Journal of Universal Computer Science, vol. 3, no. 4 (1997), 230-232
submitted: 20/12/96, accepted: 21/4/97, appeared: 28/4/97 © Springer Pub. Co.

Ten Years of Gurevich's Abstract State Machines.

Egon Börger
(Università di Pisa, Italy
boerger@di.unipi.it)

Abstract State Machines have come quite a way since they have been discovered by Yuri Gurevich in an attempt to improve on Turing's thesis so that:

> Every algorithm can be simulated in lock–step on its natural abstraction level by an ASM.

The first years after the definition of ASMs appeared in the literature (under the name of dynamic or evolving algebras, see [Gurevich 1988]) were characterized by an impressive number of challenging experiments to test this bold thesis through complex real-life case studies, ranging from semantics and implementation of full fledged programming languages to specification and verification of protocols and hardware designs. The first full definition and careful motivation for the new notion appeared in 1991 in the Bulletin of the EATCS and was completed four years later by a solid foundation and an extension to distributed ASMs (see [Gurevich 1995]). The first international ASM workshop—held as part of the IFIP 1994 World Computer Congress in Hamburg with an unexpectedly great number of participants—has marked the end of this rather successful exploratory phase. The interested reader can find a detailed overview in the ASM chapter of the congress proceedings (see [Pehrson and Simon 1994]) and through the annotated bibliography which covers the ASM literature completely up to the end of 1994 (see [Börger 1995a]). In a relatively short time the ASM approach to specification and verification of complex hw/sw systems has proved its feasability for real-life problems and is now establishing its feasability to applications under industrial constraints.

The flourishing ASM research is steadily growing as is documented in this special J.UCS issue which is devoted entirely to ASMs and came into life through a call for papers sent out last year. The great number of submissions which passed the reviewing process forced us to split the special ASM issue into two parts, one issued in April and one in May. The papers in this issue deal with foundational questions, with questions from complexity theory, with the central notion of refinement and with machine support for reasoning about ASMs.

The paper on *Recursive Abstract State Machines* by Gurevich and Spielmann presents an interesting use of distributed ASMs to interpret recursive calls as creating slaves of the calling program. Thus the usual stack which is hidden by the very concept of recursion and is used only for its implementation is reflected (although hidden) in the the master/slave hierarchy of the distributed ASM. This construction allows one to view recursive notation as mere abbreviation and thereby to combine the convenience of the use of recursion with the advantage of the simple semantical foundation of the basic ASM notion which deliberately has no built-in concept of sequencing, loop or recursion.

Two papers deal with complexity theoretic questions. The paper on *The Linear Time Hierarchy* by Blass and Gurevich suggests a new approach for attempts to prove lower bounds for natural linear time problems. It establishes a linear time hierarchy theorem for a wide class of ASMs—namely sequential ASMs without external functions—which includes RAMs with the usual arithmetic operations and Schönhage's storage modification machines. The latter machines are characterized in the paper on *Gurevich Abstract State Machines and Schönhage Storage Modification Machines* by Dexter, Doyle and Gurevich as unary sequential ASM without external functions (via ASMs lock-step equivalence). The ASM approach to linear time hierarchy theorems mentioned above reflects the experience of practical computing that a realistic complexity measure has to take the underlying data structure into account; it brings hope for a new complexity theory which may be capable of having more impact on real computing.

The next three papers are centered around the use of refinement techniques for ASMs. I have explained elsewhere (see [Börger 1995]) why Gurevich's notion of ASM offers the optimal combination of abstraction and refinement features which have been investigated for decades now. This is confirmed once more by the three papers in this volume which are devoted to refinement techniques. The paper *The constrained shortest path problem: A case study in using ASMs* by Stroetmann defines an abstract (nondeterministic) algorithm for the constrained shortest path problem and proves its correctness from a small number of natural axioms; it then defines a sequence of natural refinements which are proved to be correct and lead to an efficient C^{++} program. This illustrates very nicely the advantage a programmer can take from developing programs through sequences of stepwise refined (and possibly provably correct) ASMs leading to executable code which is well documented and inspectable by mathematical means.

The paper *Formalizing Database Recovery* by Gurevich, Soparkar, Wallace provides ASM definitions of a standard database recovery algorithm at various levels of abstraction, related by stepwise refinements which are proved to be correct. This provides a method for systematic and controllable development of validated database recovery algorithms.

The paper *A structured presentation of a closure-based compilation method for a scoping notion in logic* by Kwon deals with an extension of ASM refinement techniques I have developed together with Dean Rosenzweig for the implementation of PROLOG on Warren's abstract prolog machine, known under the name of WAM (see [Börger and Rosenzweig 1994]). Kwon's paper provides a systematic reconstruction of a compilation method for an extension of logic programming where clauses can be parameterized by binding some of their variables. It starts with defining an ASM interpreter for the programming language under consideration and then refines it stepwise until an interpreter is reached that uses the intended (closure) representation for clauses. The mathematical nature of the ASM models permits the author to accompany the refinement steps by proofs of their correctness.

The paper *Reasoning about Abstract State Machines: The WAM Case Study* by Schellhorn and Ahrendt reports on the success of a project investigating whether computer checking of mathematical ASM correctness proofs is feasible. The authors were not afraid to accept a challenging case study for their experiment, namely the by no means trivial correctness proof for a compilation scheme of Prolog programs to WAM code (see [Börger and Rosenzweig 1994]) which has been established by a) modeling PROLOG and the WAM by ASMs and b) linking the two through a dozen of refinement steps and proving each refinement step to be correct. The underlying interactive theorem prover uses dynamic logic in which ASMs are formalized; this includes, as major novelty for theorem provers, a translation of the rather general form of the commuting diagram proof technique used in [Börger and Rosenzweig 1994] where it is crucial that m abstract computation steps may be simulated by n refined computation steps without any a priori given bound on either m or n.

We hope the reader will benefit from the ASM papers appearing here and will be interested int the second part of the special ASM issue (appearing here in May) which will contain applications of ASMs to software engineering problems.

References

[Börger 1995] E. Börger: "Why Use Evolving Algebras for Hardware and Software Engineering?", M.Bartosek, J.Staudek, J.Wiedermann (Eds), SOFSEM'95 (22nd Seminar on Current Trends in Theory and Practice of Informatics); Springer LNCS 1012 (1995), 236–271.

[Börger 1995a] E. Börger: "Annotated Bibliography on Evolving Algebras"; E. Börger (ed.), Specification and Validation Methods, Oxford University Press (1995) 37-51.

[Börger and Rosenzweig 1994] E. Börger and D. Rosenzweig: "The WAM Definition and Compiler Correctness"; C.Beierle, L.Plümmer (Eds.), Logic Programming: Formal Methods and Practical Applications, North-Holland, Series in Computer Science and Artificial Intelligence (1994), 20-90

[Gurevich 1988] Y. Gurevich: "Logic and the challenge of computer science." E. Börger (ed.), Current Trends in Theoretical Computer Science. CS Press (1988) 1-57.

[Gurevich 1995] Y. Gurevich: "Evolving Algebras 1993: Lipari Guide"; E. Börger (ed.), Specification and Validation Methods, Oxford University Press (1995) 9-36.

[Pehrson and Simon 1994] B. Pehrson and I. Simon (Eds.), IFIP 13th World Computer Congress 1994, Volume I: Technology/Foundations, Stream C: Evolving Algebras, Elsevier, Amsterdam (1994), 377-444.

[Schellhorn and Ahrendt 1997] G. Schellhorn and W. Ahrendt: "Reasoning about Abstract State Machines: The WAM Case Study"; J.UCS (Journal for Universal Computer Science) 3,4 (1997).