《编程语言实现模式》

图书基本信息

书名：《编程语言实现模式》

13位ISBN编号：9787560977003

10位ISBN编号：7560977006

出版时间：2012-3-20

出版社：华中科技大学出版社

作者：Terence Parr

页数：388

译者：李袁奎,尧飘海

《编程语言实现模式》

前言

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。如果你想快速掌握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。

《编程语言实现模式》

内容概要

《编程语言实现模式》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。

《编程语言实现模式》

作者简介

Terence Parr是美国旧金山大学的计算机教授、研究生导师，他一直致力于从事ANTLR项目（antlr.org）和模板引擎（stringtemplate.org）的设计和开发工作。Terence曾担任IBM、洛克希德马丁、NeXT、雷诺汽车等公司的技术顾问，另著有《ANTLR权威指南》。

《编程语言实现模式》

书籍目录

《编程语言实现模式》

媒体关注与评论

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，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。 ——Python语言之父Guido Van Rossum‘ 我的“龙书”被打入冷宫了！ ——Android平台Dalvik虚拟机的设计者Dan Bornstein 本书对每个语言设计者来说都是一笔无价的财富。 ——泰勒大学计算机科学系副教授Tom Nurkkala博士 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 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。要想创造一个强壮的、可维护的专用语言，这本书是无价之宝。 ——Breaulty研究机构科学软件开发部门主管Kyle Ferrio博士

《编程语言实现模式》

编辑推荐

虽然《编程语言实现模式》不专门讨论如何设计编程语言，但读者在阅读的过程中将吸收丰富的相关知识。比较不同编程语言的特点、了解编程语言的发展历史是学习设计编程语言的好途径。

《编程语言实现模式》

精彩短评

1、随着java版本的更新,dsl在java中得到越来越多的重视平时开发中,也有许多思想可以借鉴书翻译的还算可以
2、读完咯~~~不容易
3、:
TP312/4444-19
4、9分
5、给初学者打开眼界和获取灵感
6、帝老师说好
7、这是一本实战型的书，没有太多的原理，没有太多的理论，适合实战派。
8、4星是评给原书的，译本最多2星。翻译和排版感觉都太随便了点，定价居然高达72，还好我是从图书馆借的……
9、大神们说有点民科，不过对我还帮助还是很大的
10、简单易懂的编译原理入门书，有趣，翻译不大好，应用性有限
11、yet another antlr reference book
12、这本书不是初学者阅读，是那些有了一定的开发经验，有某方面的编程语言基础。对于编程习惯和代码质量有很高地提升价值
13、一本很棒的讲述编译原理的书，最重要的是，从中看到了无限的可能性。
14、《编程语言实现模式》顾名思义，推荐指数A，难度系数B。推荐理由：作为码农或是准码农，搞出一些DSL来简化重复工作已经是大势所趋了，不借助工具单凭手工堆代码是不可能比平均水平高一个数量级的
15、精品的书要看看
16、深入浅出的入门书，对我帮助真的很大，好久没看到这么好的书了
17、但仍然远远输给python源码剖析了。
18、算是比较应景的阅读，主要讲编译器前端实现。正好最近在做SQL parser相关的事情。本书完全是从实用角度出发，没有太多的理论。主要采用antlr，所以到后面理解起来有些许障碍。后面1/3的翻译感觉不是那么顺畅，而且有不少印刷错误。如果能够透彻理解本书，绝对会成为一大助力。
19、如果实现的是领域专属语言的话，不必看这个了。
20、ANTLR大法好。
21、很棒的书，偏实践。配合antlr使用，但是书中的代码是antlr3，与antlr4有相当的区别。
22、帮助我理清语言开发的设计思路。
23、内容还是不错的 但是感觉过多的篇幅介绍ALTER的工具。
24、唔....通读了一遍，真的是好书，但后半段解释器实现和代码生成还没完全看懂
25、编程语言是怎样炼成的，及其一些模式化思考
26、这本书需要不少基础知识，没那么容易学习。
27、虽然不算正统的学习编译原理, 至少现在让我处理一些语言语法分析, 代码翻译上的技能缺陷补齐了.
28、模式模， 模式式，模式你，模式妹
29、更像ANTLR的使用手册，适合用来构建编译器前端时，作为参考的书籍。
30、书不错，实践性偏强。唯一不爽的是，里面用ANTLR描述文法。这玩意儿之前没用过，vm那块感觉仅仅是入门介绍，希望的JIT没出现・・の。
31、看这本书需要先熟悉antlr的用法，否则读起很多例子看不懂。
32、帮同事买的，书籍印刷还不错
33、很好很好的书。没话说。
34、编译原理的另一种讲解
35、正在学习DSL，这本书非常有帮助。
36、正版书，价格便宜，值得入手
37、经典

38、关于编译器和解释器实现的书
39、这是一本对语言实现需求者很好的应用书
40、原来语言的虚拟机就是将字节码翻译成机器码的解释器。书里面代码太多实践性太强，感觉没龙书好读
41、不再看了
42、没全看完，不过中文版不想再看了。差评它只是因为翻译问题，原书很好。关键是所有的代码都可用!
43、书被压皱了啊！！以前都是纸箱子包好送来的 这次怎么就一塑料袋子
44、如题, 好书一本 可以喝编译原理类的书一起看看
45、本书深入浅出的讲解了如何编写语言应用，作者是ANTLR的开发者，也是Lex 和 YACC的另外JAVA实现，对语言实现需求者是一本很好的应用书，值得期待。使用 Antlr 等识别工具来识别，解析，
46、Java 根本不能表述问题。
47、给antlr打广告的，看来我并不需要这本书，居然会被“ 不需要其他编译原理书 ”这样的宣传语吸引来看，惭愧。
48、适合我这种外行看咯。
49、0. 整体浅显易懂（前置条件：我上过编译原理课，用的是大抄龙书的李文生《编译原理与技术》）；ANTLR 使得可操作性极强。
1. 看完书的大部分去实战，发现「树文法」和内置的对 AST 的实现 在 ANTLR 4 里已经没有了。于是去看了 "The Definitive ANTLR 4 Reference" 的相关部分。
2. 有些段落，不知是作者还是译者原因，比较缺逻辑连接词，造成「懂的人才能看懂」……
50、好书，值得慢慢看
51、细读和体会最重要，翻译部分有些瑕疵。
52、入门级读物的中文版，适合做数据处理的用户。
53、实用主义的龙书
54、大牛们都推荐了，你还等啥
55、非常满意，包装的很好，发货也很快
56、貌似是将编译与设计模式相结合的，值得一看
57、相比龙书这类理论派的书, 更加注重工程化的实践. 书里应该是基于antlr v3的, 期待第二版>_<
58、书相当好，名词解释的很清楚。可惜，个人智商问题，导致看不明白
59、必备知识吼,但是,要配套另外两本才真心实用的起来,这本是基本地图,ANTLR 是装备,DSL 语法设计图书则是藏宝图了...
60、是一本好书，翻译至少在我看来很好 。只是想要看懂配套代码还得学习一下antlr v3，这个很麻烦，更麻烦的还有我用来写java的idea中的v3插件有bug以至于不能用了。好气啊！
61、什么鬼，不小心勾画了一句话不能退货了
62、说了很多的模式，不太适合入门时看.....还是走实战路线....以后再回来看
63、本书有个标签‘ 屠“ 龙书 ” ’哈哈哈
64、这本书42万字，价格72元，厚度和英文版几乎一样，很不便宜
虽然有英文版电子书，但还是喜欢纸质版
据说它能起到替代编译原理的作用，不知道是不是真的，这种说法有点悬
65、编译原理新的经典
66、很多东西原来都没注意过，虽然有些确实用不到，不过理念和思想是共通的。
67、泛读，学习到了一些思想和知识，细节忽略。个别句子翻译得很别扭。
68、静下心来看。。。
69、具有实践意义的好书，对于快速了解编译器和解释器的实现很有帮助。唯一的缺憾是大量依赖于ANTLR，还有各种模式N搞的头晕。。。
70、may be the easiest introductory book on compiler/interpreter design. Although focus on front-end techniques, it's enough for writing a GPL like python or js.
71、能帮助建立大局观，不会一下子掉到细节里去。也很实用，偏向于解决日常工作里碰到的语言应

用相关问题。
72、已购.
73、讲了很多实现编译器要用到的不同的模式，但是要真的从头开始学，还是要在龙书或者虎书里面选一本看。
74、要仔细阅读，这个领域还是蛮深的
75、不错的，坚持阅读
76、最后三章留着没读，前端还是需要些实践
77、编程语言实现模式
78、继续填坑，还指望着看完这个去定制下CoffeeScript呢……感谢徐老师赠书（再一次o(∩_∩)o ）
79、使用自己创造的语言来变成是每个程序员的梦想，本书非常适合。不过日常可能很少遇到，但是你很有可能需要规定一些语法来创造你项目中的DSL，此书不可错过
80、TP312/4229 盛名之下其实难副。书评里盛赞的大局观和实用性其实都很烂。

《编程语言实现模式》

精彩书评

《编程语言实现模式》

章节试读

1、《编程语言实现模式》的笔记-第176页

P176
“ 在规则enterMethod 和varDeclaration中，采用了类似的操作，用以设置方法返回值类型和变量类型的作用域。 ”

对这里不是很理解。自己理解的意思是，比如一个方法fun()返回一个类A对象，那么A的作用域是需要设置的，这样如果main里有这么一句:A a = fun();时，就可以确定a的作用域了。

但为啥在将类的作用域之前的章节里，没有说到方法需要处理返回值的scope呢？难道是因为之前章节里面AST中没有scope这个变量的原因？

2、《编程语言实现模式》的笔记-第56页

Bryan Ford形式定义了ANTLR文法的记法，并扩展了语法谓词，称为解析表达式文法（PEG），在FP里，语法谓词也称解析组合，见[[Parserc]]

3、《编程语言实现模式》的笔记-第200页

源代码转换：添加显示的类型转换？TokenRewriteStream，记下插入的位置，需要时再执行

4、《编程语言实现模式》的笔记-第251页

２００９年一月，我跟Dan Bornstein交流了一番，他设计了Dalvik虚拟机。

5、《编程语言实现模式》的笔记-第111页

注释中的Stratego/XT的网址有错误，应该为http://strategoxt.org。

6、《编程语言实现模式》的笔记-第76页

画图太麻烦了，大家要么自己看书，要么就靠想象吧=，=

今天上课看到了第四章了，其实本身也是对语法树那里比较感兴趣的，因为感觉对S表达式做词法和语法分析已经没啥问题了，所以还是语法树比较重要一点。作者讲了四种树，即解析树、同型AST、规范化异型AST和不规则异型AST。感想来自于看解析树的过程。

解析树简直就是把语言的BNF表示应用到具体的一段代码上，然后竖起来变成一颗树就可以了。其中保留了太多的信息，例如每一个子表达式的类型都存储了非叶子节点当中。然后作者就开始去掉多余的信息，构建抽象语法树了，也就是所谓的AST。这棵AST当中的内容就只有操作符和操作数了。例如x=0;这个语句，最后只有三个节点剩下：根结点=，左结点（即左子树根结点）x和右节点0。按照作者在``文本形式的树''中的记法，可以写成=(x, 0)或者(= x 0)，而后者其实就是Lisp的S表达式。

使用运算符事实上已经足够判断整个表达式的类型了，所以S表达式实际上就是AST了。这样的AST可以根据car来判断表达式的类型——当然了，Lisp里面都是函数，剩下的，都是语义的范畴了。在Lisp

的REPL中读取字符串解析为S表达式的过程中，cons（点对）类型事实上就是AST的表示方式，也就是作者提到的IR所使用的数据结构了。

刚刚实验了一下，在SBCL中，使用(read-from-string "(liutos")会抛出错误，并且接下来的(intern "LIUTOS")调用的返回值表明符号LIUTOS已经在当前的包中了，所以SBCL中应该是边读取字符串边解析边检验的。如果有事先的检查，那么字符串中括号不匹配应该是可以发现的。所以，我觉得在对S表达式的字符串形式构建AST的过程中，不需要额外的数据结构表示AST了，Lisp的点对，即cons类型就足够了。

7、《编程语言实现模式》的笔记-翻译太随意

A generic VecMathNode node embeds a walking method called print( )---p129

另一个基类VecMathNode内嵌套了遍历方法print()　---p114

书中的意译太多了，这只是一个不起眼的小细节。

```
def f(x):
  x = 1
  y = 2
```
---p236

```
def f(x):
x = 1
y = 2
```
---p223
不知道是排版问题还是译者没用过Python

The main program calls f( ), which calls g( ).　---p262
主函数既会调用f()，又会调用g()。　---p249
这个错误太明显了。

Syntactically, register code is a minor superset of stack code because of its register operands.　---p257
从语法上看，寄存器代码是栈代码的超集，因为可以使用寄存器操作数。　---p244
按照上下文来理解，原意应该是指栈代码实现的解释器因为其操作数操作比寄存器代码实现的解释器简单。

原书p291中用faire un canard举例应当理解为不可"直译"，而这本书p280用“睡觉（sleep）”这个典故显得有些莫名其妙；似乎底下的注释和这里的意译的例子正好反了。

...translating scalar and matrix multiplication.　---p303
...翻译标量到矩阵的乘法运算。　---p292
明显的错误。

8、《编程语言实现模式》的笔记-第94页

同型AST：{ Token token; List<AST> children; } //引用原始的词法单元;

9、《编程语言实现模式》的笔记-第77页

出人意料的是，记录规则的内部节点实际上没什么用，4.2节将介绍只含词法单元的树（？）

10、《编程语言实现模式》的笔记-编译器最复杂的地方

注：
IR
指令寄存器

AST
AST：abstract syntax tree
语法树。编译器前端（frontend）主要负责解析（parse）输入的源代码，由语法分析器和语义分析器协同工作。语法分析器负责把源代码中的‘单词’（Token）找出来,语义分析器把这些分散的单词按预先定义好的语法组装成有意义的表达式，语句，函数等等。例如“a = b + c;”前端语法分析器看到的是“a, =, b , +, c;”，语义分析器按定义的语法，先把他们组装成表达式“b + c”，再组装成“a = b + c”的语句。前端还负责语义（semantic checking）的检查，例如检测参与运算的变量是否是同一类型的，简单的错误处理。最终的结果常常是一个抽象的语法树（abstract syntax tree，或 AST），这样后端可以在此基础上进一步优化，处理。

11、《编程语言实现模式》的笔记-第89页

AST构建操作符 ^(VEC expr+) 能构建表示向量的子树，也是一种树模式。实际上，这里采用的就是文法到树文法的改写规则。

树文法？？？

12、《编程语言实现模式》的笔记-程序员与编译模式

程序员们常做的两种事情：一个是实现某个DSL，二是处理或翻译GPPL。换句话说，程序员可能得实现某种制图语言或者数学语言，但很少需要编写大型程序语言的编译器或解释器。通常所遇到的任务不外乎编写一些重构、格式调度、度量软件、错误查找、插桩或者翻译语言的工具。

编译器所用到的模式，往往也是实现DSL甚至GPPL所需要的关键模式。比如符号表管理模式，几乎所有语言应用都以之为基础。就好像解析器是语法分析的必要工具一样，符号表对于分析输入内容的语义也占有至关重要的地位。一句话，语法告诉我们该做什么，语义告诉我们这样做是为了什么。

注：
GPPL（通用目的编程语言），与DSL相对应。DSL即领域编程语言，它是用于解决特定领域问题的语言。

13、《编程语言实现模式》的笔记-第73页

语法导向的特征就是只用一次扫描，就可以把输入都翻译完

14、《编程语言实现模式》的笔记-第316页

ST（stringtemplae-3.2）中没有foreach循环，而是直接应用到多值属性上（jXLS似乎也支持这么做）

15、《编程语言实现模式》的笔记-第127页

topdown只能化简(* 4 (* 5 0))的最里层，而bottomup则能进一步归约化简

16、《编程语言实现模式》的笔记-第55页

gg

17、《编程语言实现模式》的笔记-第108页

树文法转换为访问者操作序列：^('+' expr expr) ==> '+' DOWN expr expr UP

18、《编程语言实现模式》的笔记-第92页

开发环境可借助解析树实行语法高亮和错误检查

19、《编程语言实现模式》的笔记-第37页

“ 向前看 ”原文是"lookahead"，按习惯还是翻译成“ 预测 ”更好。

20、《编程语言实现模式》的笔记-第45页

“ 环形 ”原文是"circular"，翻译成“ 循环 ”更好。

21、《编程语言实现模式》的笔记-第21页

自从2001年开始迷上编译原理，这么多年以来一直断断续续地学习，却一直处于半懂不懂且缺乏实践的尴尬境地，以至于都要引以为平生一憾了。

昨晚看过本书前两章，感觉有作者两点做的好：一是首先建立大局观，不一上来就陷入细枝末节，这样比较容易入门；二是追求实用性，举的例子都是实际的语言应用，这样比较容易建立直观的联系。这其实也是学习比较复杂的理论时特别要注意的两点，所以最好是先找一本类似本书的入门书，有了大局观和一定的理论和实践基础后再看龙书这样的专业教科书，这样循序渐进，可以少走不少弯路。

2012-11-13 08:27

22、《编程语言实现模式》的笔记-第124页

这里描述子树结构的代码让我想到了《The Little Schemer》里的一页说明：顺序很重要！！否则可能死循环

23、《编程语言实现模式》的笔记-开发人员如何选取编译模式

把各种模式分门别类有助于选择所需的模式。基本上，将其分为（1）读取输入的；（2）分析输入的；（3）解释执行的和（4）生成输出的。

因此，如果只需把数据读入内存，就选用第一部分的模式。编写解释器时，除了读取输入的模式（第一部分），还可能需要第三部分中执行命令的模式。编写翻译器同样要读取输入，但还要使用第四部分的模式来产生输出。除非所处理的语言十分简单，否则还会用到第2部分的模式为这两个过程构建中间的数据结构，以辅助分析输入内容。

24、《编程语言实现模式》的笔记-第120页

熟悉树文法后，就会发现它生成遍历器要比手工编写访问者更方便 ...

25、《编程语言实现模式》的笔记-第63页

带有记忆（memoize）机制的递归下降解析器又称packrat parser.（ps: 其实这种避免重复解析的做法类似于LR分析技术。。。）

26、《编程语言实现模式》的笔记-第111页

项推导：ASF+SDF？Stratego/XT？？

27、《编程语言实现模式》的笔记-第248页

常量池（下标索引访问？）

JIT引擎生成的代码一般使用PC相对寻址加载大常数，如果是索引访问常量池的话，是否能提高字节码运行性能？如何做？

28、《编程语言实现模式》的笔记-语义分析和语言应用

通俗的讲，语义分析就是搞清楚输入的具体含义（以语法为基础的都是语义）。如果语言应用类型不同，则其（编译）流水线中各个阶段的实现方式也不同，而相互间的连接方式也不同。语言应用分四大类。

注：编译流水线：输入文件 -> 翻译器(-> 解释器 -> 语义分析器) ->生成器 -> 输出

1，文件读取器：其主要工作是根据输入流建立数据结构。输入流可以是普通字符、文本，也可以是二进制数据。常见的文件读取器有配置文件读取器、方法调用分析工具之类的程序分析工具，以及JAVA的class文件载入器。

2，生成器：它会收集内部数据结构中的信息，然后产生输出。常见的有对象-关系型数据库（ORM，ORD）映射工具、序列化工具、源代码生成器及网页生成器。

3，翻译器（或称改写器）：翻译器读入字符文本或二进制数据，生成同种或不同种语言的输出，实际上就是文件读取器和生成器的组合。常见的有，把过时的语言翻译成现代语言的翻译器、wiki到HTML的翻译器、重构工具、代码插桩工具、日志报告输出器、格式调整器及宏的预处理器，等等。有些翻译器十分常用，比如汇编器和编译器，已经自成一类了。

4，解释器：通常要读入文件、解码，然后执行指令。从简简单单的计算器，到POP协议的服务器，再到编程语言Java、Ruby和Python的具体实现，都是解释器的具体应用。

《编程语言实现模式》

版权说明