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

(Extended Abstract)

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 [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.

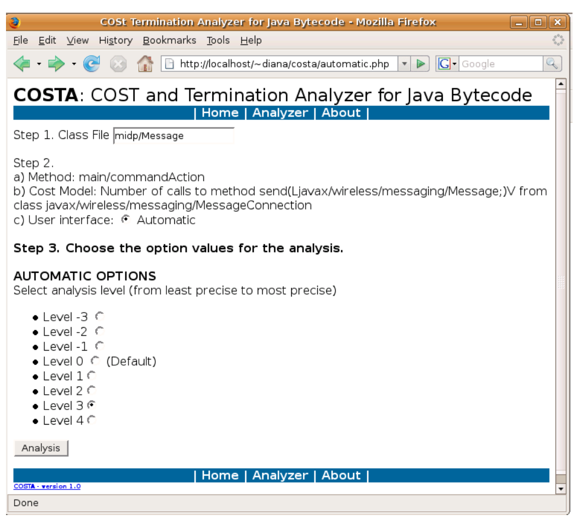

Fig. 1. Automatic analysis options

* Tool demo description.

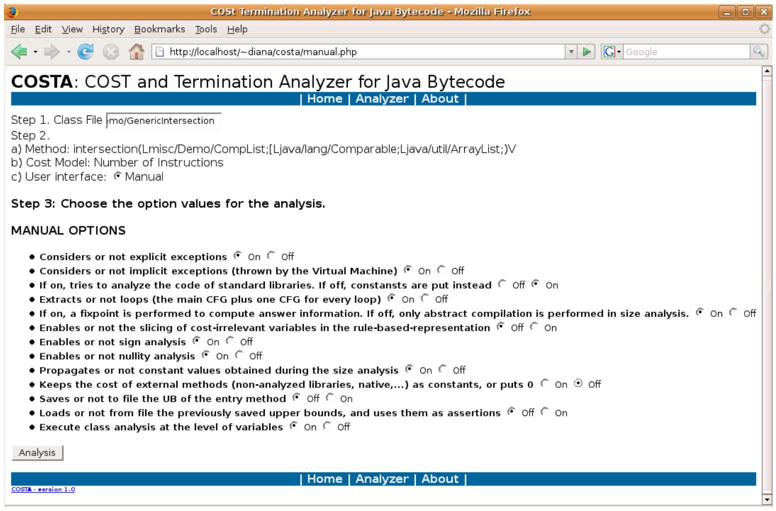

Fig. 2. Manual analysis Options

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.

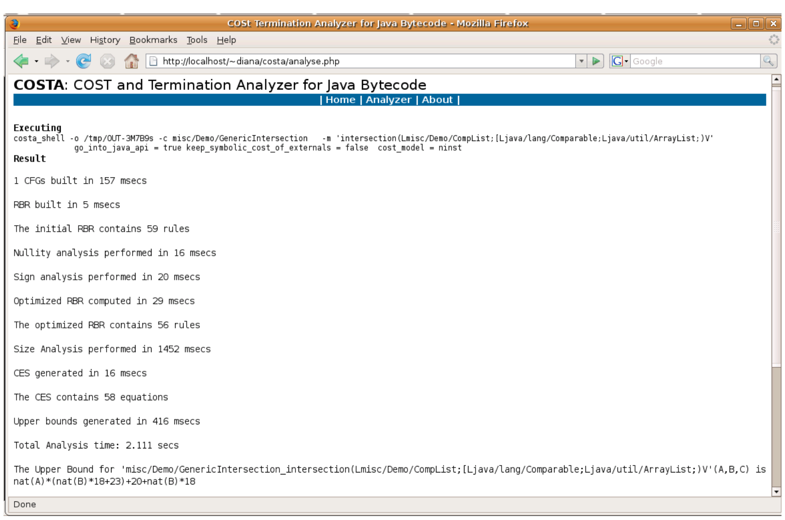

Fig. 3. Results

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

1. E. Albert, P. Arenas, M. Codish, S. Genaim, G. Puebla, and D. Zanardini. Termination Analysis of Java Bytecode. In Gilles Barthe and Frank de Boer, editors, *Proceedings of the IFIP International Conference on Formal Methods for Open Object-based Distributed Systems (FMOODS)*, volume 5051 of *Lecture Notes in Computer Science*, pages 2–18, Oslo, Norway, June 2008. Springer-Verlag, Berlin.
2. E. Albert, P. Arenas, S. Genaim, G. Puebla, , and D. Zanardini. COSTA: Design and Implementation of a Cost and Termination Analyzer for Java Bytecode. In *Postproceedings of Formal Methods for Components and Objects (FMCO'07)*, LNCS. Springer-Verlag, 2008. To appear.
3. E. Albert, P. Arenas, S. Genaim, and G. Puebla. Cost Relation Systems: a Language–Independent Target Language for Cost Analysis. In *Spanish Conference on Programming and Computer Languages (PROLE'08)*, ENTCS. Elsevier, October 2008. To appear.
4. E. Albert, P. Arenas, S. Genaim, G. Puebla, and D. Zanardini. Cost Analysis of Java Bytecode. In Rocco De Nicola, editor, *16th European Symposium on Programming, ESOP'07*, volume 4421 of *Lecture Notes in Computer Science*, pages 157–172. Springer, March 2007.
5. E. Albert, S. Genaim, and M. Gomez-Zamalloa. Heap Space Analysis for Java Bytecode. In *ISMM '07: Proceedings of the 6th international symposium on Memory management*, pages 105–116, New York, NY, USA, October 2007. ACM Press.
6. Elvira Albert, Puri Arenas, Samir Genaim, and German Puebla. Automatic Inference of Upper Bounds for Recurrence Relations in Cost Analysis. In Marı'a Alpuente and Germán Vidal, editors, *Static Analysis, 15th International Symposium, SAS 2008, Valencia, Spain, July 15-17, 2008, Proceedings*, volume 5079 of *Lecture Notes in Computer Science*, pages 221–237. Springer-Verlag, July 2008.
7. D. Le Metayer. ACE: An Automatic Complexity Evaluator. *ACM Transactions on Programming Languages and Systems*, 10(2):248–266, April 1988.
8. M. Rosendahl. Automatic Complexity Analysis. In *Proc. ACM Conference on Functional Programming Languages and Computer Architecture*, pages 144–156. ACM, New York, 1989.
9. B. Wegbreit. Mechanical Program Analysis. *Comm. of the ACM*, 18(9), 1975.