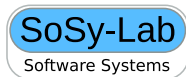


Strategy Selection for Software Verification Based on Boolean Features

Dirk Beyer and Matthias Dangl

LMU Munich, Germany



Based on:

Dirk Beyer and Matthias Dangl:

**Strategy Selection for Software Verification
Based on Boolean Features:
A Simple but Effective Approach**

Proc. ISoLA 2018

preprint: [online on our website under "Publications"](#)

Motivation

Symbolic Execution
CEGAR
Predicate Analysis
Value Analysis
PDR
Impact
Policy Iteration
Octagon Analysis
Interval Analysis
Boxes
k-Induction
Witnesses
Symbolic Memory Graphs
Invariant Generation
Bounded Model Checking
Slicing
Block-Abstraction Memoization
Automata-Based Trace Abstraction
Templates
Pointer-Alias Analysis

Motivation



Motivation

Quotes from users on our mailing list:

- ▶ *“I am a beginner in the field of Program Verification and I am using CPAchecker for the first time. So, I apologize for any stupid questions which I ask.”*
- ▶ *“How does a newcomer actually learn to use CPAchecker effectively?”*
- ▶ *“[W]hat is the best way of actually learning to use the tool and understand what all the settings do barring sitting down with an already-expert user?”*

Knowledge of verification expert required

Solution 1: Portfolios

- ▶ Create Portfolio Analyses
(B. A. Huberman, R. M. Lukose, and T. Hogg [5])
 - ▶ Parallel
 - ▶ Sequential

Examples of Parallel Portfolios

- ▶ UFO
 - ▶ Runs several analyses with different domains in parallel: Intervals, Boxes, Cartesian and Boolean predicate abstraction
 - ▶ A. Gurfinkel, A. Albarghouthi, S. Chaki, Y. Li, and M. Chechik, TACAS 2013 [4]

Examples of Parallel Portfolios

- ▶ PREDATORHP
 - ▶ Runs four configurations in parallel:
 - ▶ One configuration for verification
 - ▶ Three different “bug hunting” configurations
 - ▶ P. Müller, P. Peringer, and T. Vojnar, TACAS 2015 [6]

Examples of Sequential Portfolios

- ▶ Standard: Internal error-path validation with different (more precise) analysis
- ▶ SDV
 - ▶ Configuration “Q” first runs CORRAL for up to 1 400 s, then YOGI
 - ▶ V. Tulsian, A. Kanade, R. Kumar, A. Lal, and A. V. Nori, MSR 2014 [8]

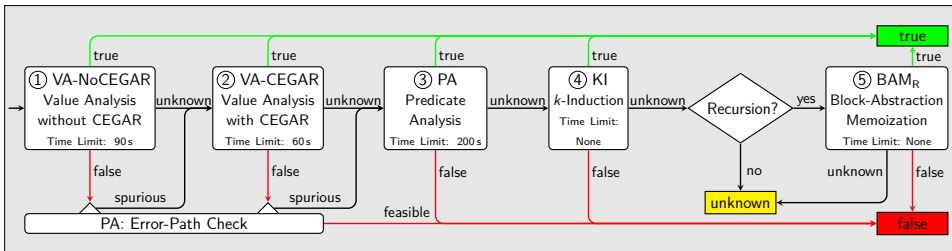
Examples of Sequential Portfolios

- ▶ CPAchecker
 - ▶ Won SV-COMP 2013 using sequential combination of value analysis and predicate analysis
 - ▶ All of our most successful competition submissions ever since
 - ▶ Won SV-COMP 2018 using sequential combination of
 - ▶ Value Analysis without CEGAR
 - ▶ Value Analysis with CEGAR
 - ▶ Predicate Analysis
 - ▶ k -Induction
 - ▶ Block-Abstraction Memoization in case the others fail due to recursion
 - ▶ P. Wendler, TACAS 2013 [9]

Hypothesis

Hypothesis 1

Combining different strategies sequentially is more effective than each individual strategy by itself.

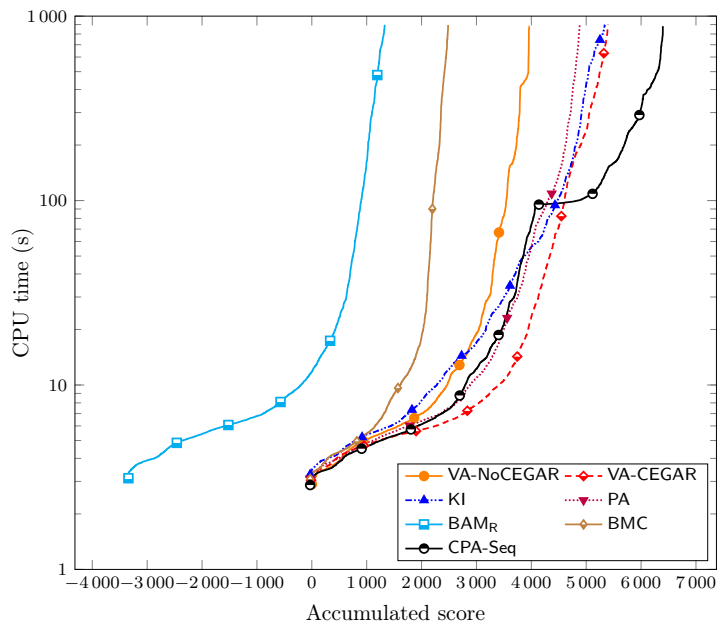


BAM_R: Configuration of block-abstraction memoization specifically for recursion but lacking support for handling pointer aliasing

Experiment Setup

- ▶ 5 687 verification tasks from SV-COMP'18 (only categories “ReachSafety” and “Systems_DeviceDrivers64_ReachSafety”)
- ▶ 15 min time limit per task (CPU time)
- ▶ 15 GB memory limit
- ▶ Measured with `BENCHEXEC`

Results



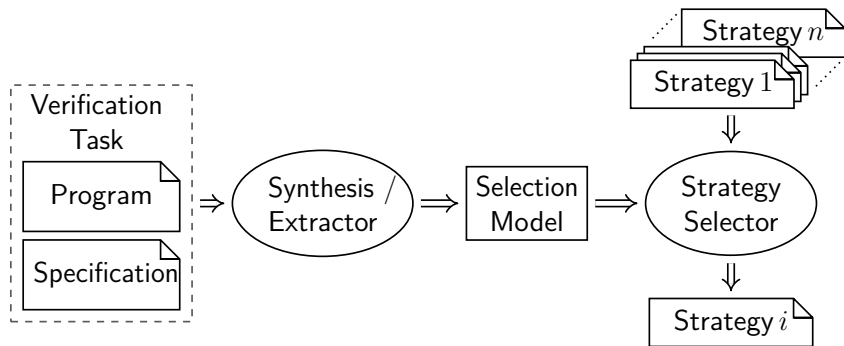
Results

Approach	VA-NoCEGAR	VA-CEGAR	PA	KI	BAM _R	BMC	CPA-Seq
Score	3 966	5 397	4 881	5 340	1 335	2 484	6 399
Correct results	2 365	3 046	2 840	3 053	2 575	1 757	3 740
Correct proofs	1 601	2 367	2 073	2 319	2 104	759	2 691
Correct alarms	764	679	767	734	471	998	1 049
Wrong proofs	0	0	0	0	10	0	0
Wrong alarms	0	1	2	2	189	2	2
Timeouts	2 376	1 554	2 497	2 236	2 167	3 379	1 715
Out of memory	1	1	14	243	128	381	194
Other inconclusive	945	1 085	334	153	618	168	36
Times for correct results							
Total CPU Time (h)	30	54	39	68	33	28	79
Avg. CPU Time (s)	45	64	49	80	46	57	76
Total Wall Time (h)	24	44	33	43	29	25	65
Avg. Wall Time (s)	36	52	42	51	40	51	63

Solution 2: Algorithm Selection

- ▶ Create Portfolio Analyses
 - ▶ Parallel
 - ▶ Sequential
- ▶ **Algorithm Selection**

Algorithm Selection



J. R. Rice: *The algorithm selection problem*, *Advances in Computers* 1976 [7]

Examples of Algorithm Selection with Machine Learning

- ▶ Verifolio (category prediction)
 - ▶ Define 13 variable roles
 - ▶ Y. Demyanova, H. Veith, and F. Zuleger, FMCAD 2013 [3]
- ▶ Ranking prediction
 - ▶ Choice based on graph representation of program
 - ▶ M. Czech, E. Hüllermeier, M. Jakobs, and H. Wehrheim, SWAN 2017 [2]
- ▶ MUX
 - ▶ Classification over 14 features
 - ▶ Choice between YOGI and CORRAL
 - ▶ V. Tulsian, A. Kanade, R. Kumar, A. Lal, and A. V. Nori, MSR 2014 [8]

Examples of Algorithm Selection without Machine Learning

- ▶ CPACHECKER
 - ▶ CPA-Seq chooses configuration depending on specification
 - ▶ Previously displayed configuration only for meta categories “ReachSafety” and “Systems_DeviceDriversLinux64_ReachSafety”
 - ▶ Special, individual configurations for
 - ▶ Concurrency
 - ▶ Memory Safety
 - ▶ Overflows
 - ▶ Termination

Hypothesis

Hypothesis 2

Given a set of sequentially composed verification strategies and a small set of features, algorithm selection can further improve effectiveness significantly.

Feature Model

$\text{hasLoop} : V \rightarrow \mathbb{B}$ with

$\text{hasLoop}((p, \cdot)) = \text{true}$ if program p has a loop,
and *false* otherwise

$\text{hasFloat} : V \rightarrow \mathbb{B}$ with

$\text{hasFloat}((p, \cdot)) = \text{true}$ if program p has a variable
of a floating-point type (`float`, `double`, and
`long double` in C), and *false* otherwise

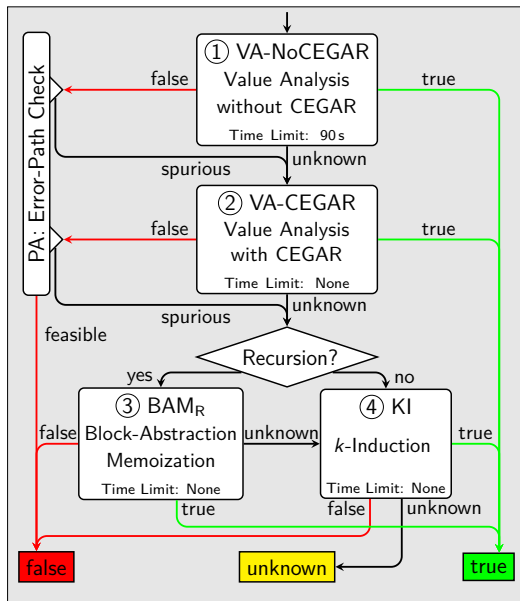
$\text{hasArray} : V \rightarrow \mathbb{B}$ with

$\text{hasArray}((p, \cdot)) = \text{true}$ if program p has a variable
of an array type, and *false* otherwise

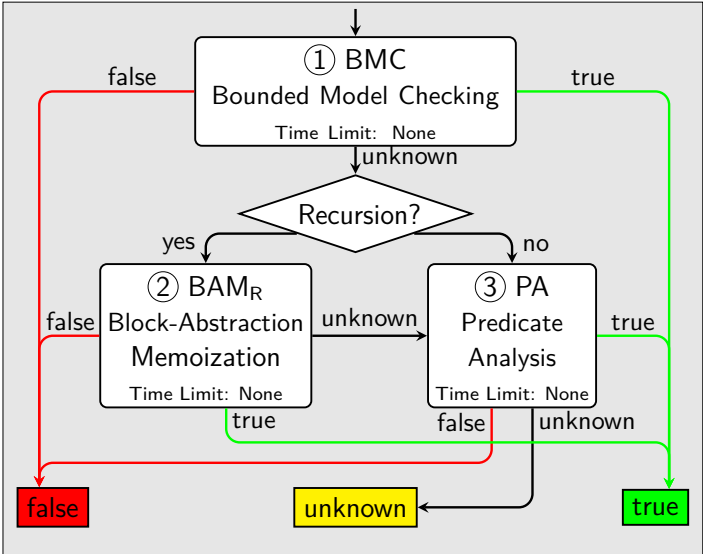
$\text{hasComposite} : V \rightarrow \mathbb{B}$ with

$\text{hasComposite}((p, \cdot)) = \text{true}$ if program p has a
variable of a composite type (`struct` and `union`
in C), and *false* otherwise

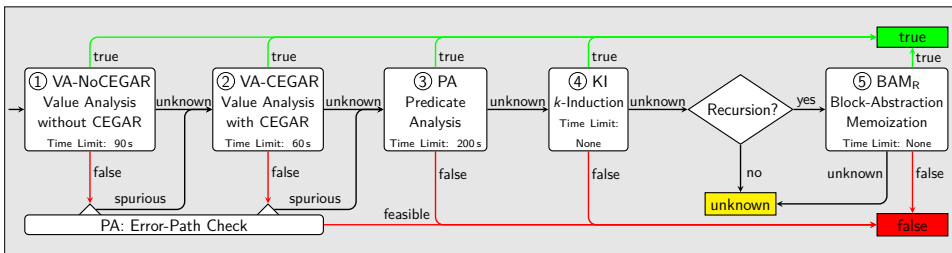
VA-BAM_R-KI



BMC-BAM_R-PA



CPA-Seq



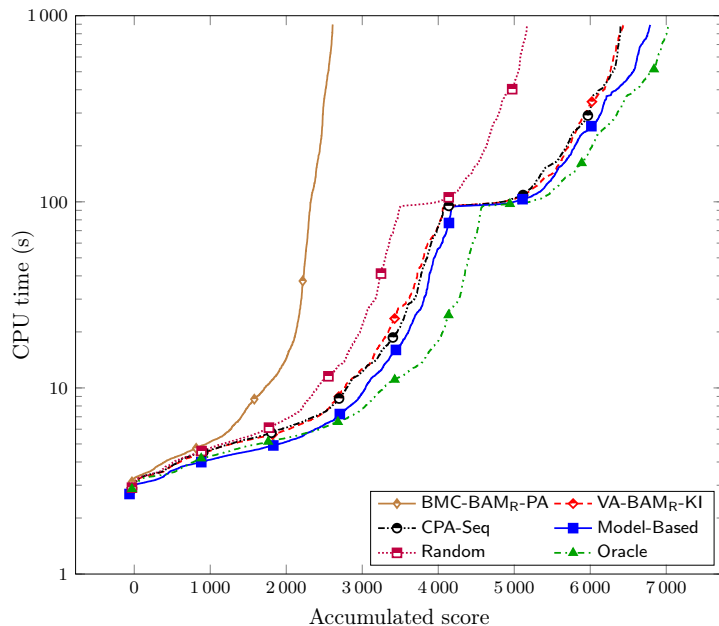
Strategy Selector

$$\text{strategy} = \begin{cases} \text{BMC-BAM}_R\text{-PA} & \text{if } \neg\text{hasLoop} \\ \text{VA-BAM}_R\text{-KI} & \text{if } \text{hasLoop} \\ \text{CPA-Seq} & \wedge(\text{hasFloat} \vee \text{hasArray} \vee \text{hasComposite}) \\ & \text{otherwise} \end{cases}$$

Experiment Setup

- ▶ 5 687 verification tasks from SV-COMP'18 (only categories “ReachSafety” and “Systems_DeviceDrivers64_ReachSafety”)
- ▶ 15 min time limit per task (CPU time)
- ▶ 15 GB memory limit
- ▶ Measured with `BENCHEXEC`

Results



Results

Approach	Sequential Combinations			Random	Model-Based	Oracle
	CPA-Seq	BMC-BAM _R -PA	VA-BAM _R -KI			
Score	6 399	2 612	6 442	5 174	6 790	7 036
% of Oracle Score	91	37	92	74	97	100
Correct results	3 740	1 840	3 740	3 122	3 932	4 111
Correct proofs	2 691	804	2 734	2 084	2 922	2 957
Correct alarms	1 049	1 036	1 006	1 038	1 010	1 154
Wrong proofs	0	0	0	0	0	0
Wrong alarms	2	2	2	2	4	2
Timeouts	1 715	3 385	1 879	2 317	1 486	1 347
Out of memory	194	406	26	202	224	185
Other inconclusive	36	54	40	44	41	42
Times for correct results						
Total CPU Time (h)	79	28	87	66	99	96
Avg. CPU Time (s)	76	54	83	76	90	84
Total Wall Time (h)	65	25	70	55	80	79
Avg. Wall Time (s)	63	48	67	63	73	69

Summary

- ▶ We define a minimalist selection model, which consists of
 - ▶ an extremely small set of features that define the selection model and
 - ▶ a minimal range of values: all features are of type Boolean.
- ▶ We define an extremely simple strategy selector, which is based on insights from verification researchers.
- ▶ We implemented our feature measures and strategy selection in CPACHECKER.
- ▶ We perform a thorough experimental evaluation on a large benchmark set.
- ▶ We provide a baseline for comparison of more sophisticated approaches to strategy selection.

Future Work

1. Define a better model
2. Find better configurations
3. Design a better selection function

References I



D. Beyer and M. Dangl.

Strategy selection for software verification based on boolean features:
A simple but effective approach.

In *Proc. ISoLA*, LNCS 11245, pages 144–159. Springer, 2018.



M. Czech, E. Hüllermeier, M. Jakobs, and H. Wehrheim.

Predicting rankings of software verification tools.

In *Proc. SWAN*, pages 23–26. ACM, 2017.



Y. Demyanova, H. Veith, and F. Zuleger.

On the concept of variable roles and its use in software analysis.

In *Proc. FMCAD*, pages 226–230. IEEE, 2013.



A. Gurfinkel, A. Albarghouthi, S. Chaki, Y. Li, and M. Chechik.

UFO: Verification with interpolants and abstract interpretation (competition contribution).

In *Proc. TACAS*, LNCS 7795, pages 637–640. Springer, 2013.

References II



B. A. Huberman, R. M. Lukose, and T. Hogg.
An economics approach to hard computational problems.
Science, 275(7):51–54, 1997.



P. Müller, P. Peringer, and T. Vojnar.
Predator hunting party (competition contribution).
In *Proc. TACAS*, LNCS 9035, pages 443–446. Springer,



J. R. Rice.
The algorithm selection problem.
Advances in Computers, 15:65–118, 1976.



V. Tulsian, A. Kanade, R. Kumar, A. Lal, and A. V. Nori.
MUX: Algorithm selection for software model checkers.
In *Proc. MSR*. ACM, 2014.



P. Wendler.
CPACHECKER with sequential combination of explicit-state analysis
and predicate analysis (competition contribution).
In *Proc. TACAS*, LNCS 7795, pages 613–615. Springer, 2013.