基于产生式的多语言程序理解的算法①

古 辉[1]，姚灵灵[1]，童李文[2]

[1](浙江工业大学 计算机科学与技术学院，杭州 310032)
[2](杭州市科技信息研究院，杭州 310001)

摘 要：提出一种程序理解实现方案，通过将程序设计语言的词法规则和语法规则以产生式表示，设计基于产生式多语言程序处理算法，将产生式自动转化为对应的词法和语法规则函数，使得不同的程序设计语言规则可以采用统一的方式描述，从而可用一套程序理解系统实现对多种程序设计语言的程序理解，以后增加某种程序设计语言的程序理解，只要定义对应程序设计语言的词法和语法规则产生式即可实现，有效地解决了程序理解系统的共享性难题。

关键词：程序理解；产生式；处理算法；规则函数

Production Rule-Based Multi-Language Program Comprehension of Algorithm

GU Hui[1], YAO Ling-Ling[1], TONG Li-Wei[2]

[1](College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310032, China)
[2](Hangzhou Science and Technology Information Research Institute, Hangzhou 310001, China)

Abstract: This paper presents a program comprehension scheme, it expresses the programming language lexical rule and grammar rule by the produce type, and designed the production rule-based multi-language program transform algorithm automatic convert it to lexical and grammar rule function. So the different programming languages can description by the same way, and then can use the same program comprehension system to achieve multi-language program comprehension.Add some programming language program comprehension,as long as defined correspondence programming language lexical and grammar rules produce type can be realized,effectively solve the problem of sharing program comprehension system.

Key words: program comprehension; production type; transform algorithm; rule function

程序理解是一个从计算机程序中获得该程序知识信息的过程，是软件工程领域的重要部分，广泛应用于软件维护和逆向工程方面[1,2]。

产生式是描述程序设计语言的一种有效方法[3]。它的表示形式简练严谨，便于刻画程序设计语言的本质，准确描述各种程序设计语言的复杂现象，并且容易转化为计算机语言。产生式是形如 A→B|C 书写规则，其中 A，B，C 都是符号串，它表示一个符号串可以被另一个符号串替换。A 称为产生式的左部，B|C 称为产生式的右部，“→”是表示替换的元符号，“|”是表示选择的符号。A、B 称为产生式的候选式。

目前，国内外的程序理解系统[4-7]，大多是面向特定的程序设计语言。为每种程序设计语言都开发一套程序理解系统将花费大量的人力和财力。本文通过研究不同程序设计语言的特性，提出一种基于词法和语法规则产生式的多语言程序理解实现方案。即用同一套程序理解系统，通过定义不同程序设计语言的词法和语法规则产生式实现多种程序设计语言的程序理解。

1 基于产生式实现多语言程序理解系统结构

不同的程序设计语言的区别主要体现在词法规则和语法规则的不同，而程序理解的方法是相似的。本文通过设计一种基于产生式的多语言程序理解处理器，

① 收稿时间:2011-05-24;收到修改稿时间:2011-06-20

将不同程序设计语言的词法规则和语法规则以统一的方式进行描述，统一了不同程序设计语言的词法和语法规则的差异，从而可用同一套程序理解系统实现多语言程序理解。

基于产生式实现多语言程序理解系统主要包括以下三个部分：基于产生式的多语言程序处理器、程序信息抽取模块和程序理解结果表示模块。基于产生式实现多语言程序理解系统结构如图 1 所示。

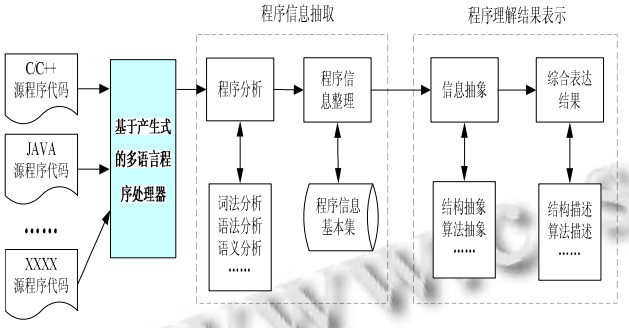

图 1　基于产生式实现多语言程序理解系统结构图

2　基于产生式的多语言程序处理器结构设计

基于产生式的多语言程序处理器主要由产生式库、基于产生式多语言程序处理模块和函数库三部分组成。将多种程序设计语言的词法规则和语法规则以产生式表示，并将其存入产生式库中。通过基于产生式多语言程序处理模块将多种程序设计语言产生式库中的词法规则和语法规则产生式转化为对应的词法规则和语法规则函数存入函数库中。函数库存储了对应的多种程序设计语言的词法规则函数和语法规则函数。通过调用函数库中的词法和语法分析函数即可实现词法和语法分析，基于产生式的多语言程序处理器结构如图 2 所示。

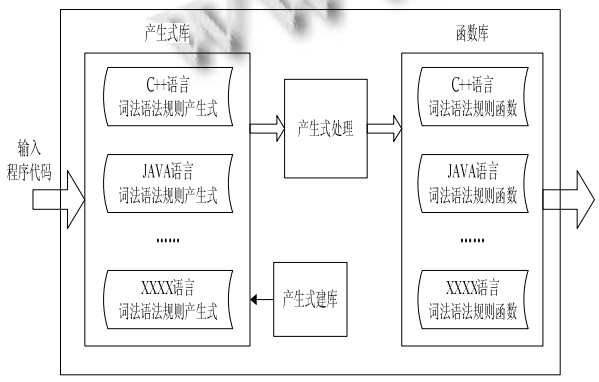

图 2　基于产生式的多语言程序处理器结构图

3　基于产生式的多语言程序处理器实现

基于产生式的多语言程序处理器是实现基于产生式的多语言程序理解系统的关键。产生式库存储了多种程序设计语言的词法规则和语法规则定义，它们以产生式表现形式存储在产生式库中。函数库存储了多种程序设计语言的词法规则函数和语法规则函数。词法规则函数和语法规则函数分别由产生式库中的词法规则和语法规则通过基于产生式多语言程序处理算法自动生成。通过调用对应语言的词法和语法规则函数即可实现对应语言的词法分析和语法分析，统一了不同程序设计语言词法和语法规则的差异，从而可用一套程序理解系统实现多语言程序理解。以后要增加某种程序设计语言的程序理解，只要在产生式库中定义对应语言的词法和语法规则产生式即可实现。

3.1 产生式库实现

3.1.1 词法规则产生式

词法分析是识别程序中的词，即识别标识符。Java 标识符是字母和数字的有限序列，它必须以字母、下划线或$开始，后面跟任意个上述三种符号或数字。根据以上定义，java 语言词法规则产生式表示举例如下：

<标识符>→<首字符><其他字符>*

<首字符>→A|...|Z|a|...|z|_|$

<其他字符>→A|...|Z|a|...|z|_|$|0.....|9

注：本文约定："<>"表示非终结符，"|"表示或，"*"表示重复 0 到多次。

3.1.2 语法规则产生式

符号串和层次结构是程序设计语言的重要特征，语法规则确定符号串和层次结构的组成方式，它规定程序设计语言的符号串和层次结构如何形成[3]。下面以 Java 语言为例，分别从表达式语法规则、语句语法规则、变量声明语法规则、方法语法规则和类相关语法规则这五个方面介绍语法规则的产生式表示。

(1) 表达式语法规则产生式

Java 表达式通常是由 java 运算符和括号将各 java 运算元素（例如常量、变量、方法、数组元素等）链接起来的一个有值的式子。表达式对应的产生式表示举例如下：

<表达式>→<表达式><双目操作符><表达式>

<表达式>→<单目操作符><表达式>

<表达式>→<数字>|<标识符号>

<双目操作符>→+|-|*|%|/|...

<单目操作符>→++|--|！|...

<数字>→0|1|2|3|4|5|6|7|8|9

(2) 语句语法规则产生式

在 Java 程序设计中的基本语句有：赋值语句，分支语句，循环语句，这几种语句对应的产生式表示举例如下：

① 赋值语句产生式表示：

<赋值语句>→<标识符>=<表达式>

② 分支语句中 if 语句产生式表示：

<if 语句>→if（<表达式>）<if 语句>|if（<表达式>）<if 语句>else<if 语句>

③ 循环语句中 while 语句产生式表示：

<循环语句>→while(<表达式>)<循环体>

<循环体>→<赋值语句>|<if 语句>|<循环语句>|<表达式>

③ 变量声明语法规则产生式

变量是 Java 程序的一个基本存储单元。在 Java 中，所有用到的变量必须先声明（或定义）后再使用。对变量的声明实际上就是给变量分配相应类型的存储空间。变量声明由变量类型后面跟着变量名组成。变量声明对应的产生式表示举例如下：

<变量声明>→<数据类型><标识符>

<数据类型>→int|short|long|float|double|char...

④ 方法语法规则产生式

方法定义了一组数据的操作，它包括方法的返回类型、方法名、方法参数列表和方法体的语句。其产生式表示举例如下：

<方法>→<返回类型><标识符>(<参数列表>){<方法体>}

<参数列表>→<数据类型><标识符>

<方法体>→<if 语句>|<循环语句>|<表达式>|<赋值语句>|...

⑤ 类相关语法规则产生式

类是 Java 的核心，是整个 java 语言的基本单元。类相关语法规则产生式举例如下：

<类>→<类说明修饰符>class<标识符><父类><接口>{<程序语句>}

<父类>→extends<标识符>

<接口>→interface<标识符>

<程序语句>→<成员变量>* <成员方法>*

3.1.3 需提取信息的规则产生式

程序设计语言在定义语法规则也同时确定和定义了语法规则所对应的每个词的语义。例如：<父类>→extends<标识符>，产生式中的<标识符>所包含的信息即为父类名。

进行程序理解过程中，需要对程序中所关注的特定语义的信息进行提取。为了对程序中的信息进行的有效提取，将所需提取特定语义的信息通过对产生式做对应的语义的标识，从而实现对程序中的信息提取。下面举例说明，需提取信息的规则产生式的特殊表示。

类产生式：<类>→<类说明修饰符>class<标识符><父类><接口>{<程序语句>}

<父类>→extends<标识符>

<接口>→interface<标识符>

为了抽取类名、父类、和接口信息我们将产生式做如下特殊表示：

<类>→<类说明修饰符>class<标识符：类名><父类><接口>{<程序语句>}

<父类>→extends<标识符：父类名>

<接口>→interface<标识符：接口名>

3.2 基于产生式多语言程序处理算法

基于产生式多语言程序处理算法将产生式库中的词法和语法规则产生式自动转化为词法和语法规则函数，通过调用对应的词法和语法函数即可实现对程序的词法和语法分析。算法分为如下 7 个步骤，算法流程图如图 3 所示。

① 提取产生式左侧信息作为函数名，函数参数为需要匹配字符串，函数返回类型为布尔型。

② 函数体为产生式右侧内容转化得到。

③ 产生式右侧如果包含终结符则将其转化为 if 条件判断，如果结果为假，则返回假。

④ 产生式右侧如果包括非终结符并且不需要提取信息，则其转化为该非终结符对应的函数进行 if 条件判断，如果为结果假，则返回假。

⑤ 产生式右侧如果包括非终结符并且需要提取信息，则其转化为该非终结符对应的函数进行 if 条件判断，如果为结果假，则返回假，否则提取非终结符的信息和“：”后面的信息提示，将提取信息保存。

⑥ 产生式右侧如果包括“|”则将所有以“|”连接的标识转化为一个 if 条件判断，之间用“或”进行连接，如果所有判断为假则返回假。

⑦ 产生式右侧如果包括“*”则将其转化为 while

语句，while 判断内容为*里面的内容转化得到，while 循环体内容为空。

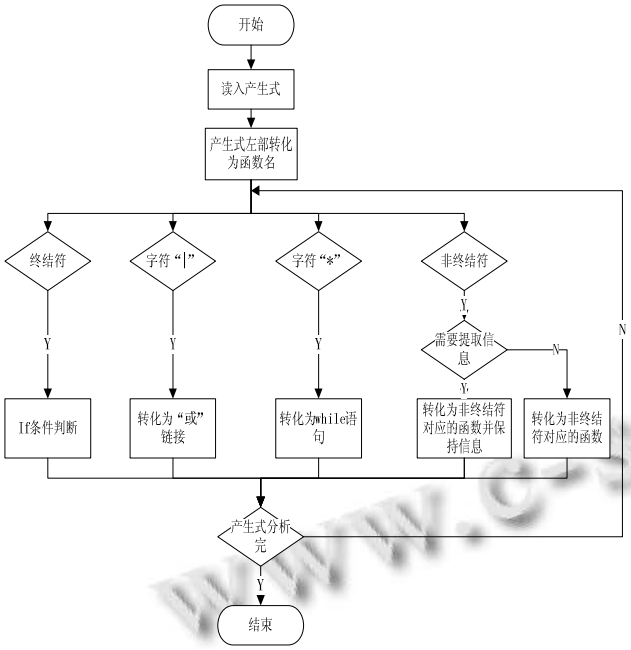

图 3　算法流程图

3.3 函数库实现

函数库由基于产生式多语言程序处理算法将产生式库中的词法规则产生式和语法规则产生式自动转化为对应的词法函数和语法函数存储在函数库中。通过调用函数库中的函数即可实现对源程序词法和语法分析，并对源程序中的信息进行提取。下面以 java 语言中类语法规则为例说明函数库中的函数表现形式。

产生式库中的类语法规则用产生式表示为：

<类>→<类说明修饰符>class<标识符：类名><父类><接口>

基于产生式多语言程序处理算法自动将产生式库中的类语法规则产生式转化为类语法规则函数存储在函数库中，其结果如下：

```
public static boolean  类(String text){
program = text;
if(类说明修饰符(program)){
}else{
program=text;
return false;
}
if(pipei(program,"class")){/pipei  函数功能为将第一个参数前部分和第二个参数匹配
}else{
program=text;
return false;
}
if(标识符(program)){
message+="类名：  "+information+"\n";
}else{
program=text;
return false;
}
if(父类(program)){
}else{
program=text;
return false;
}
if(接口(program)){
}else{
program=text;
return false;
}
return true;
}
```

3.4 基于产生式的多语言程序处理器处理实例

下面以现在比较主流的 java 和 C 两种程序设计语言为例，介绍基于产生式的多语言程序处理器处理结果。

3.4.1 对程序处理器输入如下 java 语言源码

```
public class boy extends people implements human {
public String name;
private String sex;
public int age;
public String BasicInformation() {
name = "LiMing";
sex = "male";
age = 22;
System.out.println("name:" + name + "sex:" + sex + "age:" + age);
}
public void interest(String  hobby) {
System.out.println("I love "+hobby);
```

```
}
}
```

基于产生式多语言程序处理器识别输入的程序源码种类为 java，调用 java 语言函数库中的词法函数和语法函数对其进行词法规则和语法规则分析，并对程序源码信息提取，输出结果如下：

类

类名：boy

父类：people

继承的接口：human

成员变量 变量名:name 类型:String 修饰符:public

变量名:sex 类型:String 修饰符:private

变量名:name 类型:int 修饰符:public

成员方法 方法名：BasicInformation 返回类型：String 修饰符：public 参数：空

方法体 赋值语句 标识符：name 值："LiMing"

赋值语句 标识符：sex 值："male"

赋值语句 标识符：age 值：22

调用方法 方法名 System.out.println 参数："name:" + name + "sex:" + sex + "age:" + age

成员方法 方法名：interest 返回类型：void 修饰符：public 参数：类型 string 名字 hobby

方法体 调用方法 方法名 System.out.println 参数："I love "+hobby

3.4.2 对程序处理器输入如下 C 语言源码

```
void abc()
{
int k,s=0;
for(k=1;k<10;k=k+2){
s=s+k;
}
}
mian(){
abc();
}
```

基于产生式多语言程序处理器识别输入的程序源码种类为 C 语言，调用 C 语言函数库中的词法函数和语法函数对其进行词法规则和语法规则分析，并对程序源码信息提取，输出结果如下：

函数

函数名：abc 返回类型：void 参数：空

函数体 函数变量 变量名： k 类型：int

变量名： s 类型：int

for 语句 循环初始值:k=1 循环条件:k<10 循环变化：k=k+2 循环体：s=s+k;

函数

函数名：mian 参数：空

函数体 调用函数名：abc 参数：空

通过基于产生式的多语言程序处理器，将输入的不同的程序设计语言源码进行词法和语法分析，并对程序中源码信息进行提取，提取出的信息通过程序信息抽取模块和程序理解结果表示模块将源程序文件以用户比较容易理解的方式将源程序文件信息展现。从而实现用一套程序理解系统实现对多种程序设计语言的程序理解。

4 结语

本文通过设计基于产生式的多语言处理器，将不同程序设计语言的词法和语法规则以统一的方式进行描述，从而使得可以用一套程序理解系统实现对多种程序设计语言的程序理解，有效地解决了程序理解系统的共享性难题。但本文设计的规则产生式尚不够完备，此外也没有考虑到消除递归，有待在今后的研究中进一步完善。

参考文献

1 Vinz BL, Etzkorn LH. Improving program comprehension by combining code understanding with comment understanding. Elsevier Science Publishers B. V. Knowledge-Based Systems, 2008,21(8):813–825.

2 Pich C, Nachmanson L, Robertson GG. Visual analysis of importance and grouping in software dependency graphs. ACM. Software Visualization,2008,Sep: 29–32.

3 郑洪.编译原理.北京:中国铁道出版社,2006.

4 Imagix–4D.Comprehensive program understandding tool for-C/C++.http://www.imagix.com/products/imagix4d.html.2010 March.

5 Source Insight. Program code editor and code browser .http://www.sourceinsight.com/.2010 April.

6 Source Navigator. A source code analysis tool. http://sourcenav.sourceforge.net/.2010 July.

7 Kscope. A source browsing and editing environment.http://kscope.sourceforge.net/. 2009 December.