# Automated Polynomial Formal Verification: Human-Readable Proof Generation

Rolf Drechsler*† Martha Schnieber*
*Institute of Computer Science, University of Bremen, Bremen, Germany
†Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
{drechsler, schnieber}@uni-bremen.de

***Abstract*—The importance of verification of digital circuits has increased significantly, as their complexity has grown. Simulation based techniques cannot fully guarantee their correctness, as the correctness has to be shown for every possible input assignment. Thus, formal verification has to be applied. However, formal verification methods may require exponential time and space in the worst case. Therefore, *Polynomial Formal Verification* (PFV) has been researched in the past years, where polynomial upper bounds have been proven for the complete formal verification process. By this efficient run times of the tools are guaranteed. Polynomial bounds have been proven successfully for the verification of several types of circuits, like e.g. adders and multipliers. However, due to the lack of automation techniques, all previous proofs were conducted manually. A tool enabling the automatic proof generation has recently been introduced, which demonstrates the concept of automatic proofs on the example of *Binary Decision Diagrams* (BDDs). We enhance the tool to automatically generate an extended human-readable proof, detailing the automatic reasoning, such that the produced proof is fully comprehensible.**

## I. Introduction

In the past years, the significance of verification of digital circuits has increased, due to their rising complexity. However, the correctness of a circuit can only be guaranteed by formal verification techniques, because simulation can only test the circuit for a limited amount of input assignments. To prove the correctness of a circuit, formal verification techniques based on decision diagrams, e.g. *Binary Decision Diagrams* (BDDs) [1], [2], *Kronecker Functional Decision Diagrams* (KFDDs) [3] or *Multiplicative Binary Moment Diagrams* (*BMDs) [4] can be employed, as well as *Boolean Satisfiability* (SAT) [5] or *Symbolic Computer Algebra* (SCA) [6]. However, in the worst case, all formal verification techniques require exponential time and space, potentially causing the verification to fail due to time or space constraints. Thus, in the past years, PFV [7], [8] has been introduced to predict the verification time beforehand. Here, the verification complexity of several circuits has been researched, proving polynomial upper bounds for the time and space complexity of the verification of specific circuits.

Circuits for which a polynomial time and space complexity has already been proven using BDDs include several adders, i.e. the *Ripple Carry Adder* (RCA), *Conditional Sum Adder* (CSA) and *Carry Look Ahead Adder* (CLA) [7], [9], as well as several *Prefix Adders* (PAs) [10]. Furthermore, polynomial upper bounds were also proven for the BDD-based verification of e.g. BDD circuits and tree-like circuits [11], as well as symmetric functions [12] and floating point adders [13]. Apart from BDDs, other types of decision diagrams can be used as well. Here, it was proven that PFV using KFDDs is possible for KFDD circuits [14] and general tree-like circuits [15], whereas *BMDs have been used for the PFV of Wallace-tree like multipliers [16]. Furthermore, SCA has been used to prove polynomial upper bounds for the verification of arithmetic circuits [6] and for complex multipliers [17], whereas *Answer Set Programming* (ASP) has been applied for PFV of circuits with a constant cutwidth [18].

However, all polynomial upper bounds were proven manually, leading to a time-consuming and also error-prone process. Thus, a concept for automatically generated proofs for polynomial upper bounds was introduced recently [19]. Here, a tool was developed, illustrating the concept at the example of BDDs. The concept of automatic proofs has already been explored in other research fields, e.g. automated theorem proving [20] or the four color problem [21]. However, for users to understand the automated reasoning, producing a human-readable proof is essential. Thus, in this paper, we extend the previously introduced concept of automatic proofs for PFV by generating an elaborate human-readable proof, detailing the induction and reasoning.

## II. Automated Polynomial Formal Verification

In [19], the concept for automated PFV was introduced, alongside a tool exemplarily showing the concept for BDD-based verification. In this section, the previously introduced proof engine is outlined and the extension for the human-readable proof is presented.

### A. Automatic Proof Engine

The automatic proof engine flow is shown in Figure 1. For an iterative input function, the tool automatically finds a pattern for the generalized BDD by comparing the BDD for an iteration $k$ to the BDD for the iteration $k+1$. Figure 2 shows an example, where the nodes $v_1$ and $v_2$ are added in the $(k+1)$-th iteration for a function $f(x_1, ...x_n)$. All required information for the proof is stored in the pattern, including the base case, meaning the BDD for the first iteration. Furthermore, the

This work was supported by the German Research Foundation (DFG) within the Reinhart Koselleck Project *PolyVer* (DR 287/36-1).

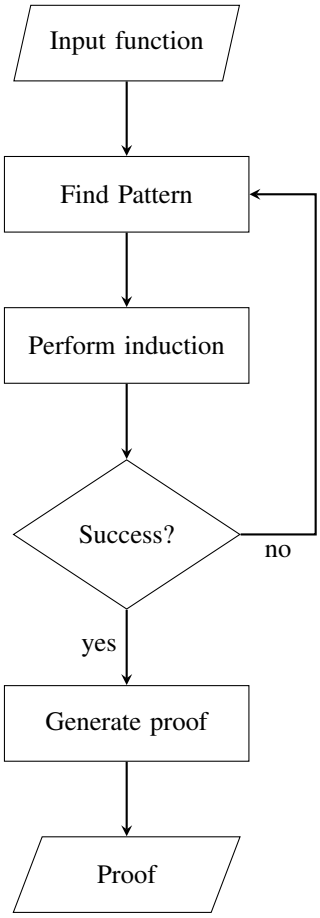


Fig. 1: Automatic proof engine flow [19]

pattern stores all new nodes in the BDD for $k+1$, as well as their respective $high$- and $low$-edges.

If a pattern is found, its correctness is proven by induction on the number of iterations $k$. To conduct the proof by induction, the base case is first checked, where the BDD for the first iteration is compared to the base case stored in the pattern. For the induction step, the *If Then Else* (ITE) operator [22] is propagated through the pattern for the BDD with $k$ iterations according to the function description, obtaining the BDD for $k+1$ based on the BDD for $k$. The correctness of the pattern can be proven, if each node from the propagated ITE operator can be matched to a node from the pattern, proving the correctness of the pattern for $k+1$ iterations, given its correctness for $k$ iterations.

If the induction is successful, the human-readable proof is generated. Otherwise, a different pattern is searched for and the process is repeated. If the induction fails for a set amount of patterns, the process terminates and the automatic proof of a polynomial upper bound is unsuccessful for the given function.

### *B. Human-Readable Proof Generation*

We define a human-readable proof as a structured proof that is easily understandable and closely resembles a proof written

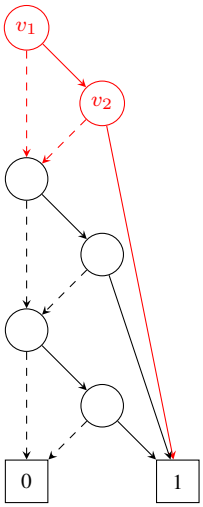


Fig. 2: BDD for $f(x_1, ...x_n)$ with $k = 2$ [19]

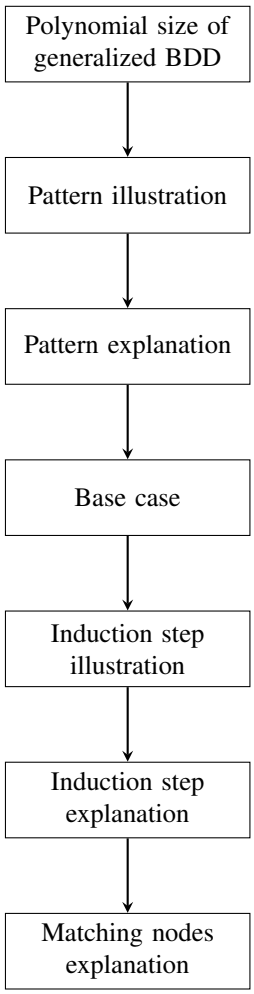


Fig. 3: Human-readable proof generation

manually by a human. The structure of the proposed extended automatic human-readable proof generation is outlined in Figure 3.

Firstly, the size of the generalized BDD is determined and printed in the proof. The size consists of the number of nodes in the base case plus the number of nodes added per iteration, which can both be extracted from the saved pattern. To underline the BDD size, an example of a BDD highlighting the added nodes is automatically printed as an illustration, as shown in Figure 2. To enhance the comprehensibility of the proof, the pattern is also textually explained. Here, the nodes added per iteration are described, including their respective $high$- and $low$-edges.

Additionally, the induction is explained in the proof. To depict the base case, both the base case stored in the pattern, as well as the BDD for the first iteration are displayed and compared. The induction step consists of an illustration depicting the propagation of the ITE operator through the pattern for $k$ iterations. Furthermore, each step of the propagation of the ITE operator is automatically explained textually. Finally, to prove the correctness of the pattern, all new nodes resulting from the propagated ITE operator are matched to a node from the pattern.

Following the proposed steps, a complete human-readable proof is generated. Thus, for each proof conducted by the engine, the generated proof can be checked manually to ensure its correctness.

## III. Conclusion

Due to the time consuming and error-prone manual work of conducting proofs for PFV, an automated approach has been introduced recently. In this paper, we have enhanced the concept of automatically generated proofs for PFV by extending the automatic human-readable proof generation to include a more detailed explanation of the proof by induction. In future work, the tool can be extended to perform proofs for a wider range of functions and support other formal verification methods, such as SAT or other types of decision diagrams.

## References


[1] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Transactions on Computers*, vol. 35, no. 8, pp. 677–691, 1986.

[2] R. Drechsler and B. Becker, *Binary Decision Diagrams: Theory and Implementation*. Springer US, 2013.

[3] ——, "Ordered Kronecker functional decision diagrams-a data structure for representation and manipulation of Boolean functions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 10, pp. 965–973, 1998.

[4] R. E. Bryant and Y.-A. Chen, "Verification of arithmetic circuits with binary moment diagrams," in *Proceedings of the 32nd Annual ACM/IEEE Design Automation Conference (DAC)*, 1995, p. 535–541.

[5] A. Kuehlmann, V. Paruthi, F. Krohm, and M. K. Ganai, "Robust Boolean reasoning for equivalence checking and functional property verification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 12, pp. 1377–1394, 2002.

[6] M. Barhoush, A. Mahzoon, and R. Drechsler, "Polynomial word-level verification of arithmetic circuits," in *2021 19th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*, 2021, pp. 1–9.

[7] R. Drechsler, "PolyAdd: Polynomial formal verification of adder circuits," in *2021 24th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*. IEEE, 2021, pp. 99–104.

[8] R. Drechsler and A. Mahzoon, "Polynomial formal verification: Ensuring correctness under resource constraints : (invited paper)," in *2022 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2022, pp. 1–9.

[9] A. Mahzoon and R. Drechsler, "Late breaking results: Polynomial formal verification of fast adders," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 1376–1377.

[10] ——, "Polynomial formal verification of prefix adders," in *2021 IEEE 30th Asian Test Symposium (ATS)*, 2021, pp. 85–90.

[11] R. Drechsler, "Polynomial circuit verification using BDDs," in *2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)*, 2021, pp. 49–52.

[12] R. Drechsler and C. Dominik, "Edge verification: Ensuring correctness under resource constraints," in *2021 34th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2021, pp. 1–6.

[13] J. Kleinekathöfer, A. Mahzoon, and R. Drechsler, "Polynomial formal verification of floating point adders," in *2023 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2023, pp. 1–2.

[14] M. Schnieber and R. Drechsler, "Polynomial formal verification of KFDD circuits," in *2023 21th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*, 2023.

[15] A. Mahzoon and R. Drechsler, "Polynomial formal verification of general tree-like circuits," in *2022 China Semiconductor Technology International Conference (CSTIC)*, 2022, pp. 1–4.

[16] M. Keim, R. Drechsler, B. Becker, M. Martin, and P. Molitor, "Polynomial formal verification of multipliers," *Formal Methods in Systen Design*, vol. 22, no. 1, pp. 39–58, 2003.

[17] R. Drechsler, A. Mahzoon, and M. Goli, "Towards polynomial formal verification of complex arithmetic circuits," in *2022 25th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, 2022, pp. 1–6.

[18] M. Nadeem, J. Kleinekathöfer, and R. Drechsler, "Polynomial formal verification of adder circuits using answer set programming," in *2023 Reed-Muller Workshop (RM2023)*, 2023.

[19] R. Drechsler and M. Schnieber, "Next-generation automatic human-readable proofs enabling polynomial formal verification," in *2023 21th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*, 2023.

[20] D. W. Loveland, *Automated Theorem Proving: A Logical Basis (Fundamental Studies in Computer Science)*. Elsevier North-Holland, Inc., 1978.

[21] K. Appel and W. Haken, "Every planar map is four colorable," *Bulletin of the American Mathematical Society*, vol. 82, no. 5, 1976.

[22] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient implementation of a BDD package," in *27th ACM/IEEE Design Automation Conference*, 1990, pp. 40–45.