Extending Apalache to Symbolically Reason about Temporal Properties of TLA+

Philip Offtermatt

philip.offtermatt@informal.systems

joint work with Jure Kukovec and Igor Konnov
Informal Systems, Vienna, Austria

Abstract

Apalache is a symbolic model checker for TLA$^+$, which implements bounded model checking and symbolic execution of TLA$^+$ by interacting with the satisfiability-modulo-theory solver Z3. Until recently, Apalache could only check four kinds of invariants: state, invariants, invariants, and inductive. This talk presents our recent work on implementing support for checking temporal properties in Apalache. Conceptually, our implementation applies the well-known technique of liveness-to-safety reduction for linear temporal logic. We show how to adapt this reduction to TLA$^+$, in order to address the unique challenges presented by its logic.

1 Overview

Apalache [2, 4] is a model checker that presents an alternative to TLC for checking TLA$^+$ specifications. The key distinction is that Apalache performs symbolic model checking, and is thus able to reason about large (or even infinite) state spaces. The main focus of Apalache development was on invariant checking. It supports checking state invariants, action invariants, inductive invariants, and trace invariants. Thus, until recently Apalache provided no support for checking arbitrary temporal properties.

This talk presents our recent work on extending Apalache with checking of temporal properties by employing an encoding of liveness as safety. In short, given a TLA$^+$ specification *Spec* and a temporal property P, Apalache produces an instrumented specification $Spec_P$ and an invariant I_P that have the following property: the temporal property P holds on the original specification *Spec* if and only if I_P is an invariant of the specification S_P. Then, Apalache checks the invariant I_P against $Spec_P$. Importantly, Apalache can also emit the intermediate specification, so it can easily be fed into and checked by other tools.

In general, we follow the encoding for bounded model checking of LTL by Biere et. al.[1]. While the technique is in principle well-known, it turns out that implementing it presents some challenges, specifically in the context of TLA$^+$. For example, while LTL generally admits a "next" operator, the only way to reason about the next state in TLA$^+$ is via primed variables. However, "next" can generally be applied many times, while double-priming a variable is forbidden in TLA$^+$. Thus, the results of the work might

give interesting insights for adapting other techniques designed for model checking of LTL formulas to the world of TLA$^+$. A challenge of the encoding is that it is complete only for finite-state systems. Temporal model checking of infinite systems specified with TLA$^+$ requires further research. For instance, we could apply the liveness-to-safety reduction by Padon et al. [3].

Our talk has three goals:

1. Explain the basics of the liveness-to-safety encoding as implemented in Apalache. We present this by a running example.

2. Highlight the challenges of applying the encoding in the context of TLA$^+$.

3. Give a brief overview of explored approaches for using Apalache to check arbitrary temporal properties, and explain the advantages and drawbacks of the chosen approach.

References

[1] Armin Biere, Keijo Heljanko, Tommi Junttila, Timo Latvala, and Viktor Schuppan. Linear Encodings of Bounded LTL Model Checking. *Logical Methods in Computer Science*, Volume 2, Issue 5, November 2006. doi:10.2168/LMCS-2(5:5)2006. URL https://lmcs.episciences.org/2236.

[2] Igor Konnov, Jure Kukovec, and Thanh-Hai Tran. TLA+ model checking made symbolic. *Proc. ACM Program. Lang.*, 3(OOPSLA), October 2019. doi:10.1145/3360549. URL https://doi.org/10.1145/3360549.

[3] Oded Padon, Jochen Hoenicke, Giuliano Losa, Andreas Podelski, Mooly Sagiv, and Sharon Shoham. Reducing liveness to safety in first-order logic. *Proc. ACM Program. Lang.*, 2(POPL):26:1–26:33, 2018. URL https://doi.org/10.1145/3158114.

[4] Informal Systems. Apalache model checker. https://github.com/informalsystems/apalache, 2022.