知识库的一致性问题和检查方法

应 晶 吴朝晖 何志均

（浙江大学人工智能研究所）

提 要

There is urgent need for sophisticated tools to help experts deal with difficult and indispensable task of knowledge acquisition. The problem of detecting inconsistencies in built knowledge base is especially crucial. As the knowledge base enlarges, this problem should commonly be solved by classical logic or other techniques. This paper analyses the definition and the notion of inconsistency, also presents several solving methods for such issue in production system (including classical logic, Petri Net and Truth Maintenance techniques) and their individual characteristics.

知识库中存在的不一致性是影响整个系统性能的重要因素之一。在系统建立初期，由于知识库比较小，内容也比较简单，只要设计人员或知识工程师对初始知识模型进行反复推敲和精心构造，这类问题还是比较容易防止的。但是随着时间的推移，人们对知识库在最初设计阶段的种种考虑逐渐淡漠，同时由于知识库中新的知识不断加入，知识库越来越大，内容也越来越丰富复杂，这时知识库中各知识单元之间的相互影响和相互联系就随之变得复杂，难以跟踪和捉摸了。在这种情况下，知识库中的不一致性检测、识别和纠正显得更为重要。

1、 知识库的一致性问题

知识库系统的组织按其结构可以划分为事实性知识（领域知识）和控制性知识，它们分别组成了事实库FB和规则库RB。

（1）规则库表达了解决问题的方法、专家的经验，通常以规则形式存在。

（2）事实库是推理数据库。在整个推理过程中初始事实和规则不断进行深层演绎得出新的事实。

事实库的知识通常以值的改变和描述事实的增删这两种方式进行维护。对于值的改变，不会产生不一致性问题。唯一可能的是描述了两个相反的事实，这可以直接推出矛盾，即 f_1, $f_2 \xrightarrow{①}$ nogood。因此事实库的一致性维护比较简单，有些可采用表面检查。然而，对于规则库，如果有下列规则：

R1: if a then b and c and e

R2: if b or d then f

R3: if f and e then g

R4: if g then not a

R5: if s and t then not s

那么对规则本身而言，R5显然是矛盾的，形成整个知识库的不一致性。另外，R1,R2,R3,R4通过三次演绎也最终导出矛盾：if a then* not a，即 R1,R2,R3,R4 $\xrightarrow{③}$ nogood。计算机要解决这种深层推导所产生的不一致性，需有合适的方法，如用数理逻辑子句表示形式对规则进行判断。这些就是我们所要讨论的知识库不一致性问题。

下面对一致性问题作些定义。

定义1 一个知识库的RB可能不一致，

如果存在一些一致的初始事实库FB有：RB，FB⊢⊗。其中逻辑结果⊗所产生的前提是所有的RB和FB内容。

例

$A \rightarrow C_1$

$B \rightarrow C_2$

$C_1, C_2 \rightarrow \otimes$

就是一个可能不一致的规则库，按定义，因为FB{A,B}产生不一致性。但是如果事实库永远不可能同时拥有{A,B}，那么规则库就不是真正不一致了。即A 和 B 事实同时存在是无意义的。

定义2 （更强的）一个知识库的RB是真正不一致的，如果存在有意义的初始一致事实库FB有：RB，FB⊢⊗。或者，一个RB是一致的，如果它不是真正的不一致，即对所有有意义的FB，

$$RB, FB \not\vdash \otimes$$

Rousset 在他的COVADIS系统[4]里利用这几种概念，引进整体性约束(Integrity Constraint，如$c_1, c_2 \rightarrow \otimes$)对所有规则进行扩展、传播，组合成一个新的规则体，利用IC消去矛盾部分。对剩下的规则体进行"意义"检查，以分析所包含的是否为有意义的事实库，从而判断不一致性。

2、 经典逻辑的解决方法

大部分专家系统都实现成基于规则的产生式系统，而且包括各类原语操作：比较，选择，分类，模式匹配，逻辑集合操作和传统代数逻辑运算。其优点主要是有固定的形式。规则被模式所激活，这依赖于正向或逆向推理策略，决定其前驱或后继部分的匹配。但是规则间的关系不明显，许多建立在知识表达层次上的AI应用都缺少一致性和完备性的知识。所以有必要在新的知识加入时不断修改知识库和维护它的一致性。知识库调试的关键一点在于检查RB的一致性(正确性)和结论的完备性。

在这方面，Hayes-Roth指出，基于规则的系统主要缺少一种合适的验证方法或技术，用来测试和检查规则集的一致性和完备性。下面我们具体分析知识库规则的不一致性及解决的方法。

对于一个规则集，如果对规则进行静态逻辑语义分析，就可以发现一个规则库中存在如下一些可能的问题和缺陷：

①冗余规则：两个规则的前提条件等价，一个或多个结果子句也等价。

②冲突规则：两个规则前提条件等价但一个或多个结果子句有矛盾，或者前提子句有矛盾而结论部分完全等价；也有可能由多条规则链形成冲突规则集。

③从属规则：两个规则有相同的结果，但其中一个包含有多余的约束条件(and或or子句)。

④循环规则：由数个规则的前提和结论形成一个循环链，最终由末尾规则的结果子句推出起始规则的前提部分。

知识库系统的发展是一个知识编码、测试、增加、改变和求精的迭代过程。这个过程经常在知识库中产生一些后患，而知识工程师和专家在知识获取阶段有所忽视。这主要包括：

①遗漏规则：在一个对象的属性的可能值(合法值)集合中有一些值没有反映或部分反映规则的前提部分。当运行时遇到没有反映的属性值时，系统无法得到解答或产生错误的结论。

②不可达子句：一条规则中的前提部分永远不可能满足。

③闭塞子句：如果一条规则的结果子句既不满足目标子句，又不能匹配其它规则的前提条件，则这条规则是死路规则(即闭塞子句)；同理，如果一条规则的前提部分不能被其它规则的结果子句或初始事实所匹配，则也产生闭塞子句。

为了解决上述不一致性问题，直观的解决方法是搜索比较与匹配。一般说来，在解决一个问题时，各种规则集分别刻画了问题求解各个阶段的情况。因此每条规则并不是与所有其它规则有关，如果把规则比较算法建

立在一个规则与其补集上，是效率不高的。因此通过预扫描把规则库分成若干集合，然后对同一集合内的相关规则进行比较，这样比较次数明显少于N条规则总体比较的(N-1)N/2次，其计算量主要依赖于规则的相关程度。

对于循环规则，我们构造规则集的if-then图，从起始规则的条件部分开始，通过if-then图搜索，如果搜索过程中最后遇到的then部分已在前面出现，则形成循环规则，否则就终止。

对于冲突规则，则需构造if-if表。对规则集内有相同if的规则子句构造规则树，形成推理图。同时建立then-then表用以判断是否有冲突规则出现。对相同if部分的规则继续用它的各自then部分作为其它可以匹配的if前提条件，递归地构造，如发现两个推理图上分别有结点在then-then表上是矛盾的，则检测出冲突规则，如图1所示：

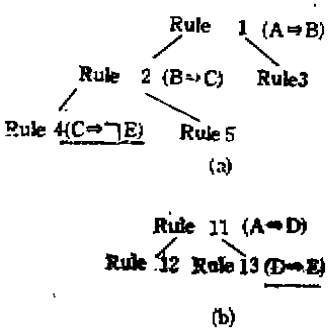

图1 规则树和推理图的构造

图中(a)部分产生(C⇒¬E)结点，其then部分是¬E，与(b)中产生的(D⇒E)结点矛盾，因此形成冲突规则。由于推理图中父结点可以到达任一子结点，所以可以检查(a)和(b)任意子结点以发现矛盾冲突所在。

对于冗余规则和从属规则的检查类似于冲突规则链的方法。不同之处是前者在推理图中的遍历是试图发现有then部分等价的两条规则。

上述这种通用技术是逻辑、集合论和归一化原理的结合，通过这种方法检测知识库的一致性，结构性强，能较完整地检查可能存在的不一致规则。规则树与推理图的建立是核心工作，特别是当规则中的原子公式组合较复杂时，规则树的搜索等方面的效率却会降低。

3、 基于 Petri网的检查方法

用以解决规则库中各类问题的模型不多，而许多已有的模型仅仅局限于检查规则中的变量值。Quinlan[2]应用分类推理帮助维护数据一致性，Lim扩展推理网，对产生式系统构造P-图，Tsang开发一个空间搜索技术检查规则库中的非类似性。

产生式规则可以包含除了领域可以精确定义数字信息之外的非数字信息。已经存在的模型难以透视规则集的推理行为传播，另外规则间不明确的相互关系和错误假设的影响也可能难以取消而保留下来。因此系统分析模型是必要的，尤其用以证明在运行时刻可能传播和验证的不一致性。Petri网模型可以用来分析和解决规则库的一致性问题。

Petri网中的结点对应于规则的前提或结论部分的一个谓词(或原子公式)，一个迁移对应一条规则对其谓词值的定义和规则名的说明。当产生点火时，一条规则被激活，同时传导到后继的各个结点和规则。这里我们定义：

Si：一个位置代表一条规则的谓词Pi；表示成○；

ti，一个迁移代表规则的激活；┃

e，一个用以控制点火进程的标志；◉

y：状态标志，标明由Si代表的谓词Pi的真值；表示成Ⓨ；

n：状态标志，标明由Si代表的谓词Pi的假值；表示成Ⓝ。

我们给出下述的一致性，完备性定义：

定义3 一个规则库是一致的，如果对于每个可能的初始标记都不能从规则库中推出矛盾结果。

定义4 一个规则库是完备的，如果对于每个可能的初始标记，一个可能的结果都

可以从规则库中推导出来。

通过Petri网构造规则库的模型能完成一致性、完备性的检查，也包括冗余规则、冲突规则、从属规则、循环规则、死路规则等的检查。试考虑下面一个产生式系统的规则库：

R1, if ¬P1 then P2
R2, if P11 then P10
R3, if ¬P4 then P8
P4, if P2 then P3
P5, if ¬P11 then P10
R6, if P3 then P4
R7, if P8 then ¬P10
R8, if ¬P8 then P9
R9, if ¬P1 and P5 then P6
R10, if P5 then P6
R11, if ¬P9 then P11
R12, if ¬P7 then ¬P1
R13, if P9 then ¬P8
R14, if P4 then P7

用Petri网表示如图2：

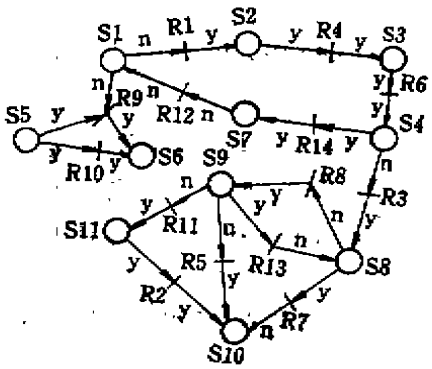

图2 规则库所建的Petri网模型

• 冲突规则(rule14)

由图3可以看出：如果S7加上ⓔ和ⓝ标志后点火激活，则通过S1，S2，S3，S4连续点火成功，最后回到S7而附上ⓔ和ⓨ的标志。因此rule14是冲突规则。

• 循环规则(rule8、rule13)

图4中由于S8和S9两种状态一直在循环点火，最终得到的标志一直是S8：ⓔⓝS9：ⓔⓨ，所以形成循环规则。

• 死路规则(rule7等)

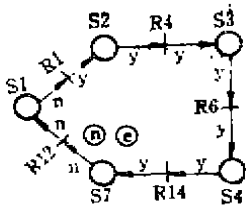

图3

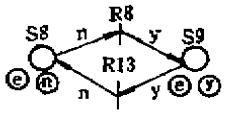

图4

图2中因为rule7的点火成功后到达S10，没有任何一条规则会由此而激活。象S10这类结点我们称为终结点。如果规则结论包含这类结点，要么与目标结点匹配成功，否则即为死路规则。

• 冗余规则(rule5)

图5中由S9点火后通过rule5或rule11、rule2到达S10ⓔⓨ具有等价效应。这里由于rule11还使S11激发点火，所以r5被视为冗余规则。

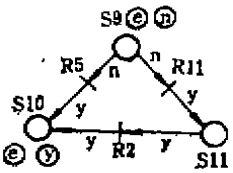

图5

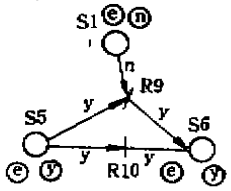

图6

• 从属规则 (rule9)

图6中由S5ⓔⓨ点火通过rule10即到达S6ⓔⓨ，使得S1ⓔⓝ的加入显得多余。删去rule9从属规则。

对于通过Petri网构造的产生式系统规则库模型来说，由于Petri网本身表达方面的优点，可以表示不同的对象类，加之出于它点火装置的自动时序，可以清晰地检查规则库的不一致性。这只要任意指定一个结点置以标志点火，则从此规则所激发的一系列结果都能分析检测。在这方面我们实现了基于Petri网的知识库一致性检查系统，采用上面描述的算法，对规则库的检查效果较好。

4、TMS技术的应用

真值维护系统TMS，又称理由维护系统RMS，主要通过与问题求解器的相互作用来

完成整个推理搜索求解过程和知识库一致性的维护（包括矛盾的检测处理）。这种一致性检查维护是动态的。问题求解器控制推理过程和空间搜索策略，把每次任务产生、推理过程描述成命题形式，以理由这种内部描述形式传给TMS。它记录事实之间的依赖关系、事实与规则之间的关系及控制性知识与知识库状态的关系。一旦在推理过程中出现不一致性，产生矛盾，则根据各种依赖关系，进行从属制导回溯，消除矛盾。TMS和问题求解器结合在一起完成了基于知识库的整个推理过程，同时保证了知识库的一致性。

产生式系统是一种较为成熟的专家系统技术，但象OPS5这样的正向推理系统在处理知识的一致性方面能力却很弱，同时也缺乏搜索功能。TMS的思想能使问题求解器推理过程保持一致，用从属制导回溯（DDB）弥补产生式系统的缺陷。作为一种非单调推理机制，DDB和状态空间信息转移较以往的测试生成法有较高的效率。

〔6〕中讨论了RMS的实现方法，它采用了数据关系依赖网作为问题求解器的扩充（包括经典的八皇后问题）。这里我们提出一种新的一致性检查方法论。利用RMS中的数据依赖关系和从属制导回溯，通过由规则库中每条规则聚集形成的关系依赖网的满足性维护来保证规则库的一致性。其主要步骤如下（DD子句和DD网在〔6〕中有具体描述）：

①对所有规则进行规范化，形成基本命题和数据依赖子句（DD子句）；

②在数据依赖网图（DD图）中加入①中基本命题；

③对每个DD子句，加入DD网图，根据标记传播算法，判断是否有置换使得网图满足。若不满足则转⑤处理不一致性。

④当③全部结束后通过当前网图判断规则库情况；

⑤撤消当前DD子句，放弃有关置换值，递归更新网图满足情况。

这种方法采用真值指派方法，逻辑上比较严密，可靠性强。但由于关系依赖网扩大到一定程度后结构复杂，递归回溯技术影响了效率。

我们已实现的知识库调试工具Zdbxtool体现了本文的一部分思想。另外基于Petri网的一致性检查算法对于几个小型规则库都能完整地检测出各种不一致规则。

总之，由于知识库构造的复杂性和庞大的容量，其一致性问题是保证求解正确性的关键。本文针对产生式系统的规则库的一致性问题作了分析并给出一些相应的解决方法。这些技术和方法已日益完善和成熟。当然由于难以达到知识获取的严密性和完整性，有些问题甚至在目前看来是不可解决的，这方面有许多问题值得探讨和研究。

参考文献

〔1〕 T.A. Nguyen, etal., "Checking an expert systems knowledge base for consistency and completeness", Proc. IJCAI'85, pp.375-379

〔2〕 Doyle. J, "A truth maintenance system", Artificial Intelligence, 12(1979), 231-272

〔3〕 Quinlan, J.R, "Internal consistency in plausible reasoning systems", New Generation Computing, Vol.3(1985) pp. 157-180

〔4〕 Marie-Christine Rousset, "On the consistency of knowledge bases: the COUADIS system", Computer Intelligence, 4. 166-170(1988)

〔5〕 N.K. Liu, T. Dillon, "Detection of consistency and completeness in expert systems using Numerical Petri Nets", Artificial Intelligence Development and Application, 1988

〔6〕 Charniak, E., Riesbeck. C. and McDermott, D., "Artificial Intelligence Programming", Lawrence Erlbaum, Hillsdale New Jersey, 1979.