Excel@FIT 2024
http://excel.fit.vutbr.cz

# Automata in Verification

Barbora Šmahlíková*

**Abstract**
Regular model checking is an automata-based technique used for verification of infinite-state systems. The configurations of a system are encoded as a finite automaton and transitions between these configurations as a finite transducer. A technique for verifying arbitrary properties of parameterized systems specified in a temporal logic LTL(MSO) has already been introduced. We present an extension of this algorithm allowing verification of hyperproperties of parameterized systems where an explicit quantification over multiple execution traces is allowed. The technique presented in this work is implemented in our tool ParaHyper.

*xsmahl00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

## 1. Introduction

Model checking is an important method for verifying finite-state systems. However, many real-life systems whose properties we would like to examine can have potentially infinite number of states. Verification of such systems has been an important area of research in the past few decades. One of the techniques that allow us to deal with parameterized and infinite-state systems is *regular model checking* [1, 2, 3, 4, 5]. In regular model checking, the states of the system are represented as finite words of arbitrary length over a finite alphabet. The sets of initial states is represented as a regular set of strings, and the transition relation is given as a finite regular length-preserving transducer.

When checking if a system satisfies some property, we would like to describe this property in some formalism. For systems that are not expected to terminate and can have possibly infinitely long runs (e.g. operating systems), temporal logics like LTL or CTL are usually used. In the case of finite-state systems, one of the approaches we can use to check if a system meets its specification given in LTL is to represent both the system and the property by a Büchi automaton and then check their language inclusion. However, verifying properties of parameterized and infinite-state systems is not as straightforward, because we cannot represent a system with infinitely many states using a finite-state Büchi automaton.

An extension of the automata-theoretic approach to regular model checking of parameterized and infinite-state systems has been proposed in [6]. The authors presented a logic LTL(MSO), which is a combination of the logics MSO over finite words for specifying sets of states and transition relations, and LTL for specifying temporal constraints. The models of LTL(MSO) are infinite sequences of words of constant length that represent computation of the specified system. The verification problem then consists of checking whether the conjunction of a system specification and a negation of the property to be verified is satisfiable.

The aim of this thesis is to extend the regular model checking algorithm of LTL(MSO) to an algorithm allowing quantification over execution traces, i.e., to be able to verify hyperproperties [7] in parameterized and infinite-state systems.

## 2. LTL(MSO)

LTL(MSO) is a two-dimensional logic for specifying properties of parameterized systems. MSO (Monadic second-order logic) is used for specification of system configurations and LTL (Linear temporal logic) is used for specification of temporal properties of the system. Models of this logic are infinite sequences of constant-length words.

## 3. Hyperproperties

The temporal logic LTL implicitly quantifies over only one execution trace. HyperLTL [8], an extension of

LTL, allows explicit quantification over multiple traces. One of the properties that can be expressed in HyperLTL but not in LTL is *observational determinism*. This property states that there is an identical output across all traces of the system and no information about high-security input is therefore leaked.

## 4. Verifying Hyperproperties

We present a logic HyperLTL(MSO) which is an extension of a logic LTL(MSO) allowing explicit quantification over multiple traces. In order to be able to represent hyperproperties, we need to use multitape automata and transducers. Every tape of an automaton represents one trace quantified in the formula. We can therefore keep track of current configurations of all traces at a certain timepoint.

## 5. Extending the System with Configuration Variables

A technique for representing all accepting runs of a system with respect to a formula in LTL(MSO) as a Büchi regular transition system has been proposed in [6]. We use the same technique to transform a body of the HyperLTL(MSO) formula without trace quantifiers. The body of the formula is translated into Büchi normal form which introduces new configuration variables in the formula. This formula can then be translated into an automaton representing restrictions on initial configurations, and transducers for restrictions on all system transitions and accepting transitions.

## 6. Generating Advice Bits

In [9], the authors presented a fully-automatic method for proving liveness over randomised parameterised systems using SAT solvers. The problem of liveness is transformed into looking for a pair of a finite automaton and a finite length-preserving transducer satisfying some conditions. The requirements on this pair are translated into a set of clauses and looking for an automaton and a transducer corresponds to generating models of the formula by a SAT solver and checking if they satisfy given conditions. Our approach was inspired by this method, but it was extended to automatically prove not only liveness, but an arbitrary formula in HyperLTL(MSO). The approach for computing advice bits consists of two steps – synthesis and verification. During synthesis, the pair $\langle A, \prec \rangle$ is generated by a SAT solver. In the verification phase, it is checked whether $\langle A, \prec \rangle$ satisfies conditions for advice bits. If the conditions does not hold, another pair is generated. The generated automata have $n_A$ and $n_\prec$ states, respectively. These parameters are initially set to 1 and then are increased. In this section, we describe how to encode a finite automaton or a finite transducer into a boolean representation for a SAT solver.

## 7. Implementation

The algorithm presented in this work is implemented in the tool ParaHyper[1]. On the input, the tool takes a HyperLTL(MSO), a finite automaton representing initial configurations of the system, a finite length-preserving transducer representing transitions of the system, and a bound on the number of states for the advice bits.

## 8. Conclusions

We presented an algorithm for regular model checking of an arbitrary formula in HyperLTL(MSO). This approach is implemented in the tool ParaHyper and it uses SAT solver for the generation of candidate pairs of advice bits.

## Acknowledgements

I would like to thank my supervisor Ing. Ondřej Lengál, Ph.D., and my consultant Ing. Vojtěch Havlena, Ph.D. for their help, ideas and guidance during my work on this thesis.

## References

[1] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. *Theoretical Computer Science*, 256(1):93–112, 2001. ISS.

[2] Pierre Wolper and Bernard Boigelot. Verifying systems with infinite but regular state spaces. In Alan J. Hu and Moshe Y. Vardi, editors, *Computer Aided Verification*, pages 88–97, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[3] Parosh Aziz Abdulla, Ahmed Bouajjani, Bengt Jonsson, and Marcus Nilsson. Handling global conditions in parametrized system verification. In Nicolas Halbwachs and Doron Peled, editors, *Computer Aided Verification*, pages 134–145, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[4] Parosh Aziz Abdulla, Bengt Jonsson, Pritha Mahata, and Julien d'Orso. Regular tree model checking. In Ed Brinksma and Kim Guldstrand Larsen,

[1] https://github.com/barbora4/ParaHyper

editors, *Computer Aided Verification*, pages 555–568, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[5] Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Mayank Saksena. A survey of regular model checking. In Philippa Gardner and Nobuko Yoshida, editors, *CONCUR 2004 - Concurrency Theory*, pages 35–48, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[6] Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, Julien d'Orso, and Mayank Saksena. Regular model checking for ltl(mso). In Rajeev Alur and Doron A. Peled, editors, *Computer Aided Verification*, pages 348–360, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[7] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. *J. Comput. Secur.*, 18(6):1157–1210, sep 2010.

[8] Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. Temporal logics for hyperproperties. In Martín Abadi and Steve Kremer, editors, *Principles of Security and Trust*, pages 265–284, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[9] Anthony Lin and Philipp Rümmer. Liveness of randomised parameterised systems under arbitrary schedulers. volume 9780, 07 2016.