现代模态逻辑在计算机科学中的应用研究

陈志远[1] 黄少滨[1] 韩丽丽[2]
(哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001)[1]
(中航工业集团空气动力研究院 哈尔滨 150001)[2]

摘 要 现代模态逻辑是在经典数理逻辑基础上发展起来的,主要包括狭义模态逻辑、道义逻辑、认知逻辑、信念逻辑、时态逻辑与动态逻辑。克里普克语义模型的建立,使得模态逻辑成为现代逻辑的重要分支之一,并成功应用到数学、经济学、社会科学、计算机科学和量子力学等众多领域。介绍了现代模态逻辑研究的主要内容,重点综述了现代模态逻辑在计算机科学的程序设计语言、知识表示与多代理系统以及模型检测、定理机器证明和非单调逻辑5个方面的应用,阐述了现代模态逻辑在计算机科学领域的研究目标、研究进展和发展趋势,最后指出现代模态逻辑研究中存在的问题,并预测其未来可能的研究与发展方向。

关键词 模态逻辑,可能世界,多Agent系统,模型检测,量子力学
中图法分类号 TP311.5 **文献标识码** A

Research on Applications of Modern Modal Logic in Computer Science

CHEN Zhi-yuan[1] HUANG Shao-bin[1] HAN Li-li[2]
(College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China)[1]
(Aerodynamics research institute, Aviation Industry Corporation of China, Harbin 150001, China)[2]

Abstract Modern modal logic is set up on the basis of classical mathematical logic, and mainly includes narrow modal logic, deontic logic, epistemic logic, belief logic, temporal logic and dynamic logic. With the emergence of the Kripke semantic model, modal logic has become one of the most important branches of logics, and has been successfully applied to many fields such as mathematics, economics, social science, computer science and quantum mechanics. This paper introduced the main contents of modern modal logic, and mainly overviewed the following three application aspects of modern modal logic which include programming language, knowledge representation, multi-agent system, model checking, mechanical theorem proving and no monotonic logic. At the same time, this paper presented the research targets, the research advances and the development trend of modern modal logic. At last, the paper pointed out the problems in research on modern modal logic and suggested the trend of future development and research.

Keywords Modal logic, Possible world, Multi-agent system, Model checking, Quantum mechanics

1 引言

模态是客观事物或主观认识的存在、运动、发展、变化、联系、情状、趋势等的性质或状态。模态反映在人们的思维中就表现为模态概念。语言中用以表示模态概念的词语就是模态词。模态词最初指反映事物的必然性、可能性的"必然"、"可能"等词。20世纪以来,模态词的范围扩大到"应当"、"必须"、"允许"、"禁止"、"过去"和"将来"以及"相信"和"怀疑"等[1]。模态逻辑(Modal Logic)[4]是在经典数理逻辑的基础上,增加一些模态词而形成的逻辑。它研究含有模态词的命题和推理的逻辑性质、逻辑形式及其规律。模态逻辑分为狭义模态逻辑(处理与模态词"必然"和"可能"有关的逻辑)和广义模态逻辑(处理与"应该"和"允许"、"过去"和"将来"以及"相信"和"怀疑"等有关的模态词的逻辑),广义模态逻辑包括道义逻辑、时态逻辑以及认知逻辑等[2,3,5],下文所提到的模态逻辑都指广义模态逻辑。

模态逻辑(Modal Logic)是逻辑学的重要分支之一,也是最古老的分支之一,其最初研究起于古希腊的亚里士多德。亚里士多德曾将命题划分为实然命题、必然命题和偶然命题[6],经典数理逻辑研究实然命题。

而模态逻辑则研究必然命题和偶然命题。现代逻辑创立于19世纪末至20世纪30年代,其创始人是美国逻辑学家刘易斯(C. I. Lewis)。刘易斯从分析经典逻辑中的实质蕴涵所存在的问题出发,提出所谓的"严格蕴涵"试图避免实质蕴涵的问题[7]。1912年刘易斯发表的《蕴涵和逻辑代数》中提出了不同于实质蕴涵的严格蕴涵,并构造出一系列严格蕴涵系

本文受国家自然科学基金(71272216,60903080),国家科技支撑计划课题(2009BAH42B02,2012BAH08B02),中央高校基本科研业务专项资金项目(100603,1212)资助。

陈志远(1978—),男,博士生,讲师,主要研究方向为模态逻辑、模型检测,E-mail:chenzhiyuan@hrbeu.edu.cn;黄少滨(1965—),男,博士,教授,主要研究方向为模态逻辑、可信计算;韩丽丽(1983—),女,硕士,工程师,主要研究方向为软件体系结构、形式化方法。

统[7]。1914年，他又发表了《严格蕴涵的运算》和《蕴涵的矩阵代数》两篇论文，提出了严格蕴涵的命题演算[8,9]。1932年，他同兰福德(C. H. Langford)合著《符号逻辑》，建立了模态逻辑系统S1和S2，后来又建立了S3、S4和S5系统，奠定了现代模态逻辑的基础。20世纪40年代末，卡尔纳普(R. Carnap)开始从语义方面研究模态逻辑[10,11]。

50年代末－60年代初，坎格尔(S. Kanger)[12]、辛提卡(J. Hintikka)[13]与克里普克(S. A. Kripke)[14]等人发展了卡尔纳普的理论，提出了比较完整的模态逻辑的模型理论。60年代以后模态逻辑有很大发展，出现了许多新的系统，特别出现了许多非标准的模态逻辑系统，如认知逻辑[16]、道义逻辑[15]、时态逻辑[17]等。模态逻辑由于研究和阐明了必然、可能、应当、知道等本体论和认识论概念的逻辑性质，因而具有深刻的哲学意义。特别是克里普克(S. A. Kripke)的可能世界语义模型的建立，使得模态逻辑成为现代逻辑中最重要的分支之一，并在数学、计算机等学科有着十分重要的应用。

2 现代模态逻辑研究的基本内容

现代模态逻辑研究的基本内容有以下4个方面：

(1)语法的研究。主要是指在形式化和公理化方法下考察有关对象的模态形式，然后建立起形式系统，考察系统的性质及系统之间的关系，并验证系统的可靠性和完全性。刘易斯对模态形式的考察以及S1－S5的构造就是语法研究。

(2)模态代数语义学，又称为模态代数，是对模态形式和模态系统的代数解释，在方法和理论方面，模态代数语义学对模态逻辑都有重要意义。

(3)逻辑语义学，包括直观语义学和严格语义学。直观语义学是对各种模态逻辑形式的直观解释。严格语义学是指由一些抽象的对象按严格方式建立起来的系统或结构。严格语义学具有很重要的作用。只有建立起严格逻辑语义学的逻辑体系才是合格的逻辑体系。由于对模态很难下一个精确的定义，所有严格语义学出现较晚，20世纪40年代，卡尔纳普引入了莱布尼兹(G. W. Leibniz)关于必然性的思想，给出了S5的严格语义定义。随后，克里普克建立了严格的语义学，即可能世界语义学[14]。克里普克可能世界语义学的建立解决了现代模态逻辑中模态概念不清并无分析工具这一极其重要的问题，并为模态逻辑的发展提供了新的理论和方法。同时，可能世界语义学的创立也标志着现代模态逻辑已经完善。此后，现代模态逻辑大体以语义学为中心展开研究。

(4)模态逻辑的研究除了关于模态逻辑系统本身的3个方面的主要内容外，模态逻辑的应用研究也日益成为模态逻辑研究的一个重要方向，包括数学、社会科学、经济学、工程学和计算机科学等领域[18]。

3 现代模态逻辑在计算机科学中的应用

计算机科学是研究计算机及其周围各种现象和规律的科学，亦即研究计算机系统结构、程序系统(即软件)、人工智能以及计算本身的性质和问题的学科。它是一门包含各种各样与计算和信息处理相关主题的系统学科，从抽象的算法分析、形式化语法等等，到更具体的主题如编程语言、程序设计、软件和硬件等。研究有关计算机系统描述和推理的形式化理论是计算机科学的一个重要研究目标与研究方向。在这方面，模态逻辑的作用尤为突出，与传统的经典逻辑相比，模态逻辑具有更少的公理，更接近自然语言的翻译和更高效的计算能力。

3.1 现代模态逻辑与程序语言

在计算机出现后的最初几十年里，计算机实质上是一个巨大的计算器，数字被录入，计算结果被输出。直到20世纪70年代，科学家们才意识到需要正确地验证这些计算结果。随着计算机硬件和软件的发展，任务和变化的数据核查变得更加困难。因此，程序员不得不考虑到时间推移下系统的行为。在这个契机下，时序逻辑被引入计算科学，这是计算科学发展历史的重要转折点[19]。作为广义模态逻辑分支之一的时态逻辑(temporal logic，也叫时序逻辑)[22]是非经典逻辑中的一种，它研究如何处理含有时间信息(现在、过去、将来；之前、之后等)的事件的命题和谓词。时态逻辑体系包含的要素有：

(1)基本符号：事件e、关系或谓词r、时间区间i等；

(2)时态谓词：$after(e,r)$、$before(e,r)$等；

(3)时态事件演算规则：初始规则、终止规则等，如$holds(before(e,r))$；$terminates(e,r)$表示终止规则，意为若事件已使谓词r失效，则在e之前且r成立的一段区间中r为真；

(4)时态逻辑运算：时态区间的并、交，时态谓词的与、或、非等。

20世纪60年代普里奥里(Arthur Prior)[21]提出介入的基于模态逻辑的特殊的时间逻辑系统，这一理论后来被艾米尔·伯努利(Amir Pnueli)等逻辑学家进一步发展。1977年，伯努利开创性地把时态逻辑引入计算机科学[19]，他的时态逻辑是非经典逻辑中的一种，研究如何处理含有时间信息的事件的命题和谓词。现在通常称为时序逻辑的计算机系统，就出现在这一年，伯努利在子编程语言与系统验证方面做出的杰出贡献具有里程碑意义。伯努利和他的同事曼纳(Z. Manna)共同开发的时态逻辑系统叫命题线性时态逻辑系统(Proposition Linear Temporal logic，PLTL)[20]。PLTL包含可数无穷多个命题变元；逻辑联结词包括否定：$\neg$，合取：$\wedge$，析取：$\vee$，蕴含：$\supset$，等价：$\equiv$；时态算子包括任一时刻：$\Box$，某一时刻：$\Diamond$，下一时刻：$\bigcirc$，直到：μ。

3.1.1 模态逻辑与XYZ/E

20世纪80年代初，中科院软件所唐稚松院士致力于将形式化的时序逻辑语言与软件工程相结合的XYZ系统的研究。在1983年国际信息处理联合会(IFIP)巴黎大会上，他提出了世界上第一个可执行时序逻辑语言XYZ/E[23]。XYZ/E既是时序逻辑系统又是具有常见程序语言风格且可实际用于编程运行的程序语言，它将程序的动态语义与静态语义结合起来，并第一次将状态转换的控制机制引入到逻辑系统之中，这一成果被国际著名计算机专家称为软件工程领域中发展可执行时序逻辑的先驱。时序逻辑语言XYZ/E被成功应用于硬件系统[25]、混成系统[26]的行为描述与验证以及程序语义的解释[27,29]，用于描述软件体系结构[28]。

XYZ/E语言最基本的语言成分是条件元(conditional element)。如形式(1)所示，条件元直接定义了程序相邻之间的转换关系；形式(2)所示条件元表示程序抽象规范，其中符号@可以是下一时刻算子$O和最终时刻算子◇；两种形式的条件元都是一个时序逻辑式，其语义即为此时序逻辑式的语

义模型。

$$LB=y\wedge R\Rightarrow \$O(v_1,v_2,\cdots,v_n)=(e_1,e_2,\cdots,e_n)\wedge \$OLB=z \quad (1)$$

$$LB=y\wedge R\Rightarrow @(Q\wedge LB=z) \quad (2)$$

$$\square[A_1,A_2,\cdots,A_n] \quad (3)$$

XYZ/E 中表示算法的语言成分为如形式(3)所示的单元(unit)。其中 $A_1,A_2,\cdots,A_n$ 是条件元,符号";"等同于逻辑词合取。单元也是一个时序逻辑式,其语义即对应于此时序逻辑式的语义模型。

3.1.2 模态逻辑与逻辑程序设计语言

逻辑程序设计语言是一种建立在逻辑学理论基础之上的程序语言,最初被运用于自然语言等研究领域。现在它已广泛地应用在人工智能的研究中,它可以用来建造专家系统、自然语言理解、智能知识库等。逻辑程序描述的是数据对象之间的关系,它的抽象层次更高且不限于函数(映射)关系。关系也是联系,对象和对象以及对象和属性的联系就是我们所说的事实。事实之间的关系以规则表述,根据规则找出合乎逻辑的事实就是推理。因此,逻辑程序设计范型是陈述事实,制定规则,程序设计就是构造证明。程序的执行是在推理,与传统程序设计范型有较大的差异。1970 年诞生的 PROLOG,以及 LISP 语言和 PROLOG 结合后的 LOGLISP 都是逻辑程序设计语言的杰出代表。

PROLOG 语言的基本语句有 3 类,分别代表事实、规则和询问,并同有头(无体、有体)和无头的霍恩子句相对应。因而用 PROLOG 语言进行的程序设计可归结为宣布事实,定义规则和提出询问。PROLOG 程序的解释执行过程采用特定的输入归结,即从目标语句出发求出它和原来子句集的一个子句的归结式,新的子句再与原来子句集的一个子句求归结式,以此类推。任一时刻都不在两个导出子句或原来子句集的两子句间求归结式。PROLOG 解释系统实际上就是一个以归结原理为基础的定理证明程序或问题求解程序。

3.2 现代模态逻辑与知识表示和多代理系统

随着计算机科学的飞速发展,人工智能(Artificial Intelligence)这个研究方向越来越备受关注。它是研究、开发用于模拟、延伸和扩展人的智能的理论、方法、技术及应用系统的一门新的技术科学[30,31]。莫尔(R. C. Moore)等人将模态逻辑应用于建立信念和知识的理论,给模态算子提供解释,发展了知识和行为的逻辑[32]。他采用可能世界的方法探讨辛提卡所提出的知识逻辑。通过改变可达性关系解释把必然性和可能性的克里普克语义转换为知识的辛提卡语义[33]。在文献[32]中,莫尔引入了 K 关系来分析 $Know(\alpha,A)$ 形式的语句,$Know(\alpha,\omega_1,\omega_2)$ 含义是可能世界 ω_2 和智能代理(Agent)α 在可能世界 ω_1 中所知道的事实也可以在可能世界 ω_2 中知道。在知识的可能世界理论中,有关知识的语句只是相对于某可能世界为真,在此前提下,莫尔引入了公理 L1、L2 和 K1。

公理 L1 $\forall A(TURE(A)\leftrightarrow T(\omega_0,A))$。

公理 L2 $(\forall\omega\forall AB)(T(\omega,AND(A,B)))\leftrightarrow(T(\omega,A)\&T(\omega,B))$。

公理 K1 $(\forall\omega tA)(T(\omega_1,KNOW(t,A))\leftrightarrow(\forall\omega_2)(K(D(\omega_1,t),\omega_1,\omega_2)\rightarrow T(\omega_2,A)))$。

有关知识形式化的可能世界分析可以使推理在一阶工作框架中进行并产生智能代理的知识。对代理(Agent)以及多代理(Multi-Agents)的研究近几年来已成为分布式人工智能(Distributed Artificial Intelligence,DAI)研究的一个热点,一些文献称代理技术是软件领域里一个意义深远的突破。基于智能代理的思想,人们提出了一种新的人工智能定义:人工智能是计算机科学的一个分支,它的目标是构造能表现出一定智能行为的代理。美国 Stanford 大学计算机科学系的海斯·罗斯(Hayes Roth)在国际人工智能联合会(IJCAI'95)的特邀报告中谈到:"智能的计算机代理既是人工智能最初的目标,也是人工智能最终的目标。"目前关于代理的研究,无论是在人工智能领域还是在计算机科学的其它领域,都十分活跃。

一个代理要有智能,就需要对自己所拥有的知识进行推理,因此需要知识的表达和推理机制。多代理体系中,知识具有局部性,而问题具有全局性,在大多数情况下,需要同其他的代理联合解决一个问题,这样代理间的信息传递不可避免,因此需要有代理通讯语言(ACL)[34]。

作为人工智能和分布式计算的结合,分布式人工智能正逐渐受到人们的重视。自从 1979 年第一次在 MIT 召开分布式人工智能研究人员的会议至今,出现了大量的理论和研究系统。分布式人工智能研究的目标是要创建自然和社会系统精确的概念模型。在分布式人工智能当中,由于智能本质上不是一个独立存在的概念且智能在团体当中实现,因此分布式人工智能研究感兴趣的主要是几个主体之间的合作、交互等方面。多 Agent 系统是分布式人工智能的一个重要研究发方向,它主要研究一组自治的智能 Agent 之间智能行为的协调。知识、规划、不同技能和自身动作的协调是一个过程,它在多 Agent 系统当中非常重要。多 Agent 系统由于更能体现人类社会的社会智能,具有更大的灵活性和适应性,更加适合开放、动态的世界环境,因此越来越受到人们重视。由于近年来 Internet 的发展,MAS 的研究从以前面向合作的角度来考虑,逐渐转变到趋向于从单个智能 Agent 的角度来考虑。更一般的问题是 Agent 应该具有什么样的结构、能力才能够在一个有限时间约束、开放的多 Agent 环境中自主地行动、决策以及与其它 Agent 交流。

目前对 Agent 以及多 Agent 系统的研究主要集中在以下几个方面[35-38]:

(1)Agent 和多 Agent 的理论;

(2)Agent 的体系结构和组织;

(3)Agent 语言;

(4)Agent 之间的协作和协调;

(5)Agent 之间的通信和交互技术;

(6)多 Agent 学习以及 MAS 应用。

在多 Agent 系统中,用模态逻辑来建立信念和知识描述可以解决一阶逻辑对"信念"和"愿望"等有意识的概念的指示含糊问题,从而具有更清晰的表达能力和更有效的推理形式。建立 Agent 的模型最好的结果是使用模态逻辑工具建立思维状态模型,其语义解释是基于辛提卡提出的可能世界语义模型。基于意识立场对模型的研究主要是通过引入不同的模态算子对 Agent 的各种思维属性进行形式化,这方面比较有影响的工作有基于正规模态逻辑的 P. R. Cohen 和 H. J. Levesque 的 Intention 理论[39],其提出了一个理性 Agent 的逻辑,列出了基本模态算子之间的关系和性质,并在这个框架

的基础上，引入了一些导出算子，从而构成了关于理性动作的部分理论[40]。A. S. Rao 和 M. P. Georgeff 的 BDI 模型结构给出了基于 beliefs，desires，intentions（简称 BDI）3 个模态词的 Agent 逻辑[41]，其是建立在分枝时序逻辑模型之上的，其中 BDI 可达世界是分枝时序结构，他们使用的形式化系统是对计算树逻辑 CTL* 的扩充。其它工作如 Kurt Konolige 和 Martha E. Pollack-BDI 模型[42]、JohnBell-BDI 模型、G. Gaspar 和 H. Coelho-BDI 模型[43]等，都是针对已有的模型存在的问题而进行的改进，其框架大多仍然是基于模态逻辑的。

3.3 现代模态逻辑与模型检测技术

模型检测技术是 20 世纪最成功的自动验证技术之一，由美国的克拉克（E. M. Clarke）和艾默森（E. A. Emerson）[45]以及法国的 J. P. Quielle 和斯发基斯（J. Sifakis）在 1981 年分别提出[46]，其主要通过显式状态搜索或隐式不动点计算来验证有穷状态并发系统的模态/命题性质。因模型检测可以自动执行，并能在系统不满足性质时提供反例路径，故在工业界比演绎证明更受推崇。尽管限制在有穷系统上是一个缺点，但模型检查可以应用于许多非常重要的系统，如硬件控制器和通信协议都是有穷状态系统。很多情况下，可以把模型检测和各种抽象[47,48]与归纳原则[49,50]结合起来验证非有穷状态系统（如实时系统）。

模型检测的基本思想是用状态迁移系统 S 表示系统的行为，用模态逻辑公式 F 描述系统的性质。这样便将“系统是否具有所期望的性质”转化为“状态迁移系统 S 是否是公式 F 的一个模型？”的数学问题，用公式表示为 $S|=F$？对有穷状态系统，这个问题是可判定的，即可以用计算机程序在有限时间内自动确定。模型检测已被应用于计算机硬件[51]、通信协议[52,53]、控制系统[54,55]、安全认证协议[56,57]等方面的分析与验证中，取得了令人瞩目的成功，并从学术界辐射到了产业界。克拉克和艾默森以及斯发基斯 3 人也因为在模型检测领域所做出的贡献而获得 2007 年的图灵奖。

模型检测中，描述系统性质是非常重要的一项工作，模态/时序逻辑是模型检测的基础，主要用 3 种模态逻辑来描述系统性质，分别是命题线性时序逻辑（Propositional Linear Temporal Logic，PLTL）[59]、计算树逻辑（Computation Tree Logic，CTL）[58]和命题 μ 演算[60]。

命题线性时序逻辑 PLTL 关心的是系统的任意一次运行中的状态以及它们之间的关系。PLTL 的字母表包括：

1）原子命题符号：$p,q,r,\cdots$；

2）逻辑连结符：$\neg$（非），$\wedge$（与），$\vee$（或）；

3）模态算子：$\Diamond$（Eventually）表示现在或以后某一状态，$\Box$（Always）表示现在和以后所有状态。

一个系统的运行可以看成是系统状态的变化，而系统状态变化的可能性可以表示成树状结构。一棵计算树逻辑（CTL）是一种分枝时序逻辑，它可以描述状态的前后关系和分支情况。计算树逻辑（CTL）的字母表包括：

1）原子命题符号：$p,q,r,\cdots$；

2）逻辑连结符：$\neg$（非），$\wedge$（与），$\vee$（或）；

3）模态算子：E（Exists）表示对于某个分支，A（Always）表示对于所有分枝，F（Future）表示现在或以后某个状态，X（Next time）表示下一个状态，U（Until）表示直到某一状态，G（Global）表示现在和以后所有状态。

前两个算子描述分支情况，后 4 个算子描述状态的前后关系，计算树逻辑中描述分支情况和描述状态的前后关系的算子成对出现，即一个描述分支情况的算子后面必须有一个描述状态的前后关系的算子。

命题 μ 演算（Propositional μ-*Calculus*）是一种面向动作的逻辑语言，它关心的是系统的动作与状态之间的关系。其优点是它的表示能力非常强，PLTL 和 CTL 都可以嵌入到它的真子集中，并且相应的子集具有与 PLTL 和 CTL 相同的复杂度的模型检测算法，因此很多学者将 μ 演算作为模型检测的一般框架加以研究[61-63]，μ 演算的主要缺点是公式不易读懂。

模型检测的优点在于可以完全自动地进行验证，现在已经开发出很多验证不同类型逻辑公式的软件工具，早期的有美国卡耐基梅隆大学开发的用以检测一个有限状态系统是否满足 CTL 公式的模型检测工具 SMV[64]。美国贝尔实验室开发的模型检测工具用以检测一个有限状态系统是否满足 PLTL 公式及其他一些性质的模型检测工具 SPIN[65,66]，另外，英国爱丁堡大学和美国北卡罗莱纳州立大学开发了几个不同版本的模型检测工具 CWB[67]。近年来，新的模型检测工具是直接面向编程语言的，比如贝尔实验室开发的 VeriSoft[68]、瑞典 Telelogic 公司的软件开发工具 TAU[69]包含系统建模、系统测试以及模型检测。

近年来，模型检测的研究取得了重大进展，特别是对实时系统的模型检测。主要表现在 3 个方面：

（1）对于实时系统的数学模型表示，引入了时间 Petri 网、时间自动机，以及各种进程代数的时间扩充。其中以时间自动机的影响和应用最为广泛；

（2）提出能描述实时系统的模态/时序逻辑，如 CTL、PLTL 和演算的实时扩充；

（3）针对这些实时系统的数学模型和逻辑设计了各种模型检测的算法，并实现了相应的分析与验证工具。

3.4 现代模态逻辑与定理机器证明

所谓定理的机器证明，是指使用计算机证明定理的成立，即把人证明定理的过程，通过一套符号体系加以形式化，变成一系列在计算机上自动实现的符号计算过程。其实质是把具有智能特点的推理演绎过程机械化。定理的机器证明是涉及人类智能问题的主要研究课题之一。从传统的手工证明到定理的机器证明，是现代数学思想方法的一次重大突破。机器证明大体上经历了这样一个过程：公理化→代数化→坐标化→机械化。

近代的机器证明思想由莱布尼兹首先提出，并直接导源于他的定理证明机械化设想。到 19 世纪末，希尔伯特等创立并发展了数理逻辑，为定理证明机械化提供了一个强有力的工具，使这一设想有了明确的数学形式。机器证明史上第一项奠基性的突破，是由美国的卡尔基大学—兰德公司协作组做出的。1956 年，此协作组的纽厄尔、西蒙和肖乌等人通过研究证明定理的心理过程，建立了机器证明的启发式搜索法，编制了一个“逻辑理论机”程序（LT），成功证明了罗素和怀特海所著的《数学原理》第二章 52 条定理中的 38 条，这一年可作为历史上计算机证明以至于人工智能研究的开端。1963 年，他们又在计算机上证明了全部 52 条定理。50 年代末，著名美籍华人、数理逻辑学家、美国洛克菲勒大学的王浩教授发明了王浩算法，把机器证明规则化。1959 年他只用 9 分钟的

机器时间，用计算机证明了《数学原理》中的一阶逻辑部分的全部350多条定理，引起数学与数理逻辑界巨大轰动。70年代，机器证明获得新的重大进展。1976年美国伊利诺斯大学的阿佩尔(K. Appel)、哈肯(W. Haken)和科奇(J. Koch)完成了构造可约构形的不可避免集的工作，在电子计算机上解决了100多年来传统人脑支配手工操作所长期未能解决的著名难题——"四色猜想"问题。随着研究的进一步深入，今天人们已经能根据机械化方法编制程序，在计算机上给出证明[71]。

随着知识工程的迅速发展，大量定理机器证明方面的学者转向自动推理的研究，这也扩大了定理机器证明的研究领域，即不再局限于数学领域，而是面对着整个新一代计算机。因此，定理机器证明以及它所派生出的自动推理的研究，是计算机科学中的基本而重要的理论研究课题。而在定理的机器证明中，逻辑学是最主要的工具，从符号逻辑到多值逻辑和模态逻辑，都作为知识的形式化表示手段被用于定理的机器证明。如今，定理的机器证明也已经扩展到模糊逻辑。如姜云飞教授研究定理机器证明中常用的归结方法与非归结方法，提出了算子模糊逻辑以及新的重写证明方法并将这种方法成功地推广到一阶逻辑，对BOYER-MOORE定理证明器提出了改进，提高了证明效率。

3.5 现在模态逻辑与非单调逻辑

在现实生活中，人们所遇到的推理问题往往面临着复杂的或者不可测的情况，并不是简单的线性推理问题。非单调推理具有一定的灵活性，所得结论具有暂时性。随着新信息的出现，可以不断修正结论。这满足了常识推理的要求。非单调推理可以处理日常情景中所遇到的复杂推理问题。

模态逻辑与非单调逻辑的结合产生了非单调模态逻辑。模态非单调逻辑是一类通过引人模态算子刻画非单调推理性质的逻辑形式，它已成为研究基于协调性的非单调逻辑的一个重要途径。传统上，模态非单调逻辑是通过经典模态算子(表示协调性)或认知算子(知道或相信)刻画非单调性。文献[72]首先将模态逻辑与非单调逻辑结合在一起，提出了一个关于典型与例外的模态逻辑g，经验世界可用一种可能世界型式的语义完全刻画，其中包含一个内在的偏序关系对可能世界进行典型性排序，而"典型真"与"例外真"分别刻画为对于这一排序世界的极大世界与极小世界，这一语义使得能在一种纯粹单调框架内捕捉一些非单调推理思想，然后由模态逻辑g很自然地加进非单调性，发展一种非单调模态逻辑，它包含经验理论作为经验规则集的外延与经验模型表达经验与例外的直觉关系及其所刻画的非单调性质。

4 总结与展望

现在模态逻辑在计算机科学中扮演着越重要的角色，是程序设计语言、知识和行为表示、多Agent系统和程序规约与模型检测等方向的理论基础。模态逻辑与计算机科学的关系类似于数学与物理学的关系。模态逻辑在促进计算机科学发展的同时，其本身也得到了长足发展和日益丰富、完善。正因为如此，现代模态逻辑与计算机科学的完美结合以及可能世界理论的引人，使得计算机科学领域的多个学科方向都得到了新的发展。

针对现代模态逻辑与计算机科学的关系及现在模态逻辑和计算机科学的发展趋势，可以预计以下方面将成为模态逻辑在计算机科学领域中新的研究热点问题

(1)在计算机科学中多种模态逻辑的组合应用问题

1)如何合理解决表达能力和计算复杂性之间的矛盾

过去讨论过把动态认知逻辑和时态逻辑组合在一起的问题，我们并不是在提倡"盲目的"组合，大家的确意识到了其中涉及的复杂性的极限问题。因为模态逻辑的表达能力和计算复杂性一直是计算机科学理论里重要的一项研究内容，所以如何合理地组合几种模态逻辑，发挥其表达能力的优势，并最大限度地降低计算复杂性，将是一个新的挑战。

2)各种逻辑间的组合原则问题

什么样的原则可以使得组合的逻辑简单？能够避免"科学悖论"，即创造的理论比现实还要复杂？能解释现实，设法简化现实，也将是一个新的研究热点。

(2)模态逻辑在量子力学和量子理论中的应用

20世纪二、三十年代量子力学的建立无疑是人类科学史上的又一次革命，量子力学引发了一系列划时代的科学发现与技术发明，对人类社会的进步做出重要贡献，标志着人类认识自然实现了从宏观世界向微观世界的重大飞跃。随着量子力学和量子理论的发展和完善，将量子力学和计算机科学结合并研制出量子计算机是人类的一大梦想。量子逻辑是量子力学的理论基础。为了解释量子力学，1972年范·弗拉森(van Fraassen)用模态逻辑的语义分析代替量子逻辑的分析。结果，这一解释被称为"量子逻辑的模态解释"试图消解微观世界的测量症结。这一解释给"测量难题"的解决提供了新的视角，也为物理学哲学的发展开辟了新的方向。该思想经过Kochen、Diecks、Vermmas、Healey、Clifton、Dickson、Bub、Bacciagaluppi等人的发展和完善，形成了一种很有影响力且最具有发展前景的量子力学解释理论。从此以后，"模态解释"一词获得了更为普遍的意义。随着量子力学和量子理论研究的逐步深入，模态逻辑在量子理论中的应用也将成为一个值得进一步研究的问题。

结束语 模态逻辑的出现使逻辑学领域又出现了一门新兴的分支。随着科学的发展，尤其是计算机技术的日益进步，模态逻辑已经从单纯的逻辑学领域进入到更多的应用学科中，而且日益发挥其重要作用，促进了各领域的快速发展。同时，各应用领域的进步也同步促进着模态逻辑理论的日益完善。相信不久的将来，模态逻辑还会在诸多新兴的领域中得到应用。

参 考 文 献

[1] Perkins M R. Modal expressions in English [M]. Ablex Publishing Corporation Oates, 1983: 22-28

[2] 弓肇祥. 广义模态逻辑[M]. 北京：中国社会科学出版社，1993：112-119

[3] 周北海. 模态逻辑导论[M]. 北京：北京大学出版社，1997：32-41

[4] Hughes G E, Cresswell M J. A new introduction to modal logic [M]. Burns & Oates, 1996: 110-118

[5] Blackburn P, De Rijke M, Venema Y. Modal logic[M]. Cambridge Univ Pr, 2002: 88-93

[6] Bull R, Segerberg K. Basic modal logic [J]. Handbook of philosophical logic, 1984, 2: 1-88

[7] Lewis C I IV. Implication and the algebra of logic[J]. Mind,

1912,21(84):522
[8] Lewis C I V. Discussions: the calculus of strict implication[J]. Mind,1914,23(1):240
[9] Lewis C I, Langford C H. Symbolic logic[M]. The Century co., 1932:32-45
[10] Carnap R. Philosophy and logical syntax[M]. K. Paul, Trench, Trubner&Co., ltd., 1935:135-146
[11] Carnap R. Testability and meaning[J]. Philosophy of science, 1936,3(4):419-471
[12] Kanger S. On the characterization of modalities[J]. Theoria, 1957,23(3):152-155
[13] Hintikka J. Models for modalities[M]. Reidel,1969:43-56
[14] Kripke S. Semantical considerations on modal logic[J]. Acta philosophica fennica,1963,16:83-94
[15] Von Wright G H I. DEONTIC LOGIC [J]. Mind, 1951, 60 (237):1
[16] Hintikka J. Individuals, possible worlds, and epistemic logic[J]. Nous,1967,1(1):33-62
[17] Cocchiarella N B. Tense and Modal Logic: a study in the topology of temporal referénce[J]. Unpublished Dissertation, UCLA, 1966:12-18
[18] 周祯祥. 现代模态逻辑的多元视野[J]. 华南师范大学学报:社会科学版,2006(5):7-11
[19] Manna Z, Pnueli A. Verification of concurrent programs: The temporal framework[J]. The correctness problem in computer science,1981:215-273
[20] Manna Z, Pnueli A. The temporal logic of reactive and concurrent systems: Specification[M]. Springer,1992:231-234
[21] Prior A N. Now[J]. Nous,1968,2(2):101-119
[22] Rescher N, Urquhart A. Temporal logic[M]. New York: Springer-Verlag,1971:203-207
[23] 唐稚松. XYZ 系统的设计思想[J]. 软件学报,1990,1:47-54
[24] Zhu Xue-yang, T Zhi-song. A Temporal Logic-Based Software Architecture Description Language XYZ/ADL[J]. Journal of Software,2003,4:35-42
[25] 韩俊刚,王岩冰,沈武威. 用 XYZ/E 语言描述和验证硬件的行为[J]. 软件学报,1996,7(11):676-682
[26] 阎安,唐稚松. 基于 XYZ/E 的混成系统[J]. 软件学报,2000,11(1):1-7
[27] 郭亮,唐稚松. XYZ/E 面向对象程序语义概述[J]. 软件学报,2003,14(3):356-361
[28] 张广泉,唐稚松. 基于时序逻辑语言 XYZ/E 的软件体系结构描述方法[J]. 淮阴师范学院学报:自然科学版,2002(1):26-31
[29] 朱雪阳,唐稚松. UML 活动图的时序逻辑语义[J]. 计算机研究与发展,2005,42(9):1478-1484
[30] Nilsson N J, Automation A M. An Application of Artificial Intelligence Techniques[C]// Proceedings of the 1st International Joint Conference on Artificial Intelligence. 1969:509-520
[31] Winston P H. New progress in artificial intelligence[J]. AITR-310,1974:77-86
[32] Moore R C, Center S P. A formal theory of knowledge and action[J]. 1984:226-238
[33] Moore R C, Center S P. Possible-world semantics for auto epistemic logic[J]. 1984:116-128
[34] Wooldridge M, Jennings N R. Intelligent agents: Theory and practice[J]. The knowledge engineering review,2009,10(02): 115-152
[35] Nwana H S. Software agents: An overview[J]. The Knowledge Engineering Review,2009,11(3):205-244
[36] Wendt A E. The agent-structure problem in international relations theory[J]. International Organization,2009,41(3):335-370
[37] Cossentino M, Burrafato P, Lombardo S, et al. Introducing pattern reuse in the design of multi-agent systems[J]. Agent Technologies, Infrastructures, Tools, and Applications for E-Services,2010:107-120
[38] Jennings N R. Commitments and conventions: The foundation of coordination in multi-agent systems[J]. The knowledge engineering review,2009,8(3):223-250
[39] Cohen P R, Levesque H J. Intention is choice with commitment [J]. Artificial intelligence,1990,42(2/3):213-261
[40] Cohen P R, Levesque H J. Communicative actions for artificial agents[C]//Proceedings of the First International Conference on Multi-Agent Systems(ICMAS-95). 1995:65-72
[41] Rao A S, Georgeff M P. BDI agents: From theory to practice[C]// Proceedings of the first international conference on multi-agent systems(ICMAS-95). 1995:312-319
[42] Konolige K, Pollack M E. A representation list theory of intention[C]//International Joint Conference on Artificial Intelligence. 1993,13:390-390
[43] Gaspar G, Coelho H. Where do intentions come from?: A framework for goals and intentions adoption, derivation and evolution [J]. Progress in Artificial Intelligence,1995:115-127
[44] Queille J, Sifakis J. Specification and verification of concurrent systems in CESAR[C]// International Symposium on Programming. 1982:337-351
[45] Clarke E M, Emerson E A. Design and verification of synchronization skeletons using branching time temporal logic[C]// Proceedings of Workshop on Logic of Programs. 2002:52-71
[46] Clarke E M, Grumberg O, Long D E. Model checking and abstraction[J]. ACM Transactions on Programming Languages and Systems(TOPLAS),1994,16(5):1512-1542
[47] Long D E. Model checking, abstraction, and compositional verification[D]. Citeseer,1993
[48] Zheng H, Yao H, Yoneda T. Modular Model Checking of Large Asynchronous Designs with Efficient Abstraction Refinement [J]. IEEE Transactions on Computers,2010,59(4):561-573
[49] De Moura L, Ruess H, Sorea M. Bounded model checking and induction: From refutation to verification[C]// Computer Aided Verification. 2003:14-26
[50] Eén N, Sorensson N. Temporal induction by incremental SAT solving[J]. Electronic Notes in Theoretical Computer Science, 2003,89(4):543-560
[51] McMillan K L. A methodology for hardware verification using compositional model checking[J]. Science of Computer Programming,2000,37(1-3):279-309
[52] Edelkamp S, Leue S, Lluch-Lafuente A. Directed explicit-state model checking in the validation of communication protocols[J]. International Journal on Software Tools for Technology Transfer(STTT),2004,5(2):247-267

[53] Argón P, Delzanno G, Mukhopadhyay S, et al. Model Checking Communication Protocols [C]//SOFSEM 2001: Theory and Practice of Informatics. 2001:160-170

[54] Faber J, Meyer R. Model checking data-dependent real-time properties of the European Train Control System[C]// Formal Methods in Computer Aided Design, FMCAD'06. 2006:76-77

[55] Clarke E, Fehnker A, Han Z, et al. Abstraction and counterexample-guided refinement in model checking of hybrid systems [M]. Citeseer, 2003:233-237

[56] Marrero W, Clarke E, Jha S. A model checker for authentication protocols[D]. Rutgers University, 1997:188-196

[57] Basin D, Modersheim S, Vigano L. An on-the-fly model-checker for security protocol analysis[D]. Computer Security-ESORICS, 2003:253-270

[58] Clarke E M, Emerson E A, Sistla A P. Automatic verification of finite-state concurrent systems using temporal logic[J]. ACM Transactions on Programming Languages and Systems, 1986, 8: 244-263

[59] Vardi M Y, Wolper P A. An automata-theoretic approach to automatic program verification[C]// Proceedings of the First Symposium on Logic in Computer Science. 1986:322-331

[60] Bradfield L, Stirling C. Modal logics and mu-calculi: an introduction[M]. Handbook of Process Algebra, 2001:293-330

[61] Emerson E A, Lei C L. Efficient model checking in fragments of the propositional mu-calculus[J]. 1986:267-278

[62] Stirling C, Walker D. Local model checking in the modal mu-calculus[C]// TAPSOFT'89. 1989:369-383

[63] Emerson E A, Jutla C S, Sistla A P. On model checking for the [mu]-calculus and its fragments[J]. Theoretical Computer Science, 2001, 258(1/2):491-522

[64] Clarke E M, Emerson E A, Sifakis J. Model checking: algorithmic verification and debugging [J]. Communications of the ACM, 2009, 52(11):74-84

[65] McMillan K L. Symbolic model checking: an approach to the state explosion problem [R]. Carnegie-Mellon University, Department of Computer Science, Report CMU-CS-92-131. 1992

[66] Holzmann G J. The model checker SPIN[J]. IEEE Transactions on software engineering, 1997, 23(5):279-295

[67] Holzmann G J, Smith M H. Software model checking: Extracting verification models from source code[J]. Software Testing, Verification and Reliability, 2001, 11(2):65-79

[68] Godefroid P. VeriSoft: A tool for the automatic analysis of concurrent reactive software [C]// Computer Aided Verification. 1997:476-479

[69] Telelogic T A U. A SDL tools for real time systems development[OL]. http://www.telelogic.com

[70] Cleavel R, Parrow J, Steffen B. The concurrency workbench: A semantics based verification tool for the verification of concurrent systems[J]. ACM Trans. on Programming Languages and Systems, 1993, 5(1):36-72

[71] 傅海仑. 定理机器证明思想的产生与发展[J]. 科技导报, 2001: 14-18

[72] 林作铨. 一个模态非单调逻辑[J]. 中国科学 E 辑, 1996, 26(3): 276-288

(上接第 49 页)

```
static void Main(string[] args)
{
  //必须使用微软企业库的功能模块来创建接口
  ICustomerService iCustomerService = PolicyInjection.Create
  <CustomerService, ICustomerService>();
  //通过接口调用方法
  string strName=iCustomerService.GetCustomerName(12);
  //如果不存在,则说明没有权限
  if(strName== null)
  {
    Console.Write("您没有权限查看此客户的姓名");
  }
  else//姓名存在则显示出来
  {
    Console.Write(strName);
  }
  }
}
```

客户端调用 GetCustomerName 方法时,会先调用方法上的特性类 JudgePrivilegeHandler 中的 CreateHandler 方法,而这也就是间接调用了 AOP 类中的 Invoke 方法,从而可以调用嵌入接口类通过权限系统判断该方法的权限。

结束语 本文采用领域驱动设计思想,结合 RBAC 模型,针对 MIS 系统,研究了基于领域驱动的权限模型的构建、实现以及实施权限控制的方法。本文所采取的方法可以较好地解决业务和权限无法分开以及系统权限控制粒度较粗等权限系统面对的难解问题,具备较强的通用性,在实际 MIS 系统开发和应用中,也具有较高的应用价值。当前该方法已应用于广西南宁市某事业单位的内部办公自动化系统开发中,取得了良好的应用效果。

参 考 文 献

[1] 王兴伟,王宇. Web 信息系统中基于 RBAC 模型的访问控制模块设计与实现[J]. 大连理工大学学报, 2005, 45(S1):284-286

[2] 范明虎,樊红,伍孝. ASP. NET 中基于 RBAC 的通用权限管理系统[J]. 计算机工程, 2010, 36(1):143-145

[3] 韩道军,高洁,翟浩良,等. 访问控制模型研究进展[J]. 计算机科学, 2010, 37(11):29-33

[4] What is Domain Driven Design? [EB/OL]. http://devlicio.us/blogs/casey/archive/2011/05/16/what-is-domain-driven-design.aspx

[5] 王忠,程磊. 基于领域驱动设计的软件开发[J]. 软件导刊, 2008, 7(2):37-39

[6] Evans E. Domain-Driven Design-Tacking Complexity in the Heart of Software [M]. Addison-Wesley Professional, 2004

[7] ASP. NET MVC 下基于 RBAC 权限认证的设计与实现[J]. 重庆理工大学学报:自然科学版, 2011, 9:75-80