CoVeriTeam

On-Demand Composition of Cooperative Verification Systems

Dirk Beyer and Sudeep Kanav

LMU Munich, Germany

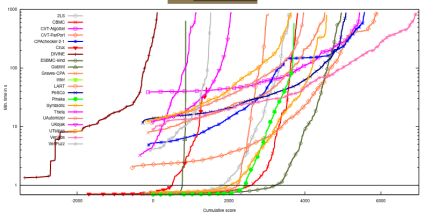
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

CPA

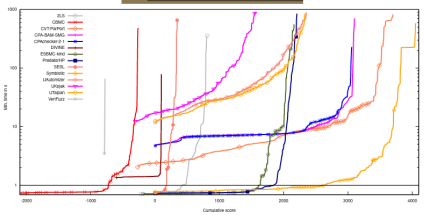
SoSy-Lab Software Systems

Many Verifiers with Different Strengths

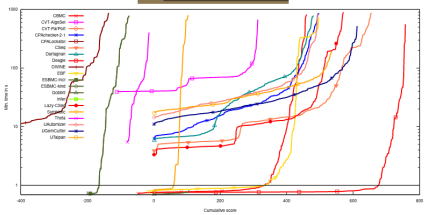
ReachSafety
1. VeriAbs
2. CPAchecker 2.1
3. PeSCo



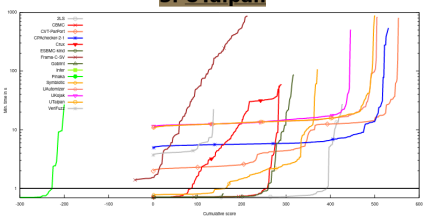
MemSafety
1. Symbiotic
2. CPA-BAM-SMG
3. CPAchecker 2.1



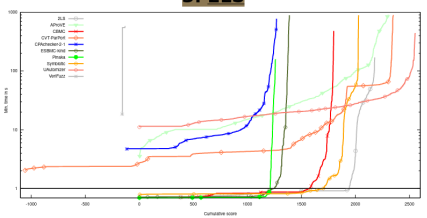
ConcurrencySafety
1. Deagle
2. CSeq
3. UGemCutter



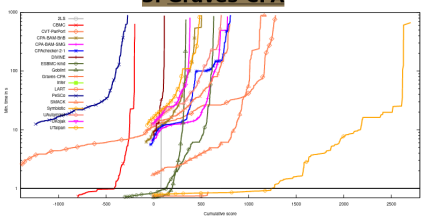
NoOverflows
1. CPAchecker 2.1
2. UAutomizer
3. UTaipan



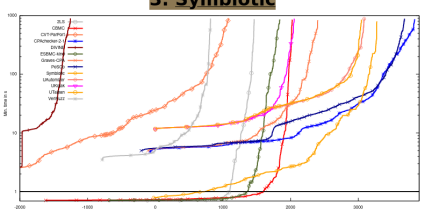
Termination
1. UAutomizer
2. AProVE
3. 2LS



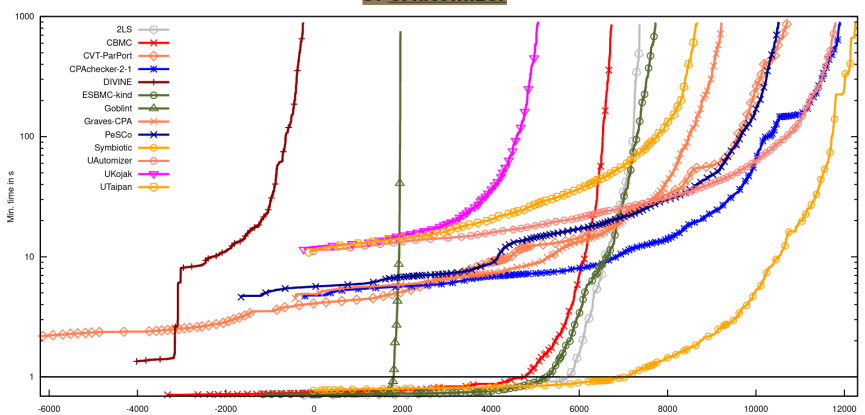
SoftwareSystems
1. Symbiotic
2. CPAchecker 2.1
3. Graves-CPA



FalsificationOverall
1. CPAchecker 2.1
2. PeSCo
3. Symbiotic



Overall
1. Symbiotic
2. CPAchecker 2.1
3. UAutomizer



https://sv-comp.sosy-lab.org/2022/results

Many Verifiers Combine Techniques

Verifier	CEGAR	Predicate Abstraction	Symbolic Execution	Bounded Model Checking	k-Induction	Property-Directed Reach.	Explicit-Value Analysis	Numeric. Interval Analysis	Shape Analysis	Separation Logic	Bit-Precise Analysis	ARG-Based Analysis	Lazy Abstraction	Interpolation	Automata-Based Analysis	Concurrency Support	Ranking Functions	Evolutionary Algorithms	Algorithm Selection	Portfolio
2LS				✓	✓			✓	✓		✓						✓			
APROVE			✓				✓	✓		✓	✓						✓			
BRICK	✓		✓	✓				✓								✓				
CBMC				✓							✓					✓				
COASTAL[∅]			✓																	
CVT-ALGOSEL[new∅]	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓
CVT-PARPORT[new∅]	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓		✓	✓		✓	✓
CPA-BAM-BNB[∅]	✓	✓					✓				✓	✓	✓	✓						
CPA-BAM-SMG[new]																				
CPALOCKATOR[∅]	✓	✓					✓				✓	✓	✓	✓		✓				
CPACHECKER	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓		✓	✓		✓	✓
CRUX[new]			✓																	
CSEQ				✓							✓					✓				
DARTAGNAN				✓							✓					✓				
DEAGLE[new]																				
DIVINE[∅]			✓				✓				✓					✓			✓	✓
EBF[new]				✓																
ESBMC-INCR[∅]				✓	✓						✓					✓				
ESBMC-KIND				✓	✓			✓			✓					✓				
FRAMA-C-SV								✓												
GAZER-THETA[∅]	✓	✓		✓			✓				✓	✓	✓	✓						✓
GDART[new]			✓								✓									✓
GOBLINT								✓								✓				
GRAVES-CPA[new]	✓	✓		✓	✓		✓	✓	✓		✓	✓	✓	✓		✓	✓		✓	✓

Competition Report [1]

https://doi.org/10.1007/978-3-030-99527-0_20

Examples of Tool Combinations from Literature

Technique	Year
Counterexample Checking [15]	2012
Conditional Model Checking [6]	2012
Precision Reuse [12]	2013
Witness Validation [3, 2]	2015, 2016
Execution-Based Validation [4]	2018
Reducer [8]	2018
CoVeriTest [7]	2019
CondTest [11]	2019
MetaVal [14]	2020

Motivation for CoVeriTeam

Tool combinations were developed *ad hoc*. We missed

- a conceptual framework for composing verification tools
- a common interface for tools
- mechanisms safeguarding from undesirable side effects of tool execution in the combinations

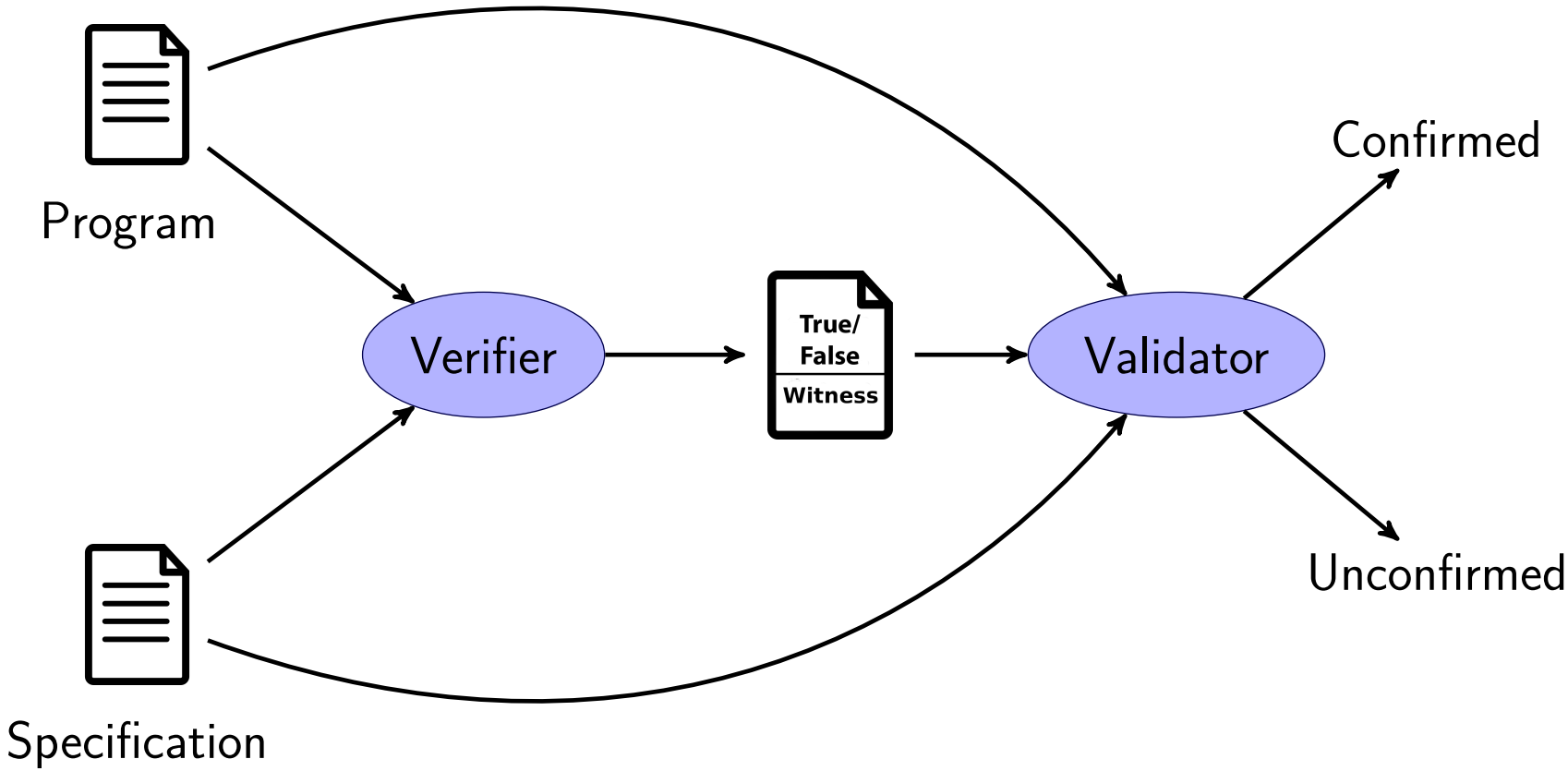
Motivation for CoVeriTeam

Tool combinations were developed *ad hoc*. We missed

- a conceptual framework for composing verification tools
- a common interface for tools
- mechanisms safeguarding from undesirable side effects of tool execution in the combinations

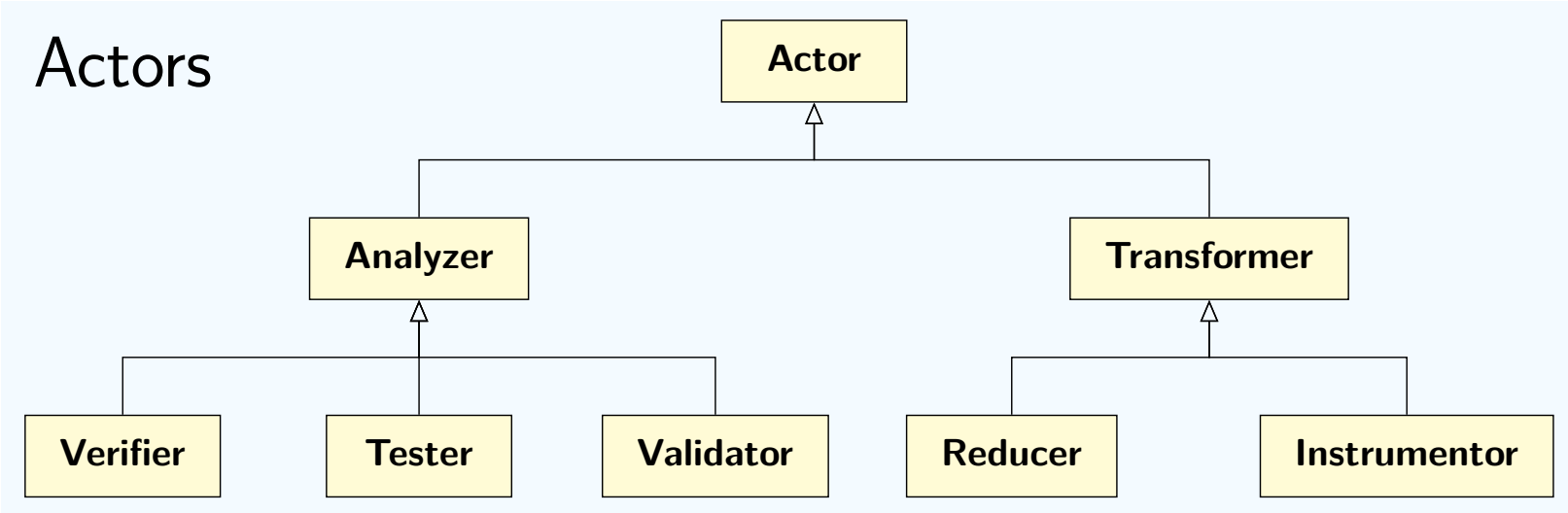
CoVeriTeam aims to make developing tool combinations systematic and easy to maintain.

Running Example

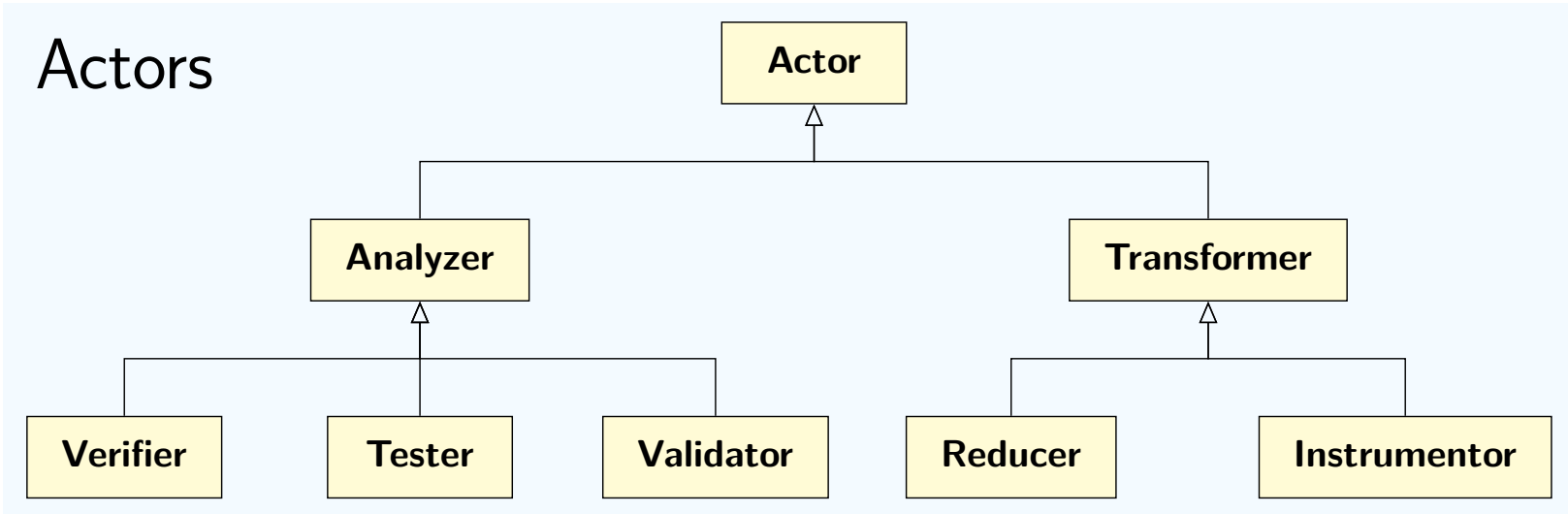


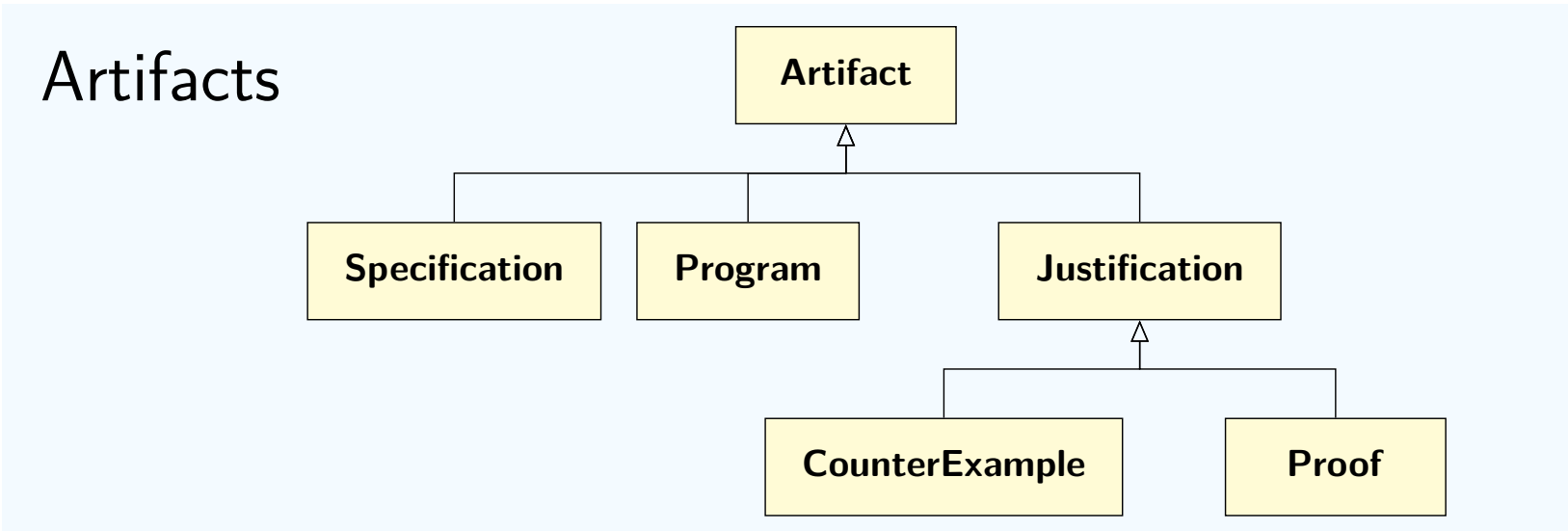
Witness validation

Simplified Domain Model

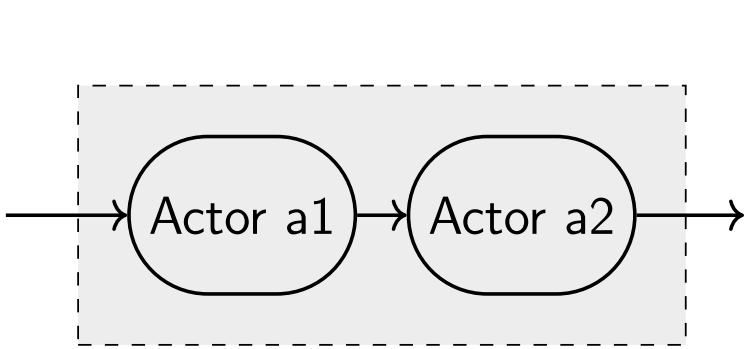


Simplified Domain Model

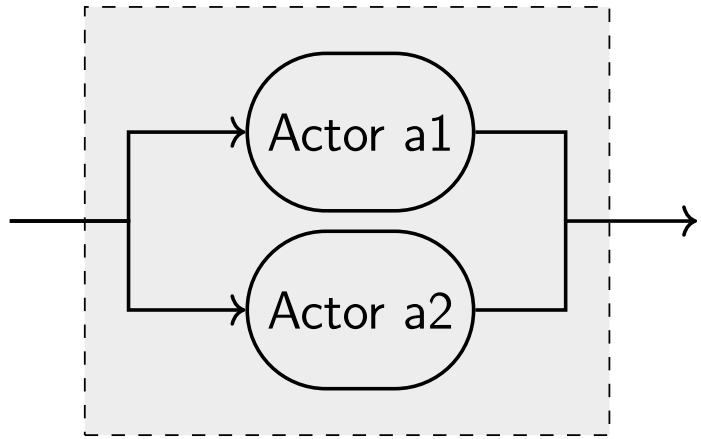




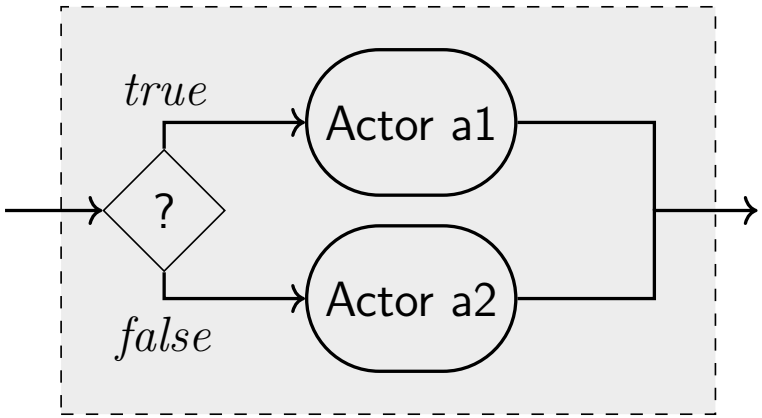
Compositions



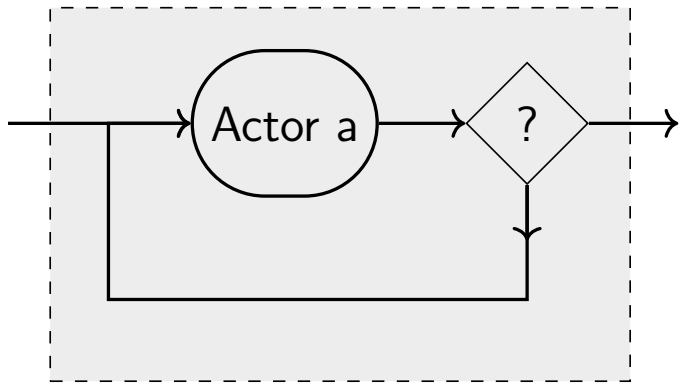
(a) SEQUENCE



(b) PARALLEL



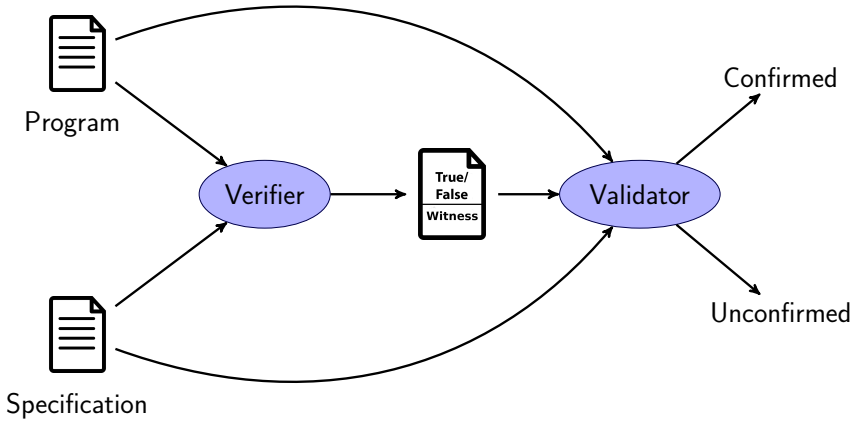
(c) ITE

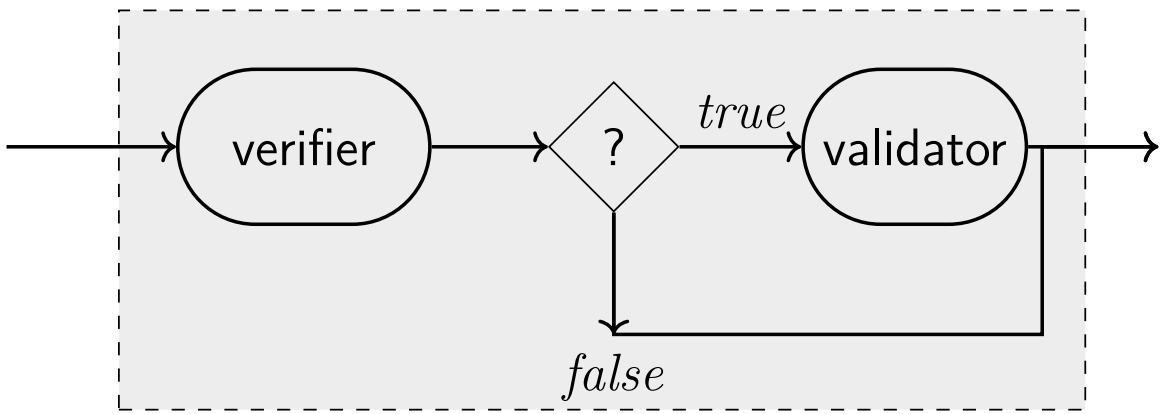


(d) REPEAT

Compositions in CoVeriTeam

Running Example





Running Example in CoVeriTeam

```
// Create actors
verifier = ...
validator = ...

// Prepare inputs
prog = ...
spec = ...
inputs = {'program':prog, 'spec':spec};

// Use validator if verdict is true or false
condition = ELEMENTOF(verdict, {TRUE, FALSE});
second_component = ITE(condition, validator);
// Verifier and second component to be executed in sequence
validating_verifier = SEQUENCE(verifier, second_component);

// Execute the new component on the inputs
res = execute(validating_verifier, inputs);
print(res);
```

Running Example in CoVeriTeam Language

Actor Execution

Actor Execution

Reusing features from BenchExec [13],

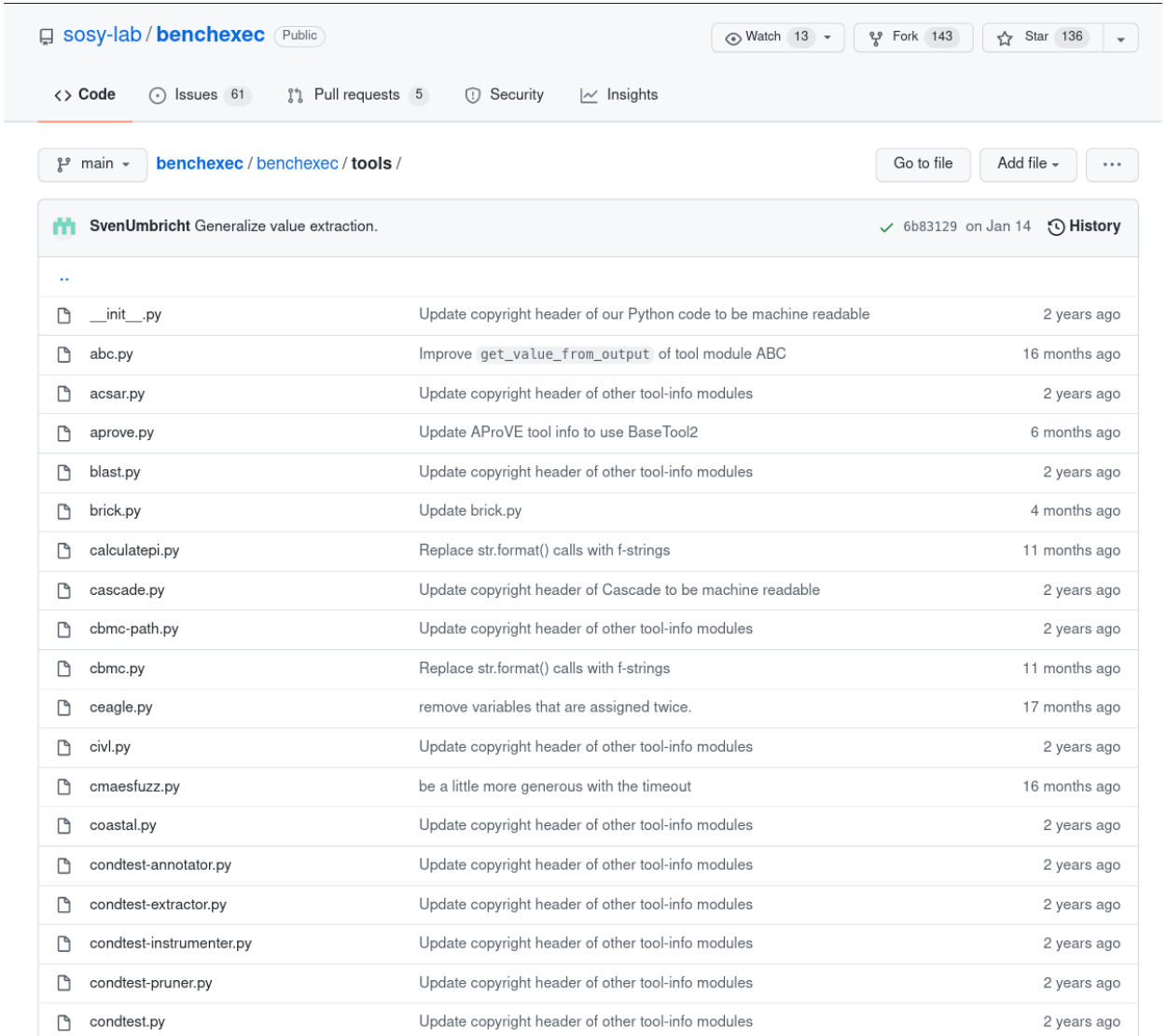
- tool-info modules
  - prepare the command for the tool
  - parse tool output

Actor Execution

Reusing features from BenchExec [13],

- tool-info modules
  - prepare the command for the tool
  - parse tool output
- RunExec
  - execute tools in isolation
  - measure resource consumption
  - enforce resource limits

Tool-info Module



sosy-lab / **benchexec** Public

Watch 13 | Fork 143 | Star 136

Code | Issues 61 | Pull requests 5 | Security | Insights

main | **benchexec** / benchexec / **tools** / | Go to file | Add file | ...

SvenUmbricht Generalize value extraction. 6b83129 on Jan 14 **History**

..		
__init__.py	Update copyright header of our Python code to be machine readable	2 years ago
abc.py	Improve `get_value_from_output` of tool module ABC	16 months ago
acsar.py	Update copyright header of other tool-info modules	2 years ago
aprove.py	Update AProVE tool info to use BaseTool2	6 months ago
blast.py	Update copyright header of other tool-info modules	2 years ago
brick.py	Update brick.py	4 months ago
calculatepi.py	Replace str.format() calls with f-strings	11 months ago
cascade.py	Update copyright header of Cascade to be machine readable	2 years ago
cbmc-path.py	Update copyright header of other tool-info modules	2 years ago
cbmc.py	Replace str.format() calls with f-strings	11 months ago
ceagle.py	remove variables that are assigned twice.	17 months ago
civl.py	Update copyright header of other tool-info modules	2 years ago
cmaesfuzz.py	be a little more generous with the timeout	16 months ago
coastal.py	Update copyright header of other tool-info modules	2 years ago
condtest-annotator.py	Update copyright header of other tool-info modules	2 years ago
condtest-extractor.py	Update copyright header of other tool-info modules	2 years ago
condtest-instrumenter.py	Update copyright header of other tool-info modules	2 years ago
condtest-pruner.py	Update copyright header of other tool-info modules	2 years ago
condtest.py	Update copyright header of other tool-info modules	2 years ago

Using tool-info modules gives us integration with many well known verification tools without any effort.

Actor Definition

```yaml
actor_name: uautomizer
toolinfo_module: "ultimateautomizer.py"
archives:
  - version: default
    location: "https://.../uautomizer.zip"
  - version: svcomp22
    doi: "10.5281/zenodo.5898989"
options: ['--full-output']
resourcelimits:
  memlimit: "8 GB"
  timelimit: "2 min"
format_version: '1.2'
```

YAML configuration for an atomic actor

Actor Definition

```yaml
actor_name: uautomizer
toolinfo_module: "ultimateautomizer.py"
archives:
  - version: default
    location: "https://.../uautomizer.zip"
  - version: svcomp22
    doi: "10.5281/zenodo.5898989"
options: ['--full-output']
resourcelimits:
  memlimit: "8 GB"
  timelimit: "2 min"
format_version: '1.2'
```

YAML configuration for an atomic actor

Actor Definition

```yaml
actor_name: uautomizer
toolinfo_module: "ultimateautomizer.py"
archives:
  - version: default
    location: "https://.../uautomizer.zip"
  - version: svcomp22
    doi: "10.5281/zenodo.5898989"
options: ['--full-output']
resourcelimits:
  memlimit: "8 GB"
  timelimit: "2 min"
format_version: '1.2'
```

YAML configuration for an atomic actor

Actor Definition

```yaml
actor_name: uautomizer
toolinfo_module: "ultimateautomizer.py"
archives:
  - version: default
    location: "https://.../uautomizer.zip"
  - version: svcomp22
    doi: "10.5281/zenodo.5898989"
options: ['--full-output']
resourcelimits:
  memlimit: "8 GB"
  timelimit: "2 min"
format_version: '1.2'
```

YAML configuration for an atomic actor

Running Example in CoVeriTeam

```
1  // Create actors
2  verifier = ActorFactory.create(ProgramVerifier,
       "actors/uautomizer.yml");
3  validator = ActorFactory.create(ProgramValidator,
       "actors/cpa-validate-violation-witnesses.yml");
4
5  // Prepare inputs
6  prog = ArtifactFactory.create(CProgram, prog_path);
7  spec = ArtifactFactory.create(BehaviorSpecification, spec_path);
8  inputs = {'program':prog, 'spec':spec};
9
10 // Use validator if verdict is true or false
11 condition = ELEMENTOF(verdict, {TRUE, FALSE});
12 second_component = ITE(condition, validator);
13 // Verifier and second component to be executed in sequence
14 validating_verifier = SEQUENCE(verifier, second_component);
15
16 // Execute the new component on the inputs
17 res = execute(validating_verifier, inputs);
18 print(res);
```

Running Example in CoVeriTeam Language

Running Example in CoVeriTeam

```
1  // Create actors
2  verifier = ActorFactory.create(ProgramVerifier,
       "actors/uautomizer.yml");
3  validator = ActorFactory.create(ProgramValidator,
       "actors/cpa-validate-violation-witnesses.yml");
4
5  // Prepare inputs
6  prog = ArtifactFactory.create(CProgram, prog_path);
7  spec = ArtifactFactory.create(BehaviorSpecification, spec_path);
8  inputs = {'program':prog, 'spec':spec};
9
10 // Use validator if verdict is true or false
11 condition = ELEMENTOF(verdict, {TRUE, FALSE});
12 second_component = ITE(condition, validator);
13 // Verifier and second component to be executed in sequence
14 validating_verifier = SEQUENCE(verifier, second_component);
15
16 // Execute the new component on the inputs
17 res = execute(validating_verifier, inputs);
18 print(res);
```

Running Example in CoVeriTeam Language

Evaluation Through Case Studies

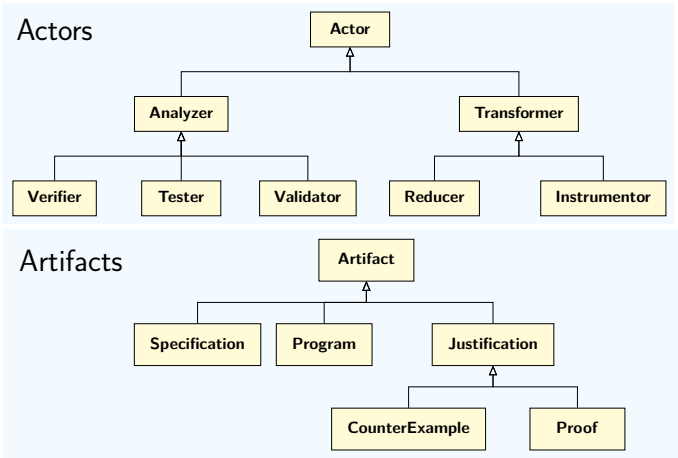
Technique	Year	Case Study	More Info
Counterexample Checking [15]	2012		
Conditional Model Checking [6]	2012		
Precision Reuse [12]	2013		
Witness Validation [3, 2]	2015, 2016	✓	
Execution-Based Validation [4]	2018	✓	
Reducer [8]	2018	✓	
CoVeriTest [7]	2019		
CondTest [11]	2019	✓	
MetaVal [14]	2020	✓	

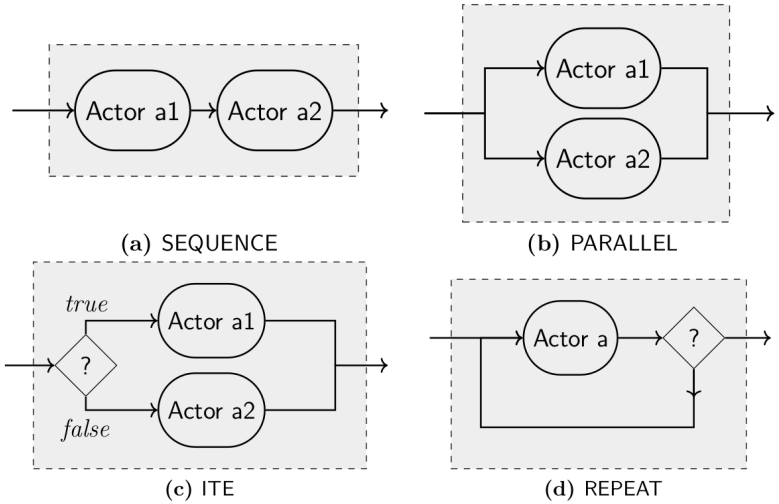
Impact

CoVeriTeam was used in:

- scientific research
  - *Construction of Verifier Combinations Based on Off-the-Shelf Verifiers* at FASE '22 [10]
  - *Decomposing Software Verification into Off-the-Shelf Components: An Application to CEGAR* at ICSE '22 [5]
- part of the workflow of software-verification competitions for last two years [1]

Conclusion






```
// Create actors
verifier = ActorFactory.create(ProgramVerifier,
    "actors/uautomizer.yml");
validator = ActorFactory.create(ProgramValidator,
    "actors/cpa-validate-violation-witnesses.yml");

// Prepare inputs
prog = ArtifactFactory.create(CProgram, prog_path);
spec = ArtifactFactory.create(BehaviorSpecification, spec_path);
inputs = {'program':prog, 'spec':spec};

// Use validator if verdict is true or false
condition = ELEMENTOF(verdict, {TRUE, FALSE});
second_component = ITE(condition, validator);
// Verifier and second component to be executed in sequence
validating_verifier = SEQUENCE(verifier, second_component);

// Execute the new component on the inputs
res = execute(validating_verifier, inputs);
print(res);
```

Technique	Year	Case Study	More Info
Counterexample Checking [15]	2012		
Conditional Model Checking [6]	2012		
Precision Reuse [12]	2013		
Witness Validation [3, 2]	2015, 2016	✓	
Execution-Based Validation [4]	2018	✓	
Reducer [8]	2018	✓	
CoVeriTest [7]	2019		
CondTest [11]	2019	✓	
MetaVal [14]	2020	✓	

GitLab

https://gitlab.com/sosy-lab/software/coveriteam/

References I

[1] Beyer, D.: Progress on software verification: SV-COMP 2022. In: Proc. TACAS (2). pp. 375–402. LNCS 13244, Springer (2022). doi:10.1007/978-3-030-99527-0_20

[2] Beyer, D., Dangl, M., Dietsch, D., Heizmann, M.: Correctness witnesses: Exchanging verification results between verifiers. In: Proc. FSE. pp. 326–337. ACM (2016). doi:10.1145/2950290.2950351

[3] Beyer, D., Dangl, M., Dietsch, D., Heizmann, M., Stahlbauer, A.: Witness validation and stepwise testification across software verifiers. In: Proc. FSE. pp. 721–733. ACM (2015). doi:10.1145/2786805.2786867

[4] Beyer, D., Dangl, M., Lemberger, T., Tautschnig, M.: Tests from witnesses: Execution-based validation of verification results. In: Proc. TAP. pp. 3–23. LNCS 10889, Springer (2018). doi:10.1007/978-3-319-92994-1_1

References II

[5] Beyer, D., Haltermann, J., Lemberger, T., Wehrheim, H.: Decomposing Software Verification into Off-the-Shelf Components: An Application to CEGAR. In: Proc. ICSE. ACM (2022)

[6] Beyer, D., Henzinger, T.A., Keremoglu, M.E., Wendler, P.: Conditional model checking: A technique to pass information between verifiers. In: Proc. FSE. ACM (2012). doi:10.1145/2393596.2393664

[7] Beyer, D., Jakobs, M.C.: CoVeriTest: Cooperative verifier-based testing. In: Proc. FASE. pp. 389–408. LNCS 11424, Springer (2019). doi:10.1007/978-3-030-16722-6_23

[8] Beyer, D., Jakobs, M.C., Lemberger, T., Wehrheim, H.: Reducer-based construction of conditional verifiers. In: Proc. ICSE. pp. 1182–1193. ACM (2018). doi:10.1145/3180155.3180259

[9] Beyer, D., Kanav, S.: CoVeriTeam: On-demand composition of cooperative verification systems. In: Proc. TACAS. pp. 561–579. LNCS 13243, Springer (2022). doi:10.1007/978-3-030-99524-9_31

References III

[10] Beyer, D., Kanav, S., Richter, C.: Construction of Verifier Combinations Based on Off-the-Shelf Verifiers. In: Proc. FASE. pp. 49–70. Springer (2022). doi:10.1007/978-3-030-99429-7_3

[11] Beyer, D., Lemberger, T.: Conditional testing: Off-the-shelf combination of test-case generators. In: Proc. ATVA. pp. 189–208. LNCS 11781, Springer (2019). doi:10.1007/978-3-030-31784-3_11

[12] Beyer, D., Löwe, S., Novikov, E., Stahlbauer, A., Wendler, P.: Precision reuse for efficient regression verification. In: Proc. FSE. pp. 389–399. ACM (2013). doi:10.1145/2491411.2491429

[13] Beyer, D., Löwe, S., Wendler, P.: Reliable benchmarking: Requirements and solutions. Int. J. Softw. Tools Technol. Transfer **21**(1), 1–29 (2019). doi:10.1007/s10009-017-0469-y

[14] Beyer, D., Spiessl, M.: MetaVal: Witness validation via verification. In: Proc. CAV. pp. 165–177. LNCS 12225, Springer (2020). doi:10.1007/978-3-030-53291-8_10

References IV

[15] Rocha, H.O., Barreto, R.S., Cordeiro, L.C., Neto, A.D.: Understanding programming bugs in ANSI-C software using bounded model checking counter-examples. In: Proc. IFM. pp. 128–142. LNCS 7321, Springer (2012). doi:10.1007/978-3-642-30729-4_10