Experiments in program verification using Event-B

Stefan Hallerstede and Michael Leuschel
Institut für Informatik, University of Düsseldorf, Universitätsstraße 1, 40225 Düsseldorf, Germany. E-mail: stefan.hallerstede@wanadoo.fr

Abstract. The Event-B method can be used to model all sorts of discrete event systems, among them sequential programs. In this article we describe our experiences with using Event-B by way of two examples. We present a simple model of a factorial program, explaining the method, and a more intricate model of the Quicksort algorithm, providing some insights into strengths and weaknesses of Event-B. The two models are interspersed with our observations and some suggestions of how, we believe, Event-B could evolve. This evaluation of Event-B is intended to serve for determining directions for the evolution of Event-B and judging progress. It is our hope that the observations and suggestions can also be put to use for similar modelling formalisms, such as Z, ASM or VDM.

Keywords: Event-B, Proof Obligation, Proof, Sequential Program Verification

1. Introduction

Event-B [Abr10] is a formalism and method for discrete systems modelling. It has been developed from the B-Method [Abr96] using many ideas of Action Systems [Bac89]. It admits various modelling domains, for instance, reactive, distributed and concurrent systems, sequential programs, digital circuits, and mixed designs, not being restricted to semantics tailored to a particular domain. The notation of Event-B has been simplified [Hal07] compared to the B-Method. Most of the structuring mechanisms and all control structures have been eliminated so as to improve reasoning facilities of the method. This is achieved by means of a closer correspondence between proof obligations and formal models. At the same time the amount of proof obligations that can be discharged automatically is increased.

This study is intended to stimulate the further development of Event-B and provide a means to measure progress of the method. We think it can also provide valuable input for developments in other state-based formal methods. An evaluation seems necessary, now that Event-B has been around for some time and a corresponding tool has been developed [ABHV06]. This article also aims to document the evolution of the method, taking a snapshot before its evolution continues. Many of the points we make are of a more technical nature. We deem this appropriate as we show the method “at work”. The actual models we present in this article serve only to convey our findings. They may be of interest by themselves, but this is not our main concern.

Correspondence and offprint requests to: S. Hallerstede. E-mail: stefan.hallerstede@wanadoo.fr
1.1. Scope of the study

In this article we focus on sequential program development, the domain of the B-Method, the predecessor of Event-B. Verification of sequential programs can either be done directly [AdBO09b] or by refinement [Mor94]. Both methods yield the same result—a verified algorithm. We believe the refinement-based approach makes it easier to master the complexity of more intricate algorithms. Direct attempts at the verification of sequential programs have to deal with all details of an implementation at once. Using refinement, one can proceed more slowly, introducing necessary details gradually. Step by step we discharge proof obligations that, taken together, establish correctness of a sequential program with respect to an abstract specification.

Event-B [Abr10] can be used to verify (models of) sequential programs [Abr03, Abr10, Mér09, TBL10]. Using the Rodin tool [ABHV06] proof obligations can be generated automatically and updated automatically whenever we change a model. As much as possible, proofs associated with a model are maintained with their corresponding proof obligations. During the design of Event-B we have taken great care to justify design choices concerning the modelling method [Hal07]. We believe Event-B and the Rodin tool present a significant advance over classical refinement-based verification approaches like the B-Method [Abr96]. Still, some of the choices made come at a price. In this article we evaluate some aspects of sequential program development in Event-B, putting the claim that we have made progress to the test.

1.2. Contribution

Although this article discusses methodological problems of using Event-B, we think that the discussion and the discussed techniques are of a more general interest. Similar problems appear in other methods such as VDM [Jon90] or Circus [SWC02] and their associated tools. For instance, the technique of decomposing proof obligations by means of witnesses is also well-applicable to Circus [WC01]. The approach usually followed in Circus is to document heuristics of refinement and proof, whereas in Event-B we try to implement very useful heuristics in the Rodin tool. For the Event-B method to be useful for sequential program development in practice, it should help in developing a sequential program and it should systematically lead to a sequential program. We have achieved progress if some tasks have become easier to carry out or we can address a larger class of problems.

Our observations should also be of interest for other state-based formal modelling methods, such as the B-Method [Abr96], Z [Spr92, WD96], Action Systems [BKS88], ASM [BS03], VDM [Jon90] or Morgan’s Refinement Calculus [Mor94]. In particular, with respect to the B-Method there should be measurable progress, as Event-B was derived from the B-Method in order to improve possible (tool-) support for reasoning about models. Z and also Object-Z [Smi00] remain at a higher abstraction level. They do not produce sequential programs, but detailed specifications of sequential programs corresponding more closely to the point of departure of our study in this article. With respect to sequential program development, abstract state machines (ASM) are similar to the B-Method in that they use programming notation for implementation and leave the user with more difficult proof obligations. The same holds for VDM and the refinement calculus. Action Systems are not intended for sequential program development and appear to be used more in theoretical studies rather than in practical applications. They have full support for monotonic predicate transformers and, correspondingly, more complicated proof obligations. Event-B has evolved from the B-Method and Action Systems, combining the possibility of strong tool support of the B-Method with a potentially simpler notion of refinement of Action Systems. Simplifying for the matter of the argument, Event-B restricts Action Systems to universally conjunctive predicate transformers and makes some concessions for tool support similar to the B-Method.