ZHICHI FANXING CHENGXU SHEJI DE
Apla-Java ZIDONG CHENGXU
ZHUANHUAN XITONG

# 支持泛型程序设计的 Apla-Java 自动程序转换系统

石海鹤 著

江西高校出版社
JIANGXI UNIVERSITIES AND COLLEGES PRESS

国家自然科学基金课题：《模型驱动的高可靠图算法构件自动生成研究》
授权号：61363013

# 支持泛型程序设计的 Apla-Java 自动程序转换系统

ZHICHI FANXING CHENGXU SHEJI DE Apla-Java ZIDONG CHENGXU ZHUANHUAN XITONG

石海鹤 著

江西高校出版社
JIANGXI UNIVERSITIES AND COLLEGES PRESS

**图书在版编目(CIP)数据**

支持泛型程序设计的 Apla－Java 自动程序转换系统/石海鹤著. —南昌: 江西高校出版社,2014.3
ISBN 978－7－5493－2423－1

Ⅰ.①支... Ⅱ.①石... Ⅲ.①JAVA 语言－程序设计－高等学校－教材 Ⅳ.①TP312

中国版本图书馆 CIP 数据核字(2014)第 048951 号

| | |
|---|---|
| 出版发行 | 江西高校出版社 |
| 社　　址 | 江西省南昌市洪都北大道 96 号 |
| 邮政编码 | 330046 |
| 总编室电话 | (0791) 88504319 |
| 销售电话 | (0791) 88513417 |
| 网　　址 | www.juacp.com |
| 印　　刷 | 天津市天办行通数码印刷有限公司 |
| 照　　排 | 江西太元科技有限公司照排部 |
| 经　　销 | 各地新华书店 |
| 开　　本 | 890mm×1240mm　1/32 |
| 印　　张 | 2.625 |
| 字　　数 | 86 千字 |
| 版　　次 | 2014 年 3 月第 1 版第 1 次印刷 |
| 书　　号 | ISBN 978－7－5493－2423－1 |
| 定　　价 | 36.00 元 |

赣版权登字－07－2014－114

# 目录

CONTENTS

# 引 言

## 一、研究背景

随着软件系统复杂度的不断增长，高效率地开发正确、可靠的软件，已成为一个亟待解决的问题。形式化方法是解决此问题的一个有前途、有希望的技术。特别是开发要求高可靠性的实时控制软件和军用软件，形式化方法的重要性和迫切性就更为明显。它建立在严格的数学基础上，其目标是希望能使系统具有较高的可靠性和正确性，并能使系统具有良好的结构，使其易维护，能较好地满足用户需求[34]。

在过去的20多年里，虽然形式化方法在研究和应用上都取得了很大的进展，也越来越为研究者和软件开发者感兴趣，但它并没有广泛地被软件产业界所接受，在许多方面也还需要更多的研究和努力：目前的许多形式化方法的倡导者并没有真正研究软件开发者所面临的实际问题；有些方法以及在这些方法基础上建立的软件开发工具和自动生成系统只能处理一些简单的问题；人们对使用形式化方法开发大型软件系统和解决特殊问题的适用性及其使用程度等问题还不很清楚[33]。当务之急是研究一种面向实际问题、简单可行、便于广大软件开发者接受的形式化方法；加强形式化方法辅助工具的易用性和集成性，同时和其他工具结合形成一个集成的开发环境，使工业界更愿意接受形式化方法。在将来，形式化方法可能会融入到信息技术的各个领域并成为一种主流的方法和技术[35]。

PAR[7]是一种简单实用的软件形式化方法。这种方法结合形式化方法和非形式化方法，正确区分开发过程中的创造性劳动和非创造性劳动。在该方法的指导下，定义了Radl(Recurrence - based Algorithm Design Language)算法设计语言[64]来描述算法规约和抽象算

法，定义了 Apla( Abstract Programming Language) [65] 语言来描述抽象程序。

根据 PAR 方法，由递推关系到循环不变式，及由算法到程序的开发，属非创造性劳动，完全可以用完全形式化方法机械地变换得到，也就是可以构造一个自动变换工具由计算机完成[19]。这个工具就是自动程序转换系统。

自动程序转换系统是整个 PAR 方法理论框架的重要组成部分，是用 PAR 方法开发算法程序的辅助工具[29]。这里的自动程序转换系统，是指将一个正确的 Apla 程序自动转换成与之等价、并可运行得到正确结果的某种高级语言程序的系统。本研究选取新型程序设计语言 Java 作为自动程序转换系统的目标语言，在课题组前期研究开发的基础上，研究和实现了支持泛型程序设计的 Apla→Java 自动程序转换系统。

目前一些形式化方法的支持工具，只能支持规约的编辑、求精或手工证明等，而本系统在以下方面体现出其特色和创新:

1. 它是 PAR 方法的重要组成部分。其算法用 Radl 语言描述，Apla 程序在 Radl 算法的基础上得到。Radl 具有引用透明性，使算法和程序相分离，有利于算法和程序的推导，从而可保证作为系统输入的 Apla 程序的正确性。

2. 允许用户在 Apla 程序中使用简单的数学符号。

3. 强迫用户使用抽象数据类型设计程序，使其注意力集中于算法的推导。

4. 在部件库的支持下，用户使用组合数据类型就像使用简单数据类型一样简单方便。

5. 集 Apla 程序的编辑、编译、运行于一体，使用方便。

6. 为我们正在研制的程序设计智能教学软件提供了支撑。

7. 能提高软件的可靠性和软件的开发效率，可以为克服“软件危机”作出有益的贡献。

本书的出版受到以下课题的资助:

国家自然科学基金课题《模型驱动的高可靠图算法构件自动生成研究》，授权号: 61363013

## 二、本书的主要工作与组织

### (一) 本书的主要工作

本研究主要进行了以下工作:

1. 用 PAR 方法推导了大量的 Apla 程序,作为输入对原有自动转换系统进行了大量的测试并作出测试报告。

2. 对整个自定义 Java 可重用部件库的结构进行了重构和优化,使得整个库的结构变得清晰,使用方便;库代码易于理解,可读性好。在此基础上,对部件库代码进行了完善,对部件库的内容进行了扩充,使其能够更好、更自然的支持高级功能的实现,同时保持了对已实现功能的支持。

3. 对转换器进行了重构,使得整个转换器代码结构清晰,而且还暴露出了一些代码处理不善的地方。对转换时使用的转换规则进行了优化。在此基础上,对转换系统的原有转换功能进行了完善,使得转换器的语法检测功能更加强大,转换性能得到提高;并为转换系统添加了一些新的特性,使得用户使用起来更加方便,增强了其可用性。

4. 进一步完善了自动程序转换系统泛型问题中的类型参数化问题,实现了泛型子程序的成功转换,添加了特殊符号直接作子程序参数名等功能。

5. 研究了 Apla 中支持的泛型程序通过现有 Java 机制实现的途径,并最终在此自动程序转换系统中实现了对泛型程序转换的支持。

6. 研究了 Apla 中用户自定义泛型 ADT 机制的 Java 实现并在本系统中实现了自定义泛型 ADT 的转换。

### (二) 本书的组织

本书的组织如下:

第一章绪论,介绍形式化方法、软件开发的形式化方法及其研究意义以及国内外相关内容的研究现状和存在的问题。

第二章 Apla→Java 程序转换系统介绍,给出了系统的结构并说明了各功能模块的含义;对系统已有的转换功能作了简单说明;介绍了系统的运行环境和使用;对系统的不足进行分析并给出了系统的

有待完善之处。

第三章对原有系统进行了重构和优化，包括对可重用 Java 部件库和转换器两部分的工作；并在重构及优化的基础上，对系统的有待完善之处作了改进，给系统添加了新的特性。

第四章介绍了 Apla 语言中的泛型程序设计，并基于 Java 中实现泛型程序设计的方法给出了它在 Apla→Java 自动程序转换系统中的实现。

第五章结束语，对整个工作做了总结，给出了进一步的工作，并对本研究的前景作了展望。

# 第一章　形式化软件开发方法概述

## 1.1　形式化方法

形式化方法的研究早在40年代就开始了，是公理方法发展的高级形态。它把逻辑推理过程完全量化了，完全抽掉了语句的具体内容而代之以符号“运算”来完成其证明。形式公理化方法具有高度的形式化和抽象化，其中的基本概念、基本关系的表达、全部命题的陈述、证明都要符号化，具体地讲，它的对象、基本关系不仅用抽象的符号表示，而且将命题表示成由符号组成的公式，命题的证明用一个公式串来表达。它主要采用数理逻辑作为它的演绎推理工具[19]。

## 1.2　形式化软件开发方法及其研究意义

目前软件开发的主要方法是自顶向下、逐步求精法。它只适用于逻辑关系简单的问题，对那些逻辑关系复杂的子目标(模块或算法)往往难以奏效。因此需要研究新的算法程序开发方法解决这一问题。

### 1.2.1　形式化软件开发方法

按传统软件开发方法设计与开发的软件产品，尤其是要求高可靠性的实时控制软件和军用软件(包括顺序、并行和实时软件)等大型软件系统，普遍存在着系统正确性及可靠性难以保证等问题，且难于实现软件自动化。八十年代以来，形式化软件开发方法的研究及应用为解决上述问题找到了一条有效的途径，使用形式化方法开发软件可以提高软件的可读性、可靠性和可维护性以及软件的开发效

率，并为实现软件开发的自动化奠定基础。

软件开发的形式化方法是以一般形式化方法为基础的。形式化方法为计算机系统的规约、实现和验证提供了合理的数学基础，首先对系统进行独立于实现的、基于一定形式化语义的抽象描述，然后经过逐步求精及变换，利用规约的数学特性检验其一致性，直至生成最终的软件系统。其优点在于它严格、精确地定义了用户需求，形式化规约的求精过程具有可推导、易证明的特性，从而有利于保证程序的正确性和实现软件自动化。尤其适宜于高安全性系统的开发，这也是形式化方法目前最主要的应用领域。但是，形式化方法要求使用者具有较好的数学背景，因而不便于最终用户参与开发过程，阻碍了它的进一步发展。另外，某些形式化方法缺乏描述软件结构的强有力机制，对大型软件的开发不太理想。

一个形式化的软件开发方法一般包括一套思维方法及描述方法、一种开发手段（如规范描述的原则、程序开发的一般过程、描述语言等）和支持工具，使开发者能利用数学概念和表示方法恰当合理地构造形式规范，根据开发过程的框架及设计原则进行规范描述和系统化的设计求精，并使用证明的概念对规范的性质和设计步骤进行分析和验证；方法还应该有工具的支持，使开发过程可行且高效。

### 1.2.2 形式化软件开发方法的分类

形式化方法按其形式化程度来分，主要有两种：完全形式化方法和部分形式化方法。

1. 完全形式化方法

对于一个给定的算法程序设计问题，先用符号化的规范描述语言，写出这个问题的形式化规范，然后采用变换方法，将问题的形式规范变换成可执行的程序。变换的开始、中间到结束、产物是一符号串，这种形式化方法称为完全形式化方法。使用完全形式化方法产生程序的过程可以用计算机机械地产生。但目前用这方法只能产生没有创造性劳动的简单程序。

2. 部分形式化方法

对于给定算法程序设计问题，先写出该问题的形式化规范，然后

使用形式化和非形式化相结合的方法，开发或证明算法程序正确，这种方法常称为部分形式化方法或数学家的形式化方法。使用这种类型的形式化方法常常可以设计和证明一类相当复杂的算法程序和软件。目前在各类数学和程序设计方法学专著和教材中使用的方法就是属于部分形式化方法。使用这种方法进行软件或算法程序设计或证明时，非形式化方法不能够由计算机机械地进行。

由前面的叙述可知，软件开发的形式化方法作为一般形式化方法的特例，也存在其固有局限性，因此软件开发过程不可能完全形式化。所以，目前软件开发的形式化方法主要是以部分形式化方法为主。

总体上，形式化软件开发方法大致可分为以下五类[32]：

第一，基于模型的方法。给出系统(程序)状态和状态变换操作的显式但亦是抽象的定义，对于并发没有显示的表示。Z 和 VDM 即属于该类方法。

第二，代数方法。通过联系不同操作间的行为关系而给出操作的隐式定义，而不是定义状态。同样，它亦未给出并发的显式表示。

第三，过程代数方法。给出并发过程的一个显式模型，并通过过程间允许的可观察的通讯上的限制(约束)来表示行为。

第四，基于逻辑的方法。有很多方法采用逻辑来描述系统的特性，包括程序行为的低级规范和系统时间行为的规范。

第五，基于网络的方法。根据网络中的数据流显式地给出系统的并发模型，包括数据在网中从一个节点流向另一个节点的条件。

## 1.3 国内外研究现状及存在的问题

本节讨论目前国内外典型的形式化软件开发方法(内容、原理、特色、局限性)及其支持工具等方面的研究情况。

### 1.3.1 VDM 方法(Cliff Jones)

VDM(The Vienna Development Method 维也纳开发方法)[59]是一种使用较早、应用较广的基于模型的形式化开发方法，也是当前较为成熟的形式化开发方法之一。七十年代初由 IBM 公司的 C. B. Jones

等在维也纳开发出其早期形式。

VDM 包含一个称为“VDM - SL”( VDM Specification Language) 的规约语言; 一组用于在抽象的需求规约和代码级的详细设计规约间建立链接的数据求精和操作求精规则; 一套证明理论,其中可用严格的参数来操作指定系统的属性和保证设计决策的正确性。

VDM 方法提供了程序正确性证明的依据,使规格说明描述简练、精确。但 VDM 也存在以下不足之处: VDM 的每一个求精步骤都是上一个求精步骤的具体化,由于各个求精步骤之间并不存在着严格的推导关系,因此整个的算法推导过程不够严谨; 其正确性是在算法推导完成后再回头验证其每一求精步骤来保证的,若有错误又得重新推导算法,然后再对推导过程进行重新验证; 同时 VDM 的每一求精步骤都包含“数据精化”和“操作分解”两个不同的过程,将数据结构和算法分离,这些都导致使用 VDM 来开发算法效率不够高。

尽管 VDM 的规约语言已由 ISO 标准化并得到很多成熟的工具的支持,但目前仍没有一个满意的工具来支持验证。

1.3.2 Z 方法( Spivey)

Z 方法[30] [59] 是以程序公理语义为基础的形式化开发方法,它的规范说明语言基于模式演算,提供了丰富的操作运算,但不可执行。Z 语言有模式匹配的符号,使用了具有强大功能的操作符,还使用了一些非形式化的英语解释,给出的规范说明比较短,易于阅读和理解。

Z 语言已经成为世界最流行的形式化规约语言。现有大量的工具支持,它们主要集中于规约的编辑和显示以及语法和类型检查方面,也有些工具支持规约的推理和精化。比较有影响的工具有: Formaliser、ProofPower、Zola、CADiZ、ForMooZ、Z/EVES 等。

Z 语言存在以下一些缺陷:

①Z 语言对大型系统的模块化能力不足。

②没有提供重用机制。

③Z 语言难以用计算机直接处理。

④不支持软件开发的全过程。

### 1.3.3 RAISE 方法(Nielsen)

RAISE 方法(Rigorous Approach to Industrial Software Engineering)是20年来对逐步求精思想的研究和实际应用的结果。RAISE方法覆盖了软件开发的关键阶段,如需求分析和项目管理。它的目标是开发出更加可靠的、更少错误的、文档齐、易维护的软件系统来。

该方法以逐步求精的思想为基础,整个开发过程由若干步骤构成,而每一步都是在"invent - and - verify"的思想指导下进行,如:规范到设计是手工进行,它们之间的关系按事后验证法(posterior verification)去验证。RAISE方法支持theory extension(see question on theory extension)逐步求精法。

RAISE的形式规范语言是RSL,它是一种强大的说明和设计语言。并且RAISE方法还提供了一定的支持工具,支持规范、设计、实现等阶段,使得形式化方法得以应用于实际工业方面的软件系统开发。

RAISE方法只是提供了开发的框架和指导方针,用户可以根据特定的环境和工程标准等因素,自由选择形式化的水平和技术。它不支持软件开发的全过程。

### 1.3.4 B 方法(J. R. Abrial)

B方法[67]于80年代初产生于法国,90年代中期发展成熟。它是在VDM和Z的基础上发展而来的,和Z尤其相似。B方法也是基于模型的形式化开发方法,并且具有很高的数学抽象程度,它提供了一种统一的支持软件开发全过程的标示语言(描述规范、设计、编码及正确性证明、测试)。这种开发方法通过一系列的数学表示(逻辑理论、集合理论:关系、集合、函数),利用谓词转换对模型进行逐步的精化,把原始规约的高级抽象数据类型逐步变换到可执行的目标程序设计语言。

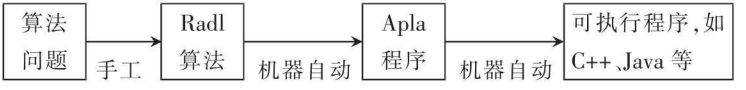
B方法的重要组成部分是抽象机(Abstract Machine),抽象机用Abstract Machine Notation(AMN)语言来描述。用B方法开发的软件系统通常由一系列的抽象机构成。每个抽象机包括数据和操作,它实际上就是一种封装机制。在抽象机的具体实现中,数据仍用集合理论模型来实现,操作却是用AMN的子集——伪程序表示。

B 方法有一个 B 方法工具包( B - Toolkit) ,支持用户编写、验证和维护软件; 它还提供了一个大的数学规则库。利用这些工具,可以自动执行程序正确性的形式化证明。但使用 B 方法书写需求规范比较复杂。

### 1.3.5 PAR 方法( Xue)

由于本书的研究是 PAR 方法[7][11][17]的一个重要组成部分,故在此,将对 PAR 方法作较为详细的阐述。

PAR 方法是薛锦云教授在多年承担国家 863 计划和国家自然科学基金课题的基础上,通过对现存算法程序设计方法局限性和大量算法程序特性的深入研究,发现寻找问题求解序列递推关系是开发简单而高效算法程序的有效途径,进而提出的一种统一实用的算法程序设计和证明方法。它的总体思想可用下图表示:




**PAR 方法开发过程略图**

PAR 方法是一种基于分划和递推的部分形式化算法程序设计方法,支持算法程序开发的全过程。它的理论基础是 Floyed、Hoare 和 Dijkstra 的程序正确性证明理论。它包括四个组成部分: 算法程序设计方法、基于递推关系的算法设计语言 Radl( Recurrence_based Algorithm Design Language) [64]、抽象程序设计语言 Apla( Abstract Programming Language) [65]及一系列转换系统。

PAR 方法开发算法和程序的全过程可以分成以下 6 步:

第 1 步,用自定义的算法设计语言 Radl 精确地描述待求解问题的功能规约。根据不同的需要,这种描述可以是形式化的,也可以是非形式化的。

第 2 步,把需求解的问题分划成若干和原问题结构相同但规模更小的子问题,分解可一直进行下去直至每个子问题可直接求解。可直接求解的子问题称为最小子问题。分划是解复杂问题,得到快速算法的一般策略。

第 3 步,基于文[20][21],构造问题求解序列的递推关系 $S_i = F$

($S_j$)，并给出在递推关系中出现的变量和函数的初值，然后将所有递推关系和初始化条件结合起来成为一个用算法设计语言 Radl 描述的抽象算法。

第 4 步，基于我们提出的循环不变式的新定义和开发新策略[6] [14~15]，在第 3 步所得递推关系的基础上，通过引入变量记录递推关系中函数的值，并约束循环控制变量的变化范围即构成了循环不变式。

第 5 步，基于所得的 Radl 算法和循环不变式开发抽象程序设计语言 Apla 书写的算法程序。用我们的 Radl→Apla 程序转换器可以机械地、自动地把 Radl 算法规范和 Radl 算法转换成 Apla 程序规范和 Apla 程序。

第 6 步，用我们的一系列程序转换器机械地、自动地把 Apla 程序转换为可执行的高级语言程序，如：Ada、C + +、Delphi、Java 等。

这种方法结合形式化方法和非形式化方法，正确区分开发过程中的创造性劳动和非创造性劳动，充分利用特殊问题中使用的分划和递推技术，为我们有效地开发出正确的算法程序提供了理论和方法学的支持。

以下给出一个本人推导的算法实例来详细阐述运用 PAR 方法推导算法的过程。

问题：求 n 的 m 次方的末 p 位数。即给定自然数 n，求它的 m 次方所得结果的末 p 位数是多少。

1. 构造问题的规范

AQ：给定 $n>0 \wedge m>0 \wedge p>0$，

用 f(n,m,p) 记 n 的 m 次方的末 p 位数，则

AR：$f(n,m,p) = \prod(i: 1 \leqslant i \leqslant m: n) \bmod \prod(i: 1 \leqslant i \leqslant p: 10)$

2. 分划原问题

根据题意，将原问题进行如下划分：

$f(n,m,p) = F(f(n,m-1,p),m)$

3. 寻找递推关系

根据问题的规范及分划，用 pp 来记 $\prod(i: 1 \leqslant i \leqslant p: 10)$，则：

$$
\begin{aligned}
f(n,m,p) &= \prod(i:\ 1\leqslant i\leqslant m:\ n) \bmod pp\\
&= \prod(i:\ 1\leqslant i\leqslant m-1 \wedge i=m:\ n) \bmod pp\\
&= (\prod(i:\ 1\leqslant i\leqslant m-1:\ n) * n) \bmod pp(\text{用 mod 的分配率得到下一步})\\
&= ((\prod(i:\ 1\leqslant i\leqslant m-1:\ n) \bmod pp * n \bmod pp) \bmod pp\\
&= f(f(n,m-1,p) * f(n,1,p),1,p)
\end{aligned}
$$

于是得到递推关系: $f(n,m,p) = f(f(n,m-1,p) * f(n,1,p),1,p)$

根据递推关系,我们得到如下解此问题的算法:

```
ALGORITHM: Pend
|[Var i,f,f1,pp: integer; ]|
{AQ∧AR}
BEGIN: i=1++1; pp=∏(i: 1≤i≤p: 10); f=1; f1=n mod pp
TERMINATION: i=m;
RECUR: f=(f* f1) mod pp
END
```

4. 构造循环不变式

用谓词精确表达上述循环变量的变化规律,让变量 f 存放 f(n,m,p) 的值,得到下列循环不变式:

$$f = f(n,i,p) \wedge 1 < i \leqslant m \wedge f1 = f(n,1,p) \wedge pp = \prod(i: 1\leqslant i\leqslant p: 10)$$

5. 构造对应的程序

利用上述递推关系和循环不变式,可得如下的算法程序:

```
program Pend;
{PQ: 给定 n>0∧m>0∧p>0}
{PR: ∏(i: 1≤i≤m: n) mod ∏(i: 1≤i≤p: 10)}
var n,m,j,pp,f1,ff,i: integer;
begin
  writeln("求 n 的 m 次方的末 p 位数--请输入 n:");
```