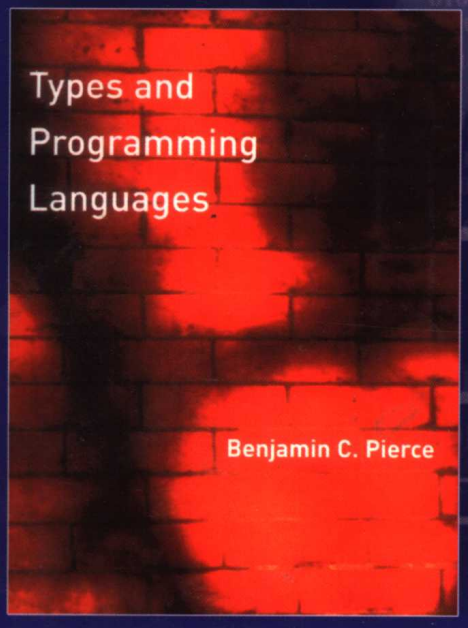
国外计算机科学教材系列

类型和程序设计语言

Types and Programming Languages

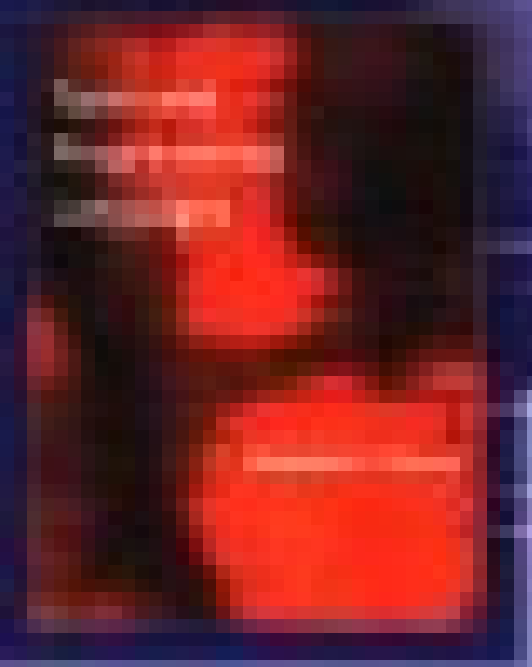


[美] Benjamin C. Pierce 著
马世龙 眭跃飞 等译
眭跃飞 审校

電子工業出版社
Publishing House of Electronics Industry
http://www.phei.com.cn

类型和程序设计语言

Types and Programming Languages



国外计算机科学教材系列

类型和程序设计语言

Types and Programming Languages

[美] Benjamin C. Pierce 著

马世龙 眭跃飞 等译

眭跃飞 审校

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内容简介

类型理论在程序设计语言的发展中起着举足轻重的作用，成熟的类型系统可以帮助完善程序设计本身，帮助运行系统检查程序中的语义错误。

要理解类型系统在程序设计语言中发挥的作用，本书将是首选读物。本书内容覆盖基本操作语义及其相关证明技巧、无类型 lambda 演算、简单类型系统、全称多态和存在多态、类型重构、子类型化、囿界量词、递归类型、类型算子等内容。本书既注重内容的广度，也注重内容的深度，实用性强。在引入语言的语法对象时先举例，然后给出形式定义及基本证明，在对理论的进一步研究后给出了类型检查算法，并对每种算法都给出了OCaml程序的具体实现。本书对类型理论中的概念都有详细的阐述，为读者提供了一个进一步理论学习的基础。本书内容广泛，读者可以根据自己的需要有选择地深入阅读。

本书适合从事程序设计的研究人员和开发人员，以及程序设计语言和类型理论的研究人员阅读。可作为计算机专业高年级学生、研究生的学习教材。

版权贸易合同登记号　图字：01-2002-6448

图书在版编目（CIP）数据

类型和程序设计语言 /（美）皮尔斯（Pierce, B. C.）著；马世龙，眭跃飞等译.
北京：电子工业出版社，2005.5
（国外计算机科学教材系列）
书名原文：Types and Programming Languages
ISBN 7-121-01149-2

Ⅰ. 类...　Ⅱ. ①皮... ②马... ③眭...　Ⅲ. 类型学（语言学）- 应用 - 程序语言 - 教材　Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2005）第 038577 号

责任编辑：李秦华
印　　刷：北京东光印刷厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
经　　销：各地新华书店
开　　本：787 × 1092　1/16　　印张：27.75　字数：710 千字
印　　次：2005 年 5 月第 1 次印刷
定　　价：58.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换；若书店售缺，请与本社发行部联系。联系电话：（010）68279077。质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

出 版 说 明

21世纪初的5至10年是我国国民经济和社会发展的重要时期，也是信息产业快速发展的关键时期。在我国加入WTO后的今天，培养一支适应国际化竞争的一流IT人才队伍是我国高等教育的重要任务之一。信息科学和技术方面人才的优劣与多寡，是我国面对国际竞争时成败的关键因素。

当前，正值我国高等教育特别是信息科学领域的教育调整、变革的重大时期，为使我国教育体制与国际化接轨，有条件的高等院校正在为某些信息学科和技术课程使用国外优秀教材和优秀原版教材，以使我国在计算机教学上尽快赶上国际先进水平。

电子工业出版社秉承多年来引进国外优秀图书的经验，翻译出版了“国外计算机科学教材系列”丛书，这套教材覆盖学科范围广、领域宽、层次多，既有本科专业课程教材，也有研究生课程教材，以适应不同院系、不同专业、不同层次的师生对教材的需求，广大师生可自由选择和自由组合使用。这些教材涉及的学科方向包括网络与通信、操作系统、计算机组织与结构、算法与数据结构、数据库与信息处理、编程语言、图形图像与多媒体、软件工程等。同时，我们也适当引进了一些优秀英文原版教材，本着翻译版本和英文原版并重的原则，对重点图书既提供英文原版又提供相应的翻译版本。

在图书选题上，我们大都选择国外著名出版公司出版的高校教材，如Pearson Education培生教育出版集团、麦格劳－希尔教育出版集团、麻省理工学院出版社、剑桥大学出版社等。撰写教材的许多作者都是蜚声世界的教授、学者，如道格拉斯·科默（Douglas E. Comer）、威廉·斯托林斯（William Stallings）、哈维·戴特尔（Harvey M. Deitel）、尤利斯·布莱克（Uyless Black）等。

为确保教材的选题质量和翻译质量，我们约请了清华大学、北京大学、北京航空航天大学、复旦大学、上海交通大学、南京大学、浙江大学、哈尔滨工业大学、华中科技大学、西安交通大学、国防科学技术大学、解放军理工大学等著名高校的教授和骨干教师参与了本系列教材的选题、翻译和审校工作。他们中既有讲授同类教材的骨干教师、博士，也有积累了几十年教学经验的老教授和博士生导师。

在该系列教材的选题、翻译和编辑加工过程中，为提高教材质量，我们做了大量细致的工作，包括对所选教材进行全面论证；选择编辑时力求达到专业对口；对排版、印制质量进行严格把关。对于英文教材中出现的错误，我们通过与作者联络和网上下载勘误表等方式，逐一进行了修订。

此外，我们还将与国外著名出版公司合作，提供一些教材的教学支持资料，希望能为授课老师提供帮助。今后，我们将继续加强与各高校教师的密切联系，为广大师生引进更多的国外优秀教材和参考书，为我国计算机科学教学体系与国际教学体系的接轨做出努力。

电子工业出版社

教材出版委员会

译 者 序

类型系统的研究，以及从类型论的角度研究程序语言已经是一个充满活力的研究领域，主要应用于软件工程、语言设计、高性能编译器的实现和安全。本书以一种很容易理解的方式介绍这一领域所涉及的基本定义、理论结果（从基本lambda演算、类型理论和程序语言理论）和实现技术。

现代软件工程广泛地使用形式化方法，确保系统按照隐式或显式的说明方式运行。这方面出现过一些过于理论化的框架，如Hoare逻辑、代数说明语言、模态逻辑和指称语义等。由于这些理论实用性不强，尤其难以被程序员所接受，所以出现了一些实用的技术，如将自动检查器放到编译器、连接器或程序分析器中的技术。至今最流行、最完善的轻量级形式化方法是类型系统，也正是作者在本书中讨论的主要内容。

作者将类型系统定义为“一种可行的语法工具，它可以根据计算值的种类对词语进行分类，从而证明某程序行为不会发生”。形式的定义可能会误导读者认为类型系统只是在理论上重要。事实上，类型系统已经成为程序语言设计的一个重要方面。类型系统为编译器提供了类型检查的手段，可以检测程序员犯下的由于类型错误而引起的语义错误，这样就可避免不规范行为的产生，如程序崩溃。我们在互联网上遇到的许多安全漏洞其实就是由于没有正确检测到类型错误而引起的。通常转而使用类型成熟语言的程序员开始时都会觉得十分惊讶，因为他们的程序无需额外的调试立即就能运行，可见类型系统有多么重要。

本书带领读者走进了一个关于类型系统设计、推理和实现的有趣领域。作者通过使用操作语义和OCaml语言举例（这些例子在书上提供的相关网站上都可以查到），将理论问题表达得清晰、易懂。

作者按照通常的方式介绍带有类型的程序语言中的语法对象，全书贯穿许多启发性例子、严格的定义、基本性质的证明、类型检查算法的描述、可靠性、完备性和可终止性的证明，并给出算法的具体实现。讨论的问题涵盖了lambda演算、简单类型、子类型、递归类型、多态和高阶类型。尽管强调的重点是函数式语言，如Haskell和ML的类型系统，但是也谈到了与理论发展有关的主流语言，如Java和C++语言之间的关系。本书理论与实践结合得非常紧密，对学习程序语言及培养语言设计人员来说帮助很大。从无类型系统到有类型系统，从简单类型到复杂类型，对各类语言特征都进行了详细地阐述，对命题和定理的证明也十分严谨。作者还开发了一套TinkerType系统，完成对书中所举的每个例子进行类型检查器的检查，并给出执行结果。该书如果有欠缺之处，那就是没有将流行的程序语言的特色、优点及缺点加以严格评论。

本书采用合适的排版式样和丰富的字体提高可读性，例如可以清晰地表达复杂冗长的lambda归约序列、操作语义和程序实例；同时给出了程序、规则和性质在表示上的差别。本书

包括不同层次难度的练习，并提供了详细的参考文献，以帮助高年级本科生、研究生以及相关学者学习和参考。初读本书的读者如果感到理解上有些困难，这并不奇怪。因为正确理解书中的内容需要读者具有一定的理论基础和程序设计的实践经验，融会贯通则更需要下一番功夫。

毫无疑问，本书是关于类型理论程序设计的重要著作，在此领域中的地位是无可替代的。

北京航空航天大学博士生导师马世龙教授及中科院计算所博士生导师眭跃飞教授组织博士研究生和硕士研究生对类型理论进行了长期的研究和讨论，在此过程中翻译了本书。其中主要翻译者有：前言及第 1 章到第 14 章由眭跃飞翻译，第 15 章到第 21 章由马丽和王婷翻译，第 22 章到第 26 章由马丽翻译，第 27 章到第 30 章由马骏翻译，第 31 章到第 32 章及附录由陈锬翻译。全书由马世龙教授统稿，眭跃飞教授审定。另外感谢毛俏琳在前期统稿中所做的工作。

由于译者水平有限，以及一些术语的翻译目前尚缺乏规范，错误之处望广大读者批评指正。

前　言

对类型系统的研究，以及从类型理论的角度研究程序语言已成为一种充满活力的领域，它们主要应用于软件工程、语言设计、高性能编译器的实现和安全。本书将对该领域的基本概念，研究成果及技术手段展开全面的阐述。

读者对象

本书针对两类读者：(1)从事程序语言和类型理论研究的研究生和研究人员；(2)希望了解程序语言理论中关键概念的计算机科学领域研究生和高年级本科生。对于前一类读者，本书覆盖了该领域的大部分内容，为读者深入研究提供了依据。对于后一类读者，提供了介绍性的材料和例子以及分析。本书既可以作为一般研究生水平的教材，也可以作为程序语言的高级研讨班的教材。

本书的目标

目标之一：在内容上力求覆盖一些核心概念，如基本操作语义及相关证明技巧、无类型lambda演算、简单类型系统、全称多态和存在多态、类型重构、子类型化、囿量词、递归类型、类型算子，以及许多其他问题的简短讨论。

目的之二：从语用的角度，注重程序语言中类型系统的使用，用大量的篇幅对一些可能包括在类型化lambda演算中的更精确问题(如指称语义)进行讨论。其中计算基础是值调用lambda演算，它与当前大部分语言相符，并且容易扩展为引用和异常等命令式结构。对每种语言的特征，关注它的实用价值、语言所含安全性的证明技巧，以及如何具体实现等方面的内容，尤其是类型检查算法的设计和分析。

目标之三：关注领域的多样性。本书覆盖了许多独立的问题和几个深刻理解的复合问题，但没有试图将所有的问题都溶入一个单一的系统中。只是对这些问题的子集提出了统一的表示方式，比如，虽然箭头类型变化多样，但在纯类型系统中均以统一的方式处理，由于整个领域发展太快，无法对其完全系统化。

本书在设计上充分考虑方便读者学习，无论是在课堂上还是以自学的方式。对大部分习题都给出了全面的解答。核心定义组织成一个图表以便于参考。概念和系统之间的依赖关系表现得尽可能清晰明了。在书的最后还提供了大量的参考书目。

最后，本书恪守诚实的原则。书中讨论的所有系统(除个别稍微提及的系统外)均有实现。每章都有从机器的角度检查所举例子的类型检查器和解释器。这些实现方式均可从本书的网站上获得，可用做编程练习，进行扩充实验以及作为更大的课堂报告内容。

为达到这些目标，其他的一些因素往往被忽略了。比如，最重要的一点就是内容的全面性。在一本书中想覆盖编程语言和类型系统的所有领域是不可能的，本书也不例外。本书只关注核心概念的详细讨论。还有，本书不考虑类型检查算法实际的效率，因为它并非一本关于如何实现具有工业力度的编译器或类型检查器的书籍。

本书的组织结构

本书的第一部分讨论无类型系统。在一个非常简单的仅含数字型和布尔型的语言中引入一些基本概念:抽象语法、归纳定义和证明、推论规则和操作语义,然后对无类型 lambda 演算重复引入这些概念进行讨论。第二部分考虑简单类型 lambda 演算和一些基本语言特征,如乘积、求和、记录、变式、引用和异常等。其中包括关于类型算术表达式一章引入了类型安全性的思想。还包括一章(可选学)用 Tait 的方法给出了简单类型 lambda 演算的规范化证明。第三部分讨论子类型的基本机制:包括一个元理论和两个扩展的实例研究。第四部分讨论递归类型,包含了简单同构递归和复杂相等递归形式。该部分的第 21 章讨论了在共归纳的数学框架中含相等递归和子类型的系统元理论。第五部分讨论多态性,包括 ML 形式的类型重构,功能更强大系统 F 和不可预言的多态性,存在量词及其与抽象数据类型的联系,带囿量词的系统中多态与子类型的组合,等等。第六部分讨论了类型算子。其中一章讨论基本概念;接下来的一章研究系统 F_ω 及其元理论;下一章将类型算子与囿量词结合起来提出系统 $F^{\omega}_{<:}$;最后一章为一个实例学习。

图 P.1 给出了章节之间的依赖关系。浅色箭头表示后一章部分依赖于前一章。

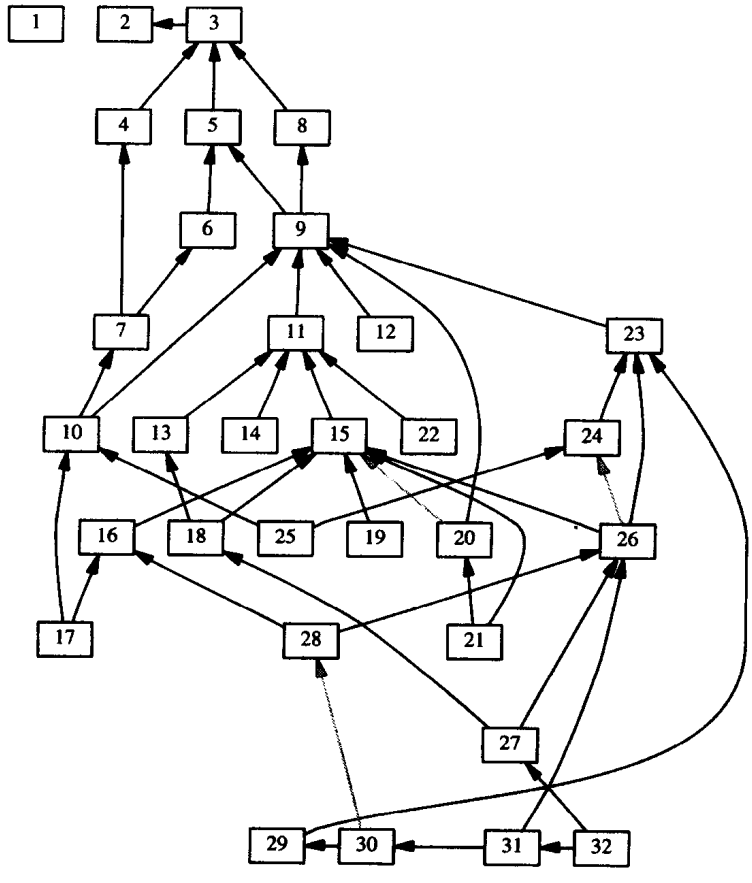


图 P.1 章节之间的依赖关系

本书对每个语言特征都采用统一的处理模式。先举一个例子,然后是形式定义;基本性质的证明,如类型安全性;通常另一章对理论进一步研究,得到类型检查算法及其可靠性,完备性

和终止性的证明；最后（在另一章中）给出这些算法的 OCaml 程序具体实现。

本书中的例子主要是对面向对象程序进行特征分析和设计的。有 4 个实例学习章节，详细地讨论了不同的内容：常规命令性对象和类的一个简单模型（参见第 18 章）；基于 Java 的一个核心演算（参见第 19 章）；用囿量词对命令性对象的一个更细致说明（参见第 27 章），以及用存在类型处理纯函数式的系统 $F^{\omega}_{<:}$ 中的对象和类（参见第 32 章）。

为使本书适合一学期的教材，必须忽略掉一些有意义和重要的论题。在本书中，指称和公理语义完全忽略掉了；类型系统与逻辑之间的联系在几个地方提到但没有详细讨论。程序语言和类型系统的许多新的特征只是提及而已，如依赖类型，交叉类型，以及 Curry-Howard 对应；只用了少量的篇幅对这些特征做了介绍以供日后有兴趣的读者自己深入学习。最后，除了简单介绍 Java 形式的核心语言外（参见第 19 章），本书完全注重基于 lambda 演算的系统；但是，这样处理的概念和机制可以直接转换到像类型并发语言，类型汇编语言，以及特别对象演算的相关领域中。

要求的背景知识

本书不要求有程序语言理论的背景知识，但读者应该具有一定的数学基础——特别是学过大学课程的离散数学、算法和基础逻辑等。

读者应该至少熟悉一种高阶函数式程序语言（如 Scheme，ML，Haskell 等）、程序语言和编译器的基本概念（抽象语法，BNF 语法，求值，抽象机器等）。这些在许多大学教材中可以找到；作者特别推荐 Friedman，Wand 和 Haynes（2001）的“Essentials of Programming Languages”和 Scott（1999）的“Programming Language Pragmatics”。本书中有几章以面向对象语言，如 Java 为例来说明一些问题十分有效。

为使本书成为浓缩的一学期的高级教材，且为减轻大部分研究生的负担，书中没有将许多有趣且重要的问题包括进来，也完全忽略了指称语义与公理语义，但有许多关于这方面的好书，可供查阅，如果在本书中将它们包括进来，会违背本书尽量实现的初衷。

关于类型检查器的具体实现中的几个重要概念是用 OCaml 语言编写的，其中 OCaml（Objective Caml 的缩写）为 ML 语言的另一个俗称。在这几章中介绍一些 OCaml 的预备知识对理解代码是有帮助的，但不一定有这个必要；本书中只用到了语言的一小部分。且有些特征在刚开始出现时就解释过了。所以这几章与本书的其他部分编写方式截然不同，完全可以略过。

目前关于 OCaml 的最好教材是 Cousineau 和 Mauny（1998）撰写的。随 OCaml 发布的资料也可在网上查阅（http://caml.inria.fr 和 http://www.ocaml.org）。

熟悉 ML 另一种称法——Standard ML 的读者可以很容易地理解 OCaml 代码。关于 Standard ML 的教材有 Paulson（1996）和 Ullman（1997）。

课程安排

一个中级或高级研究生的教材（一个学期）可以覆盖本书的大部分内容。图 P.2 给出了在 Pennsylvania 大学博士生高级课程的一个教学大纲（一周两次 90 分钟课时，假定以快速的方式覆盖了程序语言理论的最小预备知识）。

对于大学或初级研究生教材，有几个方案可以选择。程序中类型系统的课程可以注重引入各种类型特征和说明它们如何使用的章节，而忽略了大部分元理论和实现的章节。另外，基本

理论和类型系统实现的课程可以选择前面的章节，大概掠过第 12 章(以及第 18 章和第 21 章)，牺牲书后面的章节。较短课程可以选择依赖关系图 P.1 中的有关章节。

本书也是适合程序语言理论更一般的研究生课程的主要教材。这样一个课程可以花半个或三分之二个学期学习本书的主要部分，然后将剩下的时间用于学习一节基于 Milner(1999)的 pi 演算的并发理论，它介绍了 Hoare 逻辑和公理语义(如 Winskel,1993)或研究一下一些高级语言特征，如连续或模块系统。

课程	论题	阅读
1.	课程概论；历史；组织	1,(2)
2.	基础：语法，操作语义	3,4
3.	介绍 lambda 演算	5.1,5.2
4.	形式化 lambda 演算	5.3,6,7
5.	类型；简单类型的 lambda 演算	8,9,10
6.	简单扩展；导出类型	11
7.	更多的扩展	11
8.	规范化	12
9.	引用；异常	13,14
10.	子类型	15
11.	子类型的元理论	16,17
12.	指令性对象	18
13.	轻量级的 Java	19
14.	递归类型	20
15.	递归类型的元理论	21
16.	递归类型的元理论	21
17.	类型重构	22
18.	全称多态	23
19.	存在多态；*ADT*	24,(25)
20.	囿量词	26,27
21.	囿量词的元理论	28
22.	类型算子	29
23.	F_ω 的元理论	30
24.	高阶子类型	31
25.	纯函数式对象	32
26.	补充课程	

图 P.2　一个高级研究生课程的教学大纲样例

在本课程中，学期报告起了很大的作用，所以它可能会推延某些理论部分的学习(如规范化及某些元理论的章节)，在学生选择报告的内容之前可以阅读大量的例子。

练习

大部分章节包括许多练习——一些需要动笔，一些包含所讨论的演算程序，以及关于这些演算的 ML 实现。每个练习的困难程度用下面的符号表示：

★	快速检查	30 秒到 5 分钟
★★	容易	≤1 小时
★★★	中等	≤3 小时
★★★★	具有挑战性	>3 小时

标记为★的练习是用于对重要概念的实时检查。建议读者在读后面的内容之前在此稍微停一会，思考一下这些问题。在每个章节中，可作为课外作业的练习标记为“推荐”。

附录 A 给出了大部分练习的答案。标记↛ 表示练习的答案没有在附录 A 中给出。

本书约定

大部分章节以一种散漫的形式引入某类型系统的特征，然后在一个或多个图表中将推论规则集中起来对系统进行形式化定义。为参考方便，这些定义通常不仅包括目前还在讨论的新特征和新规则，同时也给出构建完整演算所需要的其余规则。

本书的另一特色是在编写过程中所有的例子都经过了类型检查器的检查，原稿经过仔细检查，例子被抽取出，产生并编译出包含所讨论的特征的类型检查器，并将其用于检查这些例子，然后将检查结果插入到文本中。完成这一艰巨任务的系统称为 TinkerType，是由 Michael Levin 和作者本人(2001)开发的，由(美国)国家自然科学基金 CCR-9701826，"Principled Foundations for Programming with Objects"和 CCR-9912352，"Modular Type Systems"所资助。

Web 资源

与本书相关的网站是：

http://www.cis.upenn.edu/~bcpierce/tapl

这个网站中的资源包括本书的勘误表，课程计划建议，读者提供的附加材料，以及每章演算的实现(类型检查器和简单翻译器)集合。

这些实现提供了本书中的例子和检查练习答案的一个环境。它们也可以用于阅读和修改，已成功地由参加作者课程的学生用做小实现练习和大课程计划的基础。实现是用 OCaml 编写的。OCaml 编译器可以免费在 http://caml.inria.fr 中下载，并可安装在大部分平台中。

本网站还为读者提供了类型论坛(Types Forum)，一个 E-mail 列表涵盖类型系统及其应用的所有方面。列表确保低容量和高信噪比。存档和预定指令在 http://www.cis.upenn.edu/~bcpierce/types中。

致谢

读后感到受益匪浅的读者应感谢 4 位导师——Luca Cardelli，Bob Harper，Robin Milner，以及 John Reynolds。他们教给作者大部分关于程序语言和类型的知识。

其余的内容是通过合作所学到的；除了 Luca，Bob，Robin 和 John 之外，在这些研究中的合作者还包括 Martín Abadi，Gordon Plotkin，Randy Pollack，David N。Turner，Didier Rémy，Davide Sangiorgi，Adriana Compagnoni，Martin Hofmann，Giuseppe Castagna，Martin Steffen，Kim Bruce，Naoki Kobayashi，Haruo Hosoya，Atsushi Igarashi，Philip Wadler，Peter Buneman，Vladimir Gapeyev，Michael Levin，Peter Sewell，Jérôme Vouillon 和 Eijiro Sumii。这些合作不仅形成了作者理解的基础，也激发了极大的兴趣。

本书的结构和组织是通过与 Thorsten Altenkirch，Bob Harper，John Reynolds 教学讨论修改的，加上了如下人员的评论和提出的错误：Jim Alexander，Penny Anderson，Josh Berdine，Tony Bonner，John Tang Boyland，Dave Clarke，Diego Dainese，Olivier Danvy，Matthew Davis，Vladimir Gapeyev，Bob Harper，Erik Hilsdale，Haruo Hosoya，Atsushi Igarashi，Robert Irwin，Takayasu Ito，Assaf Kfoury，Michael Levin，Vassily Litvinov，Pablo López Olivas，Dave MacQueen，Narciso Marti-Oliet，Philippe Meunier，Robin Milner，Matti Nykänen，Gordon Plotkin，John Prevost，Fermín Reig，Didier Rémy，John Rey-

nolds, James Riely, Ohad Rodeh, Jürgen Schlegelmilch, Alan Schmitt, Andrew Schoonmaker, Olin Shivers, Perdita Stevens, Chris Stone, Eijiro Sumii, Val Tannen, Jérôme Vouillon 以及 Philip Wadler(如果不小心漏掉某个人,将非常抱歉)。Luca Cardelli, Roger Hindley, Dave MacQueen, John Reynolds, 以及 Jonathan Seldin 对某些混乱不清的历史观点的权威看法。

在 Indiana 大学(1997, 1998)和 Pennsylvania 大学(1999, 2000)的研究生讨论班的人员认真阅读了本书的较早版本,根据他们的反映和评价,作者将本书写成现在这个样子。Bob Prior 和 MIT 出版社在出版过程的许多阶段引导修改手稿。

程序的证明对数学的社会过程来说太无聊而作用不大。

——Richard DeMillo, Richard Lipton, Alan Perlis, 1979

因此不要依赖于社会过程来验证。

——David Dill, 1999

形式方法只有到了不理解它们的人们都能使用时才会产生重大影响。

——Tom Melham

目　录

第二部分 简 单 类 型

第三部分 子 类 型 化