1 或者 Σ_2 必须有真值；因此 $\Sigma_1\cup\Sigma_2$ 就有真值。一种相似的论证可以用真值表来进行。

14.2 转换任意的合式公式为子句的合取式

命题演算中的任何合式公式都可以被转换为一个等价的子句的合取式。一个表示为子句的合取式的合式公式叫做合取范式。(一个表示为文字合取式的析取式的合式公式叫做析取范式)。用一个例子来说明转换一个任意的合式公式为合取范式的过程的每一步，用来说明这个过程的合式公式是 ¬（P Q）∨（R P）。

1) 用 ∨ 符号，用等价的形式来消除蕴含符号：

¬（¬P ∨ Q）∨（¬R ∨ P）

2) 用德·摩根定律和用消除双 ¬符号的方法来缩小 ¬符号的辖域：

（P ∧ ¬Q）∨（¬R ∨ P）

3) 首先，用结合律和分配律把它转换为CNF。

（P ∨ ¬R ∨ P）∧（¬Q ∨ ¬R ∨ P）

然后

（P ∨ ¬R）∧（¬Q ∨ ¬R ∨ P）

一个子句的合取式（即一个合式公式的CNF形式）常常（用蕴含子句的合取式）表示为一个子句的集合；因而是

{（P ∨ ¬R），（¬Q ∨ ¬R ∨ P）}

在第3步中把合式公式（或合式公式的部分）从 DNF形式转换到CNF形式，下面的过程也许是有用的。首先，把DNF合式公式写成一个矩阵，它的行元素是每个合取项中的文字；我们有蕴含行的析取式。例如，DNF形式（P ∧Q ¬R）∨（S ∧ R ∧ ¬P）∨（Q ∧ S ∧ P）可以表示为如下的矩阵：

P Q ¬R
S R ¬P
Q S P

现在，在每一行中选一个文字并生成这些文字的析取式。在例子中，这样的一个选择可以是 P ∨ R ∨ P。作出所有可能的这种选择，每一个选择对应于一个子句，并且把所有这些子句的合取式当作原始合式公式的 CNF形式。我们可以把其中某些子句简化，例如， P ∨ R ∨ P可简化为P ∨ R，还可以把某些子句消除，例如， P ∨ ¬P ∨ Q 可以被消除，因为它是永真的。也可以消除被包容（*subsumed*）在另外一个子句中的一些子句（一个子句 γ_1 包容在子句 γ_2 中，如果 γ_1 中的文字是 γ_2 中文字的子集）。例如，P ∨ R 包容在P ∨ R ∨ Q 和P ∨ R ∨ S 中。

14.3 归结反驳

归结是一种合理的推理规则，也就是说，$\Delta \vdash_{res} \gamma$ 蕴含 $\Delta \models \gamma$，其中γ是一个子句。但是归结并不完备。例如，$P \wedge R \models P \vee R$，但是不能在子句的集合{P，R}上用归结推出$P \vee R$（因为这里没有什么可以被归结的）。所以不能直接用归结来决定所有的逻辑涵蕴。尽管如此，我们还是能用归结来说明$P \vee R$的否定是与集合{P，R}不一致的，因而，使用及证法可以验证$P \wedge R \models P \vee R$。

为验证这个过程，$P \vee R$的否定是 $\neg P \wedge \neg R$。表示为子句的（合取的）集合，我们所要验证的否定是{ ，}。为说明这些子句与$P \wedge R$的不一致性，将P和R加到这个集合中，从而得到{ ，¬R，P，R}。在后面这个集合的成员上归结产生出空子句，这是一个矛盾式，所以我们间接地从$P \wedge R$中得到$P \vee R$。

一般说来，为从一个合式公式的集合 Δ中证得一个任意的合式公式 ω，一个归结反驳（*resolution refutation*）有以下过程：

1) 把Δ中的合式公式转换成子句形式——一个（合取的）子句集合。
2) 把待验证的合式公式ω的否定转换成子句形式。
3) 把从第1步和第2步得到的子句转换成一个单一的集合Γ。
4) 反复地对Γ中的子句应用归结，并且把结果加到Γ中，直到再也没有更多的归结项可被加上去，或者产生一个空子句。

不需验证，列出以下的结论[⊖]：

- 归结反驳的完备性：假如 $\Delta \models \omega$，那么归结反驳过程将导出空子句。因而，我们说命题归结是反驳完备的（*refutation complete*）。
- 由归结反驳作命题演算的可决定性：假如Δ是一个子句的有限集合，并且，假如$\Delta \not\models \omega$，那么归结反驳过程会在未导出空子句的情况下终止。

在上一章的举积木例子中所用的推理可以用归结反驳来推导。我们有合式公式的集合 Δ：

1) BAT_OK
2) ¬MOVES
3) BAT_OK ∧ LIFTABLE MOVES

第3个合式公式的子句形式是：

4) ¬BAT_OK ∨ ¬LIFTABLE ∨ MOVES

待证的合式公式的否定产生另一个子句：

5) LIFTABLE

现在我们用归结来导出下列子句：

6) ¬BAT_OK ∨ MOVES（用4归结5得出）
7) ¬BAT_OK （用2归结6得出）
8) Nil（用1归结6得出）

我们也可以表述这个反驳为一个反驳树（*refutation tree*），如图14-1所示。

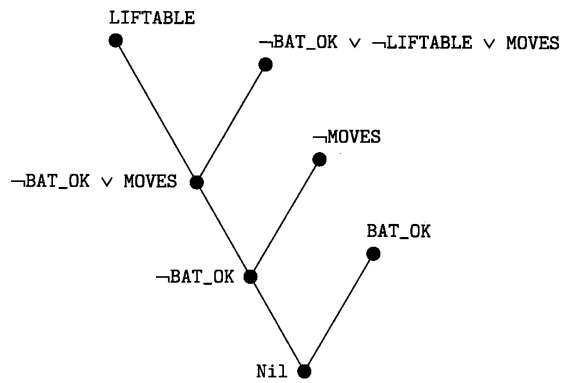

图14-1 归结反驳树

14.4 归结反驳搜索策略

虽然归结反驳过程可以很容易地被描述为“运用归结直到产生空子句为止”，但是这里有

[⊖] 有多个关于命题归结反驳完备性的证明，参见[Gensereth & Nillson 1987, P.87](命题完备性也是第16章所述一阶谓词归结反驳完备性的一个特例)。可确定性成立是由于对一个限子句集合，只有有限数目的可能归结。

一个非常重要的问题，就是应该首先运用哪个归结。同样，有些归结根本就无需运用。本节中将讨论这些问题。

14.4.1 排序策略

首先，将按什么序列执行归结？这个问题与在状态空间中下一步将扩展哪个节点的问题类似。前面已介绍了各种排序策略。例如，我们可以定义广度优先和深度优先策略。但首先作一些定义：把原始的子句（包括那些待证合式公式的否定的子句形式）叫做 0层归结项（*0-th level resolvent*）。（$i+1$）层的归结项是一个i 层归结项和一个 j 层（$j \leq i$）归结项归结的统一归结项。广度优先策略就是生成所有第1层的归结项，然后是所有第2层的归结项，依此类推。

深度优先策略将首先产生一个第 1层的归结项，然后用某些第 1层或0层的子句来归结这个子句以导出第2层的归结项，依此类推。关于深度的限制，可应用标准的回溯策略。在此书的后面，将继续讨论深度优先归结的应用。

排序归结的一个通用策略是单元优先（*unit-preference*）策略。使用这个策略，我们优先用这样的一种归结，在这种归结中至少有一个子句由一个单一的文字构成。这样的一个子句叫做单元子句（*unit clause*）。注意：图14-1中的例子并不违反单元优先策略（即在反驳树中没有一个归结是处在两个非单元子句之间）。

14.4.2 精确策略

精确策略不涉及被归结子句的排序，它们只允许某些归结发生。其中某些限制（而不是全部！）使归结反驳完备。

1. 支持集

子句γ_2 是另一个子句 γ_1的后裔（*descendant*），当且仅当：(a) γ_2 是γ_1 和另一些子句的一个归结项；或者（b）γ_2 是γ_1 的后裔与另一些子句的一个归结项。假如 γ_2是γ_1的一个后裔，那么γ_1是γ_2 的一个祖先。支持集（*set of support*）定义为由一些子句组成，这些子句或者来源于待证定理的否定，或者是这些子句的后裔。

支持集策略只允许这样一些归结：在其中正在被归结的子句中的一个在支持集中。注意图14-1中的归结反驳服从一个支持集的限制。

支持集策略是反驳完备的[Chang 和 Lee 1973，P.110]。也就是说，假如我们只对一个不可满足的子句集合运用支持集归结，那么最终会导出空子句。

2. 线性输入

线性输入（*linear-input*）策略只允许这样的归结：其中至少有一个正被归结的子句是原始子句集的一个成员（包括那些待证合式公式的否定）。图14-1中的归结反驳也服从一个支持线性输入策略。

线性输入策略不是反驳完备的，这从下列不一致的子句的集合就可以看出：

{ P ∨ Q，P ∨ ¬Q，¬P ∨ Q，¬P ∨ ¬Q }

这些子句没有模型，所以对它们来说不存在一个归结反驳。这里，把演示对这些子句来说不存在一个线性输入反驳而存在一个归结反驳留作一个练习。

3. 祖先过滤

祖先过滤（*ancestry-filtering*）策略仅允许这样一些归结：在其中至少有一个正被归结的子

句的一个或者是原始子句集的一个，或者是正被归结的别的子句的一个祖先。祖先过滤策略是反驳完备的 [Luckham 1970]。

14.5 Horn 子句

Horn子句是一种特殊类型的子句，它在人工智能和计算机科学的其他领域中都很重要。一个Horn子句就是至多只有一个肯定文字的子句。

下面是一些Horn子句的例子：

P，¬P ∨ Q，¬P ∨ ¬Q ∨ R，¬P ∨ ¬R

这种类型的子句首先是由逻辑学家Alfred Horn 研究的 [Horn 1951]。

有三种类型的Horn子句。

- 一个单一原子 —— 常被称为一个“事实”。
- 一个蕴含 —— 常被称为一个“规则” —— 它的前件由一个肯定文字的合取组成，而它的后件由一个肯定的文字组成。
- 一个否定文字的集合 —— 写成带有一个由肯定文字的合取组成的前件和一个空后件的蕴含形式。例如，当人们否定一个由肯定文字的合取组成的待证的合式公式时，这种形式即可获得。所以这类子句常称为一个“目标”。

这三类Horn 子句的例子分别是P，P ∧ Q R，P ∧ Q 。

一个有关命题的Horn子句的重要结果是，存在着线性时间的演绎算法 [Dowling 和 Gallier 1984]。直观地说，用非Horn子句作NP难题推理的原因是，在试图验证一个肯定文字的析取式如P ∨ Q 时，我们必须考虑多种情况 —— 验证P或者Q。在Horn子句中，没有肯定文字的析取式。

为验证源于Horn 子句的规则和事实的目标证明系统，通常以下面的方式提供排序信息给系统：用一定的序列写出事实和规则；用一定的序列写出在每个规则和目标的前件中的文字；然后在基于这些序列的深度优先的样式中搜索一个验证。我们将在讨论了谓词演算之后更详细地研究这个过程。

习题

14.1 验证从DNF转换到CNF的矩阵程序保留了等价性。

14.2 头，我赢；尾，你输。在命题演算中表达这些陈述（加上别的你可能需要的陈述），然后用归结验证我赢（换一种方法，你可以改变这个问题，假如你喜欢：头，你赢；尾，我输）。

14.3 下面的合式公式有时被用在命题演算中作为原子的实例：

1) 蕴含引入：P (Q P)

2) 蕴含分配：(P (Q R)) ((P Q) (P R))

3) 矛盾体现：(Q ¬P) ((Q P) ¬Q)

用归结反驳来验证这些公式中的每一个。

14.4 考虑下列不可满足的子句集合：

P ∨ Q

P ∨ ¬Q

¬P ∨ Q

¬P ∨ ¬Q

1) **为下列的每一种策略导出归结反驳：**

(a) **支持集归结（其中支持集是在上述的子句列表中的最后一个子句）。**

(b) **祖先过滤形式归结。**

(c) **一种既违反支持集也违反祖先过滤的策略。**

2) **验证不存在这种不可满足的子句集合的线性输入归结反驳。**

14.5 **转换下列命题演算合式公式为子句：**

¬[（(P ∨ ¬Q） R） （P ∧ R）]