NuSeen: an eclipse-based environment for the NuSMV model checker

Paolo Arcaini
CNR – IDPA
paolo.arcaini@idpa.cnr.it

Angelo Gargantini
University of Bergamo
angelo.gargantini@unibg.it

Paolo Vavassori
University of Bergamo
paolo.vavassori@unibg.it

1. INTRODUCTION

NuSMV [5] is a symbolic model checker originated from the reengineering, reimplementation and extension of CMU SMV, the original BDD-based model checker developed at CMU by McMillan [4]. The NuSMV project aims at the development of a state-of-the-art symbolic model checker, designed to be applicable in technology transfer projects: it is a well structured, open, flexible and documented platform for model checking, and is robust and close to industrial systems standards [3].

NuSMV has a rich and powerful language that can be used to describe complex systems, which contain the specification of the system behavior as Finite State Machines and its expected requirements (often given by temporal formula).

It can also be used as a model checker, both as a BDD-based symbolic model checker, and as a Bounded model checker. It has around 50 options when it is called in a batch mode. It is widely used as back end for the verification of properties of systems given by means of other formal notations (as for the Abstract State Machines in [1]).

In [2], we have developed a model advisor for NuSMV models. The model advisor performs automatic review of NuSMV models, with the aim of determining if a model is of sufficient quality, where quality is measured as the absence of certain faults. 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.

2. NUSEEN: A NUSMV ECLIPSE-BASED ENVIRONMENT

NuSeen is an eclipse-based environment for NuSMV, with the aim of helping NuSMV users. It mainly focuses in easing the use of the NuSMV tool by means of graphical elements like buttons, menu, text highlighting, and so on. It features:

- A *language* defined by a grammar (concrete syntax) and provided with metamodel (abstract syntax)
- An *editor* that can be used to write NuSMV models and provides an useful feedback like syntax highlighting, autocompletion, and outline.
- A way to *execute* the NuSMV model checker inside eclipse.
- An integrated version of the model advisor which can be executed in eclipse.

2.1 Language and editor

NuSeen introduces the definition of the language and of the editor for NuSMV by Xtext [6]. Xtext is a framework for development of programming languages and domain specific languages. It covers all aspects of a complete language infrastructure, from parsers, over linker, compiler or interpreter to fully-blown top-notch Eclipse IDE integration. It comes with good defaults for all these aspects and at the same time every single aspect can be tailored to your needs. We have defined a grammar for the NuSMV notation in the Xtext format. From that, we automatically obtain:

1. a *meta-model* for the language in the form of an EMF (eclipse metamodeling framework) model in the ecore format. It represents the *abstract* syntax of our language in terms of classes and relations. From the meta-model, we automatically obtain the Java APIs that are able to manage (query, create, and modify) NuSMV models and parts of them. This allows the integration of NuSMV with other Java-based tools.

2. an *editor* with *syntax coloring* based on the lexical structure, *content assist* that proposes valid code completions at any place in the document, helping users with the syntactical details of the language, *validation and quick fixes*, linking errors, the outline view, find references, etc.

The editor can be easily extended in order to support extra semantics validation rules, particular rules for indentation and outlining, and other ad hoc editing rules.

2.2 NuSMV executor

NuSeen allows the user to run NuSMV from within eclipse. We exploit the *launching framework* in Eclipse. We have performed the following steps:

1. The first step consists in declaring a config type for NuSMV models, by extending the non-ui extension point `launchConfigurationTypes` in the (non-ui) package `org.eclipse.debug.core`.

2. The extension is implemented in a class that actually executes NuSMV with the model chosen by the user.

3. We then prepare the launch configuration dialog (often called LCD) by defining an extension for the extension

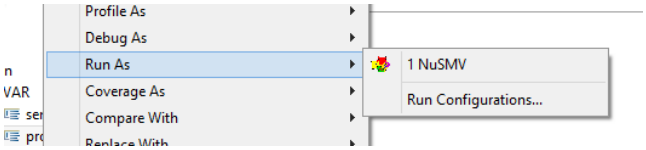

Figure 1: The launch shortcut for NuSMV

point `launchConfigurationTabGroups` in the package `org.eclipse.debug.ui`. It contains the configuration with all the options a user can select when launching NuSMV.

4. We than declare and implement a launch shortcut that allows users to identify a resource in the workbench (either via selection or the active editor) and launch that resource with a single click without bringing up the launch configuration dialog. An example of its use is shown in Fig. 1.

2.3 Model Advisor

The model advisor it analyzes a NuSMV specification using NuSMV itself. For a detailed theoretical motivation of the kind of review performed by the model advisor, please see [2]. The NuSMV models are read and queried by the model advisor by using the API generated from the metamodel introduced by the language component. The model adivsor runs NuSMV using the APIs defined in the executor. To run the model advisor from the eclipse UI, we exploit again the launching framework of eclipse. We

3. ARCHITECTURE

NuSeen is divided in three main components, as shown in Fig. 2, each divided in one or more plugin eclipse projects:

1. The component containing the definition of the language and the editor (dsl by xtext). It is divided into two plugins: one contains the definition of the grammar and the other the classes for the editor.

2. The component containing the runner, which simply introduces the launching framework for the execution of NuSMV.

3. The model advisor which contains the model advisor itself, its UI part in order to ingrate it within eclipse and NuSeen, and a third project which allows the user to call the model advisor form outside eclipse. This latter plugin wraps everything needed by the model advisor in an executable jar file.

4. FUTURE WORK

We plan to maintain and extend NuSeen in the following directions. About the language, we would like to test it and to add more precise semantic rules. Up to now, the editor can still consider correct files that are actually invalid according to the real NuSMV semantics and that can be checked only by the NuSMV parser. We plan to integrate the execution of NuSMV inside eclipse, by releasing the user from the burden of installing and configuring NuSMV. Up to now, we assume that the user has already installed NuSMV

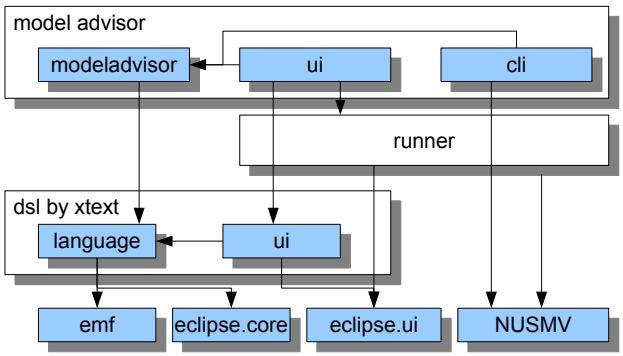

Figure 2: Architecture

(which must be in the path). We plan to use technologies like JNA to ship NuSMV together with NuSeen.

The feedback given by the NuSMV parser and by the model advisor is simply shown in the console. We plan to read possible errors and warnings found by NuSMV and by the advisor and show them directly in the editor by using appropriate markers.

5. ACKNOWLEDGEMENTS

We would like to thank Siamak Haschemi for the initial version of the XTEXT for NuSMV grammar.

6. REFERENCES

[1] P. Arcaini, A. Gargantini, and E. Riccobene. AsmetaSMV: a way to link high-level ASM models to low-level NuSMV specifications. In M. Frappier, U. Glässer, S. Khurshid, R. Laleau, and S. Reeves, editors, *ABZ 2010*, volume 5977 of *Lecture Notes in Computer Science*, pages 61–74. Springer, 2010.

[2] P. Arcaini, A. Gargantini, and E. Riccobene. A model advisor for NuSMV specifications. *Innovations in Systems and Software Engineering*, 7:97–107, 2011.

[3] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. NUSMV: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer (STTT)*, 2(4):410–425, Mar. 2000.

[4] K. L. McMillan. The SMV system, symbolic model checking - an approach. Technical Report CMU-CS-92-131, Carnegie Mellon University, 1992.

[5] The NuSMV website. `http://nusmv.fbk.eu/`.

[6] Xtext. http://www.eclipse.org/xtext/.