Upper Bounds of Resource Usage for Java Bytecode using COSTA and its Web Interface

(Extended Abstract)

E. Albert1, P. Arenas1, S. Genaim2, G. Puebla2, D. Ramírez2, and D. Zanardini2

1 DSIC, Complutense University of Madrid, E-28040 Madrid, Spain
2 CLIP, Technical University of Madrid, E-28660 Boadilla del Monte, Madrid, Spain

COSTA [2] is a research prototype which performs automatic program analysis, in the style of [9,7,8], and which is able to infer cost [4] and termination [1] information about Java bytecode programs. The system receives as input a bytecode program and a cost model chosen from a selection of resource descriptions, and tries to bound the resource consumption of the program with respect to the given cost model. COSTA provides several non-trivial notions of resource, such as the amount of memory allocated on the heap [5], the number of bytecode instructions executed, the number of billable events (such as sending a text message on a mobile phone) executed by the program. When performing cost analysis, COSTA produces a cost equation system [3], which is an extended form of recurrence relations. In order to obtain a closed (i.e., non-recursive) form for such recurrence relations which represents an upper bound, COSTA includes a dedicated solver [6]. An interesting feature of COSTA is that it uses pretty much the same machinery for inferring upper bounds on cost as for proving termination (which also implies the boundedness of any resource consumption).

In this presentation we will show the recently developed COSTA web interface. It allows users to try out the system on a set of representative examples, and also to upload their own bytecode programs such as class or jar files. As the behaviour of COSTA can be customized using a relatively large set of options, the web interface allows two different alternatives for choosing the values for such options.

Fig. 1. Automatic analysis options

* Tool demo description.
The first alternative, which we call automatic (see Figure 1), allows the user to choose from a range of possibilities which differ in the analysis accuracy and overhead. Starting from level 0, the default, we can increase the analysis accuracy (and overhead) by using levels 1 through 3. We can also reduce analysis overhead (and accuracy) by going down to levels -1 through -3. All this, without requiring the user to understand the different options implemented in the system and their implications in analysis accuracy and overhead.

The second alternative is called manual (see Figure 2) and it is meant for the expert user. There, the user has access to all of the analysis options available, allowing a fine-grained control over the behaviour of the analyzer. Some of these options include whether to analyze the Java standard libraries, to take exceptions into account, to perform or not a number of pre-analyses, to write/read analysis results to file in order to reuse them in later analyses, etc.
We will show analyses using different cost models and also analyze applications for both Standard Edition Java and Micro Edition Java (in particular, for the MIDP profile for mobile phones). Figure 3 shows the output of COSTA on an example program. In addition to showing the result of termination analysis and an upper bound on the execution cost, some data is displayed about the intermediate steps performed by the analyzer. In this case, the upper bound obtained for a method is shown and includes the cost of calls to some Java library methods.

Acknowledgments  This work was funded in part by the Information Society Technologies program of the European Commission, Future and Emerging Technologies under the IST-15905 MOBIUS project, by the Spanish Ministry of Education under the TIN-2005-09207 MERIT project, and the Madrid Regional Government under the S-0505/TIC/0407 PROMESAS project.

References