一种基于并行对象的可视化描述*)

A Visual Description Based on Concurrent Objects

黄永忠　李国巨　郭金庚
(解放军信息工程大学　郑州450002)

Abstract This paper puts forward a visual concurrent programming model based on concurrent objects, which absorbs the basic thought of UML, class diagram is used to describe concurrent classes, shared classes, general classes in SPC^{++} and the relationships among these classes. Through the visual description system can generate the code framework automatically.

Keywords Concurrent object, Visualization, POVPE, SPC^{++}

一、前言

随着并行计算机应用的深入，人们对并行程序的开发需求不断增加。相对于串行程序而言，并行程序必须考虑同步、互斥、通信等问题，使得并行程序设计难度较大，要求科技研究人员具备编写高效的并行程序的能力，影响了并行机的应用水平。

将并行机制引入面向对象语言构成并行面向对象语言已成为当前面向对象技术的一个重要研究方向。利用面向对象的特点来有效实现并行任务的划分、数据的分配、并行进程间的通信和同步，可以降低并行程序设计难度。我们设计的适用于MPP机或集群系统的并行C^{++}语言(SPC^{++})，是一个面向对象的并行程序设计语言，充分体现了C^{++}语言面向对象的程序设计思想和特点，将对象机制和并行机制有机结合起来，大大降低并行程序设计难度。SPC^{++}利用了C^{++}语言的对象封装性、函数重载、操作符重载等功能对对象消息传递模型进行抽象，简化消息函数的使用。

可视化是指使用一些可视的表现形式来表达程序、数据、系统结构或系统的动态行为。在软件开发中，简便实用的图形用户界面改变了原有的程序设计模式，可视化程序设计成了必然趋势。目前串行机的可视化开发环境比较丰富，如VB、VC、C^{++} Builder等，可视化、组件化、速成化正推动着面向对象工艺的发展。相比较而言，并行环境的重点一直放在运行环境上，对编程环境的方便性和实用性注意不够，高性能并行、分布式程序设计语言难以得到推广的主要原因在于它们难于使用，如何将可视化、组件化、速成化等手段应用到并行软件开发是一个值得关注的问题，它们也代表未来并行软件开发的发展方向。基于SPC^{++}语言我们开发了一个可视化编程环境(POVPE)，该环境利用拖放等GUI手段进行快速应用开发，提供强有力的重用手段，大大提高科学计算并行应用程序的开发效率和设计质量。

我们在POVPE中，设计并实现了基于对象并行的可视化描述。紧密结合SPC^{++}语言、根据语言特点进行模型设计，并充分吸收市场上流行的串行可视化编程环境如Visual C^{++}、C^{++} Builder以及软件工程环境Rational Rose 2000等产品的特性及界面风格。本文先对SPC^{++}做简单介绍，然后介绍POVPE中基于对象并行的可视化描述的基本思想。

二、SPC^{++}语言简述

在SPC^{++}语言中，引入了并行对象概念。并行对象是程序的一个基本并行运行单位。并行对象封装了相对独立的数据和代码，对象内部一般通过共享对象进行通信，对象间通过消息传递相互作用。这样，一个节点或超节点可以抽象为一个并行对象，节点间的通信就抽象为并行对象间的通信。程序员创建一个并行对象，并给定相应构造函数参数，就可以在指定节点上加载并行任务，并激活该对象的Run方法函数。

在面向对象并行程序设计模型中，支持对象间和对象内二级并行。对象间的基本并行单位是并行对象，并行对象是一个主动对象，程序员创建一个并行对象，即自动加载一个节点，对象内的基本并行单位是并行对象的成员函数方法，由并行对象主方法Run控制对

*)本课题得到国防重点工程资助。黄永忠　讲师，研究方向为面向对象、软件工程、并行算法设计。李国巨　硕士研究生，研究方向为面向对象，并行算法设计，图形设计。郭金庚　教授，博士生导师，研究方向为面向对象，数据库。

象其它方法的运行。

并行类的说明与普通类的说明类似，只要在类说明关键字"class"前加上并行对象关键字"concurrent"即可。用户必须实现并行对象主函数run()，它的作用、功能与并行C程序中的main主函数相似，主要是创建节点共享对象、调用循环子程序和功能划分控制函数。

共享类是并行对象内部并行通信和数据分布的关键。它的使用方法和普通类基本一样，只要在类说明关键字"class"前加上共享类关键字"shared"和"dis_shared"即可。只能在共享类中定义分布变量，共享类的实例化必须在循环子函数撤开之前，实例化并行对象之后。

普通类可作为并行类和共享类的父类，或描述成它们的组成部分，通常都是用来做并行运算的服务类，提供一个串行的高效的算法。

例：求矩阵积的并行类定义。

(1)定义矩阵乘类 mat_mul：

```
concurrent class mat_mul
{
  matrix *map;
  matrix *mbp;
  matrix *mcp;
  public:
   mat_mul();
   void run();
   void matmul();
};
```

(2)共享矩阵类 matrix：

```
shared class matrix
{
  int array[M][N];
  public:
   matrix();
}
```

(3)矩阵乘类主方法：

```
void mat_mul::run()
{
  map=new matrix;//创建共享矩阵对象 map、mbp 和 mcp
    mbp=new matrix;
    mcp=new matrix;
  //与其它并行对象通信
    m_fork(matmul);//并行执行 matmul 方法
  }
```

(4)主程序

```
#include(parallel++.h)
main()
{
//创建并行对象 obj_mat，
//加载到含有8个 PE 的节点上，并启动执行 run 方法
  mat_mul  obj_mat("obj_mat",8);
//创建其它并行对象。
delete conobj;
}
```

三、对象并行的可视化描述

在POVPE中，图形区是一个简单的OOA/OOD工具，利用类图描述问题域中类及类关系，充分体现了面向对象的特点。类图是面向对象方法的核心技术，类图的应用十分广泛，其基本概念在许多地方都会经常用到。类图具有很强的表达能力，在串行环境中，我们可以很方便地使用类图对一个系统建模。但是在并行环境中，由于并行类、共享类的存在，产生了类关系的特殊性。

类是模型的主体，用户可以方便地建立一个类，并且可以从类库中选择合适的类作为这个新类的父类，也可以从主窗口的元件板上拖放已有的类。使用一个有边框、带类名的矩形表示一个类，类的属性和方法在矩形的内部，表示属性和方法是封装在这个类中的。

在类关系的描述上，参照已有面向对象方法中使用的类关系可视化表示方法，本着实用、简单的原则我们选择了三个常用的类关系：继承关系、通信关系、聚合关系。根据SPC++的语法，并行类、共享类、普通类之间存在如下关系：

1)*并行类和其它类的关系*　并行类和并行类之间只能有通信关系，因此用户不管选择哪一种类型的边，系统都会自动地纠正，描述为通信关系。在相应的并行类的源码中会自动加入通信语句。

并行类和共享类之间是聚合关系。因为共享类是并行算法中不可缺少的部分，共享类的方法实现和属性的分布方式往往决定了一个并行算法的具体实现，因此我们将共享类和并行类之间关系绑定为聚合关系。当用户连接并行类和共享类时，不管用户原来选择的是什么关系的边、箭头是什么方向，都会由系统自动地纠正过来，即箭头朝向并行类，边的关系为组成关系。代码区会自动地为用户在并行类中添加实例化与它相连的共享类的代码。对象名由系统自动为用户生成。

并行类和普通类之间有两种关系：继承关系和聚合关系。普通类和并行类之间连边时如果边是从并行类拉向普通类，并且选择边的关系为继承关系，那么普通类和并行之间会被描述为继承关系；其它各种情况均被认为是聚合关系。

2)*共享类之间*　除了通信关系外，基本上没有其它的关系。因此，连接共享类之间的边无论用户选择哪一种边，最终都被系统自动地纠正描述为通信关系。

3)*普通类之间*　没有限制，可以连接三种边的任何一种。在代码编辑区会为用户自动生成相应的代码。

4)*边的头节点为普通类，尾节点为共享类*　它们之间只能连接继承关系和通信关系。因为共享类不能作为普通的一部分，所以也就没有聚合关系。当用户连接关系错误时，系统会自动地把它纠正为继承关系。

5)*边的头节点是共享类，尾节点是普通类*　它们

（下转第94页）

证明:(1)显然。

(2)$\because P_-\subseteq Q_-$,$P_-\subseteq R$,由粗集的运算性质知$P_-\subseteq Q_-\cup R_-\subseteq(Q\cup R)_-$,$P_-\subseteq Q_-\cap R_-=(Q\cap R)_-$,$\therefore$ $Q\cup R$和$Q\cup R$都内依赖于P。

(3)$\because P_-\subseteq R_-$,$Q_-\subseteq R_-$,$\therefore(P\cap Q)_-=P_-\cap Q_-\subseteq R$,即$R$内依赖于$P\cap Q$。

(4)$\because P_-\subseteq R_-$,$Q_-\subseteq R_-$,$\therefore(P\cup Q)_-=P_-\cup Q_-\subseteq R$。证毕。

定理1给出了内依赖性的四条性质,性质(1)说明不确定性知识的核的扩充性可以传递,性质(2)则表明核的扩充还可以传播给知识的并和交,而性质(3)反映的是核的扩充可以进行交组合,但并组合一般不成立,只有在满足条件$(P\cup Q)_-=P_-\cup Q_-$时,才有性质(4)成立,下面的例子说明了这一点。

例4 条件同前,考虑$P=\{\omega_1,\omega_2,\omega_3,\omega_4\}$、$Q=\{\omega_4,\omega_5,\omega_6\}$、$R=\{\omega_{4,5},\omega_6\}$。$\because$ $P_-=\alpha_3$、$Q_-=\alpha_0\cup\alpha_1$,$R_-=\alpha_0\cup\alpha_1\cup\alpha_6$,$\therefore$ R内依赖于P、Q,显然R也内依赖于$P\cap Q$,但由$P\cup Q=\{\omega_1,\omega_2,\omega_3,\omega_4,\omega_5,\omega_6\}$知$\{P\cup Q\}_-=\alpha_0\cup\alpha_1\cup\alpha_2$,$\therefore$ R不内依赖于$P\cup Q$。

事实上,一般情况下只有$(P\cup Q)_-\supseteq P_-\cup Q_-$[4]。

关于外依赖性,我们有下面的性质。

定理2 设P、Q、R是知识系统$S=(U,T,C,\Gamma)$的三个不确定性的知识。

(1)若Q外依赖于P,R外依赖于Q,则R外依赖于P;

(2)若Q、R都外依赖于P,则$Q\cup R$外依赖于P;

(3)若Q、R都外依赖于P,且Q、R是正则关联的,则$Q\cap R$外依赖于P;

(4)若R既外依赖于P,又外依赖于Q,则R外依赖于$P\cap Q$和$P\cup Q$。

证明:(1)显然。

(2)$\because P^-\subseteq Q^-$,$P^-\subseteq R^-$,$\therefore$ $P^-\subseteq Q^-\cup R^-=(Q\cup R)^-$,$\therefore$ $Q\cup R$外依赖于P。

(3)$\because P^-\subseteq Q^-$,$P^-\subseteq R^-$,且由$Q$、$R$正则知$(P\cap Q)^-=P^-\cap Q^-$,$\therefore P^-\subseteq Q^-\cup R^-=(Q\cup R)^-$。

(3)$\because P^-\subseteq R^-$,$Q^-\subseteq R^-$,$\therefore(P\cap Q)^-\subseteq P^-\cap Q^-\subseteq R^-$,$(P\cup Q)^-=P^-\cup Q^-\subseteq R^-$,即$R$外依赖于$P\cap Q$和$P\cup Q$。证毕。

外依赖的性质(1)说明不确定性知识的可定义域的扩充性可以传递,性质(2)反映出可定义域的扩充还可以传播给知识的并,(3)指出可定义域的扩充可以传播给两个正则知识的交,而性质(4)则表明可定义域的扩充可以进行交、并组合。

结束语 上面给出了不确定性知识的精确性的数字特征,提出不确定性知识之间关联性的概念及其数量表示,还讨论了关联性的若干性质。从以上结果中可以看出,通过粗集来刻画不确定性知识,其本质已纳入了精确化的轨道,并且这种精确化是通过知识系统自身的信息来实现的,在此基础上展开的推理,即是一种精确化的操作,又便于机器实现,有关这方面的研究,我们正在进行。

参考文献

1 Pawlak Z. Rough set. International Journal of Information and Computer Science, 1982(11): 341～356
2 Pawlak Z. Rough set-theoreticsl aspects of reasoning about data. Dordrecht: Kluwer Academic Publishers, 1991
3 Pawlak Z, et al. Rough sets. Communications of ACM, 1995, 38(11): 89～95
4 曾黄麟. 粗集理论及其应用. 重庆大学出版社, 1998
5 胡涛, 吕炳朝, 等. 基于粗糙集的不确定知识表示方法. 计算机科学, 2000, 27(3): 90～92
6 刘清, 刘群. 各种不精确理论的 Rough 集解释. 计算机科学, 1999, 26(12): 5～8

(上接第120页)

之间只能连接聚合关系和通信关系。因为共享不能是普通类的父类,所以就没有继承关系。当用户连接错误时系统自动会纠正为聚合关系。

建立简明、准确的表示模型是把握复杂系统的关键,模型可以从全局上把握系统的概貌及其相关部件的关系。用户使用该模型可以为工程产生一个基本的代码框架,并且从整体上描述了组成工程的类和各个类的关系,并且为构架的重用提供了良好的基础。为了方便用户编程,我们采用了图形区和代码区之间可交互的方式,用户可以一边写程序代码,一边通过调整可视编辑区来不断完善工程中需要的新类和类的关系。

结束语 试用表明,并行程序设计不再是专业人员的专利,从事特定领域研究的非计算机科技人员运用 SPC^{++}语言特性及友好直观的可视化程序设计环境,快速生成高效的并行程序实际上是一件很轻松的事情。

丰富的类库是该环境推广的先决条件,如何抽取特定领域内的类 & 对象,形成该领域的类层次是下一步要完善的主要工作。

参考文献

1 周之英. 现代软件工程(中). 科学出版社, 2000
2 邱仲潘. UMLwith Rational Rose 从入门到精通. 电子工业出版社
3 Philippi. S Visual programing of concurrent object-oriented system. University of Koblenz-Landau
4 Giese H, Wirtz G. Visual Modeling of Object-Oriented Distributed System. http://wwwmath.uni-muenstrer.de/cs/u/versys/index.html