Software Engineering SJTU 2012

LogoScript
补充规约(语言)

版本 **<Revision 1.0>**

修订历史记录

日期	版本	说明	作者
<7th ,March,2012>	Draft_1	第一版	顾城
<11th ,March,2012>	Revision 1.0	修改小错误。	顾城

目录

补充规约

1. 简介

1.1 目的

本补充规约（语言）旨在规范地定义 LogoScript 软件中所使用的脚本语言，用于补充前景文档以及用例建模中无法定义的语言规范。

1.2 范围

本补充规约（语言）定义的范围为 LogoScript 语言的所有语法特性和功能特性，包括表达式、语句定义，保留关键词，分支，循环，函数以及变量等，并对语言的可靠性、易用性和其他需求作了约定。

1.3 定义、首字母缩写词和缩略语

词汇	含义
function	函数
statement	语句
expression	表达式
expression$_n$	只含有不低于优先级为 n 的操作符的表达式
bin-op	二元操作符
bin-op$_n$	优先级别为 n 的二元操作符
factor	表达式元
token	词法单元
identifier	标识符
->	可被定义为
宿主	搭载 LogoScript 语言的程序，用来执行 LogoScript 语言编写的程序完成一定的工作

约定中使用蓝色粗斜体表示保留关键字，黑色粗斜体表示语言中的非终结符号或终结符号。
对于既可以作为非终结符号又可以以保留关键字作为终结符号的词法单元，蓝色粗斜体表示以保留关键字作为终结符号使用。
其他词汇定义请参见词汇表。

1.4 参考资料

可用的参考资料有：
PC Logo 语言语法手册：记录着 Logo 语言的语法规范和函数约定。
主流编程语言语法手册：记录着其他主流编程语言的语法规范。

1.5 概述

本文档首先介绍语言的功能性需求，包括语法定义，功能特性，然后再介绍语言的可靠性要求，易用性要求以及性能要求。

2. 功能

2.1 基本功能

LogoScript 语言被宿主搭载，宿主可将自身的函数通过一定的接口导入 LogoScript 语言的运行环境中。按照以下词法和语法定义所正确编写的 LogoScript 程序可以被编译，并在宿主所提供的语言环境中执行，如果运行时发生任何错误，LogoScript 语言虚拟机将汇报宿主。任何与词法、语法相冲突的程序将不能被编译，编译过程遇到的任何错误将通知宿主程序。

2.2 词法定义

2.2.1 词法约定

LogoScript 是区分大小写的语言，大小写不同但是字符相同的两个名称将被视为不同。空格在语言中仅用于分割两个标识符或关键字使用，多余的空格字符将被忽略。

2.2.2 保留关键字

以下为 LogoScript 的 8 个保留关键字:

```
function local if else
while break continue return
```

以下为其他使用的关键字：

```
= == < > >= <= != `
+ - * / && || ! ; ,
```

2.2.3 注释

任何以`开头的行将被视为程序注释，最终编译时整行被编译器忽略。

2.2.4 常量

LogoScript 语言仅支持数字常量，任何只包含数字 0-9 以及最多一个小数点(.)的词法单元将被视为数字。

以下是合法的数字常量：

123 45.6 78. .80 045

以下是非法的数字常量：

1.2.3 a.7 6.b 0x123

2.2.5 标识符

任何仅包含_,小写字母，大写字母以及数字 0-9（第一个字符不是数字）的词法单元将被视作合法的标识符。标识符不能为保留关键字。

以下是合法的标识符：

Test my_logo LogoScript1 _ ifResult T_T

以下不是合法的标识符：

@_@ 0Logo d-3 f%@# ... break

2.3 语法定义

2.3.1 类型

LogoScript 语言提供以下几种类型：

数值（双精度浮点类型）
函数
其他由宿主程序定义的类型

除非宿主提供相应函数，否则不同类型的变量不能互相转换。任何比较和表达式运算只能在所有参与类型都为数值型的情况下进行，有函数类型参与的运算或者试图调用一个数值都会导致运行时错误。

2.3.2 变量及作用域

LogoScript 语言支持全局变量和局部变量。全局变量的作用域为整个程序，局部变量的作用域为当前函数。如果当前函数环境存在同名的局部变量和全局变量，局部变量优先。

2.3.2.1 全局变量

LogoScript 语言中的全局变量不需要定义，任何非局部变量都被视作全局变量。在全局变量作为右值之前，必须至少被赋值过一次。未初始化过的全局变量作为右值会产生一个运行时错误。

2.3.2.2 局部变量

在函数中可以使用保留关键字 ***local*** 定义局部变量，格式为
local 标识符 1[,标识符 2,...,标识符 n];
在同一函数中可以有任意多条 ***local*** 语句，但 ***local*** 语句所定义过的局部变量仅在定义之后的语句生效。重复的局部变量将被忽略。局部变量的初始值是不确定的，因此推荐初始化局部变量。

2.3.3 函数

2.3.3.1 函数定义

LogoScript 语言的函数定义如下：
function 标识符([形参 1],[形参 2],...,[形参 n]) 语句
其中的语句即为函数体，函数体内不能有函数定义。函数体可以为单条语句，也可以为复合语句（见 2.3.5.2）。任何函数只能返回一个值。
形参可以为任何合法的标识符，任何形参在函数体中都算本函数的局部变量，函数可以任意使用或修改这些值。

2.3.3.2 函数调用

LogoScript 语言的函数调用为：
函数标识符([实参 1],[实参 2],...,[实参 n])

其中函数标识符必须为存在的为函数类型的全局变量或局部变量。如果该变量不存在或者不为函数类型则会产生运行时错误。函数允许调用自身。
实参可以是任何表达式，包括变量，其他函数调用。多余的实参将被忽略；如果实参个数少于函数定义的形参个数，未被指定的形参将被设置为 0。

2.3.4 表达式

表达式是可以被求值的式子，LogoScript 语言中的表达式由表达式元，二元操作符组成。

2.3.4.1 表达式元

LogoScript 语言中的表达式元可以是：

- 数值
- 数值型变量
- 函数调用
- 一元操作符!和任意表达式元
- 被括号(和)括起来的任何表达式

表达式元将被以最高优先级求值。

2.3.4.2 二元操作符以及优先级

LogoScript 语言的所有二元操作符如下：

操作符	说明	优先级（数字越大表示优先级越低）
* /	算术乘除	1
+ -	算术加减	2
> < == != >= <=	大于 小于 等于 不等于 大于等于 小于等于	3
&&	逻辑与	4
||	逻辑或	5

所有的运算符都是左结合的，逻辑运算表达式如果结果为真其结果值为 1，如果为假，其值为 0。但是，任何非 0 值在逻辑表达式中都被认为是真。
在含有多优先级的表达式中，优先级较高的子表达式将首先被计算。

2.3.4.3 表达式的递归定义
LogoScript 语言中的表达式定义如下
expression -> expression bin-op expression | factor
由此可见，表达式的定义是递归的，两个表达式被二元操作符连接，可以产生一个新的表达式。表达式元也可直接构成表达式。
此产生式暂时无法体现表达式优先级，如要正确体现优先级，可以参见后文消除左递归的完整产生式。

2.3.5 语句
和表达式不同，LogoScript 语言的语句定义了程序行为，语句是不能被求值的式子。任何语句都必须在某个函数体内。

2.3.5.1 表达式作为语句
任何表达式加上;即可成为一条语句，表达式将被计算，但是计算结果会被丢弃。

2.3.5.2 复合语句
在花括号{}内的一条或者多条语句将被视为一条复合语句。因为函数定义中的函数体，分支，循环体均定义为一条语句，因此如果需要在函数体、分支或循环体内执行多条语句，需要使用{}将其转化为一条复合语句。复合语句可以嵌套。

2.3.5.3 赋值语句
LogoScript 语言中的赋值语句定义为：
标识符 = 表达式;
其中标识符必须合法，表达式将首先被求值，然后标识符所对应的变量的值将被设置。注意 LogoScript 语言中的赋值为语句而非表达式，因此不支持 a=b=c 这种赋值方式。

2.3.5.4 分支语句
LogoScript 语言中的分支语句为：
if(表达式) 表达式为真时执行的语句 [*else* 表达式为假执行的语句]
表达式将首先被求值，如果表达式求值为真，将只执行表达式为真时执行的语句，如果表达式求值为假并且对应 *else* 块存在，则只执行表达式为假执行的语句，否则不执行任何语句。
else 块是可选的，任何 *else* 块将与最近的 *if* 语句匹配。

2.3.5.5 循环语句
LogoScript 语言循环语句为：
while(表达式) 循环体语句
表达式将被首先求值，在求值为真的情况下，循环体将被执行，并在此重复求值-执行循环体的操作，直到表达式求值为假，循环体将不被执行，本语句结束。

2.3.5.5.1 ***continue*** 语句

在循环体中，可以使用 ***continue*** 语句，此时循环体之后的语句在本次循环内将不被执行，直接进行一下次表达式求值。***continue*** 语句将与最近的 ***while*** 匹配。

2.3.5.5.2 ***break*** 语句

在循环体中，可以使用 ***break*** 语句终止循环，此时将直接执行循环语句后的一条语句。***break*** 语句将与最近的 ***while*** 语句匹配。

2.3.5.6 返回语句

在任何函数体中，可以使用 ***return*** 语句设定本函数的返回值，并终止函数的执行，将控制权移交给调用当前函数的函数。其格式为：

return 表达式;

2.3.6 完整的上下文无关语法定义

function -> ***function*** identifier(parmopt) statement
parmopt -> ε | parms
parms -> identifier | parms,identifier
function_call -> identifier(argsopt)
argsopt -> ε | args
args -> expression | args,expression
expression -> $expression_5$
$expression_n$ -> $expression_{n-1}$ $bin\text{-}op_n$ $expression_{n-1}$ (n>0) | $expression_{n-1}$
$expression_1$ -> factor $bin\text{-}op_1$ factor | factor
factor -> number | function_call | (expression) | !factor
statement -> expression; | identifier = expression; | ***if***(expression) statement [***else*** statement] | ***while*** (expression) statement | ***continue***; | ***break***; | {stmts} | ***local*** params; | ***return*** expression;
stmts -> ε | stmts statement

2.3.7 示例程序

以下示例程序将使用 2 种方法输出 5 的阶乘。其中 print 为宿主提供的函数用于输出变量的值。

```
function fact(n)
{
	if(n>1)
		return n*fact(n-1);
	else
		return 1;
}

function fact2(n)
{
	local i,r;
	i = 0;
	r = 1;
	while(1)
	{
		i = i + 1;
		r = r * i;
```

```
            if (i == n) break;
        }
        return r;
    }

    function main()
    {
        print(fact(5));
        i = fact2(5);
        print(i);
    }
```

2.4 其他约定

程序执行时将从 main 函数开始执行，因此，main 全局变量必须存在，并且为函数类型。一般情况下，均需要用户定义 main 函数，如果 main 函数返回，则终止程序的执行。

3. 可用性

LogoScript 不包含复杂的指针、类、对象、模板等概念，因此可以轻易学习。对于没有编程基础的人，学习 LogoScript 语言将比较容易，同时由于 LogoScript 语言语法与传统 C、C++、JavaScript 等较为相似，任何有编程基础的人均可轻易掌握语言的基本语法。

4. 可靠性

LogoScript 语言将是一种可靠的语言。任何因为用户编写的 LogoScript 程序导致的错误，包括语法错误和运行时错误，将不会导致宿主崩溃或者发生错误。
在 99.99%或以上的情况中，LogoScript 编译系统将不会产生错误的中间代码，LogoScript 虚拟机将不会错误地解释任何中间代码。即使发生任何错误，也不会导致宿主崩溃。

5. 性能

在 2012 年主流的计算机上，LogoScript 编译每千行代码(平均每行一条语句)的时间将小于 300 毫秒。

LogoScript 语言的执行速度达到如今主流脚本语言同一数量级的执行效率。

6. 设计约束

LogoScript 语言的宿主为 C#编写，因此 LogoScript 编译器以及虚拟机也将由 C#编写。

7. 联机用户文档和帮助系统需求

LogoScript 语言的规范和示例程序集将随宿主程序而发行。同时，该内容也可通过访问产品的网站联机获得。

8. 接口

LogoScript 语言将为宿主提供以下接口：

编译、执行脚本。

将宿主函数导入语言运行环境。

设置断点、单步调试、跳出。

编译错误、运行时错误反馈。