77 (1-2) 2007

1. **Preface** **i-iv**

2. **Refinement, Decomposition, and Instantiation of Discrete Models: Application to Event-B**

Jean-Raymond Abrial, Stefan Hallerstede *1-28*

We argue that formal modeling should be the starting point for any serious development of computer systems. This claim poses a challenge for modeling: at first it must cope with the constraints and scale of serious developments. Only then it is a suitable starting point. We present three techniques, refinement, decomposition, and instantiation, that we consider indispensable for modeling large and complex systems. The vehicle of our presentation is Event-B, but the techniques themselves do not depend on it.

3. **Retrenching the Purse: The Balance Enquiry Quandary, and Generalised and (1, 1) Forward Refinements**

Richard Banach, Czeslaw Jeske, Michael Poppleton, Susan Stepney *29-69*

Some of the success stories of model based refinement are recalled, as well as some of the annoyances that arise when refinement is deployed in the engineering of large systems. The way that retrenchment attempts to alleviate such inconveniences is briefly reviewed. The Mondex Electronic Purse formal development provides a highly credible testbed for examining how real world refinement difficulties can be treated via retrenchment. The contributions of retrenchment to integrating the real implementation with the formal development are surveyed, and the extraction of commonly occurring 'retrenchment patterns' is recalled. One of the Mondex difficulties, the 'Balance Enquiry Quandary' is treated in detail, and the way that retrenchment is able to account for the system behaviour is explained. The problem is reconsidered using generalised forward refinement, and the simplicity of the resolution of the quandary, both by retrenchment, and by generalised forward refinement, inspires the creation of a genuine $(1, 1)$ forward refinement for Mondex, something long thought impossible. The forward treatment exhibits a similar balance enquiry quandary to the backward refinement, as it must, given that both are refinements of an atomic action to a non-atomic protocol, and the forward quandary is dealt with as easily by retrenchment as is the backward case. The simplicity of the retrenchment treatment foreshadows a general purpose retrenchment *Atomicity Pattern* for dealing with atomic-versus-finegrained situations.

4. CoreASM**: An Extensible ASM Execution Engine**

Roozbeh Farahbod, Vincenzo Gervasi, Uwe Glässer *71-103*

In this paper we introduce a new research effort in making *abstract state machines* (ASMs) executable. The aim is to specify and implement an execution engine for a language that is as close as possible to the mathematical definition of pure ASMs. The paper presents the general architecture of the engine, together with a high-level description of the extensibility mechanisms that are used by the engine to accommodate arbitrary backgrounds, scheduling policies, and new rule forms.

5. **Model Checking Abstract State Machines with Answer Set Programming**

Calvin Kai Fan Tang, Eugenia Ternovska *105-141*

The quality of a computer system can be enhanced by modelling its design and verifying the correctness of the design before implementation is done. Abstract State Machines (ASMs) provide a mathematical framework for system modelling, while Model Checking is a technology for verification of system properties. Together, they form a powerful tool for checking systems. Bounded Model Checking (BMC) based on Answer Set Programming (ASP) is a competitive model checking approach due to its ability to compactly encode BMC problems. In this paper, we present a method of applying ASP to BMC of ASMs. Given an ASM and a temporal property, we show how to efficiently translate the BMC problem for the ASM into a problem of answer set computation. Experimental results for our method using the answer set solvers SMODELS and CMODELS are also given.

6. **Time in State Machines**

Susanne Graf, Andreas Prinz *143-174*

State machines are a very general means to express computations in an implementation-independent way. There are ways to extend the abstract state machine (ASM) framework with distribution aspects, but there is no unifying framework for handling time so far.

We propose event structures extended with time as a natural framework for representing state machines and their true concurrency model at a semantic level and for discussing associated time models. Constraints on these timed event structures and their traces (runs) are then used for characterising different frameworks for timed computations. This characterisation of timed frameworks is independent of ASM and allows to compare time models of different modelling formalisms.

Finally, we propose some specific extensions of ASM for the expressions of time constraints in accordance with the event-based semantic framework and show the applicability of the obtained framework on an example with a standard time model and a set of consistency properties for timed computations.

7. **RAM Simulation of BGS Model of Abstract-state Machines**

Comandur Seshadhri, Anil Seth, Somenath Biswas *175-185*

We show in this paper that the BGS model of abstract state machines can be simulated by random access machines with at most a polynomial time overhead. This result is already stated in [5] with a very brief proof sketch. The present paper gives a detailed proof of the result. We represent hereditarily finite sets, which are the typical BGS ASM objects, by membership graphs of the transitive closure of the sets. Testing for equality between BGS objects can be done in linear time in our representation.