基于范畴计算的多目标语言程序生成架构

王金全[1] 郑宇军[1,2]

(北京装备系统工程研究所 北京 100093)[1] (中国科学院软件研究所 北京 100080)[2]

摘 要 提出了一种基于范畴论的多目标语言程序生成架构,程序元素的元类型在程序元模型范畴中定义,常用的软件开发模式由元模型实例组成并带有可配置的参数,模式到可执行语言的表达式、函数、类型等映射由范畴函子统一定义。在实际应用开发时,通过函子计算将抽象模式精化到不同的目标语言程序范畴。各种语言的精化计算方式具有统一的契约规范,从而支持高度的灵活性和重用度水平。

关键词 范畴论,程序生成,函子,模型变换

中图法分类号 TP301 **文献标识码** A

Category Theoretic Framework of Multi-language Program Generation

WANG Jin-quan[1] ZHENG Yu-jun[1,2]

(Systems Engineering Institute of Engineer Equipments, Beijing 100093, China)[1]

(Institute of Software, Chinese Academy of Sciences, Beijing 100080, China)[2]

Abstract The paper proposed a multi-language program generation framework based on category theory. Meta-types of basic program elements are defined in the category of meta-models. Typical patterns in software development are parameterized and composed by instances of meta-models and can be transformed into different programs through refinements to different executable language categories, the computations of which are based on categorical functors defining the mappings to language expressions, functions and types and complying with a unified contract. Thus the framework significantly improves the flexibility and reusability of program generation.

Keywords Category theory, Program generation, Functor, Model transformation

1 引言

随着信息系统规模和复杂度的不断提高,程序生成技术在信息系统开发中的作用日益增强。特别是自20世纪90年代以来,程序变换(program transformation)、可操作规约(operational specification)、生成式程序设计(generative programming)等软件开发新范型(paradigm)受到了广泛的关注[1,2],并被应用于对象和构件、数据库访问程序、组合优化算法、Web服务等多种类型程序的快速生成[3-7]。

在大型综合信息系统(如电子商务/电子政务、军事指挥系统等)中,许多子系统和构件都含有相同或相似的功能模块[8]。另一方面,大型开发团队和较长的开发周期也使系统很难限制在一种开发语言或环境上。因此,在程序生成架构中支持多目标语言,能够在兼顾开发效率和灵活性的同时保持系统的一致性和完整性。

本文以范畴论(category theory)为基础,提出了一种支持多目标语言的程序生成架构。作为程序元模型的实例,常见的软件开发模式被定义为参数化的泛型模型,它们可根据功能和配置需求被精化为特定语言的可执行模型(即目标程序),而各种不同语言的模型精化工具都遵循同一抽象契约,这就显著提高了开发的机械化水平和重用度。

2 范畴论基础

范畴论是现代数学的一个年轻分支,它发源于代数拓扑,目前已渗透到理论计算机科学中的多个领域[9]。下面首先介绍其中的一些基本概念和定义。

定义1(范畴) 一个范畴(category)C包括一组对象的集OB_C、对象之间的态射集MOR_C(dom和cod分别表示态射的定义域和协域)以及态射之间的复合操作,且满足:

一对任意的两个态射$f:a\to b$和$g:b\to c$,都存在一个复合态射$g\circ f:a\to c$;

一态射$f:a\to b, g:b\to c$和$h:c\to d$满足结合律$h\circ(g\circ f)=(h\circ g)\circ f$;

一对于C中的每个对象a,都存在一个恒等态射$id_a:a\to a$,它对于任意的态射$f:a\to b$和$g:b\to a$都有$f\circ a=f, a\circ g=g^*$。

定义2(协锥) 给定范畴C及其图解(diagram)D,一个协锥(cocone)$D\to c$包括一个对象$c\to OB_C$以及一族态射$\{f: d_i\to c \mid i\in I_D\}$,且对于任意的$i,j\in I_D$和$g:d_i\to d_j$均有$f_j\circ g=f_i$,如图1(a)所示。

到稿日期:2010-05-13 返修日期:2010-08-19 本文受国家自然科学基金(60773054)资助。

王金全(1960—),男,学士,高级工程师,主要研究方向为管理信息系统,E-mail:seibeijing-wang@yahoo.cn;**郑宇军**(1979—),男,博士,工程师,主要研究方向为软件形式化与自动化。

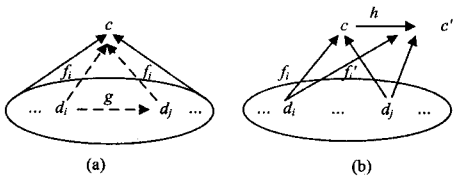

图 1　协锥与协极限的交换图

定义 3（协极限）　给定范畴 C 及其图解 D，D 的一个协极限(colimit)是一个可交换的协锥 $D\to c$，且对于 D 的其它任意协极限 $D\to c'$ 都存在一个唯一的态射 $h:c\to c'$，它对任意的 $f_i:d_i\to c$ 和 $f_i':d_i\to c'(i\in I_D)$ 均满足 $h\circ f_i=f_i'$，如图 1(b)所示。

定义 4（函子）　给定两个范畴 C 和 D，一个从 C 到 D 的函子(Functor) F 是一对操作($F_{OB}:OB_C\to OB_D$，$F_{MOR}:MOR_C\to MOR_D$)，其满足：

$-F_{MOR}(g\circ f)=F_{MOR}(g)\circ F_{MOR}(f)$，$f,g\in MOR_C$；

$-F_{OB}(id_a)=id_{FOB(a)}$，$a\in OB_C^*$。

定义 5（Monad）　给定范畴 C 和函子 $T:C\to C$，C 上的 Monad 是一个三元组$\langle T,\mu:T^2\to T,\eta:Id_C\to T\rangle$，其满足 $\mu\circ(\eta T)=\mu\circ(T\eta)=Id_T$。

3　模型变换与程序生成的范畴框架

3.1　程序元模型

程序元模型包括各种程序基本元素的元类型，如数据类型、函数、常量、变量、运算符、表达式等。我们定义这些元模型为对象构成的范畴 MPM，其中的态射为元模型之间的(可选)构成关系，如从变量到函数的态射及从函数到类型的态射，这样的态射显然满足结合律和单位元律。MPM 范畴中的一类元模型可以组成一个子范畴，如赋值、算术、关系、逻辑等各型运算符的元模型可组成 MPM-OP 子范畴，而各型表达式的元模型可组成 MPM-EXP 子范畴。

3.2　程序模型

复合类型、构件、用户界面、算法程序等都是程序元模型的实例及其组合，即程序模型或逻辑模型。以这些模型为对象可以定义范畴 PM。若其中的两个模型 pm1 和 pm2 分别是 MPM 范畴中的元模型 mm1 和 mm2 的实例，那么其复合模型 cpm 也是 mm1 和 mm2 的复合元模型 cmm 实例，如图 2(a)所示。推广到多个子模型的情况，那么 MPM 中的态射都将在 PM 中得以保持，MPM 中的元模型关系通过范畴计算就可以直接应用到 PM 中的模型关系上。

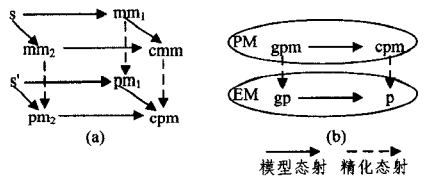

图 2　模型复合与精化

根据泛型程序设计的范畴语义[10]，抽象数据类型可以被定义为其载体(carrier)范畴到自身的函子，基于同一载体的一组类型则可以被抽象为一个泛型 T，其中 TOB 和 TMOR 分别指代泛型数据和泛型操作。那么 PM 可以被分为两个独立的子范畴：含有泛型的程序模型对象所组成的范畴 GPM；不含泛型的具体程序模型对象所组成的范畴 CPM。可重用的软件开发模式在 GPM 上定义，其存储形式可以是领域知识库、类库、构件库、抽象类型和算法库等。在具体系统设计过程中，其中的泛型参数根据系统配置被替换为具体的代数结构，即将 GPM 中的模型转换到 CPM 中的模型。

3.3　可执行程序模型

可执行模型即在具体目标语言上实现的程序模型实例。给定一个可执行目标语言 P，其程序对象组成一个范畴 EMP。类似地，如果语言 P 支持泛型机制，那么 EMP 可以划分为 GEMP 和 CEMP 两个子范畴。这样，在 P 上实现某个软件开发模式 gpm 的途径有两种：一是先将 gpm 实例化为模型 cpm，再将其精化到 CEMP 中的具体程序 p；二是先将 gpm 精化到 GEMP 中的泛型程序 gp，再将其实例化为具体程序 p，如图 2(b)所示。基于范畴计算语义，这两种方式得到的程序是等价的。

4　实现过程

基于上述范畴框架，支持重用的软件开发可以分为 3 个层次：元模型开发，这通常由科研开发团队或平台厂商来完成；逻辑模型开发，主要由基础构件开发团队完成；可执行模型开发，主要由应用开发团队完成。下面首先介绍一个多语言程序生成工具的设计框架，再通过一个数据访问程序模式的应用来说明其实现方式。

4.1　多语言代码生成器

ICodeGen 定义了一个代码生成器的标准接口，其方法契约 GenerateExpressionCode 要求为表达式元模型的实例提供代码生成的功能，GenerateFunctionCode，GenerateTypeCode 则分别要求为函数和类型元模型的实例提供定义代码生成的功能。针对每个目标程序语言可定义具体的代码生成器，如 C++代码生成器、Pascal 代码生成器、C# 代码生成器等。每一种具体生成器应将程序元模型中的预定义基础类型(如布尔、整数、实数)、函数(如取整、求最大值)、运算符(如加、减、乘、除)等映射到该语言的预定义元素，并且提供对 ICodeGen 各种方法契约的实现，从而支持从抽象程序模型到可执行模型的代码转换[11]。

以 C# 代码生成器为例，下面演示了其实现基本运算符元模型映射的方法代码：

```
static string GetCsOp(Operator op)
{
  if(op == Operator. Ev) return "=";
  else if(op == RelOperator. Equal) return "==";
  else if(op == RelOperator. NotEqual) return "! =";
  else if(op == LogicalOperator. Not) return "!";
  else if(op == LogicalOperator. And) return "&&";
  ……
  else return op. ToString();
}
```

在进行表达式、函数、类型等模型精化的过程中，对其中的子表达式、子函数、子类型都要通过递归变换来生成目标代码，递归的终止条件是表达式为单个变量、常量或函数调用式。下面演示了 C# 表达式模型精化的部分方法代码：

```
public override string GenerateExpressionCode(Expression exp)
{
  if(exp is ConstExpression) //常量表达式
```

```
    return new exp. Const. ToString();
  else if(exp is ArithExpression) //二元算术表达式
    return string. Format("({0}) {1}({2})";
  GenerateExpressionCode(exp. SubExprs[0]);
  GetCsOp(exp. Op);
  GenerateExpressionCode(exp. SubExprs[1]));
  ……
}
```

4.2 模式定义与实现

抽象开发模式一般在 GPM 范畴中定义，它带有一组可配置的泛型参数，且与具体目标语言无关。例如在数据库应用程序中，一个典型模式是为对象类生成数据存储方法，下面的代码就演示了如何为抽象类型 T 创建 Save 函数模型：

```
public Method GenerateSaveMethod(DateType T)
{
  Variable id=new Variable(DataType. Long,"id");
  Method save=new Method("Save"+T. Name);
  save. Parameters. Add(id);
  Variable cmd=new Variable(DataType. String,"cmd");
  save. Statements. Add(cmd. InitExpession
  (string. Format("Insert {0} values(",T. Name)));
  save. Statements. Add(new AddAssignExpession(cmd,id));
  save. Statements. Add(new AddAssignExpession(cmd,","));
  foreach(Variable v in T. Fields) {
    if(! v. Modifier. Public) continue;
    save. Statements. Add(new AddAssignExpession(cmd, v. Name
  +"="));
    save. Statements. Add(new AddAssignExpession(cmd,new Ac-
  cessFieldExpression(this,v. Name)));
    save. Statements. Add(new AddAssignExpession(cmd,","));
  }
  save. Statements. Add(new CallMethodExpession(cmd," Re-
  place","cmd. Length-1",")"));
  save. Statements. Add(new CallMethodExpession(new InitExpres-
  sion("DbCommand",cmd),"ExecuteNonQuery"));
  return save;
}
```

一个数据库应用开发模式可以包含一系列这样的抽象程序。在模式应用过程中，将相应的模式对象和 ICodeGen 对象传递给模式管理器，就可以生成相应的框架代码，如图 3 所示。

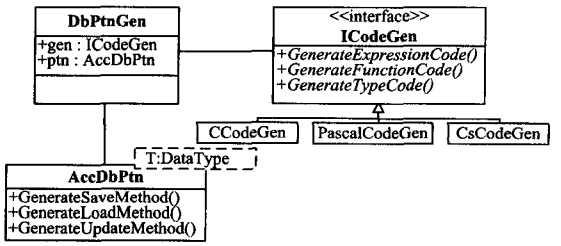

图 3 数据库访问程序模式生成关系图

以 C# 代码生成器为例，给定一个对象类 Customer，设其包含 name，age 和 grade 3 个公有属性，那么基于此模式为其生成的 Save 方法代码如下：

```
void SaveCustomer(long id)
{
  string cmd="Insert Customer values(";
  cmd += id;
  cmd += ",";
  cmd += this. name;
  cmd += ",";
  cmd += this. age;
  cmd += ",";
  cmd += this. grade;
  cmd += ",";
  cmd. Replace(cmd. Length-1,")");
  (new DbCommand(cmd)). ExecuteNonQuery();
}
```

结束语 信息系统开发中存在大量可重用的模式，而且常常需要支持多种目标语言。本文提出了一种基于范畴论的多目标语言程序生成架构。软件开发模式由程序元模型实例构成，并通过泛型参数进行配置；通过参数实例化以及从逻辑模型到可执行模型多种精化工具，就可以方便地为该模式生成多语言的目标框架代码，从而支持高度的灵活性和重用度水平。

参 考 文 献

[1] 范少锋，张乃孝. 生成式程序设计研究概述[J]. 计算机科学，2005，32(3)：12-17

[2] 杨波. 一种新型程序设计范型概述[J]. 计算机工程与应用，2009，45(34)：71-73

[3] 段川，蒋凡. Quine 和自修复式程序生成算法研究与实现[J]. 计算机工程与应用，2004，40(6)：124-127

[4] 陈翔，王学斌，吴泉源. 代码生成技术在 MDA 中的实现[J]. 计算机应用研究，2006，23(1)：147-150

[5] Fan Yong-kai，Lin Jun，Sun Tian-ze. Program generation mechanism based on module relationship match reasoning[J]. Journal of Jilin University(Engineering and Technology Edition)，2006，36(6)：939-944

[6] 叶力，陈俊亮. 基于程序生成的自动化服务组合技术[J]. 计算机工程，2007，33(18)：15-17

[7] Huang Shuang-shuang，Zook D，Smaragdakis Y. Domain-specific languages and program generation with meta-AspectJ[J]. ACM Transactions on Software Engineering and Methodology，18(2)：1-32

[8] 王金全，郑宇军，王侃. 基于 DSL 的装备保障领域建模[J]. 计算机工程，2008，34(2)：66-68，71

[9] Goguen J A. A categorical manifesto[J]. Mathematical Structures in Computer Sciences，1991，1(1)：49-67

[10] Zheng Yun-jun，Hu Qi-min，Shi Hai-he，et al. Introducing advanced generic programming：categorical foundations and applications[C]//Proceedings of 3rd Int'l Conf. Computer Science & Education. Kaifeng，2008：1495-1500

[11] 郑宇军. 基于 PAR 的高可信装备保障算法形式化推演[D]. 北京：中国科学院软件研究所，2009