# 面向对象软件自动化系统 JDAUTO/0*

金淳兆 全炳哲

(吉林大学计算机科学系 北京 130023)

**摘要** 本文介绍了面向对象软件自动化系统 JDAUTO/0 的设计与实现. JOOSL 是一种面向对象软件形式规约语言,基于它,实现了概要设计到详细设计的自动工具 PDAUTO 和详细设计到 C++代码的自动转换工具 DDAUTO.

**关键词** 形式规约语言,面向对象设计,自动程序设计,软件自动化,软件工具.

软件自动化是提高软件生产率的途径之一. 自从 70 年代提出面向对象概念以来,由于它很好地支持了软件开发的各侧面,面向对象软件技术已成为重要的软件开发新技术,而且面向对象软件的自动化问题已成为重要的研究课题.

软件自动化的前提是形式化. 目前,已设计出了多种面向对象形式规约语言. COLD-K 语言[1]是一种基于代数规约技术的面向对象设计形式描述语言,可用于描述概要设计和详细设计,其中为描述详细设计的算法特性,提供一组表达式. Object-Z 语言[2]是 Z 语言的一种扩充,其中引入了类及其继承的描述机制. Z++语言[3]是 Z 语言的另一种扩充,其中引入了类的描述机制.

软件自动化方法有多种,如:(1)基于演绎推理的程序综合;(2)基于知识的途径;(3)程序变换方法;(4)基于归纳推理的程序推导;(5)基于复用构件的途径. Tableau 方法[4]是一种较典型的基于演绎推理的程序综合方法. Nuprl 系统[5]是基于 Martin-Löf 类型理论的程序自动综合系统. PX 系统[6]中用 px-realizability 抽出 Lisp 程序. CIP 系统[7]是基于程序变换技术的一种软件自动化系统. 南京大学研制的 NDAUTO[8]系统以 HOS 方法为基础,采用转换与过程化相结合的方法,研究了软件自动化的问题;在 NDADAS 系统[9]中研究了算法设计的自动化问题. 本文介绍一种基于面向对象设计形式描述的软件自动化系统 JDAUTO/0 的原理.

## 1 JDAUTO/0 系统概要

JDAUTO/0 由用户界面、类库管理工具、PDAUTO 工具、DDAUTO 工具、概要设计库

* 本文研究得到国家"八五"重点科技攻关计划、国家 863 高科技项目和国家自然科学基金项目资助. 作者金淳兆,1937 年生,教授,主要研究领域为面向对象技术,程序自动化,软件工程. 全炳哲,1961 年生,副教授,主要研究领域为面向对象技术,程序自动化.

本文通讯联系人:金淳兆,长春 130023,吉林大学计算机科学系

本文 1995-09-07 收到修改稿

和详细设计库构成,图 1 是其系统结构图. JDAUTO/0 系统的基础是软件设计形式规约语言 JOOSL[10],它分为 2 个部分,JOOADL 语言和 JOODDL 语言. 在面向对象软件开发中,不同阶段之间存在许多固有的重叠现象,皆以对象(或类)为中心进行开发活动. 尤其是需求分析阶段和概要设计阶段更体现这种重叠. JOOADL 的设计目标就是既能描述面向对象软件的需求规约,也能描述面向对象软件的概要设计. 软件高层定义到目标代码的过渡中,总是需要有中间阶段,这就是详细设计. JOODDL 的设计目标是:(1)能够方便地描述对象的数据和操作的主要算法;(2)提供具有较高抽象度的描述机制,以支持详细设计;(3)可实现详细设计到目标代码的自动转换.

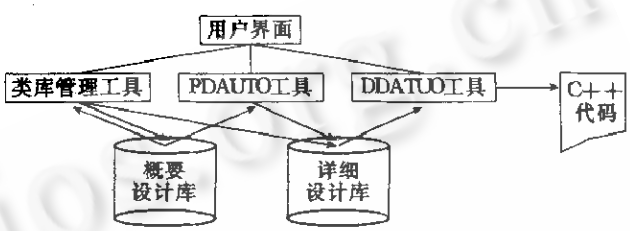

图1 JDATTO/0系统结构

类库管理工具用于管理概要设计库和详细设计库,它提供编辑功能、类的创建和删除功能、浏览类库功能等.

PDAUTO 是概要设计到详细设计的一种自动转换工具,它采用了基于验证程序的一种程序综合方法,其输入是用 JOOADL 描述的软件概要设计,输出是用 JOODDL 描述的软件详细设计. JOOADL 的基本描述手段是一阶谓词. 如果能够消去规约中的所有量词,可较容易地综合程序. 因此,PDAUTO 的主要研究目标是消去量词,其关键是从规约中获得关于变量的详细信息.

DDAUTO 是详细设计到代码的一种自动转换工具,它把用 JOODDL 描述的软件自动转换成 C++代码. DDAUTO 所涉及到的关键问题是 2 个语言之间的对应方法和高抽象级的成份到代码的自动转换方法.

## 2 JOOSL

JOOSL(Jida object-oriented specification language) 是以 COLD-K 语言为基础、吸收 Z 语言的描述方法和 Recos 语言[11]语法格式的一种面向对象软件设计规约语言. 从面向对象软件开发模型[12~14]角度看,不需要严格区分不同的开发阶段,开发过程是一个螺旋形的渐进过程. 但从软件开发的形式描述和软件自动化角度看,应该用形式方法定义软件的需求规约和软件设计结果. 在 JOOSL 中,用 3 个层次描述面向对象软件:(1)需求规约;(2)概要设计规约;(3)详细设计规约. 其中用 JOOADL 描述前 2 个层次,用 JOODDL 描述第 3 个层次.

### 2.1 JOOADL

在面向对象分析(OOA)中,首先,要准备软件需求文档;其次,要识别对象,包括确定对象属性和服务;第 3,要识别对象之间的关系,包括引用关系和继承关系. 因此,需求规约中应给出对象的属性描述、服务描述、引用关系和继承关系的描述. 由于 OOA 的主要目标是严格确定用户需求,所以在对象的描述中应主要考虑系统中的各对象及其关系,而不考虑实现方法. 面向对象概要设计(OOPD)是 OOA 的延续和扩充. OOPD 的主要目标是从实现角度进一步分析类及它们之间的关系. 首先,进一步分析系统所涉及到的类,从实现角度引入附加类;其次,分析类之间的公共属性,构造抽象类;第 3,要选择操作的算法特性,这就常常

需要引入附加操作,这种操作属实现细节,应该作为类内部的私有操作.虽然,OOA 和 OOPD 的目标不同,但从描述形式上看,需求规约和概要设计规约在很大程度上类同,这也就反映了 OO 开发模型中任务和任务之间的重叠.在 OOA 和 OOPD 阶段,均可用 JOOADL 描述软件.

在本语言中,软件的基本单位是类.为提高软件高层描述的抽象度,在 JOOADL 中用操作统一地描述类的数据特性及所提供的服务.这就为实现者提供更多的灵活性,以选择不同的实现方案.操作分为函数操作、谓词操作和过程操作.函数操作用于观察对象的当前内部状态.函数没有副作用,它的特点是它们可以用来表示类的实例变量.但它属于实现细节.实际实现中,对象状态可用一些变元记录,也可用一段代码实现.谓词操作用于判定对象当前状态下的某种条件的成立与否.它们的作用类似于函数,但它无返回值,它们在某种状态下要么成立,要么不成立.具体实现中,谓词可对应于布尔型函数或布尔型实例变量.过程操作用于修改对象的当前状态,也可用于描述完成某种功能的操作.由于用函数和谓词描述对象状态,对象状态的改变就指修改函数的返回值或改变谓词的成立与否.在过程的说明中,需要描述过程的修改权限,即指定过程所修改的函数和谓词.

操作的功能基本上用一阶谓词逻辑的公式描述.但是,在下述几个方面区别于经典逻辑.首先,增加不可交换的合取联结词,称为弱合取,记为分号(;).例如,$A;B$,其中 $A$ 或 $B$ 可以是引用过程操作的子公式,其形式是:<过程引用>[TRUE],意味着过程引用结束后,整个子公式取"真"值.由于过程引用引起状态变化,所以弱合取右边的公式的值依赖于新状态.因而弱合取不可交换.当 $A$ 和 $B$ 都不是过程的引用时,$A;B$ 等价于 $A \wedge B$.其次,操作的描述中,允许使用归纳方法.归纳定义方法的形式如下:

$$\text{IND for } t_i: T(PB(t_1, t_2, \ldots, t_n), PI(t, t_1, t_2, \ldots, t_n));$$

它是施归纳于 $t_i$,其中 $PB$ 和 $PI$ 是谓词,分别对应于基始和归纳.这里 $T$ 应是良序类型,包括非负整数、非正整数和表.第 3,谓词公式中,可使用表示下一个状态的算子,用撇号(')表示.例如,$a'$ 表示下一个状态下的变元 $a$,函数 $a$ 或谓词 $a$ 的值.第 4,原子公式只有关系表达式和操作的引用.

## 2.2 JOODDL

JOODDL 中程序的基本单位仍然是类,其描述形式类似于 JOOADL 中类的形式.JOODDL 和 JOOADL 的区别在于 JOODDL 中可直接说明类的实例变量,且用算法化的形式描述操作.变量类型可以是基本型或类,其中基本型包括整型、实型、字符型、布尔型和数组型.操作分为函数和过程,有返回值的操作称为函数,无返回值的操作叫做过程.操作的算法是用一些结构化的控制语句描述.JOODDL 提供 if 语句、for 循环语句、while 循环语句、repeat 循环语句、消息发送语句、输入/输出语句、return 语句和 compute 语句.控制语句中条件表达式可以是受限谓词.compute 语句是赋值语句的一种拓广,可描述算法特性较明显的一系列处理动作,其形式如下:

compute <变元序列>':'<受限谓词>

所谓受限谓词的含义是:首先,形式上和一阶谓词逻辑中的公式相同;其次,作为详细设计的描述手段,必须提供完整的算法特性.对谓词公式的限定体现在如下 5 个方面:(1)公式中出现的变元必须已定义;(2)受限谓词 $P$ 中,量词所约束的变元的类型必须是离散量,且

有上、下界的规定;(3)谓词公式应体现完整的计算特性;(4)谓词中允许出现带撇号(')的变元,撇号是表示下一个状态的算子,但不允许出现变元之间的循环定义;(5)等式公式的左边必须是作用撇号算子的变元.

## 3　PDAUTO 工具

PDAUTO 是一个程序综合工具,它的输入是用 JOOADL 描述的软件概要设计,输出是用 JOODDL 描述的软件详细设计. 在 PDAUTO 中,采用构造规约的验证程序的方法综合出对应的程序.

**定义 1(验证程序).** 设 $F$ 是 JOOADL 中的一个操作,$S$ 是 $F$ 的规约,$P$ 是 JOODDL 中的一个操作. $F$ 的输入参数为 $x_1,\cdots,x_n$,结果为 $y$(若有返回值),ENV 是执行 $F$ 前的环境,ENV'是执行 $F$ 后的环境. 若 $P$ 满足下列条件,则称 $P$ 为 $S$ 的验证程序.

(1)$P$ 和 $F$ 具有相同数目和类型的输入参数和输出参数;

(2)当 $F$ 是函数操作时,若 $S(x_1,\cdots,x_n,y)$是在 ENV 下某理论 TH 中的定理,则函数子程序 $P$ 终止,且 $S(x_1,\cdots,x_n, P(x_1,\cdots,x_n))$是在 ENV 下 TH 中的定理;

(3)当 $F$ 是谓词操作时,则函数子程序 $P$ 终止,且若 $S$ 为在 ENV 下某理论 TH 中的定理,则 $P$ 的结果值为真,否则为假;

(4) 当 $F$ 是有返回值的过程操作时,若 $S'(x_1,\cdots,x_n,y)$(它是把 $S$ 中的所有引用撇号算子的原子公式换成 TRUE 后的公式)为在 ENV 下某理论 TH 中的定理,则函数子程序 $P$ 终止,且 $S''(x_1,\cdots,x_n, P(x_1,\cdots,x_n))$(它是用 ENV' 中的相应操作之结果替换 $S$ 中所有作用撇号算子的操作后的公式)是在 ENV 下 TH 中的定理;

(5) 当 $F$ 是一个无返回值的过程操作时,若 $S'$(它是把 $S$ 中的所有引用撇号算子的原子公式换成 TRUE 后的公式)为在 ENV 下某理论 TH 中的定理,则过程 $P$ 终止,且 $S''$(它是用 ENV'中的相应操作的结果替换 $S$ 中所有作用撇号算子的操作后的公式)是在 ENV 下 TH 中的定理.

在构造验证程序的过程中,确定约束变元的范围起关键性的作用. 任何一种程序规约或多或少都含有变元范围的信息. 利用这些信息可构造相应的程序. 从规约中获得的关于变元信息越多,构造验证程序中需要的人工干预越少.

**定义 2(已知变元).** 若变元 $x$ 满足如下递归定义的条件之一,则称变元 $x$ 为已知变元.

(1) 它是输入变元;

(2) 它等于表达式 $E$ 的值,且 $E$ 中出现的所有变元皆已知;

(3) 它可用 2 个表达式 $E_1$ 和 $E_2$ 限定,即 $x\leqslant E_1$ 且 $x\geqslant E_2$,且 $E_1$ 和 $E_2$ 中所有变元皆已知;此时,也称 $E_1$ 为 $x$ 的上界,称 $E_2$ 为 $x$ 的下界.

我们称非已知变元为未知变元. 对一个有确定值的变元,可认为它具有相同的上、下界.

**定义 3(原子公式的未知度、项的未知度).** 原子公式(项)中含有的未知变元的个数,称为原子公式(项)的未知度.

对规约进行分析时,从关系表达式或项中获取越多的信息,它们的未知度会越低. 当所有的原子公式的未知度都为 0 时,可自动构造对应的程序,不需要人工干预. 在程序综合中,不同的关系表达式起不同的作用,从这些关系表达式中提取的信息也不同. 为此将关系表达

式分为如下 4 类：(1)计算型：只含有一个未知变元的一元一次方程；(2)边界型：关系表达式中的关系为>、≥、<、≤，且其中只含一个线性的未知变元；(3)方程型：等式关系表达式中含有多个未知变元，或含有高次未知变元；(4)条件型：其它形式的关系表达式. 为提取变元信息，我们采用了如下的一些策略.

**交换**：设有合取式 $A \wedge B$，如果 $B$ 的未知度为 0；或者 $A$ 和 $B$ 中都出现变元 $x$，且在顺序分析时，$A$ 中的 $x$ 未知，而 $B$ 中 $x$ 已知，则将 $A \wedge B$ 交换为 $B \wedge A$. 交换的主要目的是降低原子公式的未知度.

**移项**：根据变元的未知度，在关系表达式中，将未知度为零的项移到右部，将未知度为非零的项移到左部. 移项的目的在于确定变元的上、下界. 为了确定上、下界，需要将关系表达式中的已知变元全部移到一起，而将未知变元单独分离出来. 这样，在关系表达式的左部只是一个未知变元时，就可以直接得到它的上界或下界. 移项过程中，要严格遵守数学中的一些基本原理：如变号、分配律、交换律、结合律等.

**替换**：设 $x=E$，$rexp$ 是一个关系表达式. 若当用 $E$ 替换 $rexp$ 中的所有 $x$ 的出现时，$rexp[E/x]$的未知度降低，则实施替换. 替换的目的在于把方程型或条件型表达式转换成计算型或边界型表达式，以便获得关于变元的更多信息.

抽取变元范围的信息后，有时需要对规约进行等价变换. 例如，对于

$$\forall x \forall y(y \geqslant 1 \wedge y \leqslant 10 \wedge x \geqslant y+1 \wedge x \leqslant 2y \wedge P(x,y)) \tag{1}$$

变元 $x$ 和 $y$ 是已知的，但 $x$ 的上、下界依赖于 $y$，称这种关系为间接定义关系. 根据变元的间接定义关系、依赖度小的变元在外、依赖度大的变元在内的原则，进行规约的等价变换. 例如，可把(1)式变换成(2)式：

$$\forall y \forall x(y \geqslant 1 \wedge y \leqslant 10 \wedge x \geqslant y+1 \wedge x \leqslant 2y \wedge P(x,y)) \tag{2}$$

由于(1)式和(2)式逻辑上等价，所以可把(2)的验证程序看成(1)式的验证程序.

根据变元范围的信息，对变换后的规约构造相应的验证程序. 验证程序的基本构造方法是对公式中的每个量词构造循环控制结构，对用归纳方法定义的规约构造递归程序. 但对全称量词和存在量词采用不同的构造方法. 对合取式、析取式和蕴涵式，根据其语义解释和逻辑子式的形式，构造相应的程序段. 详细的构造方法另文讨论.

## 4 DDAUTO 工具

DDAUTO 是一个程序变换工具，它的输入是用 JOODDL 描述的软件详细设计，输出是 C++代码. 转换中采用了横向转换方法和纵向转换方法. 横向转换是指相同抽象级间的转换，而纵向转换是指较高抽象级到较低抽象级的转换.

DDAUTO 中横向转换有 2 种：①JOODDL 的控制结构到 C++语言的控制结构的转换. 由于 2 个语言的控制结构相近，这种转换是直接的；②JOODDL 的程序结构到 C++程序结构的转换. 从程序结构上看，JOODDL 程序由一些类和驱动部件(主程序)组成. 转换时，可以以类作为转换单位，把 JOODDL 的类转换成 C++的类. 但是，根据驱动部件的不同，一个类经转换后的代码也应该具有一定的可复用性，应该能够灵活地与其它类代码相连接，组成不同的程序. 我们采用了这样一种策略：转换 JOODDL 的类时，既形成对应于类本身的 C++代码，又记载这部分代码与其它代码段之间的关系. 经转换后，JOODDL 的类对

应C++程序的2个文件,一个文件记录类的继承和引用关系的信息;另一个文件记录类的具体实现,包括实例变量和操作的实现.

纵向转换也有2种:①判定条件的转换,判定条件出现于JOODDL的选择和循环结构;②对compute语句的转换.2种转换都涉及到受限谓词的处理问题,实现时可采用同一种策略.纵向转换中所采用的基本方法是根据变元描述和受限谓词中抽取的变元信息,构造一段程序代码.对于判定条件它能确定给定公式的真假值;对于compute语句,它能确定变元的新值,并使相应的谓词为真.考虑到谓词形式的多样性,在转换过程中,对不同形式的谓词采用不同的转换方法.为了有效地组织运用这些转换方法,建立了相应的知识库,在知识库中不但存放了关于通用解法的知识,还存放了针对性的特殊知识,因而可得到质量较高的代码.

## 5　结束语

JDAUTO/0是为研究面向对象程序自动化而研制的一个实验性系统,它是在SUN工作站上,用C语言开发的.JOOSL是JDAUTO/0的基础,其特点是:(1)它是一种广谱语言,不仅可用于描述面向对象软件的概要设计和详细设计,而且在一定程度上可描述面向对象软件的需求规约;(2)以谓词逻辑为基础,引入弱合取和归纳定义形式,提高了描述能力,支持面向对象软件的形式说明;(3)引入受限谓词的概念,保证准确地描述算法特性的前提下,提高了描述的抽象级,以支持详细设计的形式描述;(4)在JOODDL的设计中,充分考虑了程序自动化的问题.

PDAUTO中探讨了基于验证程序的程序综合方法.利用这种方法可解决演绎推理方法所不能解决或很难解决的问题.演绎推理方法的核心是定理证明.但对某些问题很难给出证明或根本不能给出证明,对这些问题演绎推理方法就无能为力.然而,常常可构造关于这些问题的验证程序,作为所需程序.当规约中的变元皆为已知时,可保证验证程序的完全正确性,当存在量词所约束的变元部分有界时,可构造部分正确的验证程序.经分析后,某些变元仍为未知时,需要人工干预.DDAUTO实现了详细设计到C++代码的完全自动化.由于采用了以类为单位来记载相关性信息的转换方法,有效地解决了自动转换中目标代码的复用问题.

本文的研究工作是探索性的,今后的进一步工作主要有:(1)增强JOOSL的描述能力;(2)研究面向对象需求规约到概要设计的过渡问题;(3)PDAUTO中增加表类型的处理方法,并引进基于演绎推理的程序综合方法;(4)进一步分析软件的规约技术,提取更多用于转换的知识,以便在DDAUTO中生成高效的目标代码等.

## 参考文献

1　Jonkers H B M. An introduction to COLD－K, algebraic methods: theory, tools and applications. Berlin: Springer－Verlag, 1990. 139～200.

2　Carrington David *et al*. Object－Z: an object－oriented extension to Z. In: Formal Description Techniques (FORTE'89), North Holland, 1990.

3　Lano K. A specification－based approach to maintenance. Software Maintenance: Research and Practice 1991,

1991,13(4):193～213.

4 Manna Z, Waldinger R. Fundamentals of deductive program synthesis. IEEE Trans. on Software Engineering, Aug. 1992,18(8):674～704.

5 Constable R L *et al.* Implementing mathematics with the nuprl proof development system. Prentice—Hall, 1986.

6 Hayashi S, Nakano H. PX, a computational logic. MIT Press, Cambridge, 1988.

7 Bauer F *et al.* Formal program construction by transformations—computer—aided, intuition—guided programming. IEEE Trans. Soft. Eng., Feb. 1989,15(2):165～180.

8 徐家福,戴敏,袁峰等. 实验性软件自动化系统 NDAUTO. 计算机学报,1989,15(2),92～97.

9 徐家福,戴敏,吕建. 算法自动化系统 NDADAS. 计算机研究与发展,1990,27(2):1～5.

10 全炳哲,金淳兆. 面向对象软件规格语言的设计. 软件学报,1995,6(12):705～711.

11 全炳哲,余江,金淳兆. 可重用构件及其描述语言. 软件学报,1994,5(1):42～46.

12 Henderson—Sellers B, Edwards J M. The object—oriented system life cycle. CACM, 1990,33(9):142～159.

13 Henderson—Sellers B, Edwards J M. The O—O—O methodology for the object—oriented life cycle. ACM SIGSOFT Soft. Eng. Notes, 1993,18(4):54～60.

14 Hodge L R, Mock M T. A proposed object—oriented development methodology. Software Engineering Journal, March 1992. 119～129.

# JDAUTO/0: An OBJECT—ORIENTED SOFTWARE AUTOMATION SYSTEM

Jin Chunzhao Quan Bingzhe

(*Department of Computer Science Jilin University Changchun* 130023)

**Abstract** This paper describes the design and implementation of an object—oriented software automation system JDAUTO/0. JOOSL is an object—oriented software formal specification language. Based on JOOSL, the authors have developed the PDAUTO tool, a semiautomatic tool which synthesizes detailed designs from preliminary designs, and the DDAUTO tool, an automatic tool which transforms detailed designs to C++ programs.

**Key words** Formal specification language, object—oriented design, automatic programming design, software automation, software tool.