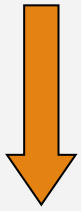
编译原理Bison作业

2015.11.09

Overview

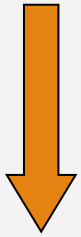
使用Flex、Bison解释正则表达式，以文本方式输出其语法结构。

输入：A|B*|C

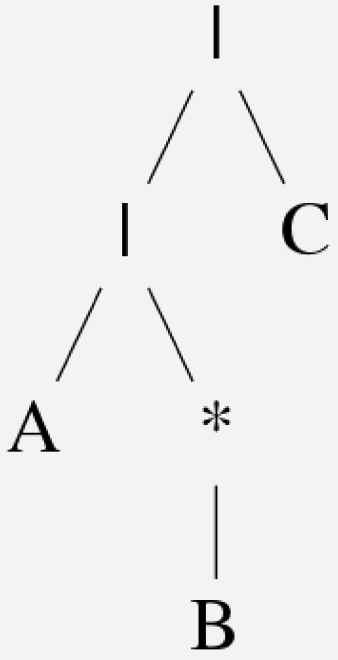


Flex、
Bison

中间表示：语法树



先序遍历



输出：Alt(Alt(Lit(A), Star(Lit(B))), Lit(C))

正则表达式

元字符	行为	示例
*	零次或多次匹配前面的字符或子表达式。	zo* 与“z”和“zoo”匹配。
+	一次或多次匹配前面的字符或子表达式。	zo+ 与“zo”和“zoo”匹配，但与“z”不匹配。
?	零次或一次匹配前面的字符或子表达式。 任何其他限定符（*、+、?）之后时，匹配模式是非贪婪的。 非贪婪模式匹配搜索到的、尽可能少的字符串， 而默认的贪婪模式匹配搜索到的、尽可能多的字符串。	zo? 与“z”和“zo”匹配，但与“zoo”不匹配。 o+? 只与“oooo”中的单个“o”匹配，而 o+ 与所有“o”匹配。
|	指示在两个或多个项之间进行选择。	z|food 与“z”或“food”匹配。 (z|f)ood 与“zood”或“food”匹配。
.	匹配除换行符 \n 之外的任意单个字符。	a.c 与“abc”、“a1c”和“a-c”匹配。
()	标记一个子表达式的开始和结束，对括号中匹配的内容进行捕获。	A(and|or)B(and|or)C 与“AandBorC” 匹配，而且捕获括号匹配的内容分别为“and”和“or”。
(?:)	标记一个子表达式的开始和结束，但不捕获括号中匹配的内容。	A(?:and|or)B(and|or)C 与“AandBorC” 匹配，只捕获第二个括号匹配的内容为“or”。

正则表达式

字母表

- ASCII字符。
- 除了前面定义的特殊符号，其它字符均表示匹配该字符本身。
- 无转义操作，因此不能匹配特殊符号本身。
  - 如："1.2"匹配"1.2"、"1-2"或"112"。
  - 如："1\.2"匹配"1\.2"、"1\-2"或"1\12"。

正则表达式

子表达式捕获

- 捕获，即获取正则表达式某个部分实际匹配到的内容。
- 以所有捕获块左括号出现的次序作为该括号匹配内容的索引。
  - 如：“((a)(?:b)(c))”匹配“abc”，则$1=“abc”，$2=“a”，$3=“c”。
- 捕获块和非捕获块可以随意嵌套。
  - 如：“((a))”、“((?:a))”、“((?:a))”、“(?:(a))”都是合法的。
  - 如：“(?:(a)(b))”匹配“ab”，则$1=“a”，$2=“b”。
- 捕获块和非捕获块不能为空。
  - 如：“a()b”或“a(?:)b”都是不合法的。

正则表达式

优先级（高到底）

- () (?:)
- * + ?
- 连接
- |

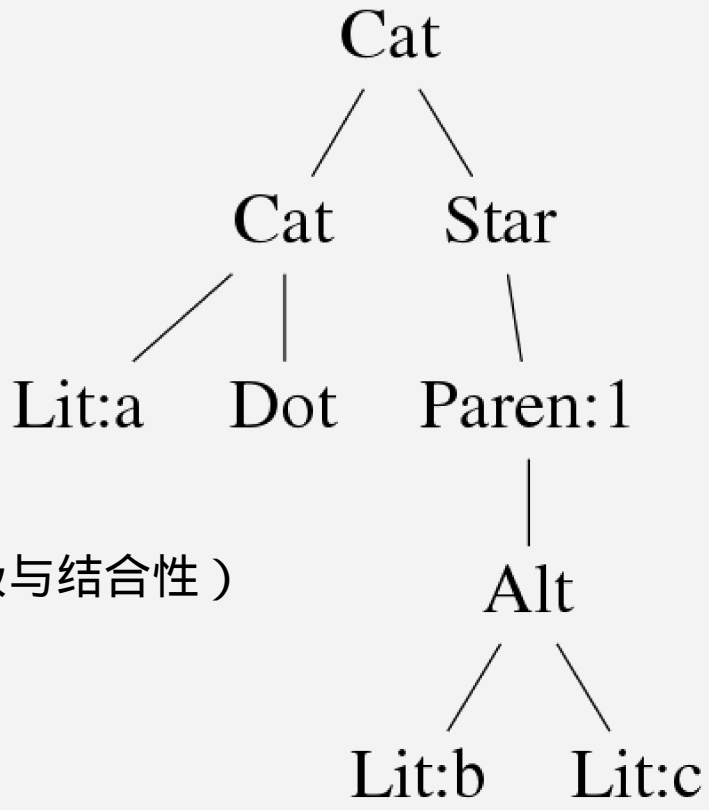
结合性

- 左结合性：连接 |
  - 如：忽略捕获性，"abc"相当于"(ab)c"
  - 如：忽略捕获性，"a|b|c"相当于"(a|b)|c"

语法结构

Example

a.(b|c)*



正则表达式自左向右（遵循符号优先级与结合性）
语法树自底向上

语法结构

叶子节点

- 单个字符(Lit:c)，"."(Dot)

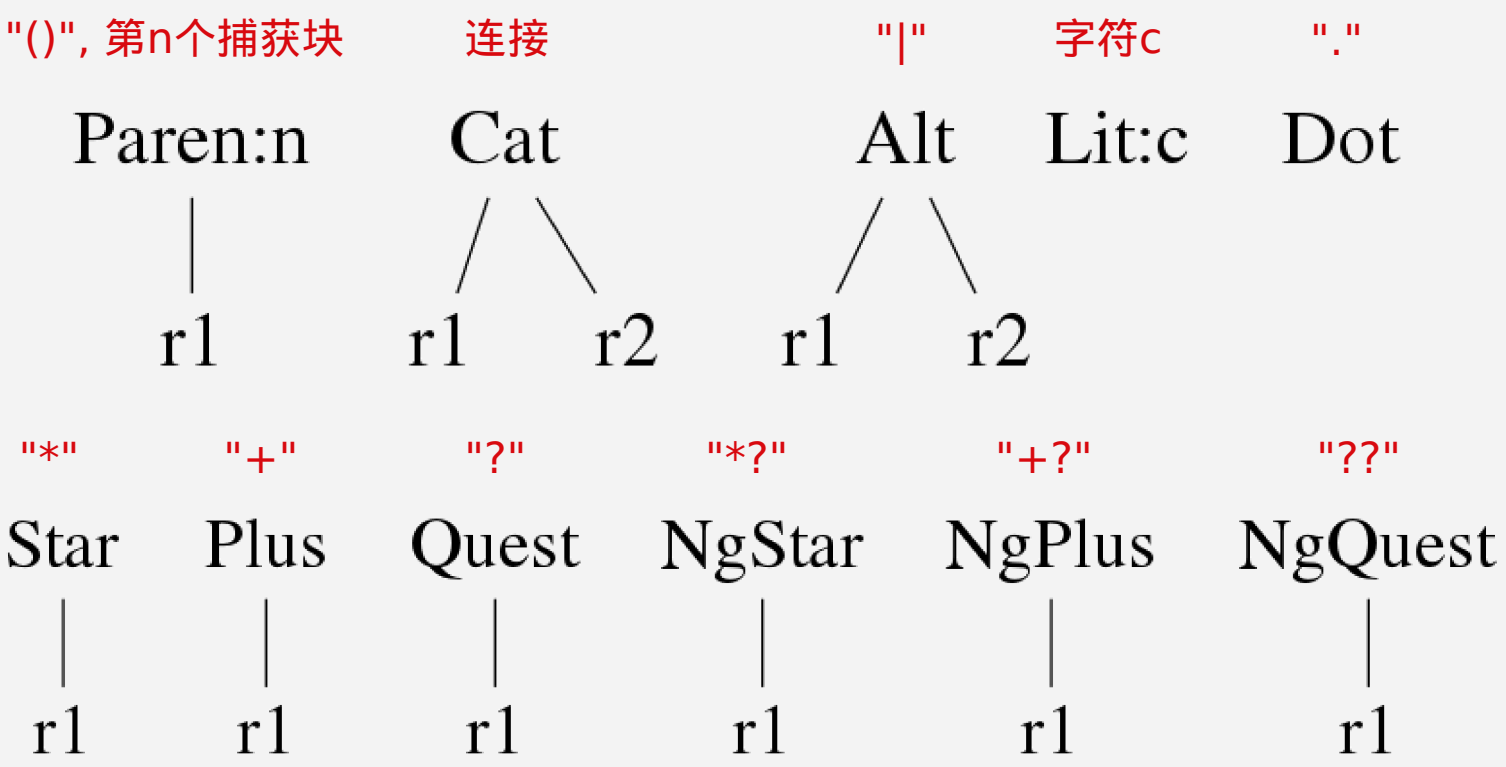
中间节点

- 包括所有的特殊符号和连接操作符

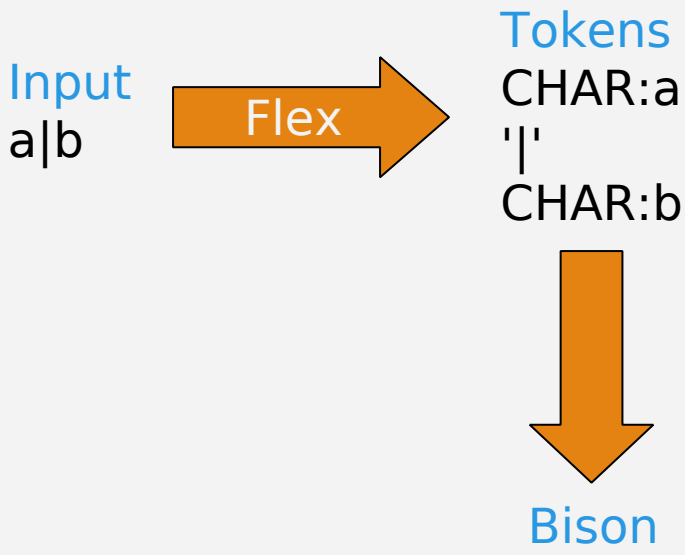
语法结构

"(?:)"无需任何表示

r1、r2表示正则表达式中该特殊符号操作的(左右)对象生成的语法结构。



语法树构造




```
alt: concat
     | alt '|' concat    {$$=createNode(Alt,$1,$3);}
single: CHAR             {$$=createNode(Lit);}
```

语法树构造（参考）

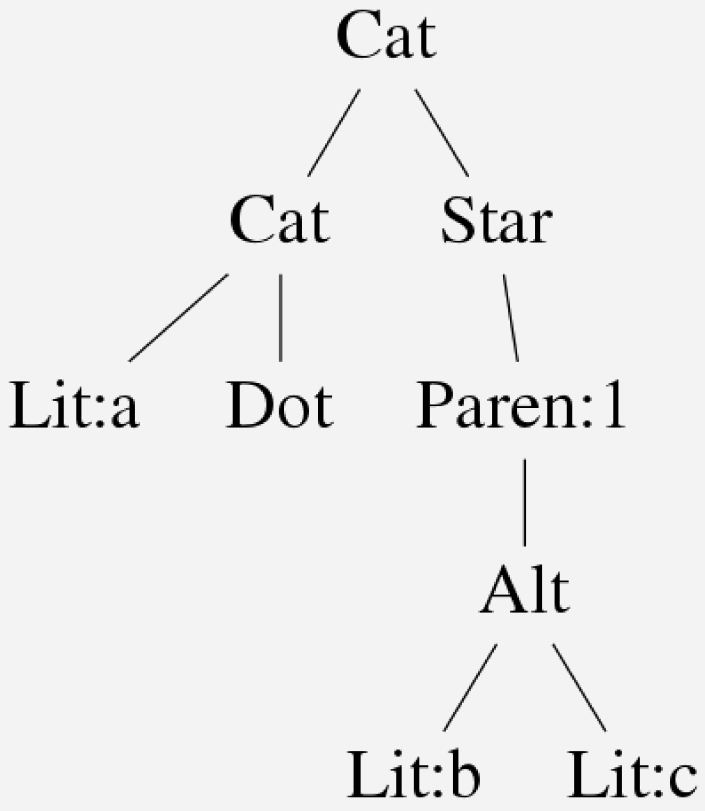
	PRODUCTION	SEMANTIC RULES
1)	$E \rightarrow E_1 + T$	$E.node = \textbf{new}\ Node('+', E_1.node, T.node)$
2)	$E \rightarrow E_1 - T$	$E.node = \textbf{new}\ Node('-', E_1.node, T.node)$
3)	$E \rightarrow T$	$E.node = T.node$
4)	$T \rightarrow (E)$	$T.node = E.node$
5)	$T \rightarrow \textbf{id}$	$T.node = \textbf{new}\ Leaf(\textbf{id}, \textbf{id}.entry)$
6)	$T \rightarrow \textbf{num}$	$T.node = \textbf{new}\ Leaf(\textbf{num}, \textbf{num}.val)$

Figure 5.10: Constructing syntax trees for simple expressions

翻译方案

Example

a.(b|c)*



翻译结果：Cat(Cat(Lit(a), Dot), Star(Paren(1, Alt(Lit(b), Lit(c)))))

翻译方案

r1、r2表示左右子树的翻译结果。

Dot	->	"Dot"
Lit:c	->	"Lit(c)"
Alt	->	"Alt(r1, r2)"
Cat	->	"Cat(r1, r2)"
Paren:n	->	"Paren(n, r1)"
Star	->	"Star(r1)"
Plus	->	"Plus(r1)"
Quest	->	"Quest(r1)"
NgStar	->	"NgStar(r1)"
NgPlus	->	"NgPlus(r1)"
NgQuest	->	"NgQuest(r1)"

作业说明

内嵌代码语言

- C语言

输入输出

- 使用重定向，linux下即*./regex < regex.input > regex.output*

特殊函数

- main、yywarp、yyerror函数可以自定义，也可以使用gcc编译选项-ly -ll，编译器优先选择用户自定义的函数
- 如果没有自定义main函数，编译选项-ly -ll顺序不能颠倒，-ly链接语法分析的main函数，而-ll链接词法分析的main函数

格式说明

输入文件

- 每一行为一条合法的正则表达式，不含空行，最后一条正则表达式后面有换行符。

输出文件

- 每一行对应该行正则表达式的翻译结果，每个逗号后面紧接一个空格，最后一行的换行符可保留也可去掉。

作业提交

提交文件：regex.l regex.y 其它用户定义头文件等
注意：需保证*bison -d regex.y && flex regex.l && gcc lex.yy.c regex.tab.c -ly -ll -o regex*能正常生成可执行程序

提交时间：11月22号 晚12:00前提交

提交方式：ftp://115.28.17.113，帐号密码sdcs

命名规范：学号-名字拼音-hw2.zip（里面直接就是代码文件）
如：12345678-xiaoming-hw2.zip
二次提交：12345678-xiaoming-hw2-v2.zip，以此类推