Extracted from:

The Definitive ANTLR 4 Reference

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

The Definitive ANTLR 4 Reference

Terence Parr

Edited by Susannah Davidson Pfalzer

Printed in the United States of America.
ISBN-13: 978-1-93435-699-9
Encoded using the finest acid-free high-entropy binary digits.
Book version: B1.0—September 19, 2012

Welcome Aboard!

ANTLR v4 is a powerful parser generator that you can use to read, process, execute, or translate structured text such as program source code, data, and configuration files. That means you can use it to build all sorts of useful tools like legacy code converters, domain-specific language interpreters, Java to C# translators, wiki renderers, XML/HTML parsers, DNA pattern recognizers, bytecode assemblers, language pretty printers, and of course full compilers.

From a formal language description called a grammar, ANTLR generates a program that reads in files conforming to that format and builds a data structure representing the input (a parse tree). It also automatically generates tree walkers that we can use to visit the nodes of those trees to execute application-specific code.

This book is both a reference for ANTLR v4 and a guide to using it to solve language recognition problems. You're going to learn how to:

- Identify grammar patterns in language samples and reference manuals in order to build your own grammars.
- Build grammars for simple languages like JSON all the way up to complex programming languages like R. You'll also solve some tricky recognition problems from Python and XML.
- Implement language applications based upon those grammars by walking the automatically-generated parse trees.
- Customize recognition error handling and error reporting for specific application domains.
- Take absolute control over parsing by embedding Java actions into a grammar.

Unlike a textbook, the discussions are example-driven in order to make things more concrete and to provide starter kits for building your own language applications.

Who Is This Book For?

This book is specifically targeted at any programmer interested in learning to use ANTLR to build data readers, language interpreters, and translators. Beginners and experts alike will need this book to use ANTLR v4 effectively. To get your head around the advanced topics in Part III, you'll need some experience with ANTLR by working through the earlier chapters. Readers should know Java to get the most out of the book.

The Honey Badger Release

ANTLR v4 is named the "Honey Badger" release after the fearless hero of the YouTube sensation, "The Crazy Nastyass Honey Badger,"[a] It takes whatever grammar you give it; it doesn't give damn!

a. http://www.youtube.com/watch?v=4r7wHMg5Yjg

What's So Cool about ANTLR v4?

The v4 release of ANTLR has some important new capabilities that reduce the learning curve and make developing grammars and language applications much easier. Here are the key features:

- ANTLR v4 gladly accepts every grammar you give it (with one exception described in the next bullet). There are no grammar conflict or ambiguity warnings as ANTLR translates your grammar to executable, Human–readable parsing code. If you give your ANTLR-generated parser valid input, it will always recognize it properly, no matter how complicated the grammar. (Of course, it's up to you to make sure that the grammar accurately describes the language in question.)

  ANTLR parsers use a brand-new parsing technology called *Adaptive LL(*)* or *ALL(*)* ("all star") that I developed with Sam Harwell.[1] *ALL(*)* is an extension to v3's *LL(*)* that performs grammar analysis on-the-fly rather than trying to analyze grammars statically, before the generated parsers execute. Because *ALL(*)* parsers have access to actual input sequences, they can always figure out how to recognize the sequences by appropriately weaving through the grammar.

- ANTLR v4 dramatically simplifies the grammar rules used to match syntactic structures like programming language expressions. Expressions have always been a hassle to specify with ANTLR grammars (and to rec-

1. http://tunnelvisionlabs.com

ognize by hand with recursive-descent parsers). The most natural grammar to recognize expressions is invalid for traditional top-down parser generators like ANTLR v3. Now, with v4, you can match expressions with rules that look like this:

```
expr : expr '*' expr  // match subexpressions joined with '*' operator
     | expr '+' expr  // match subexpressions joined with '+' operator
     | INT            // matches simple integer atom
     ;
```

Self-referential rules like expr are recursive and, in particular, *left recursive* because one of its alternatives immediately refers to itself.

ANTLR v4 automatically rewrites left-recursive rules such as expr into non-left-recursive equivalents. The only constraint is that the left recursion must be direct, where rules immediately reference themselves. Rules cannot reference another rule on the left side of an alternative that eventually comes back to reference the original rule. See Section 4.4, *Dealing with Precedence, Left Recursion, and Associativity*, on page ? for more details.

- Language applications need to execute code snippets when they see specific input sentences, phrases, or tokens. To make this easier, ANTLR-generated parsers automatically build convenient representations of the input called *parse trees* that an application can walk to trigger code snippets as it encounters constructs of interest. Previously, v3 users had to augment the grammar with tree construction operations. Moreover, ANTLR automatically generates parse tree walkers for us in the form of *listener* and *visitor pattern* implementations. Listeners are analogous to XML document handler objects that respond to SAX events triggered by XML parsers.

ANTLR v4 is much easier to learn because of those awesome new features, but also because of what it does not carry forward from v3:

- The biggest change is that v4 deemphasizes embedding actions (code) in the grammar, favoring listeners and visitors instead. Those mechanisms decouple grammars from application code, nicely encapsulating an application instead of fracturing it and dispersing the pieces across a grammar. Without embedded actions, you can also reuse the same grammar in different applications without even recompiling the generated parser. ANTLR still allows embedded actions, but doing so is considered advanced in v4. Such actions give the highest level of control, but at the cost of losing grammar reuse.

- Because ANTLR automatically generates parse trees and tree walkers, there's no need for you to build tree grammars in v4. You get to use familiar design patterns like the visitor instead. That means that, once you've learned ANTLR grammar syntax, you get to move back into the comfortable and familiar Java space to implement the actual language application.
- ANTLR v3's *LL(*)* parsing strategy was weaker than v4's *ALL(*)* and sometimes relied on backtracking to properly parse input phrases. Backtracking makes it hard to debug a grammar by stepping through the generated parser because the parser might parse the same input multiple times (recursively).

ANTLR v4 is the result of a minor 25 year detour I took in graduate school. I guess I'm going to have to change my motto slightly:

> Why program by hand in 5 days what you can spend *25* years of your life automating.

ANTLR v4 is finally what I want in a parser generator and I can finally get back to the problem I was originally trying to solve in the 80s. Now, if I could just remember what that was.

What's in This Book?

This book is the best, most complete source of information on ANTLR v4 that you'll find anywhere. The free, online documentation provides enough to learn the basic grammar syntax and semantics but doesn't explain ANTLR concepts in detail. Only this book explains how to identify grammar patterns in languages and how to express those as ANTLR grammars. The examples woven throughout the text give you the leg up you need to start building your own language applications. This book helps you get the most out of ANTLR and is required reading to become an advanced user.

This book is organized into four parts:

- Part I introduces ANTLR, provides some background knowledge about languages, and gives you a tour of ANTLR's capabilities. You'll get a taste of the syntax and what you can do with it.
- Part II is all about designing grammars and building language applications using those grammars in combination with tree walkers. We'll also look at how to customize the error handling of ANTLR-generated parsers.
- Part III shows you how to embed actions in the grammar when it's simpler or more efficient than building a tree and walking it. Related to actions,

you'll also learn how to use *semantic predicates* to alter the behavior of the parser to handle some challenging recognition problems.

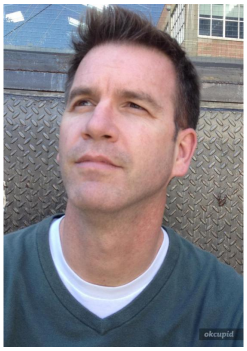
- Part IV is the reference section and lays out all of the rules for using the ANTLR grammar meta-language and its runtime library.

Readers who are totally new to grammars and language tools should definitely start by reading chapters Chapter 1, *Meet ANTLR*, on page ? and Chapter 2, *The Big Picture*, on page ?. Experienced ANTLR v3 users can jump directly to chapter Chapter 3, *A Quick Tour*, on page ? to learn more about v4's new capabilities.

The source code for all examples in this book are available online. For those of you reading this electronically, you can click on the gray box above the source code and it will display the code in a browser window. If you're reading the paper version of this book, or would simply like a complete bundle of the code, you can grab it at the book website.[2] In order to focus on the key elements being discussed, most of the code snippets shown in the book itself are partial. The downloads show the full source.

Learning More about ANTLR Online

At the http://www.antlr.org website, you'll find the ANTLR download, the ANTLRWorks2 graphical user interface (GUI) development environment, documentation, prebuilt grammars, examples, articles, and a file-sharing area. The tech support mailing list[3] is a newbie-friendly public Google group.



Terence Parr

University of San Francisco, August 2012

2. http://pragprog.com/titles/tpantlr2/source_code
3. https://groups.google.com/d/forum/antlr-discussion