Precision Reuse in CPAchecker *

Dirk Beyer [1], Stefan Löwe [1], Evgeny Novikov [2], Andreas Stahlbauer [1], and Philipp Wendler [1]

[1] University of Passau, Innstr. 33, 94032 Passau, Germany
[2] ISP RAS, A. Solzhenitsyn St. 25, 109004 Moscow, Russia

Abstract: Continuous testing during development is a well-established technique for software-quality assurance. Continuous model checking from revision to revision is not yet established as a standard practice, because the enormous resource consumption makes its application impractical. Model checkers compute a large number of verification facts that are necessary for verifying if a given specification holds. We have identified a category of such intermediate results that are easy to store and efficient to reuse: *abstraction precisions*. The precision of an abstract domain specifies the level of abstraction that the analysis works on. Precisions are thus a precious result of the verification effort and it is a waste of resources to throw them away after each verification run. In particular, precisions are reasonably small and thus easy to store; they are easy to process and have a large impact on resource consumption. We experimentally show the impact of precision reuse on industrial verification problems created from 62 Linux kernel device drivers with 1 119 revisions.

Overview

Verification tools spend much effort on computing intermediate results that are needed to check if the specification holds. In most uses of model checking, these intermediate results are erased after the verification process — wasting precious information (in failing and succeeding runs). There are several directions to reuse (intermediate) results [BW13]. *Conditional model checking* [BHKW12] outputs partial verification results for later re-verification of the same program by other verification approaches. *Regression verification* [HJMS03, SG08, HKM+96] outputs intermediate results (or checks differences) for re-verification of a changed program by the same verification approach.

In program analysis, e.g., predicate analysis, shape analysis, or interval analysis, the respective abstract domain defines the kind of abstraction that is used to automatically construct the abstract model. The *precision* for an abstract domain defines the level of abstraction in the abstract model, for example, which predicates to track in predicate analysis [BHT08], or which pointers to track in shape analysis [BHT06]. Such precisions can be obtained automatically; interpolation is an example for a technique that extracts precisions for predicate analysis from infeasible error paths.

We propose to reuse precisions as intermediate verification results. Precisions are costly to compute and represent precious intermediate verification results. We treat these abstraction precisions as reusable verification facts, because precisions are easy to extract from model checkers that automatically construct an abstract model of the program (e.g., CEGAR), have a small memory footprint, are tool-independent, and are easy to use for regression verification because they are rather insensitive to changes in the program source code (compared to previous approaches).

*This is a summary of a full article on this topic that appeared in Proc. ESEC/FSE 2013 [BLN+13].

The technical insight of our work is that reusing precisions drastically reduces the number of refinements. The effort spent on analyzing spurious counterexamples and re-exploring the abstract state space in search for a suitable abstract model is significantly reduced.
We implemented precision reuse in the open-source verification framework CPACHECKER[1] [BK11] (a supplementary web page is also available[2]) and confirmed the effectiveness and efficiency (significant impact in terms of performance gains and increased number of solvable verification tasks) of our approach with an extensive experimental study on industrial code. The benchmark verification tasks were extracted from the Linux kernel, which is an important application domain [BP12], and prepared for verification using the LDV toolkit [MMN+12]. Our study consisted of a total of 16 772 verification runs for 4 193 verification tasks that are available online[3], composed from a total of 1 119 revisions (spanning more than 5 years) of 62 Linux drivers from the Linux-kernel repository.
Precision reuse is applicable to all verification approaches that are based on abstraction and automatically computing the precision of the abstract model (including CEGAR). Both the efficiency and effectiveness of such approaches can be increased by reusing precisions.
As a result of our experiments, a previously unknown bug in the Linux kernel was discovered by the LDV team, and a fix was submitted to and accepted by the maintainers[4].

References

[BHKW12] D. Beyer, T. A. Henzinger, M. E. Keremoglu, P. Wendler. Conditional Model Checking: A Technique to Pass Information between Verifiers. In *Proc. FSE*. ACM, 2012.

[BHT06] D. Beyer, T. A. Henzinger, and G. Théoduloz. Lazy Shape Analysis. In *Proc. CAV*, LNCS 4144, pages 532–546. Springer, 2006.

[BHT08] D. Beyer, T. A. Henzinger, and G. Théoduloz. Program Analysis with Dynamic Precision Adjustment. In *Proc. ASE*, pages 29–38. IEEE, 2008.

[BK11] D. Beyer and M. E. Keremoglu. CPACHECKER: A Tool for Configurable Software Verification. In *Proc. CAV*, LNCS 6806, pages 184–190. Springer, 2011.

[BLN+13] D. Beyer, S. Löwe, E. Novikov, A. Stahlbauer, and P. Wendler. Precision reuse for efficient regression verification. In *Proc. ESEC/FSE*, pages 389–399. ACM, 2013.

[BP12] D. Beyer and A. K. Petrenko. Linux Driver Verification. In *Proc. ISoLA*, LNCS 7610, pages 1–6. Springer, 2012.

[BW13] D. Beyer and P. Wendler. Reuse of Verification Results - Conditional Model Checking, Precision Reuse, and Verification Witnesses. In *Proc. SPIN*, pages 1–17, 2013.

[HJMS03] T. A. Henzinger, R. Jhala, R. Majumdar, and M. A. A. Sanvido. Extreme model checking. In *Proc. Verification: Theory and Practice*, pages 332–358. Springer, 2003.

[HKM+96] R. H. Hardin, R. P. Kurshan, K. L. McMillan, J. A. Reeds, and N. J. A. Sloane. Efficient Regression Verification. In *Proc. WODES*, pages 147–150, 1996.

[MMN+12] M. U. Mandrykin, V. S. Mutilin, E. M. Novikov, A. V. Khoroshilov, and P. E. Shved. Using Linux device drivers for static verification tools benchmarking. *Programming and Computer Software*, 38(5):245–256, 2012.

[SG08] O. Strichman and B. Godlin. Regression Verification — A Practical Way to Verify Programs. In *Proc. Verified Software: Theories, Tools, Experiments*, pages 496–501. Springer, 2008.

[1] `http://cpachecker.sosy-lab.org`
[2] `http://www.sosy-lab.org/~dbeyer/cpa-reuse/`
[3] `http://www.sosy-lab.org/~dbeyer/cpa-reuse/regression-benchmarks/`
[4] `https://patchwork.kernel.org/patch/2204681/`