多态性 分类 特性 程序语言 (5)

计算机科学 1994 Vol. 21 №. 1

19-22 TP312

多态性的分类及其特性

杜 诚 郭浩志
(国防科技大学计算机系 长沙 410073)

摘要 This paper presents a detailed discussion on the concept of the polymorphism and an analysis on the polymorphically typed system using type abstraction method.

一、前言

关于多态性概念的论述最早见于文[5]。在文[4]中，R. Milner 具体地提出了多态类型原则和自动类型检查算法 ω。80 年人们在 ML 语言中实现了多态类型的思想，其后相继出现了 Miranda，Russell，Poly 等具有多态类型系统的语言。随着类型在程序设计语言中的地位越来越受到人们的重视，要求语言中有一个强有力的类型系统能够提供抽象机制以增强语言的表达能力，并利于编写出正确无误的程序[2]，而多态类型系统就是能够满足上述要求的一种系统。由于在多态语言(polymorphic language)中，值和变量，函数的参数以及其返回值可有多种类型，其类型可被定义成具有适于不同类型参数的操作的多态类型(polymorphic type)。因此，多态性思想允许同一实体在不同的环境中具有不同的行为解释，因而发挥出不同的功能，具有灵活有力的描述表达能力，支持数据抽象，利于方便自然地进行程序设计。多态性的具体表现形式多种多样，而且也不只局限于具有

杜诚 硕士．研究方向：计算机网络、软件工程。郭浩志 教授．研究方向：软件工程、程序语言。

明非定理，这些非定理带有旧知识库中缺少的信息，能使知识库的能力增强。在开放逻辑系统[6]中，R-重构和 N-重构都依赖于某种模型的存在。

七、自动构模技术

可满足性问题在一阶逻辑中是不可判定的，在命题逻辑中也是 NP-难的，因此自动构模很困难。对付这种困难有两类办法。一类是只考虑特殊的逻辑，比如说，命题 Horn 逻辑的可满足性问题有线性时间复杂度的算法。

另一类办法是只考虑有限模型。在域有限的情况下，通过穷举，总能判定可满足性。在实际系统中，有限模型的构造通常可看作约束满足(constraint satisfaction)问题。这需要将演绎和搜索技术结合起来(通常采用回溯搜索)[4]-[14]-[16]。目前，针对一阶理论的通用自动构模程序还不多，除了前面提到的日本的 MGTP 以外，还有澳大利亚的 FINDER (Finite Domain Enumerator)[11]。

八、结束语

模型的自动构造具有重要意义，在很多场合有着广泛的应用，同时也是个很难的问题，在理论上有不少反面的结论。有限模型的构造技术目前还不成熟，希望在不久的将来能有所突破，并推动相关领域的发展。(参考文献共 16 篇略)

多态类型系统的语言。深刻地剖析种种多态性机制，有利于我们更好地理解把握多态性思想的实质。

二、多态性的分类

传统的有类型语言（比如 Pascal）采用的是基于过程和函数的思想，这种语言中的操作数只能有一种类型，程序中的每个值和变量具有且仅具有一种类型。这样的语言称为单态语言（monomorphic language）。而在所谓的多态语言中，某些值和变量可以有多种类型。对于某些函数，它的操作数可以是多种类型的，这样的函数称为多态函数。对于一个类型，若它的操作适于多种类型的操作数，则称之为多态类型。在文 [3] 中，对多态性进行了分类，如图 1 所示。

多态性包括一般多态性和特殊多态性。一般多态性用来系统地刻画语言上相关的一组类型[8]。一般多态函数能够对无限数量的类型（所有类型均具有一给定的公共结构）进行操作。特殊多态性刻画语义上无关联的类型间的关系。特殊多态函数只能针对一组有限、不同、无关联的类型进行操作。对于一般多态性，可以确定某些值具有多态类型，而对特殊多态性，则很难确定。从某种意义上讲，一个特殊多态函数只是一组单态函数。从实现角度看，一个一般多态函数对不同参数类型的执行是用同一段代码，而一个特殊多态函数可能对不同的参数类型执行不同的代码。

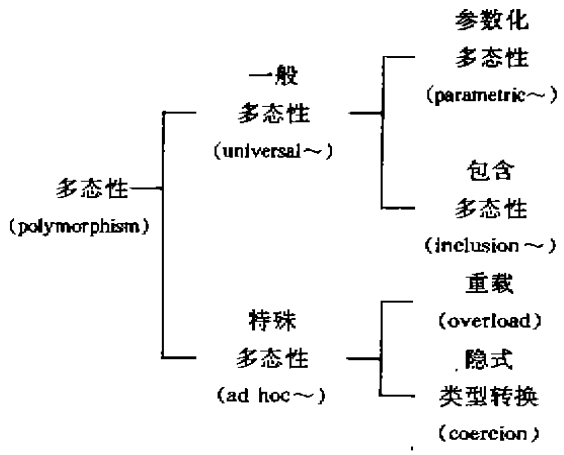

图 1　多态性分类图

一般多态性有两种主要形式：参数化多态性和包含多态性。在参数化多态性中，一个多态函数有一个隐式或显式的类型参数，该类型参数确定函数在每次执行时其中操作数的类型。这种函数也称为类属（generic）函数[9]。参数化多态指的是当一个函数统一地对若干类型参数操作时，这些类型表现出某些公共的语义特性，而该函数就是用来描述该特性的。包含多态性指的是子类型和继承。在包含多态性中，一个对象可以被看作属于不同的类，其间包含关系的存在意味着公共结构的存在。

特殊多态性也有两种形式：重载与隐式类型转换。重载[9]是指用一个名字表示不同的语义结构，如重载函数。在实现重载函数时要根据其所处上下文来确定该函数的同名实例以具体表示这个函数。从某种意义上讲，重载只是一种出于方便的语法上的省略。隐式类型转换则是一种语义操作，它被用来将一种类型转化为函数要求的另一种类型，从而防止类型错误。

应该说，一般多态性是真正的多态性；特殊多态性只是表面的多态性，因为重载只允许某一个符号有多种类型，而它所代表的值分别具有不同的、不相兼容的类型。类似地，隐式类型转换也不是真正的多态，因为在操作开始前，各值必须转换为要求的类型，而输出类型也与输入类型无关。相比之下，子类与继承却是真正的多态。例如图 2 中用 SIMULA 定义的子类 bus 的对象可以和父类 vehicle 的对象进行操作。在实现中不需要依赖隐式类型转换。参数化多态性也是一种纯正的多态，同一对象或函数在不同的类型上下文中统一地使用而不需采用隐式类型转换、运行时检测或其它各种限制。例如图 3 中给出的用 ADA 描述的类属过程 SWOP，以它为模板可生成针对某一数据类型的实例。

```
class vehicle
  begin
    integer licenseno;
    real weight, load, maxload;
  end
vehicle class bus (seating)
    integer seating;
    bus 的其它说明
  begin
    bus 的语句
  end
ref (vehicle) p1;
ref (bus) p2;
p1: -new vehicle;
p2: -new bus;
p1.load: =1000;
p2.load: =p1.load+1000;
```

图 2 子类与父类举例

```
generic
  type ITEM is private
procedure SWOP (x, y: in out ITEM);
procedure SWOP (x, y: in out ITEM) is   T: ITEM;
begin
  T: =x;
  x: =y;
  y: =T;
end  SWOP
declare
  package SWAP is new SWOP (INTEGER);
  I1, I2: INTEGER;
begin
  ...
  SWAP (I1, I2);
  ...
end;
```

图 3 类属过程举例

以上讨论的各种多态性在不同语言中均有所表现，ML[7]具有典型的参数化多态性，而 ADA[9]中的类属概念则是参数化多态性的特殊形式。SIMULA 的类概念是包含多态性的体现[7]。ALGOL68 则是具有隐式类型转换模式[7]。

三、多态性与类型抽象

分类是人们认识客观世界的一种模式[8]，类型概念就反映了这种模式。从这种意义上讲，类型划分是一种抽象机制。抽象是对系统或一组对象的简化描述，它突出该组对象的一些共性的、能反映本质的特性，而对其它的一些非共同且次要的细节暂时不予以考虑。类型抽象把变量的共同属性（值与操作）聚集到一个显式或隐式的说明中，用类型名来指称这些共同性质，并且保护内部表示，约束对象与对象之间的作用方式。这样做既有利于表示，又有利于维护程序的正确性，同时也提高了程序的可靠性与可读性。

这种抽象机制的抽象范围越广，层次越高，相应语言在表达能力，可靠性等方面获得的益处就越大，因为一个类型的性质在很大程度上决定了该类型变量的行为与使用。传统的单态语言局限于割裂式[1]的、孤立的类型划分，而且对类型间的相互关系、相互作用方式缺乏足够的表达能力。这造成了单态语言的局限性。一个单态类型系统将其对象约束成只具有一种行为，不允许各种值和语法上用来表达该值的符号在不同的上下文中表现出不同的行为。这些特性极大地降低了语言的表达能力。从这个角度看，多态类型系统是一个大大的进步。由于 Scott 卓越的工作[6]，类型本身可以象其它类型的值一样直接参与运算和处理，这样就利于我们站在更高层次上对类型的共性加以抽象。比如，包含多态性概括了类型的层次关系。类型化多态性有利于我们把握算法的本质特性。下面图 4 是用 ML 语言定义的表连接函数 append[7]。ML 多态系统根据定义式可得出 append 的类型表达式为：

('a list * 'a list) →'a list

这里'a list 为多态类型。这样定义就抓住了不同类型对象构成的表的连接操作的共性。可见多态类型更关心数据对象的高层结构特征。

```
append 定义式：
  ——fun append (nil, L) =L
    |append (hd:: tl, L) =hd:: append (tl, L)
类型表达式：
  >val append=fun 'a list * 'a list→'a list
```

图 4 append 函数定义

四、小结

多态类型语言表达能力强，支持程序正

计算机科学 1994 Vol. 21 №. 1

分布式系统负载分布研究综述

袁道华 编译

摘要 负载分布算法通过巧妙地重新分布系统中的工作负载，能够显著地提高分布式系统的性能。本文概述各种具有代表性的负载分布算法并比较了它们的性能。

低廉的微处理机以及通信技术的提高极大地推动了分布式系统的发展，这些系统的主要优点是高性能比、高可用性和可扩展性。然而，为了实现这些好处，系统设计人员还必须解决分布式系统中处理容量的分配问题，以达到最充分地发挥其优越性。

本文集中讨论了如何将系统负载巧妙且透明地在系统各节点中进行重新分布，以使整个系统的性能达到最佳。我们首先讨论通用系统中负载分布的几个关键问题，然后概述各种典型的负载分布算法，比较了它们的性能，并给出了哪一种算法可能有助于实现负载分布的大多数效益。

一、负载分布中的若干问题

我们首先讨论关于理解负载分布复杂性的几个至关重要的问题。文中，可以互换地使用术语：计算机，处理器，机器，工作站和节点。

1. 负载分布

分布式系统是由局部通信网络连接的一组自治计算机的集合，用户在其主计算机上提交处理任务。随机到达的任务可能引起某些计算机负载过重，而另一些计算机过轻甚至空闲。负载分布通过把任务从负载过重、服务贫乏的计算机转移到负载较轻的计算机，以使得任务能够利用那儿的计算能力。

设计动态负载分布算法的一个关键问题是标识一个合适的负载索引(Load Index)，它包含 CPU 队列长度，某个时期平均的 CPU 队

确性、可靠性、可维护性。对多态性问题的深入研究是程序语言设计中值得重视的一个方向。

参考文献

[1] John Backus. Can programming be liberated from the von Neuman style? A functional style and its algebra of programs. Comm. Acm 21 (8). 1978

[2] R. Barbuti. Static type checking for languages with parametric types and polymorphic procedures. International Symposium on Programming. Lecture Notes in Computer Science. 22-24 (April) 1980

[3] Luca Cardelli. Peter Wegner. On understanding types. data abstraction and polymorphism. Acm. Computer Surveys. Vol. 17 No. 4 1985

[4] Robin Milner. A theory of type polymorphism in programming. J. Computer and System Science. Vol. 17. No. 3 1978

[5] C. Strachey. Fundemental concepts in programming languages. Lecture Notes for International Summer school in Computer Programming. (Aug.) 1967

[6] D. S. Scott. Data type as lattics. SIAM J. Computer. 5. 3. 523—587

[7] 郭浩志主编.《程序设计语言概论》. 国防科技大学出版社. 1989. 6

[8] 杨芙清.《分类机制与面向对象程序设计》. 计算机工程. Vol. 18 No. 2. 1992

[9] 郭浩志.《ADA 程序设计基础》. 科技文献出版社重庆分社. 1989. 2