Abstract State Machines and High-Level System Design and Analysis

Egon Börger [a,*],

[a]*Dipartimento di Informatica, Università di Pisa, Via F. Buonarroti 2, I-56127 Pisa, Italy*

Abstract

The background and the themes of this ASM-centered special issue of TCS are shortly described.

Key words: Abstract State Machines, High-Level System Design, Mathematical System Analysis, Refinement, Verification, Testing, Coordination, Mobile Networks, C#

The year 2003 was a landmark in the short history of the Abstract State Machines (ASM) method [1], a mathematics-based industrially successful method for the design and the analysis of complex computer-based systems. After the authoritative formulation of the underlying notion of ASMs—its definition was presented in the Summer School on *Specification and Validation Methods* I organized in 1993 on the Lipari island and was published in [2]—only ten years passed to see the first monograph and textbook on the ASM method appear [3] and the *10th International Workshop on Abstract State Machines* [4] be held in Taormina. The workshop aimed at an integration of ASM-based modeling, validation and verification techniques into neighboring system engineering methods and thus featured invited lectures on object-oriented, component-based design and program verification techniques, mobile computing, testing, concurrency and refinement frameworks. This triggered the invitation by Don Sanella, which I accepted with pleasure, to edit a special issue of this Journal with elaborations of selected papers from ASM 2003 and of related recent work on themes in the spectrum of the workshop. The pre-selection was based upon the reviews written for the Proceedings of ASM 2003 [4], followed by a final selection through a standard reviewing process which resulted in the seven

* Corresponding author.
Email address: boerger@di.unipi.it (Egon Börger).

[1] For a detailed historical account see [1].

papers appearing here[2]. Their unifying thread is the systematic use of mathematical theories and techniques, whether ASM-based or not, to contribute to the solution of system engineering problems.

The first paper is on a promising method to combine runtime verification with automatic test case generation, based on systematically exploring the input domain of the program using a model checker with symbolic execution. Execution traces are monitored and verified against properties expressed in temporal logic, with capabilities for analyzing traces for concurrency errors, such as deadlocks and data races.

In the second paper the ASM models in [5] for Java and the JVM are reused to dis-cover the mathematical structure that underlies the semantics of C♯, providing also a basis for a precise comparative analysis of the two languages and their implementations. Theorem proving systems are challenged to machine-assist this verification work.

In the third paper asynchronous (also called distributed) ASMs are used as a mathematical reference to model, analyze and validate (via prototypical execution) a routing layer protocol for mobile ad hoc networks. The protocol provides senders with the needed information on the most recent physical location of the destination node, which each network node can find via GPS-like navigation technologies.

In the fourth paper the authors continue their previous work on a general algebraic framework to analyze the consistency of simultaneous, possibly nested, modifications of structured data by independent agents. The ASM-based approach is extended from counters, sets and maps to sequences and labeled ordered trees.

In the fifth paper the authors' logic for ASMs is extended to accomodate proving security properties of ASMs and correctness of refinements of parallel ASMs to sequential C-like programs. A read predicate is introduced that allows to make precise statements about the accesses of locations of an ASM. The logic is shown to be complete for hierarchical ASMs and sound for turbo ASMs.

In the sixth paper some major coordination models, tailored for use in mobile computing to describe the interaction of independent system components, are analyzed. The constructs these models provide are shown to be reducible to simple schema definitions in Mobile UNITY, an extension of the well-known UNITY language—which is conceptually close to the language of ASMs (see [6]).

[2] The reviewing procedure for the second paper, where I am a co-author, has been managed separately by the main TCS editor Don Sanella.

In the seventh paper the relation between the notions of ASM refinement and data refinement is explored. It is shown that forward simulation in the behavioral approach to data refinement can be viewed as a specific instance of ASM refinement with 1:1 diagrams without control structure refinement. Also weak and coupled refinement, two recent generalizations of data refinement, are shown to be instances of the notion of ASM refinement.

To understand the papers in this issue no previous knowledge of ASMs is needed since ASMs can be correctly understood as pseudo-code (guarded commands) operating on abstract data, or as extension of finite state machines by updates of abstract states (structures in the Tarskian sense, known from logic). Therefore I abstain from repeating here the technical definition of Gurevich's concept of ASMs, which is the first constituent of the ASM method. The reader who wants to see the details may consult the above mentioned Lipari Guide [2] or the AsmBook [3]. A general explanation and survey of the other two constituents I introduced into the ASM system design and analysis method, namely the concept of ASM ground models—to faithfully capture informal requirements—and the general notion of ASM refinement—to correctly turn abstract models by human-controllable steps into code—is available in two recent papers [7,8].

We hope that some theoretically inclined reader will be attracted by the papers in this volume to contribute to the further development of the theory of ASMs or to enlarge the body of successful applications of the ASM method through the description and the mathematical investigation of real-life complex computer-based systems.

Egon Börger, Pisa 17.5.2004

Acknowledgements

I gratefully acknowledge the generous and decisive help of the numerous reviewers who have to remain anonymous. Without their constructive criticism this issue of TCS would not be what it is.

References

[1] E. Börger, The Origins and the Development of the ASM Method for High-Level System Design and Analysis, J. Universal Computer Science 8 (1) (2002) 2–74.

[2] Y. Gurevich, Evolving Algebras 1993: Lipari Guide, in: E. Börger (Ed.), Specification and Validation Methods, Oxford University Press, 1995, pp. 9–36.

[3] E. Börger, R. F. Stärk, Abstract State Machines. A Method for High-Level System Design and Analysis, Springer-Verlag, 2003.

[4] E. Börger, A. Gargantini, E. Riccobene (Eds.), Abstract State Machines 2003–Advances in Theory and Applications, Vol. 2589 of Lecture Notes in Computer Science, Springer-Verlag, 2003.

[5] R. F. Stärk, J. Schmid, E. Börger, Java and the Java Virtual Machine—Definition, Verification, Validation, Springer-Verlag, 2001.

[6] E. Börger, Abstract State Machines: A Unifying View of Models of Computation and of System Design Frameworks, Annals of Pure and Applied Logic (to appear).

[7] E. Börger, The ASM Ground Model Method as a Foundation of Requirements Engineering, in: N.Dershowitz (Ed.), Verification: Theory and Practice, Vol. 2772 of LNCS, Springer-Verlag, 2003, pp. 145–160.

[8] E. Börger, The ASM Refinement Method, Formal Aspects of Computing 15 (2003) 237–257.