reTLA: Towards an automatic transpiler from TLA$^+$ to VMT

TLA+ Conference 2022

Jure Kukovec[1,5], Aman Goel[2], Igor Konnov[1], Stephan Merz[3], and Karem Sakallah[4]

[1]Informal Systems, Austria
[2]Amazon Web Services, USA
[3]University of Lorraine, CNRS, Inria, LORIA, Nancy, France
[4]University of Michigan, Ann Arbor, USA
[5]Email: jure@informal.systems

In this talk, we present our ongoing attempts to bring the capability of automatically inferring inductive invariants for specifications written in a subset of TLA+, through automatic translation of a restricted fragment of TLA$^+$ to VMT [3][1] using Apalache [10], followed by automatic inference of quantified inductive invariants using IC3PO [6, 5].

IC3PO for inductive invariant inference. IC3PO is a recently-developed model checker that utilizes different structural features in protocol specifications to prove the safety of distributed protocols by automatically inferring compact quantified inductive invariants. The key insight underlying IC3PO is that structural regularity and quantification are closely related concepts that express protocol invariance under different re-arrangements of its components and its unbounded evolution over time. IC3PO utilizes the inherent spatial and temporal regularity in protocol specifications to significantly boost IC3/PDR-style[2] verification and learn quantified predicates. For safe protocols, IC3PO systematically computes quantified inductive invariants over protocol instances of increasing sizes, until protocol behaviors saturate, concluding with an inductive proof that works for all protocol instances. Else, a finite counterexample trace is produced if instead the protocol can violate the safety property. In particular, IC3PO was recently shown in [7] to automatically infer an inductive invariant for Paxos [8], that identically matches the human-written proof previously derived with significant manual effort using interactive theorem proving.

[1]VMT format is an extension of SMT-LIB [1] to represent symbolic transition systems.
[2]IC3 [2] and PDR [4] are state-of-the-art model checking algorithms that are widely adopted for hardware verification.

Translating TLA$^+$ to VMT in Apalache. As TLA$^+$ is an incredibly expressive language, we first restrict the fragment, for which we attempt an automatic translation. We dub this fragment reTLA, short for relational TLA$^+$, as the fragment is centered around relations or, in a more general sense, functions with primitive-valued (co)domains (e.g., integers or strings, but not records). A complete characterization of reTLA is available in the Apalache documentation [9]. For each valid reTLA expression, we construct an automatic translation to VMT. The restrictions imposed on reTLA ensure that the translated formulas fall into the (quantified) logic of equality with uninterpreted functions. We leverage Apalache's type system [11], to determine the sorts of the expressions in the VMT encoding. For example, take the TLA$^+$ definition

$$A \triangleq [F \text{ EXCEPT } ![1] = \text{TRUE}]$$

where F is some function whose Apalache type is $Int \to Bool$. Assume that the VMT translation of F is some function f, declared as (declare-fun f (Int) $Bool$). The translation of A gives us a function a, defined as

$$(\text{define-fun } a\ ((x\ Int))\ Bool\ (\text{ite } (=\ x\ 1)\ true\ (f\ x)))$$

Note that reTLA currently does not permit sets-valued data. Concretely, (certain) sets are allowed to be quantified over, but set operators, such as union or intersection, are disallowed in the fragment.

Integers as a total order. Our encoding supports integer values, but not integer arithmetic. We do this as a form of syntactic sugar: in our encoding, Apalache integers are translated not to VMT integers, but to an uninterpreted sort Int with a defined operator $<_{Int}$, which satisfies the axioms of a total order. Then, integer literals are translated to distinct constants of the Int sort, such that their relative order w.r.t. $<_{Int}$ matches their standard integer order. One may use integer (in)equality notation ($<, >, \leq, \geq$), which gets translated to the equivalent combination of $<_{Int}$ and equality. Importantly, this allows us to remain outside the integer-arithmetic fragment of VMT. Note that, because we treat integers as syntactic sugar, there are a few consequences which are counter-intuitive, such as the formula $\exists j \in Int\colon 1 < j < 2$ not being trivially false. Since 1 and 2 are just syntax sugar for two totally-ordered constants of the Int sort, there is no constraint on the existence of "gap" values. We accept this as a trade-off, to retain the ease of use, which comes from being able to use the standard integer and (in)equality notation in TLA$^+$.

Evaluation and future work. As a proof of concept, we were able to express a number of distributed algorithms in reTLA, including a client-server protocol, decentralized lock, sharded key-value store, and two-phase commit, to automatically translate them into VMT using Apalache, and verify their safety properties for arbitrary instance sizes using IC3PO. Other protocols such as versions of Paxos and Raft are currently outside the scope of reTLA. These preliminary results encourage us to extend the scope of the translation and hopefully provide another tool for the automatic verification of TLA$^+$ specifications.

References

[1] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. *The Satisfiability Modulo Theories Library (SMT-LIB)*. `www.SMT-LIB.org`. 2016.

[2] Aaron R. Bradley. “SAT-Based Model Checking without Unrolling”. In: *Proceedings of the 12th international conference on Verification, model checking, and abstract interpretation*. VMCAI’11. Austin, TX, USA: Springer-Verlag, 2011, pp. 70–87. ISBN: 978-3-642-18274-7. URL: `http://dl.acm.org/citation.cfm?id=1946284.1946291`.

[3] Alessandro Cimatti et al. *Verification Modulo Theories*. `http://www.vmt-lib.org`. 2011.

[4] Niklas Een, Alan Mishchenko, and Robert Brayton. “Efficient Implementation of Property Directed Reachability”. In: *Proceedings of the International Conference on Formal Methods in Computer-Aided Design*. FMCAD ’11. Austin, Texas: FMCAD Inc, 2011, 125–134. ISBN: 9780983567813.

[5] Aman Goel and Karem A. Sakallah. *IC3PO: IC3 for Proving Protocol Properties*. `https://github.com/aman-goel/ic3po`.

[6] Aman Goel and Karem A. Sakallah. “On Symmetry and Quantification: A New Approach to Verify Distributed Protocols”. In: *13th Intl. Symp. NASA Formal Methods (NFM 2021)*. Ed. by Aaron Dutle et al. Vol. 12673. LNCS. Springer, 2021, pp. 131–150.

[7] Aman Goel and Karem A. Sakallah. “Towards an Automatic Proof of Lamport’s Paxos”. In: *Formal Methods in Computer-Aided Design (FMCAD)*. Ed. by Ruzica Piskac and Michael W Whalen. New Haven, Connecticut, 2021, pp. 112–122. DOI: `10.34727/2021/isbn.978-3-85448-046-4_20`.

[8] Leslie Lamport. “The part-time parliament”. In: *ACM Transactions on Computer Systems (TOCS)* 16.2 (1998), pp. 133–169.

[9] *Relational TLA$^+$*. `https://apalache.informal.systems/docs/adr/016adr-retla.html`.

[10] *The Apalache model checker for TLA$^+$*. `https://apalache.informal.systems`.

[11] *Types in Apalache*. `https://apalache.informal.systems/docs/adr/002adr-types.html`.