Preface

Abstract state machines (ASM) sharpen the Church-Turing thesis by the consideration of bounded resources for computing devices. They view computations as an evolution of a state. It has been shown that all known models of computation can be expressed through specific abstract state machines. These models can be given in a representation-independent way. That is one advantage of transferring these models to ASM. The main advantage is, however, to provide a unifying theory to all of these models. At the same time ASM can be refined to other ASMs. Stepwise refinement supports separation of concern during software development and will support component-based construction of systems thus providing a foundation of new computational paradigms such as industrial programming, programming-in-the-large, and programming-in-the-world.

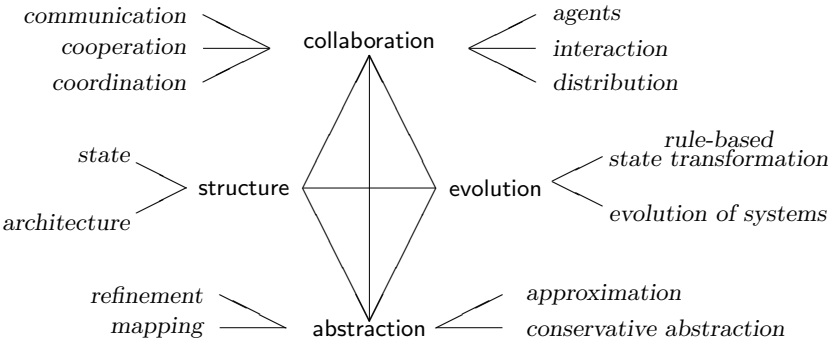
ASM 2004 continued the success story of the ASM workshops. Previous workshops were held in the following European cities: Taormina, Italy (2003); Dagstuhl, Germany (2002); Las Palmas de Gran Canaria, Spain (2001); Monte Verita, Switherland (2000); Toulouse, France (1999); Magdeburg, Germany (1998); Cannes, France (1998, 1997); Paderborn, Germany (1996); and Hamburg, Germany (1994). The ASM workshops have had predecessors, e.g., the famous Lipari Summer School in 1993, whose influential outcome was the fundamental Lipari Guide.

The success story of the ASM workshops is based upon a number of advantages of the ASM method for high-level design, analysis, validation, and verification of computing systems:

- The specification method improves industrial practice by proper orchestration of all phases of software development, by supporting high-level modeling at any level of abstraction, and by providing a scientific and formal foundation for systems engineering. All other specification frameworks known so far only provide a loose coupling of notions, techniques, and notations used at various levels of abstraction.
  By using the ASM method, a system engineer can derive a general application-oriented understanding, may base the specification on a uniform algorithmic view, and may refine the model until the implementation level is achieved. The three ingredients needed to achieve this generality are the notion of the ASM itself, the ground-model techniques, and the proper treatment of refinement.
  Using this specification method the system architect, the application engineer, the developer, and the programmer obtain a common view of the system they are building, changing, maintaining, or documenting. The system construction process is accompanied by common understanding, by common theoretical foundations, and by the ability to prove validity of properties such as satisfaction of quality criteria. By doing so the construction process supports quality from the very beginning of the process until the implementation of the system.

At the same time, the ASM method supports software development in changing environments. The method supports abstraction and extension of models, stepwise detailing of models, and control through execution of the model for their experimental validation.

– Abstract state machines entirely capture all four principles of computer science: structure, evolution, collaboration, and abstraction.



This coverage of all principles has not been achieved in any other approach of any other discipline of computer science. Due to this coverage, the ASM method underpins computer science as a whole.

– the ASM method is clearly based on a number of postulates restricting evolution of systems. For instance, sequential computation is based on the postulate of sequential time, the postulate of abstract state, and the postulate of bounded exploration of the state space. These postulates may be extended to postulates for parallel and concurrent computation, e.g., by extending the last postulate to the postulate of finite exploration.

The generality of the model, the deep foundation, the test of the concept by providing rigorous semantics to a large variety of real-life software and hardware products, and – at the same time – the development of proper tools supporting the entire specification process is the work of a community headed by two congenial friends – Yuri Gurevich and Egon Börger – competing at the same time with their results. The former decided to prove the concept in real industrial praxis by building up a group at Microsoft Research, whilst simultaneously deepening the theory of ASM. Beyond further development of the ASM theory, the latter attracted a number of researchers for a program that will entirely change computer science. It will improve development and implementation of languages by providing rigorous semantics and by acquiring tools to prove and to maintain properties of the developed products, and will thus support quality at each level of software specification. The community is growing. The success story can be traced at the website `http://www.eecs.umich.edu/gasm/`.

ASM 2004 contributed to ASM research in several ways:

Extending ASM foundations: ASM research has brought up a good number of difficult and open problems in computer science. This volume contributes by

providing solutions to slicing ASM, intra-step interaction, theory of monodic ASMs, and interchange languages for ASM. Abstract state machines have recently extended to turbo abstract state machines. This abstraction mechanism has already been used in theory of computation. Furthermore, transition theory contributes to foundations of ASM.

Highlighting new application areas: Cryptographic machines, security logics, and service specification are currently hot research areas. As demonstrated in this volume, ASM methods may substantially improve understanding in these areas. Furthermore, the volume shows that even .NET models can be based on ASM.

Tackling problems in already proven application areas: Programming language semantics is based on ASM methods for C# and SSA. UML needs more foundations and ASM methods can be used to formalize UML diagrams, e.g. sequence diagrams. Timed systems are one of the areas where ASM methods have successfully been used. The workshop continues to provide a deeper insight into embedded systems and their semantics. ASM semantics has already been used for exploring database behavior. This workshop provides a deeper view on query processing.

ASM 2004 invited four distinguished researchers to bring other aspects to ASM research. Yuri Gurevich extended the postulates of sequential computation to interaction. Hans-Michael Hanisch showed how ASM research might contribute to research on embedded control systems. Hans Langmaack discussed associations between ALGOL semantics and Turbo ASM introduced recently. Jan Van den Bussche demonstrated how database processing may be based on finite cursor machines.

The LNCS proceedings are accompanied with local proceedings demonstrating recent research in ASM development. These papers demonstrate the development of tools, give an insight into ongoing projects, and provide results on ASM theory that may lead to kernel papers of the next ASM workshop.

We thank the members of the program committee and the additional reviewers for their support in evaluating the papers submitted to ASM 2004. We thank both Springer-Verlag for publishing the proceedings with the invited and research papers in the LNCS series and the Martin-Luther-University Halle-Wittenberg for publishing the proceedings of the short papers. We appreciate the diligent service of the organization team: Thomas Kobienia, Michael Schaarschmidt, and Ramona Vahrenhold. We thank our colleagues and the students of our universities for their help in workshop organization. We thank Martin-Luther-University Halle-Wittenberg Microsoft Research, the Stiftung Leucorea, and the Winzervereinigung Freyburg-Unstrut eG for their support of the workshop. Last, but not least, we thank the participants of ASM 2004 for having made our work useful.

March 2004

Wolf Zimmermann
Bernhard Thalheim

Program Co-chairs

Wolf Zimmermann	(Martin-Luther-University Halle-Wittenberg, Germany)
Bernhard Thalheim	(University of Kiel, Germany)

Program Committee

Egon Börger	(University of Pisa, Italy)
Alessandra Cavarra	(Oxford University, UK)
John Derrick	(University of Kent, UK)
Uwe Glässer	(Simon Fraser University, Canada)
Elvinia Riccobene	(University of Catania, Italy)
Robert Stärk	(ETH Zurüch, Switzerland)
Peter H. Schmitt	(Universität Karlsruhe, Germany)
Anatol Slissenko	(University of Paris, France)
Bernhard Thalheim	(Kiel University, Germany)
Margus Veanes	(Microsoft Research, Redmond, USA)
Wolf Zimmermann	(Universität Halle-Wittenberg, Germany)

Additional Reviewers

E. Asarin	D. Beauquier	P. Caspi
P. Cegielski	A. Chou	A. Cisternino
J. Cohen	R. Farahbod	N. G. Fruja
A. Gargantini	R. Ge	S. Graf
T. Gross	Y. Gurevich	J. Jacky
M. Kardos	L. Nachmanson	S. Nanchen
A. Prinz	H. Rust	P. Scandurra
M. Schaarschmidt	W. Schulte	M. Vajihollahi

Organization Committee

Wolf Zimmermann	University Halle-Wittenberg
Ramona Vahrenhold	University Halle-Wittenberg
Michael Schaarschmidt	University Halle-Wittenberg
Thomas Kobienia	Cottbus University of Technology

Sponsoring Institutions of the ASM 2004 Conference

Martin-Luther-University Halle-Wittenberg, Germany
Microsoft Research, Redmond, WA, USA
Stiftung Leucorea, Germany
Winzervereinigung Freyburg-Unstrut eG, Germany

Volume Editors

Wolf Zimmermann
Martin-Luther-University Halle-Wittenberg, Germany
Department of Mathematics and Computer Science
Institute of Computer Science
von-Seckendorff-Platz 1, D-06099 Halle/Saale
E-mail: zimmer@informatik.uni-halle.de

Bernhard Thalheim
Kiel University
Institute of Computer Science and Applied Mathematics
Olshausenstr. 40, D-24098 Kiel, Germany
E-mail: thalheim@is.informatik.uni-kiel.de