Interpretation-based Violation Witness Validation for C code

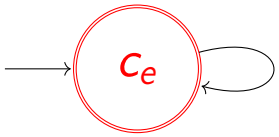
4th International Workshop on CPAchecker

Philipp Berger, Jan Svejda and Joost-Pieter Katoen

RWTH AACHEN UNIVERSITY

Witnesses

- similar to certificates in complexity theory $\Rightarrow$ they should ease verification
- two types are currently in use: violation and correctness witnesses
- violation witnesses (can) restrict the state-space, (can) provide resolution of non-determinism and (can) guide the search towards an error location



berger@cs.rwth-aachen.de

Motivation

Situation: testing model checkers on complex and large industry-sourced C code

- Conflicting results - now what?
  - Back then only two violation witness verifiers existed
  - We observed: if the tools can not verify the property alone, it can not verify the witness
  - Very time and memory intensive
  - Scenario: property violation detected by CBMC, correctness established by CPAchecker
    - Issue: witnesses by CBMC often not accepted by CPAchecker nor Ultimate (format)
  - Scenario: CPAchecker and Ultimate Automizer do not agree
    - Issue: When Ultimate and CPAchecker do not agree - do I trust their witness verification?
  - Some of the tools output a significant amount of `unknown` results

The NITWIT Validator

`iNterpretation-based vIolaTion WITness` Validator

It:

- is a new execution-based validator,
- explores only a single path through the program,
- is fast and memory efficient and
- is applicable to a large variety of C programs and witnesses from different verifiers.

The NITWIT Validator

iNterpretation-based vIolaTion WITness Validator

NITWIT T-Shirts available soon!

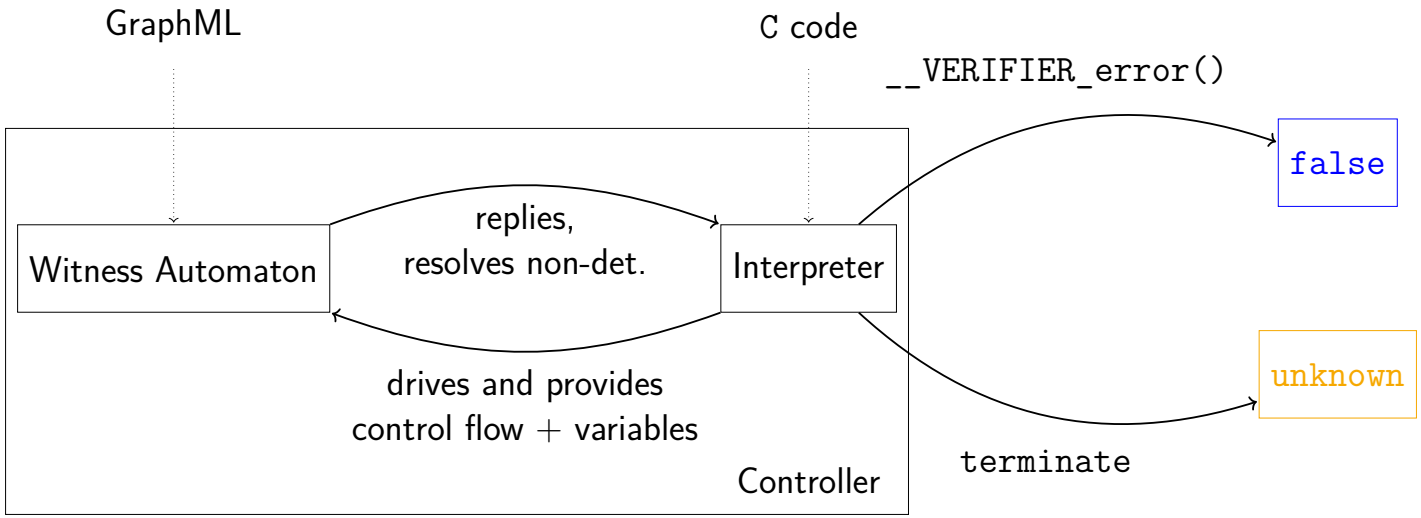
It:

- is a new execution-based validator,
- explores only a single path through the program,
- is fast and memory efficient and
- is applicable to a large variety of C programs and witnesses from different verifiers.

berger@cs.rwth-aachen.de

The NITWIT Validator - Implementation A

- The interpreter explores a path while simultaneously feeding control flow locations and variable values into an internal representation of the witness automaton.
- The witness automaton in turn resolves nondeterministic variables if appropriate assumptions in the witness are available.



The NITWIT Validator - Implementation B

Checking assumptions: Easy with interpreters!

1. Back up the stack and heap,
2. execute the state-space guard in the context of the program,
3. check whether non-deterministic variables were resolved and
4. restore the stack and heap.

berger@cs.rwth-aachen.de

Extending PicoC

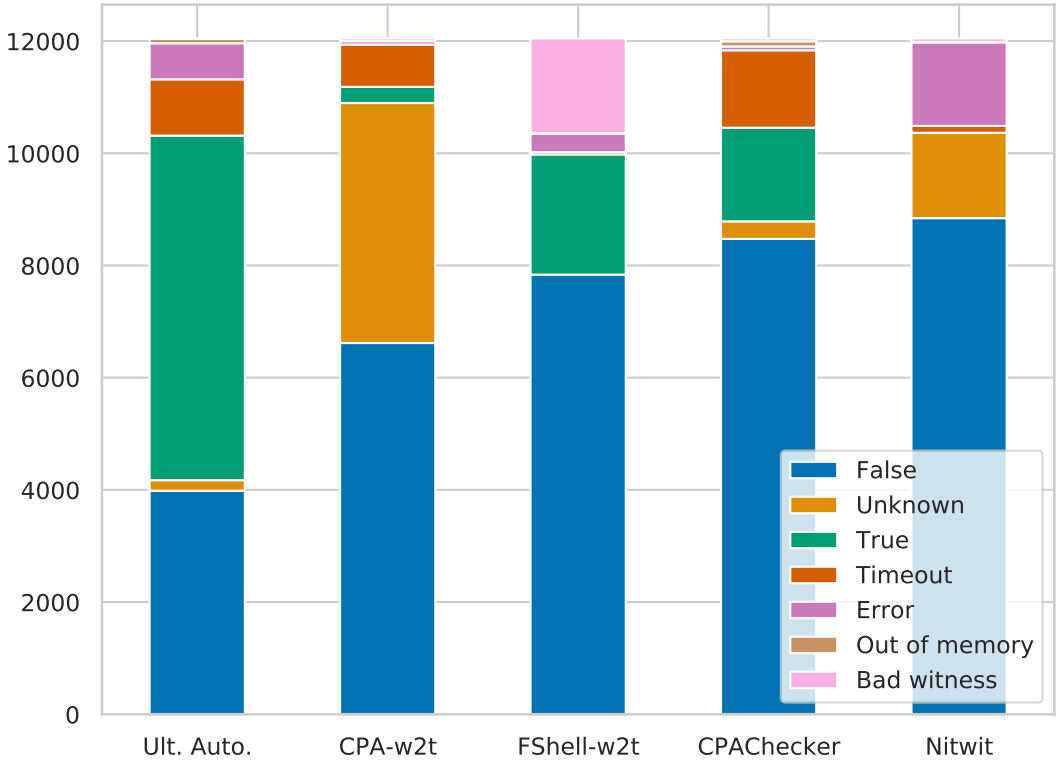
Most of the work consisted in extending PicoC for our purposes and in adding new features to support the required parts of C99:

- callbacks to witness automaton at important program locations,
- ND types + functions,
- `goto` constructs,
- function pointers,
- distinguishing `double`/`float` types + operations,
- `long longs`, fixed many bugs in integer arithmetic,
- `const` types (partially), proper parsing of numerical constants,
- variable shadowing,
- `struct` initialization,
- bit fields
- ...

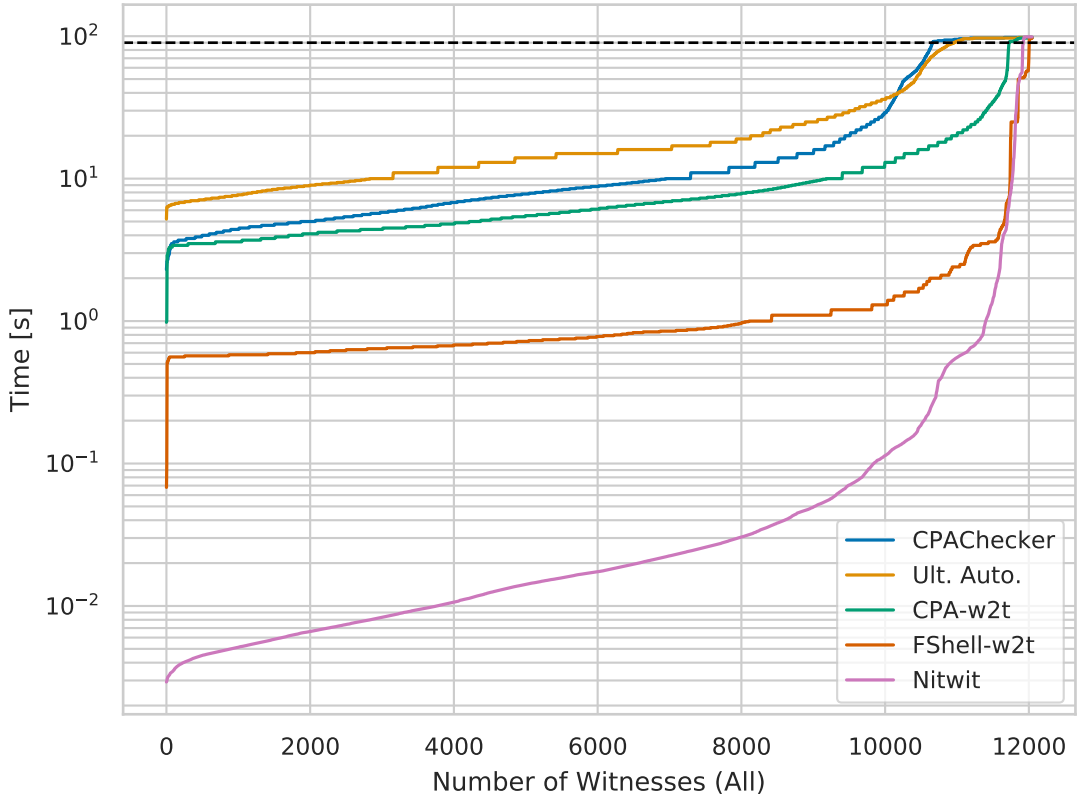
berger@cs.rwth-aachen.de

Results on the SV-COMP '19 dataset (including other validators)

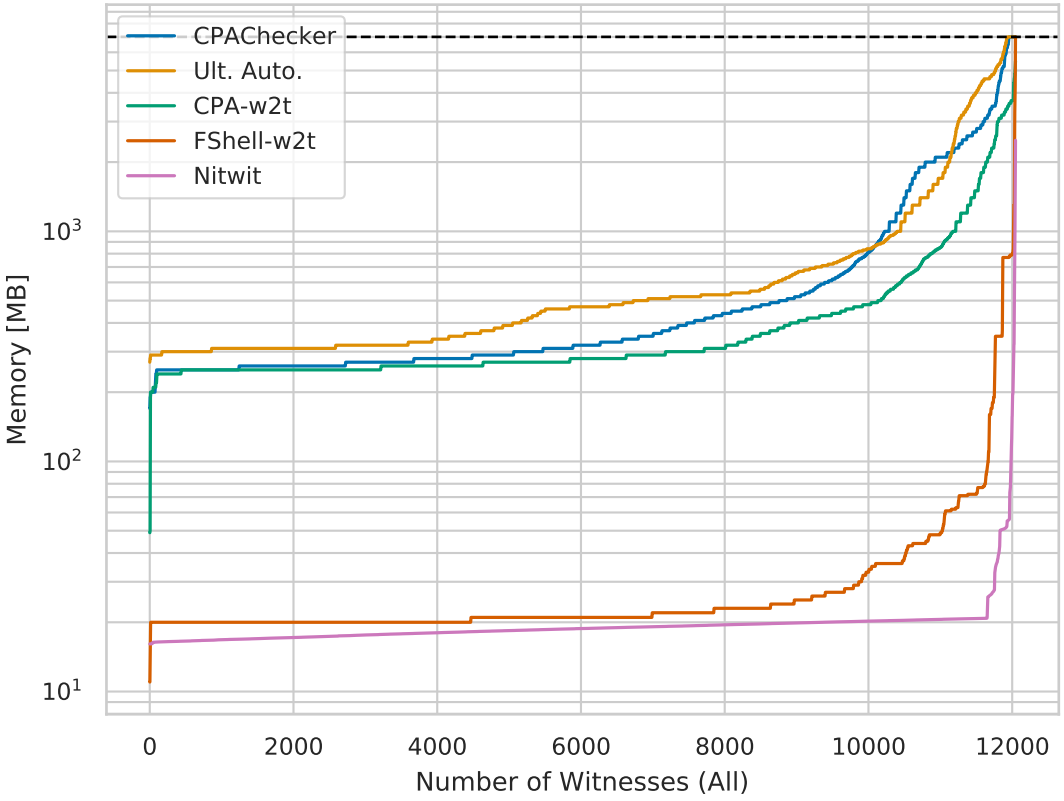
berger@cs.rwth-aachen.de



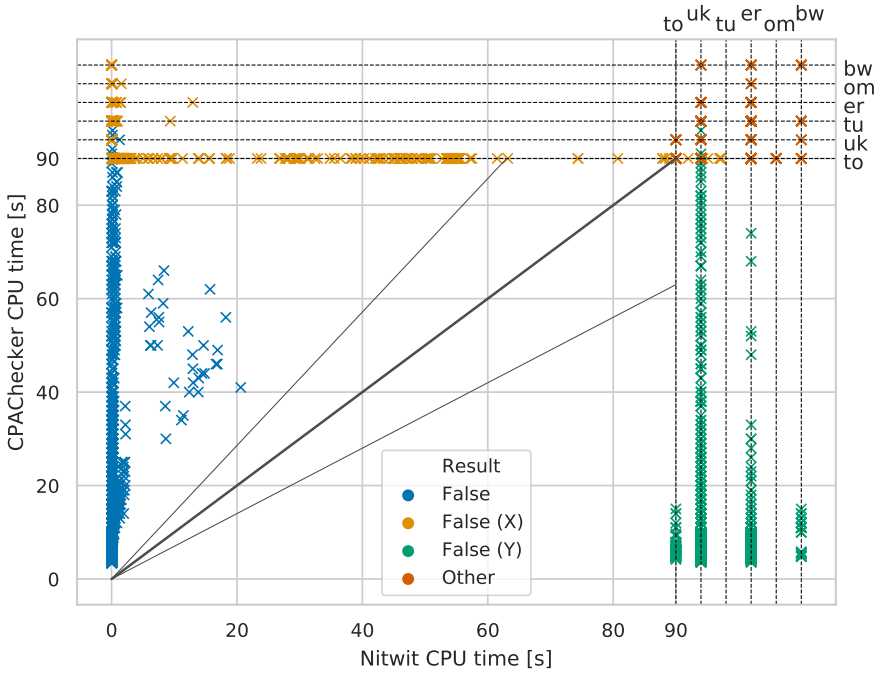
berger@cs.rwth-aachen.de



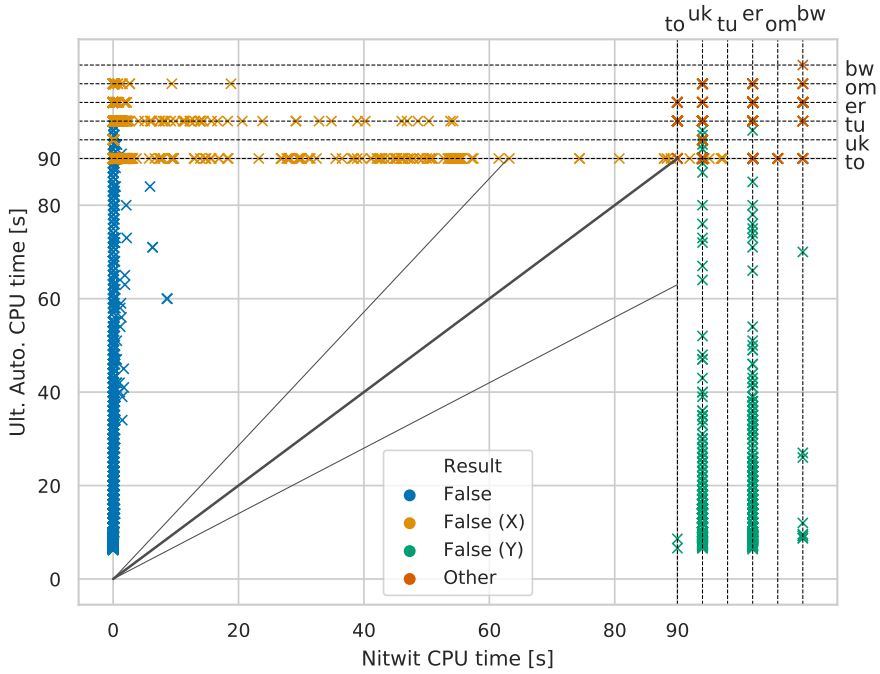
berger@cs.rwth-aachen.de



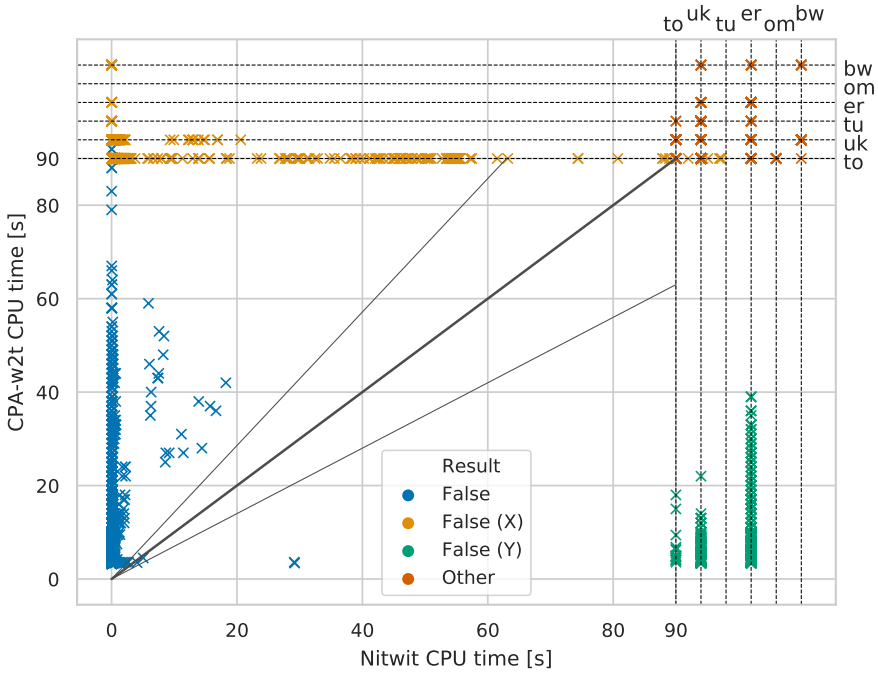
berger@cs.rwth-aachen.de



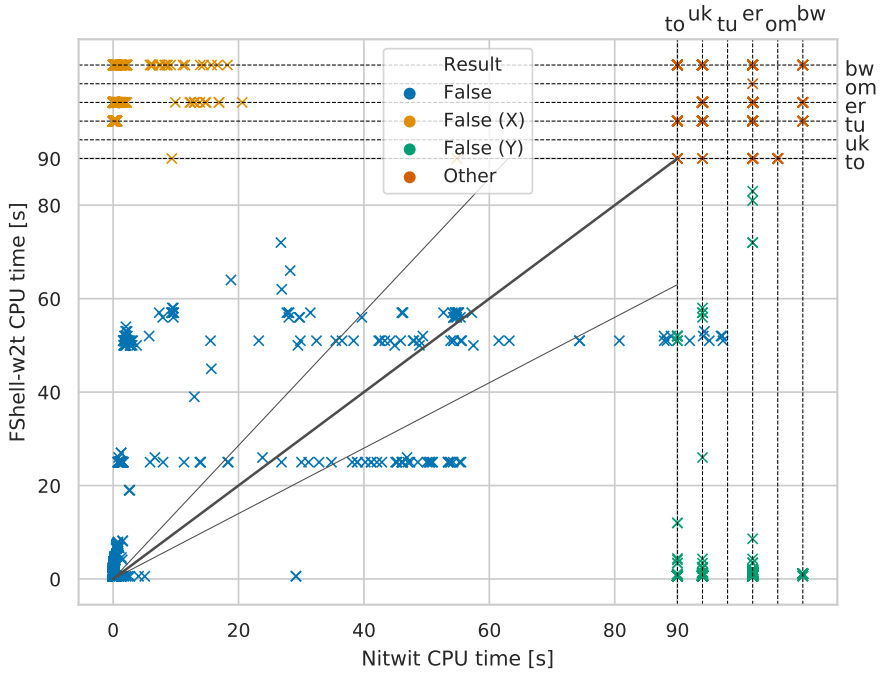
(a) CPAChecker



(b) Ultimate Automizer

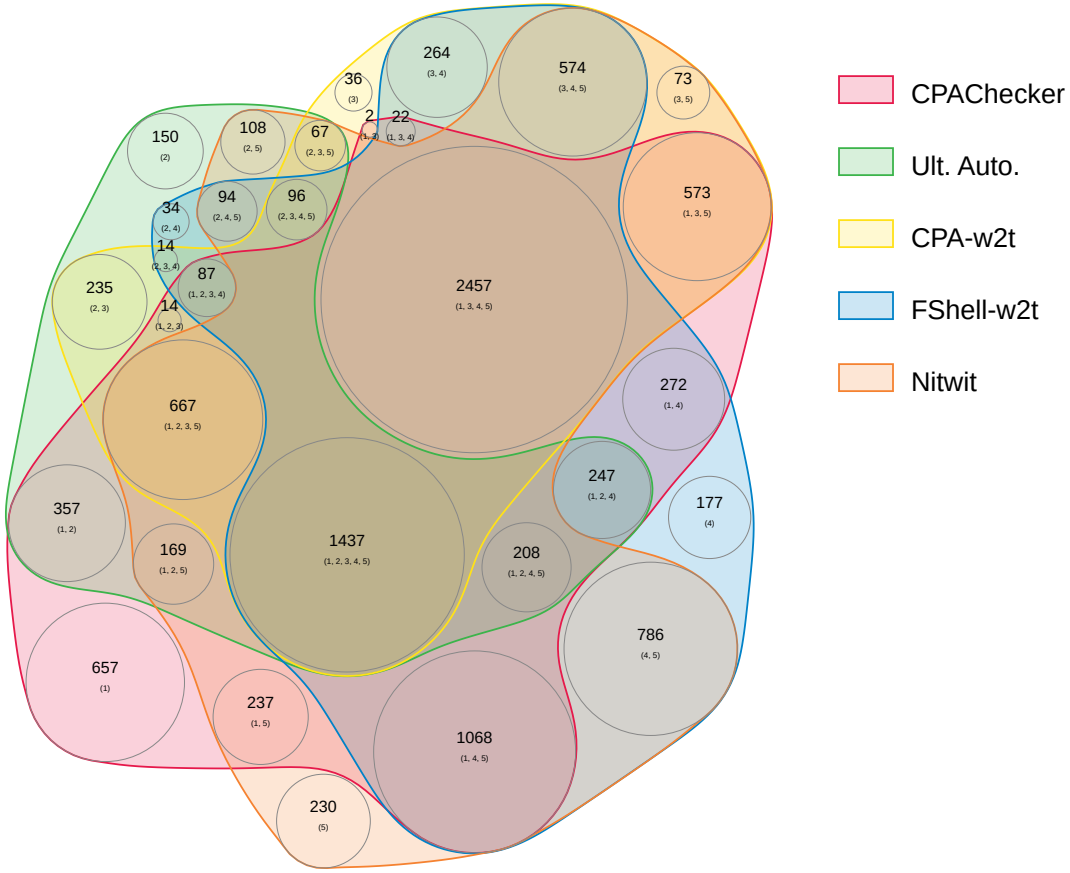


(c) CPA-witness2test



(d) FShell-witness2test

berger@cs.rwth-aachen.de



berger@cs.rwth-aachen.de

Succ. validations per verifier	CPAchecker	Ult. Auto.	CPA-w2test	FShell-w2t	Nitwit	Virt. best	In total
CBMC	458	323	486	537	**577**	714	804
CBMC-Path	110	125	227	240	**263**	323	331
CPA-Seq	**1038**	692	853	431	1008	1109	1123
DIVINE-SMT	**185**	15	0	121	84	192	205
DIVINE-explicit	120	21	97	**127**	105	141	145
DepthK	493	80	434	**617**	614	755	778
ESBMC-kind	599	76	579	**810**	736	950	970
Map2Check	75	32	100	**149**	115	153	160
PeSCo	**1011**	655	785	532	941	1045	1062
Pinaka	594	420	421	**630**	622	659	670
PredatorHP	**57**	38	13	46	22	68	71
SMACK	**423**	13	375	353	395	521	622
Skink	86	35	92	**96**	82	104	105
Symbiotic	**801**	13	398	455	538	877	1099
UAutomizer	267	**419**	214	185	145	471	483
UKojak	264	**276**	42	158	132	321	324
UTaipan	260	**380**	183	183	141	427	438
VIAP	44	32	43	**82**	52	100	112
VeriAbs	549	295	399	1000	**1070**	1220	1274
VeriFuzz	1040	44	877	1085	**1202**	1262	1271
Total	8474	3984	6618	7837	**8844**	11412	12047

Result summary

A dataset of 12 047 witnesses in total (all `C` programs from the ReachSafety category)

- 8844 witnesses validated by us
- Time (only validated): on average 0.79s, median 0.017s, std. deviation 6.04s. In total about 7000s.
- Time (all): on average 1.69s, median 0.017s, std. deviation 11.12. In total about 20 000s.
- Memory (all): on average 21.6 MB, median 18.8 MB, std. deviation: 47.0 MB.

Our contribution

- The NITWIT Validator is open-source and free to use:
  `https://github.com/moves-rwth/nitwit-validator`
- It is fast, memory efficient and, compared to SV-COMP '19, finds the most violations for witnesses from the *ReachSafety* category.
- It was developed independently of existing model checkers.
- It borrows C semantics from your favorite compiler!
- It supports 32-bit and 64-bit programs.
- Users have some more compilation options for control of specific validation details.
- **Planned feature:** witness refinement

berger@cs.rwth-aachen.de