20 Years after *A New Thesis*: The Diversification of the ASM Method

Danièle Beauquier[1] and Egon Börger[2] and Anatol Slissenko[1]

[1] Université Paris 12, Laboratoire d'Algorithmique, Complexité et Logique, Paris, France
{beauquier,slissenko}@univ-paris12.fr

[2] Università di Pisa, Dipartimento di Informatica, I-56125 Pisa, Italy boerger@di.unipi.it
On sabbatical leave at SAP Research, Karlsruhe, Germany egon.boerger@sap.com

Abstract. 20 years ago Gurevich formulated *A New Thesis* [10], soliciting descriptions of computational devices to unfold the underlying notion of "dynamic structures". 10 years later, after extensive experimentation with "dynamic algebras" by a at the time still small community, Gurevich discovered a mathematical definition for "evolving algebras" [11], which remained stable except for the final name change in 1996 to *Abstract State Machines*. Combining the ASM notion with Börger's concept of *ASM ground model* [3] and with his general *ASM refinement* definition [4] led to the ASM method for high-level system design and analysis, which was shaped through the work of the by now established ASM community, became visible in practically every branch of the science of computing and quickly found its way into industrial practice. Five years later, a mathematical investigation of the conditions for the possibility of this success appeared in Gurevich's penetrating epistemological analysis of sequential algorithms in terms of sequential ASMs [12].
The *International Workshops on Abstract State Machines* are the by now traditional place where new ideas for the further development of the ASM concept and method are presented and discussed, relating them to similar developments in other approaches to system design. The Proceedings of ASM 2005 show where we are today in this rather dynamic process. We hope they will help to forge new ideas and to discern the ones worth to be transfered into publications and software systems.

Anniversary of A New Thesis. ASM 2005, the 12th in the series of annual *International Workshops on ASMs*, marks the 20th anniversary of *A New Thesis*, formulated by Gurevich in 1985 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) [10]

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" [9, p. 1]. After 10 years of successful research and experimentation with "descriptions of computational devices", carried out by the (at the time small but fast growing) ASM community and surveyed in the first

International ASM Workshop in Hamburg in 1994 [13, Stream C (Evolving Algebras) pp. 377–441], Gurevich discovered a remarkably natural mathematical definition for the concept [11]. It did not change any more since then and permitted to sharpen the formulation of the sequential version of the new thesis to "Every sequential algorithm can be *step-by-step* simulated by an appropriate sequential ASM". Furthermore, the definition provided the basis for Börger's definitions of the ASM ground model concept [3] and of the general ASM refinement notion [4], both of which pervade the applications of ASMs since their appearance in the early work on ASM models for the ISO Prolog standard [2]. They made the ASM method to become a working tool for design, analysis and documentation of complex real-life systems, including systems of industrial size.

We all know the success the ASM method found in both computation theory and the practice of engineering software systems [1]. Within five years after the definitive mathematical definition of ASMs, a mathematical investigation of the conditions for the possibility of this success appeared in Gurevich's penetrating epistemological analysis of sequential algorithms in terms of sequential ASMs [12]. On the practical side, we only mention here the emergence, during these five years, of three new tools for use in validating ASMs in challenging industrial projects, namely the ASM Workbench [7], AsmGofer [14] and XASM [1]—which can also be used to support the validation of examples in courses where ASMs are taught. 2000 is also the year when the development of AsmL [8] at Microsoft Research was started, a rather successful system to which many contributions in these Proceedings refer and which we look forward to see integrated into the new extensible execution engine and tool environment CoreASM proposed at this year's ASM workshop (Farahbod et al., p. 153).

These Proceedings. The wide range from theory to industrial practice spanned by the ASM method is reflected in these Proceedings, which document the *invited lectures*, the submitted *research papers* and the numerous short texts submitted for the presentation of *work in progress*.

On the practical side this volume contains a great number of papers providing **ASM ground models**, illustrating the fundamental role such mathematical high-level system blueprints play in various domains. You find ASMs for modeling such different things as GUIs (Paiva et al., p. 329), fault and event handlers in web services (Fahland & Reisig, p. 131), data warehouses (Schewe & Zhao, p. 369), the runtime environment CLR for executing .NET applications (Fruja, p. 175), an interpreter for the extension C#2.0 of C# (Jula, p. 263), security protocols (Haneberg et al., p. 247), object-oriented desing patterns (Börger, p. 91), but also urban crime patterns (Brantingham et al., p. 101). This illustrates the trend one can observe towards a diversification of the ground model method. The method seems to find its way into new application domains, which are characterized by rule-based system evolutions of processes that are moving fast as conceptual and methodological challenges from industrial applications into the focus of computing science research, like administration and business processes, medical and biological systems, automotive (car, train, flight) control systems, etc.

Ground models would be of less practical impact couldn't they be validated. Numerous papers in these Proceedings are devoted to the theme of how to validate and verify ASMs, which was the focus of the Special Session on Tools, Verification and Validation, organized by U. Glässer and M. Veanes. You find papers dealing with the following: an approach to test GUIs (Paiva et al., p. 329), a new exploration algorithm to be used for test generation (Campbell & Veanes, p. 119), validation of web-based applications (Binemann-Zdanowicz et al., p. 417); model checking using answer set programming (Tang & Ternovska, p. 397)and model checking of AsmL specifications (Kardos, p. 289); an integration of the Property Specification Language PSL into AsmL so that properties of an AsmL program can be model checked for the FSM generated from the program by the AsmL tool (Gawanmeh et al., p. 201); last but not least you find the derivation of a concrete ASM syntax from a MOF-compliant ASM metamodel (Scandurra et al., p. 345)and the architecture description for the new extensible ASM execution engine CoreASM striving for maximal flexibility and minimal encoding (Farahbod et al., p. 153).

[1] For the newcomers we point to the detailed historical survey in the AsmBook [6, Ch.9]

The tool supported verification and validation theme was also at the heart of the invited lectures in the Special Session on Tools, Verification and Validation, showing related new developments and future trends in other system development methods:

- M. Butler: Tools and techniques for system-level validation with abstract machines
- B. Cook: Finding deep device driver bugs at compile-time
- R. Leino: Program verification and programming methodology

Refinement is the second major practical theme in these Proceedings where you find an extension of the ASM model for C# developed in [5] to one for C# 2.0 (Jula, p. 263), a refinement of the ASM model for C# to an AsmL executable model (Jula & Fruja, p. 275), definition and analysis of refinement rules for on-line analytical processing systems (Schewe & Zhao, p. 369)and for design patterns in object-oriented programmming (Börger, p. 91). We observe here a similar challenging trend for diversification as mentioned above for the use of ASM ground models: the programming-centric view of refinement as a link between in-line-documented requirements and the implementing code is going to be replaced by general ASM refinement patterns supporting the mathematically controllable construction of hierarchies (often sequences) of stepwise detailed ASM models, which document the major design decisions and allow one to trace the original application-domain-oriented requirements to their final implementation in executable code. This trend comes out clearly also in the four invited lectures in the Special Session on Refinement, organized by E. Börger to provide an insight into the relations between the ASM refinement notion and the refinement definitions used in other approaches:

- J-R Abrial: Decomposition, refinement and instantiation of generic models
- R. Banach: Retrenchment and the Mondex Electronic Purse
- G. Schellhorn: ASMs and refinement of state-based systems
- J. Woodcock:Unifying program refinement calculi

On the **theoretical side** one finds an elegant new proof for the poly-time RAM simulation of BGS (Seshandri et al., p. 377), a new poly-time algorithm for slicing ASMs (Nowak, p. 313), a characterization of a class of ASMs to compute the primitive recursive functions (Andary et al., p. 77), ideas on exploiting linearity restrictions for ASMs to algorithmically check certain simple invariants (Novikov & Yavorskiy, p. 305)or to analyse computations satisfying certain temporal logic formulae considered as descriptions of scenarios (Filatov & Yavorskiy, p. 167). One also finds contributions to challenging foundational conceptual issues: an analysis of the notion of time for state transition systems including ASMs (Graf & Prinz, p. 217), an investigation of the notion of implementation for real-time ASMs with instantaneous actions by machines with delayed actions (Slissenko & Cohen, p. 387)and a small-step trace-based fixpoint semantics of an AsmL core (Habibi & Tahar, p. 233). The theory-related part of the workshop was opened by the following invited lecture, where ASMs are used for a foundation of a new computation theory:

- Y. Gurevich: Behavioral Computation Theory 2005

The **security theme** was covered in the invited lecture in the Special Session on Security, organized by A. Slissenko, and during the invited lecture presenting Active XML:

- S. Abiteboul: Active XML: embedding Web services into XML
- H. Comon: Normal attacks on security protocols

There was one invited lecture in the Special Session on Teaching ASMs, organized by C. Wallace:

- E. Börger: The ASM Method: a Cornerstone in Computer Science Education

Besides the diversification of the ASM method, the Proceedings show that much remains to be done to further develop it, in particular to integrate it further with other successful design, verification and validation approaches and to make it fit for new theoretical and practical challenges.

The 12th ASM Workshop. ASM 2005 is the 12th in the *International Workshop on Abstract State Machines* series and the first one held in Paris. The previous editions were held in the following European cities: Wittenberg, Germany (2004); Taormina, Italy (2003); Dagstuhl, Germany

(2002); Las Palmas de Gran Canaria, Spain (2001); Monte Verità, Switzerland (2000); Toulouse, France (1999); Magdeburg, Germany (1998); Cannes, France (1998, 1997); Paderborn, Germany (1996); Hamburg, Germany (1994).

We thank U. Glässer (Simon Fraser University, Vancouver, Canada) and M. Veanes (Microsoft Research, Redmond, USA) for having organized as part of ASM 2005 the special session on Tools, Verification and Validation. The same thanks go to C. Wallace (Michigan Technological University, Michigan, USA) for the special session on ASMs in Education. We thank the invited speakers for having contributed to the intensive exchange of ideas in the workshop.

We thank our colleagues in the program committee for their support in evaluating the papers submitted to ASM 2005, thus providing a basis for the evaluation of the contributions that will be submitted to the special journal issue we intend to publish after the workshop to contain the revised final versions of the papers. We thank our colleagues and students of the Université Paris 12 for their generous help with the organization of ASM 2005. For financial support of the workshop we thank the sponsoring institutions listed below. Last but not least we thank the participants of ASM 2005 for having made our work useful.

February 2005, Paris and Pisa/Karlsruhe

Danièle Beauquier
Egon Börger
Anatol Slissenko

1 Organization Committee

D. Beauquier	(France)
P. Cegielski	(France)
J. Cohen	(France)
C. Dima	(France)
M. Duflot	(France)

2 Program Committee

E. Asarin,	(France)
D. Beauquier	(France)
E. Börger	(Italy) (co-chair)
P. Cegielski	(France)
A. Gargantini	(Italy)
U. Glässer	(Canada)
S. Grigorieff	(France)
J. Janneck	(USA)
E. Riccobene	(Italy)
A. Slissenko	(France) (co-chair)
W. Schulte	(USA)
B. Thalheim	(Germany)
M. Veanes	(USA)
C. Wallace	(USA)
W. Zimmermann	(Germany)

3 Sponsoring Institutions

Université Paris 12, France
Microsoft Reasearch, Redmond, WA, USA

Laboratoire d'Algorithmique, Complexité et Logique (LACL), France
Conseil Général du Val de Marne, France

References

1. M. Anlauff and P. Kutter. Xasm Open Source. Web pages at `http://www.xasm.org/`, 2001.
2. E. Börger. *Berechenbarkeit, Komplexität, Logik*, chapter Dynamische Algebren und Semantik von Prolog, pages 476–499. Vieweg, 3rd edition, 1992.
3. 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.
4. E. Börger. The ASM refinement method. *Formal Aspects of Computing*, 15:237–257, 2003.
5. E. Börger, G. Fruja, V. Gervasi, and R. Stärk. A high-level modular definition of the semantics of C#. *Theoretical Computer Science*, 2004.
6. E. Börger and R. F. Stärk. *Abstract State Machines. A Method for High-Level System Design and Analysis*. Springer, 2003.
7. G. Del Castillo. *The ASM Workbench. A Tool Environment for Computer-Aided Analysis and Validation of Abstract State Machine Models*. PhD thesis, Universität Paderborn, Germany, 2001.
8. Foundations of Software Engineering Group, Microsoft Research. AsmL. Web pages at `http://research.microsoft.com/foundations/AsmL/`, 2001.
9. 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.
10. Y. Gurevich. A new thesis. *Abstracts, American Mathematical Society*, 6(4):317, August 1985.
11. Y. Gurevich. Evolving algebras 1993: Lipari Guide. In E. Börger, editor, *Specification and Validation Methods*, pages 9–36. Oxford University Press, 1995.
12. Y. Gurevich. Sequential Abstract State Machines capture sequential algorithms. *ACM Trans. Computational Logic*, 1(1):77–111, July 2000.
13. B. Pehrson and I. Simon. I: Technology/foundations. In *IFIP 13th World Computer Congress 94*, Elsevier, Amsterdam, 1994.
14. J. Schmid. Executing ASM specifications with AsmGofer. Web pages at `http://www.tydo.de/AsmGofer`.