The Little Leex & Yecc

ZhenyuanLua

前言

Leex & Yecc 是 Erlang 版的 Flex & Yacc(Bison).
Flex & Yacc 用于开发编译器和解释器, 适用于任何使用模式匹配输入的应用,
如计算器/解释器/编译器/领域特定语言.
Flex & Yacc 应用领域, Leex & Yecc 亦可往之.

本书内容

较 <Flex & Bison>, 本书先理解 Leex & Yecc 规范, 使用 Leex & Yecc, 增加语法可视化内容.
具体内容如下:
第一章, 简介语法分析相关知识.
第二章, 简介 BNF 文法, W3C EBNF 记法, 语法可视化.
第三章, 理解 Leex 规范, 熟悉 Leex 词法文件定义, 生成的词法解析器代码.
第四章, 理解 Yecc 规范, 熟悉 Yecc 语法文件定义. 生成的语法解析器代码.
第五章, Leex 实战, 使用 Leex 开发单词统计应用.
第六章, Yecc 实战, 使用 Yecc 联合 Leex 开发计算器应用.
第七章, 理解 LALR-1 分析算法.
第八章, 基于 Yecc/Leex 实现 JSON 解释器.

排版约定

本书应用了以下排版约定:
标点符号: 使用英文标点加空格.

本书范例

本书的范例程序可以在线获得:

```
https://www.github.com/zhenyuanlau/leex-yecc-book/code
```

致谢

- Flex & Bison
- TheBeamBook
- AsciiDoc

Chapter 1. 简介

代码分析过程分为词法分析和语法分析两个部分

- 词法分析, 把输入分割成一个个有意义的词条, 称为 Token.
- 语法分析, 确定 Token 间的关系.

1.1. 词法分析

词法分析是在输入中寻找字符的模式, 正则表达式是对模式的描述.
Leex 生成的词法分析器, 读取输入, 匹配输入与指定的正则表达式, 执行匹配所关联的代码.
词法分析器内部是一个 DFA.

1.1.1. 正则表达式

1.1.2. DFA

1.2. 语法分析

语法分析器的任务是找出输入 Token 之间的关系, 构建语法分析树.

1.2.1. BNF 文法

BNF 是上下文无关文法的标准, 用来描述 Token 流转换成语法分析树的规则.

Chapter 2. BNF 文法

2.1. 语法描述

W3C EBNF Syntax

```
Grammar  ::= Production*
Production
         ::= NCName '::=' ( Choice | Link )
NCName   ::= [http://www.w3.org/TR/xml-names/#NT-NCName]
Choice   ::= SequenceOrDifference ( '|' SequenceOrDifference )*
SequenceOrDifference
         ::= (Item ( '-' Item | Item* ))?
Item     ::= Primary ( '?' | '*' | '+' )*
Primary  ::= NCName | StringLiteral | CharCode | CharClass | '(' Choice
')'
StringLiteral
         ::= '"' [^"]* '"' | "'" [^']* "'"
CharCode ::= '#x' [0-9a-fA-F]+
CharClass
         ::= '[' '^'? ( Char | CharCode | CharRange | CharCodeRange )+ ']'
Char     ::= [http://www.w3.org/TR/xml#NT-Char]
CharRange
         ::= Char '-' ( Char - ']' )
CharCodeRange
         ::= CharCode '-' CharCode
Link     ::= '[' URL ']'
URL      ::= [^#x5D:/?#]+ '://' [^#x5D#]+ ('#' NCName)?
Whitespace
         ::= S | Comment
S        ::= #x9 | #xA | #xD | #x20
Comment  ::= '/*' ( [^*] | '*'+ [^*/] )* '*'* '*/'
```

2.2. EBNF 相关概念

2.2.1. 语法

语法是产生规则的集合.

2.2.2. 产生规则

产生规则格式如下:

```
symbol ::= expression
```

2.2.3. 非终结符

非终结符可以在规则的左手边, 也可以在规则的右手边.

产生规则的左手边是一个符号, 右手边是一个表达式.
表达式由终结符/非终结符/操作符构成.

2.2.4. 终结符

终结符出现在规则的右手边, 可以是符号/字符串/模式.

2.2.5. 语法操作符

Table 1. 语法操作符表

操作符	说明
A?	可选 A
A B	匹配 A 后跟着 B
A | B	匹配 A 或者 B
A - B	匹配 A 不匹配 B
A+	至少匹配一个 A
A*	匹配 0 个或多个 A

2.2.6. EBNF 代码示例

算术表达式语法

```
expression ::= term | term "+" expression
term       ::= factor | factor "*" term
factor     ::= constant | variable | "(" expression ")"
variable   ::= "x" | "y" | "z"
constant   ::= digit | digit constant
digit      ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

2.3. 语法可视化

Chapter 3. Leex 参考

Chapter 4. Yecc 参考

Chapter 5. Leex 实战

Chapter 6. Yecc 实战

Chapter 7. LALR-1 算法分析

Chapter 8. JSON 解析器

参考

W3C EBNF Notation
Railroad Diagram Generator
EDoc Leex
EDoc Yecc