

语法分析

(4. ANTLR4 递归下降语法分析器)

魏恒峰

hfwei@nju.edu.cn

2021 年 11 月 26 日



ANTLR3 语法分析器是如何工作的？

ANTLR3 语法分析器是如何工作的？

ANTLR4 是如何处理左递归与优先级的？

ANTLR3 语法分析器是如何工作的？

ANTLR4 是如何处理左递归与优先级的？

ANTLR4 是如何进行错误报告与恢复的？

ANTLR3 语法分析器是如何工作的?

带记忆功能的可回溯的递归下降的语法分析器



(适用于 ANTLR3 与部分 ANTLR4)

甚至可以使用谓词解析器处理上下文相关文法

$LL(1)$

tpdsl: rd/NameList.g4

LL(1)

tpdsl: rd/NameList.g4

tpdsl: rd/ListParser.java
(elements())

LL(1)

```
tpdsl: rd/NameList.g4
```

```
tpdsl: rd/ListParser.java  
      (elements())
```

```
tpdsl: rd/NameListParser.java
```

$LL(k = 2)$

tpdsl: multi/NameListWithAssign.g4

$LL(k = 2)$

tpdsl: multi/NameListWithAssign.g4

tpdsl: multi/LAParser.java
(element())

$LL(k = 2)$

```
tpdsl: multi/NameListWithAssign.g4
```

```
tpdsl: multi/LAParser.java  
      (element())
```

```
tpdsl: multi/NameListWithAssignParser.java  
      (.adaptivePredict())
```

Backtrack (回溯)

```
tpdsl: backtrack/NameListWithParallelAssign.g4
```

Backtrack (回溯)

```
tpdsl: backtrack/NameListWithParallelAssign.g4
```

```
tpdsl: backtrack/BacktrackParser.java  
      (stat())
```

Backtrack (回溯)

```
tpdsl: backtrack/NameListWithParallelAssign.g4
```

```
tpdsl: backtrack/BacktrackParser.java  
      (stat())
```

```
tpdsl: backtrack/NameListWithParallelAssignParser.java  
      (.adaptivePredict())
```

Backtrack (回溯)

```
tpdsl: backtrack/NameListWithParallelAssign.g4
```

```
tpdsl: backtrack/BacktrackParser.java  
      (stat())
```

```
tpdsl: backtrack/NameListWithParallelAssignParser.java  
      (.adaptivePredict())
```

ANTLR4 不需要回溯, 这是 ANTLR4 的一大创新之处

ANTLR: A Predicated- $LL(k)$ Parser Generator

T. J. PARR

*University of Minnesota, AHPCRC, 1100 Washington Ave S Ste 101, Minneapolis, MN 55415, U.S.A.
(email: parrt@acm.org)*

AND

R. W. QUONG

*School of Electrical Engineering, Purdue University, W. Lafayette, IN 47907, U.S.A.
(email: quong@ecn.purdue.edu)*

$LL(*)$: The Foundation of the ANTLR Parser Generator

Terence Parr

*University of San Francisco
parrt@cs.usfca.edu*

Kathleen Fisher *

*Tufts University
kfisher@eecs.tufts.edu*

Adaptive $LL(*)$ Parsing: The Power of Dynamic Analysis

Terence Parr

*University of San Francisco
parrt@cs.usfca.edu*

Sam Harwell

*University of Texas at Austin
samharwell@utexas.edu*

Kathleen Fisher

*Tufts University
kfisher@eecs.tufts.edu*

ANTLR: A Predicated- $LL(k)$ Parser Generator

T. J. PARR

*University of Minnesota, AHPCRC, 1100 Washington Ave S Ste 101, Minneapolis, MN 55415, U.S.A.
(email: parrt@acm.org)*

AND

R. W. QUONG

*School of Electrical Engineering, Purdue University, W. Lafayette, IN 47907, U.S.A.
(email: quong@ecn.purdue.edu)*

$LL(*)$: The Foundation of the ANTLR Parser Generator

Terence Parr

*University of San Francisco
parrt@cs.usfca.edu*

Kathleen Fisher *

*Tufts University
kfisher@eecs.tufts.edu*

Adaptive $LL(*)$ Parsing: The Power of Dynamic Analysis

Terence Parr

*University of San Francisco
parrt@cs.usfca.edu*

Sam Harwell

*University of Texas at Austin
samharwell@utexas.edu*

Kathleen Fisher

*Tufts University
kfisher@eecs.tufts.edu*

[courses-at-nju-by-hfwei/compilers-papers-we-love](#)

ANTLR4 如何处理左递归与优先级的?

parserllantlr/LRExpr.g4

parserllantlr/LRExpr.g4

antlr4 LRExpr -Xlog

Grammar Rewriting

2021-11-25 17:44:23:815 left-recursion LogManager.java:25 expr

```
: ( {} INT<tokenIndex=45>  
  | ID<tokenIndex=51>  
  )  
  (  
    {precpred(_ctx, 4)}?<p=4> '*'<tokenIndex=27> expr<tokenIndex=29,p=5>  
    | {precpred(_ctx, 3)}?<p=3> '+'<tokenIndex=37> expr<tokenIndex=39,p=4>  
  )*  
;
```

stat : expr ';' EOF;

```
expr : expr '*' expr  
      | expr '+' expr  
      | INT  
      | ID  
      ;
```

```

expr[int _p]
: ( INT
  | ID
  )
  ( {4 >= $_p}? '*' expr[5]
    | {3 >= $_p}? '+' expr[4]
  )*
;

```

expr[int _p]

```

stat : expr ';' EOF;

```

```

expr : expr '*' expr
      | expr '+' expr
      | INT
      | ID
;

```

```

expr[int _p]
: ( INT
  | ID
  )
  ( {4 >= $_p}? '*' expr[5]
    | {3 >= $_p}? '+' expr[4]
  )*
;

```

1 + 2 + 3 1 + 2 * 3 1 * 2 + 3

parserllantlr/LRExprParen.g4

```
stat : expr ';' EOF;
```

```
expr : expr '*' expr  
      | expr '+' expr  
      | '(' expr ')'         
      | INT  
      | ID  
      ;
```

parserllantlr/LRExprParen.g4

```
stat : expr ';' EOF;
```

```
expr : expr '*' expr  
      | expr '+' expr  
      | '(' expr ')'         
      | INT  
      | ID  
      ;
```

antlr4 LRExprParen -Xlog

ANTLR4 是如何进行**错误报告与恢复**的?

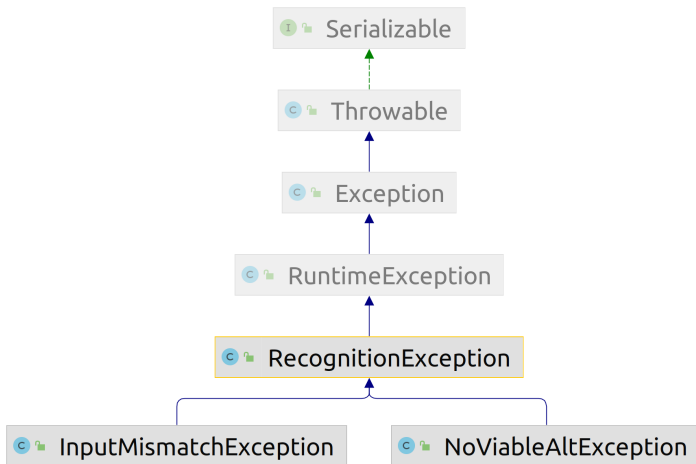
语法分析阶段的主题之三: 错误恢复



报错、**恢复**、继续分析



恐慌/应急 (Panic) 模式: 假装成功、调整状态、继续进行



InputMismatchException

InputMismatchException

如果下一个词法单元符合预期,
则采用“单词法符号移除”或“单词法符号补全”策略

InputMismatchException

如果下一个词法单元符合预期,
则采用“单词法符号移除”或“单词法符号补全”策略

`Class.g4`

`Class-RemoveToken.txt`

`Class-AddToken.txt`

NoViableAltException

NoViableAltException

采用“同步-返回 (sync-and-return)”策略,
从当前非终结符中恢复

NoViableAltException

采用“同步-返回 (sync-and-return)”策略,
从当前非终结符中恢复

Group.g4

Group-Sync.txt

NoViableAltException

采用“同步-返回 (sync-and-return)”策略,
从当前非终结符中恢复

Group.g4

Group-Sync.txt

注意 FOLLOW (静态) 集合与 FOLLOWING (动态) 集合的区别

如何从子规则中优雅地恢复出来？

Class.g4 (member+)

如何从子规则中优雅地恢复出来？

Class.g4 (member+)

Class-Subrule-Start.txt (“单词法符号移除”)

如何从子规则中优雅地恢复出来？

Class.g4 (member+)

Class-Subrule-Start.txt (“单词法符号移除”)

Class-Subrule-Loop.txt (“另一次 member 迭代”)

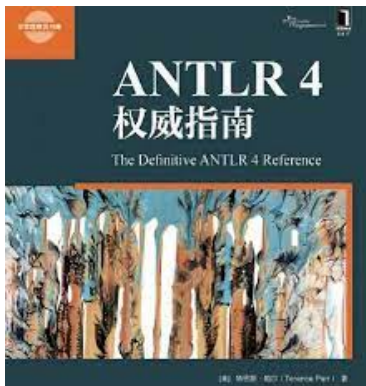
如何从子规则中优雅地恢复出来？

Class.g4 (`member+`)

Class-Subrule-Start.txt (“`单词法符号`移除”)

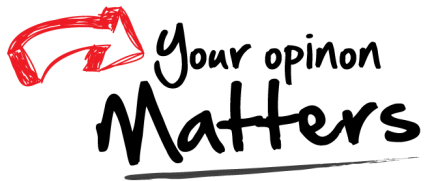
Class-Subrule-Loop.txt (“`另一次 member` 迭代”)

Class-Subrule-End.txt (“`退出当前 classDef` 规则”)



第 9 章: 错误报告与恢复

Thank
You!



Office 926

hfwei@nju.edu.cn