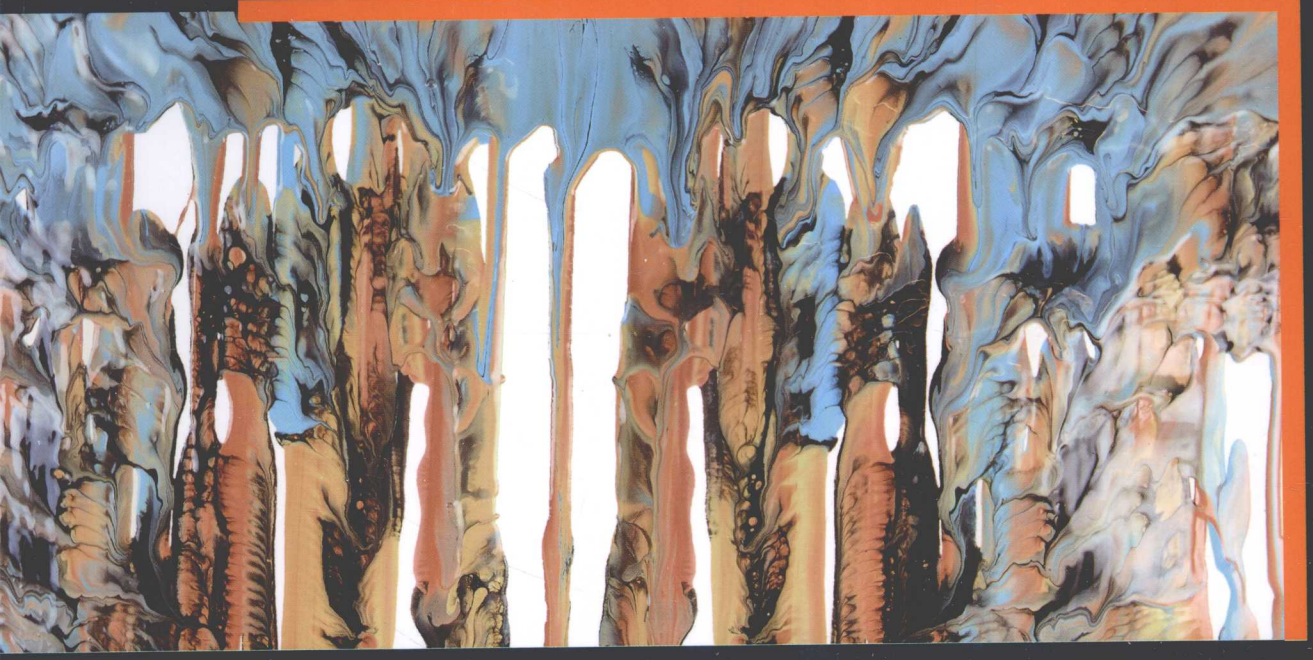
华章程序员书库

The Pragmatic Programmers

HZ BOOKS 华章IT

ANTLR 4 权威指南

The Definitive ANTLR 4 Reference

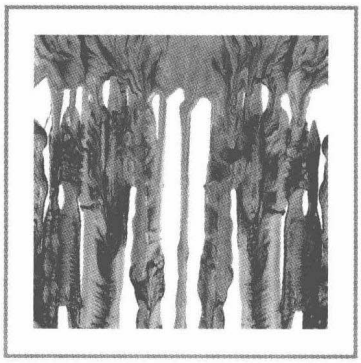


[美] 特恩斯·帕尔（Terence Parr） 著

张博 译　孙岚 石寒舟 审校

机械工业出版社
China Machine Press

华章程序员书库



The Definitive ANTLR 4 Reference

ANTLR 4 权威指南

[美] 特恩斯·帕尔（Terence Parr） 著

张博 译 孙岚 石寒舟 审校

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

ANTLR 4 权威指南 /（美）特恩斯·帕尔（Terence Parr）著；张博译 . —北京：机械工业出版社，2017.5
（华章程序员书库）
书名原文：The Definitive ANTLR 4 Reference

ISBN 978-7-111-56648-9

I. A… II. ①特… ②张… III. 程序语言－程序设计 IV. TP312

中国版本图书馆 CIP 数据核字（2017）第 086847 号

本书版权登记号：图字：01-2017-0156

ANTLR 4 权威指南

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：陈佳媛　　责任校对：殷　虹
印　　刷：北京市荣盛彩色印刷有限公司　　版　　次：2017 年 5 月第 1 版第 1 次印刷
开　　本：186mm × 240mm　1/16　　印　　张：17.5
书　　号：ISBN 978-7-111-56648-9　　定　　价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

The Translator's Words 译 者 序

四年前，我在读研究生时曾经参考龙书编写过一个简单的编译器前端。经过一个星期的实践后，我意识到，从头实现一个编译器前端的难度远远超出了一般开发者的能力。编写编译器前端所需要的理论基础、技术功底和精力都远非普通软件可比。

幸运的是，ANTLR 的出现使这个过程变得易如反掌。ANTLR 能够根据用户定义的语法文件自动生成词法分析器和语法分析器，并将输入文本处理为（可视化的）语法分析树。这一切都是自动进行的，所需的仅仅是一份描述该语言的语法文件。

一年前，我在为淘宝的一个内部数据分析系统设计 DSL 时，第一次接触到了 ANTLR。使用 ANTLR 之后，我在一天之内就完成了整个编译器前端的开发工作，从而能够迅速开始处理真正的业务逻辑。从那时起，我就被它强大的功能所深深吸引。简而言之，ANTLR 能够解决别的工具无法解决的问题。

软件改变了世界。数十年来，信息化的浪潮在全球颠覆着一个又一个的行业。然而，整个世界的信息化程度还远未达到合理的高度，还有大量传统行业的生产力可以被信息化所解放。在这种看似矛盾的情形背后存在着一条鸿沟：大量从事传统行业的人员拥有在本行业中无与伦比的业务知识和经验，却苦于跟不上现代软件发展的脚步。解决这个问题的根本方法就是 DSL（Domain Specific Language），让传统行业的人员能够用严谨的方式与计算机对话。其实，本质上任何编程语言都是一种 DSL，殊途同归。

而实现 DSL 的主要困难就在编译器前端。编译器被称为软件工程皇冠上的明珠。一直以来，对于普通的开发者而言，编译器的设计与实现都如同诗中描述的那样：“白云在青天，可望不可即。”

ANTLR 改变了这一切。ANTLR 自动生成的编译器前端高效、准确，能够将开发者从繁杂的编译理论中解放出来，集中精力处理自己的业务逻辑。ANTLR 4 引入的自动语法分析树创建与遍历机制，极大地提高了语言识别程序的开发效率。

时至今日，ANTLR 仍然是 Java 世界中实现编译器的不二之选，同时，它对其他编程语言也提供了不同程度的支持。在开始学习 ANTLR 时，我发现国内有关 ANTLR 的资料较为贫乏，这催生了我翻译本书的念头。我期望通过本书的翻译，让更多的开发者能够更加自如地解决职业生涯中碰到的难题。

本书没有冗长的理论，而是从一些具体的需求出发，由浅入深地介绍了语言的背景知识、ANTLR 语法的设计方法以及基于 ANTLR 4 实现语言识别程序的详细步骤。它尤其适用于对语言识别程序的开发感兴趣的开发者。不过，假如你现在没有这样的需求，我仍然建议你阅读本书，因为它能够开拓你的眼界，让你深入实现层面加深对编程语言的理解。

感谢原作者 Terence Parr 教授向这个世界贡献了如此优秀的软件。您编写的 ANTLR 极大地提高了开发效率，这实际上等于延长了广大开发者的生命。

感谢孙岚和石寒舟两位前辈对本书审校付出的心血，您二位的宝贵建议令我受益匪浅。

感谢华章公司的和静编辑对本书的翻译提供的支持与帮助。

感谢我的妻子张洁珊女士，你的理解和陪伴保障了翻译过程如期完成。

感谢每一位读者，你的潜心研习与融会贯通将会令本书更有价值。

截止本书译完的 2016 年 12 月，ANTLR 已经演进到了 4.6。在这个过程中，一些 Breaking Change 出现了，本书中的部分示例代码已经不再有效。因此，我尽自己所能，结合勘误表，使用最新版的 ANTLR 对它们进行了逐个验证。对于失效的代码，我通过译注的方式予以修正。由于译者水平有限，书中出现错误与不妥之处在所难免，恳请读者批评指正。

张　博

2017 年 1 月

Preface 前　言

ANTLR 是一款强大的语法分析器生成工具，可用于读取、处理、执行和翻译结构化的文本或二进制文件。它被广泛应用于学术领域和工业生产实践，是众多语言、工具和框架的基石。Twitter 搜索使用 ANTLR 进行语法分析，每天处理超过 20 亿次查询；Hadoop 生态系统中的 Hive、Pig、数据仓库和分析系统所使用的语言都用到了 ANTLR；Lex Machina[⊖] 将 ANTLR 用于分析法律文本；Oracle 公司在 SQL 开发者 IDE 和迁移工具中使用了 ANTLR；NetBeans 公司的 IDE 使用 ANTLR 来解析 C++；Hibernate 对象 – 关系映射框架（ORM）使用 ANTLR 来处理 HQL 语言。

除了这些鼎鼎大名的项目之外，还可以利用 ANTLR 构建各种各样的实用工具，如配置文件读取器、遗留代码转换器、维基文本渲染器，以及 JSON 解析器。我编写了一些工具，用于创建数据库的对象 – 关系映射、描述三维可视化以及在 Java 源代码中插入性能监控代码。我甚至为一次演讲编写了一个简单的 DNA 模式匹配程序。

一门语言的正式描述称为语法（grammar），ANTLR 能够为该语言生成一个语法分析器，并自动建立语法分析树——一种描述语法与输入文本匹配关系的数据结构。ANTLR 也能够自动生成树的遍历器，这样你就可以访问树中的节点，执行自定义的业务逻辑代码。

本书既是 ANTLR 4 的参考手册，也是解决语言识别问题的指南。你会学到如下知识：

- 识别语言样例和参考手册中的语法模式，从而编写自定义的语法。
- 循序渐进地为从简单的 JSON 到复杂的 R 语言编写语法。同时还能学会解决 XML 和 Python 中棘手的识别问题。
- 基于语法，通过遍历自动生成的语法分析树，实现自己的语言类应用程序。
- 在特定的应用领域中，自定义识别过程的错误处理机制和错误报告机制。
- 通过在语法中嵌入 Java 动作（action），对语法分析过程进行完全的掌控。

⊖ http://lexmachina.com

本书并非教科书，所有的讨论都是基于实例的，旨在令你巩固所学的知识，并提供语言类应用程序的基本范例。

本书的读者对象

本书尤其适用于对数据读取器、语言解释器和翻译器感兴趣的开发者。虽然本书主要利用 ANTLR 来完成这些工作，你仍然可以学到很多有关词法分析器和语法分析器的知识。初学者和专家都需要本书来高效地使用 ANTLR 4。如果希望学习第三部分中的高级特性，你需要先了解之前章节中的 ANTLR 基础知识。此外，读者还需要具备一定的 Java 功底。

Honey Badger 版本

ANTLR 4 的版本代号是“Honey Badger”，这个名字来源于一段著名的 YouTube 短片 The Crazy Nastyass Honey Badger（网址为：http://www.youtube.com/watch?v=4r7wHMg5Yjg）中的勇敢无畏的主角——一只蜜獾。它敢吃你给它的任何东西，根本不在乎那是什么！

ANTLR 4 有哪些神奇之处

ANTLR 4 引入了一些新功能，降低了入门门槛，使得语法和语言类应用程序的开发更加容易。最重要的新特性在于，ANTLR 4 几乎能够处理任何语法（除了间接左递归，稍后会提到）。在 ANTLR 将你的语法转换成可执行的、人类可读的语法分析代码的过程中，语法冲突或者歧义性警告不会再出现。

无论多复杂的语法，只要你提供给 ANTLR 自动生成的语法分析器的输入是合法的，该语法分析器就能够自动识别之。当然，你需要自行保证该语法能够准确地描述目标语言。

ANTLR 语法分析器使用了一种名为自适应 *LL(*)* 或者 *ALL(*)*（读作“all star”）的新技术，它是由我和 Sam Harwell[⊖] 一起开发的。*ALL(*)* 是 ANTLR 3 中的 *LL(*)* 的扩展，在实际生成的语法分析器执行前，它能够在运行时以动态方式对语法执行分析，而非先前的静态方式。由于 *ALL(*)* 语法分析器能够访问实际的输入文本，通过反复分析语法的方式，它最终能够决定如何识别输入文本。相比之下，静态分析必须考虑所有可行的（无限长的）输入序列。

在实践中，拥有 *ALL(*)* 意味着你无须像在其他语法分析器生成工具（包括 ANTLR

⊖ http://tunnelvisionlabs.com

3）中那样，扭曲语法以适应底层的语法分析策略。如果你曾经为 ANTLR 3 的歧义性警告和 yacc 的归约 / 归约冲突（reduce/reduce conflict）而抓狂，ANTLR 4 就是你的不二之选！

另外一个强大的新功能是 ANTLR 4 极大地简化了匹配某些句法结构（如编程语言中的算术表达式）所需的语法规则。长久以来，处理表达式都是 ANTLR 语法（以及手工编写的递归下降语法分析器）的难题。识别表达式最自然的语法对于传统的自顶向下的语法分析器生成器（如 ANTLR 3）是无效的。现在，利用 ANTLR 4，你可以通过如下规则匹配表达式：

```
expr : expr '*' expr  // 匹配乘号连接的子表达式
     | expr '+' expr  // 匹配加号连接的子表达式
     | INT            // 匹配简单的整数因子
     ;
```

类似 expr 的自引用规则是递归的，更准确地说，是*左递归*（left recursive）的，因为它的至少一个备选分支直接引用了它自己。

ANTLR 4 自动将类似 expr 的左递归规则重写成了等价的非左递归形式。唯一的约束是左递归必须是直接的，也就是说规则直接引用自身。一条规则不能引用另外一条规则，如果后者的备选分支之一在左侧直接引用了前者（而没有匹配一个词法符号）。详见 5.4 节。

除了上述两项与语法相关的改进，ANTLR 4 还使得编写语言类应用程序更加容易。ANTLR 生成的语法分析器能够自动建立名为*语法分析树*（parse tree）的视图，其他程序可以遍历此树，并在所需处理的结构处触发回调函数。在先前的 ANTLR 3 中，用户需要补充语法来创建树。除了自动建立树结构之外，ANTLR 4 还能自动生成语法分析树遍历器的实现：*监听器*（listener）或者*访问器*（visitor）。监听器与在 XML 文档的解析过程中响应 SAX 事件的处理器相似。

由于拥有以下几点 ANTLR 3 所不具备的新特性，ANTLR 4 显得非常容易上手：

- 最大的改变是 ANTLR 4 降低了语法中内嵌动作（代码）的重要性，取而代之的是监听器和访问器。新机制将语法和应用的逻辑代码解耦，使得应用程序本身被封装起来，而非散落在语法的各处。在没有内嵌动作的情况下，你可以在多个程序中复用同一份语法，甚至都无须重新编译生成的语法分析器。虽然 ANTLR 仍然允许内嵌动作的存在，但是在 ANTLR 4 中，它们更像是一种进阶用法。这样的行为能够最大程度地掌控语法分析过程，但其代价是语法复用性的丧失。
- 由于 ANTLR 能够自动生成语法分析树和树的遍历器，在 ANTLR 4 中，你无须再编写树语法。取而代之的是一些广为人知的设计模式，如访问者模式。这意味着，在学

会了 ANTLR 语法之后，你就可以重回自己熟悉的 Java 领域来实现真正的语言类应用程序。

- ANTLR 3 的 *LL(*)* 语法分析策略不如 ANTLR 4 的 *ALL(*)* 强大，所以 ANTLR 3 为了能够正确识别输入的文本，有时候不得不进行回溯。回溯的存在使得语法的调试格外困难，因为生成的语法分析器会对同样的输入进行（递归的）多趟语法分析。回溯也为语法分析器在面对非法输入时给出错误消息设置了重重障碍。

ANTLR 4 是 25 年前我读研究生时所走的一小段弯路的成果。我想，我也许会稍微改变我曾经的座右铭。

为什么不花 5 天时间编程，来使你 25 年的生活自动化呢？[⊖]

ANTLR 4 正是我所期望的语法分析器生成器，现在，我终于能够回头去研究我原先在 20 世纪 80 年代试图解决的问题——假如我还记得它的话。

本书的主要内容

本书是你所能找到的有关 ANTLR 4 的信息源中最好、最完整的。免费的在线文档提供了足够多有关基础语法的句法和语义的资料，不过没有详细解释 ANTLR 的相关概念。在本书中，识别语言的语法模式和将其表述为 ANTLR 语法的内容是独一无二的。贯穿全书的示例能够在构建语言类应用程序方面助你一臂之力。本书可帮助你融会贯通，成为 ANTLR 专家。

本书由四部分组成。

- 第一部分介绍了 ANTLR，提供了一些与语言相关的背景知识，并展示了 ANTLR 的一些简单应用。在这一部分中，你会了解 ANTLR 的句法以及主要用途。
- 第二部分是一部有关设计语法和使用语法来构建语言类应用程序的“百科全书”。
- 第三部分展示了自定义 ANTLR 生成的语法分析器的错误处理机制的方法。随后，你会学到在语法中嵌入动作的方法——在某些场景下，这样做比建立树并遍历之更简单，也更有效率。此外，你还将学会使用语义判定（semantic predicate）来修改语法分析器的行为，以便解决一些充满挑战的识别难题。

  本部分的最后一章解决了一些充满挑战的识别难题，例如识别 XML 和 Python 中的上下文相关的换行符。

⊖ Terence Parr 原本的座右铭是“Why program by hand in five days what you can spend five years of your life automating?”这里为了承接上文，改成了“Why program by hand in five days what you can spend twenty-five years of your life automating?”。——译者注

- 第四部分是参考章节，详细列出了 ANTLR 语法元语言的所有规则和 ANTLR 运行库的用法。

完全不了解语法和语言识别工具的读者请务必从头开始阅读。具备 ANTLR 3 使用经验的用户可从第 4 章开始阅读以学习 ANTLR 4 的新功能。

有关 ANTLR 的更多在线学习资料

在 http://www.antlr.org 上，你可以找到 ANTLR、ANTLRWorks2 图形界面开发环境、文档、预制的语法、示例、文章，以及文件共享区。技术支持邮件组是一个对初学者十分友好的公开讨论组[⊖]。

Terence Parr

2012 年 11 月于旧金山大学

⊖ https://groups.google.com/d/forum/antlr-discussion

致　谢 *Acknowledgements*

大约 25 年前，我开始致力于 ANTLR 的相关工作。那时，在许多人的帮助下，ANTLR 工具的句法和功能逐渐成形，在此，我向他们致以由衷的感谢。要特别感谢的是 Sam Harwell[⊖]，他是 ANTLR 4 的另一位开发者。他不仅帮助我完成了此软件，而且在 *ALL(*)* 语法分析算法上做出了突出的贡献。Sam 也是 ANTLRWorks2 语法 IDE 的开发者。

感谢以下人员对本书进行了技术审阅：Oliver Ziegermann、Sam Rose、Kyle Ferrio、Maik Schmidt、Colin Yates、Ian Dees、Tim Ottinger、Kevin Gisi、Charley Stran、Jerry Kuch、Aaron Kalair、Michael Bevilacqua-Linn、Javier Collado、Stephen Wolff 以及 Bernard Kaiflin。同时，我还要感谢那些在本书和 ANTLR 4 软件处于 beta 版本时报告问题的热心读者。尤其要感谢的是 Kim Shrier 和 Graham Wideman，他们二位的审阅格外认真。Graham 的审阅报告之仔细、翔实和广博，令我不知是该紧握他的手予以感谢，还是该为自己的疏漏羞愧难当。

最后，我还要感谢编辑 Susannah Davidson Pfalzer，她一如既往地支持我完成了三本书的创作。她提出的宝贵建议和对本书内容的精雕细琢使本书更加完美。

⊖ http://tunnelvisionlabs.com

Contents 目　录

译者序

前言

致谢

第一部分　ANTLR 和计算机语言简介

第 1 章　初识 ANTLR …… 3

1.1　安装 ANTLR …… 3

1.2　运行 ANTLR 并测试识别程序 …… 5

第 2 章　纵观全局 …… 9

2.1　从 ANTLR 元语言开始 …… 9

2.2　实现一个语法分析器 …… 11

2.3　你再也不能往核反应堆多加水了 … 13

2.4　使用语法分析树来构建语言类应用程序 …… 15

2.5　语法分析树监听器和访问器 …… 17

第 3 章　入门的 ANTLR 项目 …… 20

3.1　ANTLR 工具、运行库以及自动生成的代码 …… 21

3.2　测试生成的语法分析器 …… 23

3.3　将生成的语法分析器与 Java 程序集成 …… 25

3.4　构建一个语言类应用程序 …… 26

第 4 章　快速指南 …… 29

4.1　匹配算术表达式的语言 …… 30

4.2　利用访问器构建一个计算器 …… 35

4.3　利用监听器构建一个翻译程序 …… 38

4.4　定制语法分析过程 …… 41

4.5　神奇的词法分析特性 …… 45

第二部分　使用 ANTLR 语法开发语言类应用程序

第 5 章　设计语法 …… 53

5.1　从编程语言的范例代码中提取语法 …… 54

5.2　以现有的语法规范为指南 …… 56

5.3　使用 ANTLR 语法识别常见的语言模式 …… 56

5.4　处理优先级、左递归和结合性 …… 62

5.5 识别常见的词法结构 …… 66
5.6 划定词法分析器和语法分析器的界线 …… 71

第 6 章 探索真实的语法世界 …… 74

6.1 解析 CSV 文件 …… 75
6.2 解析 JSON …… 77
6.3 解析 DOT 语言 …… 83
6.4 解析 Cymbol 语言 …… 88
6.5 解析 R 语言 …… 91

第 7 章 将语法和程序的逻辑代码解耦 …… 98

7.1 从内嵌动作到监听器的演进 …… 99
7.2 使用语法分析树监听器编写程序 …… 100
7.3 使用访问器编写程序 …… 103
7.4 标记备选分支以获取精确的事件方法 …… 105
7.5 在事件方法中共享信息 …… 107

第 8 章 构建真实的语言类应用程序 …… 114

8.1 加载 CSV 数据 …… 114
8.2 将 JSON 翻译成 XML …… 117
8.3 生成调用图 …… 121
8.4 验证程序中符号的使用 …… 124

第三部分 高级特性

第 9 章 错误报告与恢复 …… 133

9.1 错误处理入门 …… 133
9.2 修改和转发 ANTLR 的错误消息 …… 137
9.3 自动错误恢复机制 …… 141
9.4 勘误备选分支 …… 152
9.5 修改 ANTLR 的错误处理策略 …… 152

第 10 章 属性和动作 …… 156

10.1 使用带动作的语法编写一个计算器 …… 157
10.2 访问词法符号和规则的属性 …… 162
10.3 识别关键字不固定的语言 …… 165

第 11 章 使用语义判定修改语法分析过程 …… 168

11.1 识别编程语言的多种方言 …… 169
11.2 关闭词法符号 …… 172
11.3 识别歧义性文本 …… 174

第 12 章 掌握词法分析的“黑魔法” …… 180

12.1 将词法符号送入不同通道 …… 181
12.2 上下文相关的词法问题 …… 184
12.3 字符流中的孤岛 …… 194
12.4 对 XML 进行语法分析和词法分析 …… 198

第四部分 ANTLR 参考文档

第 13 章 探究运行时 API …… 209

13.1 包结构概览 …… 209
13.2 识别器 …… 210

13.3 输入字符流和词法符号流 …… 212
13.4 词法符号和词法符号工厂 …… 213
13.5 语法分析树 …… 215
13.6 错误监听器和监听策略 …… 216
13.7 提高语法分析器的速度 …… 217
13.8 无缓冲的字符流和词法符号流 …… 217
13.9 修改 ANTLR 的代码生成机制 …… 219

第 14 章 移除直接左递归 …… 221

14.1 直接左递归备选分支模式 …… 222
14.2 左递归规则转换 …… 223

第 15 章 语法参考 …… 226

15.1 语法词汇表 …… 226
15.2 语法结构 …… 229
15.3 文法规则 …… 232
15.4 动作和属性 …… 241
15.5 词法规则 …… 246
15.6 通配符与非贪婪子规则 …… 250
15.7 语义判定 …… 253
15.8 选项 …… 257
15.9 ANTLR 命令行参数 …… 259

参考文献 …… 263

第一部分 *Part 1*

ANTLR 和计算机语言简介

- 第 1 章　初识 ANTLR
- 第 2 章　纵观全局
- 第 3 章　入门的 ANTLR 项目
- 第 4 章　快速指南

在第一部分中，我们会安装ANTLR，尝试通过它来识别一个简单的“hello world”语法，并概览语言类应用程序的开发过程。在此基础上，我们会构造一个语法来识别和翻译形如{1，2，3}的花括号中的一列整数。最后，我们将通过一系列的简单语法和程序来快速了解ANTLR的特性。

第 1 章 Chapter 1

初识 ANTLR

在本书的第一部分中，我们的目标是大体上知道 ANTLR 能做什么。除此之外，我们还希望探究语言类应用程序的架构。在概览之后的第 2 章中，我们将会通过许多真实的例子来循序渐进地、系统性地学习 ANTLR。在开始之前，我们需要首先安装 ANTLR，然后尝试用它编写一份简单的“hello world”语法。

1.1 安装 ANTLR

ANTLR 是用 Java 编写的，因此你需要首先安装 Java[⊖]，哪怕你的目标是使用 ANTLR 来生成其他语言（如 C# 和 C++）的解析器。（我希望在不远的未来 ANTLR 可以支持更多语言[⊜]。）ANTLR 运行所需的 Java 版本为 1.6 或更高。

为什么本书使用命令行

在整本书中，我们都会使用命令行（shell）来运行 ANTLR 和构建我们的程序。因为开发者使用的开发环境和操作系统五花八门，因此只有操作系统的 shell 才是我们公用的“界面”。使用 shell 也使得开发语言程序的每一个步骤更加清晰和明确。在本书中我将会一直使用 Mac OS X 作为示例，不过这些示例命令理论上应该能够在任何类 UNIX 系统的 shell 中正常工作，同时，在稍作修改后，它们应该能够适用于 Windows。

⊖ http://www.java.com/en/download/help/download_options.xml

⊜ ANTLR 现已支持多种目标语言，详见 https://github.com/antlr/antlr4/tree/master/doc。——译者注