Introduction to ANTLR (I)

Nai-Wei Lin, Peng-Sheng Chen

Department of Computer Science
National Chung Cheng University

1

Lex and Yacc

- Lex: lexical analyzer generator
- Written by Mike Lesk and Eric Schmidt, 1975.
- Variants: **flex** (GNU)

- Yacc (Yet Another Compiler Compiler): parser generator
- It is a program designed to compile a LALR(1) grammar and to produce the source code of the syntactic analyzer of the language produced by this grammar.
- It is also possible to make it do semantic actions.
- Written by Stephen C. Johnson, 1975.
- Variants: YACC(AT&T), **BISON** (GNU), PCYACC.

Reference Books

- **lex & yacc, 2/E**
書名：lex & yacc 中譯本（修訂版）
作者：John R. Levine, Tony Mason &
Doug Brown
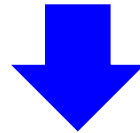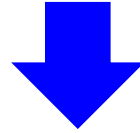譯者：林偉豪
書號：A020
ISBN：957-8247-09-5
頁數：394頁
出版日期：1999 年 03 月
售價：680

Token description file
(xxx.g)

ANTLR

Lexical analyzer
(source code)

ANTLR

- **AN**other **T**ool for **L**anguage **R**ecognition

- A language tool that provides a framework for constructing recognizers, interpreters, compilers, and translators from grammatical descriptions containing actions in a variety of target languages.

- Author: Terence Parr since 1989

ANTLR v3

ANTLR 官網首頁 (www.antlr.org)

- The latest version of ANTLR is 4.8, released January 16, 2020.
- As of 4.8, **we have Java, C#, Javascript, Python, C++, Go, Swift, and PHP targets**.

ANTLR 3 官網首頁

http://www.antlr3.org/

ANTLR v3

Latest version is 3.5.2
Download now! »
DOWNLOAD

What is ANTLR?
ANTLR, ANother Tool for Language Recognition, is a language tool that provides a framework for constructing recognizers, interpreters, compilers, and translators from grammatical descriptions containing actions in a variety of target languages. ANTLR provides excellent support for tree construction, tree walking, translation, error recovery, and error reporting. There are currently about **5,000** ANTLR source downloads a month.

ANTLR has a sophisticated grammar development environment called ANTLRWorks, written by Jean Bovet.

Terence Parr is the maniac behind ANTLR and has been working on language tools since 1989. He is a professor of computer science at the University of San Francisco.

More...

Testimonials

Thanks to all the ANTLR team !
gwenael.chailleu@gmail.com
Just a few words to say that we used ANTLR V3.4 to create our own extension...

Antlr for NLP
Mihai Surdeanu
At Lex Machina (lexmachina.com) we use Antlr for information extraction...

Using ANTLR to build a query parser and translator
Atul Dambalkar (atul@entrib.com)
We have been using ANTLR to build a query parser and then converting the...

ANTLR makes building custom DSLs a breeze
Mukundan Agaram (Enterprise Architect at...
We used ANTLR to build and parse custom Domain Specific Business Rule Languages...

More...

Showcase
Release14
Damir Franusic *Wed Sep 19, 2012 08:09*
Release14 product line uses ANTLR in various modules like CLI, SMS Filtering,...

UDK/UnrealScript plugin for Eclipse
Ori Hanegby *Wed Aug 17, 2011 08:28*
UDK/UnrealScript support for eclipse. Adds support for code completion,...

Open Cloud Computing Interface
Andy Edmonds *Tue Jun 21, 2011 08:49*
The OCCI grammar supplies a lexer and parser that will validate any valid...

More...

TRY NEW ANTLR v4!

If you like ANTLR, check out the StringTemplate template engine for all your structured text generation needs.

SEARCH

News   Documentation   Articles

ANTLR Grammar File Structure

- **Four kinds of ANTLR grammars (GrammarType):**
  - combined lexer and parser
  - lexer
  - parser
  - tree

```
/** This is a document comment */
grammarType grammar name;
«optionsSpec»
«tokensSpec»
«attributeScopes»
«actions»

/** doc comment */
rule1 : ... | ... | ... ;
rule2 : ... | ... | ... ;
...
```

Reference from: http://www.antlr.org

ANTLR: Lexer

- Lexical rules all begin with an uppercase letter in ANTLR and typically refer to character and string literals.

```
ID   :    ('a'..'z'|'A'..'Z')+ ;
INT  :    '0'..'9'+ ;
NEWLINE:'\r'? '\n' ;                    /* CR ? LF */
WS   :    (' '|'\t'|'\n'|'\r')+ {skip();} ;
```

Rule WS (whitespace) is the only one with an action (**skip()**;) that tells ANTLR to throw out what it just matched and look for another token.

Rule Elements

- () - Parentheses. Used to group several elements, so they are treated as one single token

- ? - Any token followed by ? occurs 0 or 1 times

- * - Any token followed by * can occur 0 or more times

- + - Any token followed by + can occur 1 or more times

- . - Any character/token can occur one time

- ~ - Any character/token following the ~ may not occur at the current place

- .. - Between two characters .. spans a range which accepts every character between both boundaries inclusive

- '«one-or-more-char»' - String or char literal in single quotes

Example

Use the Java language target,
ANTLR generates Tlexer.java, …

action

```
grammar T;
options {
    language=Java;
}
@members {
String s;
}
r : ID '#' {s = $ID.text; System.out.println("found "+s);} ;
ID: 'a'..'z'+ ;
WS: (' '|'\n'|'\r')+ {skip();} ; // ignore whitespace
```

11

Reference from: http://www.antlr.org

Comments

```
grammar T;

/* a multi-line
   comment
 */

/** This rule matches a declarator for my language */
decl : ID ; // match a variable name
```

12

Identifiers

- Token names (lexer rules) : start with a capital letter.
- Nonlexer rules: start with a lowercase letter.
- The initial character can be followed by uppercase and lowercase letters, digits, and underscores.
- Only ASCII characters are allowed in ANTLR names.

```
ID, LPAREN, RIGHT_CURLY // token names/rules
expr, simpleDeclarator, d2, header_file // rule names
```

13

Literals

- ANTLR does not distinguish between character and string literals.

- All literal strings one or more characters in length are enclosed in single quotes such as ';', 'if', and '>='.

- Special escape sequences: '\n' (newline), '\r' (carriage return), ...

- Literals can contain Unicode escape sequences of the form \uXXXX, where XXXX is the hexadecimal Unicode character value.

14

Actions

- Actions are code blocks written in the target language.
- Syntax: arbitrary text surrounded by curly braces.
- The action text should conform to the target language as specified with the **language** option.

```
grammar T;
options {
    language=Java;
}
@members {
String s;
}
r : ID '#' {s = $ID.text; System.out.println("found "+s);} ;
ID: 'a'..'z'+ ;
WS: (' '|'\n'|'\r')+ {skip();} ; // ignore whitespace
```

action

How to Define Tokens

This keyword fragment tells ANTLR that you intend for this rule to be called only by other rules and that it should not yield a token to the parser. **(regular definition)**

```
NUMBER : (DIGIT)+
        ;


ID : (LETTER)(LETTER | DIGIT)*
    ;



fragment LETTER : 'a'..'z'
                | 'A'..'Z'
                ;



fragment DIGIT : '0'..'9'
                ;
```

How to Define Tokens

- The earlier a token is defined, the higher is the precedence if a certain input can be matched by two or more tokens.

- This means that using the tokens command to define keywords will match those keywords instead a more general ID rule.

```
WS:        (‘ ’ | ‘\n’)+  { skip(); };

NEWLINE: ‘\n’;
```

```
IF:        ‘if’;

ID:        (‘a’..’z’)+;
```

Example (1)

```
lexer grammar test1;

options {
  language = Java;
}

NUMBER : (DIGIT)+;

ID : (LETTER)(LETTER | DIGIT)*;

fragment LETTER : 'a'..'z' | 'A'..'Z';

fragment DIGIT : '0'..'9';

WS  : (' '|'\r'|'\t'|'\n')+;
```

Example (2)

```java
import org.antlr.runtime.ANTLRFileStream;
import org.antlr.runtime.CharStream;
import org.antlr.runtime.Token;

public class testLexer {
    public static void main(String[] args) {
        CharStream input = new ANTLRFileStream(args[0]);
        test1 lexer = new test1(input);
        Token token = lexer.nextToken();
        while (token.getType() != -1) {  // -1 is EOF.
            System.out.println("Token: "+ token.getType() + " "
                                    + token.getText());
            token = lexer.nextToken();
        }
    }
}
```

Install and Compile

ANTLR 官網首頁

ANTLR

| Download v4.7.1 | Dev Tools | Book | Doc | About ANTLR | Support | Bugs |

Quick Start | Samples

What is ANTLR?

ANTLR (ANother Tool for Language Recognition) is a powerful parser generator for reading, processing, executing, or translating structured text or binary files. It's widely used to build languages, tools, and frameworks. From a grammar, ANTLR generates a parser that can build and walk parse trees.

Looking for ANTLR v3?

Terence Parr is the maniac behind ANTLR and has been working on language tools since 1989. He is a professor of computer science at the University of San Francisco.

Quick Start
See Getting Started with ANTLR v4

```
OS X
$ cd /usr/local/lib
$ sudo curl -O http://www.antlr.org/download/antlr-4.7.1-complete.jar
$ export CLASSPATH=".:/usr/local/lib/antlr-4.7.1-complete.jar:$CLASSPATH"
$ alias antlr4='java -jar /usr/local/lib/antlr-4.7.1-complete.jar'
$ alias grun='java org.antlr.v4.gui.TestRig'
```

21

ANTLR 官網首頁 (v3)

Download

Latest version is 3.5.2
Download now! »   DOWNLOAD

Testimonials

Thanks to all the ANTLR team !
gwenael.chailleu@gmail.com
Just a few words to say that we used ANTLR V3.4 to create our own extension...

Antlr for NLP
Mihai Surdeanu
At Lex Machina (lexmachina.com) we use Antlr for information extraction...

Using ANTLR to build a query parser and translator
Atul Dambalkar (atul@entrib.com)
We have been using ANTLR to build a query parser and then converting the...

ANTLR makes building custom DSLs a breeze
Mukundan Agaram (Enterprise Architect at...
We used ANTLR to build and parse custom Domain Specific Business Rule Languages...

More...

Showcase
Release14
Damir Franusic *Wed Sep 19, 2012 08:09*
Release14 product line uses ANTLR in various modules like CLI, SMS Filtering,...

UDK/UnrealScript plugin for Eclipse
Ori Hanegby *Wed Aug 17, 2011 08:28*
UDK/UnrealScript support for eclipse. Adds support for code completion,...

Open Cloud Computing Interface
Andy Edmonds *Tue Jun 21, 2011 08:49*
The OCCI grammar supplies a lexer and parser that will validate any valid...

More...

TRY NEW **ANTLR v4**!

If you like ANTLR, check out the **StringTemplate template engine** for all your structured text generation needs.

ANTLR v3

ANTLR v3

Latest ANTLR version is 3.5.2, released March 25, 2014. v3 is written in ANTLR v3 and StringTemplate 4.0.8. Parsers that build template output still use StringTemplate v3 for backward compatibility. In antlr-3.5.2-complete.jar, you'll find everything you need to make ANTLR and its generated parsers work. Please see **release info**.

- ANTLR 3.5.2 (Source for tools, targets)
- Complete ANTLR 3.5.2 Java binaries w/o legacy ANTLR v2 jar (complete ANTLR 3.5.2 tool, Java runtime and ST 4.0.8 which should be able to run the tool and the generated code unless you are using output=template; For backward compatibility reasons, ANTLR 3.5.2 still generates code that uses ST v3 at parse-time.)
- Complete ANTLR 3.5.2 Java binaries jar (complete ANTLR 3.5.2 tool, Java runtime, ST 3.2.1, ANTLR v2, and ST 4.0.8; for use when you use output=template)

Runtime libraries

- ANTLR 3.5.2 Java runtime complete binary jar (includes gunit, StringTemplate, antlr-2.7.7.jar)
- Python runtime distributions
- C runtime distributions

Use ANTLR from the command-line

- `$java -jar antlr-3.5.2-complete.jar`
- `$java -cp antlr-3.5.2-complete.jar org.antlr.Tool`

```
pschen@mcore8[12:56pm]~/compiler_design> java -cp ./antlr-3.4-complete.jar org.a
ntlr.Tool
ANTLR Parser Generator   Version 3.4
usage: java org.antlr.Tool [args] file.g [file2.g file3.g ...]
  -o outputDir          specify output directory where all output is generated
  -fo outputDir         same as -o but force even files with relative paths to d
ir
  -lib dir              specify location of token files
  -depend               generate file dependencies
  -report               print out a report about the grammar(s) processed
  -print                print out the grammar without actions
  -debug                generate a parser that emits debugging events
  -profile              generate a parser that computes profiling information
  -trace                generate a recognizer that traces rule entry/exit
  -nfa                  generate an NFA for each rule
  -dfa                  generate a DFA for each decision point
  -message-format name  specify output style for messages
  -verbose              generate ANTLR version and other information
  -make                 only build if generated files older than grammar
  -version              print the version of ANTLR and exit.
  -language L           override language grammar option; generate L
  -X                    display extended argument list
pschen@mcore8[12:56pm]~/compiler_design>
```

Use ANTLR from the command-line

- `$java -cp antlr-3.5.2-complete.jar org.antlr.Tool test1.g`

- 產生
  - **test1.java**
  - **test1.tokens**

26

Use ANTLR from the command-line

DEMO

- Compile

  - $javac -cp ./antlr-3.5.2-complete.jar testLexer.java test1.java

- Execute

  - $java -cp ./antlr-3.5.2-complete.jar:. testLexer input.c

ANTLR Runtime Library for C

- $tar –zxvf libantlr3c-3.4.tar.gz

- Check the files: "README" and "INSTALL"

- $cd libantlr3c-3.4
- $./configure --prefix=/cshome/professor/pschen/linuxtools
- $make
- $make install

28

Errors

```
     -O2 -MT antlr3baserecognizer.lo -MD -MP -MF .deps/antlr3baserecognizer.Tpo -
c -o antlr3baserecognizer.lo `test -f 'src/antlr3baserecognizer.c' || echo './'`
src/antlr3baserecognizer.c
libtool: compile:  gcc -DHAVE_CONFIG_H -I. -Iinclude -O2 -MT antlr3baserecognize
r.lo -MD -MP -MF .deps/antlr3baserecognizer.Tpo -c src/antlr3baserecognizer.c -
fPIC -DPIC -o .libs/antlr3baserecognizer.o
In file included from /usr/include/stdio.h:34,
                 from include/antlr3defs.h:248,
                 from include/antlr3baserecognizer.h:39,
                 from src/antlr3baserecognizer.c:9:
/usr/lib/gcc/x86_64-linux-gnu/4.3.2/include/stddef.h:214: error: duplicate 'unsi
gned'
/usr/lib/gcc/x86_64-linux-gnu/4.3.2/include/stddef.h:214: error: two or more dat
a types in declaration specifiers
src/antlr3baserecognizer.c: In function 'getRuleMemoization':
src/antlr3baserecognizer.c:1881: warning: cast from pointer to integer of differ
ent size
src/antlr3baserecognizer.c:1881: warning: cast to pointer from integer of differ
ent size
make[1]: *** [antlr3baserecognizer.lo] Error 1
make[1]: Leaving directory `/CSDATA_NFS/professor/pschen/compiler_design/libantl
r3c-3.4'
make: *** [all] Error 2
pschen@linux[libantlr3c-3.4] 06:20 $ 
```

Modification

- Modify "antlr3config.h" (manually)
  - //#define size_t unsigned int
  - #define ANTLR3_NODEBUGGER 1

- Modify "Makefile" (manually)
  - CFLAGS = ~~-m32~~ -O2 -Wall

C Target:
Use ANTLR from the command-line

- `$java -cp antlr-3.5.2-complete.jar org.antlr.Tool test1.g`

- 產生
  - test1.c
  - test1.h
  - test1.tokens

Your C Program to Call Lexer

```c
#include    "test1.h"

int ANTLR3_CDECL
main    (int argc, char *argv[])
{
    pANTLR3_UINT8                       fName;
    pANTLR3_INPUT_STREAM                input;
    ptest1                             lxr;
    pANTLR3_COMMON_TOKEN_STREAM        tstream;
    pANTLR3_VECTOR                      the_all_tokens;
    int i;

    if (argc == 2 && argv[1] != NULL) {
        fName = (pANTLR3_UINT8) argv[1];
    } else {
        exit(1);
    }
```

Your C Program to Call Lexer

```c
// Create the input stream using the supplied file name.
input       = antlr3FileStreamNew(fName, ANTLR3_ENC_8BIT);

if (input == 0) {
    fprintf(stderr, "Failed to open file %s\n", (char *)fName);
    exit(1);
}

lxr    = test1New(input);        // test1New is generated by ANTLR

if (lxr == NULL) {
    fprintf(stderr, "Unable to create the lexer due to malloc() failure1\n");
    exit(1);
}

tstream = antlr3CommonTokenStreamSourceNew(ANTLR3_SIZE_HINT,
                                           TOKENSOURCE(lxr));

if (tstream == NULL) {
    fprintf(stderr, "Out of memory trying to allocate token stream\n");
    exit(1);
}
```

Your C Program to Call Lexer

```c
tstream = antlr3CommonTokenStreamSourceNew(ANTLR3_SIZE_HINT,
                                            TOKENSOURCE(lxr));

if (tstream == NULL) {
    fprintf(stderr, "Out of memory trying to allocate token stream\n");
    exit(1);
}


the_all_tokens = tstream->getTokens(tstream);
for (i=0; i<the_all_tokens->elementsSize; i++) {
    pANTLR3_COMMON_TOKEN the_token = (pANTLR3_COMMON_TOKEN) the_all_tokens->get(the_all_tokens, i);
    printf("Token:%d  %s\n", the_token->getType(the_token),
                             the_token->getText(the_token)->chars);
}


tstream ->free  (tstream);        tstream = NULL;
lxr     ->free  (lxr);            lxr = NULL;
input   ->close (input);          input = NULL;

return 0;
}
```

Compile Your C Program

- Set C_INCLUDE_PATH
  - $export C_INCLUDE_PATH= ...   (bash)
  - $setenv C_INCLUDE_PATH ...   (csh/tcsh)

- We should link libantlr3c.a
  - $gcc   -static   -L/cshome/professor/pschen/linuxtools/lib testLexer.c test1.c   -lantlr3c

Reference

- http://www.antlr.org

- Target C
  - C Runtime APIs
  - http://www.antlr.org/api/C/index.html

- Target Java
  - Try eclipse IDE

36

Backup

ANTLRWorks (1)

- The ANTLR GUI Development Environment.

- A novel grammar development environment for ANTLR v3 grammars written by Jean Bovet.

- Goal:
  - Make grammars more accessible to the average programmer
  - Improve maintainability and readability of grammars by providing excellent grammar navigation and refactoring tools
  - Address the most common questions and problems encountered by grammar developers

ANTLRWorks (2)

ANTLRWorks (3)

TUNNEL VISION
LABORATORIES

WORK   **PRODUCTS**   RESEARCH   CONTACT

ANTLRWorks 2

ANTLRWorks 2 is a complete rewrite of the previous grammar tool using several new techniques developed at Tunnel Vision Labs. The core framework we use for NetBeans® development was originally created for this project, but easily extends to other languages and is the starting point for all of our NetBeans-based applications.

Development Time: 18 Months (including NetBeans framework)

Download ANTLRWorks 2
v2.1 Released 8/28/13

http://tunnelvisionlabs.com/products/demo/antlrworks

ANTLRWorks+ANTLR

ANTLRWorks+ANTLR

ANTLRWorks is a GUI development environment for building ANTLR v3 grammars. It is a stand-alone Java application that you can just click on to start using ANTLR. It contains all necessary jars and is the easiest way to get started using ANTLR. 1.5 is the latest stable release and contains ANTLR v3.5.
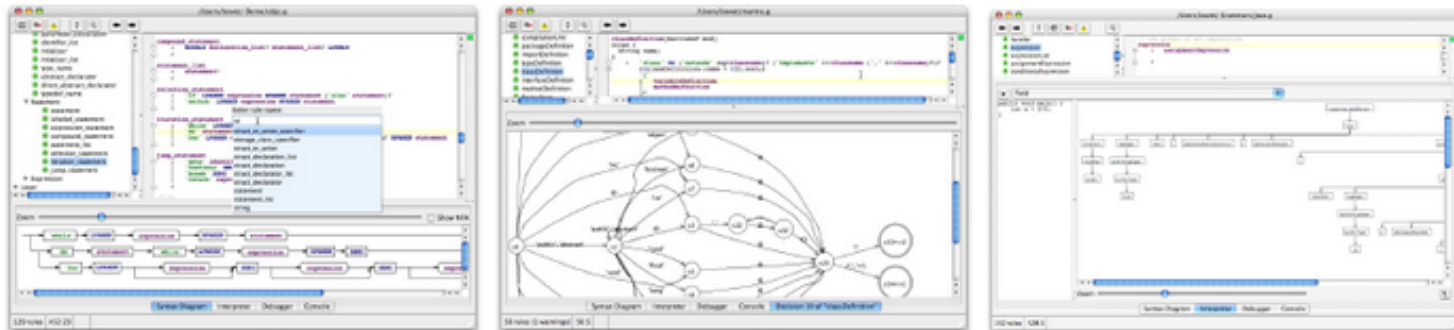
Note: ANTLRWorks requires Java 1.5 or later to run

- Version 1.5 - for Windows, Linux and Mac OS X
- Version 1.5 - bundled for Mac OS X
- Version 1.5 - source code (BSD license)