Z形式规约的自动求精研究

王宏生 著

国防工业出版社
National Defense Industry Press

形式规约的自动求精研究

王宏生 著

国防工业出版社
·北京·

图书在版编目(CIP)数据

Z形式规约的自动求精研究/王宏生著. —北京:国防工业出版社,2009.1
ISBN 978-7-118-06044-7

Ⅰ.Z... Ⅱ.王... Ⅲ.电子计算机—算法理论
Ⅳ.TP301.6

中国版本图书馆CIP数据核字(2008)第177770号

※

国防工业出版社出版发行
(北京市海淀区紫竹院南路23号 邮政编码100048)
天利华印刷装订有限公司印刷
新华书店经售
*
开本850×1168 1/32 印张7⅝ 字数194千字
2009年1月第1版第1次印刷 印数1—2500册 定价26.00元

(本书如有印装错误,我社负责调换)

国防书店:(010)68428422 发行邮购:(010)68414474
发行传真:(010)68411535 发行业务:(010)68472764

前　言

为了克服自然语言和程序设计语言描述规格说明产生的缺陷，人们提出了一种新的软件开发范型，其基本思想是对系统建立一个数学模型，研究和提供一种基于数学的或形式语义学的规格说明语言，用这种语言严格地描述所开发的软件功能，并由自动程序设计的加工模型来得到可执行的代码。在软件开发系统的开发过程中，系统需求分析和规格说明非常重要。该阶段形成的过程说明文档既是软件开发人员和用户之间的规约，又是软件开发的起点。

Z 形式规约作为一种软件工程语言和形式化方法，在软件文档规范化的应用中成效显著。近年来，国内外对 Z 形式规约的研究也逐渐增多。Z 形式规约与其他符号系统相比，一个显著特点是它的规格说明可以用于推理和求精，它不仅可以在需求分析和系统设计阶段描述软件系统，而且可以证明设计的软件系统是否正确。

Z 形式规约描述的软件系统是给人看的，不具有机器可执行性。如果给出了一个 Z 描述的软件系统，但不知道它对应的可执行机器码或者高级语言源代码，其应用必然十分有限。因此，如何将 Z 形式规约转换为机器可执行代码（即求精）成为世界范围关注的课题。以往的这种求精过程都由人工完成，需要专业的数学知识，而且对于大型软件，Z 形式规约描述系统也是极其庞大的，转换过程极易出错。

目前，人们对 Z 形式规约到可执行机器代码的自动求精进行了广泛的研究，提出了各种方法，却进展缓慢。本书采用了另一种研究方式。

本书以Z的自动求精为目的，设计了一个Z的子集Smart Z。Smart Z在设计中继承了Z语言的整型、集合、幂集、笛卡儿积、谓词、关系、函数、序列和包等形式规约语言所具有的特征，同时保证了其规格说明可自动求精性，并以一套形式化的方法——正则表达式和EBNF范式描述了它的语言体系，再通过一系列变换转化成无二义性的SI－NS图，根据此图就可以写出相应的各个子程序。提出了用STL中的不同容器表示Smart Z规格说明各数据类型的思想，利用C++及STL技术设计了Z中集合、关系、包、序列等数据类型的求精规则；结合C++语言的模板、重载技术和STL模板库对数据结构和通用算法的强大支持功能，为Smart Z各操作算子到C++代码自动求精提供了相应的函数模板，制定了Smart Z各算子的求精规则。全书共10章和6个附录，具体内容如下。

第1章，介绍了形式化软件开发的意义、Z形式规约的诞生与发展、Z形式规约的体系以及本书研究的内容和意义。

第2章，介绍了本书自动求精的目标代码体系——C++代码和STL标准模板库。本书的观点是：如果Z形式规约能够自动转换为C++和STL，就表明Z形式规约的自动求精已完成，因为C++和STL是可以编译为可执行机器代码的。

第3章，通过理论推导，证明了Z形式规约经过适当的约束后，其一阶逻辑算子的自动求精是可以实现的。这是Z规格说明自动求精目标实现的一个重要前提，也是为实现自动化程序设计做出的一次有意义的尝试。

为了研究Z的自动求精，摒弃了Z的难以自动求精的成分，将Z精简为可以自动求精的形式规约——Smart Z。在设计中Smart Z继承了Z语言的整型、集合、幂集、笛卡儿积、谓词、关系、函数、序列和包等形式规约语言所具有的特征，同时保证了其规格说明可自动求精性，并以一套形式化的方法——正则表达式和EBNF范式描述了它的语言体系。

第4章，研究了Smart Z自动求精的整体过程，从词法分析到语法分析再到自动求精技术的关键——语义分析。语法分析中采

用了 SI－NS 图，将第 3 章 Smart Z 的 EBNF 范式通过一系列变换转化成无二义性的 SI－NS 图，根据此图就可以写出相应的各个子程序。语义分析研究了符号表的设计、语义树的建立、数据求精和过程求精等技术。

第 5 章，讨论了一阶逻辑表达式的处理方法，给出了一阶逻辑算子包括量词和连接词的自动求精技术。

第 6 章，结合 C++语言的模板、重载技术和 STL 模板库对数据结构和通用算法的强大支持功能，为集合论中各操作算子到 C++代码自动求精提供了相应的函数模板，制定了集合论各算子的求精规则。

第 7 章，通过对幂集类型以及结构的深入分析，找到与之相似的 STL 容器的存储结构与数据结构，讨论了幂集类型在自动求精过程中的实现方式，并给出在求精过程中的算法以及规则。

第 8 章，研究了笛卡儿积算子的数据求精和过程求精。笛卡儿积本质上是 n 个集合元素组成序偶的集合，根据笛卡儿积自身的特点，数据求精后转化成 C++中的记录类型，而在过程求精中则采用了 C++模板和 STL 技术，首先编写了两个乃至多个集合所形成的笛卡儿积的模板，其次制定了两个笛卡儿积之间的交、并、补的求精规则。

第 9 章，研究了关系和函数的存储、关系和函数转换为中间代码的方法，设计了中间代码的自定义函数，实现了全入射函数、部分入射函数、全满射函数和部分满射函数的自动求精。

第 10 章，利用 C++及 STL 技术设计了序列和包各种操作的求精规则（这里采用 STL 中的 deque 容器表示 Z 规格说明中的序列，map 容器表示 Z 规格说明中的包），并产生目标程序，即把 Z 规格说明转换为 C++代码，实现了序列和包的自动求精。

附录 1～附录 6，给出了 Z 形式规约、Smart Z 形式规约的文法等必要的技术说明。

本书完成如下工作：

（1）以自动求精为目的，精简 Z 形式规约为 Smart Z，设计了

Smart Z 的词法、语法和语义；

（2）证明了规格说明求精的可行性，为 Smart Z 的自动求精实现奠定理论基础；

（3）分析 Smart Z 的词法，将其特有的性质与其他语言的性质区分开，进行特别处理，实现了 Smart Z 的词法分析器；

（4）在 Smart Z 文法的基础上，对文法进行分析，对存在二义性和左递归的文法进行修改，并通过程序图形化表示法 SI－NI 图将文法转换为语法分析程序；

（5）设计与建立符号表和语义树，为语义分析和自动转换提供完整信息；

（6）对 Smart Z 中的变量以及一阶谓词公式进行语义分析，并实现其自动转换程序；

（7）对幂集算子的自动求精过程进行分析与研究，提出 STL 容器与广义表结构来实现幂集对象；

（8）完成了 Smart Z 规格说明序偶和集合的自动求精，使序偶和集合的操作可以直接由中间代码转换为 C++代码；

（9）运用 C++中的 STL 中的 deque 容器和 map 容器模板实现了 Smart Z 中的序列和包的各种操作的自动求精；

（10）利用 C++及 STL 技术，设计了对于 Smart Z 中的笛卡儿积数据类型和过程类型的求精模板及规则，实现了其操作的自动求精；

（11）利用 C++及 STL 技术，设计了 Smart Z 中的关系、函数、序列、包等数据类型的求精规则，实现了其操作的自动求精。

本书的技术研究历时 3 年，有王晓龙、李巍巍、苏北、文欣、于云赫、古丽 6 名研究生参与，他们的硕士学位论文都是本书的一部分，在此向为本书做出贡献的研究生表示感谢！

感谢国防工业出版社和李宝东编辑对本书出版的大力支持！

著 者

2008 年 10 月

目 录

第1章 Z形式规约

1.1 软件开发的形式化方法

1.1.1 形式化方法的意义

软件开发的过程是从对用户的需求进行分析开始的，经过设计和实现，最后产生可以正确执行的代码以及有关使用和维护的各种文档。完成这个过程的方法称为软件开发方法。

传统的软件开发方法引入一些非形式的图形和文字符号，提供一定的设计原则，协助开发人员按照一定的步骤，比较明确和简练地书写设计文档，用人工方式开发出所要的软件。从各软件加工模型来看，用户需求的规格说明在软件开发过程中具有非常重要的地位。其使用者包括用户、系统分析员、设计与编程人员、测试和验收人员。

许多软件项目开发的经验表明，软件开发费用的大部分是用于纠正在测试阶段所发现的各种错误。而更详细的调查告诫我们，这些错误中的很大一部分可追溯到项目开发的早期阶段，即在实现和测试阶段所花的高额费用是由于需求分析和设计阶段虚假的“节省”所引起的。需求分析阶段的错误通常就是规格说明的描述不精确。

众所周知，采用自然语言描述的非形式的规格说明具有易读、易理解的优点，但它通常具有模糊性和歧义性（二义性），经常是不精确和不完整的。这往往引起规格说明的使用者对同一规格说明产生不同的理解，最终导致用户对已完成的系统不满意。另外，

传统的开发方法由于采用不严格的非形式技术来描述目标软件系统的规格说明，因此，也很难得到计算机的有效支持，显然，这不利于软件质量和生产率的提高。

提高软件生产率的根本途径之一是软件开发过程的自动化技术，另外，实现软件开发过程各阶段的自动化也是软件工程的重要目标之一。软件自动化的前提是形式化，即以一种精确的方法严格地描述软件系统的定义和需求，并且这种精确的表述方法是建立在严格的数学基础上的：它通过集合、谓词逻辑、函数映射等数学方法，保证软件系统从规格说明到代码生成每一阶段的每一次形式变换都有严格的数学推演和正确性证明，从而保证其语义不变。

形式化方法的研究和应用已有 30 多年的历史。最初的产生是由 Dijkstra 和 Hoare 在程序验证方面的工作，以及 Scott、Stratchey 和其他学者在程序语义方面的工作基础上发展起来的。形式化方法（Formal Methods）是一种新的软件开发范型，即通过形式化、规范化的数学理论，用描述“做什么”来取代“怎么做”。其基本思想是对系统建立一个数学模型，研究和提供一种基于数学的或形式语义学的规格说明语言，用这种语言严格地描述所开发的软件功能，并由自动程序设计的加工模型来得到可执行的代码。

形式化方法的优越之处在于它具有严格的数学基础与描述性，在软件开发的过程中使用形式化方法具有下列优点：

(1) 形式规格说明是精确的；

(2) 由误解引起的错误减少；

(3) 形式规格说明利于系统实现；

(4) 能够对形式规格说明进行正确性证明。

软件形式规格说明是形式化方法最基本的部分，它精确地描述了用户的需求、软件系统的功能和各种性质。形式化方法的规格说明具有良好的数学基础，易于研究软件规格说明性质，如规格说明的一致性、完备性以及规格说明间的等价性等，便于机器的自动处理。而非形式化方法的规格说明虽然在书写、理解和使

用上有其独到的优点，且适于描述软件需求规格说明，但其缺陷也是很明显的，那就是语义的歧义性和描述的不完备性对于软件的自动化增加了重重障碍。

形式化软件规格说明不仅是对用户需求，也是对软件系统的严格定义，在软件开发中有着相当重要的作用。另一方面，为了提高软件生产率，便于从软件规格说明自动或者半自动地导出正确的程序，对软件规格说明语言相应地提出了新的需求，即要求它不仅要能适合描述大型软件，便于用户使用，利于软件产品的可靠性，而且还要有利于机器的自动处理。因此，将形式化的理论和方法用于需求分析与规格说明极为必要，也是实现软件自动化[8]生产的根本前提。

随着计算机产业的快速发展，软件的开发规模不断扩大，对软件开发效率和安全性的要求也越来越高，各种开发方法应运而生。在这种环境下,一种新的软件开发范型即形式化方法被提出来，形式化方法开发软件日益受到了工业界的高度重视，并得到了越来越广泛的应用。著名的欧洲 Esprit 项目的 Large Correct System，美国 HP 公司 Analysis Information Library 等都是采用了形式化方法。

形式化方法是基于数学方法来描述目标软件系统性质的一门技术，用严格的数学符号和数学法则对目标软件系统的结构与行为进行有效的综合分析和推理，它为系统的说明、开发和验证提供了一个框架，以利于发现目标软件系统需求中的不一致性、不完整性等情况。形式化方法一般需要形式规约语言的支持。形式规约语言提供了一个称为语法域的记号系统和一个称为语义域的目标集合，以及一组精确地定义哪些目标系统满足哪个规格说明的规则。

使用形式规约语言来描述软件的需求规格说明，具有以下优点：

(1) 基于严格的数学概念和理论，避免了用自然语言描述时可能带来的模糊性和歧义性；

(2) 具有很强的抽象性，避免了在需求分析阶段对数据结构

和算法细节的详细描述；

(3) 形式语言可以交由计算机自动处理，可利用相应的软件工具对形式规约进行分析、查错、验证以及求精变换；

(4) 便于对形式规约的各种性质进行推理和证明。

不过，形式规约的理论和技术远未达到可以在工业界应用的程度，特别是对于大规模软件系统的设计更是如此。然而，形式规约说明的缺陷正逐步被克服，随着形式化方法研究的深入和支撑工具的增加，许多具体的形式化方法研究中所提出(或出现)的技术在实践中被广泛采用。形式规约在工业界推广应用的可能性也在逐步增加。

形式化方法的基本含义：借助数学的方法来研究计算机科学中的有关问题，数学中通常采用一种严格的、不便于交流的形式手段来进行高度抽象的推理和证明过程；与数学家的形式化不完全一样，计算机科学家利用形式技术来描述具体的问题，如形式语言、形式语义、形式推理规则、形式证明等。

1.1.2 Z 形式规约[①]的产生与发展

已有的形式化方法主要是采用形式规约语言。在用某种程序设计语言具体实现目标软件系统之前先准备一个严格的形式规约说明，形式规约指明目标软件系统“做什么”，而不考虑“怎么做”。形式规约语言有很多，例如，Z、VDM、LOTOS、OBJ、Larch等。不同的形式规约语言在表达能力、术语、准确性以及支持形式处理的能力方面有所不同,每种语言的侧重点也不同。语言涉及的范围也很广，从通用的到专业的，从具有较高表达能力的到有些限制的，从多态的到不可变类型的。有些语言可以进行语法分析、语义分析并提供运行机制，有些语言则主要起到规格说明的作用。

不论何种形式规约语言，在软件开发过程中都会应用于需求

① “Z 形式规约”英语为 Z Specification，国内有的文献译为“Z 规格说明”。本书采用了“Z 形式规约”。但在有的叙述中，如形式语言叙述，“Z 规格说明”一词更恰当，因此，本书中“Z 形式规约”和“Z 规格说明”可理解为同义，未作严格区分。