ANTLR: Parser generator

Jiří Kučera, xkucer28@stud.fit.vutbr.cz
Jiří Raška, xraska07@stud.fit.vutbr.cz

November 13, 2011

Abstract

ANTLR (ANother Tool for Language Recognition) is a powerfull parser generator developed by Terence Parr. This tool, written in *Java* programming language, is result of 20 years[1] of development (the current major version is 3). ANTLR parser generator generates human-readable code (compared to other parser generators, like famous `yacc` or `bison`) into many target languages, like *C*, *C#*, *Java*, *Python* etc. ANTLR also contains graphical IDE (called *ANTLRWorks*) in which we can write our grammar definition and then debug our written grammar. Other ANTLR features[1] are powerfull string template engine called *StringTemplate* (with this engine, we can extend set of supported target languages, simply by writing source code template for our favourite programming language), run-time support for target languages to bind generated parser with our application, powerfull parsing algorithm (called *LL(*)*[2]) used by generated parsers, and great documentation. The *LL(*)* parsing technique used by ANTLR is similar to *LL(k)* parsing technique (used by earlier versions of ANTLR), but with one difference — the lookahead is unlimited[1] (in *LL(k)* the lookahead is given by *k*). This difference is denoted by * in *LL(*)*.

The language of grammar definition file is very simple, based on EBNF. The grammar definition file consists of three[1] parts: the definition of scanner, the definition of parser, and the definition of tree grammar — the grammar used for the *abstract syntax tree* (as known as *computational tree*) parser description. These parts are also included in the generated parser. In our application, we call parser firstly (the scanner is called by parser), which give us an abstract syntax tree as its result, and after that we call tree parser, which do the rest of the job (intermediate code generation, optimization and target code generation). Because the ANTLR generates[1] recursive descent parsers (not table-driven, like `yacc` or `bison`) we cannot use rules with left recursion in our grammars.

References

[1] PARR, Terence. *The Definitive ANTL Reference : Building Domain-Specific Languages*. First printing. Printed in the United States of America : The Pragmatic Bookshelf, May 2007. 369 p. ISBN-10 0-9787392-5-6. ISBN-13 978-09787392-4-9.

[1] `http://www.antl.org/about.html`

[2] The name *LL(*)* for parsing technique used in ANTLR was introduced by *Sriram Srinivasan*[1], co-inventer of this technique and Terence Parr's friend.