Conditional Model Checking

Dirk Beyer

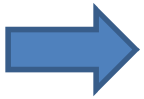
Joint work with Tom Henzinger, Erkan Keremoglu, Philipp Wendler

Software Verification

C program

```
int main() {
  int a = foo();
  int b = bar(a);

  assert(a == b);
}
```



Verification Tool

SAFE

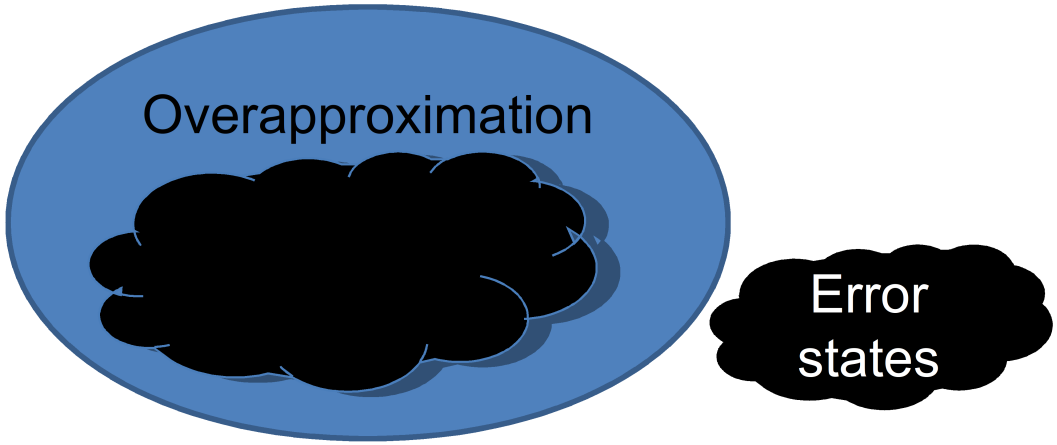
i.e. assertions cannot be violated

UNSAFE

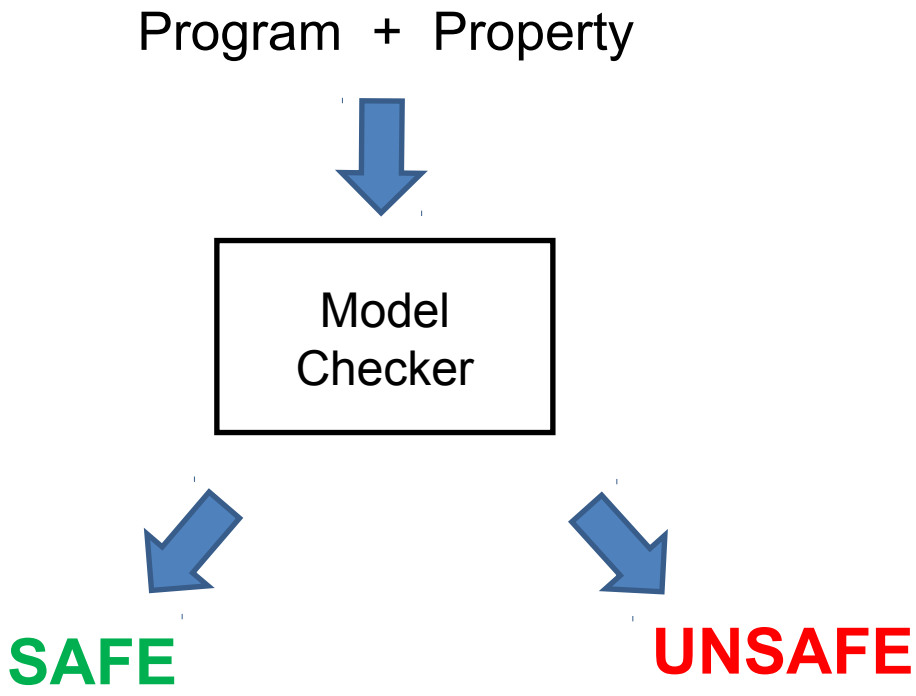
Safety Properties

General method:
Create an overapproximation of the program states

Soundness:



Model Checking



ACM Turing Award 2007

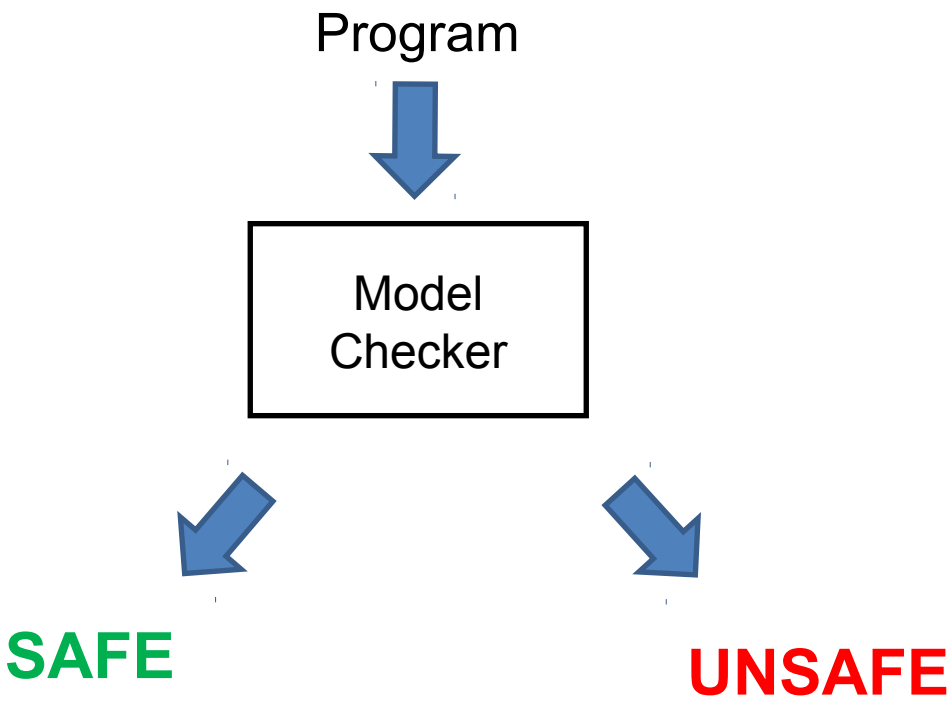
Edmund Clarke

Allen Emmerson

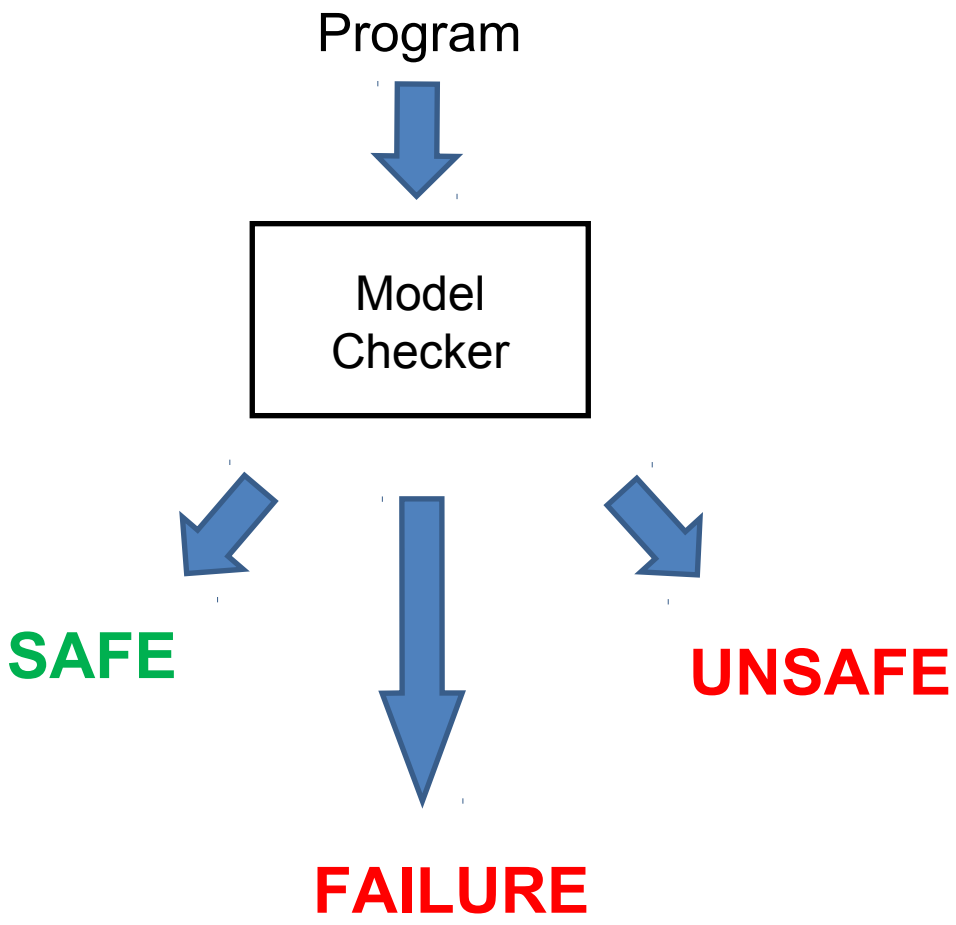
Joseph Sifakis

Invention: “Model Checking”

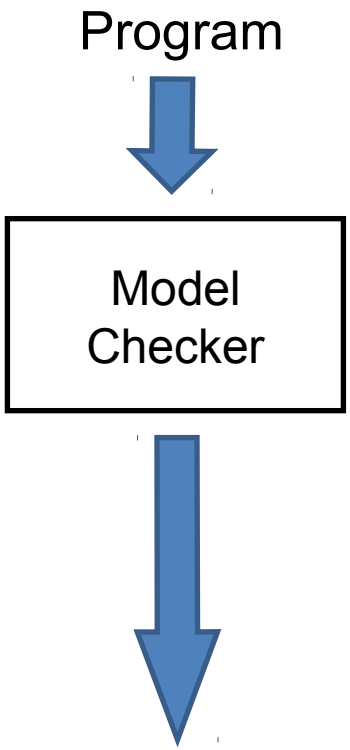
Classic Model Checking



Classic Model Checking



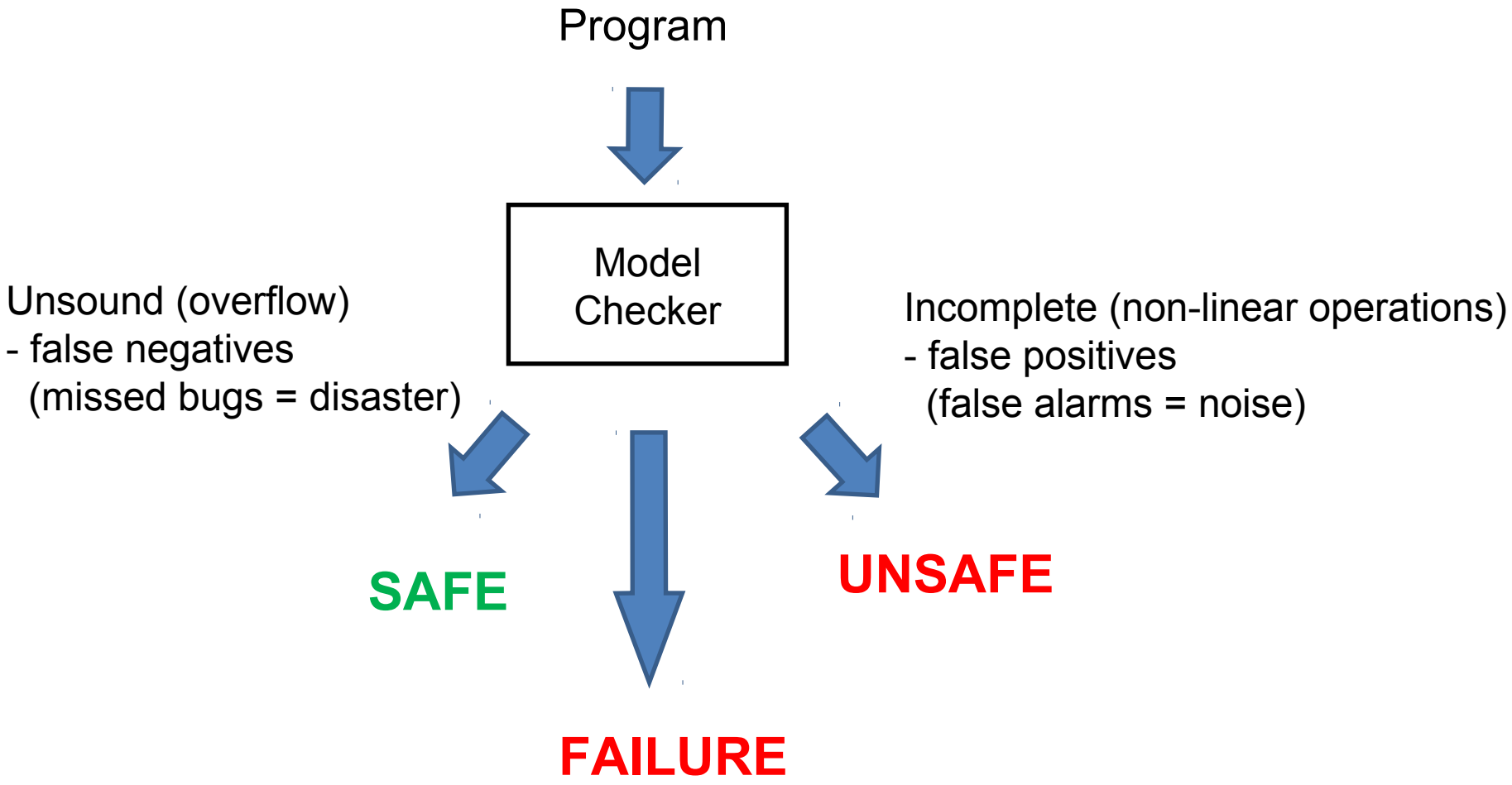
Classic Model Checking



- timeout
- out of memory
- crash of component
- operand exception

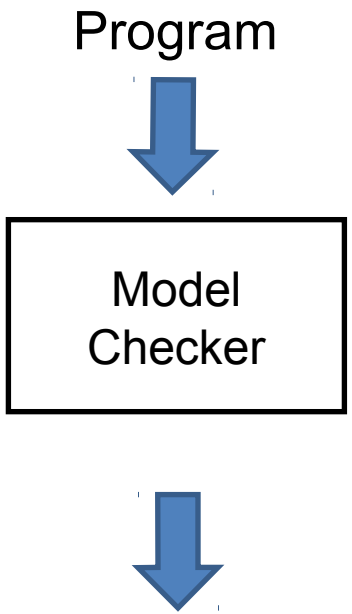
Enormous amounts of resources **wasted**!

Classic Model Checking



Conditional Model Checking

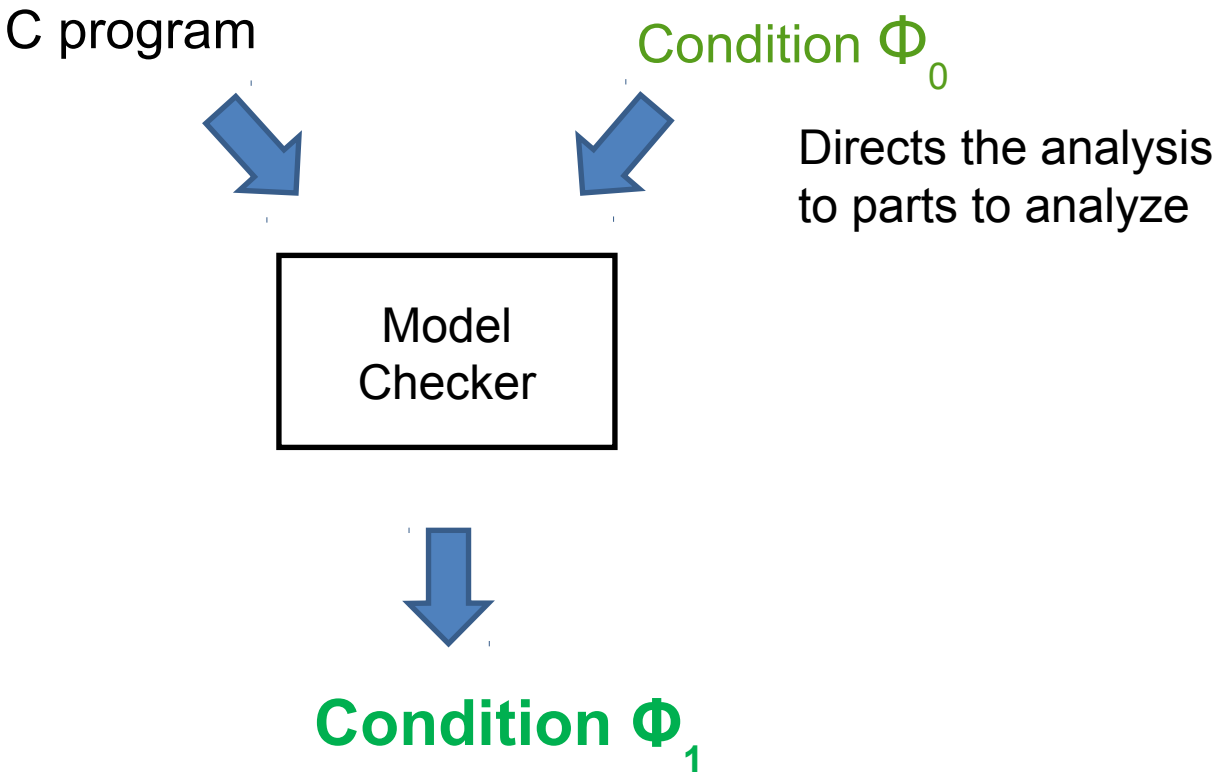
Conditional Model Checking



SAFE under Condition Φ

Examples:
- **Φ =** true: previous SAFE
- **Φ =** false: previous UNSAFE
- general: condition for safety

Conditional Model Checking



Examples:
- **Φ =** true: previous SAFE
- **Φ =** false: previous UNSAFE
- general: condition for safety

Conditional Model Checking

- Never crash!
  → I Condition: specify time and memory
- Always dump results!
  → O Condition: report partial results
- Sequential composition
  → Solve harder problems
- Comparison of Checkers:
  → Winner is who has Weakest Condition!
  (i.e., has proved “most”)

Input Conditions

Two main use cases for input conditions

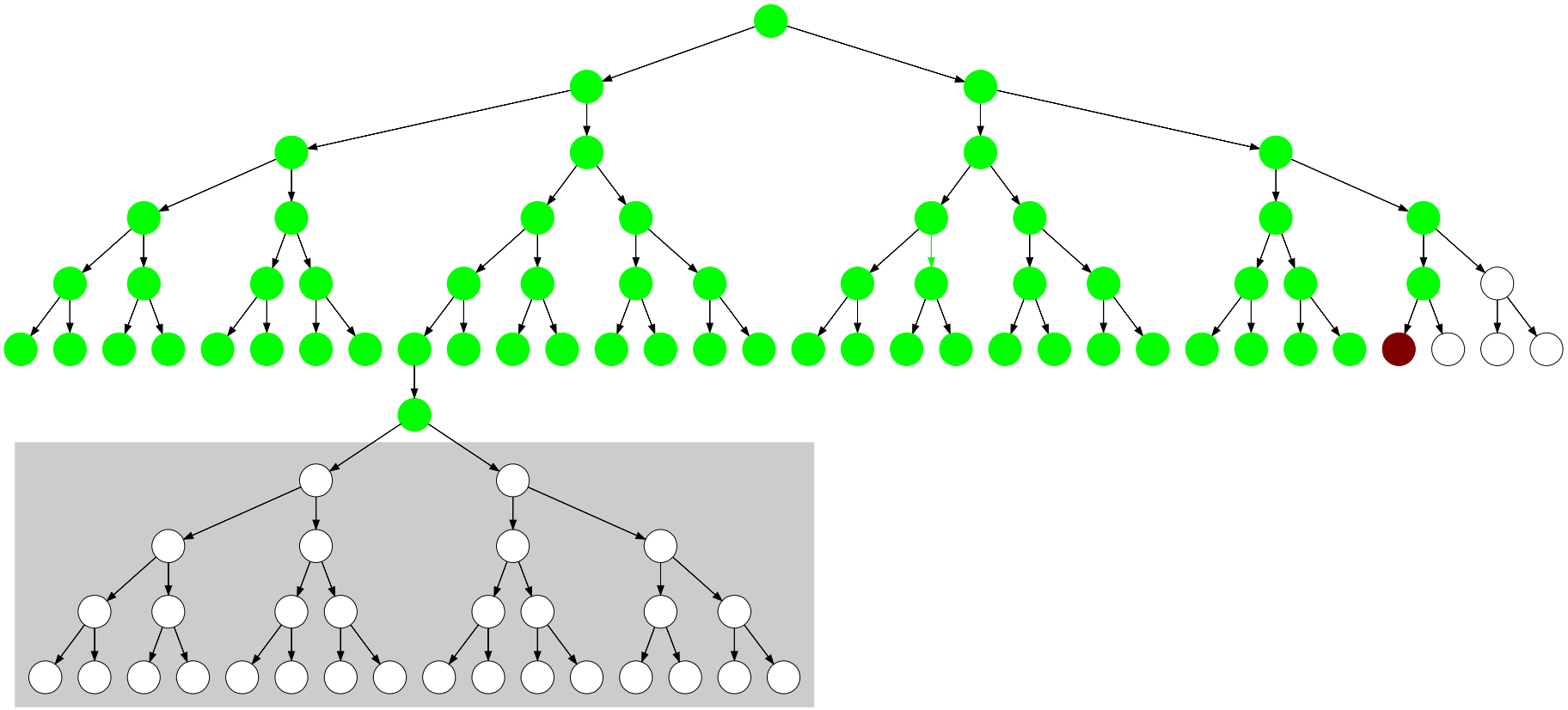
A) Use conditions for restricting the search (similar to bounded model checking)

B) Use output of another model checker

Quantitative Input Conditions

Component	Name	Condition	Impl.
Global Progress	Total Time	$time$	✓
	Total Space	mem	✓
	# Abstract States	$\lvert\mathsf{reached}\rvert$	✓
	# Abstract States per Loc.	$\#(\mathsf{reached}, loc)$	
	Busy Edge	$\#(edge)$	✓
Post Computation	Time for Post	$time(\rightsquigarrow)$	✓
	Space for Post	$mem(\rightsquigarrow)$	
	Size of State	$mem(state)$	
	Path Length	$length(path)$	✓
	Time Spent in Path	$time(path)$	✓
	Repeating Locs. in Path	$\#(path, loc)$	✓
	Assume Edges in Path	$assumes(path)$	✓
CEX Analysis	Time for Refinement	$time(ref)$	✓
	Space for Refinement	$mem(ref)$	
	Size of Path Formula	$mem(pf)$	✓

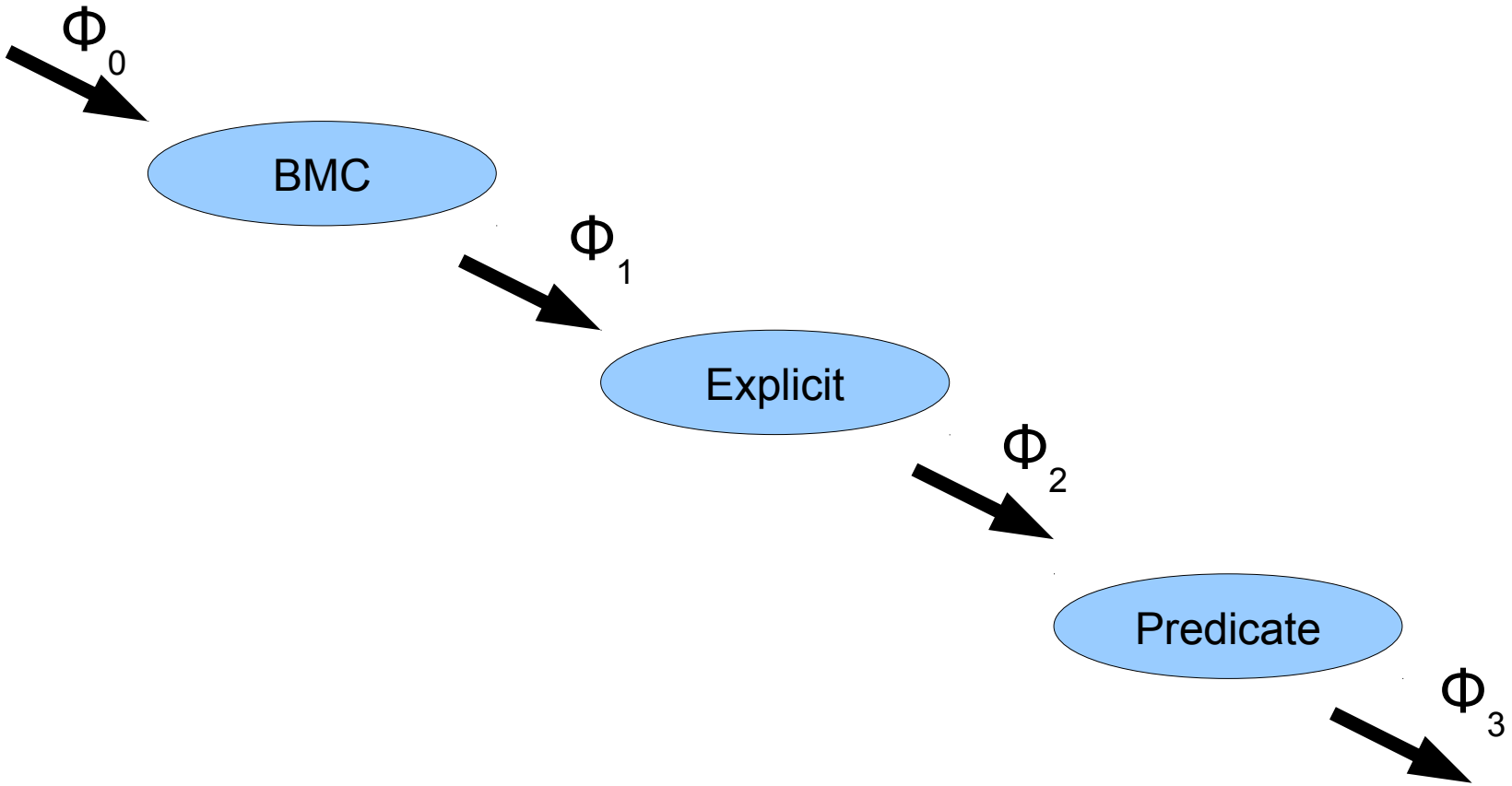
Input Condition: Path Length



Input Condition: Path Length

```
1   void main() {
2     int x = 1;
3     if (nondet_int()) {
4       while (x < 10000) {
5         x++;
6       }
7       assert (x == 10000);
8     } else {
9       x = 0;
10    }
11    assert (x != 0);
12  }
```

Experiment: Sequential Composition



	CBMC				CPACHECKER					
					Explicit		Explicit		Predicate	
	$k = 1$		$k = 10$		$time(10s)$					
cdaudio_simpl1	1.0	✓	1.0	✓	3.5	✓	3.2	✓	17	✓
cdaudio_simpl1_BUG	.99	✓	.99	✓	2.8	✓	2.5	✓	10	✓
diskperf_simpl1	.25	-	.26	-	13	-	900	-	15	✓
floppy_simpl3	.16	✓	.15	✓	2.9	✓	2.4	✓	8.5	✓
floppy_simpl3_BUG	.16	✓	.16	✓	2.2	✓	2.5	✓	7.8	✓
floppy_simpl4	.28	✓	.28	✓	3.5	✓	3.7	✓	12	✓
floppy_simpl4_BUG	.30	✓	.30	✓	3.0	✓	2.9	✓	11	✓
kbfiltr_simpl1	.05	✓	.06	✓	3.1	✓	2.2	✓	3.7	✓
kbfiltr_simpl2	.11	✓	.10	✓	3.1	✓	3.2	✓	5.2	✓
kbfiltr_simpl2_BUG	.11	✓	.12	✓	2.2	✓	2.1	✓	4.1	✓
NT drivers total	3.4	9	3.4	9	39	9	920	9	94	10
s3_clnt_1	.03	-	4.3	-	9.8	✓	8.3	✓	8.1	✓
s3_clnt_1_BUG	.03	-	4.2	-	4.4	✓	3.6	✓	6.2	✓
s3_clnt_2	.03	-	4.6	-	10	✓	8.2	✓	7.2	✓
s3_clnt_2_BUG	.03	-	4.3	-	4.5	✓	3.5	✓	5.4	✓
s3_clnt_3	.03	-	5.3	-	9.7	✓	8.1	✓	5.8	✓
s3_clnt_3_BUG	.03	-	5.3	✓	5.1	✓	3.5	✓	6.0	✓
s3_clnt_4	.03	-	4.8	-	9.8	✓	8.5	✓	10	✓
s3_clnt_4_BUG	.03	-	4.3	-	4.3	✓	3.5	✓	6.4	✓
s3_srvr_1	.03	-	4.1	-	3.3	✓	2.4	✓	21	✓
s3_srvr_1_BUG	.03	-	6.4	✓	2.2	✓	1.7	✓	4.8	✓
s3_srvr_2	.03	-	5.5	-	2.8	✓	2.4	✓	150	✓
s3_srvr_2_BUG	.03	-	6.2	✓	1.8	✓	1.7	✓	4.1	✓
s3_srvr_3	.03	-	5.6	-	3.6	-	2.6	-	9.0	✓
s3_srvr_4	.03	-	5.6	-	3.1	-	2.5	-	28	✓
s3_srvr_6	.03	-	6.6	-	14	-	250	✓	230	✓
s3_srvr_7	.03	-	6.3	-	14	-	200	✓	47	✓
s3_srvr_8	.03	-	6.1	-	2.8	✓	3.0	✓	23	✓
SSH total	0.51	0	90	3	110	13	510	15	570	17
bist_cell	.02	-	.65	-	9.4	-	9.2	-	210	✓
kundu	.02	-	120	-	7.3	✓	6.2	✓	900	-
kundu1_BUG	.02	-	18	-	2.0	✓	1.8	✓	15	✓
kundu2_BUG	.02	-	120	-	1.6	✓	1.9	✓	510	✓
pc_sfifo_1	.02	-	12	-	14	-	900	-	5.3	✓
pc_sfifo_2	.02	-	6.3	-	14	-	900	-	9.8	✓
token_ring.01	.02	-	16	-	1.7	-	1.6	-	8.1	✓
toy2_BUG	.03	-	30	-	2.7	✓	2.1	✓	65	✓

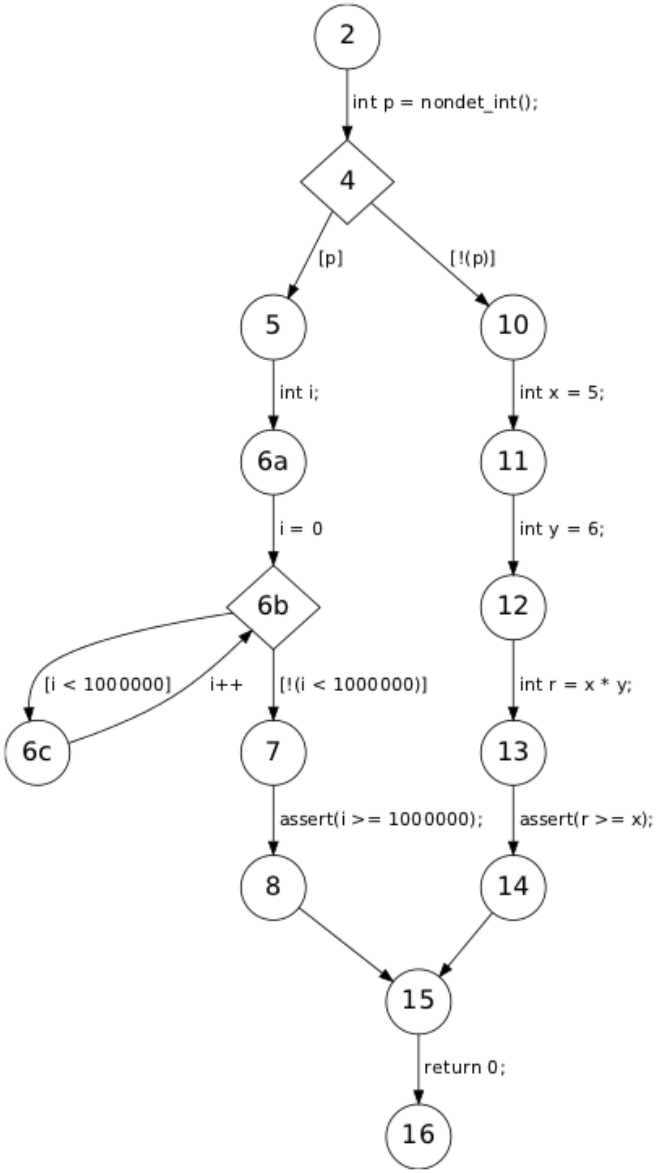
	CBMC				CPACHECKER									
					Explicit						Comb. A		Comb. B	
	$k=1$		$k=10$		$time(10s)$		Explicit		Predicate		Expl. + Pred.		CBMC+Expl.+Pred.	
cdaudio_simpl1	1.0	✓	1.0	✓	3.5	✓	3.2	✓	17	✓	3.5	✓	1.0	✓
cdaudio_simpl1_BUG	.99	✓	.99	✓	2.8	✓	2.5	✓	10	✓	2.8	✓	.99	✓
diskperf_simpl1	.25	-	.26	-	13	-	900	-	15	✓	28	✓	28	✓
floppy_simpl3	.16	✓	.15	✓	2.9	✓	2.4	✓	8.5	✓	2.9	✓	.16	✓
floppy_simpl3_BUG	.16	✓	.16	✓	2.2	✓	2.5	✓	7.8	✓	2.2	✓	.16	✓
floppy_simpl4	.28	✓	.28	✓	3.5	✓	3.7	✓	12	✓	3.5	✓	.28	✓
floppy_simpl4_BUG	.30	✓	.30	✓	3.0	✓	2.9	✓	11	✓	3.0	✓	.30	✓
kbfiltr_simpl1	.05	✓	.06	✓	3.1	✓	2.2	✓	3.7	✓	3.1	✓	.05	✓
kbfiltr_simpl2	.11	✓	.10	✓	3.1	✓	3.2	✓	5.2	✓	3.1	✓	.11	✓
kbfiltr_simpl2_BUG	.11	✓	.12	✓	2.2	✓	2.1	✓	4.1	✓	2.2	✓	.11	✓
NT drivers total	3.4	9	3.4	9	39	9	920	9	94	10	54	10	31	10
s3_clnt_1	.03	-	4.3	-	9.8	✓	8.3	✓	8.1	✓	9.8	✓	9.8	✓
s3_clnt_1_BUG	.03	-	4.2	-	4.4	✓	3.6	✓	6.2	✓	4.4	✓	4.4	✓
s3_clnt_2	.03	-	4.6	-	10	✓	8.2	✓	7.2	✓	10	✓	10	✓
s3_clnt_2_BUG	.03	-	4.3	-	4.5	✓	3.5	✓	5.4	✓	4.5	✓	4.5	✓
s3_clnt_3	.03	-	5.3	-	9.7	✓	8.1	✓	5.8	✓	9.7	✓	9.7	✓
s3_clnt_3_BUG	.03	-	5.3	✓	5.1	✓	3.5	✓	6.0	✓	5.1	✓	5.1	✓
s3_clnt_4	.03	-	4.8	-	9.8	✓	8.5	✓	10	✓	9.8	✓	9.8	✓
s3_clnt_4_BUG	.03	-	4.3	-	4.3	✓	3.5	✓	6.4	✓	4.3	✓	4.3	✓
s3_srvr_1	.03	-	4.1	-	3.3	✓	2.4	✓	21	✓	3.3	✓	3.3	✓
s3_srvr_1_BUG	.03	-	6.4	✓	2.2	✓	1.7	✓	4.8	✓	2.2	✓	2.2	✓
s3_srvr_2	.03	-	5.5	-	2.8	✓	2.4	✓	150	✓	2.8	✓	2.8	✓
s3_srvr_2_BUG	.03	-	6.2	✓	1.8	✓	1.7	✓	4.1	✓	1.8	✓	1.8	✓
s3_srvr_3	.03	-	5.6	-	3.6	-	2.6	-	9.0	✓	13	✓	13	✓
s3_srvr_4	.03	-	5.6	-	3.1	-	2.5	-	28	✓	31	✓	31	✓
s3_srvr_6	.03	-	6.6	-	14	-	250	✓	230	✓	240	✓	240	✓
s3_srvr_7	.03	-	6.3	-	14	-	200	✓	47	✓	61	✓	61	✓
s3_srvr_8	.03	-	6.1	-	2.8	✓	3.0	✓	23	✓	2.8	✓	2.8	✓
SSH total	0.51	0	90	3	110	13	510	15	570	17	420	17	420	17
bist_cell	.02	-	.65	-	9.4	-	9.2	-	210	✓	220	✓	220	✓
kundu	.02	-	120	-	7.3	✓	6.2	✓	900	-	7.3	✓	7.3	✓
kundu1_BUG	.02	-	18	-	2.0	✓	1.8	✓	15	✓	2.0	✓	2.0	✓
kundu2_BUG	.02	-	120	-	1.6	✓	1.9	✓	510	✓	1.6	✓	1.6	✓
pc_sfifo_1	.02	-	12	-	14	-	900	-	5.3	✓	19	✓	19	✓
pc_sfifo_2	.02	-	6.3	-	14	-	900	-	9.8	✓	24	✓	24	✓
token_ring.01	.02	-	16	-	1.7	-	1.6	-	8.1	✓	9.8	✓	9.8	✓
toy2_BUG	.03	-	30	-	2.7	✓	2.1	✓	65	✓	2.7	✓	2.7	✓

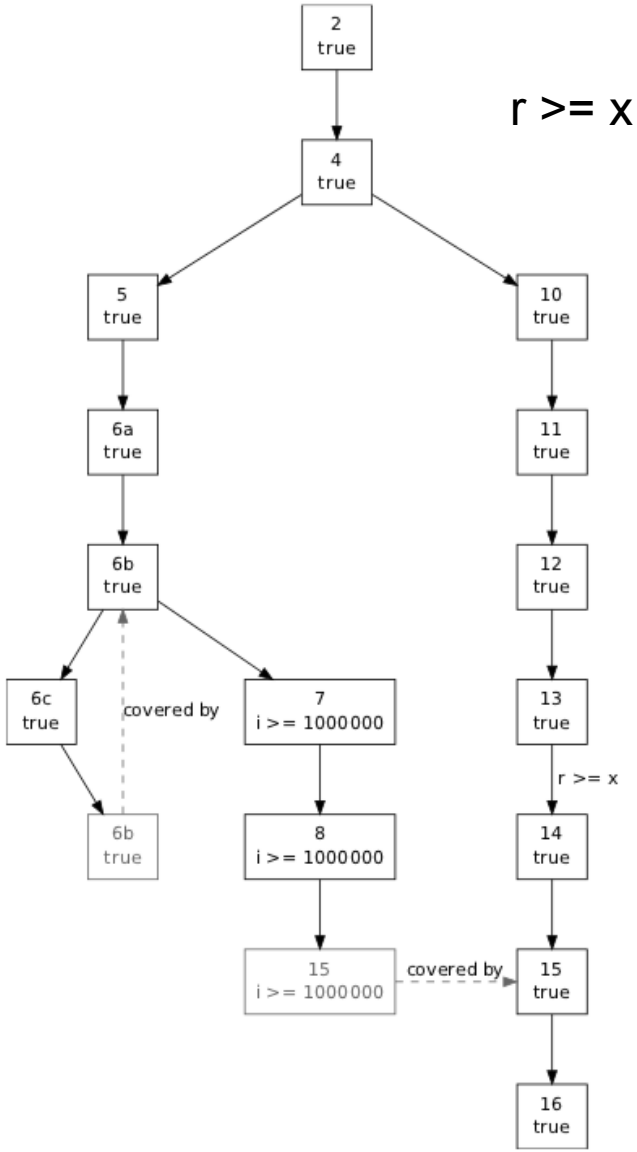
kbfiltr_simpl2_BUG	.11	✓	.12	✓	2.2	✓	2.1	✓	4.1	✓	2.2	✓	.11	✓
NT drivers total	3.4	9	3.4	9	39	9	920	9	94	10	54	10	31	10
s3_clnt_1	.03	-	4.3	-	9.8	✓	8.3	✓	8.1	✓	9.8	✓	9.8	✓
s3_clnt_1_BUG	.03	-	4.2	-	4.4	✓	3.6	✓	6.2	✓	4.4	✓	4.4	✓
s3_clnt_2	.03	-	4.6	-	10	✓	8.2	✓	7.2	✓	10	✓	10	✓
s3_clnt_2_BUG	.03	-	4.3	-	4.5	✓	3.5	✓	5.4	✓	4.5	✓	4.5	✓
s3_clnt_3	.03	-	5.3	-	9.7	✓	8.1	✓	5.8	✓	9.7	✓	9.7	✓
s3_clnt_3_BUG	.03	-	5.3	✓	5.1	✓	3.5	✓	6.0	✓	5.1	✓	5.1	✓
s3_clnt_4	.03	-	4.8	-	9.8	✓	8.5	✓	10	✓	9.8	✓	9.8	✓
s3_clnt_4_BUG	.03	-	4.3	-	4.3	✓	3.5	✓	6.4	✓	4.3	✓	4.3	✓
s3_srvr_1	.03	-	4.1	-	3.3	✓	2.4	✓	21	✓	3.3	✓	3.3	✓
s3_srvr_1_BUG	.03	-	6.4	✓	2.2	✓	1.7	✓	4.8	✓	2.2	✓	2.2	✓
s3_srvr_2	.03	-	5.5	-	2.8	✓	2.4	✓	150	✓	2.8	✓	2.8	✓
s3_srvr_2_BUG	.03	-	6.2	✓	1.8	✓	1.7	✓	4.1	✓	1.8	✓	1.8	✓
s3_srvr_3	.03	-	5.6	-	3.6	-	2.6	-	9.0	✓	13	✓	13	✓
s3_srvr_4	.03	-	5.6	-	3.1	-	2.5	-	28	✓	31	✓	31	✓
s3_srvr_6	.03	-	6.6	-	14	-	250	✓	230	✓	240	✓	240	✓
s3_srvr_7	.03	-	6.3	-	14	-	200	✓	47	✓	61	✓	61	✓
s3_srvr_8	.03	-	6.1	-	2.8	✓	3.0	✓	23	✓	2.8	✓	2.8	✓
SSH total	0.51	0	90	3	110	13	510	15	570	17	420	17	420	17
bist_cell	.02	-	.65	-	9.4	-	9.2	-	210	✓	220	✓	220	✓
kundu	.02	-	120	-	7.3	✓	6.2	✓	900	-	7.3	✓	7.3	✓
kundu1_BUG	.02	-	18	-	2.0	✓	1.8	✓	15	✓	2.0	✓	2.0	✓
kundu2_BUG	.02	-	120	-	1.6	✓	1.9	✓	510	✓	1.6	✓	1.6	✓
pc_sfifo_1	.02	-	12	-	14	-	900	-	5.3	✓	19	✓	19	✓
pc_sfifo_2	.02	-	6.3	-	14	-	900	-	9.8	✓	24	✓	24	✓
token_ring.01	.02	-	16	-	1.7	-	1.6	-	8.1	✓	9.8	✓	9.8	✓
toy2_BUG	.03	-	30	-	2.7	✓	2.1	✓	65	✓	2.7	✓	2.7	✓
transmitter.01.BUG	.02	-	5.3	✓	1.7	✓	1.5	✓	2.2	✓	1.7	✓	1.7	✓
transmitter.02.BUG	.02	-	21	✓	2.0	✓	1.7	✓	5.1	✓	2.0	✓	2.0	✓
transmitter.03.BUG	.03	-	92	-	2.0	✓	1.8	✓	36	✓	2.0	✓	2.0	✓
transmitter.04.BUG	.03	-	270	-	2.2	✓	2.5	✓	900	-	2.2	✓	2.2	✓
transmitter.05.BUG	.04	-	670	✓	2.5	✓	2.3	✓	900	-	2.5	✓	2.5	✓
transmitter.06.BUG	.04	-	900	-	3.4	✓	3.0	✓	900	-	3.4	✓	3.4	✓
transmitter.07.BUG	.04	-	900	-	3.8	✓	3.3	✓	900	-	3.8	✓	3.8	✓
transmitter.08.BUG	.05	-	900	-	6.3	✓	4.6	✓	900	-	6.3	✓	6.4	✓
transmitter.09.BUG	.05	-	900	-	11	✓	9.4	✓	900	-	11	✓	11	✓
SystemC total	0.49	0	5000	3	88	13	1900	13	7200	10	320	17	320	17
Total	4.4	9	5100	15	230	35	3300	37	7800	37	790	44	770	44

Predicate Analysis not Effective

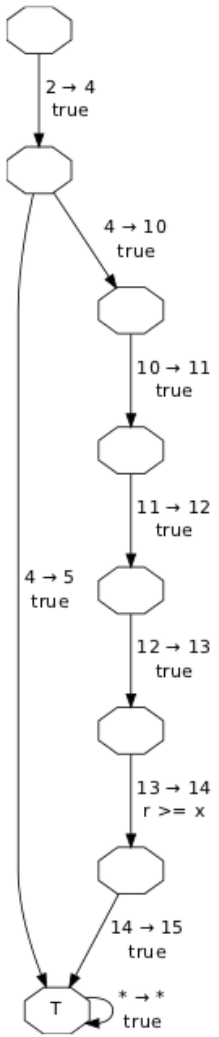
```
1  int main() {
2    int p = nondet_int();
3
4    if (p) {
5      int i;
6      for (i = 0; i < 1000000; i++);
7      assert(i >= 1000000);
8
9    } else {
10     int x = 5;
11     int y = 6;
12     int r = x * y;
13     assert(r >= x);
14   }
15   return 0;
16 }
```

Output Condition after Predicate

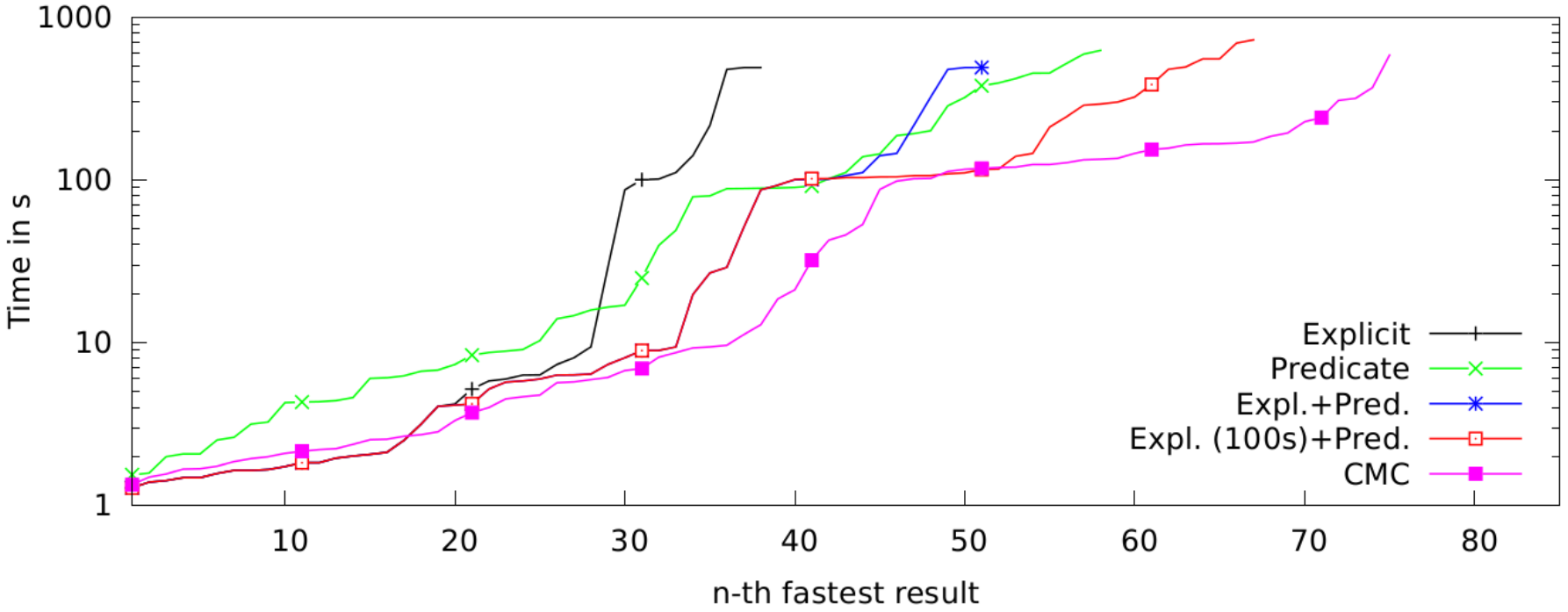




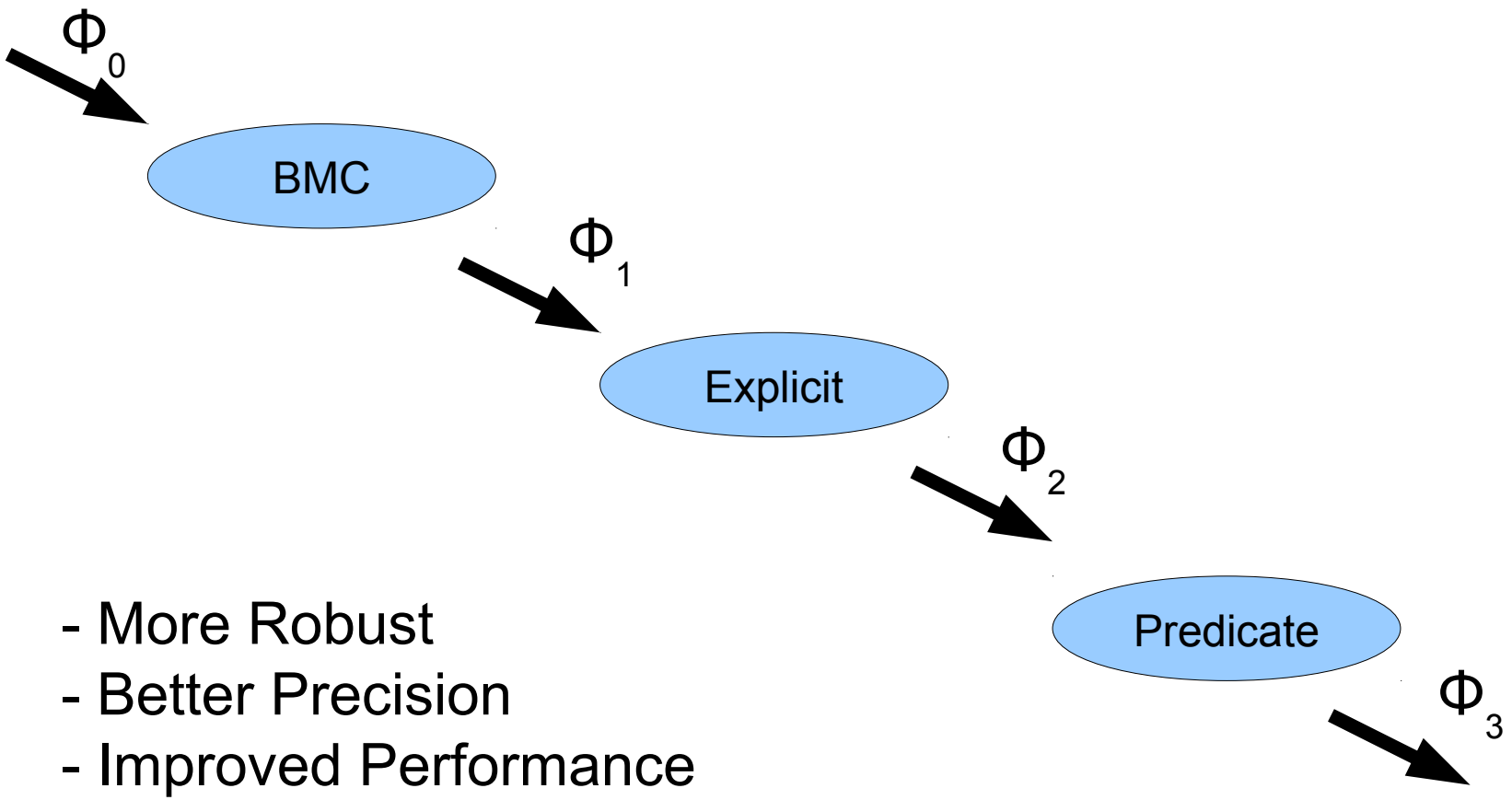
r >= x not proved



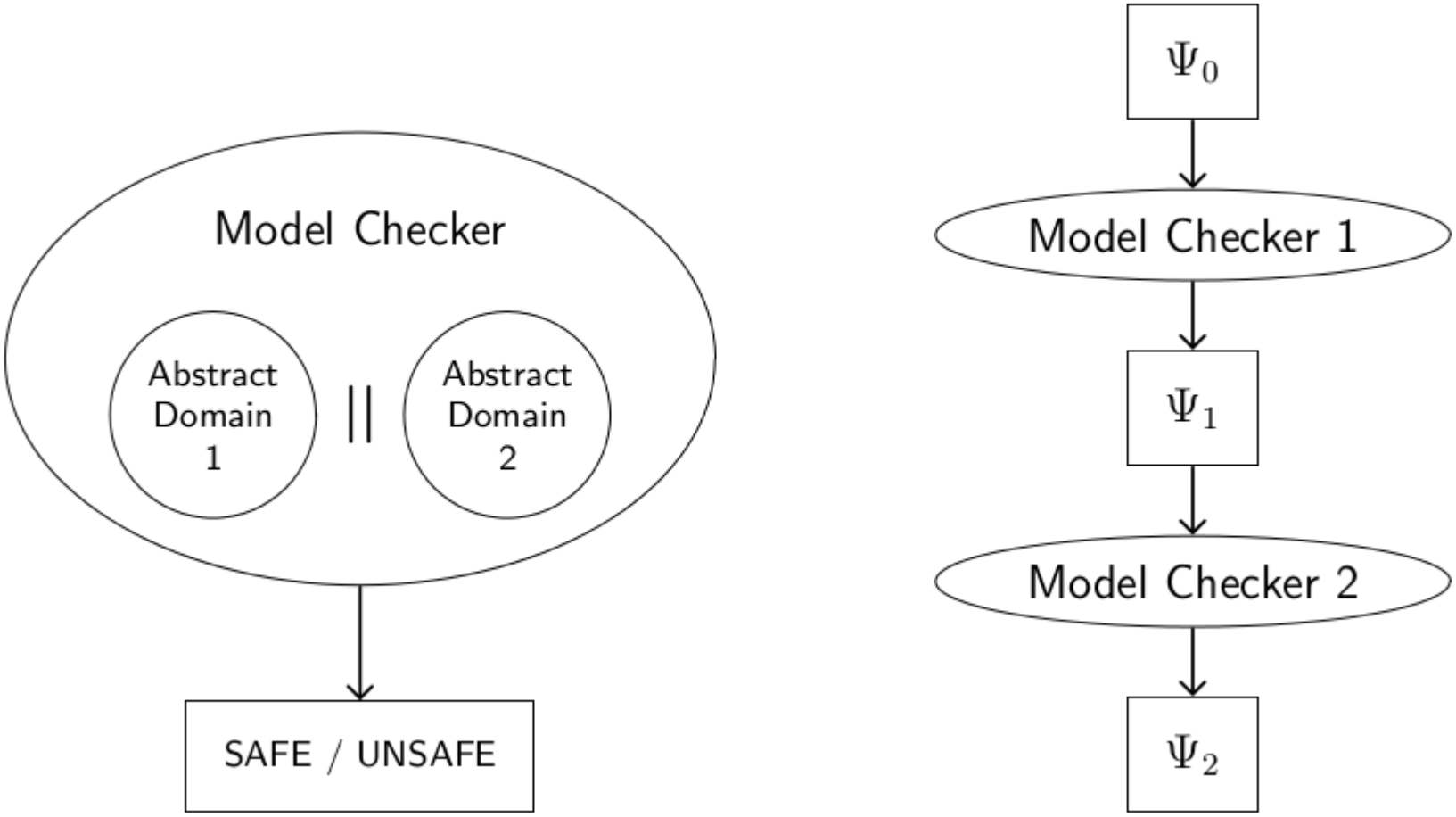
Program	Explicit		Predicate		Comb. A Explicit + Predicate		Conditional MC Explicit + Predicate	
token_ring.01.BUG	1.6	-	4.0	✓	5.6	✓	2.6	✓
token_ring.01	1.9	-	8.9	✓	11	✓	2.4	✓
token_ring.02.BUG	2.3	-	24	✓	26	✓	4.3	✓
token_ring.02	1.7	-	900	-	900	-	4.0	✓
token_ring.03.BUG	2.6	-	900	-	900	-	5.4	✓
token_ring.03	2.5	-	900	-	900	-	5.1	✓
token_ring.04.BUG	2.7	-	900	-	900	-	9.1	✓
token_ring.04	2.5	-	900	-	900	-	8.5	✓
token_ring.05.BUG	3.8	-	900	-	900	-	16	✓
token_ring.05	3.2	-	900	-	900	-	17	✓
token_ring.06.BUG	4.8	-	900	-	900	-	34	✓
token_ring.06	5.4	-	900	-	900	-	40	✓
token_ring.07.BUG	9.1	-	900	-	900	-	140	✓
token_ring.07	8.3	-	900	-	900	-	180	✓
token_ring.08.BUG	25	-	900	-	900	-	580	✓
token_ring.08	6.0	-	900	-	900	-	720	✓
token_ring.09.BUG	120	-	900	-	900	-	900	-
token_ring.09	130	-	900	-	900	-	900	-
mem_slave_tlm.1	2.0	-	900	-	900	-	5.2	✓
mem_slave_tlm.2	2.8	-	900	-	900	-	6.3	✓
mem_slave_tlm.3	3.0	-	900	-	900	-	7.5	✓
mem_slave_tlm.4	3.4	-	900	-	900	-	8.1	✓
mem_slave_tlm.5	3.8	-	900	-	900	-	10	✓
toy	2.6	-	900	-	900	-	7.6	✓
Total	350	0	19000	3	19000	3	3600	22


1000
100
10
1
Time in s
10
20
30
40
50
60
70
80
n-th fastest result
Explicit
Predicate
Expl.+Pred.
Expl. (100s)+Pred.
CMC

Sequential Composition



- More Robust
- Better Precision
- Improved Performance
- More Flexible (combinations of binaries)


Model Checker
Abstract Domain 1
||
Abstract Domain 2
SAFE / UNSAFE
Ψ_0
Model Checker 1
Ψ_1
Model Checker 2
Ψ_2

Towards a Unifying Framework

...

CPAchecker – History

- 2003 – 2008 BLAST (UC Berkeley)
  Last version 2.5 released in 2008
  by B, Jhala, Majumdar, Henzinger

- 2008 – 2012 CPAchecker (SFU, Passau)

- Complete reimplementation of BLAST
  - New, more flexible architecture
  - More efficient algorithms

CPAchecker – Framework

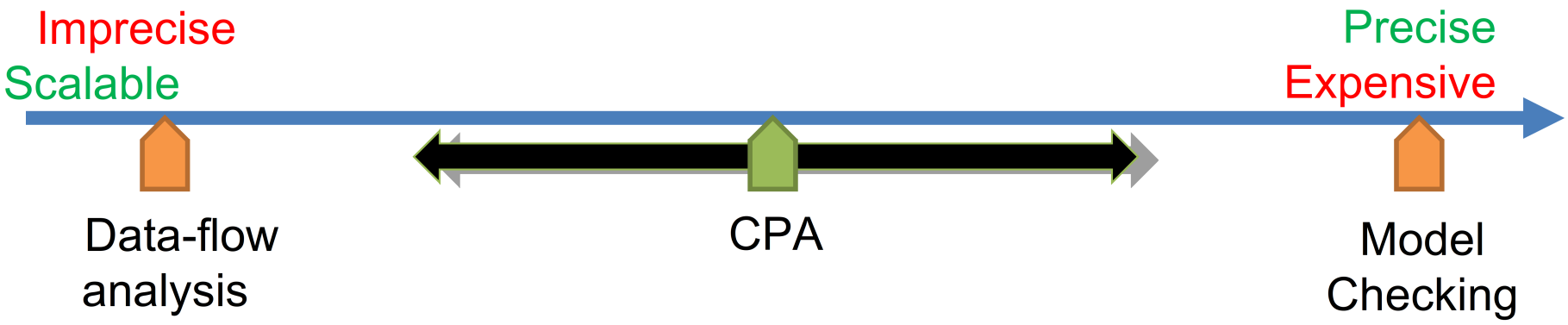
- Software model checker
- Open source (Apache 2), written in Java
- Follows strictly the concept of **Configurable Program Analysis** [CAV'07, ASE'08]
- Input language: C

CPAchecker – Features

- Integrated most successful SMC ideas:
  - **Predicate analysis** (1997)
  - **CEGAR** (2000)
  - **Lazy abstraction** (2002)
  - **Interpolation** for predicate discovery (2004)
  - **Configurable program analysis** (2007)
  - **Large block-encoding** (2009)
- Strongest domain:
  Predicate Analysis (faster than BLAST)

Configurable Program Analysis

- Better combination of abstractions
  ➔ Configurable Program Analysis
  [B / Henzinger / Theoduloz CAV'07]



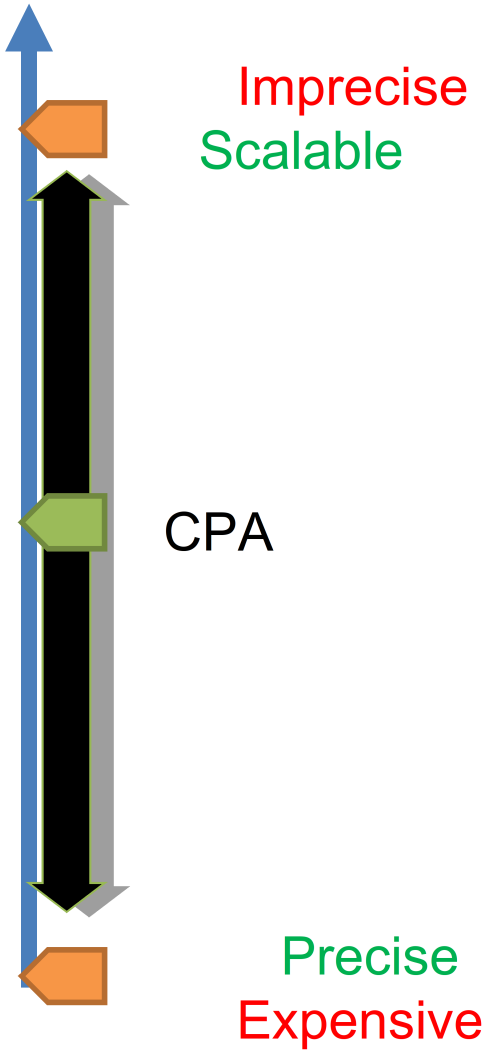
Unified framework that enables intermediate algorithms

Dynamic Precision Adjustment

Better fine tuning of the precision of abstractions
➔ Adjustable Precision
[B / Henzinger / Theoduloz ASE'08]

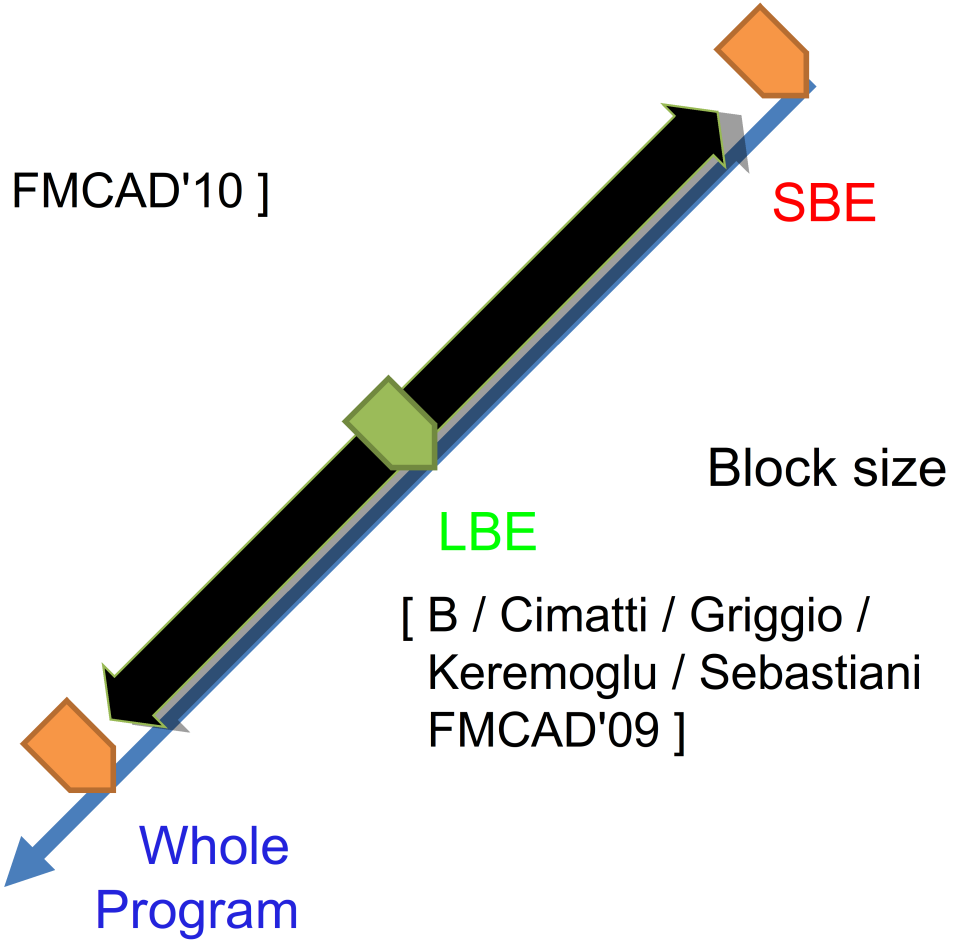
Unified framework enables:
- switch on and off different analyses, and can
- adjust each analysis separately

- Not only **refine**, also **abstract**!



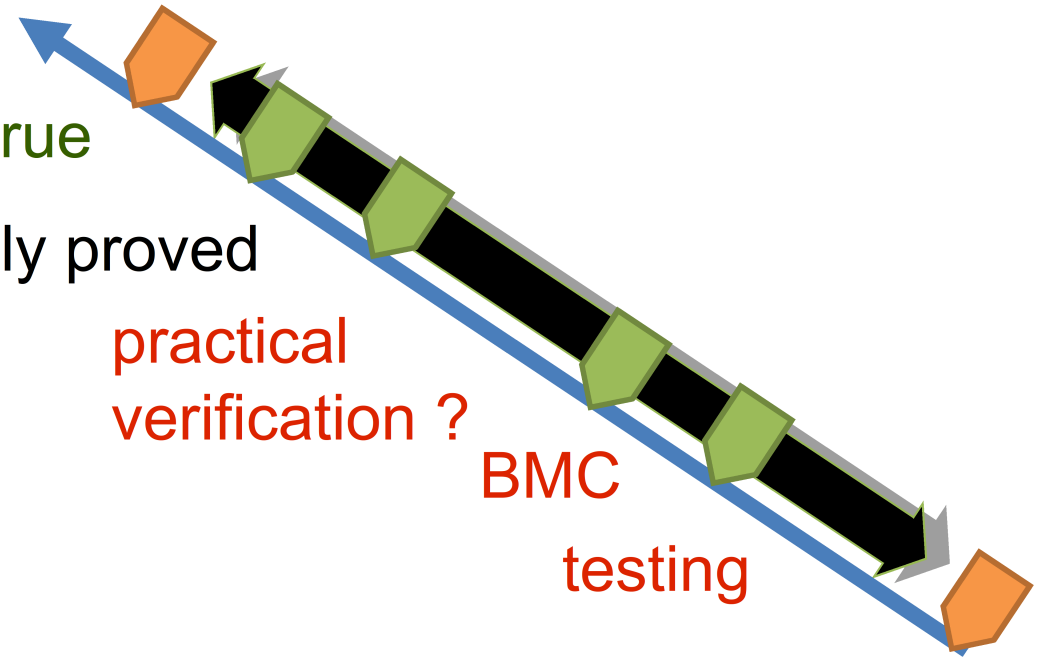
Adjustable Block Size

[B / Keremoglu / Wendler FMCAD'10]



Orthogonal Improvement: Block Caching [Wehrheim / Wonisch 2011]

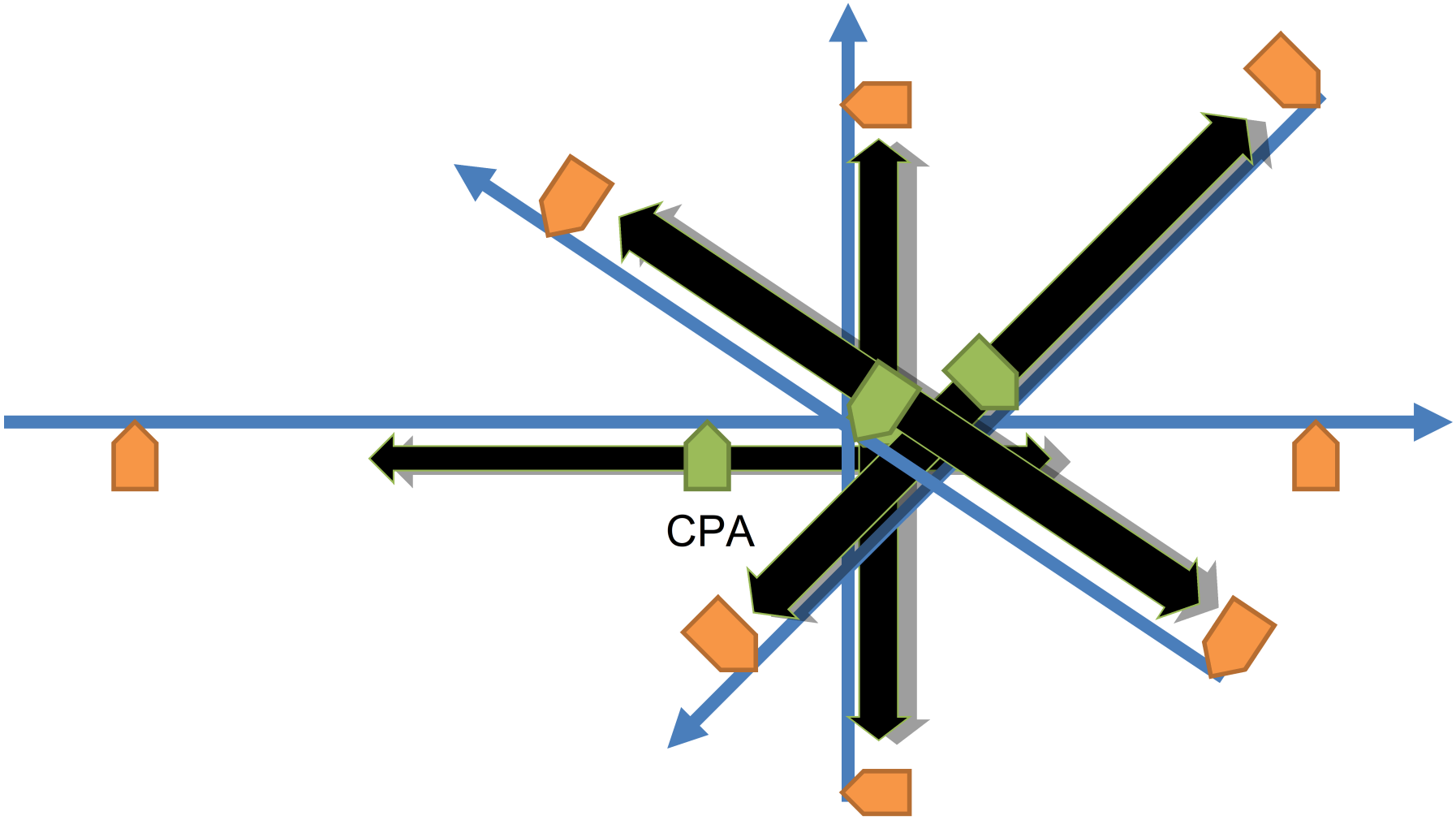
Conditional Verification



Weak Condition =
Good Model Checker

Condition

CPAchecker



Coming Soon

- Competition in Software Verification
  at TACAS 2013 (March 2013)
  http://sv-comp.sosy-lab.org

- Distributed Model Checking
  → Model Checking in the Cloud

Summary

- Conditional Model Checking
  → Resource-aware (green)
  → Terminates with Useful Results
  → Effective Sequential Composition
  → Unified View on Existing Approaches
- CPAchecker – Verification Framework
  → Designed for Extension & Plug-in
  → One of the Most Efficient
  Software Model Checkers

Additional Material

Adjustable-Block Encoding

Abstract Successors

Abstract state: (Φ, ψ)

Φ: Strongest Post ψ: Abstract Formula

Abstract Successors

Abstract state: (Φ, ψ)

Φ: Strongest Post ψ: Abstract Formula

Example:

Precision: { x > 0 }

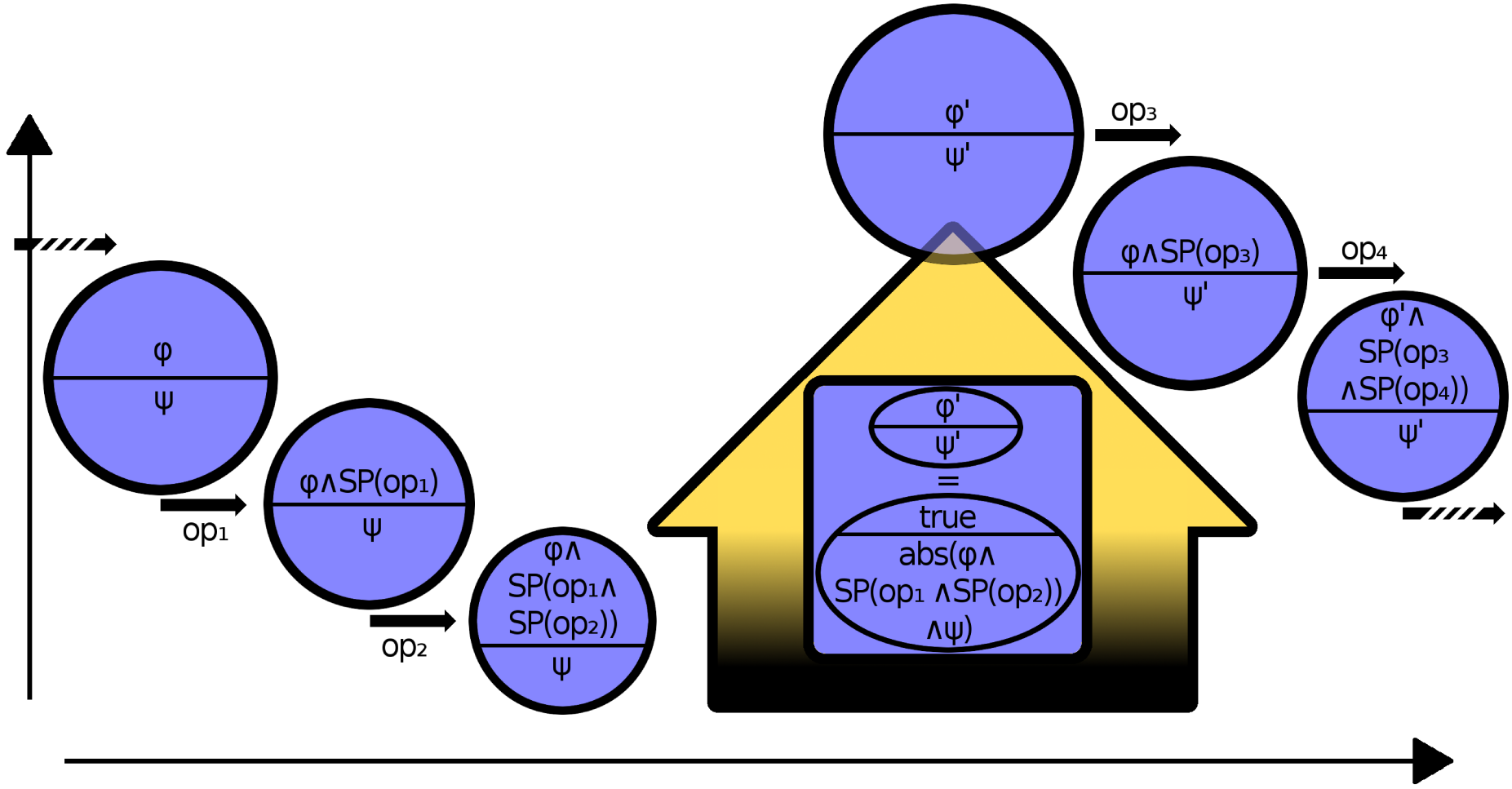
Current abstract state: (true, x > 0)

CFA edge: x := 1

Successor abstract state: (x = 1, x > 0)

After predicate abstraction: (true, x > 0)

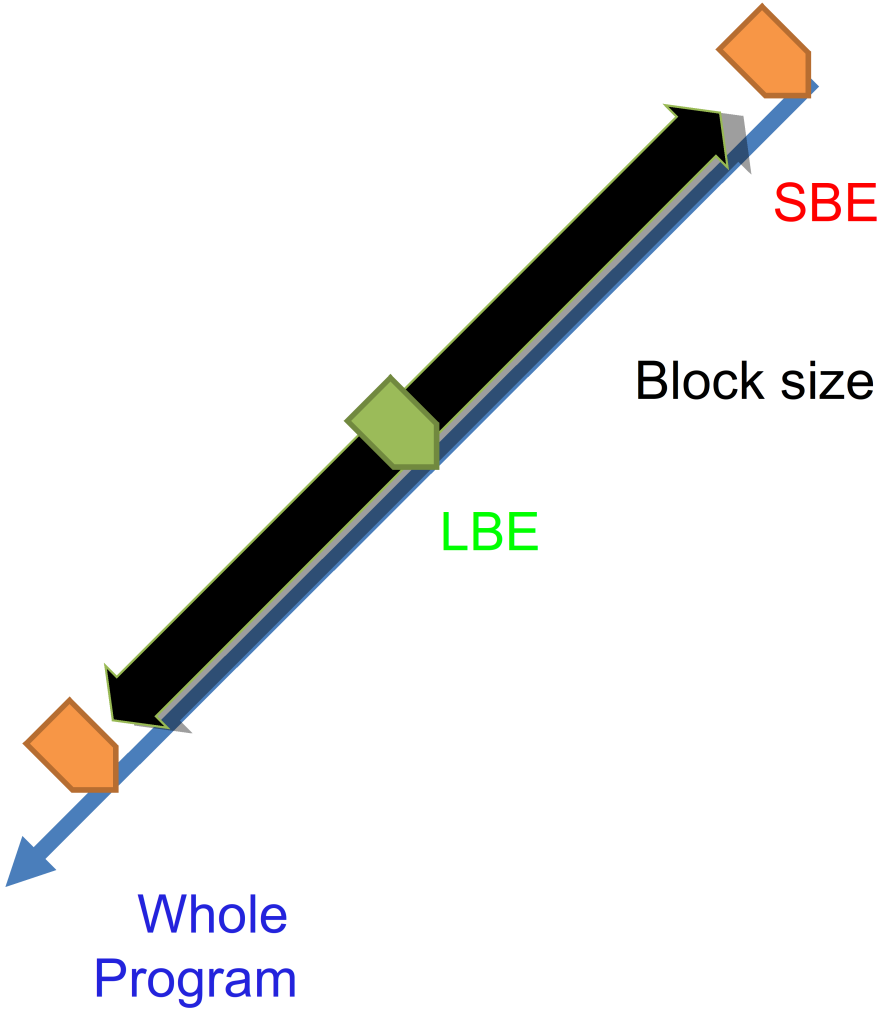
Abstract Successors



Adjustable-Block Encoding

- Boolean Abstraction (not Cartesian)
- Arbitrary Block Size
- We can use more power of SMT
  SMT Fans: Attention!
- Disjunctions not handled explicitly
  ART not forced to grow exponentially
- Reduced number of abstractions
- Reduced number of refinements

Adjustable Block Size



CPAchecker - Summary

- Unification of several approaches
  → reduced to their essential properties
- Allow experimentation with new configurations that we would never think of
- Flexible implementation as framework

http://cpachecker.sosy-lab.org

Dirk Beyer (Uni Passau)

Gregor Endler (Uni Passau)

Alberto Griggio (Uni Trento)

Andreas Holzer (TU Wien)

Erkan Keremoglu (SFU)

Stefan Löwe (Uni Passau)

Alexander v. Rhein (Uni Passau)

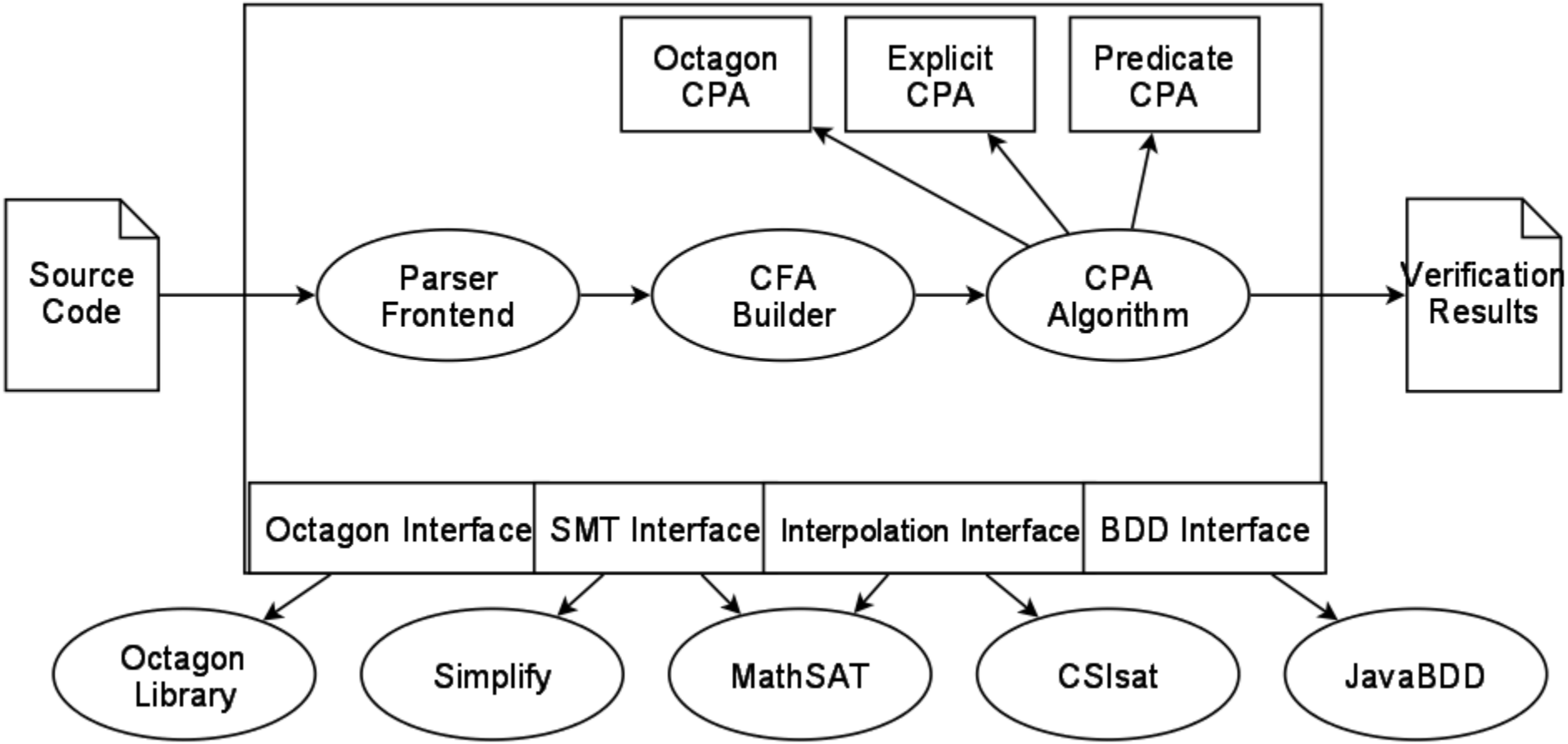
Michael Tautschnig (Oxford Uni)

Gregory Theoduloz (EPFL)

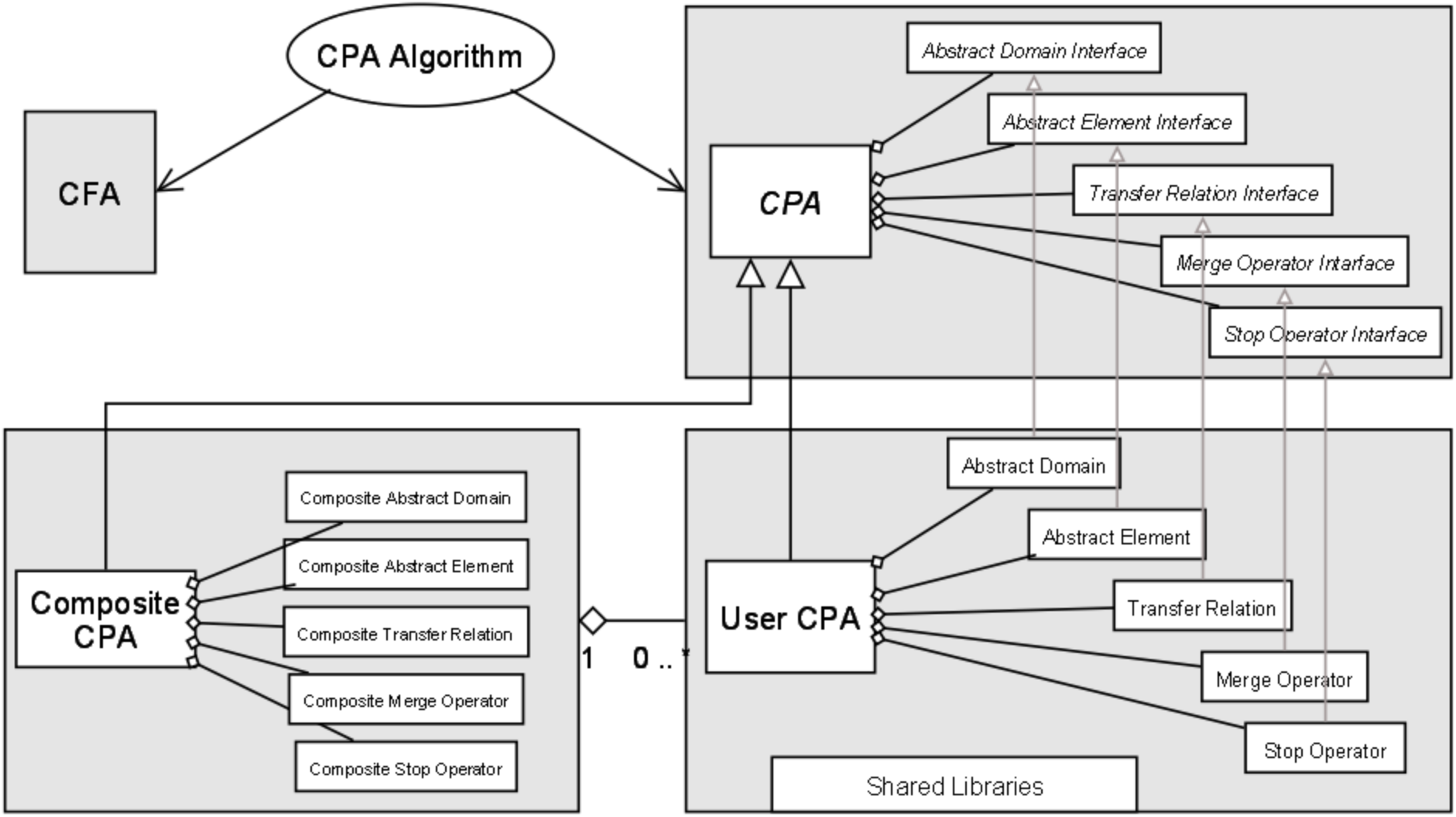
Philipp Wendler (Uni Passau)

Daniel Wonisch (Uni Paderborn)

CPAchecker Architecture and Flow



CPAchecker - Design



Future: Powerful Parameters ...

	SBE	LBE
SAT check after	error	error
Abstraction after	1	func/loop
Unroll loops	no	no
Inline functions	no	no
Merge	never	non-abstraction

Future: Powerful Parameters ...

	SBE	LBE	BMC
SAT check after	error	error	threshold
Abstraction after	1	func/loop	never
Unroll loops	no	no	yes
Inline functions	no	no	yes
Merge	never	non-abstraction	always

Future: Powerful Parameters ...

	SBE	LBE	McMillan	BMC
SAT check after	error	error	every	threshold
Abstraction after	1	func/loop	never	never
Unroll loops	no	no	yes	yes
Inline functions	no	no	yes	yes
Merge	never	non-abstraction	never	always