Software Verification with PDR:

An Implementation of the State of the Art

Proc. TACAS 2020, https://doi.org/10/f377

Dirk Beyer

Joint work with Matthias Dangl

LMU Munich, Germany

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

SoSy-Lab Software Systems

Contributions

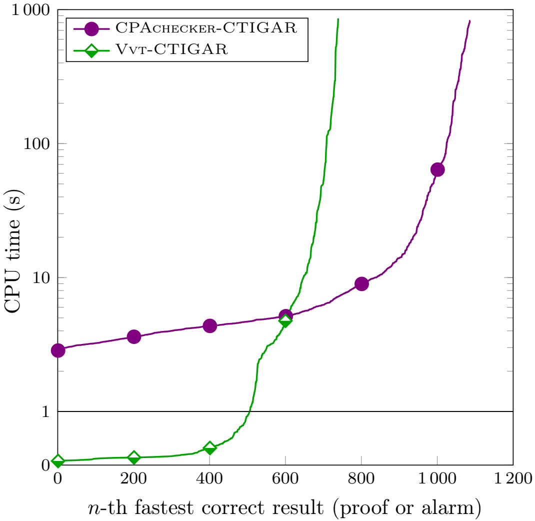
- Implement one adaptation of PDR to software verification
- Design and implement the invariant generator KIPDR (helping k-induction by providing better invariants)
- Large experimental study to compare several approaches (VTT, CPAchecker, k-induction, SeaHorn, VVT-Portfolio)
- Set of small examples that need 'difficult' invariants

Background

- Property-Driven Reachability (PDR) [7, 8, 9, 14]
  received a lot of attention and
  successful for hardware designs
- CTIGAR and Vienna Verification Toolkit (VVT) [6, 10]
  the approach that we followed
- Integrate with k-induction [4, 5, 13]
  our solution for a fully integrated approach
- Compare with VVT and SeaHorn [10, 12, 11]
  establish a new state-of-the-art reference implementation

Eval: Suitability of CPAchecker for PDR.

- Two implementations of CTIGAR [6, 10]
- Is our implementation competitive? ✓



Eval: KIPDR versus Data-Flow Techniques

- Four k-induction-based configurations
- Manually crafted examples (click on program to open), on which k-induction alone does not succeed
- In-depth discussion of the examples in technical report [2]
- 'T' means time limit, 'M' means memory limit exceeded

Task	KI←DF Boxes	KI←DF Boxes, Eq	KI←DF Boxes, Eq, Mod2	KI←⟲←KIPDR
const.c	3.3 s	3.3 s	**3.2 s**	3.8 s
eq1.c	T	**3.2 s**	3.3 s	4.9 s
eq2.c	M	M	M	**3.9 s**
even.c	T	T	**3.5 s**	3.9 s
odd.c	T	T	**3.4 s**	4.1 s
mod4.c	T	T	T	**3.6 s**
bin-suffix-5.c	M	M	M	**3.6 s**

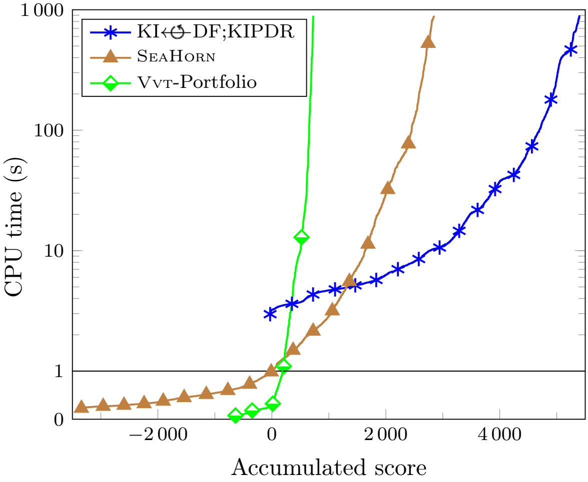
Eval: Comparison with Non-PDR Approaches

- Results of SV-COMP 2019 [1]
- Reports the six verifiers that performed best on those problems
- 'T' means time limit, 'O' means verifier gave up

Task	SV-COMP 2019						KI⟲KIPDR
	SKINK	UAUTOMIZER	UKOJAK	UTAIPAN	VERIABS	VIAP	
`const.c`	4.2 s	8.7 s	9.1 s	8.2 s	13 s	110 s	**3.8 s**
`eq1.c`	290 s	7.8 s	7.6 s	8.3 s	14 s	57 s	**4.9 s**
`eq2.c`	4.1 s	8.1 s	8.6 s	7.6 s	14 s	4.7 s	**3.9 s**
`even.c`	**3.7 s**	7.4 s	8.2 s	8.6 s	140 s	4.5 s	3.9 s
`odd.c`	O	9.6 s	T	11 s	140 s	4.6 s	**4.1 s**
`mod4.c`	4.0 s	8.4 s	8.4 s	7.7 s	140 s	4.5 s	**3.6 s**
`bin-suffix-5.c`	O	14 s	T	13 s	13 s	4.7 s	**3.6 s**

Eval: Comparison with PDR-Based Verifiers

- Accumulated score of solved tasks



Conclusion

- Implementation of PDR for software that is
  - Open source
  - Reproducible
  - Competitive
  - Integrated in CPAchecker
- Reference implementation for comparison
- Usable as invariant generator
- Interesting example programs
- Artifact available [3]

References I

- Beyer, D.: Automatic verification of C and Java programs: SV-COMP 2019. In: Proc. TACAS (3). pp. 133–155. LNCS 11429, Springer (2019). https://doi.org/10.1007/978-3-030-17502-3_9
- Beyer, D., Dangl, M.: Software verification with PDR: Implementation and empirical evaluation of the state of the art (August 2019), http://arxiv.org/abs/1908.06271
- Beyer, D., Dangl, M.: Replication package for article 'Software verification with PDR: An implementation of the state of the art'. Zenodo (2020). https://doi.org/10.5281/zenodo.3678766
- Beyer, D., Dangl, M., Wendler, P.: Boosting k-induction with continuously-refined invariants. In: Proc. CAV. pp. 622–640. LNCS 9206, Springer (2015). https://doi.org/10.1007/978-3-319-21690-4_42
- Beyer, D., Dangl, M., Wendler, P.: A unifying view on SMT-based software verification. J. Autom. Reasoning **60**(3), 299–335 (2018). https://doi.org/10.1007/s10817-017-9432-6
- Birgmeier, J., Bradley, A.R., Weissenbacher, G.: Counterexample to induction-guided abstraction-refinement (CTIGAR). In: Proc. CAV. pp. 831–848. LNCS 8559, Springer (2014). https://doi.org/10.1007/978-3-319-08867-9_55

References II

- Bradley, A.R.: SAT-based model checking without unrolling. In: Proc. VMCAI. pp. 70–87. LNCS 6538, Springer (2011).
  https://doi.org/10.1007/978-3-642-18275-4_7
- Bradley, A.R., Manna, Z.: Property-directed incremental invariant generation. Formal Asp. Comput. **20**(4-5), 379–405 (2008).
  https://doi.org/10.1007/s00165-008-0080-9
- Cimatti, A., Griggio, A.: Software model checking via IC3. In: Proc. CAV. pp. 277–293. LNCS 7358, Springer (2012).
  https://doi.org/10.1007/978-3-642-31424-7_23
- Günther, H., Laarman, A., Weissenbacher, G.: Vienna Verification Tool: IC3 for parallel software (competition contribution). In: Proc. TACAS. pp. 954–957. LNCS 9636, Springer (2016)
- Gurfinkel, A., Kahsai, T., Komuravelli, A., Navas, J.A.: The SeaHorn verification framework. In: Proc. CAV. pp. 343–361. LNCS 9206, Springer (2015). https://doi.org/10.1007/978-3-319-21690-4_20
- Gurfinkel, A., Kahsai, T., Navas, J.A.: SeaHorn: A framework for verifying C programs (competition contribution). In: Proc. TACAS. pp. 447–450. LNCS 9035, Springer (2015).
  https://doi.org/10.1007/978-3-662-46681-0_41

References III

- Jovanovic, D., Dutertre, B.: Property-directed k-induction. In: Proc. FMCAD. pp. 85–92. IEEE (2016). https://doi.org/10.1109/FMCAD.2016.7886665
- Lange, T., Prinz, F., Neuhäußer, M.R., Noll, T., Katoen, J.: Improving generalization in software IC3. In: Proc. SPIN'18. pp. 85–102. LNCS 10869, Springer (2018). https://doi.org/10.1007/978-3-319-94111-0_5