第1章. 文法设计实验

1.1语言与语言规范

设计新的计算机程序设计语言是非常具有挑战性的一项工作，主要涉及语言规范的制定、语言的实现、语言的维护和升级等方面的内容。在这其中，语言的规范起到了非常重要的作用，如下图所示:

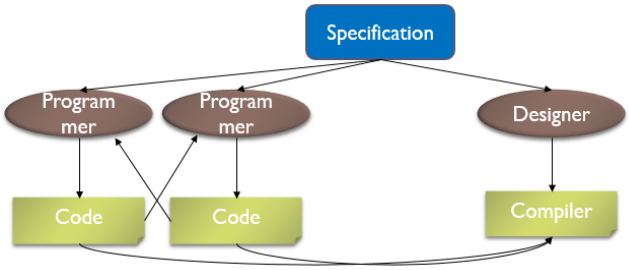


图 1 语言标准规范的重要性

语言标准规范是每个语言的说明文档，定义了语言的字符集、词法规则、语法规则和语义规则，也包括了对程序结构、编译过程、标准库程序以及语言实现等各方面的内容。一个标准规范描述为不同视角的参与者提供了一个一致的参考标准，例如程序员可以根据规范标准编写程序，系统设计者则可以根据这一标准规范设计编译器、完成运行时环境和库的构造和实现。由于编译器基于同一标准规范设计，因此才能对程序员编写完成的各式各样的程序进行不同层面验证，并将其最终翻译为目标代码。同理，所有的程序设计人员都基于同一个标准设计程序，按照给定的接口调用库函数，因此大家才能交换共享并理解对方的代码，并协同工作完成大规模软件项目的开发。

1.2C 语言标准规范的演化

生命力较强的程序设计语言的经历了长时间的发展演化，形成了阶段性的标准规范，以 C 语言为例，C 语言最初是贝尔实验室的 Dennis Ritchie 在 1972 年到 1973 年开发的运行在 Unix 上的工具，后来用于重写 Unix 操作系统。1978 年，Brian Kernighan 和 Dennis Ritchie 发布了《The C Programming Language》

一书，这本书也被广大 C 程序简称为 K&R，长期作为非正式的语言规范[1]。

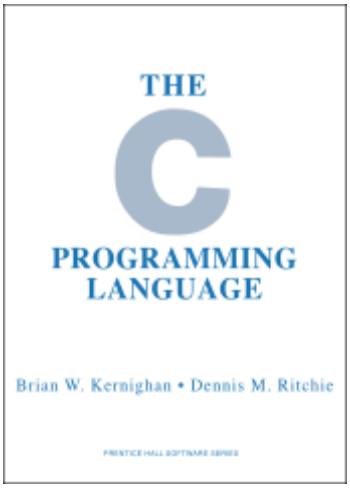


图 2　Brian Kernighan 和 Dennis Ritchie 的第一版 C 语言书[1]

1983 年，美国美国国家标准学会成了一个 X3J11 委员会，基于 Unix 系统中 C 语言的实现完成 C 语言的标准规范，但是 Unix C 库中不能移植的部分被交给了 IEEE 1003 工作组，该工作组在此基础上完成了 1988 年的 POSIX 标准。1989 年，ANSI X3.159-1989 “Programming Language C” 正式发布，这也意味着第一个 C 语言标准诞生，我们经常将其简称为 ANSI C 或者 C89。

1990 年，ANSI C 标准被国际标准化组织 ISO 接收成为 ISO/IEC 9899:1990，这也就是常说的 C90。所以 C89 和 C90 实际上是同一个 C 语言版本，只是被不同的标准组织认可和发布。

C 语言标准化后的几年中一直没有太大变化，直到 1995 年进行了一次规范修订，本次修订修改了一些细节并增加了国际字符集的支持。后来再一次修改是在 1999 年，ISO 发布了 ISO/IEC 9899:1999，通常我们将其称为 C99。C99 版本有了 inline 的函数，支持 long long int，我们经常用的单行注释//也是在这个版本中出现的。

2007 年开始了又一轮 C 语言标准规范的修订，可能是不知道什么时候能完成，刚开始的时候非正式版本称为 C1X，直到 2011 年 12 正式发布，才被大家称为 C11，这个版本中支持了 unicode，原子操作，多线程等内容，并改进了与 C++ 的兼容性。

2018 年发布的 C18 是目前最新版本的 C 语言标准规范，这个版本并没有增加新的内容，只是修正了 C11 中的一些缺陷。

1.3实验目的

本次实验的主要目的是了解程序设计语言的演化过程和相关标准的制定过程，深入理解与编译实现有关的形式语言理论，熟练掌握文法及其相关的概念，并能够使用文法对给定的语言进行描述，为后面的词法分析和语法分析做准备。

1.4实验内容

本次实验的内容为:

（1）阅读附件提供的 C 语言和 Java 语言的规范草稿，了解语言规范化定义应包括的具体内容。

（2）选定 C 语言子集，并使用 BNF 表示方法文法进行描述，要求至少包括表达式、赋值语句、分支语句和循环语句；或者设计一个新的程序设计语言，并使用文法对该语言的词法规则和文法规则进行描述。

以上语言定义首先要给出所使用的字母表，在此基础上使用 2 型文法描述语法规则。

设文法 $G=(V_N,V_T,S,P)$，对 P 中的每个产生式限制为形如:

$$A\to\alpha$$

其中，$A\in V_N$，$\alpha\in(V_T\cup V_N)^*$，则称文法 G 为 2 型文法。

设文法 $G=(V_N,V_T,S,P)$，对 P 中的每个产生式限制为形如:

$$A\to\alpha B \quad 或\ A\to\alpha$$

其中，A，$B\in V_N$，$\alpha\in V_T^*$，则称文法 G 为右线性文法。

设文法 $G=(V_N,V_T,S,P)$，对 P 中的每个产生式限制为形如:

$$A\to B\alpha \quad 或\ A\to\alpha$$

其中，A，$B\in V_N$，$\alpha\in V_T^*$，则称文法 G 为左线性文法。

左线性文法和右线性文法统称为 3 型文法，亦称为正则文法或线性文法。

1.5实验过程与方法

本次实验提供 C 语言和 Java 语言的规范草稿，请大家参考相关内容完成语言的设计和定义工作。此外，可以使用 ANTLRWorks 辅助完成语法的设计工作。

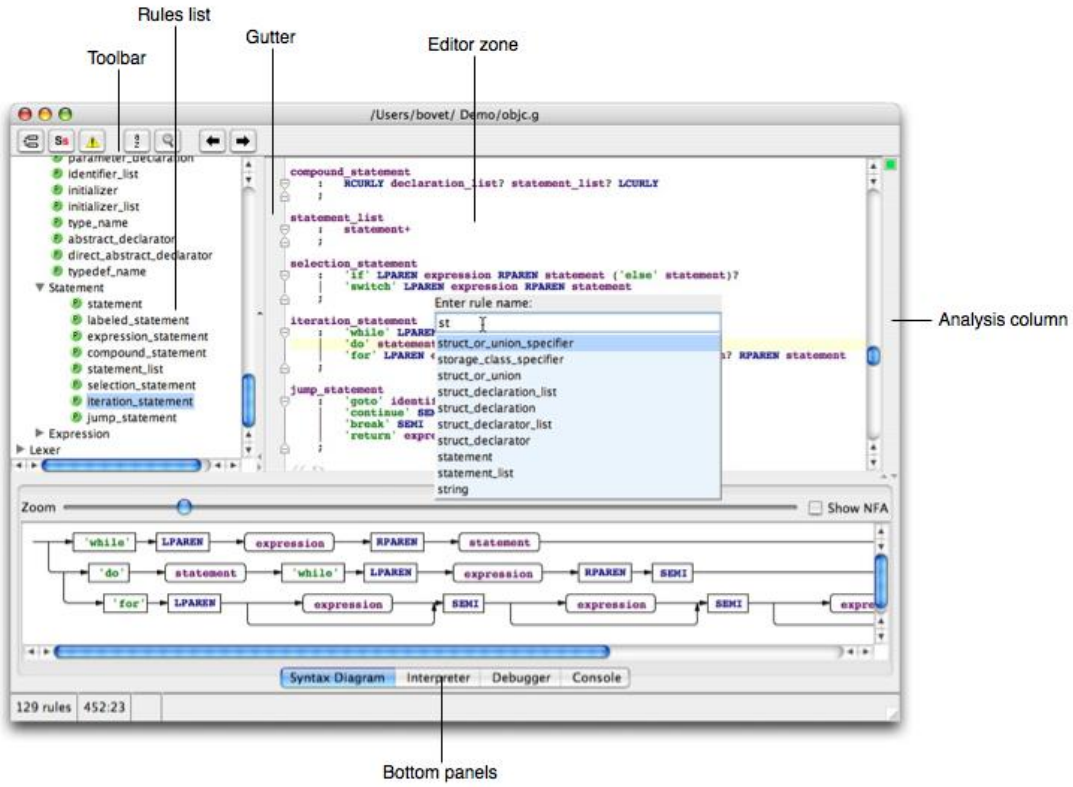
需要说明的是：ANTLR 使用的文法描述方法与教材上所采用的的文法描述方法不同，如有需要使用 ANTLR 相关工具，请自行 ANTLR 相关文档资料。

ANTLRWorks（http://www.antlr3.org/download/antlrworks-1.5.2-complete.jar）是由 Jean Bovet 等人基于 ANTLR 设计并实现的文法设计及检验工具，用户可以使用友好的用户界面编辑、查看、解释执行和调试文法。主要的功能包括：文法编辑及语法高亮显示、动态语法图显示、集成的文法解释器、集成的语言诊断调试器等。

ANTLRWorks 是使用 Java 语言编写的，因此可以在安装了 JRE 1.6 以上版本的多个系统平台上运行。目前该程序打包为一个 jar 包进行分发，因此可以按照如下命令运行：

$ java -jar antlrworks.jar

ANTLRWorks 为每个文法文件提供了一个编辑窗口，该窗口又进一步分为不同的区域，如下图所示：



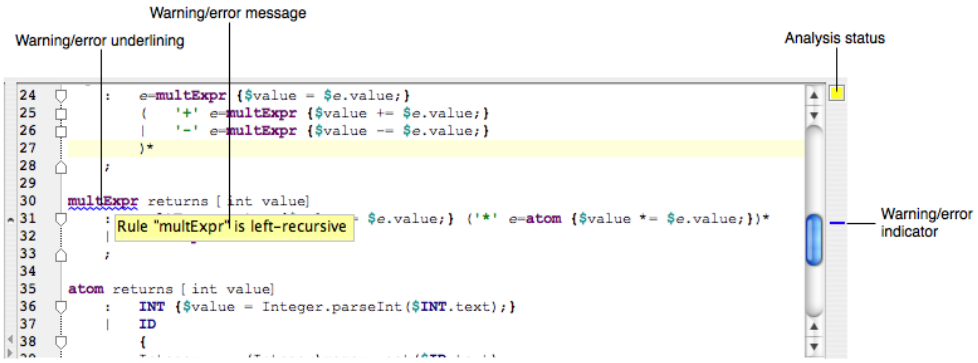
主要的部分包括:

编辑区域: 用户可以在该区域编辑文法，能够实时的对输入的文法进行高亮着色显示，并给出文法中的相关错误。在编辑区的左侧显示了规则对应的行号，规则限制以及调试断点信息;

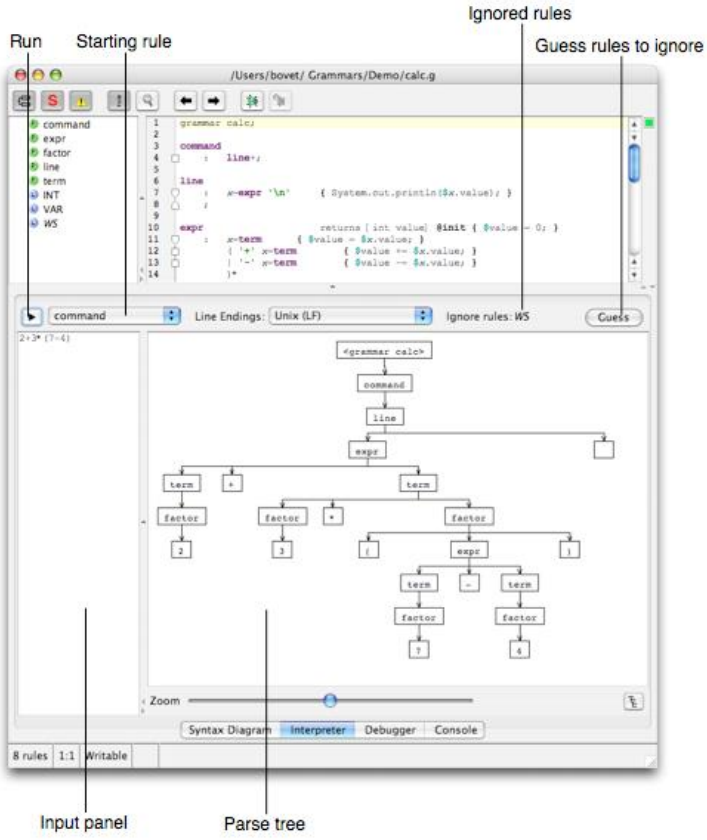
规则列表: 所有文法规则的列表区域，给出了所有词法和语法相关的规则，点击相关的规则可以跳转到该规则上，也可以对规则进行分组（右击）。

底部区域: 包括语法图、解释器、调试器、控制台等标签窗口。

ANTLRWorks 使用与 Eclipse 类似的方式显示关于文法的错误和警告信息，以左递归文法为例，在文法非终结符的底部以波浪线提示，鼠标放上去之后会给出具体的警告或者错误信息。



ANTLRWorks 最大好处在于可以使用集成的解释器立即对输入的文本进行分析，并不需要提前根据文法生成相应的词法分析器、语法分析器，因此对文法的设计与验证起到即时验证作用。如下所示，点击左下的 Interpreter，再输入框中输入测试代码文本，选择相应的开始产生式，点击运行，即可看到以分析树呈现的解析结果。



例如，在文本框里面输入下面的关于表达式的文法:

```
grammar Expr;

@header {
package test;
import java.util.HashMap;
}

@lexer::header {package test;}

@members {
/** Map variable name to Integer object holding value */
HashMap memory = new HashMap();
}

prog:   stat+ ;

stat:   expr NEWLINE {System.out.println($expr.value);}
    |   ID '=' expr NEWLINE
        {memory.put($ID.text, new Integer($expr.value));}
    |   NEWLINE
    ;

expr returns [int value]
    :   e=multExpr {$value = $e.value;}
        (   '+' e=multExpr {$value += $e.value;}
        |   '-' e=multExpr {$value -= $e.value;}
        )*
```

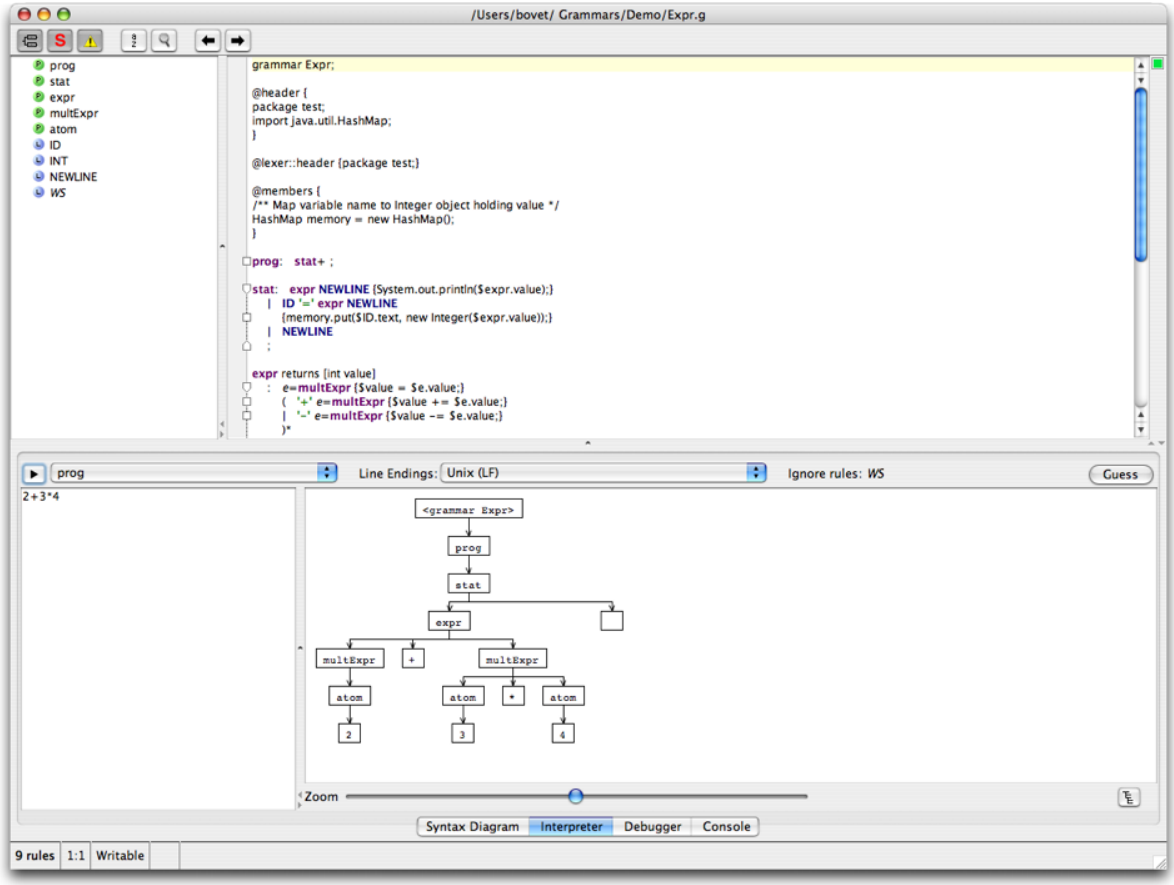
```
    ;

multExpr returns [int value]
    :   e=atom {$value = $e.value;} ('*' e=atom {$value *= $e.value;})*
    ;

atom returns [int value]
    :   INT {$value = Integer.parseInt($INT.text);}
    |   ID
        {
        Integer v = (Integer)memory.get($ID.text);
        if ( v!=null ) $value = v.intValue();
        else System.err.println("undefined variable "+$ID.text);
        }
    |   '(' e=expr ')' {$value = $e.value;}
    ;

ID  :   ('a'..'z'|'A'..'Z')+ ;
INT :   '0'..'9'+ ;
NEWLINE:'\r'? '\n' ;
WS  :   (' '|'\t')+ {skip();} ;
```

在输入文本框里面输入测试文本“2+3*4”，并选择开始符号“prog”并点击按钮▶运行解释器，就会看到相应的语法树：



需要注意的是：ANTLRWorks 输入文法时可以采用扩展的 BNF 表示方法，因此可以使用“[]”、“+”等多种扩展符号。**另外其表示方法与教材上的表示方法有所不同，如果需要使用该工具，则需要阅读其相关的文档。**

1.6实验提交内容

本次实验只提交实验报告。建议内容包括对语言规范的理解、选定的某语言文法设计报告和心得体会 3 部分内容。要求书写规范，阐述清楚。

1.7参考文献

[1]. https://en.wikipedia.org/wiki/C_(programming_language)