The COSTA Cost and Termination Analyzer for Java Bytecode and its Web Interface

E. Albert[1], P. Arenas[1], S. Genaim[2],
G. Puebla[2], D. Ramírez[2], and D. Zanardini[2]

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

COSTA is a research prototype which performs automatic program analysis and which is able to infer COST [**?**] and TERMINATION [**?**] 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 [**?**], 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*, 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 [**?**]. 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 the proposed demo 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. 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.

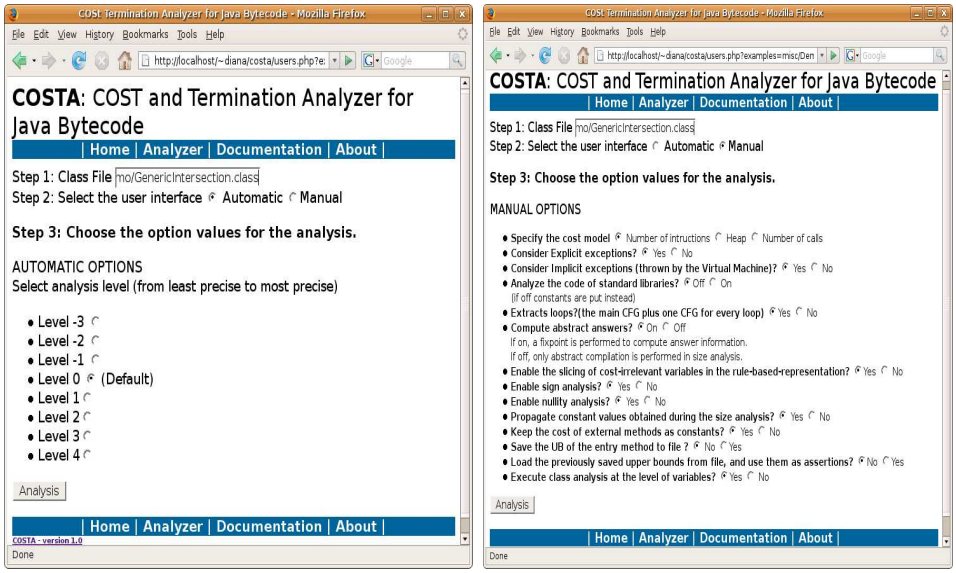

Fig. 1. Two ways of setting values for analysis options

The first alternative, which we call *automatic* (see Figure 1, left-hand side), 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 1, right-hand side) 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. In the demo, 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).

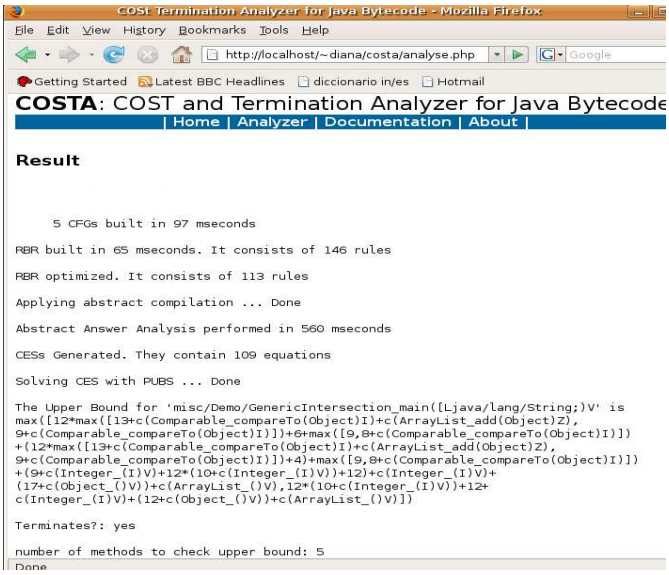

Fig. 2. Results

Figure 2 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 program is proved to terminate and an upper bound is shown which includes the cost of calls to several Java library methods.