

2005 年 9 月 8 日

6.035 词法分析/语法分析程序项目

6.035 词法分析/语法分析程序项目

Punyashloka Viswal

麻省理工学院电气工程与计算机科学系

今天

- 项目信息
- 词法分析/语法分析程序
 - 是什么？为什么？
 - 工具

项目信息

- 请在 6.035 小组内部进行工作
- 今日稍后时间，学生将得到分组信息
- 小组成员和教师对本小组的信息有全部的访问权限——其他人则没有
- 可使用 CVS——很好记录程式码修改的过程！
- 在 MIT 服务器上（-Xmx64M）上提供有 Java 1.5.0 存储器
- 在课程 6.170 部分提供有最新版本的 Apache Ant（add -f 6.170）
- 在课程 6.035 服务器上提供有讲座中使用的代码

词法分析程序

- 将字符流转化为符号流
- **符号**：可以视为一个整体的字符序列
- **于**编译器相关的都为符号串

- 不考虑注释，空格
- 使用标点符号来定义符号（如字符串）
- 把几乎其它所有的都视为标志符

词法分析器示例

```
• class program{  
//不提供信息的注释  
    void main() {  
        callout(“printf”,“%d”,42);  
    }  
}
```

变为：

CLASS 标志（“Program”）左大括弧 VOID 标志（“main”）左括弧 左大括弧 CALLOUT 左括弧 字符串（“printf”） 通用字符串（“%d”） 通用数（42） 右括弧 分号 右大括弧 右

大括弧

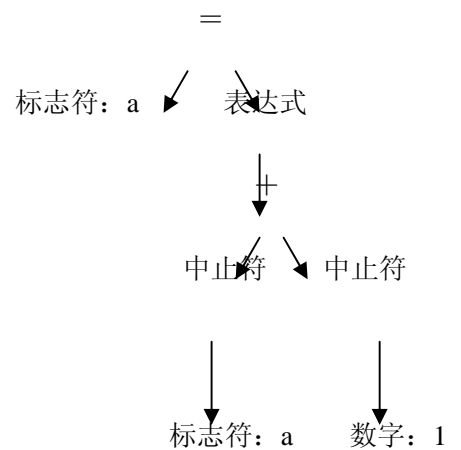
词法分析器生成

- ~~用手写~~ 使用 **JLex**
- 词法分析器生成器：词汇规范 (.lex) → 词法分析器自动机 (.java)
- 和语法分析器交互使用：使用从 CUP 二来的符号名
- 规则示例：

```
if {return tok(sym.IF,NULL);}
[a-z][a-z0-9]* {return tok(sym.ID,ytext());}
```
- 常规表达式并不能做很多事情，但是他们却决不会递归！

语法分析器

- 将符号流转换为语言结构实体
- **语法分析树**捕捉了语言结构
- `a=a+1` 变为 →
- 编译器的后一阶段就在语法分析树上操作 () 而不是程序文本

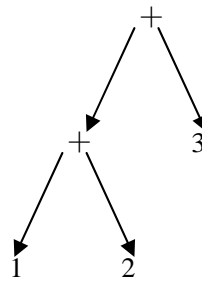
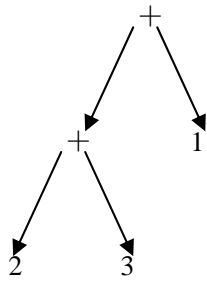


语法分析器生成

- **CUP 生成语法分析器** (parser.java 和 sym.java)
- ...从上下文无关语法:
 - 中止符 INT 类型, 标志符,加号, 赋值 (大写);
 - 非中止符: 赋值 (小写), 表达式;
 - 开始于 赋值 (小写);
 - 赋值: =标志符赋值于表达式;
 - 表达式: =表达式赋值于表达式;
 - 表达式: =INT;
 - 表达式: =标志符;
- 中止符直接对应于符号
- 非中止符由中止符和其他非中止符构建成

语法分析器生成 II

- 矛盾——什么时候在同一时间运用两个不同的规则
- 如：如何对 `1+2+3` 进行语法分析



或者

?

- 可以通过使用优先指令来解决
- 可能会有更多的生成方法，他们可能更易读和易于使用

错误

- **恢复**——希望在单次执行中汇报尽可能多的错误
- **报告**——希望能以对使用者有用的方式汇报错误
 - 在代码中的准确位置
 - 详尽的信息
- 比起来课程 6.035 中，在现实世界中更重要…但是在调试编译器的过程中会发现很有用
- 在词法分析器中——存储关于词汇位置的相关信息并且存储以备后面部分使用
- 在语法分析器中——使用标记错误规则，取代破坏语法树块，这样可以保证语法分析继续进行
- 在一个特别部分失败后，允许放弃：如不要对词法错误进行语法分析

语用论——词法分析器

- 命令行: java JLex,Main file
- 使用 ant

```
<target name="scanner" depends="init">
  <java classname="JLex.Main"
    classpathref="project.class.path">
    <arg value="${src}/minimal.lex" />
  </java>
  <move file="${src}/minimal.lex.java"
    tofile="${genfiles}/Yylex.java" />
</target>
```

语用论——语法分析器

- 命令行: java java_cup.main < file
- 使用 ant

```
<target name="parser" depends="init">

  <java classname="java_cup.Main"
        classpathref="project.class.path"
        input="${src}/minimal.cup"/>

  <move todir="${genfiles}">
    <fileset dir=".">
      <include name="parser.java" />
      <include name="sym.java" />
    </fileset>
  </move>

</target>
```

参考

-尽早开始
-在组内对个人责任描绘清除
-使用资源控制 (cvs)
-使用构建系统 (ant 或者 make)
-对代码进行注释评注
-寻找乐趣!