计算机科学 1998 Vol. 25№. 2

形式化方法的发展及展望*)

The Development and Prospect of Formal Methods

姜 利[1,2] 孙永强[2]

(丹东师范高等专科学校计算机系 丹东 118003)[1] (上海交通大学计算机系)[2]

摘 要 The paper is contributed to give the definition and classification of formal methods after simply looking back the history of formal methods. Based on the analysis of the history, the current development and applications of formal methods, the paper puts forward the trends and application prospect of formal methods.

关键词 Formal methods, Software engineering, Real-time system, Formal specification, Formal notations, Object

一、前言

形式化方法的研究和应用已有近 30 年的历史了,其产生是 Dijkstra 和 Hoare 在程序验证方面的工作和 Scott, Stratchey 以及其他学者在程序语义方面的工作基础上发展起来的,从最简单的形式化方法,即用一阶逻辑和等式组成的规范语言,至 80 年代较为复杂的 Z 语言,直至今天具有代表性的形式规范语言——面向实时及分布式的 LOTOS 语言和面向对象的 Z^{++} 语言等等。至今为止,形式化方法已经形成了较为完整的体系。

形式化方法对软件工程的作用应与应用数学对其它工程领域的作用相同,这一点已取得了世界范围内绝大部分学者的共识,形式规范在许多包括重要系统在内的领域内,已被认为是有重要价值的方法。

关于形式化方法的定义颇多,虽然说法各异,但其基本内涵是相同的。所谓形式化方法是指"基于形式系统的开发方法"[1],这一定义是概括性的,文[2]的定义给出了形式化方法的实质,即形式化方法是由一形式系统和语用组成,一个形式系统是二元组 $\langle L, C_n\rangle$,其中,L 是一种语言,它是由符号和描述在该语言中如何写句子的语法组成的(或称为一种形式记法),C_n 是对应的关系,即在 L 中字符串的映射,它是由推理规则组成的。对于给定的语言 L,其结构 $\langle U, I\rangle$ 是由全域 U(universe)和相应的解释 I 组成的。全域 U 是值的集合(如整数、实数和布尔值)。解释 I 是语言 L 到全域 U 的映射。从本质上说,一个形式系统的结构给出的是用 L 写成的语句的含义,即语义。在这样一种定义下,形式化的描述技术与编程语言是十分相似的,但是,与编程语言相比,形式化的描述技术不一定会有相应的操作语义。

根据上面的定义,形式化方法可以理解为在系统的开发过程中,使用一种形式系统(记法)来表示系统模型的方法。而根据 IEEE 的定义,形式规范是使用形式系统(记法)写成的、用于正确性验证的规范。在这样的定义下,可以看出,形式化方法主要体现为形式规范。因此,在本文的讨论中,在上下文清楚的前提下,我们对这两个概念不作区分。

形式化方法从其形式化的程度来分,可分成形式化方法和半形式化方法两种。从概念上讲,形式化方法是一种准确的方法,它是以数学的方式定义的,与非形式化方法有明显的区别,半形式化方法,即在该方法中,对软件的描述中有一部分不是用准确的、数学的方式定义的。在许多情况下,半形式化方法严格地定义了形式记法的句法,但却没有给出形式记法的语义定义。

形式记法从其表现形式上可分成两类:文本方式和图形方式。比如,用于实时系统的 SA/RT[3]是典型的基于文本方式的形式化的方法,文[4]使用的是状态图的方法。而在文[5]中,使用了一种新的基

*)得到国家教委职教项目资助。

于图形的 Modechart 方法，与用文本记法(比如，用基于逻辑的方法)来表达的规范相比，用这种方法产生的规范，其可读性更好，更加易于理解。由于这种新的图形记法具有明显的形式语义，与其它一些形式化和半形式化方法相比，更易于进行形式分析。对应于形式记法的分类，对应的形式模型也分成两类，文本形式模型和图形形式模型。这些形式模型分成四个主要的类别：基于逻辑的模型，基于状态机的模型，基于扩展的进程代数的模型以及混合的模型。基于逻辑的模型以及基于扩展的进程代数的模型常常是用文本方式表示的，基于状态机的模型是以图形记法来表示的。

早期的形式化方法主要是局限于程序的形式化验证，而早期的代数规范的工作是由 Guttag 于 1975 年，在他的博士论文中完成的，在早期的规范语言中，两种典型的有代表性的，由工具支持的语言是 AFFIRM 语言和 OBJ 语言，这两种语言的主要功能也是进行程序验证。

八十年代产生的形式语言中，较有代表性的有 Z 语言和 VDM 语言。Z 和 VDM 都是基于模型的形式规范语言，它们使用集合论和逻辑方法来构造所需系统的抽象模型。Z 语言作为形式化方法是使用较为成功而有效的语言，最初是由 J. R. Abrial 发明的。Z 语言的应用范围很广，包括许多 CASE 工具。作为 Z 语言的一个较为复杂的应用，它也被用于说明 CICS 系统及其工具。八十年代以后，Z 语言得到了相应的扩充，并且在工业界得到了较为广泛的应用。VDM 方法是一种基于指称语义的技术，它不仅能被用于规范，而且能被用于设计和对应的实现中。Z 语言和 VDM 语言有一个共同的弱点是较难确定该语言描述的软件总体结构和范围。

COLD-K 作为一个宽域的规范语言，是在荷兰爱因和文市的 Philips 研究实验室开发出来的，设计 COLD-K 的宗旨是想在整个软件开发过程中的各个阶段使用该语言，包括在实现阶段亦使用它。该语言用数学语言定义了相应的句法和语义。它既可以用于描述程序，也可用于描述规范。

在八十年代末期，形式化方法发展中的另一个有代表性的方向是对代数方法和软件技术结合的(AMAST)研究[6]，Hartmut Ehrig 是用代数方法研究软件技术的先驱，他领导研究并创造了 ACT 方法，开发了相应的使用环境。ACT 方法使用具有约束和松散语义的代数规范来描述功能需求的规范，在设计阶段，使用具有初始语义的代数规范。并且，允许利用 ACT 的环境来执行规范和证明某一性质。ACT 促进了相应的软件工具的发展，用户使用这些软件工具可以有效地从数学角度说明他们的对象，在 LOTOS 中引入 ACT ONE(ACT 的一种扩展)形成的 LOTOSPHERE 成为描述并发和分布式系统的一种较为有效的形式语言。此外，使用代数方法的规范语言还有一些较有影响的语言，如 PLUSS 和 FOOPS 等。

二、形式化方法的理论基础

逻辑理论、软件结构理论、集合论、代数理论构成了形式化方法的基础。逻辑理论提供了一些方法用于规范的推理和程序的执行，而软件结构理论提供了一个计算模型，该模型描述了所开发系统的一些重要方面，如软件结构及其组成成份等。由于设计的计算模型定义了被推理的成份(即模块)、状态机和过程的形式上的对应模型，因而它与逻辑理论是同样重要的。近几年来，逻辑理论有了较大的发展，特别是时序逻辑，高阶逻辑在一定程度的发展，为形式化方法的发展和应用提供了更加有效的理论基础。其它理论的发展，如代数规范，范畴论，进程代数等理论的发展也促进了形式化方法的发展。形式化方法与这些技术及其它技术的结合，形成了应用于不同领域的不同的形式化方法。这些方法主要有：

(1)基于代数的方法。该方法又分成三种，即基于文本符号方法，基于图形的方法和前两种方法混合的方法。

(2)基于逻辑的方法。该方法又可分为：基于 a. 时序逻辑的方法；b. 实时区间逻辑的方法；c. 直觉命题逻辑方法；d. 高阶逻辑的方法。

(3)基于操作语言的方法。该方法又可分为：基于 a. 抽象机方法；b. 杂合自动机的方法；c. 时序自动机方法；d. Petri 网的方法；e. 扩展的 Z 语言的方法；f. 扩展的 VDM 的方法；g. 时间扩延的(time-extended)LOTOS 方法。

三、形式化方法的最新发展

3.1 面向对象的形式化方法的发展

将面向对象技术与形式化技术结合起来已成为软件工程研究较热的一个领域，同时也吸引了世界

许多国家学者及公司的兴趣，因为面向对象技术与形式化方法具有较高的相容性，这主要表现在：①面向对象技术使用抽象的数据类型，而形式化方法提供了一些方法来准确地描述这样的抽象。②面向对象技术提供了结构化的将形式化技术应用于大型系统的机制和开发的规则。③对复杂的面向对象的机制，比如，对象的聚集或继承，形式化方法技术允许给出准确的定义（但比较复杂）。目前，在这个领域内有两个主要的研究分支：一是使用面向对象的结构来提高形式记法的形式表达能力，二是使用形式方法来分析面向对象的语义或提高这些记法的表达对象概念的能力。

最初的面向对象结构规范语言的主要缺点是缺乏完整的形式语义的定义，因而面向对象结构规范语言复杂推理的程度较低。直到 Object-Z 和 Z^{++}出现以后，面向对象的规范语言才有了相应的形式语义描述。Z^{++}语言起源于欧洲的 ESPRIT 项目中的一个子项目 REDO。Z^{++}语言从 1989 年产生以后，得到了不同程度的扩展，比如，引进对象引用的语义，引进实时逻辑等。对该语言的理论研究主要集中于获得所需特征的简洁而完善的语义定义，其中一个重要的方向为将该语言与 OMT 等现存的图形化方法相结合。现在，Z^{++}语言已被应用到许多的领域，如人工智能，逆向工程和交互系统等，其中，以静态的分析工具的设计及面向对象程序规范设计上的应用最多。

VDM^{++}是 VDM-SL 语言的面向对象的扩展，它对 VDM 的扩展主要包括：类说明和类型定义，它用规范语句或杂合（各种方法混合）的方法定义取代了 VDM 操作的定义，同时也增加了用于定义动态和并行行为的机制。该方法在欧洲一些项目上获得了一定程度的应用。

全炳哲和金淳兆在面向对象规范（原文用规格）语言的研究中，提出了一种面向对象软件的形式描述语言 JOOSL[7]，使用该语言可以描述面向对象软件需求规范，概要设计和详细设计。JOOSL 是以 COLD-K 为基础，吸收 Z 和 Recos 语言的语法格式的一种面向对象的软件描述语言，其主要目标是研究面向对象程序的自动化技术。

面向对象的形式化方法存在的主要的问题是，该语言过于复杂且难以对如下的概念给出准确的说明：①对象聚集；②对象的多态和动态联编；③对象的多重继承；④对象的主从关系（clientship）。

3.2 实时系统中形式化技术的发展及应用

在工业上的应用中，形式化方法主要应用在：①对需求规范的描述上；②对系统预期属性的数学分析上。在这些系统中所使用的一些主要方法，其中，三个较有代表性的使用实时逻辑进行推理的形式化方法有 RTL，TRIO 和 ASTRAL[8]。作为描述实时系统的、基于一阶逻辑的语言，TRIO 支持形式意义上清楚的规范的构造，但由于 TRIO 没有支持说明复杂系统的设施，因而该规范的应用受到限制。$TRIO^{+}$作为 TRIO 扩展的一种语言，融进了面向对象的概念及结构，扩大了 TRIO 的应用范围。ASTRAL 是基于逻辑的语言，它是一个将具有时序特点的 TRIO 和一个描述非实时系统的语言 ASLAN 结合起来的语言，ASTRAL 以操作的方式，通过定义系统的状态及其转换的方式来描述一个系统，它以演绎的方法，通过对规范的形式分析的方式来实现程序的证明。

LOTOS（Language of Temporal Ordinary Specification）是进程代数家族中的一种形式规范语言，这种语言结合进了两种成份，一个成份用于说明事件的时序序列，一个成份用于描述数据结构，该成份用抽象的数据类型语言 ACT-ONE 组成。LOTOS 语言首先应用于描述 ISO-OSI（开放系统内部互连的）通讯网络结构的协议和相应的服务。现在已扩展到了许多其它的领域。如实时系统以及并发和分布式系统等等。利用 LOTOS 语言构造的规范有三种不同形式的风格，即面向约束的，面向资源的和面向状态的。

3.3 混合的形式规范技术

在软件开发中，至目前为止，一些被广泛使用而又十分成功的方法是将形式化方法与结构化方法结合起来或将形式化方法与其它语言或方法结合起来的方法。比如将结构化方法 SSADM，Yourdon 或 OMT 与形式化方法结合起来的方法[9]，作为一种开发方法，这种结合有一些优点：可以使用这两种技术各自的优势，也可以使用现存的软件工程的专业知识。最近的西方调查结果显示，在工业界使用的形式化方法的公司中，大约有 31%的公司是结合结构化的方法来使用形式化方法的。相对地，Fusion 和 SAZ 方法是一种混合的方法，其目的是在软件生命周期的几个阶段支持混合使用文本和图形记法。

SAZ 是 Z 和 SSADM 混合使用的方法。

3.4 形式规范语言的支持工具及形式规范语言的执行

众所周知，无论是用形式的或非形式的方式验证一个规范是困难的。目前，已存在一些形式化的技术用于验证规范或程序，这种验证包括规范的浏览、证明和执行等。在规范的执行方面，Knott，Krance 和 Dick 等[10]的工作是一个有前途的方法。为了模拟形式规范的执行，他们研究了将 Z 语言自动翻译成形式规范的方法，并成功地实现了许多规范的翻译和执行。此外，Johnson 和 Sanders[11]也将几个形式规范的实例翻译成类 Miranda 语言表示的形式，虽然该翻译的过程不是自动化的，但其基本思想却是有启发性的，而且该方法能提供许多潜在的功能。未来的工作是要开发或用文本存储的方式存储该方法，以期产生可使用的支持工具。此外，作为直接可执行规范的代表，用 TRIO 语言描述的规范也是可执行的规范，它是通过一个特殊的解释器来实现的。规范语言 ACT 也是一种可执行的规范语言。

直接利用规范语言进行程序设计，并直接执行规范的另一个有价值的工作，是文[12]的工作。为了能有效地开发正确的程序，该文的作者们研究了基于重写方法的程序开发系统的设计和实现，这个系统使用代数规范说明语言和扩展的函数式语言合成而形成的混合语言进行程序设计。系统将代数规范转换为合流的重写系统，从而实现直接执行规范的操作。在这一领域内，作者认为未来的有前景的工作是如何扩展该理论的应用范围，或利用这种混合语言作为一种软件规范和某种程序语言的中间语言，以期真正实现软件工程的自动化。

3.5 形式化方法其它一些应用

近几年，形式化方法的应用范围越来越广，主要在以下几个方向上有较为成功的应用：(1)用于硬件的设计和软件的开发。其中最令人信服的和有影响的是使用形式化方法进行完整的硬件验证设计，该设计的结果是一个计算逻辑的 FM9001 的微处理器，在该处理器中，使用了 Boyer-Moore 的定理证明器，在逻辑门一级的网表(网表是描述一些部件的逻辑门和这些门电路相互作用的一个表)上进行验证。(2)在人工智能上以及在计算机辅助制造上的应用[13]。(3)由于形式化方法具有准确的特点，因此在工程的标准化的定义中可以使用形式化方法。

四、存在的问题及其展望

尽管，形式化方法在软件工程及其相关的计算机领域已取得了成功，但与面向对象技术和 GUI (Graphic User Interface)技术相比，社会，特别是商业系统对形式化方法的兴趣仍然不是很大(个别领域除外)，使用具有较强数学特点的规范语言，比如对 Z 和 VDM 的研究和使用，仍然在很大程度上局限于大学、科研机构以及对安全性要求极高的工业。对形式化方法存在的问题，文[14]等皆有论述。综合他们的工作，我们认为，目前，形式化方法存在的主要问题是：①由于形式化方法是以计算逻辑、代数理论和软件结构为基础的，因而形式化方法从本质上是一种较为严格的、灵活性较差的方法。至今为止，由于尚没有对形式化方法提供完全自动化的、用户接口良好的、易学，易对功能进行扩展的完善的支持工具，所以形式化方法仍无法为大多数系统设计者和程序设计者接受。②现在最流行的许多形式化方法在解决较小的规模方面是有效的，但却无法或很难应用于较大一些的实例中。③实际规范和实际代码间的差异仍然很大。尽管人们在类型理论、最弱前提条件、程序变换和程序设计等方面已做出了长期而艰辛的努力，但是程序编码在很大的程度上仍然是手工完成的。④由于形式化方法本身的特点，形式化方法与软件开发过程较难平滑地结合起来。⑤尚没有一种形式化方法能对软件工程生命周期的各个阶段提供全面的支持。

近年来，形式化方法正以前所未有的速度发展着，除了在小型和中型的项目中大量使用形式化方法之外，形式化方法在大系统开发中的应用也不乏其例，始于 1994 年秋的“形式化方法应用研究的竞赛”[15]，再一次显示了形式化方法的应用价值。在这种设计中，形式化方法应用的难度远远高于纯学术范围的研究范例，各国从事形式化方法研究的学者，再一次在大型的工业实践中，验证了形式化方法的巨大优势及其潜在的发展前景。

综上所述，我们认为未来形式化方法的研究应在以下几个方面进行：①在充分研究现有的形式化方法的基础上，建立一个设计形式化方法的基本规则或标准。②对现有的、在应用上较为成功的形式化方法，积极推进它们的标准化工作。③对实践已证明是成功的形式化方法或对已标准化的形式化方法，

积极研究具有良好用户界面的、容易学习和操作简单的形式化方法的支持工具。④鉴于形式规范与实际代码的差异，借鉴软件重用的基本思想，作者认为，我们需要研究一种介于规范语言和程序语言之间的语言，这种中间语言主要完成在程序部件库中，寻找相应的部件，并完成部件返送的工作。对此进行的研究尚刚刚开始。

主要参考文献

[1] Constance Heitmeyer et al. ,Formal methods for Real-Time Computing,John,Wiley & Sons Ltd

[2] Dan Craigen et al. ,Formal Methods for Trustworthy Computer Systems,Springer-Verlag

[3] P. T Ward and S. J. Mellor,Structured Development for Real-Time System,Yourdon Press,1985

[4] D. Harel,Statecharts:A Visual Formalism For Complex Systems,Science of Computer Programming,8 (1)1987

[5] F. Jahanian et al. ,A Specification Language For Real-time Systems,IEEE Trans. on Software,Eng,20 (10)1994

[6] Ingo Claben et al. ,Algebraic Specification Techniques and Tools for Software Development:ACT Approach,1993

[7] 全炳哲、余江、金淳兆，可重用构件及其描述语言，软件学报,5(1),1994

[8] A. Coen-Porisini et al. ,A Formal Method for ASTRAL Intralevel Proof Obligations,IEEE Tran. On Software Eng. ,20(8)

[9] L. Semmens et al. ,Integrated Structured Analysis and Formal Specification Techniques,The Computer Journal,35(6)1992

[10] Dick A. J. J. et al. ,Computer Aided Transformation of Z into Prolog, Proc. 4th Z Users Meeting, Springer-Verlag,1989

[11] Johnson M. and Sanders P. ,From Z Specification to Functional Implementations,Same to[10]

[12] 林凯、孙永强、陆朝俊，基于重写方法的程序开发系统的设计和实现，计算机学报,19(9)1996

[13] Mandayam Srivas and Albert Camilleri(Eds.),Formal Methods in Computer-Aided Design,FMCAD' 96,Nov. 1996,Proc. LNCS 1166

[14] Authony Hall,Seven Myths of Formal Methods, IEEE Software,Sep. 1990

[15] Jean-Raymond Abrial et al. ,(Eds.),Formal Methods for Industrial Applications,LNCS,Springer

---

(上接第 54 页)

参考文献

[1] G. Booch, Object-Oriented Development, IEEE Trans. on Soft. Engi. SE-1(2)1986

[2] G. Booch,Object-Oriented analysis and design. ,The C++ Report,3(8)1991

[3] G. Booch,Object-Oriented Analysis and Design with Applications,Benjamin/Cummings,1994

[4] G. Booch et al. ,The Unified Modeling Language, http://www.rational.com/uml

[5] Peter Coad and Edward Yourdon,Object-Oriented Analysis,Yourdon Press,1990

[6] Peter Coad and Edware Yourdon,Object-Oriented Design,Yourdon Press,1991

[7] I. Jacobson et al. ,Object-Oriented Software Engineering:A Use Case Driven Approach, Addison-Wesley,1992

[8] J. Rumbaugh et al. ,Object-Oriented Modeling and Design,Prentice Hall,1991

[9] S. Shlaer and S. J. Mellor,Object-Oriented Systems Analysis:Modeling the World in Data,Englewood Cliffs,NJ:Yourdon Press,1988

[10] S. Shlaer and S. J. Mellor,Object Lifecycles:Modeling the World in States,Englewood Cliffs,NJ:Yourdon Press,1992

[11] Daniel Tkach et al. ,Visual Modeling Technique:Object Technology Using Visual Programming,Addison-Wesley,1996

[12] R. Wirfs-Brock et al. ,Designing Object-Oriented Software,Prentice Hall,1990