Accassias

User Manual

Eric Therond

Sommaire

Chapitre 1

Introduction

1.1 Présentation d'Accassias

1.1.1 Description générale

Accassias est un outil pour l'aide à l'analyse statique de codes sources. Il est écrit dans un but d'apprentissage, à la fois l'étude de la programmation en *C++* et la recherche automatisée de défauts logiciel.

Chapitre 2

Utilisation

On execute ici le programme example1.aca écrit en langage *Accassias*

```
aca$
eric@linux-r8as:       /accassias/src> cat ../tests/example1.aca
declr $a;
$a = 6;

if($a > 3)
{
  $stdlib->print(5555);
}
else
{
  $stdlib->print(6666);
}

$stdlib->print(1000);
```

```
eric@linux-r8as:~/     /accassias/src> ./accassias ../tests/example1.aca



5555
1000
aca$
```

2.1 Calcul de l'arbre de syntaxe abstraite

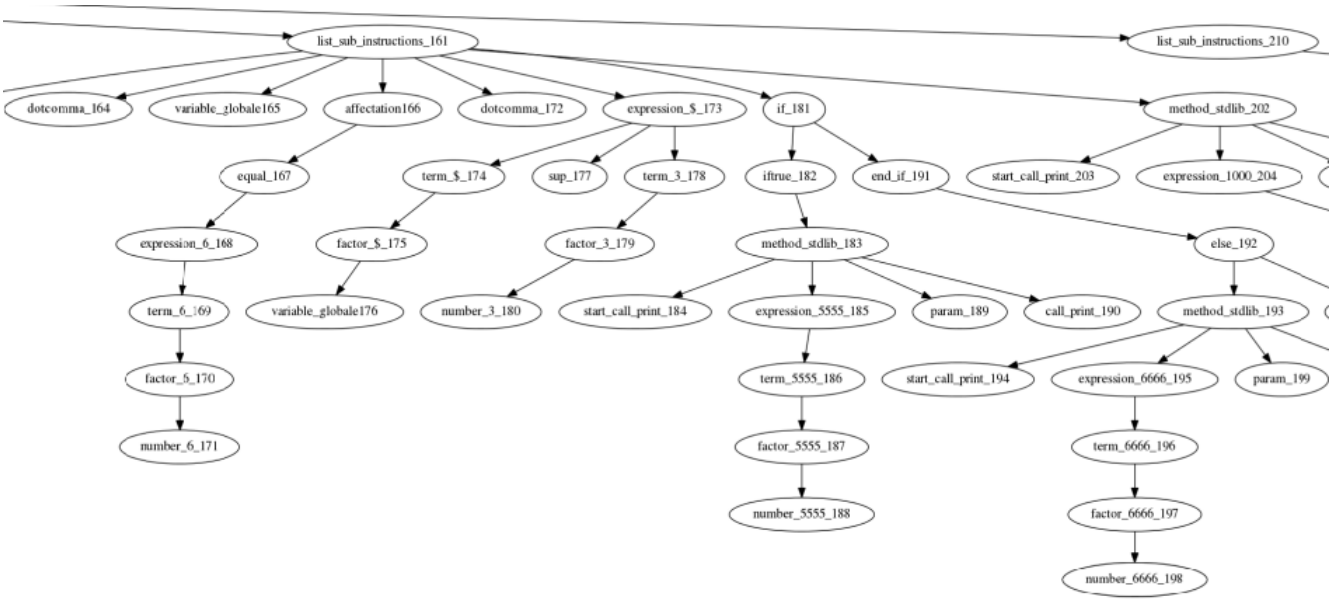
On lance l'écriture de l'ast au format dot à l'aide de l'appel à la méthode

```
$ast->dot("dotexample1.dot");
```

```
eric@linux-r8as:~/     /accassias/src> ./accassias ../tests/example1.aca



5555
1000
aca$ $ast->dot("dotexample1.aca");
abstract syntax tree has been printed into dotexample1.aca
aca$
```



2.2 Calcul du three address code

On affiche le three address code à l'aide de l'appel à la méthode

```
$stdlib->system("show_tac");
```

```
aca$ $stdlib->system("show_tac");
 : THREEADDRESSCODE : three address code here

jump 0
instructions
subinstructions 8
temp0 ;
instance class 0 2
temp1 ;
temp2 ;
temp0 = temp0
temp1 ;
method poly
start call
temp3 = temp4
temp2 = temp3
temp1 = temp2
subinstructions 6
temp5 ;
instance class 1 0
temp5 = temp1
subinstructions 2
temp6 ;
instance class 2 0
temp6 = temp2
subinstructions 2
temp7 ;
instance class 3 0
temp7 = temp3
subinstructions 1
temp8 ;
instance class 4 0
temp8 = temp4
subinstructions 12
```

2.3 Calcul du control flow graph

On calcul le control flow graph à l'aide de l'appel à la méthode

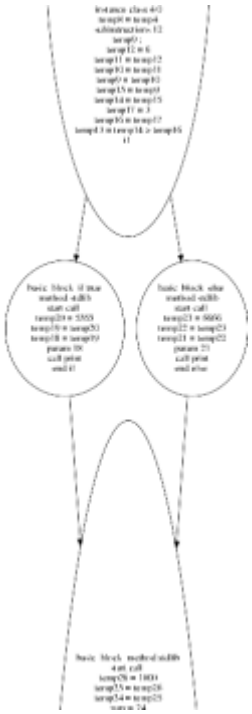
```
$cfg->compute();
```

```
aca$ $cfg->compute();
```

On écrit le control flow graph au format dot à l'aide de l'appel à la méthode

```
$cfg->dot("cfgexample1.dot");
```

```
aca$ $cfg->dot("cfgexample1.dot");
```



2.4 Calcul du code machine

On affiche le code machine à l'aide de l'appel à la méthode

`$stdlib->system("show_code");`

```
 : CODE : code here

0    JUMP 2
2    INSTRUCTIONS
3    PILE 8
5    INSTANCE_CLASS 0 2
8    EMPL 0
10    DEPL 0
12    PILE 1
14    EMPL 4
16    DEPL 3
18    EMPL 3
20    DEPL 2
22    EMPL 2
24    DEPL 1
26    INSTRUCTIONS
27    PILE 6
29    INSTANCE_CLASS 1 0
32    EMPL 1
34    DEPL 5
36    INSTRUCTIONS
37    PILE 2
39    INSTANCE_CLASS 2 0
42    EMPL 2
44    DEPL 6
46    INSTRUCTIONS
47    PILE 2
49    INSTANCE_CLASS 3 0
52    EMPL 3
54    DEPL 7
56    INSTRUCTIONS
57    PILE 1
59    INSTANCE_CLASS 4 0
```

2.5 Affichage de l'état de la machine virtuelle

On affiche l'état de la machine virtuelle à l'aide de l'appel à la méthode

`$stdlib->system("show_statevm");`

```
 : VM : state and stack here

|        co |       bel |        sp | instruction                 | stack
|       410 |     20057 |     20057 | SYSTEM -3                   | 0 20001 327 7 0 20001 292 6 20001 20001 260 5 20001 20001 229 4 20001 20001 7 7 7 6
|       412 |     20057 |     20057 | RETURN 1                    | 0 20001 327 7 0 20001 292 6 20001 20001 260 5 20001 20001 229 4 20001 20001 7 7 7 6
|       327 |     20001 |     20055 | PILE -1                     | 327 7 20001 20001 292 6 20001 20001 260 5 20001 20001 229 4 20001 20001 7 7 7 6 6 6
|       329 |     20001 |     20054 | EXIT                        | 7 20001 20001 292 6 20001 20001 260 5 20001 20001 229 4 20001 20001 7 7 7 6 6 6 5
aca$
```