A model advisor for NuSMV specifications

Paolo Arcaini · Angelo Gargantini · Elvinia Riccobene

Received: 13 September 2010 / Accepted: 26 February 2011 / Published online: 12 March 2011 © Springer-Verlag London Limited 2011

Abstract Among possible model validation techniques able to identify defects early in the system development, model review aims also at determining if a model is of sufficient quality, where quality is measured as the absence of certain faults. In this paper, we tackle the problem of automatic reviewing NuSMV formal specifications by developing a model advisor which helps to assure given model qualities for NuSMV programs. Vulnerabilities and defects a developer can introduce during the modeling activity using NuSMV are expressed as the violation of formal meta-properties. These meta-properties are then mapped to temporal logic formulas, and the NuSMV model checker itself is used as the engine of our model advisor to notify meta-properties violations, so revealing the absence of some quality attributes of the specification. As a proof of concept, we also report the result of applying this review process to several NuSMV specifications.

Keywords Model advisor · Model review · Model checking · NuSMV

1 Introduction

System validation is an essential activity of any development process since it permits detecting faults and assuring that given requirements are guaranteed by the system under development. Nowadays, the use of formal methods, based on rigorous mathematical foundations, is becoming extremely important for system design and development, since abstract formal models allow detecting faults in the specification as early as possible with limited effort. Model validation should precede the application of more expensive and accurate verification methods that should be applied only when a designer has enough confidence that the specification really reflects the user perceptions. Otherwise (right) properties could be proved true for a wrong specification.

Among validation techniques, model review, also known as “model walk-through” or “model inspection”, allows to critically examine modeling efforts to determine if a model not only fulfills the intended requirements, but also is of sufficient quality to be easy to develop, maintain, and enhance. This process should, therefore, assure a certain degree of quality. The assurance of quality, namely ensuring readability and avoiding error-prone constructs, is one of the most essential aspects in the development of safety critical reactive systems, since the failure of such systems—often attributable to modeling and, therefore, coding flaws—can cause loss of property or even human life [20]. When model reviews are performed properly, they can have a big payoff because they allow to identify defects early in the system development, reducing the cost of fixing them. Usually model review, which comes from the code-review idea, is performed by a group of external qualified people, often both technical staff and project stakeholders, who meet together to evaluate models and documents.
A weak aspect of the review process is that it is usually done by hand. This requires a great effort that might be tremendously reduced if performed in an automatic way by systematically checking specifications for known vulnerabilities or defects. The question is what to check on and how to automatically check the model. In other words, it is necessary to identify classes of faults and defects to check, and to establish a process by which to detect such deficiencies in the model. If these faults are expressed in terms of formal statements, these can be assumed as a sort of “measure” of the model quality assurance. A tool is also necessary to make the process automatic. It would work as model advisor to check a model for conditions and configuration settings that can result in inaccurate or under-/over-specified behavior of the system that the model represents.

In this paper, we tackle the problem of automatically reviewing NuSMV [1,11] formal specifications. We develop a model advisor for NuSMV programs which helps the developers to assure given model qualities. We first detect a family of vulnerabilities and defects a developer can introduce during the modeling activity using NuSMV and we express such faults as the violation of formal properties. These properties refer to model attributes and characteristics that should hold in any NuSMV model, independently from the particular model to analyze. For this reason they are called meta-properties. They should be true in order for a NuSMV model to have the required quality attributes. Therefore, they can be assumed as measures of model quality assurance. Depending on the meta-property, its violation indicates the presence of actual faults, or only of potential faults.

These meta-properties are defined in terms of logical operators here defined, Always, Sometime, InitiallyA/InitiallyS, to capture properties that must be true in every state, eventually true in at least one state, or referring to initial states (always or sometimes true) of the NuSMV model under analysis. Then, we map these logical operators to temporal logic formulas and we exploit the NuSMV model checking facilities to check for meta-property violations.

A similar model reviewing technique was presented in [6] for the Abstract State Machine (ASM) formal specifications [9]. We here refer to NuSMV formal specification method which is endowed with a simulator and a model checker that make possible to handle and to automatize our approach. There exists a wide repository of NuSMV specification case studies available for testing the model advisor we propose. In any case, the results obtained for the ASM and NuSMV models can be adapted to other state-transition-based formal approaches.

The NuSMV formal method and the structure of a formal specification are briefly presented in Sect. 2. Section 3 defines a function, later used in the meta-properties definition, that statically computes the assignment condition under which a model variable is updated upon state changing. Meta-properties able to guarantee certain quality attributes of a specification are introduced in Sect. 4. In Sect. 5, we describe how it is possible to automatize our model review process by exploiting the use of NuSMV itself as a model checker to check for possible violations of meta-properties. The general architecture of our model advisor is described in Sect. 6. As a proof of concept, in Sect. 7, we report the results of applying our model advisor to a certain number of benchmark models of various degree of complexity: some taken from the NuSMV source distribution, others found on the Internet, others obtained from ASM models of real case studies. We also discuss the fault detection capability of our analysis. In Sect. 8, we present other works related to the model review process. Section 9 concludes the paper and indicates some future directions of this work.

2 NuSMV

NuSMV [1] is known as a model checker derived from the CMU SMV [17]. It allows for the representation of synchronous and asynchronous finite state systems, and for the analysis of specifications expressed in Computation Tree Logic (CTL) and Linear Temporal Logic (LTL), using Binary Decision Diagrams (BDD)-based and SAT-based model checking techniques. Heuristics are available for achieving efficiency and partially controlling the state explosion.

A NuSMV specification describes the behavior of a Finite State Machine (FSM) in terms of a “possible next state” relation between states that are determined by the values of variables. Transitions between states are determined by the updates of the variables. According to the model operational description, a NuSMV specification is made of three principal sections:

- **VAR** that contains variable declarations. A variable type can be boolean, integer defined over intervals or sets, or an enumeration of symbolic constants.
- **ASSIGN** that contains the initialization (by the instruction init) and the update mechanism (by the instruction next) of variables. A variable cannot be initialized and in this case, at the beginning NuSMV creates as many initial states as the number of values of the variable type; in each state the variable assumes a different value.
- **CTLSPEC** (resp. LTLSPEC) that contains the CTL (resp. LTL) properties to be verified.

A **DEFINE** statement can also be used as a macro to syntactically replace an identifier with the expression it is associated with. The associated expression is always evaluated in the context of the statement where the identifier is declared.