Strategy Selection for Software Verification Based on Boolean Features

Matthias Dangl
Joint work with Dirk Beyer

LMU Munich, Germany

Based on:

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

Motivation


CEGAR
Policy Iteration
Value Analysis
PDR
Pointer-Alias Analysis
Octagon Analysis
Boxes
Witnesses
Bounded Model Checking
Symbolic Memory Graphs
Invariant Generation
Slicing
Automata-Based Trace Abstraction
Templates

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?"*

Problem

Verification-Expert knowledge required

Solution: Portfolios

- Create Portfolio Analyses
  - 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 (Proc. TACAS 2013)]

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 (Proc. TACAS 2015)]

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 (Proc. MSR 2014)]

Examples of Sequential Portfolios

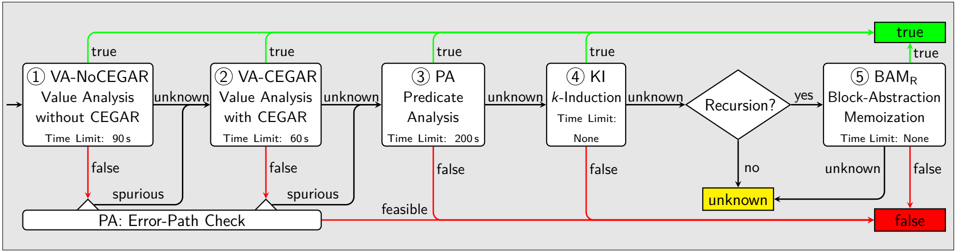
- CPAchecker
  - Won SV-COMP 2013 using sequential combination of value analysis and predicate analysis
  - All of our most successful competition submissions ever since
  - Won SV-COMP 2018 using sequential combination of
    - Value Analysis without CEGAR
    - Value Analysis with CEGAR
    - Predicate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

Hypothesis

Hypothesis 1

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

CPA-Seq

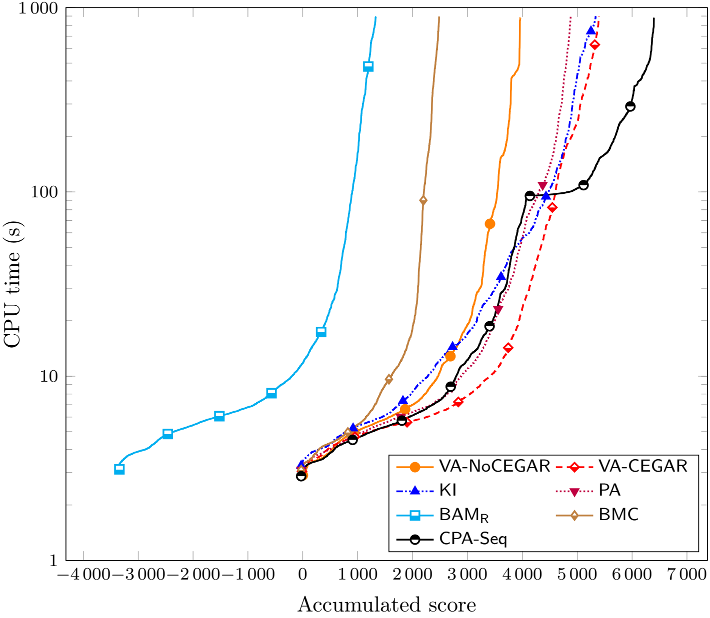


BAM_R: Configuration of block-abstraction memoization specifically for recursion but lacking support for handling pointer aliasing

Experiment Setup

- 5 687 verification tasks from SV-COMP'18 (only categories "ReachSafety", "Systems_DeviceDrivers64_ReachSafety")
- 15 min time limit per task (CPU time)
- 15 GB memory limit
- Measured with BenchExec

Results



1 000
100
10
1
CPU time (s)
−4 000
−3 000
−2 000
−1 000
0
1 000
2 000
3 000
4 000
5 000
6 000
7 000
Accumulated score
VA-NoCEGAR
VA-CEGAR
KI
PA
BAM_R
BMC
CPA-Seq

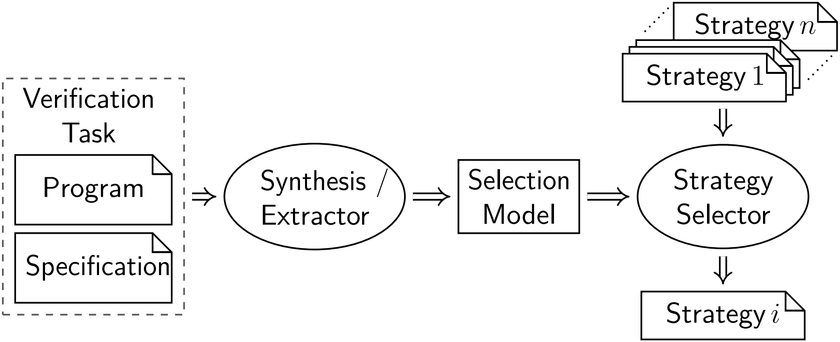
Results

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: Algorithm Selection

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

Algorithm Selection



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

Examples of Algorithm Selection with Machine Learning

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

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

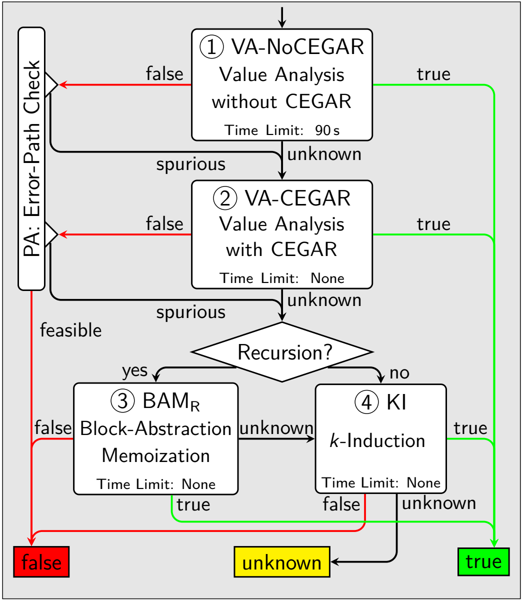
hasLoop : $V \to \mathbb{B}$ with
hasLoop$((p, \cdot)) = \mathit{true}$ if program p has a loop, and false otherwise

hasFloat : $V \to \mathbb{B}$ with
hasFloat$((p, \cdot)) = \mathit{true}$ if program p has a variable of a floating-point type (`float`, `double`, and `long double` in C), and false otherwise

hasArray : $V \to \mathbb{B}$ with
hasArray$((p, \cdot)) = \mathit{true}$ if program p has a variable of an array type, and false otherwise

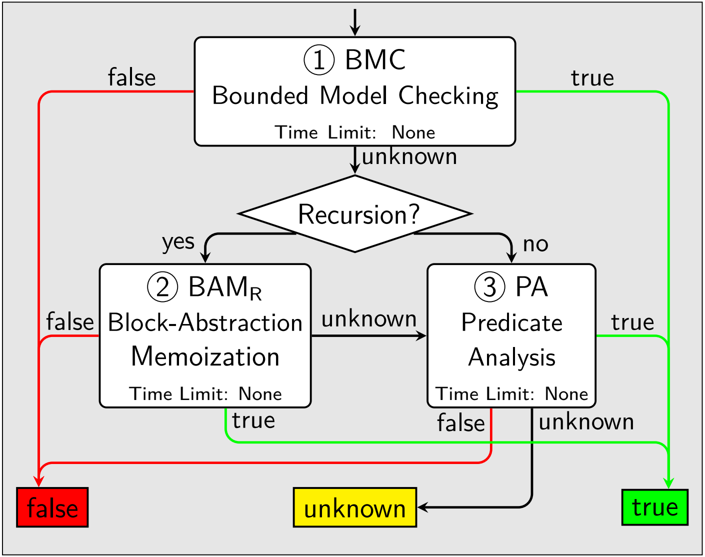
hasComposite : $V \to \mathbb{B}$ with
hasComposite$((p, \cdot)) = \mathit{true}$ if program p has a variable of a composite type (`struct` and `union` in C), and false otherwise

VA-BAM$_R$-KI



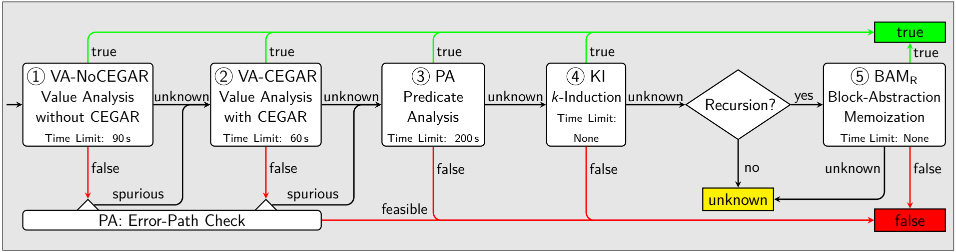
PA: Error-Path Check
① VA-NoCEGAR
Value Analysis
without CEGAR
Time Limit: 90 s
false
true
spurious
unknown
② VA-CEGAR
Value Analysis
with CEGAR
Time Limit: None
false
true
spurious
unknown
feasible
Recursion?
yes
no
③ BAM$_R$
Block-Abstraction
Memoization
Time Limit: None
false
unknown
true
④ KI
k-Induction
Time Limit: None
true
false
unknown
false
unknown
true

BMC-BAM$_R$-PA



① BMC
Bounded Model Checking
Time Limit: None
false
true
unknown
Recursion?
yes
no
② BAM$_R$
Block-Abstraction
Memoization
Time Limit: None
false
unknown
③ PA
Predicate
Analysis
Time Limit: None
true
true
false
unknown
false
unknown
true

CPA-Seq


① VA-NoCEGAR
Value Analysis
without CEGAR
Time Limit: 90 s
② VA-CEGAR
Value Analysis
with CEGAR
Time Limit: 60 s
③ PA
Predicate
Analysis
Time Limit: 200 s
④ KI
k-Induction
Time Limit:
None
Recursion?
⑤ BAM$_R$
Block-Abstraction
Memoization
Time Limit: None
true
false
unknown
spurious
feasible
yes
no
PA: Error-Path Check
true
unknown
false

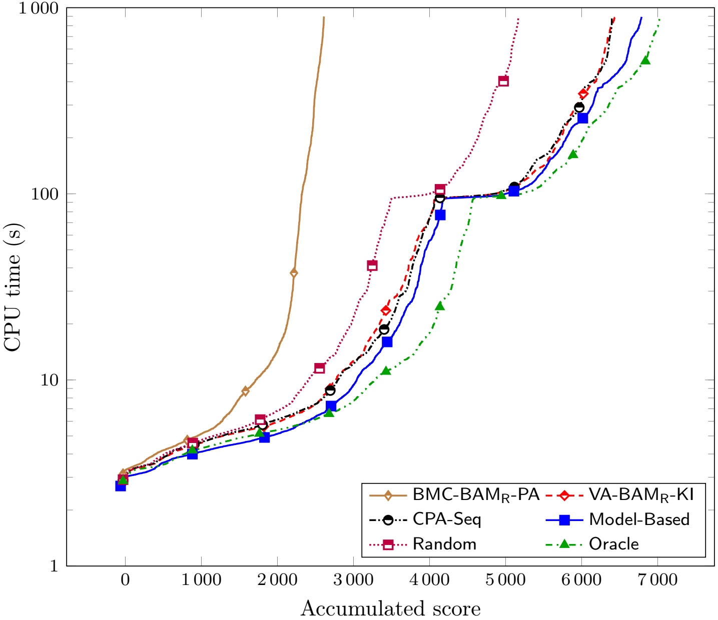
Strategy Selector

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

Experiment Setup

- 5 687 verification tasks from SV-COMP'18 (only categories "ReachSafety", "Systems_DeviceDrivers64_ReachSafety")
- 15 min time limit per task (CPU time)
- 15 GB memory limit
- Measured with BenchExec

Results



CPU time (s)
1 000
100
10
1
0
1 000
2 000
3 000
4 000
5 000
6 000
7 000
Accumulated score
BMC-BAM$_R$-PA
VA-BAM$_R$-KI
CPA-Seq
Model-Based
Random
Oracle

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