Proceedings of the
Ninth International Workshop on
Automated Verification of Critical Systems
(AVOCS 2009)

A Semantic Embedding of Promela-Lite in PVS

Shamim H. Ripon, Alice Miller and Alastair F. Donaldson

3 pages
A Semantic Embedding of Promela-Lite in PVS

Shamim H. Ripon\textsuperscript{1}, Alice Miller\textsuperscript{2} and Alastair F. Donaldson\textsuperscript{3}

\textsuperscript{1}shamim@dcs.gla.ac.uk \textsuperscript{2}alice@dcs.gla.ac.uk
Department of Computing Science, University of Glasgow, UK

\textsuperscript{3}alastair.donaldson@comlab.ox.ac.uk
Computing Laboratory, University of Oxford, UK

Abstract: We present a case study in applying mechanical verification via theorem proving to Promela-Lite. We show how the syntax and semantics of Promela-Lite can be embedded in the general purpose theorem prover PVS, so that consistency of the syntax and semantics can be interactively proved.

Keywords: Promela-Lite, Semantics, PVS, Theorem Proving, Symmetry.

1 Introduction

Promela-Lite [DM08] is a specification language that captures the core features of Promela - the input language for the SPIN model checker [Hol03]. Unlike Promela, Promela-Lite has a rigorously defined semantics, making it a suitable vehicle for proving correctness of verification and state-space reduction techniques for Promela. The language was designed for the purpose of proving correct an automatic symmetry detection technique for Promela [DM08], which derives state-space preserving automorphisms from the text of a Promela specification, to be exploited during search to reduce the space and time requirements of model checking via symmetry reduction [CEF+96, ES96, ID96]. This symmetry detection technique, based on static channel diagram analysis, has been implemented as part of TopSPIN, a symmetry reduction package for the SPIN model checker [DM06].

The soundness of model checking depends critically on the correctness of underlying algorithms and reduction techniques. For example, an erroneous symmetry detection method may compute state-space permutations which are not structure-preserving, potentially resulting in incorrect verification results. For this reason, it is highly desirable that correctness proofs for model checking techniques, such as the proof by hand presented in [DM08], are automatically verified.

Mechanical verification is widely used as a tool to verify the syntax and the semantic models of a language. Verification of language properties may identify flaws in the language, which can be remedied to give increased confidence in the language definition. Theorem provers are heavily used as a tool to mechanically verify language properties. PVS [ORS92] is an automated framework for specification and verification. It supports high-order logic, allows abstract datatypes to model process terms, and has a strong support for induction mechanisms. PVS also supports interactive proof checking, where the user applies proof commands to simplify the goal to be proved until it can be proved automatically by its decision procedure.

In this paper, inspired by other successful attempts to embed specification languages into PVS [POS04, RB08], we show how the Promela-Lite syntax and semantics can be embedded
into PVS, and how this embedding is used to interactively prove both consistency of the syntax/semantics definitions, and language properties. In particular, we concentrate on proving theorems related to automatic symmetry detection which have previously been proved only by hand.

2 Embedding Promela-Lite in PVS

Promela-Lite is embedded into PVS in a step by step manner starting from the definition of types, syntax and semantics, and extending to the definition of various properties and theorems of the language.

The encoding of a language in PVS requires one to define the available types of the language. We first define the primitive types int and pid, where int represents the integer values and pid denotes the process id values. Both types are defined as a subtype of natural numbers. A channel type in Promela-Lite can accept any other types including a channel type itself as its parameter, which results in a recursive type for channels. The DATATYPE mechanism of PVS is used to define Promela-Lite types as a datatype in PVS.

Proofs about languages which have a BNF style syntax definition often require induction over the terms of the language. PVS provides a powerful mechanism to define abstract datatypes and it generates an induction scheme for them. The BNF formed Promela-Lite syntax is defined in PVS using an abstract datatype. When these definitions are type checked, PVS generates new files containing a large number of useful definitions and properties of the datatypes, as well as an induction scheme.

Typing rules for Promela-Lite are defined to ensure that the language terms are well-formed. These are also important while proving language properties. The Promela-Lite semantics are defined in PVS in such a way to ensure that the definitions preserve the typing rules. The definition of the semantics also requires the notion of a state of a specification. A state can be represented as an ordered tuple consisting of current values of each variable and channel, or as a set of propositions. In PVS we represent a state as a set of variables, and values, with a mapping from variables to values which allows us to identify the current value of any variable at a state. An important part of our semantic definition is the update rules which include the rules for assignments and for reading from and writing to channels. By repeatedly applying the update rules we can define a sequence of updates.

3 Proving Language Properties

After defining the syntax and semantics of Promela-Lite, we prove some theorems and supporting lemmas which have been previously proved by hand. The static channel diagram associated with a Promela-Lite specification $P$ is a coloured, bipartite digraph $SCD(P) = (V, E, Y)$. The group of all automorphisms of $SCD(P)$ is denoted by $Aut(SCD(P))$. We can find this group easily, using the computational group theory package, GAP [gap06].

Our ultimate goal is to prove that a subgroup $G$ of $Aut(SCD(P))$, known as the valid automorphisms of $P$, is an automorphism group of $M$, the underlying Kripke-structure of $P$. The proof of this relationship depends on various underlying definitions and supporting lemmas.
One of the main hurdles that needs to be overcome to automate the proof steps is to identify all the supporting rules and definitions that are required. We do this by extracting definitions and lemmas from the published hand proofs. The major challenges are to define:

- permutations for expressions, guards and update statements. Separate permutation rules are needed for expressions based on the types of returned values when the expressions are evaluated. Inductive definitions are needed for guards.
- permutations of Promela-Lite channels (recursive definition).
- supporting lemmas for expressions, guards and update statements. The proof of each lemma extensively uses various definitions and permutation rules. The challenge here is to apply the rules and definitions appropriately in the proof steps. Most of these rules are not defined separately in the hand proof. Identifying these definitions requires a deep understanding of the semantics and the proof steps. We have defined permutations for expressions and guards, and completed proofs of the corresponding lemmas. We are currently proving these lemmas for update statements.

Bibliography


