


An Brief Introduction to ANTLR4

“ ANother Tool for Language Recognition ”

References

- 官网 <https://www.antlr.org/>
 - 包括 book *antlr4* 权威指南 
- Github Repo
 - docs 
- 简明教程-中

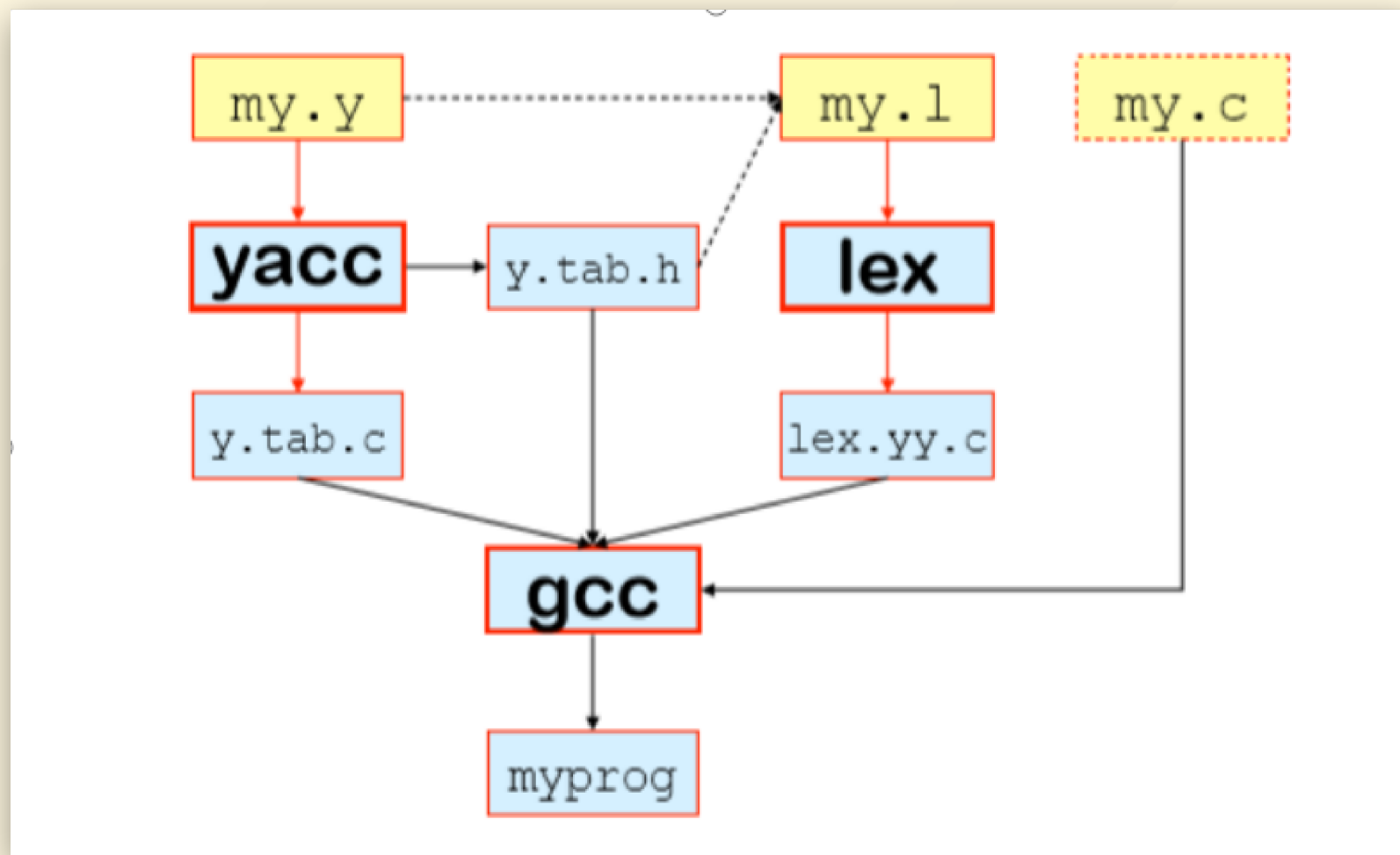
Overview

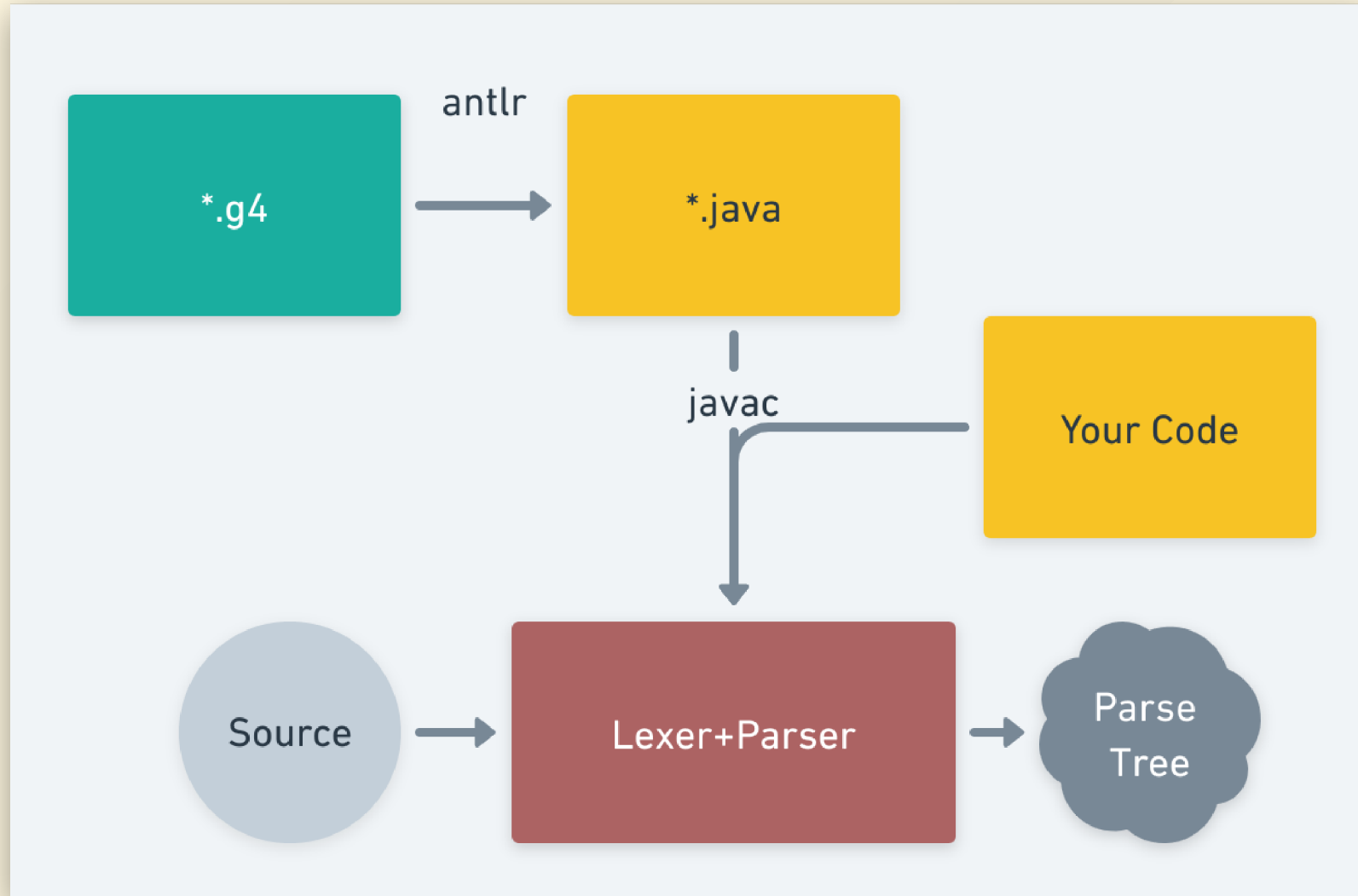
$$ANTLR = flex + bison$$

$$source \xrightarrow{flex} tokens \xrightarrow{bison} parse\ tree$$

$$source \xrightarrow{ANTLR} parse\ tree$$

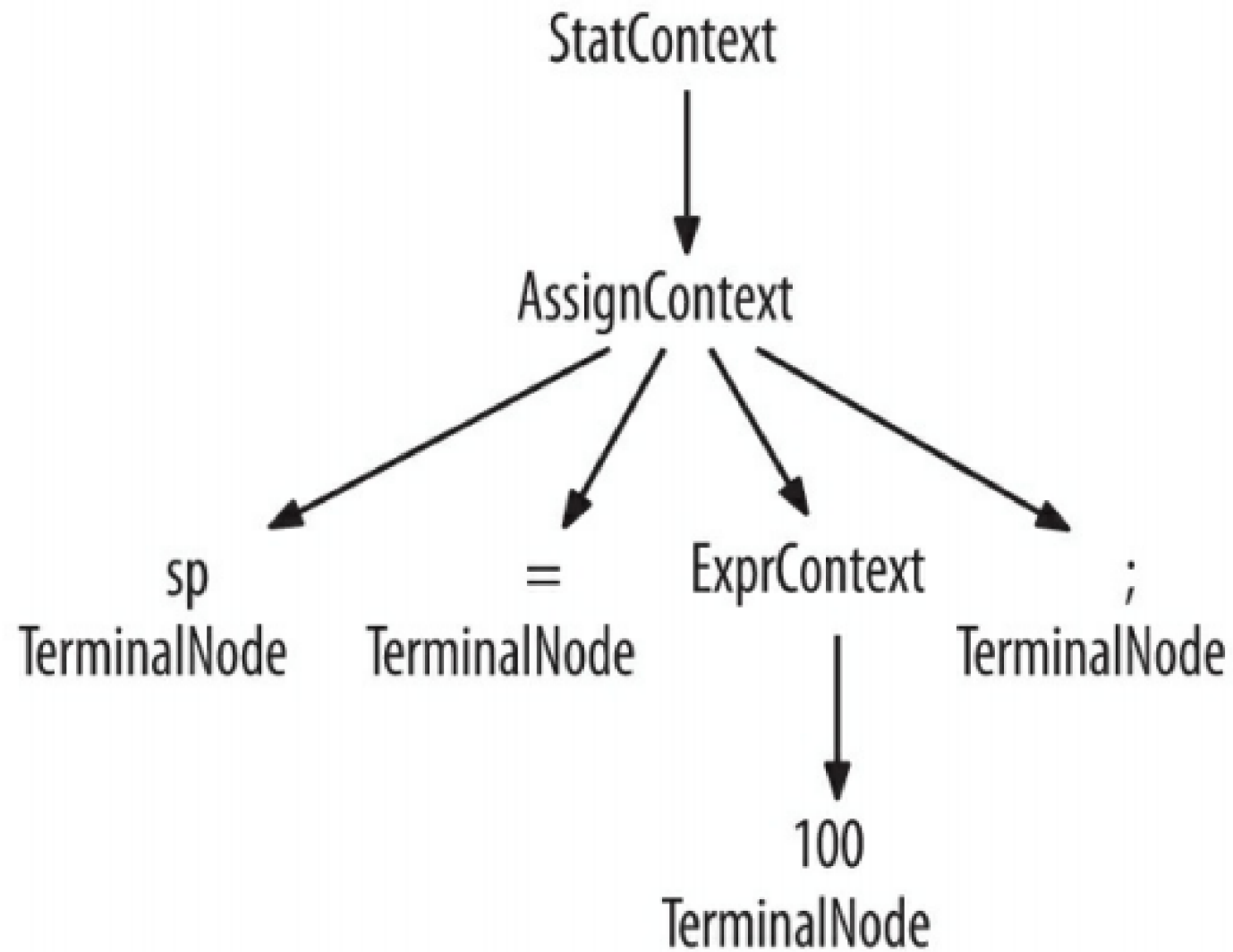
ALL(*)的自顶向下的语法分
析器







第二个 词法分析器 应为 语法分析器



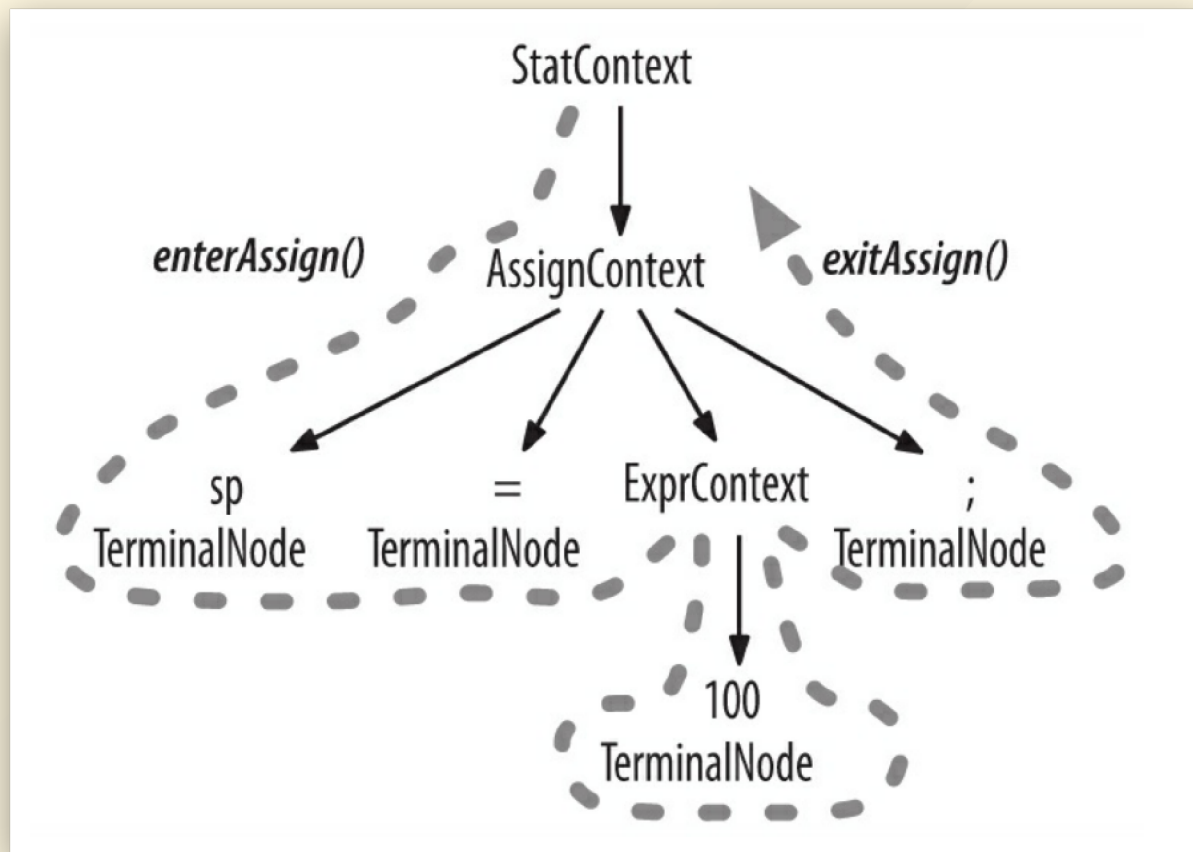
类似 EBNF 的语法

```
grammar Hello;  
  
r  :  'hello' ID;  
  
ID :  [a-z]+;  
WS :  [ \t\r\n]+ -> skip;
```

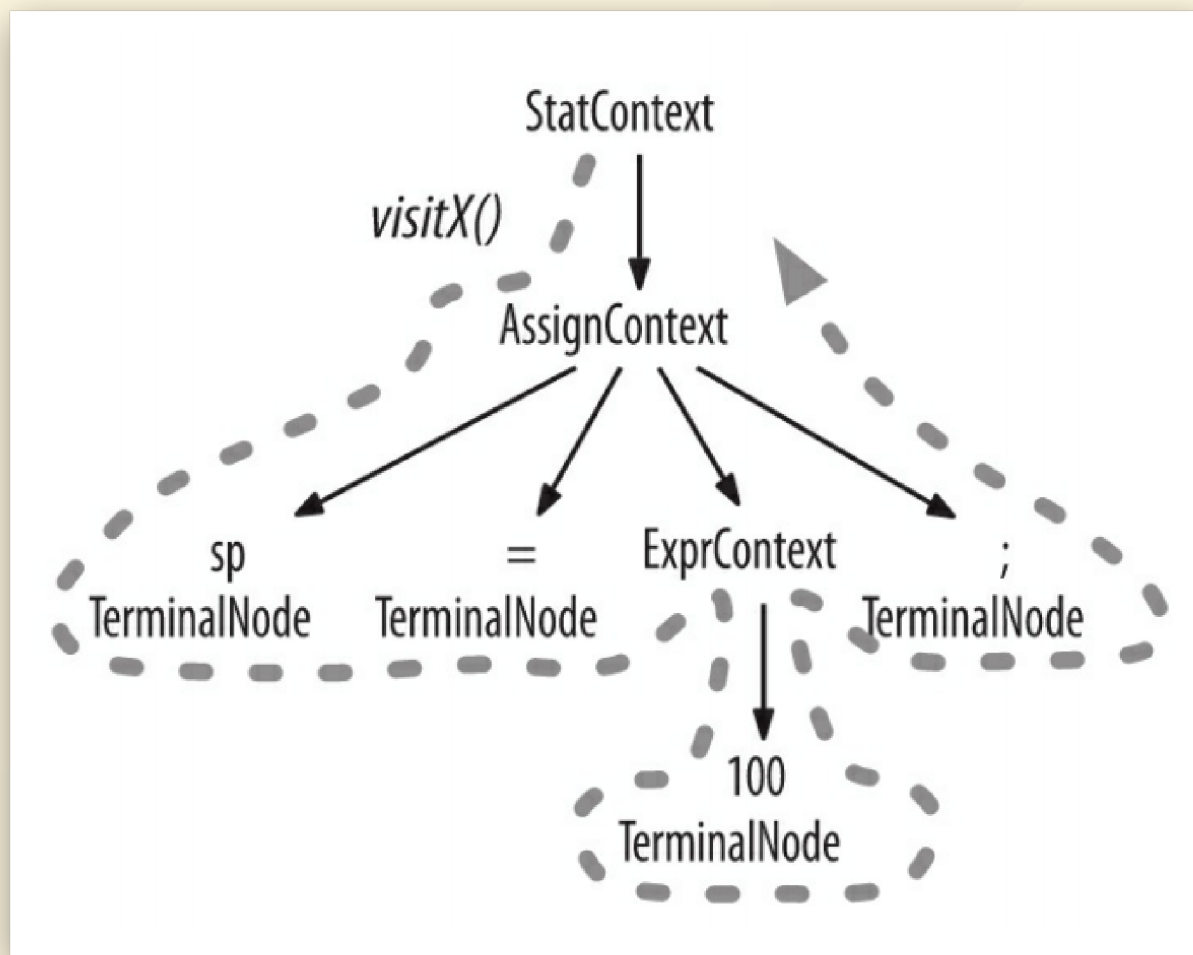

antlr 会生成什么功能的文件？

- 提供基本的语言识别能力
 - *Lexer.java
 - *Parser.java
- 提供访问生成的 Parse Tree 的能力
 - *Listener.java
 - 被动的在树被遍历的时候触发
 - *Visitor.java 默认不生成
 - 主动访问某一个节点

Listener



Visitor



.g4 语法格式

- 类似 Java 的 package 机制
- 词法规则使用大写字母开头，语法规则使用小写字母开头
- 别名
- 分支按照上下顺序区分优先级
- 其他
<https://github.com/antlr/antlr4/blob/master/doc/grammars.md>

```
// 注意和文件名一样
grammar Expr;
// 导入其他的语法规则
import CommonLexerRules;

prog: stat+;

stat: expr NEWLINE          # printExpr
    | ID '=' expr NEWLINE # assign
    | NEWLINE # blank
    | CLEAR # clear
    ;

expr: expr op=('*' | '/') expr # MulDiv // `#` 后面内容用于定义别名, 可选
    | expr op=('+' | '-') expr # AddSub
    | INT # int
    | ID # id
    | '(' expr ')' # parens
    ;

MUL: '*';
DIV: '/';
ADD: '+';
SUB: '-';
```

```
// 表示是 lexer 文件, 注意要与文件名一样  
lexer grammar CommonLexerRules;
```

```
// Lexer Rule  
ID: [a-zA-Z]+;  
INT: [0-9]+;  
NEWLINE: '\r'? '\n';  
WS: [ \t] -> skip;  
CLEAR: 'clear';
```

an example

1. 从 `.g4` 到 `*.java`
2. 使用内置的测试工具 `TestRig` 打印 Parse Tree
3. 结合我们的启动代码，打印 Parse Tree
4. 使用 `visitor` 访问树并解析

其他注意事项 ⚠

- 导出包名 📦

- `antlr -package bit.minisys.minicc.parser.internal.antlr ...`

the Things you should do next 🚀

- 学习 antlr4 cli 工具
 - <https://github.com/antlr/antlr4/blob/master/doc/tool-options.md>
- 尝试跟着教程复现一个小例子 🍄 以学习 `Listener` `Visitor` 等操作
- 可选的尝试 antlr4 提供的 Maven 和 Gradle 的插件。