一种基于面向对象环境的因果推理方法*)

金 芝 李德意 胡守仁

（长沙工学院计算机系）

摘 要

It is clear that expert system's technology is one of AI's greatest successes so far. Currently we see an ever increasing application of expert systems. Yet there are also a number of well-recognized problems associated with the new technology. To alleviate the limitations of the current generation expert systems, the key is to develop the deep knowledge of the problem domain. This paper proposes an approach to the object-oriented causal reasoning about the implication of causal event stream, in arbitrarily complex causal network. The aim of our work is to develop an environment for capturing the deep knowledge in a special application domain and to support deep reasoning. This system has been prototyped in an object-oriented extension to Prolog. The impact that these two paradigms, logic and objects, have had on the design is discussed. Several example applications have also been presented to illustrate the extensibility of the environment.

一、引 言

当今专家系统已逐步成为人工智能中影响最大、应用最广泛的领域之一。然而，尽管专家系统技术在具有良结构的狭窄领域内

完成了"加速"学习，但交互之后，学习系统就无需用户的协助，便可解决类似的问题了。

3. 进一步的工作

在本文介绍的构架中，有待于进一步探索动态操作性的引入，亦即，目标的操作性允许定义和再定义，以期得到更加一般的推广。同时，正如文中第三部分所提及的那样，新模态算子的引入，还值得进一步研究。

在本文完成过程中，作者自始至终得到徐家福先生的指导和鼓励。他仔细审阅并修改了全文，特此致谢！

主要参考文献

〔1〕徐家福，软件笔谈，计算机科学，1990年第3期

〔2〕费宗铭，吕建，徐家福等，机器学习，计算机科学，1991年第1期

〔3〕Scott Dietzen, Frank Pfenning, High-order and Modal Logic as A Framework for Explanation-based Generalization, Tech. Rep. CMU-CS-89-160, Oct. 16, 1989

〔4〕Gopalan Nadathur, Dale Miuer, An Overview of λProlog. Proc. of 15th Inter. Conf. & Symp. Vol.1, 1989

〔6〕王元元，计算机科学中的逻辑学，科学出版社，1989年

〔6〕H. Hirsh, Explanation-based generalization in a logic programming environment, IJCAI-87, pp. 221-227

*) 国家高技术发展计划资助课题

表现出很高的性能，但仍存在许多问题。如：

• 处理困难或不常见问题时性能急剧下降；

• 由于系统知识库的非结构化，系统知识难以维护和修改；

• 系统不能积累以往问题求解的经验；

• 系统行为的解释仅仅是启发式推理规则的再现，缺乏理论支持，不能令人信服。

之所以存在上述问题，主要是由于缺乏问题领域的深层知识。近几年来，人们已经开始在问题领域因果结构和模型推理的基础上研究深层知识。正如L. Steels[3]所说，基于规则的专家系统只是把问题领域看作为一组描述输入/输出的产生式规则，并将问题领域的内部机制当作一个黑箱；而因果模型则恰恰相反，它把问题领域的完整结构（包括系统部件及其部件间的联系）显式表示出来，对系统各基本结构部件进行推理，从而预测整个系统的行为。因果模型基于这样一个假设，即系统行为可以从系统的结构中推导出来。这里，“结构”是指系统部件、部件属性及部件间的连接关系，“行为”是指可观察到的部件行为和整个系统的状态变化。

当然，因果推理要对一个庞大的结构空间进行盲目的搜索，为了减少这种盲目性，有必要把启发式规则和基于问题领域模型的深层推理结合起来，让专家系统在常规情况下采用基于启发式规则的推理，而当所有启发式规则都不可用时，再求助于基于模型的推理。这种问题求解策略是为了使专家系统在常规情况下获得最大效率，而在遇到困难情况时又不致于彻底失败。

但是，当前在专家系统中引入因果模型的一些尝试，大多将系统部件的单属性值作为独立的因果条件，如L. Steels 在文[3]中所述汽车发动机的例子。从某种角度来看，这种因果网络实际上是一种广义的语义网。由于这种因果网络的结点粒度很小，具有多个属性值的部件要用多个结点描述，各部件自身的行为和状态变化难以模拟，不能反映从系统部件行为推导出系统行为的原理。

本文提出了一种基于面向对象 Prolog环境的因果推理方法，它用对象描述问题领域的结构和功能部件，用方法模拟部件的行为，并用消息来传播部件之间的相互影响。由于一个对象比单属性因果条件有更强的表达能力，这种因果推理模型可以方便地描述问题领域的深层结构知识并对其进行推理。下面我们将分别介绍这种基于面向对象Prolog环境的因果推理模型，它的实现以及应用实例。

二、因果推理模型

因果模型的提出已有很长的历史[7]，但是在人工智能系统中，特别是在专家系统中将它作为一种领域知识表示形式，还是近几年的事。作为专家系统中一种深层知识描述方法（对应于浅层启发式规则），因果模型出自于这样一种假设，即各系统部件（或事件）的行为由它们在领域完整结构的时序因果网络中所处的位置决定。因果模型对问题领域的物理或功能结构进行编码，分别描述各个独立部件的属性和行为，并显式地表示领域部件之间的因果关系。因果模型与传统启发式知识表示的区别是明显的，前者包含了问题领域可见和不可见的全部属性；而后者仅包含了问题领域的外界可见属性。例如，图1(a)所示的门电路是一个无任何结构知识的黑箱，只有启发式规则“Output==true”描述了该门电路的输入输出特性，系统推理按输入信号和这些启发式规则来进行。与之相反，图1(b)表示了与图1(a)相同的门电路，但其内部结构已被显式地描述出来，系统知识包括各种逻辑门的属性，如反相器属性为 Output==～(Input)，与非门属性为 Output==～ (Input1∧Input2)，和其间的连接关系。

因此，给定如图1(b)所示的框架，如果已知输入端信号，则可根据反相器和与非门

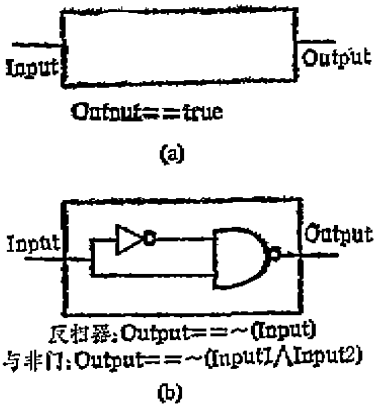


图1 门电路的两种表示方法

的属性及其内部连接关系推导出输出端信号的变化。必须注意的是，图1(b)中的推理涉及到该门电路各个阶段的状态变化情况，能更精确细致地刻画系统的行为，增强了系统的强壮性，也增强了解释该门电路各种变化的能力。

2.1 因果知识的表示

形式化问题领域因果模型的一种最常用的方法是传统有向图形式[8]，其中，结点表示因果模型中引起系统行为变化的属性变量，结点间的有向弧表示一个属性变量对另一个属性变量的直接因果影响，属性变量间的间接因果影响则由图中的有向路径(包含≥1条有向弧且有向弧的方向一致)确定。

这种因果知识表示方法能很好地刻画仅含单属性事件(或部件)的问题领域的深层因果模型，并能有效地实现因果推理。但对含多种属性的事件(或部件)，必须将其属性分解到多个节点上作为全局属性加以表示和处理，这样，不仅增加了因果推理的复杂度，而且不能体现深层因果推理中单一事件(或部件)行为的局部性和独立性[8]。

为此，我们从下述两个方面对传统的用以表示因果模型的有向图方法进行扩充：(a)增大有向图中结点的粒度，以组合问题领域中复杂事件(或部件)的全部相关功能和属性，用单一结点表示独立事件，从而在知识表示一级支持该事件行为的局部性和独立性；(b)对每个结点增加一个元级机制，以定义该节点所表示事件(或部件)的特殊问题求解方法。当因果模型中的独立事件本身又是一个复杂不可分的物理过程(如：约束求解过程或其它数学过程)时，增加的这种元级机制为系统提供了更强的知识表示能力。

例如，上述门电路因果模型用单属性有向图可表示为如图2所示因果网；而用扩展因果图则表示为如图3所示。显然，图3所示方法更为简洁、自然，系统推理搜索空间小。另外，若将结点中的属性值描述改为部件功能属性描述，则可模拟系统的行为。因此，它为具有显式结构的问题领域，提供了一种统一用于预测的行为模型和用于诊断的

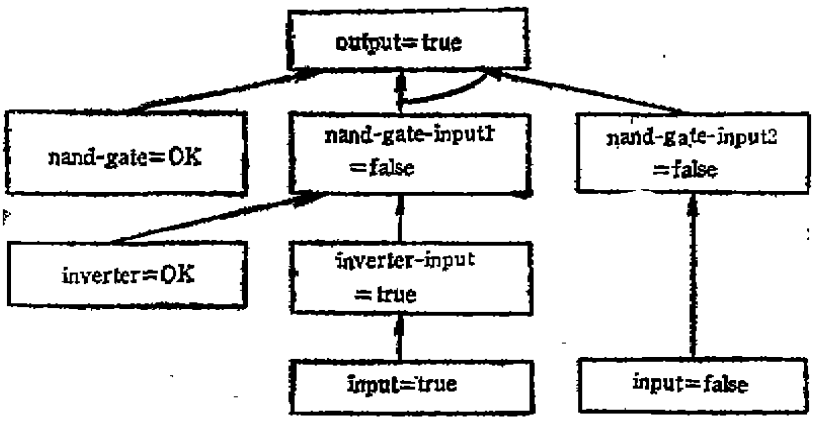


图2 单属性因果网

因果模型的框架。

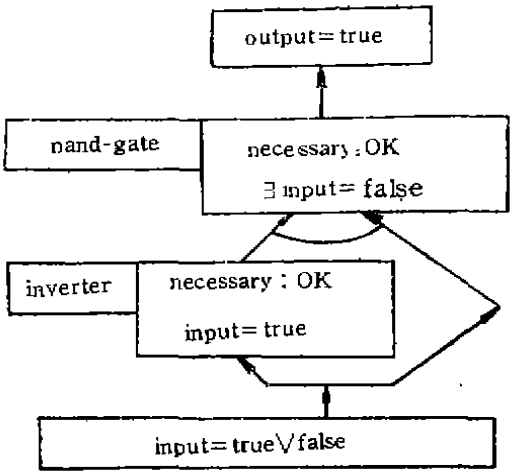


图3 组合多属性的独立部件因果图

2.2 基于面向对象环境的因果模型

基于面向对象环境的因果模型是在多种新颖思想的基础上提出的，如基于对象的模拟、说明性程序设计等，它是一种灵活的因果推理环境。这种因果模型可看作为这样一个框架，它将领域独立部件表示成对象，部件间的相互影响则通过对象发送消息来传播，并将领域部件对象库、领域部件间的拓扑结构、推理算法和启发式知识库组合在一个对象环境中。其中，部件对象库包含领域所有独立部件，每个对象定义一个独立部件的属性值、特性或行为。如在门电路领域中，领域部件对象包括非门、或门、或非门、与门和与非门等。模型实际部件则作为现有类对象的一个实例。领域拓扑结构包括两类对象，即描述领域有向因果图结点的部件集合和描述各结点间有向关系的路径集合。启发式知识库是关于部件间影响产生的约束条件库，主要是为了提高因果推理的速度。推理算法也定义为对象，主要用于控制有向因果图的搜索。另外，因果推理算法还包括对各部件对象属性值的检测，而行为预测算法则包括对各部件对象特性或行为的推断。将推理算法作为对象看待，可在同一个框架中融合多种深层推理模型，体现环境功能的多样性。该因果模型的概念框架如图4所示。

这个因果模型框架可供领域专家建立特定领域的因果推理模型，其建立任务包括：(a)将特定领域模型划分为相对独立的部件流图；(b)定义该领域中各独立部件，包括部件属性值、特性或行为等；(c)定义领域各独立部件间的拓扑关系；(d)系统对每个对象的省缺推理方法为逻辑推理，对于领域中存在特定行为预测算法的独立事件，领域专家需要另外定义该独立事件的元级对象，以控制它的行为推理。

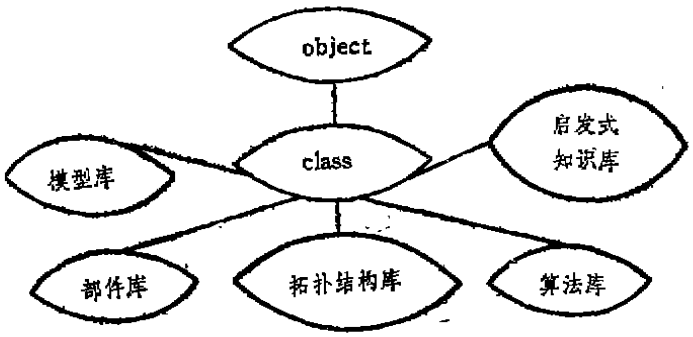


图4 因果模型的概念框架

三、结构化可通信知识描述语言

为支持上述因果模型的模拟和开发，我们实现了一种合成逻辑和对象的集成式语言——结构化可通信知识描述语言SCKL(Structural Communication Knowledge Language)。

SCKL语言模型是标准Prolog的一个超集，它基于标准Prolog，并在下述四个方面引入面向对象语言范例中的典型概念。

(a)模块化 在SCKL语言模型中，程序由对象组成，每个对象都包含一组独立的相关Prolog子句。类是同一类对象建立的模板，每一类的所有对象按数据抽象原则组织成层次结构，利用类的继承性，低层类可继承与之相连的高层类中定义的谓词。

(b)消息 独立的Prolog程序模块(类)之间可通过发送消息进行通信。发送一条消

息给某一类可视为请求建立该类的一个新实例，并用该实例的私有状态来证明一个目标。消息的语法形式为：Object::Goal。

中缀谓词::/2可以任意嵌入程序中。从形式上来说，消息的语义可定义为："在对象Object定义的环境中证明目标Goal"。

在试图证明目标"Object::Goal"时，SCKL首先找到程序模块Object，再利用继承机制寻找可证明目标Goal的方法。

(c)实例创建 实例对象是通过向类发送消息new/2创建的，实例创建消息形为：

class::new(Instance-name, Initial-property)

其中，Class是一个已存在的类对象，Instance-name是新实例对象的对象标识符，如果对象标识符在消息发送时还未例示，则系统内部产生一个唯一标识符。Initial-property为该实例的初始状态。

(d)继承性 继承有两种，即类/实例继承和超类/类继承，前者是通过实例对象所属类的对象标识符来寻找它的类模块。如果在该类对象中可以找到证明目标的方法，则用该方法来证明目标。如果在该类对象中没有找到方法的定义，则通过第二种继承机制在其超类中寻找。这种搜索沿着程序组织的层次结构一直向上进行，直至找到所需方法，若搜索到根对象仍然没有找到，则目标证明失败，引起回溯。

多继承靠回溯支持。一个类对象可能有多个超类，这时继承路径有多条，多继承通过回溯依次在这些路径上搜索。定义超类的顺序决定了继承路径的搜索顺序。

(e)元级机制 在SCKL语言模型中，元对象主要用于控制目标级对象的目标证明过程。当一个目标级对象与一个元级对象联系在一起时，该目标级对象每次证明目标时都自动与其元级对象通信，并按元级对象描述的过程控制目标对象的行为。由于这种元级机制允许程序员定义控制目标级对象行为的方法，它可方便地用于定义领域模型中独立事件的特殊问题求解方法。

在基于面向对象环境的因果模型中，Prolog强有力的知识表示和知识库查询能力，以及Prolog的一致化和回溯机制，为部件对象行为、属性的定义提供了有力手段，通过合成基于规则的说明性风格和基于对象的结构化、层次化，SCKL语言模型成为建立因果推理环境的理想语言。

四、因果推理环境的体系结构

图5给出了面向对象因果推理环境的基本组成。其中，矩形框表示软件部分，椭圆框表示SCKL语言模型的内部类对象和实例对象。对象编译器主要编译领域专家输入的SCKL源代码，并构造领域因果模型。推理机制根据领域因果模型进行因果推理。对象库管理子系统用于管理和维护因果模型中对象

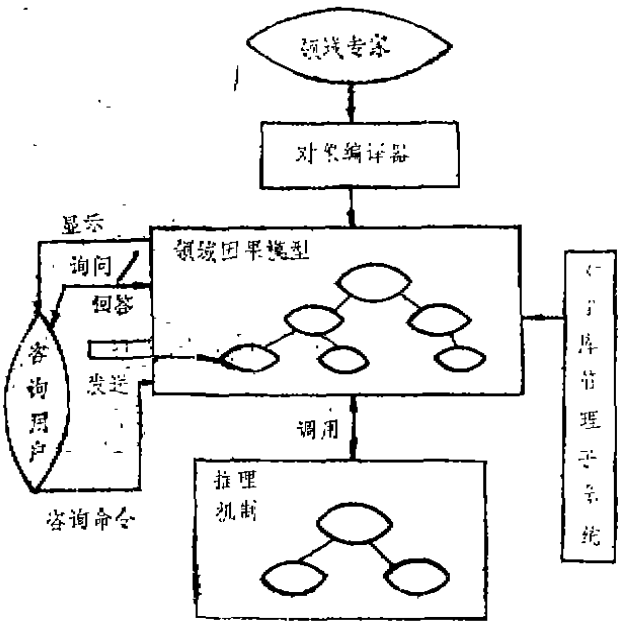


图6 因果推理环境的体系结构

结构的相容性和一致性，负责对象库的修改和扩充。

为了系统元素的一致性，该环境将领域专家和咨询用户都作为对象看待。用户对象

向因果模型发送咨询命令，因果模型按领域因果关系进行推理，找到咨询结论，并向用户对象显示。采用该环境建立并使用领域因果模型的过程可分为三个阶段：

(1) 领域专家定义目标领域因果模型，进行模型规格说明（其中，目标领域深层模型的规格说明包括领域部件、领域因果拓扑结构以及各部件自身的特殊问题求解算法），并按SCKL源代码编程输入；

(2) 对象编译器将专家输入的SCKL程序编译成推理使用的对象结构；

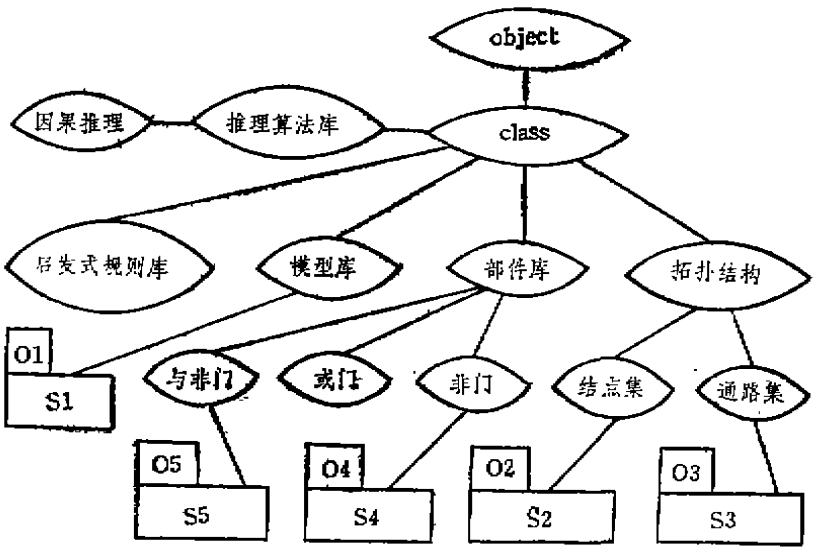
(3) 根据因果推理环境与终端用户对象间的询问/回答对话进行深层因果推理。

我们仍采用图1(b)的例子来说明该因果模型的原型特征。为了构造上述门电路的因果模型，领域专家输入如下SCKL语言源程序：

```
Model ::new(Gate-Circuit, [ ]).
Inverter::new(Inobj, [in-port(1),
        out-port(1), necessary(OK),
        input(true), delay(3)]).
Nand-Gate::new(Nandobj, [in-port(2),
        out-port(1), necessary(OK),
        ∃input=false, delay(5)]).
Gate-Circuit::add-obj([Inobj,Nandobj]).
Gate-Circuit::add-topology([route(Input,
        Inobj),
        route(Input, Nandobj),
        route(Inobj, Nandobj),
        route (Nandobj,Output)]).
```

当对象编译器接收到这些源代码时，将产生相应的SCKL对象并构造面向对象的因果模型，如图6所示，图中椭圆表示类，方框表示具有私有特性的实例对象。

领域因果模型建立后，终端用户就可以向它发送查询消息。假设已知门电路输出为false，其它相关的观察结果如下：




```
S1: name(Gate-Circuit)
    obj-col(02)
    route-col (03)
S2: collection([04, 05])
S3: collection([route(input, 04),
    route(input, 05),
    route(04, 05),
    route(05,output)])
S4: in-port(1)
    out-port(1)
    necessary(OK)
    input=true
    delay(3)
S5: in-port(2)
    out-port(1)
    necessary(OK)
    ∃input=false
    delay(5)
```

图6 门电路模型的部分对象层次

Input=true

∀nand-gate-input=true

向后检查Output=false的起因是nand-gate-output=false，根据与非门的特性，在条件∀nand-gate-input=true的情况下，其输出为false满足了功能要求，necessary(OK)成立，则跟踪到属性值∀nand-gate-input=true是异常情况。继续向后检查该异常属性，找到最终原因是inverter-necessary(OK)不成立，即非门发生故障，产生错误输出。

用户查询消息可以采用如下形式：

Gate-Circuit::query(Observer, Anomalous, Cause)

其中，Observer表示当前观察结果，Anomalous表示要寻找其异常原因的异常现象，Cause在推理结束时约束为最终原因。另外，系统推理采用与用户交互的方式进行，提示用户去观察新的现象，并向用户提供直接的反馈。这样，也将用户作为一个知识源对象，利用用户的经验或原理知识作为系统知识的一个补充，加速推理效率并提高推理的准确度。

五、结 论

本文介绍了一种基于面向对象环境的因果推理模型。与其它因果网络实现相比，该模型的主要特点是：(1) 它不仅能揭示问题独立部件间的因果相关性，而且能表示它们各自的原理性知识，因而能较全面地反映问题领域的深层特性；(2) 在该因果模型中，深层推理由扩展的因果有向图驱动，各个结点则利用其私有状态和属性进行推理，具有对复杂不可分独立部件的表示和处理能力；(3) 因果图结点粒度大，对异常属性的解释可通过在相应结点上附加解释信息实现，也可直接使用原理性知识提供更一般的解释；(4) 利用该因果推理环境，用户很容易建立特定领域的因果模型，只需描述该领域的独立部件属性和拓扑结构，并将它们作为新对象加入模型库即成。

深层因果推理较之浅层推理方法有着不可比拟的优点，但其推理速度影响了它的实用性。在我们提出的因果推理模型中，提高推理速度的关键是寻求快速的图搜索算法，或加入启发式知识剪去无用搜索分支，以提高有向因果的搜索速度。另外，文中描述仅涉及静态结构的深层因果推理，对于动态系统的处理涉及到一些时变属性，需要特殊的机制进行处理，这也是我们下一步要研究的问题。

参考文献

[1] Edward H.Freeman, A Logic-based Prototyping Environment for Process Oriented Second Generation Expert System, SPIE Vol.937, Application of Artificial Intelligence Ⅶ, 1988

[2] E.T.Keravnou and J.Washbrook, What is a deep expert system?An analysis of the architectural requirements of second-generation expert systems, The Knowledge Engineering Review, No.4:3, 1989

[3] L.Steels, Second-Generation Expert Systems, Research and Development in Expert System, Bramer MA(ed.),Cambridge, Cambridge University Press, 1986

[4] A.Abu-hanna and Y.I.Gold, Adaptive, Multilevel Diagnosis and Modeling of Dynamic Systems, International Journal of Expert Systems, Vol.3, No.1, 1990

[5] B.Meye, Genericity Versus Inheritance, Proc. OOPSLA'86, Sep.1986

[6] D.G.Bobrow, Qualitative Reasoning about Physical Systems:An Introduction, AI 24, 1984

[7] H.Simon, Causal Ordering and Identifiability, Models of Discovery, D.Reidel ed.Derdrecht, Holland, 1954

[8] Edward H.Freeman, The implementation of effect decomposition methods for two general structural covariance modeling systems, Ph.D Dissertation, Dept.of Psychology, UCLA, Los Angeles, 1982

[9] L.Steels,Diagnosis with a Function-Fault Model, Applied AI, 1989