Empirical Software Metrics for Benchmarking of Verification Tools

Yulia Demyanova  Thomas Pani  Helmut Veith und Florian Zuleger

Abstract: In recent work [De15, PVZ15, DVZ13], we study empirical metrics for software (SW) source code, which can predict the performance of verification tools on specific types of SW. Our metrics comprise variable usage patterns, loop patterns, as well as indicators of control-flow complexity and are extracted by simple data-flow analyses. We demonstrate that our metrics are powerful enough to devise a machine-learning based portfolio solver for SW verification. We show that this portfolio solver would be the (hypothetical) overall winner of both the 2014 and 2015 International Competition on Software Verification (SV-COMP). This gives strong empirical evidence for the predictive power of our metrics and demonstrates the viability of portfolio solvers for SW verification.

Keywords: Software verification, software metrics, portfolio solver, machine learning.

A modern verification tool needs to pick and choose how to combine a multitude of methods from the fields of model checking, static analysis, shape analysis, SAT solving, SMT solving, abstract interpretation, termination analysis, pointer analysis etc. The trade-offs are based on both technical and pragmatic aspects: many tools are either optimized for specific application areas (e.g. device drivers), or towards the in-depth development of a technique for a restricted program model (e.g. termination for integer programs).

In [De15] we demonstrate that the results of the annual International Competition on Software Verification (SV-COMP) [Be15] can be explained by intuitive metrics on the source code. In fact, the metrics are strong enough to enable us to construct a portfolio solver which would (hypothetically) win SV-COMP 2014 and 2015. Here, a portfolio solver is a SW verification tool that uses heuristic preprocessing to select one of the existing tools.

As an approach to SW verification, portfolio solving brings interesting advantages: (1) a portfolio solver optimally uses available resources, (2) it can avoid incorrect results of partially unsound tools, (3) machine learning in combination with portfolio solving allows us to select between multiple versions of the same tool with different runtime parameters, (4) the portfolio solver gives good insight into the state-of-the-art in SW verification.

To choose the SW metrics, we consider the zoo of techniques discussed above along with their target domains, our intuition as programmers, as well as the tool developer reports in their competition contributions. The obtained metrics are naturally understood in three dimensions: program variables, program loops, and control flow.

In [De15, PVZ15, DVZ13] we describe metrics which correspond to these dimensions, and are based on simple data-flow analyses. Our algorithm for the portfolio is based on

1 TU Wien, Arbeitsbereich Formal Methods in Systems Design, Institut für Informationssysteme 184/4, Favoritenstraße 9–11, 1040 Wien, Austria
machine learning using support vector machines (SVMs) [CV95] over the metrics defined above. Figure 1 depicts our experimental results on SV-COMP’15: Our tool TP is the overall winner and outperforms all other tools.

|          | blast | cascade | cbmc   | cpa-checker | predator-hp | smack   | ultimate-kojak | ulcseq | TP    |
|----------|-------|---------|--------|-------------|-------------|---------|----------------|--------|-------|
| Overall  | 737   | 806     | 684    | 2228        | 389         | 1542    | 1215           | 273    | 2511  |
|          | 4546  | 5146    | 11936  | 6288        | 96          | 8727    | 12563          | 6260   |
| Medals   | 1/0/0 | 0/0/0   | 1/1/1  | 2/1/5       | 1/0/1       | 2/1/1   | 0/2/0          | 0/0/0  | 1/6/1 |

Fig. 1: Experimental results for the eight best competition participants in SV-COMP’15 Overall, plus our portfolio TP, given as arithmetic mean of 10 experiments on randomly selected 40% subsets chosen for testing. The first row shows the Overall SV-COMP score and beneath it the runtime in minutes. We highlight the Overall gold, silver, and bronze medal in dark gray, light gray and white+bold font, respectively. The second row shows the number of gold/silver/bronze medals won in individual categories.

While portfolio solvers are important, we also think that the SW metrics we define in this work are interesting in their own right. Our results show that categories in SV-COMP have characteristic metrics. Thus, the metrics can be used to 1) characterize benchmarks not publicly available, 2) understand large benchmarks without manual inspection, 3) understand presence of language constructs in benchmarks.

Summarizing, our work makes the following contributions:

- We define software metrics along the three dimensions – program variables, program loops and control flow – in order to capture the difficulty of program analysis tasks.
- We develop a machine-learning based portfolio solver for software verification that learns the best-performing tool from a training set.
- We experimentally demonstrate the predictive power of our software metrics in conjunction with our portfolio solver on the software verification competitions SV-COMP’14 and SV-COMP’15.

References

[Be15] Beyer, Dirk: Software Verification and Verifiable Witnesses - (Report on SV-COMP 2015). In: TACAS. pp. 401–416, 2015.

[CV95] Cortes, Corinna; Vapnik, Vladimir: Support-vector networks. Machine learning, 20(3):273–297, 1995.

[De15] Demyanova, Yulia; Pani, Thomas; Veith, Helmut; Zuleger, Florian: Empirical Software Metrics for Benchmarking of Verification Tools. In: CAV. pp. 561–579, 2015.

[DVZ13] Demyanova, Yulia; Veith, Helmut; Zuleger, Florian: On the Concept of Variable Roles and its Use in Software Analysis. In: FMCAD. pp. 226–229, 2013.

[PVZ15] Pani, Thomas; Veith, Helmut; Zuleger, Florian: Loop Patterns in C Programs. ECEASST, 72, 2015.