Tiger Compiler Report

1. Tiger语言描述
2. 使用工具说明

ANTLR, 语言识别的另一个工具(ANother Tool for Language Recognition)，是一种语言工具，它提供了一个框架，可以通过包含Java,C++,或C#动作的语法描述来构造语言识别器，编译器和解释器。它是由Terence Parr 领导开发（以前叫做PCCTS，Purdue Compiler Construction Tool Set，普渡大学编译器构建工具集），也是一种分析器自动生成工具，它可以接受语言的文法描述，并能产生识别这些语言的程序。

相比于YACC使用的LR/LALR的算法，ANTLR通过更好的灵活性、错误处理机制、简易的调式部分来做递归下降的语法分析。这里介绍的是ANTLR结合了手工语法处理和语法生成器的方便。ANTLR比其他语言工具容易使用。

ANTLR采用的是top-down的递归下降LL分析方法，而不是表驱动的LL/LR分析方法，相比较手工建立的递归下降分析器，表驱动的LL/LR分析器没有足够强的分析能力，并很难理解和调试。

语法分析器不仅仅用于识别语言，还应于词法分析器交互，报告语法分析的错误，构建抽象语法树，调用各种用户的动作。

ANTLR语法分析器用于构建可读的递归下降的LL(K)语法分析器，LL(K)相对于LL(1)更加易读，易于让人接受和易于表达，并且也减少左提供公因子的个数。当K>1时，支持断言。

ANTLR相比较于其他的语法分析器优点在于：

1、ANTLR集成了词法和语法的分析。

2、ANTLR接受由EBNF范式构建的语法。

3、ANTLR能自动构建语法抽象树。

4、ANTLR使用的是递归下降的语法分析器，因此语法规范和ANTLR输出之间的对应关系比较清楚。比较容易理解。

5、ANTLR能方便的自动和手工进行错误恢复和报错，自动机制能简单有效地分析许多语法情况:手动的机制被称为“语法分析异常处理”，简化了高质量的错误处理。

6、ANTLR允许每个语法规则都有参数和返回值，方便在语法分析中的属性传递。一个规则的参数会被转换成一个函数的参数，ANTLR中可以有多个返回值。

7、ANTLR还有众多的功能，使其成为一个产品，而不仅仅是研究性的工程。而且ANTLR是用java语言写的，可方便移植和调式。很容易整合到项目的应用程序中。

ANTLR并不是使用纯正的LL(K)算法，它使用了最小lookahead的决策算法对LL(K)算法进行改进，加速了语法的分析和分析速度，并减少了 分析代码的尺寸，但对于大多数的决策，向前lookahead 1个token已经足够，因此ANTLR语法分析器的效率近似于LL(1)递归下降分析器。也使其复杂度从O(m^K)降到O(m\*k);因此也大大提高 了分析效率。

此外，ANLLR的LL(K)算法还有一个重要特性，即支持断言，能有效的消除冲突，断言分为语法断言和语义断言。

1. 词法分析lexer

COMMENT: '/\*' (COMMENT | ~('\*'|'/') | '\*'~'/' | '/'~'\*')\*? '\*/' -> skip;

STRING: '"' (SLASH | ~'\\')\*? '"';

ASSIGN: ':=';

COMMA: ',';

SEMI: ';';

COLON: ':';

DOT: '.';

LT: '<';

LE: '<=';

GT: '>';

GE: '>=';

EQ: '=';

NEQ: '<>';

AND: '&';

OR: '|';

PLUS: '+';

MINUS: '-';

MUL: '\*';

DIV: '/';

LPAR: '(';

RPAR: ')';

LSQR: '[';

RSQR: ']';

LBCE: '{';

RBCE: '}';

NIL: 'nil';

NEW: 'new';

IF: 'if';

THEN: 'then';

ELSE: 'else';

WHILE: 'while';

DO: 'do';

FOR: 'for';

TO: 'to';

BREAK: 'break';

LET: 'let';

IN: 'in';

END: 'end';

CLASS: 'class';

EXTENDS: 'extends';

VAR: 'var';

TYPE: 'type';

METHOD: 'method';

FUNCTION: 'function';

ARRAY: 'array';

OF: 'of';

PRIMITIVE: 'primitive';

IMPORT: 'import';

ID: LETTER (LETTER|DIGIT|'\_')\* | '\_main';

INT: DIGIT+;

WHITESPACE : [ \t\r\n]+ -> skip;

fragment LETTER : [a-zA-Z] ;

fragment DIGIT : [0-9] ;

fragment SLASH : '\\'([abfnrtv'"''\\'] | [0-7] | [1-7][0-7] | [1-3][0-7][0-7] | [xX][0-9a-fA-F] | [xX][1-9a-fA-F][0-9a-fA-F]);

1. 语法分析grammar
2. 语法分析树
3. 语义分析semantic
4. 错误检查与修复
5. 额外功能
6. 体会