# Certifying Hardware Model Checking Results*

Zhengqi Yu[1], Armin Biere[1], and Keijo Heljanko[2]

[1] Johannes Kepler University Linz, Austria
[2] University of Helsinki, Finland

**Abstract.** Model checking is used widely as a formal verification technique for safety-critical systems. Certifying the correctness of model checking results helps increasing confidence in the verification procedure. This can be achieved by additional book-keeping inside existing model checkers. Based on this, we extended an existing BDD-based model checker as well as an IC3-based incremental inductive model checker, to generate certificates during the model checking procedure. We also introduce a proof checker which provides a standardised way to validate certificates generated from model checkers in conjunction with a SAT solver. The main goal is to establish a certification process for the hardware model checking competition.

## 1 Introduction

The verification of software and hardware systems has become increasingly important in modern world with the increase in complexity of system design and analysis, especially for safety-critical systems. Model checking [1–3] is a formal method widely adopted for automatic system verification, such as model checking of safety critical software in the nuclear engineering domain [4]. A model checker typically takes the model of a system and some properties corresponding to certain specification as inputs, and verifies if the properties are satisfied in the given model. As model checkers themselves are complicated programs, any programming errors can potentially lead to incorrect verification results, which can directly affect the analysis of system designs. It is therefore crucial to ensure the correctness in the process of system verification. Certifying model checkers increases confidence in model checking results, since the certificates can be validated by proof checkers or SAT solvers which are much simpler pieces of software with less complexity.

Various work has been done in this area. In SAT, certifying proofs is an established technology [5] and for instance mandatory in the SAT competition since 2013. In [6], the authors present an approach for certifying liveness properties using the $k$-liveness [7,8] approach to map the problem into a safety property and then proving an inductive invariant. The paper uses an IC3-based model checker and suggests validating the invariants provided by IC3 using a SAT solver but provides no experimental data on the invariant validation. In [9], the authors

* Funded by FWF project W1255-N23 and Academy of Finland project 325300.

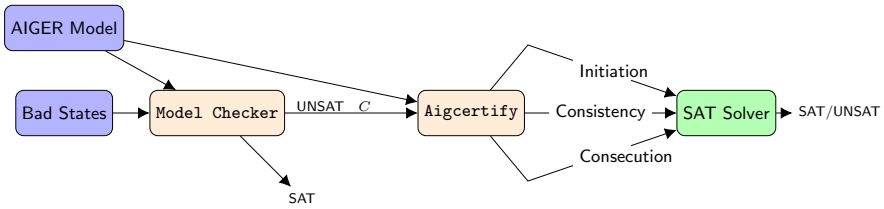


**Fig. 1.** Certifying Procedure

discussed certifying LTL model checking and their experimental results are also obtained from an IC3-based model checker.

In this paper, we present an automatic proof checker AIGCERTIFY which provides a *standardised* way to certify different types of model checkers with AIGER format. We experimented our tool with AIGTRAV, a BDD-based model checker developed at JKU Linz, and IIMC [10], which is an IC3-based model checker. Our ongoing work involves certifying $k$-induction-based model checkers.

## 2 Approach

Figure 2 shows the data flow of the certifying procedure, and here we are only dealing with examples with positive model checking results, as the results of counter-examples can be checked using counter-example validation.

The main idea is to use an inductive invariant as certificate, which is stored as an AIGER file. This certificate has to imply the given safety property (the negation of the bad state predicate). Here we use $I$ to denote Boolean formula encoding of the initial states, $C$ is the certificate given by the model checker and $C'$ represents the certificate after a transition, $B$ is the predicate representing the bad states, and $T$ is the transition relation. There are three conditions that the inductive invariant must satisfy:

| **Condition** | **Formula** | **The inductive invariant ...** |
|---|---|---|
| Initiation | $I \implies C$ | ... must hold at all initial states. |
| Consistency | $C \implies \neg B$ | ... must hold at states that are not bad states. |
| Consecution | $C \wedge T \implies C'$ | ... is preserved during the transition. |

For instance in BDD-based model checking the Boolean formula encoding the set of reachable states is such an inductive invariant $C$.

We have implemented AIGCERTIFY as a proof checker, which accepts a certificate as an AIGER file, either in binary or ASCII format, which will generate CNF proof obligations in DIMACS format. As an automatic proof checker, it is designed to work with different types of model checkers. We therefore define the format of a certificate that AIGCERIFY accepts, which can be obtained during the verification procedure inside a model checker. An AIGER model consists of $M$ (maximum variables), $I$ (inputs), $L$ (latches), $O$ (outputs), $A$ (AND gates)

which are represented as literals indicated as numbers. Due to the space limit, we cannot include a full description of the AIGER format here, but more information can be found in [11]. The format of a certificate is defined as follows:

**Definition 1 (Format).** *Given a model $M$ with $(M_M, I_M, L_M, O_M, A_M)$, a certificate $C$ with $(M_C, I_C, L_C, O_C, A_C)$, the format of a certificate in AIGER must satisfy the following:*

- $I_C = L_M$
- $L_C = 0$
- *Let $m_0, ..., m_{L_M-1} \in \mathbb{N}$ be the set of latches of $M$, and $c_0, ..., c_{I_C-1} \in \mathbb{N}$ be the set of inputs of $C$, for an arbitrary $c_i$ with $0 \leq i < I_C$, $c_i = m_i - I_M$.*

We extended AIGTRAV, to generate the inductive invariant in AIGER format after it finishes model checking. Our implementation also includes a conversion from the BDD structure to AIGER format. The resulting invariant can then be verified by the proof checker AIGCERTIFY.

In addition, we experimented with IIMC [10], which already provides an internal proof certificate as one of the features of the IC3 model checking algorithm. The proof certificate of IC3-based model checking can be obtained directly from the one-step inductive strengthening. We extended the source code to provide the inductive invariant in AIGER format which can then be used by AIGCERTIFY.

## 3 Implementation and Experiments

We have implemented AIGCERTIFY as a proof checker in order to verify the correctness of certificates generated from model checkers to increase confidence in verification. It accepts a model and a certificate both in AIGER format as inputs, as defined in Definition 1, and generates three conditions (explained in Section 2) as separate AIGER files, which are then checked by an existing SAT solver (like PicoSAT [12]). For additional validation the SAT solver can also generate proofs that can be further checked by a SAT proof checker (such as DRAT-TRIM [5]), resulting in a two stage proof validation.

The variable ordering of the AIGER model is assumed to be the same as that of the certificate, which ensures they are referring to the same set of latches. We utilise the AIGER library `simpaig.c` which provides a simple AIG data structure that allows operations on variables and AND gates.

The preservation condition requires the certificate to hold after each transition. Typically, a transition relation is the conjunction of the values of current inputs and states, and the values of next states. In AIGER format, the next states are represented by the next values defined for the latches.

The three output AIGER files are converted to CNF files by the AIGER library `aigtocnf.c` using Tseitin encoding, and in our experimental results they are then validated by the existing SAT solver, PicoSAT [12]. These three conditions form the proof for the model checking procedure, and if all three CNFs are verified to hold, the certificate is proved successfully. It is further possible to certify SAT solving by generating and checking propositional DRAT proofs [13]. We are currently applying our approach to benchmarks from HWMCC'17 [14].

## 4 Discussion and Conclusion

We introduced our tool AIGCERTIFY which is designed to be a uniform proof checker for different types of model checkers, including BDD-based and SAT-based. We experimented with existing model checking tools to work in conjunction with it. The ongoing work of our project includes generating certificates from $k$-induction-based model checkers which can then be verified by AIGCERTIFY, which also involves formally defining the inductive invariant of $k$-induction-based model checking. This might also draw inspiration from [15,16]. Even though our focus is currently on providing certificates for the hardware model checking competition, similar ideas might be applicable to software model checking too.

## References

1. E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, Eds., *Handbook of Model Checking*. Springer, 2018.
2. E. M. Clarke, O. Grumberg, D. Kroening, D. Peled, and H. Veith, *Model Checking*. MIT press, 2018.
3. C. Baier and J. Katoen, *Principles of model checking*. MIT Press, 2008.
4. J. Lahtinen, J. Valkonen, K. Björkman, J. Frits, I. Niemelä, and K. Heljanko, "Model checking of safety-critical software in the nuclear engineering domain," *Rel. Eng. & Sys. Safety*, vol. 105, pp. 104–113, 2012.
5. M. Heule, W. Hunt, and N. Wetzler, "Trimming while checking clausal proofs," in *FMCAD 2013*, 2013, pp. 181–188.
6. T. Kuismin and K. Heljanko, "Increasing confidence in liveness model checking results with proofs," in *HVC*, ser. LNCS, vol. 8244. Springer, 2013, pp. 32–43.
7. K. Claessen and N. Sörensson, "A liveness checking algorithm that counts," in *FMCAD 2012, Cambridge, UK*. IEEE, 2012, pp. 52–59.
8. X. Gan, J. Dubrovin, and K. Heljanko, "A symbolic model checking approach to verifying satellite onboard software," *Sci. Comput. Program.*, vol. 82, 2014.
9. A. Griggio, M. Roveri, and S. Tonetta, "Certifying proofs for LTL model checking," in *FMCAD 2018*, N. Bjørner and A. Gurfinkel, Eds. IEEE, 2018, pp. 1–9.
10. A. Bradley, F. Somenzi, and Z. Hassan, *IIMC: incremental inductive model checker*. [Online]. Available: http://www.github.com/mgudemann/iimc
11. A. Biere, K. Heljanko, and S. Wieringa, "AIGER 1.9 and beyond," FMV Reports Series, Institute for Formal Models and Verification, Johannes Kepler University Linz, Austria, Tech. Rep., 2011.
12. A. Biere, "Lingeling, Plingeling, PicoSAT and PrecoSAT at SAT race 2010," FMV Reports Series, Institute for Formal Models and Verification, Johannes Kepler University Linz, Austria, Tech. Rep., 2010.
13. N. D. Wetzler, M. J. H. Heule, and W. A. Hunt Jr., "DRAT-trim: Efficient checking and trimming using expressive clausal proofs," in *SAT*, ser. LNCS, vol. 8561. Springer, 2014, pp. 422–429.
14. A. Biere, T. van Dijk, and K. Heljanko, "Hardware model checking competition 2017," in *FMCAD*, D. Stewart and G. Weissenbacher, Eds. IEEE, 2017, p. 9.
15. H. G. V. Krishnan, Y. Vizel, V. Ganesh, and A. Gurfinkel, "Interpolating strong induction," in *CAV 2019, Proceedings, Part II*, 2019, pp. 367–385.
16. N. Bjørner, A. Gurfinkel, K. L. McMillan, and A. Rybalchenko, "Horn clause solvers for program verification," in *Fields of Logic and Computation II*, ser. LNCS, vol. 9300. Springer, 2015, pp. 24–51.