Fundamenta Informaticae 77 (2007) i–iv
IOS Press

The Abstract State Machines Method

Preface

This special issue of Fundamenta Informaticae contains a selection of revised versions of papers presented to ASM'05 [2], the 12th of a series of *International Workshops on Abstract State Machines* (ASMs) dedicated to mathematical techniques and their implementation for high-level system design and analysis related to ASMs.

What became known as the *ASM Method* [15] is based upon the mathematical definitions of *ASM* [22], *ASM ground model* [9] and *ASM refinement* [10], which were guided by practical applications. The definition of ASMs in [22] came out of 10 years of extensive research and experimentation with "descriptions of computational devices", carried out by the at the time small but fast growing ASM community (see the survey in the first International ASM Workshop in Hamburg in 1994 [24, Stream C (Evolving Algebras) pp. 377–441]) and solicited to support *A New Thesis* [21], which in 1985 was formulated by Yuri Gurevich in the *Notices of the American Mathematical Society* as follows:

> Turing's thesis is that every computable function can be computed by an appropriate Turing machine. The informal proof of the thesis gives more: every computing device can be simulated by an appropriate Turing machine. The following much stronger form of the thesis seems to be very much accepted today: every sequential computing device can be simulated by an appropriate Turing machine in polynomial time. First, we adapt Turing's thesis to the case when only devices with bounded resources are considered. Second, we define a more general kind of abstract computational device, called dynamic structures, and put forward the following new thesis: Every computational device can be simulated by an appropriate dynamic structure – of appropriately the same size – in real time; a uniform family of computational devices can be uniformly simulated by an appropriate family of dynamic structures in real time. In particular, every sequential computational device can be simulated by an appropriate sequential dynamic structure. A contribution of Andreas Blass is acknowledged. Descriptions of computational devices are solicited for further confirmation of the thesis. (Received May 13, 1985) (Sponsored by Andreas Blass) [21]

Whereas in the technical report preceding the formulation of *A New Thesis* the concept of "dynamic structures" was still considered as a notion which "is to remain informal" [20, p. 1], its mathematical definition in [22] had two methodological effects. On the theoretical side it permitted to sharpen the formulation of the sequential version of the new thesis, namely to "Every sequential algorithm can be *step-by-step* simulated by an appropriate sequential ASM", so that it could eventually be turned into a theorem one can prove from three natural axioms [23]. On the practical side the definition provided the basis for the definitions of the *ASM ground model* concept [9] and of the generalization of Wirth's [27] and related traditional refinement notions [28, 1, 16, 18, 17] to that of *ASM refinement* [10], both of which pervade the applications of ASMs since their appearance in the early work on ASM models for the ISO Prolog standard [7, 8] and made the ASM method to become a working tool for design, analysis and documentation of complex real-life systems, including systems of industrial size (see the survey in the history chapter of the AsmBook [15, Ch.9]).

The wide range from theory to practice spanned by the ASM method, reaching from complexity [4] and logic [3, 5] over verifiable design and implementation of real-life programming languages [25, 19, 13] and hardware architectures [12, 26] to industrial software reengineering [14], is reflected also in this special issue of FI, which contains papers on the refinement method, on abstract-machine-based simulation and verification techniques and on mathematical investigations into the concept of timed computations and a class of PTIME ASMs.

In the paper *Refinement, Decomposition, and Instantiation of Discrete Models: Application to Event-B* by Jean-Raymond Abrial and Stefan Hallerstede model refinement is coupled to two further system construction techniques, namely decomposition and generic instantiation. The formulations are in terms of Event-B machines and thereby apply to ASMs via the characterization of Event-B machines by a class of ASMs [11].

In the paper *Retrenching the Purse: The Balance Enquiry Quandardy, and Generalized and (1,1) Forward Refinements* by Richard Banach, Michael Poppleton, Czeslaw Jeske and Susan Steney the notion of retrenchment, proposed to alleviate some technical difficulties with traditional refinement notions, is applied to the famous Mondex Purse case study.

In the paper *CoreASM: An Extensible ASM Execution Engine* by Roozbeh Farahbod, Vincenzo Gervasi and Uwe Glässer an ASM ground model is defined for the design of a new platform-independent engine to execute ASMs, which comes equipped with a well-defined interface for the integration of other software development, validation and verification tools, including interactive visualization facilities.

In the paper *Model Checking Abstract State Machines with Answer Set Programming* by Calvin Kai Fan Tang and Eugenia Ternovska it is shown how to solve the bounded model checking problem for ASMs by computing an answer set for a corresponding logic program.

In the paper *Time in State Machines* by Susanne Graf and Andreas Prinz a true concurrency model for timed (e.g. ASM) computations is defined by enriching event structures with a notion of partially ordered time, in such a way that by appropriate constraints on event structures and their runs one can describe the major timed computation models in the literature.

In the paper *RAM simulation of BGS model of abstract-state machines* by Comandur Seshadhri, Anil Seth and Somenath Biswas it is proved that the model of PTIME ASMs defined in [6] can be simulated by RAMs with at most polynomial overhead.

We thank the 34 colleagues who helped with reviewing (a) the elaborated full versions of ASM'05 papers that were submitted in the Fall of 2005 to this special FI issue and (b) in the Spring and Early Summer of 2006 the revised versions of those selected for publication in this special issue of FI.

The guest editors
Egon Börger
Università di Pisa
Dipartimento di Informatica, I-56125 Pisa, Italy
boerger@di.unipi.it

Anatol Slissenko
Université Paris 12,
Département d'Informatique, 94010 Crteil, France
slissenko@univ-paris12.fr

Pisa and Paris, Summer 2006.

References

[1] R. J. R. Back and J. von Wright. *Refinement Calculus: A Systematic Introduction*. Springer-Verlag, 1998.

[2] D. Beauquier, E. Börger, and A. Slissenko, editors. *Proceedings of the 12th International Workshop on Abstract State Machines ASM'05*. Université Paris 12, March 2005.
Available at *www.univ-paris12.fr/lacl/dima/asm05*

[3] D. Beauquier and A. Slissenko. A first-order logic for specification of timed algorithms: Basic properties and a decidable class. *Annals of Pure and Applied Logic*, 113(1–3):13–52, 2001.

[4] A. Blass and Y. Gurevich. The linear time hierarchy theorems for Abstract State Machines. *J. Universal Computer Science*, 3(4):247–278, 1997.

[5] A. Blass and Y. Gurevich. The logic of choice. *J. Symbolic Logic*, 65(3):1264–1310, 2000.

[6] A. Blass, Y. Gurevich, and S. Shelah. Choiceless polynomial time. *Annals of Pure and Applied Logic*, 100:141–187, 1999.

[7] E. Börger. A logical operational semantics for full Prolog. Part I: Selection core and control. In E. Börger, H. Kleine Büning, M. M. Richter, and W. Schönfeld, editors, *CSL'89. 3rd Workshop on Computer Science Logic*, volume 440 of *Lecture Notes in Computer Science*, pages 36–64. Springer-Verlag, 1990.

[8] E. Börger. A logical operational semantics of full Prolog. Part II: Built-in predicates for database manipulation. In B. Rovan, editor, *Mathematical Foundations of Computer Science*, volume 452 of *Lecture Notes in Computer Science*, pages 1–14. Springer-Verlag, 1990.

[9] E. Börger. The ASM ground model method as a foundation of requirements engineering. In N.Dershowitz, editor, *Verification: Theory and Practice*, volume 2772 of *LNCS*, pages 145–160. Springer-Verlag, 2003.

[10] E. Börger. The ASM refinement method. *Formal Aspects of Computing*, 15:237–257, 2003.

[11] E. Börger. From finite state machines to virtual machines (Illustrating design patterns and event-B models). In E. Cohors-Fresenborg and I. Schwank, editors, *Präzisionswerkzeug Logik–Gedenkschrift zu Ehren von Dieter Rödding*. Forschungsinstitut für Mathematikdidaktik Osnabrück, 2006. ISBN 3-925386-56-4.

[12] E. Börger and G. Del Castillo. A formal method for provably correct composition of a real-life processor out of basic components (The APE100 Reverse Engineering Study). In B. Werner, editor, *Proc. 1st IEEE Int. Conf. on Engineering of Complex Computer Systems (ICECCS'95)*, pages 145–148, November 1995.

[13] E. Börger, G. Fruja, V. Gervasi, and R. Stärk. A high-level modular definition of the semantics of C#. *Theoretical Computer Science*, 336(2-3):235–284, 2004.

[14] E. Börger, P. Päppinghaus, and J. Schmid. Report on a practical application of ASMs in software design. In Y. Gurevich, P. Kutter, M. Odersky, and L. Thiele, editors, *Abstract State Machines: Theory and Applications*, volume 1912 of *Lecture Notes in Computer Science*, pages 361–366. Springer-Verlag, 2000.

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

[16] W. P. de Roever and K. Engelhardt. *Data Refinement: Model-Oriented Proof Methods and their Comparison*. Cambridge University Press, Cambridge, 1998.

[17] J. Derrick and E. Boiten. *Refinement in Z and Object-Z.* Formal Approaches to Computing and Information Technology. Springer-Verlag, 2001.

[18] J. Fitzgerald and P. G. Larsen. *Modeling Systems. Practical Tool and Techniques in Software Development*. Cambridge University Press, Cambridge, 1998.

[19] U. Glässer, R. Gotzhein, and A. Prinz. Formal semantics of SDL-2000: Status and perspectives. *Computer Networks*, 42(3):343–358, June 2003.

[20] Y. Gurevich. Reconsidering Turing's Thesis: Toward more realistic semantics of programs. Technical Report CRL-TR-36-84, EECS Department, University of Michigan, September 1984.

[21] Y. Gurevich. A new thesis. *Abstracts, American Mathematical Society*, 6(4):317, August 1985.

[22] Y. Gurevich. Evolving algebras 1993: Lipari Guide. In E. Börger, editor, *Specification and Validation Methods*, pages 9–36. Oxford University Press, 1995.

[23] Y. Gurevich. Sequential Abstract State Machines capture sequential algorithms. *ACM Trans. Computational Logic*, 1(1):77–111, July 2000.

[24] B. Pehrson and I. Simon. I: Technology/foundations. In *IFIP 13th World Computer Congress 94*, Elsevier, Amsterdam, 1994.

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

[26] J. Teich, R. Weper, D. Fischer, and S. Trinkert. A joint architecture/compiler design environment for ASIPs. In *Proc. Int. Conf. on Compilers, Architectures and Synthesis for Embedded Systems (CASES2000)*, pages 26–33, San Jose, CA, USA, November 2000. ACM Press.

[27] N. Wirth. Program development by stepwise refinement. *Commun. ACM*, 14(4), 1971.

[28] J. C. P. Woodcock and J. Davies. *Using Z: Specification, Refinement, and Proof.* Prentice-Hall, 1996.