Comparative analysis of model checking tools

Boyarkin Nikita
Peter the Great St.Petersburg Polytechnic University
Department of Computer Systems & Software Engineering
Email: portudwar@gmail.com

***Abstract*—This article we analyzes the popular tools for the formal verification of software systems, in addition, we compare them by the entered criteria. For analysis and subsequent comparison, we chose the following tools: Cadence SMV, NuSMV, SPIN, HyTech, UPPAAL and PRISM. The choice was made in favor of general-purpose model checkers, however, for comparison were added two additional highly specialized model checkers, such as HyTech and PRISM.**

Keywords*—*Formal Verification, Model Checking, Cadence SMV, NuSMV, SPIN, HyTech, UPPAAL, PRISM.

I. Introduction

Model checking is a method for formally verifying finite-state concurrent systems. Specifications about the system are expressed as temporal logic formulas, and efficient symbolic algorithms are used to traverse the model defined by the system and check if the specification holds or not [1]

Existing model checking tools differ mainly in their field of application. We divide them into verifiers of hybrid systems, real-time system verifiers, probabilistic system verifiers, and verifiers of large systems.

In addition, an important criterion is the logic for formalizing the requirements for the model. Typically, such logics are temporal LTL and CTL [2] Specific logic such as TCTL, PCTL and others usually follows from the field of application.

We will also consider secondary criteria, such as the availability of a GUI, the type of license, and the possibility of formal proof of theorems.

The method of research that is used in this work is mainly to study the documentation of these tools, the manual check of certain criteria, as well as the structuring information from borrowed articles.

Before the our research we have studied such important researches as "Verification of programs by the method of Model Checking" [1], "Verification of parallel and distributed software systems" [2], "Overview of Approaches to the Verification of Distributed Systems" [3] and others.

II. Model Checking tools analysis

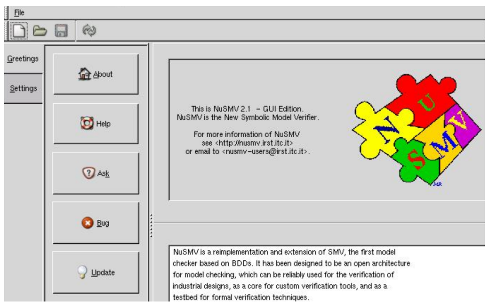
In this section, we consider the main objectives and features of such model checking tools as: Cadence SMV, NuSMV, Spin, HyTech, UPPAAL and PRISM. All the information for comparison we add to the resulting table.

A. *Cadence SMV*

Cadence SMV – a symbolic model-checker developed at Cadence Berkeley Laboratories for Windows and Unix. SMV can be used to teach the general principles of model verification and verification based on refinements [3] SMV is equipped with two modeling languages - extended SMV and synchronous verilog. SMV allows for several forms of specification of requirements that include temporal logics CTL and LTL, state machines, built-in assertions and specification refinements. Supports graphical user interface and generation of counterexamples.

B. *NuSMV*

NuSMV – symbolic model-checker, which is an extension of SMV. In comparison with SMV, it provides such capabilities as finite state machine interaction, invariant analysis, implementation of decomposition methods, LTL and CTL model verification [3] Supports graphical user interface, generation and visualization of counterexamples.



This is NuSMV 2.1 – GUI Edition.
NuSMV is the New Symbolic Model Verifier.

For more information of NuSMV
see <http://nusmv.irst.itc.it>
or email to <nusmv-users@irst.itc.it>.

NuSMV is a reimplementation and extension of SMV, the first model checker based on BDDs. It has been designed to be an open architecture for model checking, which can be reliably used for the verification of industrial designs, as a core for custom verification tools, and as a testbed for formal verification techniques.

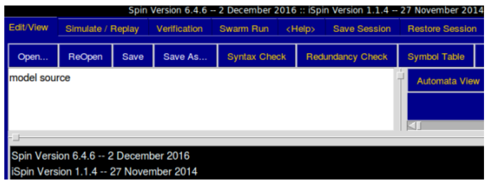
Fig. 1. NuSMV GUI welcome page

C. *Spin*

Spin – tool for formal verification of distributed systems. Developed in Bell Labs in a group at the Computing Sciences Research Center, since 1980 [3] Spin can be used in three main modes:

- simulator, allowing rapid prototyping with random, controlled or interactive modeling;
- verifier, able to strictly prove the validity of the correctness requirements specified by the user;
- a system of reasonable approximation, which is able to confirm the validity of large-scale protocols with the maximum coverage of the state space.

Of the specific features, the C-like syntax of the model description language of Promela should be distinguished. It allows you to describe the code of popular C-like programming languages almost one to one.



```
Spin Version 6.4.6 -- 2 December 2016
iSpin Version 1.1.4 -- 27 November 2014
```

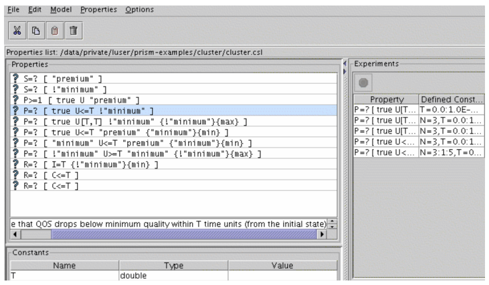
Fig. 2. iSpin the main GUI for Spin

D. PRISM

PRISM – probabilistic symbolic model-checker, developed at the University of Birmingham, for the analysis of probabilistic systems [3] Supports three models:

- discrete-time Markov chains (DTMCs);
- continuous-time Markov chains (CTMCs);
- Markov processes of regulation (MDPs).

Formalized requirements are described by temporal logics CSL and PCTL. Supports graphical user interface.



Properties list: /data/private/luser/prism-examples/cluster/cluster.csl

```
S=? [ "premium" ]
S=? [ !"minimum" ]
P>=1 [ true U "premium" ]
P=? [ true U<=T !"minimum" ]
P=? [ true U[T,T] !"minimum" {!"minimum"}{max} ]
P=? [ true U<=T "premium" {"minimum"}{min} ]
P=? [ "minimum" U<=T "premium" {"minimum"}{min} ]
P=? [ !"minimum" U>=T "minimum" {!"minimum"}{max} ]
R=? [ I=T {!"minimum"}{min} ]
R=? [ C<=T ]
R=? [ C<=T ]
```

e that QOS drops below minimum quality within T time units (from the initial state)

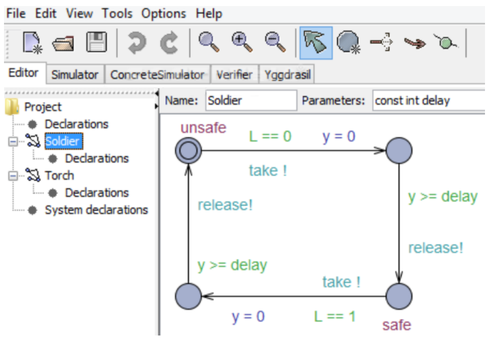
Property	Defined Const...
P=? [true U[T...	T=0.0:1.0E-...
P=? [true U[T...	N=3,T=0.0:1...
P=? [true U[T...	N=3,T=0.0:1...
P=? [true U<...	N=3,T=0.0:1...
P=? [true U<...	N=3:1:5,T=0...

Name	Type	Value
T	double	

Fig. 3. PRISM GUI window

E. UPPAAL

UPPAAL – an integrated environment for the development, validation and verification of real-time systems, modeled as networks of temporary state machines extended with data types [3] Formalized requirements are described by temporal logic TCTL. UPPAAL has a built-in graphical interface and supports generation of counterexamples.



Name: Soldier   Parameters: const int delay

Fig. 4. UPPAAL GUI main window

F. HyTech

HyTech – symbolic model-checker for hybrid systems. To model hybrid systems, we introduce the notion of a hybrid automaton – a finite automaton with a finite number of real variables that change continuously but not at a constant rate, as in a conventional time automaton, but at a rate expressed in terms of differential equations and differential inequalities [3] Formalized requirements are described by temporal logics LTL, TCTL and Monitor automata.

III. Results

The result of our research is expedient to present in the form of a table:

TABLE I. Comparative analysis of model checking tools

Criterion	**Cadence SMV**	**NuSMV**	**UPPAAL**	**HyTech**	**Spin**	**PRISM**
Theorem Proving	+	–	–	–	–	–
Modelling language	SMV, Verilog	SMV	Timed automata	Hybrid automata	Promela	PEPA, PRISM
Properties language	CTL, LTL	CTL, LTL, PSL	TCTL	CTL, Monitor automata	LTL	CSL, PLTL, PCTL
Real-time systems verification	–	–	+	+	–	–
Hybrid systems verification	–	–	–	+	–	–
Probabilistic systems verification	–	–	–	–	–	+
Large systems verification	+	+	–	–	+	–
GUI	+	+	+	+	+	+
Counter example visualize	–	+	+	+	+	–
License	FUSC	Free	FUSC	Free	FUSC	Free

IV. Conclusion

As a result of the comparative analysis it was revealed that there are no universal solutions in this knowledge area (Table I) Each of the considered means has its own advantages and is used for specific tasks.

References

[1] A.M. Mironov, *Verification of programs by the method of Model Checking*. 2012.

[2] Y.G. Karpov, *Verification of parallel and distributed software systems.*. Saint-Petersburg, Russia: BHV-Petersburg, 2010.

[3] I.B. Burdonov, A.S. Kosachev, V.N. Ponomarenko, V.Z. Schnitman, *Overview of Approaches to the Verification of Distributed Systems*. Moscow, Russia: ISP RAS, 2006.