FORCEFIELD语言说明

云计算脚本语言

王远轩 肖梦华

2010.8

FORCEFIELD 语言说明

云计算脚本语言

王远轩 肖梦华

2010.08

语言简介

我们的参赛作品被命名为ForceField语言。它是一种应用于云计算平台的轻量级脚本语言。它有数字、字符串、字典及函数四种对象，支持基本的循环、赋值、函数调用等基本功能，同时支持远程调用、分层作用域、函数递归等语义。此外，我们为ForceField语言实现了一套在线调试系统，方便开发者调试自己的程序。

基本语法

根据比赛要求，ForceField语言使用了一系列大写字符作为它的关键字。由于比赛页面中已经给定了SET、RETURN、IF等关键字的用法，这里不再赘述。接下来我们将分别介绍比赛页面中没有涉及到的内容，其中也包括了我们自己设计的一些语法。

字典类型

比赛页面中提到了HTTP这一内置字典，为了保持语法的一致性，我们引入了字典这一类型。ForceField的字典类型和常见语言的字典类型几乎一样。它的声明方式也很简单，我们并没有提供初始化某一个字典的语法，需要创建一个字典对象时只需直接赋值即可。例如，我们要声明一个名为foo的字典对象，同时将它的bar键置为hello：

```
SET foo[“bar”] = hello
```

这条命令会先在可访问的作用域中查找名为foo的变量，如果已存在foo字典则直接修改bar键，否则先创建foo字典，再修改，如果foo存在但它的类型不是字典则报错。

远程函数调用

当ForceField程序员需要调用一个远程脚本时，他需要修改两个地方：在server.conf中声明远程服务器脚本的URL，例如

```
script1, http://127.0.0.1:12345/add
```

然后在本地脚本中声明调用这个脚本需要的参数，以ADDNUM为例：

```
REMOTE @script1 ADDNUM(x, y)
```

这样就能调用远程的ADDNUM脚本了。

变量作用域

类似Python，在ForceField中只有函数调用才会创建新的作用域，同时能访问外层变量）。

但和Python不同的是，Python中函数对外层作用域的访问是只读的，例如这个程序是非法的：

```
def f():
    x += 1

x = 0
f()
```

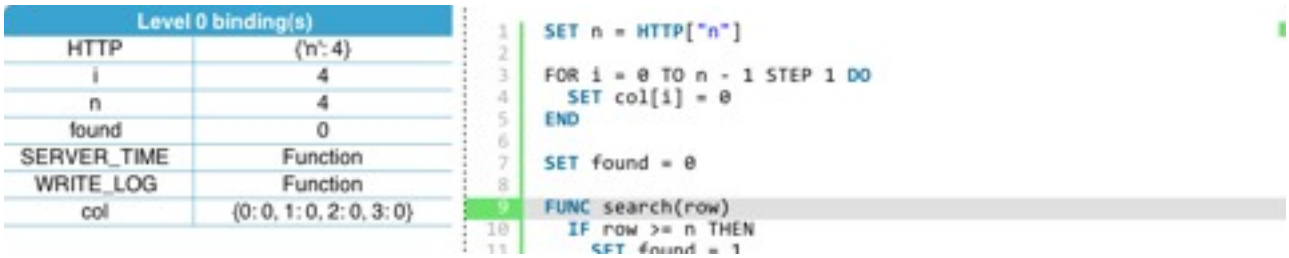
在ForceField语言中，拥有类似语义的这个程序则是被允许的：

```
FUNC F()
     SET x = x + 1
     RETURN x
END

SET x = 0
CALL F()
RETURN x
```

最终这个脚本会返回1。

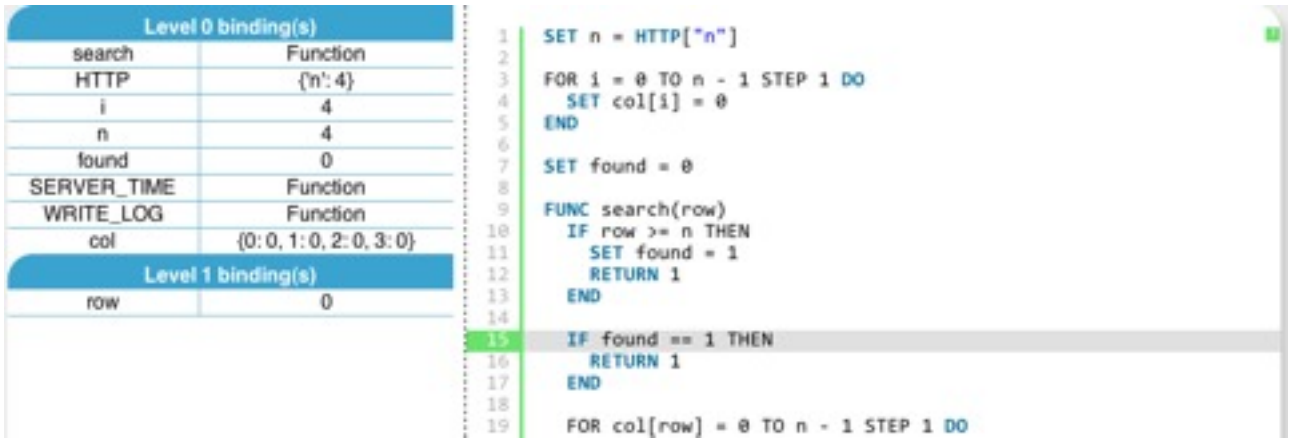
为了更好的说明作用域的机制，我们接下来将通过深度优先搜索解八皇后程序的运行(queen.input)为例，来说明ForceField的作用域的设定。（所有图片均截图自ForceField在线调试工具）



Level 0 binding(s)	
HTTP	{'n': 4}
i	4
n	4
found	0
SERVER_TIME	Function
WRITE_LOG	Function
col	{0: 0, 1: 0, 2: 0, 3: 0}

```
1  SET n = HTTP["n"]
2
3  FOR i = 0 TO n - 1 STEP 1 DO
4    SET col[i] = 0
5  END
6
7  SET found = 0
8
9  FUNC search(row)
10   IF row >= n THEN
11     SET found = 1
```

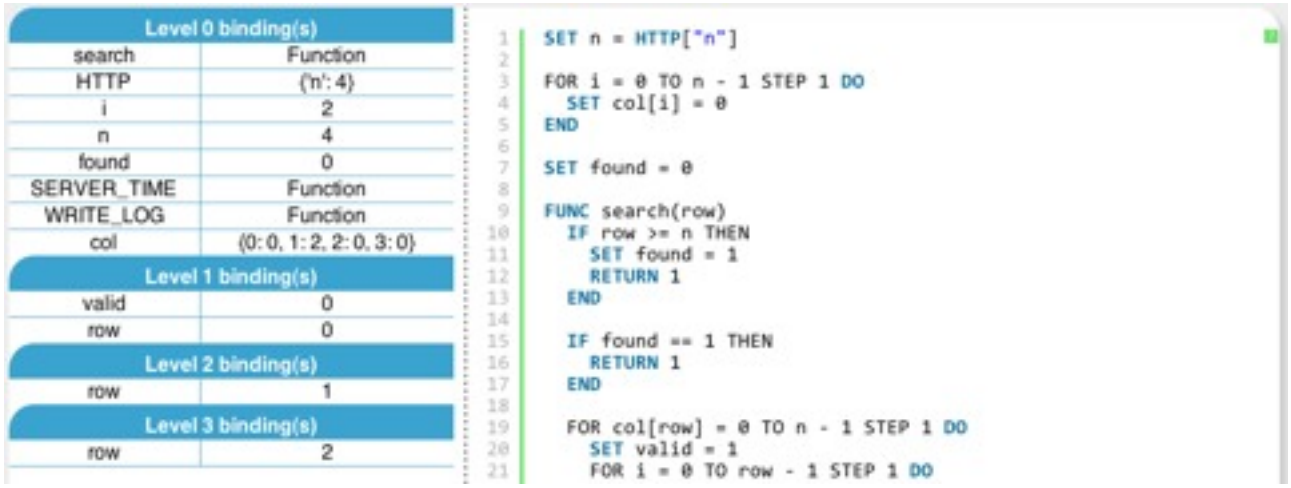
执行到第9行时，最外层作用域有HTTP、SERVER_TIME和WRITE_LOG三个默认变量，以及程序运行到现在产生的变量。当第9行程序执行完成后，最外层作用域会增加一个search函数。



Level 0 binding(s)	
search	Function
HTTP	{'n': 4}
i	4
n	4
found	0
SERVER_TIME	Function
WRITE_LOG	Function
col	{0: 0, 1: 0, 2: 0, 3: 0}
Level 1 binding(s)	
row	0

```
1  SET n = HTTP["n"]
2
3  FOR i = 0 TO n - 1 STEP 1 DO
4    SET col[i] = 0
5  END
6
7  SET found = 0
8
9  FUNC search(row)
10   IF row >= n THEN
11     SET found = 1
12     RETURN 1
13   END
14
15   IF found == 1 THEN
16     RETURN 1
17   END
18
19   FOR col[row] = 0 TO n - 1 STEP 1 DO
```

此处为调用search(0)后的结果，参数row被置为0，且属于第一层作用域。另外可以看到最内层作用域已经有一个search变量，且类型为Function。



Level 0 binding(s)	
search	Function
HTTP	{'n': 4}
i	2
n	4
found	0
SERVER_TIME	Function
WRITE_LOG	Function
col	{0: 0, 1: 2, 2: 0, 3: 0}
Level 1 binding(s)	
valid	0
row	0
Level 2 binding(s)	
row	1
Level 3 binding(s)	
row	2

```
1  SET n = HTTP["n"]
2
3  FOR i = 0 TO n - 1 STEP 1 DO
4    SET col[i] = 0
5  END
6
7  SET found = 0
8
9  FUNC search(row)
10   IF row >= n THEN
11     SET found = 1
12     RETURN 1
13   END
14
15   IF found == 1 THEN
16     RETURN 1
17   END
18
19   FOR col[row] = 0 TO n - 1 STEP 1 DO
20     SET valid = 1
21     FOR i = 0 TO row - 1 STEP 1 DO
```

此处为search递归过程中的作用域情况，可以看到一、二、三三层作用域中均包含row变量，此时读取row时优先使用最外层（及第三层）的作用域，值为3。

语言实现

我们使用了ANTLR (http://www.antlr.org/)开源框架对ForceField脚本进行词法、语法分析。由于我们旨在实现一个脚本语言的原型，并不注重它的运行性能，因此略去了编译后端的代码生成、优化等阶段，而是直接使用Python解释执行由ANTLR生成的抽象语法树。ForceField脚本的词法、语法分析主要包含在Expr.g文件中，而解释执行则和Eval.g，environtment.py等多个文件有关。

在解释执行时，我们采用了惰性求值技术简化了解释器的实现。在environment.py中我们定义了Stmt、Expr、Function、RemoteCall四种基本语言元素，它们的特点都是在生成的时候并不执行任何实际代码，而是在需要执行的时候才会通过eval/call等方法解释执行，和SICP 3.5章中提到的force函数比较相似。

测试脚本

为了保证新增的特性不影响之前脚本的正确运行，我们在scripts目录下维护了一组回归测试脚本用例，所有以_test.input结尾的脚本都会在回归测试中覆盖。同时为了方便自动化测试，我们加入了ASSERT关键字，这个关键字可以穿插在程序中，保证某一时刻程序的中间变量的正确性。

回归测试覆盖了基本运算、字典操作、递归函数、远程调用以及其他一些语言要素，在ForceField目录下运行python regression.py即可进行自动化的回归测试。