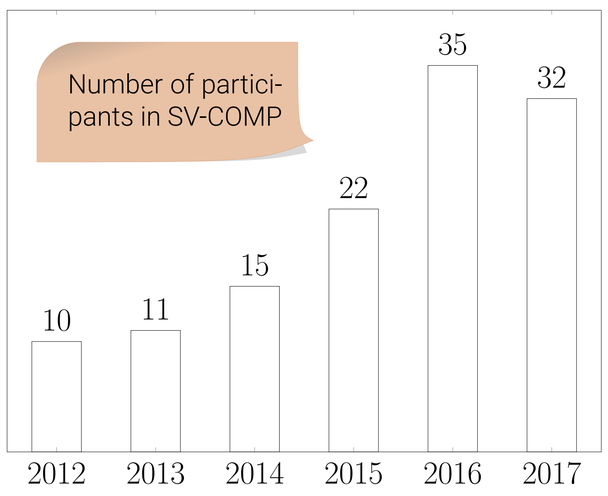
FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

for(syte
Formal Methods in Systems Engineering

Empirical Software Metrics for Benchmarking of Verification Tools

Yulia Demyanova, **Thomas Pani**, Helmut Veith, Florian Zuleger
TU Wien

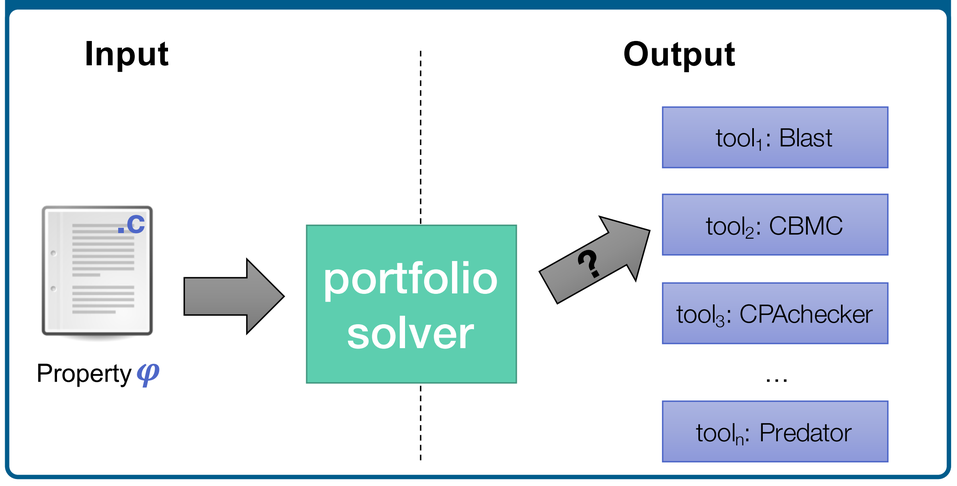
Evergrowing number of verification tools



But: Just Verify Software?

- Tools are are largely complementary – Focus on
  - specific application domains (e.g. device drivers)
  - restricted program models (e.g. termination of integer programs)
- Can't test all tools (setup effort, computational resources, …)

Idea: Portfolio



Portfolios

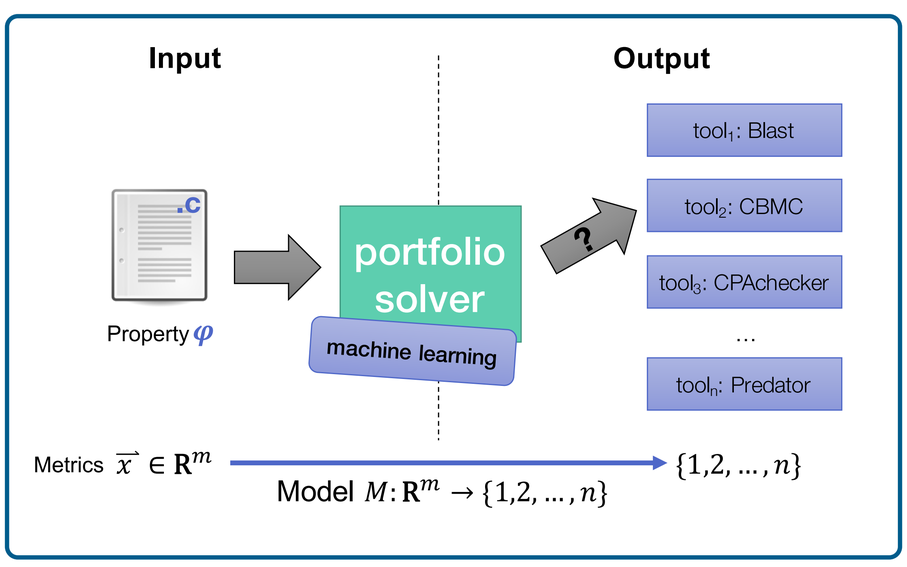
Successful (and controversial) in combinatorially cleaner domains:

- Boolean satisfiability (SAT)
- Quantified boolean satisfiability (QBF)
- Answer set programming (ASP)
- Various constraint satisfaction problems

Benefits of portfolios

- Optimally uses available resources
- Avoids incorrect results of partially unsound tools
- Selection between multiple versions of the same tool (via commandline flags)
- Gives insight in to the state-of-the-art by combining complementary efforts

Portfolio Construction



Metrics

What doesn't work

Classical software metrics, such as LOC, number of decision points, …

Interesting stuff

Program Variables. Bit-precision or mathematical integers? `int`s used as indices, bit-masks or in arithmetic? Dynamic data structures? Arrays?

Program Loops. Reducible loops or goto programs? FOR-loops or advanced ranking functions? Widening, loop acceleration, termination analysis, or loop unrolling?

Control Flow. Recursion? Function pointers? Multithreading? Straight-line code or complex branching?

Challenge: Cheap but precise and descriptive metrics.

Variable Roles [1]

27 patterns of variable usage (~types).

```
int n = 0, y = x;
while (x) {
  n++;
  x = x & (x-1);
}

/* ----------------- */

int fd = open(path,
 flags);
int c, val=0;
while (read(fd, &c, 1) >
 0 && isdigit(c)) {
  val = 10*val + c-'0';
}
```

`n`: COUNTER
`x`: BITVECTOR

`fd`: FILE_DESCR
`val`: LINEAR_ARITH

Loop Patterns [3]

Identify structured iteration (enumerating a range) via 5 syntactic patterns.

```
while (i < 100) {
  if (nondet())
    i += 2;
  else
    i += 3;
  i--;
}
```

Overhead:
feature extraction: $\tilde{x} = 0.5$s
ML prediction: $\tilde{x} = 0.5$s

Control Flow

5 control-flow metrics

- CFG: basic blocks, max indegree
- function pointer calls / args
- recursive function calls

Results (SV-COMP'16)

	cpa-bam	cpa-kind	cpa-refsel	cpa-seq	esbmc	esbmc-depthk	smack	uautomizer	VeriFolio
Overall	898 11775	1678 12587	1151 10240	**1907** **12509**	1699 8396	1283 9920	1684 14218	**1965** **11210**	**3269** **8544**
Medals	0/0/0	0/1/1	1/0/0	2/1/2	0/2/0	0/0/0	0/0/1	0/2/0	2/1/3

Ongoing & Future Work

Conceptual understanding, design, and improvement

- understand the trained ML model (e.g., feature selection, …)
- generalize the model (e.g., predict runtime, memory, k "best" tools, …)
- further ML algorithms, in particular deep neural networks

Guiding lower level heuristics (command line flags)

- predicate abstraction [5], bit-precision, …

Practical improvements

- update to SV-COMP'17, extend feature set, compare features

References

[1] Y. Demyanova, H. Veith, and F. Zuleger. "On the concept of variable roles and its use in software analysis". In: *FMCAD*. IEEE, 2013, pp. 226–230.
[2] Y. Demyanova, T. Pani, H. Veith, and F. Zuleger. "Empirical Software Metrics for Benchmarking of Verification Tools". In: *CAV (1)*. Vol. 9206. LNCS. Springer, 2015, pp. 561–579.
[3] T. Pani, H. Veith, and F. Zuleger. "Loop Patterns in C Programs". In: *ECEASST* 72 (2015).
[4] Y. Demyanova, T. Pani, H. Veith, and F. Zuleger. "Empirical software metrics for benchmarking of verification tools". In: *Formal Methods in System Design* 50.2-3 (2017), pp. 289–316.
[5] Y. Demyanova, P. Rümmer, and F. Zuleger. "Systematic Predicate Abstraction Using Variable Roles". In: *NFM*. Vol. 10227. LNCS. 2017, pp. 265–281.

pani@forsyte.at