Cooperative Verification

Sudeep Kanav
sudeep.kanav@lmu.de

Supervisors: Dirk Beyer

Collaborators: Tobias Kleinert & Cedric Richter & Henrik Wachowitz (ConVeY)

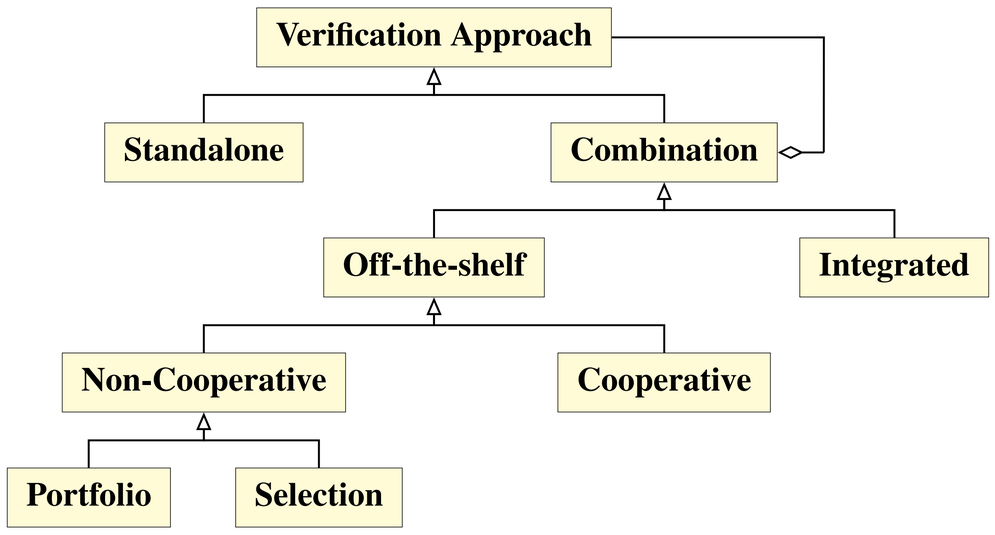
ConVeY

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Goal

1. Provide an overview of cooperative verification techniques published in the scientific literature
2. Organize this knowledge by developing a classification for these techniques

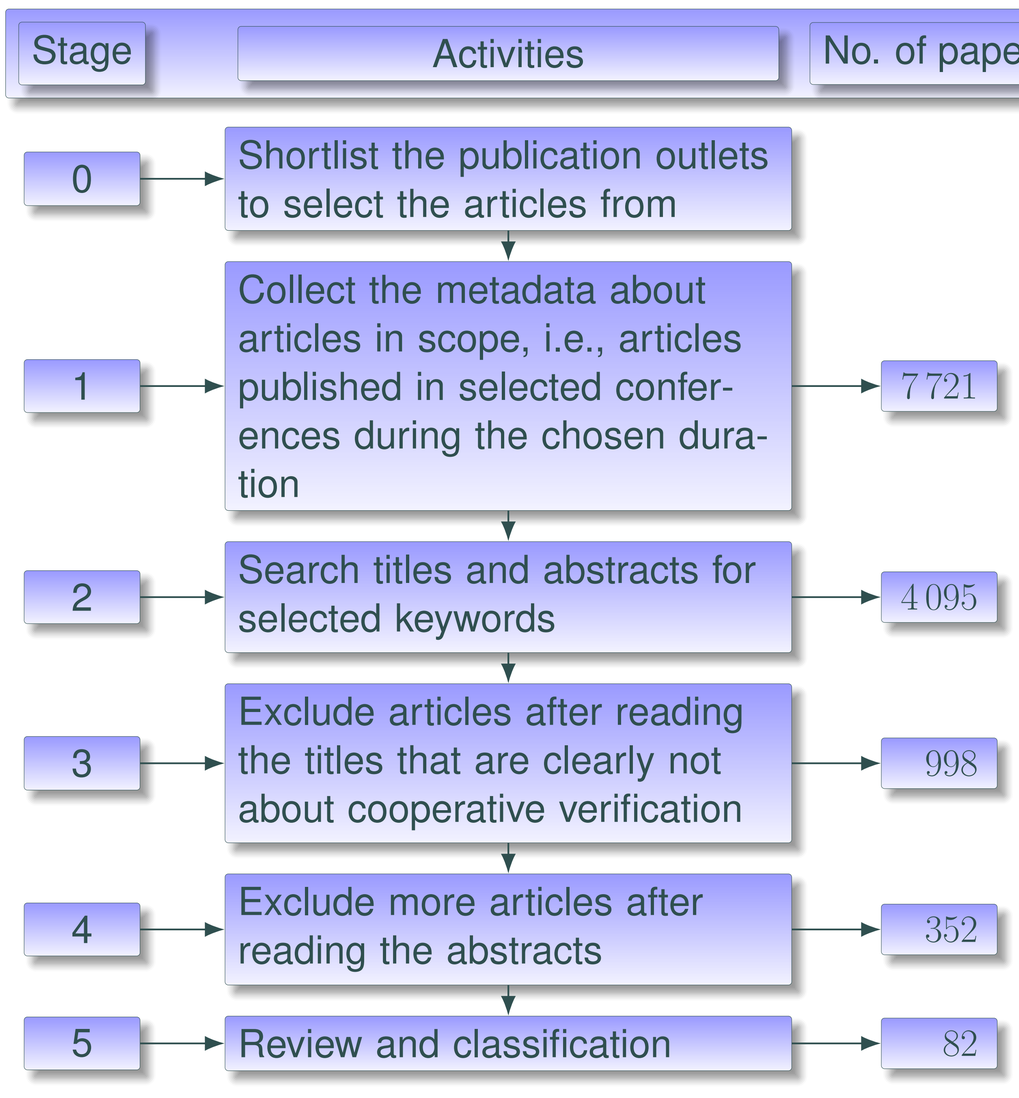
Definition



A verification approach is called **cooperative**, if *identifiable verification actors* pass information in form of *identifiable verification artifacts* towards the common objective of solving a verification problem, where at least two of these actors are analyzers.

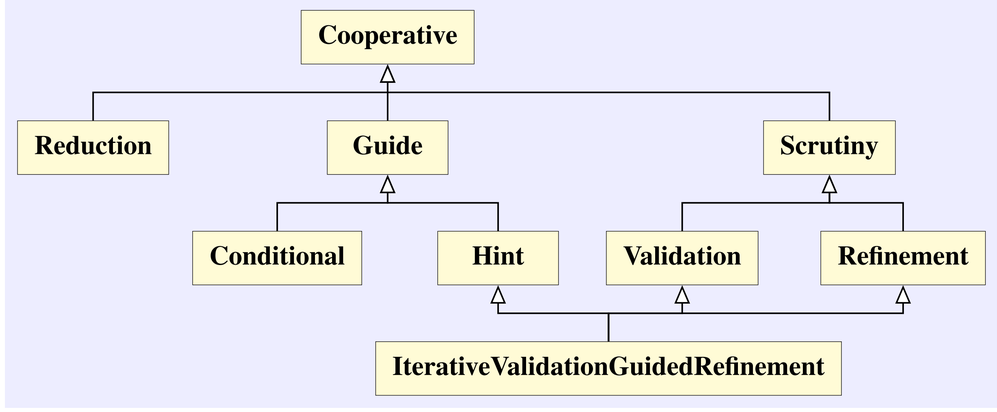
The restriction of *at least two analyzers* excludes tool combinations where the input is merely preprocessed to make it amenable to being verified by an analyzer. We exclude tools like syntactic transformers and slicers based on syntactic criteria from the definition of analyzers.

Selection Process



Classification of Selected Techniques

Class name	Explanation	Examples	Count
Reduction	The verification task is reduced such that it can be solved by another analyzer.	[7, 8]	9
Guide	The artifact produced by an analyzer acts to guide another analyzer.		
Conditional	First analyzer tries to solve the verification problem and produces an artifact that summarizes the work done; another actor then uses this information to focus only on the unsolved parts of the task.	[2, 3]	13
Hint	An analyzer generates hints that are then used to guide the verification of another analyzer.	[12, 10]	28
Scrutiny	The artifact produced is scrutinized by another analyzer.		
Validation	The result produced by one analyzer is validated by another analyzer.	[6, 4]	17
Refinement	The artifact produced by one analyzer is refined by another analyzer.	[9, 1]	4
Iterative Validation Guided Refinement	The artifact produced is first validated, and then the result of validation is used to guide the process of refinement. This sequence is repeated until a solution is found.	[5, 11]	11



Insights

1. Our classification reflects the ideas commonly used in verification
2. Researchers are actively working on specific parts of the commonly performed tasks during verification and combining them

References

[1] A. Albarghouthi et al. “From Under-Approximations to Over-Approximations and Back”. In: *Proc. TACAS*. LNCS 7214. Springer, 2012, pp. 157–172.

[2] D. Beyer et al. “Conditional Model Checking: A Technique to Pass Information between Verifiers”. In: *Proc. FSE*. ACM, 2012.

[3] D. Beyer et al. “Conditional Testing: Off-the-Shelf Combination of Test-Case Generators”. In: *Proc. ATVA*. LNCS 11781. Springer, 2019, pp. 189–208.

[4] D. Beyer et al. “Correctness Witnesses: Exchanging Verification Results Between Verifiers”. In: *Proc. FSE*. ACM, 2016, pp. 326–337.

[5] D. Beyer et al. “Decomposing Software Verification into Off-the-Shelf Components: An Application to CEGAR”. In: *Proc. ICSE*. ACM, 2022, pp. 536–548.

[6] D. Beyer et al. “Witness Validation and Stepwise Testification across Software Verifiers”. In: *Proc. FSE*. ACM, 2015, pp. 721–733.

[7] O. Inverso et al. “Bounded Model Checking of Multi-threaded C Programs via Lazy Sequentialization”. In: *Proc. CAV*. LNCS 8559. Springer, 2014, pp. 585–602.

[8] S. K. Lahiri et al. “Differential assertion checking”. In: *Proc. FSE*. ACM, 2013, pp. 345–355.

[9] K. Li et al. “Residual investigation: predictive and precise bug detection”. In: *Proc. ISSTA*. ACM, 2012, pp. 298–308.

[10] L. Ma et al. “GRT: Program-Analysis-Guided Random Testing (T)”. In: *Proc. ASE*. IEEE, 2015.

[11] D. Neider et al. “Invariant Synthesis for Incomplete Verification Engines”. In: *Proc. TACAS*. Springer, 2018, pp. 232–250.

[12] V. Wüstholz et al. “Targeted greybox fuzzing with static lookahead analysis”. In: *Proc. ICSE*. ACM, 2020.