

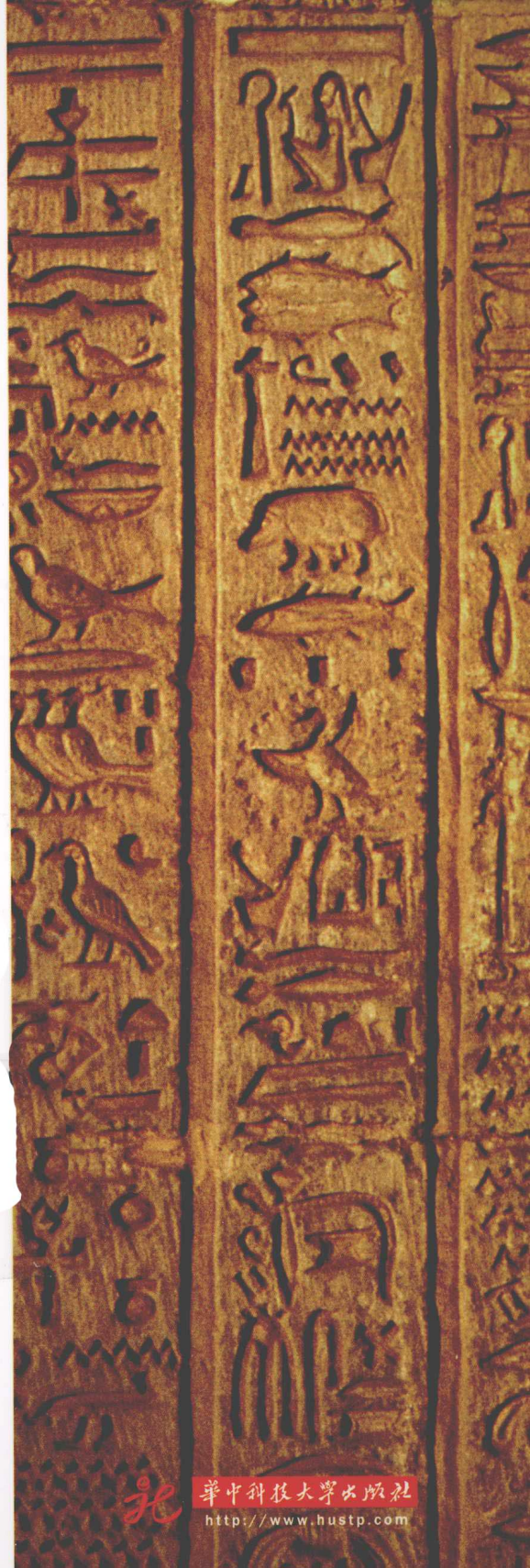
The
Pragmatic
Programmers

Language
Implementation Patterns

编程语言 实现模式

Create Your Own Domain-Specific
and General Programming Languages

[美] Terence Parr 著
李袁奎 译
尧飘海 审校



华中科技大学出版社
<http://www.hustp.com>

编程语言实现模式

[美] Terence Parr 著

李袁奎 译

尧飘海 审校

华中科技大学出版社

中国·武汉

内 容 简 介

本书旨在传授开发语言应用(工具)的经验和理念,帮助读者构建自己的语言应用。这里的语言应用并非特指用编译器或解释器实现编程语言,而是泛指任何处理、分析、翻译输入文件的程序,比如,配置文件读取器、数据读取器、模型驱动的代码生成器、源码到源码的翻译器、源码分析工具、解释器,以及诸如此类的工具。为此,作者举例讲解已有语言应用的工作机制,拆解、归纳出 31 种易于理解且常用的设计模式(每种模式都包括通用数据结构、算法、策略)。虽然示例是用 Java 编写的,但相信读者可以触类旁通,利用这些设计模式构建针对其他编程语言(既包括特定领域语言,也包括通用编程语言)的应用。

978-7-5609-7700-3: Language Implementation Patterns.

Copyright © 2011 The Pragmatic Programmers, LLC. All rights reserved.

湖北省版权局著作权合同登记 图字:17-2011-198 号

图书在版编目(CIP)数据

编程语言实现模式/(美)Terence Parr 著;李袁奎 译;尧飘海 审校. —武汉:华中科技大学出版社,2012.4

ISBN 978-7-5609-7700-3

I. 编… II. ①T… ②李… ③尧… III. 程序语言 IV. TP312

中国版本图书馆 CIP 数据核字(2012)第 011104 号

编程语言实现模式

[美]Terence Parr 著 李袁奎 译 尧飘海 审校

策划编辑:徐定翔

责任校对:周 娟

责任编辑:熊 慧 江 津

责任监印:周治超

出版发行:华中科技大学出版社(中国·武汉)

武昌喻家山 邮编:430074 电话:(027)87557437

录 排:华中科技大学惠友文印中心

印 刷:湖北新华印务有限公司

开 本:787mm×960mm 1/16

印 张:24.5

字 数:428 千字

版 次:2012 年 4 月第 1 版第 1 次印刷

定 价:72.00 元



本书若有印装质量问题,请向出版社营销中心调换
全国免费服务热线:400-6679-118 竭诚为您服务
版权所有 侵权必究

读者对本书的赞誉

别看那些编译原理的书了！这本书教你编写真正实用的解析器、翻译器、解释器等语言应用，Terence Parr 在书中细致地讲解了先进的语言工具和语言应用中设计模式的用法。无论是编写自己的领域专用语言（DSL），还是挖掘已有代码、查错或是寻宝，你都能从这本简单易懂的书中找到示例和模式，因为它基本上覆盖了解析技术的方方面面。

► Guido Van Rossum

Python 语言之父

我的“龙书”被打入冷宫了！

► Dan Bornstein

Android 平台 Dalvik 虚拟机的设计者

本书对每个语言设计者来说都是一笔无价的财富。

► Tom Nurkkala 博士

泰勒大学计算机科学系副教授

Terence 清晰地阐释了语言设计中的概念。如果你想独创一门语言却又无从下手，或者觉得它高不可攀，那么，从这本书开始吧！

► Adam Keys

<http://therealadam.com>

这本书行文风格浅显却又不失韵味，以这个经久不衰的热门话题为中心，娓娓道来，颇有大师风范。《编程语言实现模式》不光讲述创造语言的方法，还指引我们在这个过程中该思考些什么。要想创建一个强壮的、可维护的专用语言，这本书是无价之宝。

► Kyle Ferrio 博士

Breaulty 研究机构科学软件开发部门主管

致谢

Acknowledgments

首先，要感谢我的编辑，Susannah Pfalzer。8 个月来，我们多次讨论，不断尝试，才最终确定了这本书现在的样子。这本书得以出版，她功不可没。

然后，要感谢这本书的审校者（排名不分先后）：Kyle Ferrio、Dragos Manolescu、Gerald Rosenberg、Johannes Luber、Karl Pfalzer、Stuart Halloway、Tom Nurkkala、Adam Keys、Martijn Reuvers、William Gallagher、Graham Wideman 和 Dan Bornstein。Wayne Stewart 虽不是正式的审校者，但也通过本书的勘误网站提供了大量的反馈信息。Martijn Reuvers 为本书的源代码编写了 ANT 配置文件。

这里要再次感谢 Gerald Rosenberg 和 Graham Wideman，他们给书稿提供了细致的审查意见，还通过电话跟我沟通，对书中的不足提出了诚挚的批评并进行了探讨。

序言

Preface

随着你不断编写语言应用，这个过程中所蕴涵的模式就会逐渐变得清晰而明朗。其实，大多数的语言应用在架构上都是相似的。每次编写语言应用的时候，我都不断告诉自己：“先建立解析器，用它在内存中把数据结构建立起来。然后从中抽取信息，必要时还要改变其结构。最后再写一个能根据这些信息自动输出代码或者报告的工具。”看吧，这不就是模式？在这些任务中总能发现一些相似的算法和数据结构。

一旦掌握了这些语言实现的设计模式或者架构，编写起语言应用来就得心应手了。如果你想快速获得编写语言应用的能力，这本书正适合你。本书奉行实用主义，从本质上挖掘并提炼语言应用中的设计模式。你会了解模式的重要性，学习如何实现这些模式，如何组合这些模式。很快你就能成为开发语言应用的行家里手！

创造新的语言其实不需要深厚的理论知识做铺垫。你可能不信，毕竟所有语言应用方面的书都会占用大量的篇幅讲解编译器知识。我承认，为通用编程语言编写编译器确实需要扎实的计算机科学知识，然而，大多数程序员并不需要编写这种编译器。因此，本书的重心是解决程序员平时最可能遇到的问题：配置文件读取、数据读取、模型驱动的代码生成、源代码之间的翻译、源代码分析和解释器的实现。同理，我们没有使用 Scheme 等学术界推崇的语言，而是跟随业界的发展采用 Java 编写所有的示例，以便你能快速地在实际项目中大显身手。

本书讲什么

What to Expect from This Book

本书讲解的工具和技术能满足日常语言应用开发的需要。对于那些相当棘手、艰深的问题，我们不予讨论。比如，因为篇幅有限，本书无法涉及机器码的生成、寄存器分配、垃圾回收、线程模型及苛求性能的解释器。读完本书后，你将会成为编写语言应用的专家，而对于复杂些的语言处理和转换，相信你也能应付。

本书将剖析现存语言应用的工作原理，当然这是为了抛砖引玉，最终还是希望你能编写自己的语言应用。为此，将它们分解成一些易懂的具有普适性的模式。不过，有一点必须说明，本书不是语言实现方面的代码库，而是一本旨在辅助你自己学习的工具书。虽然书中有不少示例代码，但这些代码只是为了让我们的讨论言之有物，以便你在编写自己的语言应用时有个拿得出手的起点。

还有一点提请注意，本书着重分析现有的成熟的语言（也可以是你自己设计的但与之相仿的语言）。因为语言设计是另一个话题，它强调的是规划语法（合法的语句）和制定语义（任意输入的含义）两个内容。虽然本书不会专门探讨语言的设计，但读完后，在耳濡目染中你也会了解相关的知识。学习语言设计的诀窍就是要审视各种不同的语言，研究程序设计语言的发展历史，熟谙语言随时代变迁的历程。这些都对学习语言设计大有好处。

语言应用不仅仅包括实现语言的编译器或解释器。广义来说，能够处理分析和翻译输入文件的程序都是语言应用。实现一门语言是指，编写一个应用程序，它能够根据以这种语言书写的语句来执行任务。对于一门语言来说，除了这个，我们其实还能做很多事情。比如，根据C语言的定义，能编写C的编译器、从C到Java的翻译器、自动插桩C代码以定位内存泄露的工具。类似地，回想Eclipse集成开发环境中的各种工具，除了Java的编译器，还有能对Java代码进行重构、格式调整、搜索和语法高亮的工具。

书中的模式可以用来为任何计算机语言编写语言应用，其中自然也包括领域专用语言（domain-specific language, DSL）。领域专用语言是指为特定领域专门设计的语言，它可以帮助用户在此领域中高效完成任务。比如，Mathematica 语言、shell 脚本语言、wiki 标记语言、UML、XSLT、makefiles、PostScript、形式语法，甚至连 CSV 和 XML 这样的数据格式也算。与 DSL 相对应的是通用编程语言（general-purpose programming language, GPPL），比如，C、Java 和 Python。DSL 通常会更小巧轻捷些，因为它们面向较为单一的应用环境。当然也有例外，比如 SQL 就比一般的 GPPL 大不少。

本书的组织

How This Book Is Organized

本书分为四个部分。

- 解析起步：首先熟悉语言应用的整体框架，然后深入学习更重要的语法解析模式。
- 分析语言：为了分析某种语言（不管是 DSL 还是 GPPL），得用解析器构造树形结构来表示这种语言。之后通过对树的遍历，能跟踪和识别输入内容中的各种符号（包括变量名和函数名），还能计算表达式的返回类型（比如 int 和 float）。这部分的模式主要介绍怎样确保输入是合法的。
- 构建解释器：这部分包含 4 种不同的解释器模式。它们在效率和实现难度上各有差异。
- 翻译和生成语言：本书的最后一部分会讲解语言的翻译，主要是如何借助模板引擎 `StringTemplate` 生成文本。第 13 章还会介绍几个语言应用的架构，以供你在编写语言应用时作为参考。

各个部分中的篇章有序编排过，能让读者从前往后循序渐进，一步步实现自己的语言应用。1.2 节“模式概览”里讲解了如何把所有模式组合到一起。

书中的模式

What You'll Find in the Patterns

书中共有 31 种模式。每一种模式都描述了与语言应用相关的数据结构、算法、处理策略。每种模式由四部分构成。

- 目的：这部分简要叙述模式的存在意义，指出它能解决的问题。比如，模式二十一“自动类型提升”能够“自动地、安全地提升算术操作数的类型”。在阅读模式的具体内容之前最好先看看这部分，以便心里对它的功能有个底。
- 讨论：这部分会进一步讨论所要面临的问题，解释如何应用模式，以及模式如何解决问题。
- 实现：每种模式都附带用 Java 编写的示例（可能会借助 ANTLR 之类的语言工具）。示例不是解决实际问题的代码库，只是用代码来证明前面谈论的内容。
- 相关模式：这部分列出一些模式，它们要么能解决本模式所解决的问题，要么是本模式所依赖的模式。

每一章的导言材料都值得一读，它们勾勒出不同模式的特性并加以比较，能加深读者对各种模式的认识。

本书面向的读者

Who Should Read This Book

如果你是软件从业人员或者计算机系的学生，想编写处理计算机语言的程序，就应该看看这本书。这里的计算机语言是泛指，从数据格式、网络协议、配置文件、特殊的数学语言及硬件描述语言到通用编程语言都在其中。

阅读本书并不需要形式语言理论方面的知识，但是最好拥有丰富的编程经验，不然就难以理解书中的代码和讨论的内容。

还有，递归这个概念很重要，要想读懂本书，必须熟悉这种思维方式。书中的许多算法和流程从本质上来说都是递归的，不管是输入的识别、遍历语法树，还是产生输出的解释器，都离不开递归。

如何阅读本书

How to Read This Book

如果你在这方面还是个新手，请从第 1 章“语言应用初探”开始，这一章概括语言应用的架构。接下来是第 2 章“基本解析模式”和第 3 章“高阶解析模式”，从中你能了解（形式语言描述的）语法和语言识别的背景知识。

如果通过某些计算机课程，你已经掌握了这些，那就可以直接跳到第 4 章“从语法树构建中间表示”或者第 5 章“遍历并改写树形结构”。如果你对构造语法树也很熟悉，对这种遍历过程已得心应手，那么这些也可以跳过，但是其中的模式十四“树文法”和模式十五“模式匹配器”这 2 种模式还是值得一看的。

如果你会编写简单的语言应用，知道怎样把输入内容组织成树的形式并对其遍历，那么你可以直接跳到第 6 章“记录并识别程序中的符号”和第 7 章“管理数据聚集的符号表”，这里面介绍了如何构造符号表。符号表主要用于描述某种映射关系，对于输入符号 x ，它能告诉你 x 到底是什么东西。符号表是第 8 章“静态类型检查”中某些模式所依赖的数据结构。

更熟练的读者可以直接跳到第 9 章“构建高级解释器”和第 12 章“使用模板生成 DSL”。如果连这些都懂了，那么你可以任意翻阅此书，从书中寻找自己想深究的模式。如果一时不想细看，也可以直接从书中某种模式中找出示例作为参考，自己编写新语言（需要时再从书中查找具体的解释）。

读者可以通过本书的网站¹或者 ANTLR 用户邮件列表²，与我或其他读者

¹ <http://www.pragprog.com/titles/tpdsl>.

² <http://www.antlr.org/support.html>.

进行交流。

通过本书的网站还能提交勘误信息、下载源代码。

书中采用的语言和工具 Languages and Tools Used in This Book

书中的代码和示例都是用 Java 编写的，但是其中的逻辑放到其他通用语言上也适用。只选用一种语言是为了前后保持一致，选择 Java 自有道理，因为它是业界最常用的语言。^{3,4}无论如何，这本书还是以设计模式为重心的，而不是语言宝典。模式的示例也无法直接用到实际项目中。

本书尽量采用最先进、最高效的语言和工具。比如，在解析语言时，会采用自动生成解析器的工具（当然，得先学会手工编写解析器）。使用自动工具之前，得了解解析器的工作原理。这就好像不可能让不懂算术的人使用计算器一样。同样，我们还要先学会手动编写语法树的遍历器，接下来再使用相应的自动生成工具。

本书常会用到 ANTLR 工具。它能自动生成解析器和语法树的遍历器。该工具是我编写的，凝结了过去 20 年里编写语言应用的心得。当然用其他的工具也行，但既然在写自己的书，就没有道理不用自己的工具。不过这本书的主角却不是 ANTLR，本书要讲的是语言应用中具有普适性的种种模式，具体的代码只是为了辅助理解。

本书的第 12 章还大量使用模板引擎 `StringTemplate` 来生成输出。`StringTemplate` 就好比“逆解析器”的生成工具，模板相当于输出内容的语法规则。如果不用模板引擎，也可以用一堆不规整的嵌有输出语句的代码来完成，但其中的逻辑就没有那么清晰了⁵。

即使对 ANTLR 和 `StringTemplate` 不甚了解，也不会影响对这些模式的学习效果，因为它们只在示例代码中出现。当然，要想完全掌握这些模式，确实应该看懂这些代码。

3 <http://langpop.com>.

4 <http://www.tiobe.com/index.php.content/paperinfo/tpci/index.html>.

5 译注：类似于用 JSP 代码生成 HTML 文档。

想看懂代码，最好多学学 ANTLR 项目中的工具。看了 4.3 节，你会对其有个大概的了解。想进一步深入学习这些工具，还可以访问网站上的文档和示例，或者购买《ANTLR 权威指南》(Par07)。

无论怎样，要实现语言就需要一些工具。这本书中的知识总能延伸到其他工具上，就好像学习飞行技术一样，不管怎样，都得随便找一架飞机先开着，学到的技术自然对其他飞机也适用。学习的关键在于掌握飞行技巧，而不是去纠结飞机上某个坐垫上的纹理图案。

希望这本书能够激起你对语言应用的兴趣，帮助你设计顺手的 DSL，写出提高程序员效率的语言工具。

Terence Parr

2009 年 9 月

parrt@cs.usfca.edu



目录

Contents

致谢.....	vii
序言.....	ix
第 1 部分 解析起步.....	1
第 1 章 语言应用初探.....	3
1.1 大局观.....	3
1.2 模式概览.....	5
1.3 深入浅出语言应用.....	9
1.4 为语言应用选择合适的模式.....	17
第 2 章 基本解析模式.....	21
2.1 识别式子的结构.....	22
2.2 构建递归下降语法解析器.....	24
2.3 使用文法 DSL 来构建语法解析器.....	26
2.4 词法单元和句子.....	27
模式一 从文法到递归下降识别器.....	29
模式二 LL(1)递归下降的词法解析器.....	34
模式三 LL(1)递归下降的语法解析器.....	38
模式四 LL(k)递归下降的语法解析器.....	43
第 3 章 高阶解析模式.....	49
3.1 利用任意多的向前看符号进行解析.....	50
3.2 记忆式解析.....	52
3.3 采用语义信息指导解析过程.....	52
模式五 回溯解析器.....	55

模式六 记忆解析器.....	62
模式七 谓词解析器.....	68
第 2 部分 分析语言	71
第 4 章 从语法树构建中间表示	73
4.1 为什么要构建树.....	75
4.2 构建抽象语法树.....	77
4.3 简要介绍 ANTLR.....	84
4.4 使用 ANTLR 文法构建 AST	86
模式八 解析树.....	90
模式九 同型 AST	94
模式十 规范化异型 AST	96
模式十一 不规则异型 AST	99
第 5 章 遍历并改写树形结构	101
5.1 遍历树及访问顺序.....	102
5.2 封装访问节点的代码.....	105
5.3 根据文法自动生成访问者	107
5.4 将遍历与匹配解耦.....	110
模式十二 内嵌式遍历器.....	113
模式十三 外部访问者.....	116
模式十四 树文法.....	119
模式十五 模式匹配器.....	123
第 6 章 记录并识别程序中的符号	131
6.1 收集程序实体的信息	132
6.2 根据作用域划分符号	134
6.3 解析符号	139
模式十六 单作用域符号表.....	141
模式十七 嵌套作用域符号表.....	146
第 7 章 管理数据聚集的符号表	155
7.1 为结构体构建作用域树	156

7.2 为类构建作用域树	158
模式十八 数据聚集的符号表	161
模式十九 类的符号表	167
第 8 章 静态类型检查	181
模式二十 计算表达式类型	184
模式二十一 自动类型提升	193
模式二十二 检查类型安全	201
模式二十三 多态类型检查	208
第 3 部分 解释执行	217
第 9 章 构建高级解释器	219
9.1 高级解释器存储系统的设计	220
9.2 高级解释器中的符号记录	222
9.3 处理指令	224
模式二十四 语法制导解释器	225
模式二十五 基于树的解释器	230
第 10 章 构建字节码解释器	239
10.1 设计字节码解释器	241
10.2 定义汇编语言语法	243
10.3 字节码机器的架构	245
10.4 如何深入	250
模式二十六 字节码汇编器	252
模式二十七 栈解释器	259
模式二十八 寄存器解释器	267
第 4 部分 翻译和生成语言	277
第 11 章 语言的翻译	279
11.1 语法制导的翻译	281
11.2 基于规则的翻译	282
11.3 模型驱动的翻译	284

11.4 创建嵌套的输出模型	292
模式二十九 语法制导的翻译器	296
模式三十 基于规则的翻译器	302
模式三十一 特定目标的生成类	308
第 12 章 使用模板生成 DSL	313
12.1 熟悉 StringTemplate	314
12.2 StringTemplate 的性质	317
12.3 从一个简单的输入模型生成模板	318
12.4 在输入模型不同的情况下复用模板	321
12.5 使用树文法来创建模板	324
12.6 对数据列表使用模板	331
12.7 编写可改变输出结果的翻译器	337
第 13 章 知识汇总	349
13.1 在蛋白质结构中查找模式	349
13.2 使用脚本构建三维场景	350
13.3 处理 XML	351
13.4 读取通用的配置文件	353
13.5 对代码进行微调	354
13.6 为 Java 添加新的类型	355
13.7 美化源代码	356
13.8 编译为机器码	357
参考书目	359
索引	361