SRA

http://abs-models.org/sra-tutorial/

Abel García
Elena Giachino

ENVISAGE
http://www.envisage-project.eu/

Contents

1 General Overview

We prototype a static analysis technique that computes upper bounds of virtual machine usages in a dialect of ABS, called vml, whose syntax will be covered by the examples in this tutorial, with explicit acquire and release operations of virtual machines. In our language it is possible to delegate other (ad-hoc or third party) concurrent code to release virtual machines (by passing them as arguments of invocations). Our technique is modular and consists of (i) a type system associating programs with behavioural types that records relevant information for resource usage (creations, releases, and concurrent operations), (ii) a translation function that takes behavioural types and return cost equations. It is integrated with the solver CoFloCo that given the cost equations produces the result.

2 Resource Analysis

In this section we present how to compute the cost of a vml program in term of virtual machine usage.

First, select "Resource Analysis (SRA)" from the pull-down menu at the top of the window on the center-left. The parameters of the selected analysis are automatically set, so there is nothing to be configured in the Settings section in the top-left corner.

As an example, open the program doubleRelease.vml:

```
1
2  Int doubleRelease(VM x, VM y) {
3          release x; release y;
4          return 0 ;
5  }
6
7  Int user1() {
8          VM x ; VM y ; Fut<Int> f ;
9          x = new VM() ; y = new VM();
10         f = this!doubleRelease(x, y);
11         Int a = f.get ; return 0 ;
12 }
13
14 {
15   Fut<Int> fuser1 = this!user1();
16   Int a = fuser1.get;
17 }
```

Let us analyze the program. Click on Apply to perform the analysis.

The output of the analysis is shown in three tabs of the console, which are generated by the tool:

Types contains the behavioural types generated for the input program

Equations contains the cost equations resulted by the translation of the behavioural types

UBs which is the overall output and shows the upper bounds. For the doubleRealese.vml we get:

```
Partitioned cost of main(MAINVM):
2
 if []
Partitioned cost of doubleRelease012net(THISVM,X,Y):
-2
 if [THISVM=1,X=1,Y=1]
-1
 if [THISVM=1,X=1,Y>=2]
 or [THISVM=1,Y=1,X>=2]
0
 if [THISVM=1,X>=2,Y>=2]
 or [THISVM=3]
Partitioned cost of user10net(THISVM):
```

```
0
 if [3>=THISVM,THISVM>=1]
Partitioned cost of doubleRelease012peak(THISVM,X,Y):
0
 if [3>=THISVM]
Partitioned cost of user10peak(THISVM):
0
 if [THISVM=3]
2
 if [2>=THISVM]
Time statistics:
Total analysis performed in 208 ms.
```