Explicit-State Model Checking Based on CEGAR and Interpolation

Dirk Beyer – Stefan Löwe

UNIVERSITÄT PASSAU
Fakultät für Informatik und Mathematik

State of the art: SV-COMP'12 participants

Predicate Abstraction

- BLAST
- CPAchecker ABE
- CPAchecker ABM
- QARMC-HSF
- SATABS
- Wolverine

Bounded Model Checking

- ESBMC
- FShell
- LLBMC

All these rely on expensive calls to underlying decision procedure

… Dramatization ...

mem_slave_tlm.1.cil.c	safe	120	out of memory	270	out of memory	270	timeout	910	safe
mem_slave_tlm.2.cil.c	safe	560	segmentation fault	210	out of memory	410	timeout	910	safe
mem_slave_tlm.3.cil.c	timeout	910	out of memory	260	timeout	900	timeout	910	safe
mem_slave_tlm.4.cil.c	unknown	220	segmentation fault	390	timeout	900	timeout	910	safe
mem_slave_tlm.5.cil.c	unknown	220	out of memory	300	timeout	900	timeout	910	safe
pipeline.cil.c	timeout	910	safe	16	timeout	900	timeout	900	safe
token_ring.01.cil.c	safe	3.0	safe	3.9	safe	6.9	timeout	910	unsafe
token_ring.02.cil.c	safe	93	safe	7.1	safe	49	timeout	910	unsafe
token_ring.03.cil.c	safe	580	safe	33	timeout	900	timeout	900	unsafe
token_ring.04.cil.c	timeout	910	safe	140	timeout	900	timeout	900	unsafe
token_ring.05.cil.c	timeout	910	out of memory	300	out of memory	550	timeout	900	unsafe
token_ring.06.cil.c	timeout	910	out of memory	380	segmentation fault	650	timeout	910	unsafe
token_ring.07.cil.c	timeout	910	timeout	900	timeout	900	timeout	910	unsafe
token_ring.08.cil.c	timeout	910	timeout	900	timeout	900	timeout	910	unsafe
token_ring.09.cil.c	timeout	910	timeout	900	timeout	900	timeout	900	timeout
token_ring.10.cil.c	timeout	910	timeout	900	timeout	900	timeout	900	timeout
token_ring.11.cil.c	timeout	910	out of memory	880	timeout	900	timeout	900	timeout

How to circumvent?

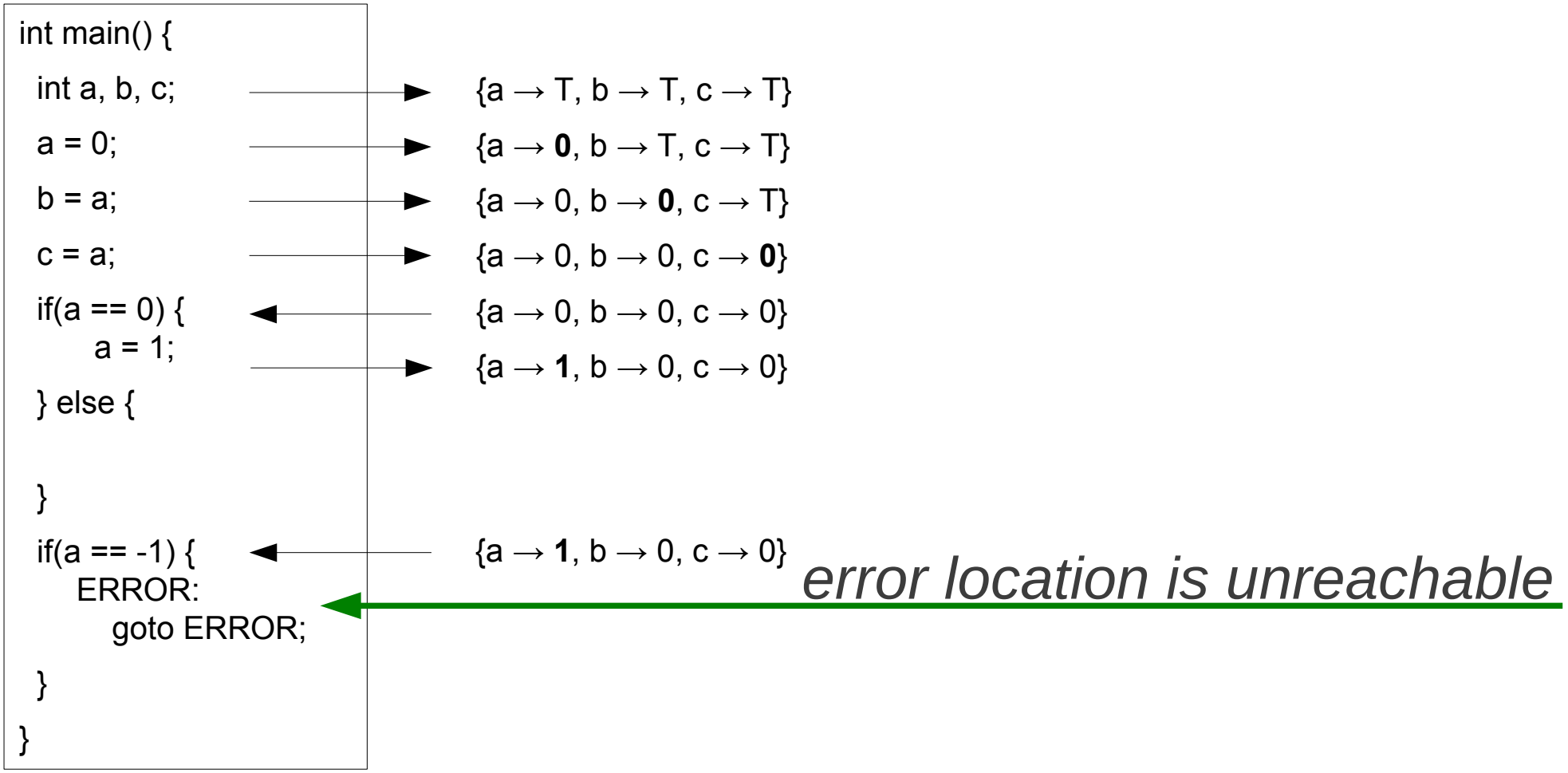
- Use a less expensive domain
  - Signs (-, isZero, +)
  - Explicit values ({ a → 1, b → -5, c → T })

➢ More efficient successor computation

➢ Less precise state representation

state-space explosion still a major issue

Explicit-Value Model Checking



The Good

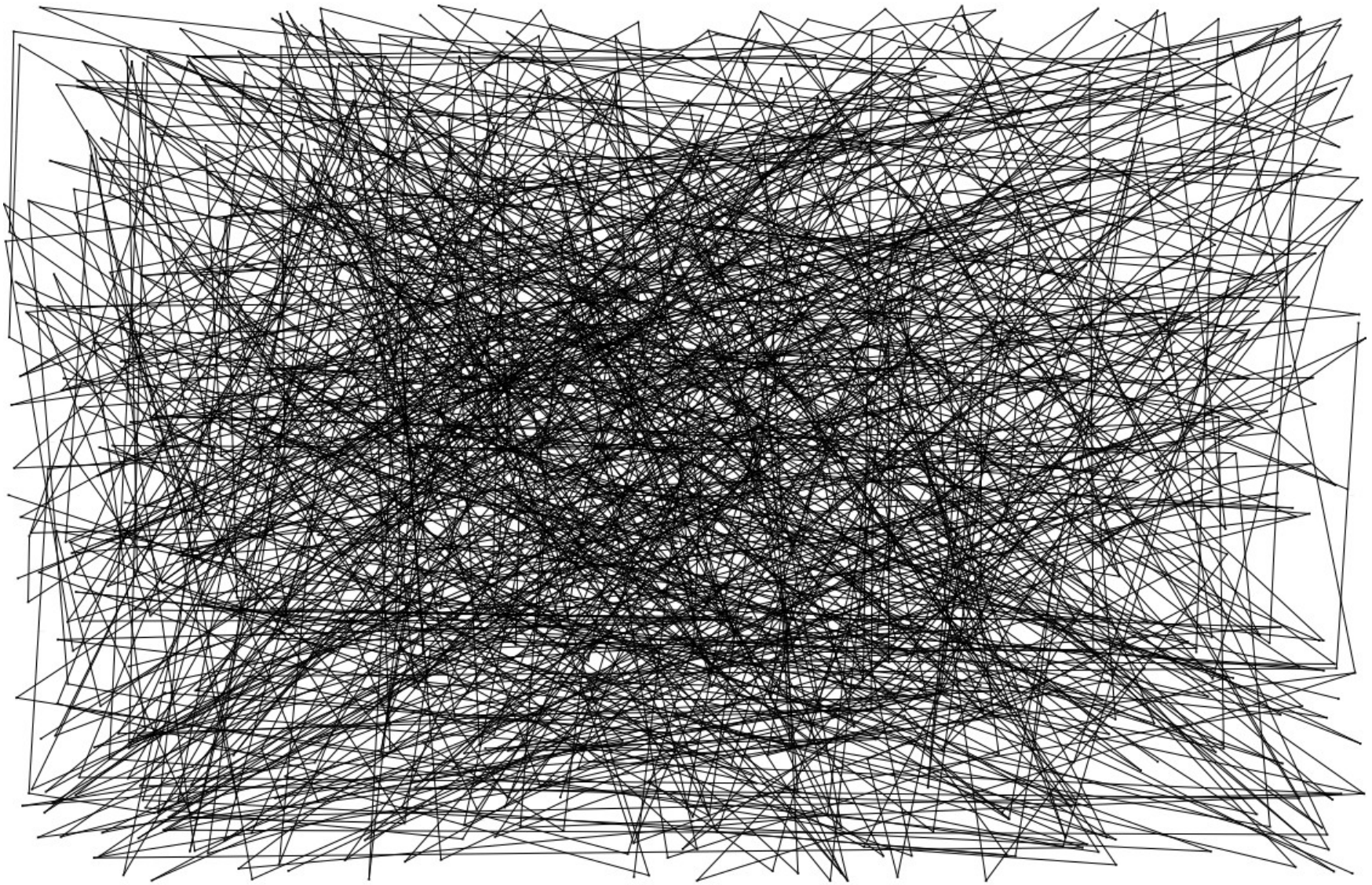
Tool	CPAchecker 1.1-svn					
Test	explicitAnalysis-cbmc					
test/programs/benchmarks/	status	cputime	walltime		total	cpa time
ntdrivers/cdaudio.BUG.i.cil.c	unsafe	6.54	12.43		11.231s	0.083s
ntdrivers/diskperf.BUG.i.cil.c	unsafe	4.69	10.52		8.962s	0.102s
ntdrivers/floppy.BUG.i.cil.c	unsafe	46.73	51.75		50.633s	0.106s
ntdrivers/kbfiltr.BUG.i.cil.c	unsafe	136.67	140.90		139.515s	0.226s
ntdrivers/parport.BUG.i.cil.c	unsafe	7.06	18.60		16.810s	0.134s
ntdrivers/cdaudio.i.cil.c	safe	8.26	11.31		9.725s	0.586s
ssh/s3_srvr.blast.01.BUG.i.cil.c	unsafe	3.07	2.77		1.778s	0.079s
ssh/s3_srvr.blast.02.BUG.i.cil.c	unsafe	2.93	2.84		1.929s	0.071s
ssh/s3_srvr.blast.03.BUG.i.cil.c	unsafe	3.00	3.56		1.836s	0.073s
ssh/s3_srvr.blast.04.BUG.i.cil.c	unsafe	3.07	2.78		1.712s	0.071s
ssh/s3_srvr.blast.01.i.cil.c	safe	61.66	58.75		49.367s	47.898s
ssh/s3_srvr.blast.02.i.cil.c	safe	58.18	54.96		44.955s	43.701s
ssh/s3_srvr.blast.06.i.cil.c	safe	821.48	802.42		763.905s	762.368s
ssh/s3_srvr.blast.07.i.cil.c	safe	589.96	587.35		547.721s	545.769s

Scores some 200 points in SV-COMP setting – winner had 280

The Bad

Tool	CPAchecker 1.1-svn				
Test	explicitAnalysis-cbmc				
test/programs/benchmarks/	status	cputime	walltime	total	cpa time
ldv-drivers/usb_urb-drivers-media-dvb-tt...i.pp.cil.c	timeout	1491.19	1441.18		
ldv-drivers/usb_urb-drivers-net-can-usb-...i.pp.cil.c	timeout	909.99	916.13	915 205s	0 409s
ldv-drivers/usb_urb-drivers-net-usb-catc...i.pp.cil.c	timeout	1028.72	1030.08	1027 456s	0 887s
ldv-drivers/usb_urb-drivers-staging-lirc...i.pp.cil.c	timeout	938.86	939.59	928 819s	2 338s
ldv-drivers/usb_urb-drivers-usb-misc-iow...i.pp.cil.c	timeout	1017.04	998.04		
ldv-drivers/module_get_put-drivers-atm-e...i.pp.cil.c	timeout	934.87	934.88	908 204s	899 954s
ldv-drivers/module_get_put-drivers-block...i.pp.cil.c	timeout	1016.88	1030.26		
ldv-drivers/module_get_put-drivers-block...i.pp.cil.c	timeout	937.89	922.41	902 830s	899 955s
ldv-drivers/module_get_put-drivers-bluet...i.pp.cil.c	timeout	1016.43	889.71		
ldv-drivers/module_get_put-drivers-char-...i.pp.cil.c	timeout	6.45	5.18	3 897s	1 233s
ldv-drivers/module_get_put-drivers-gpu-d...i.pp.cil.c	out of memory	605.10	600.54		
ldv-drivers/module_get_put-drivers-hid-h...i.pp.cil.c	timeout	1017.02	930.81		
ldv-drivers/module_get_put-drivers-hwmon...i.pp.cil.c	out of memory	983.87	937.24		
ldv-drivers/module_get_put-drivers-net-a...i.pp.cil.c	timeout	1016.98	959.73		
ldv-drivers/module_get_put-drivers-net-p...i.pp.cil.c	timeout	1016.61	1004.29		
ldv-drivers/module_get_put-drivers-net-s...i.pp.cil.c	timeout	953.29	951.86	908 554s	899 806s
ldv-drivers/module_get_put-drivers-usb-...i.pp.cil.c	timeout	950.36	940.60	910 871s	899 944s
ldv-drivers/module_get_put-drivers-staging...i.pp.cil.c	timeout	967.40	959.23	912 302s	899 948s
ldv-drivers/module_get_put-drivers-video...i.pp.cil.c	timeout	931.31	930.36	907 460s	899 943s

The Ugly



Explicit-Value Model Checking

Up to now: plain and simple

- Abstraction
- Counterexample-Guided Abstraction Refinement
- Interpolation

All known in the predicate domain for years

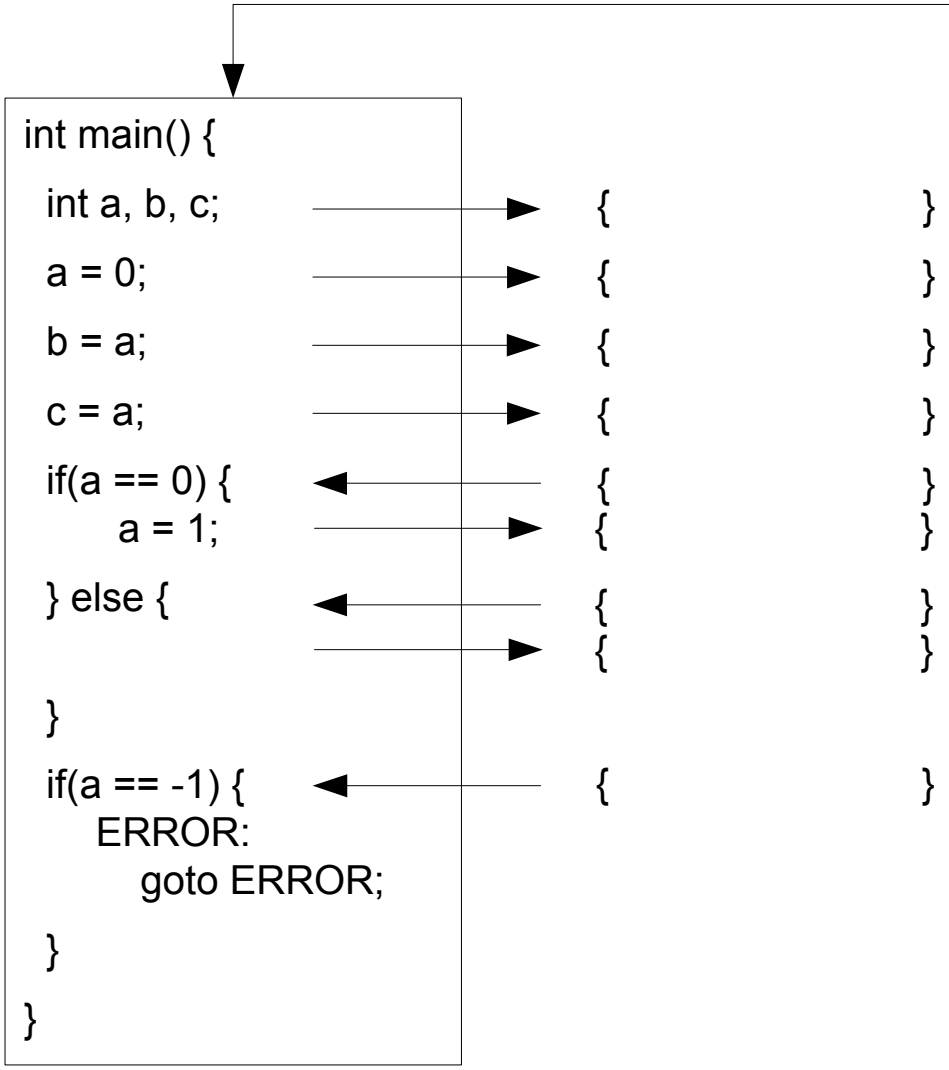
Explicit-Value Model Checking

As of now: with CEGAR and Interpolation

- Abstraction – Easy, just drop information
- Counterexamples – We get these for free
- Refinement – This is the hardest part

➢ Explicit-Value Model Checking
based on *CEGAR*
and *Interpolation*

Abstraction



This spurious counterexample trace will always be reported …

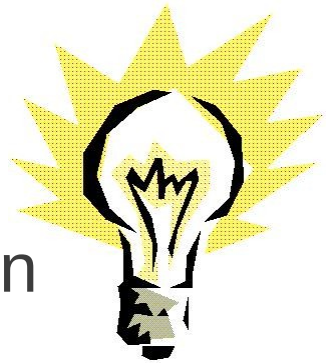
… unless we define a *precision π*, i.e. a *mapping from program locations to a set of variable identifiers*, e.g. {N2 → {a, b}, N7 → {a, c}}

Craig Interpolation

For a pair of *formulas* φ^- and φ^+ such that,
$\varphi^- \wedge \varphi^+$ is *unsatisfiable*, a Craig interpolant
ψ is a *formula* that fulfills
the following requirements:

1) φ^- implies ψ
2) $\psi \wedge \varphi^+$ is *unsatisfiable*
3) ψ only contains symbols that are common to both φ^- and φ^+.

→ use this for the Explicit Domain

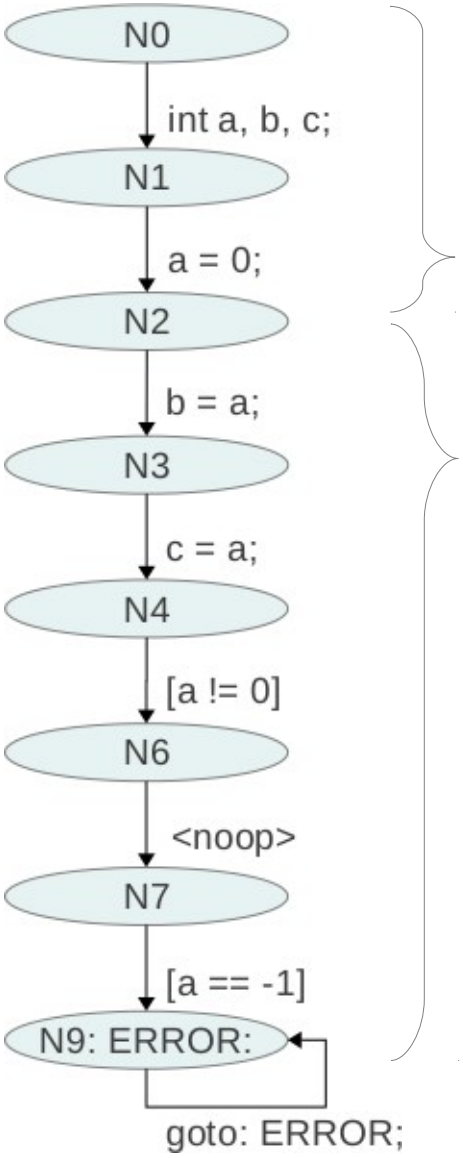


“Explicit” Craig Interpolation (1)

For a pair of *path assignments* φ^- and φ^+ such that, φ^- and φ^+ are *contradicting*, a Craig interpolant ψ is a *variable assignment* that fulfills the following requirements:

1) φ^- implies ψ
2) ψ and φ^+ are *contradicting*
3) ψ only contains symbols that are common to both φ^- and φ^+.

„Explicit“ Craig Interpolation (2)



✔ Check if path is infeasible

$\varphi^-_2 = \{a \rightarrow 0\}$

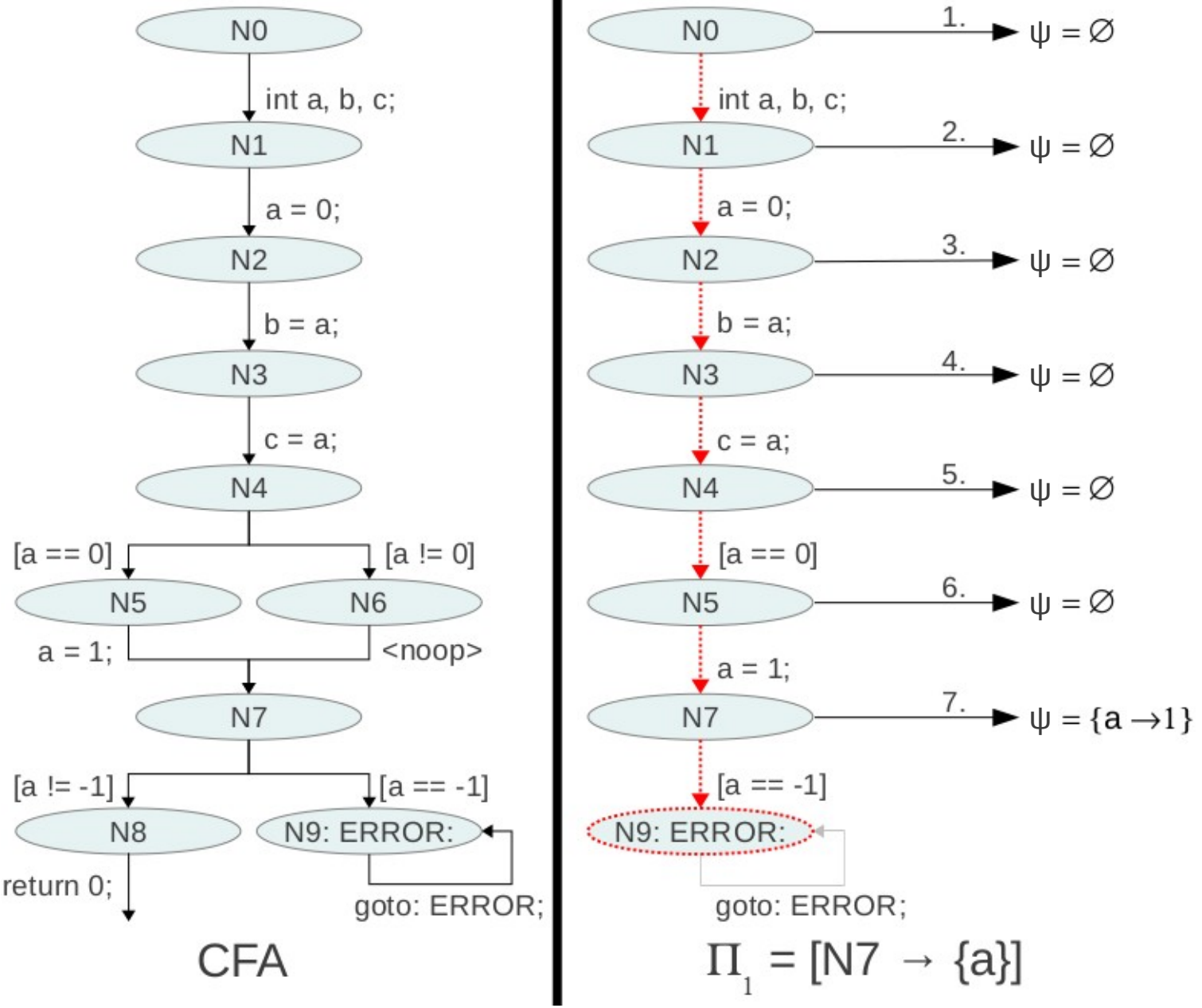
$\varphi^+_2 = \{a \rightarrow -1\}$

$\psi = \{a \rightarrow 0\}$

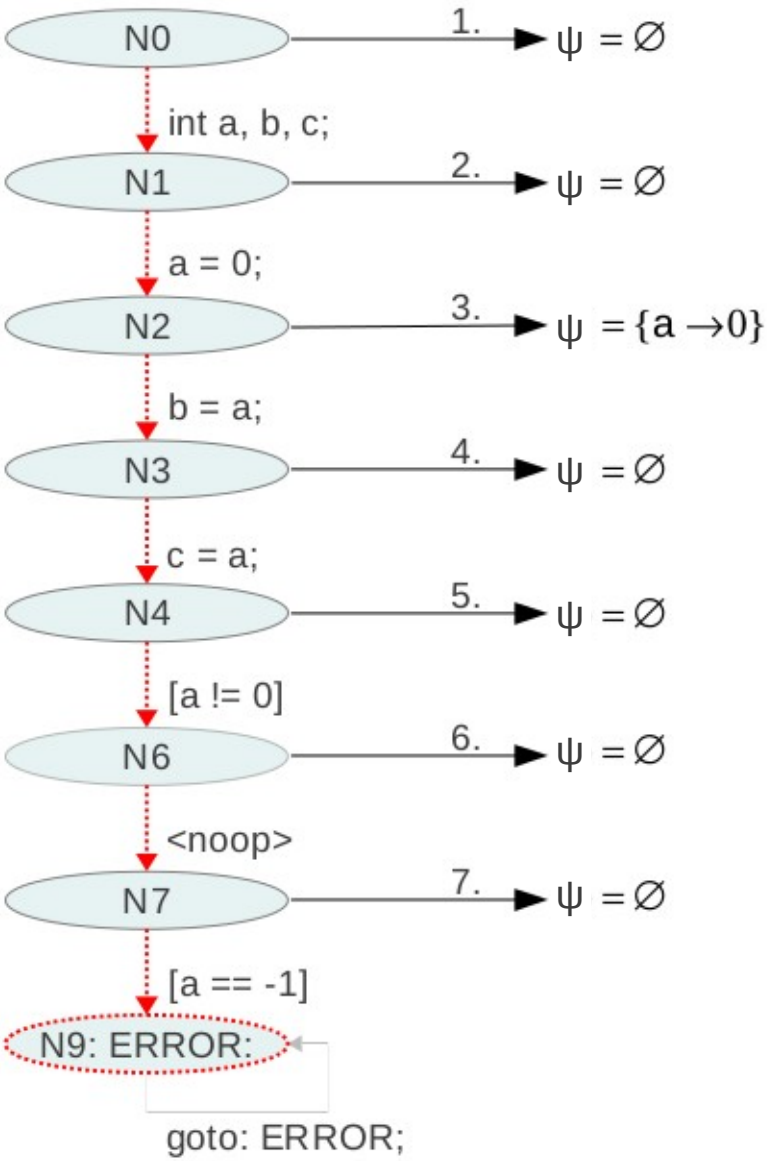
- ✔ φ^-_2 implies ψ
- ✔ φ^-_2 and φ^+_2 are contradicting
- ✔ common symbols

➢ Add [N2 → {a}] to the precision

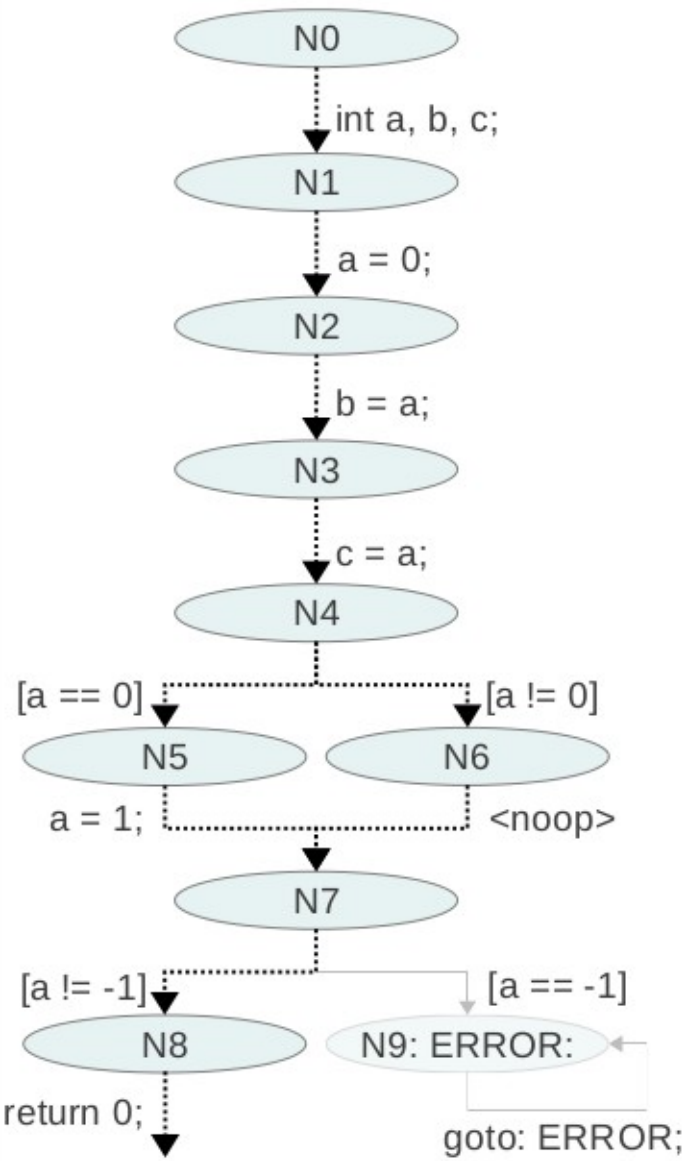
„Explicit“ Craig Interpolation (3)



„Explicit“ Craig Interpolation (4)



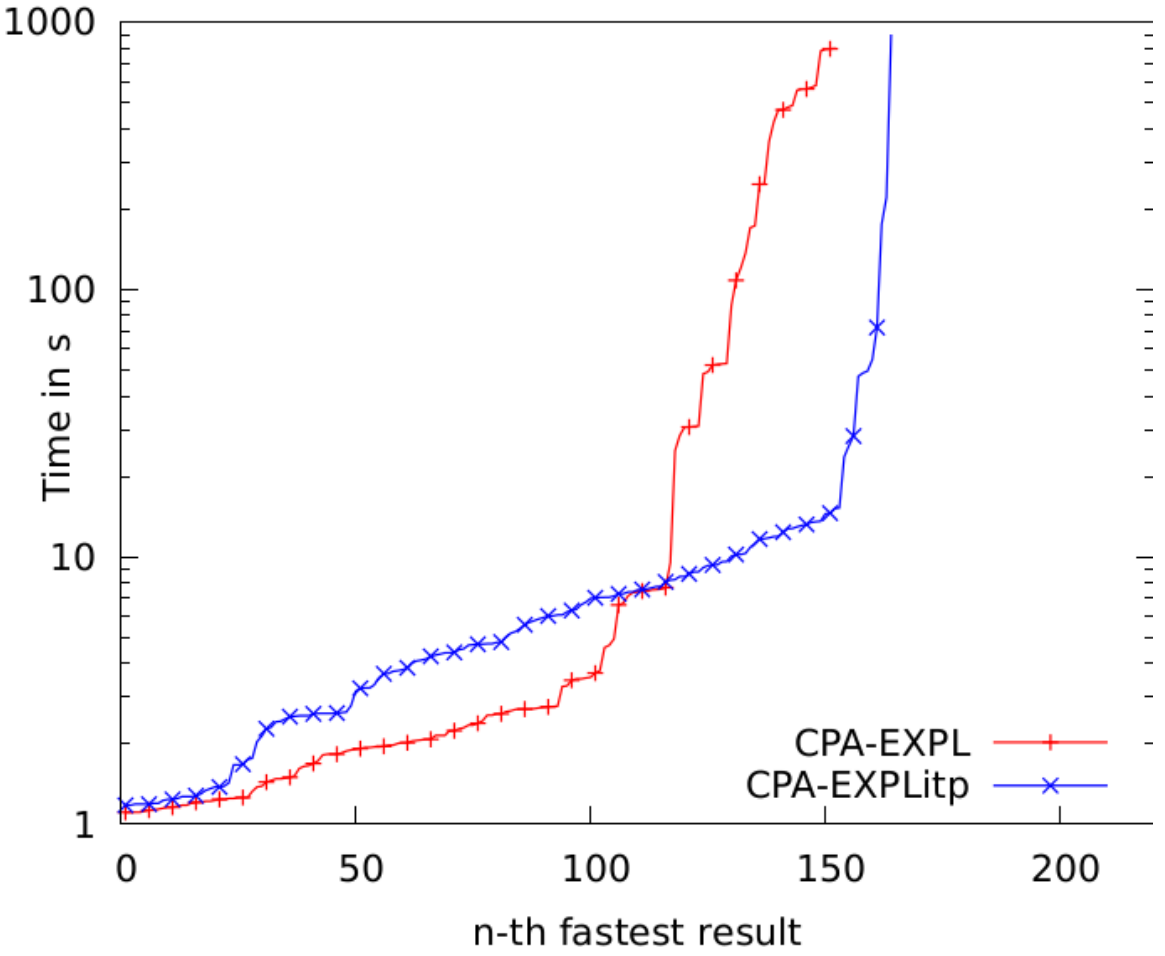
$$\Pi_2 = \Pi_1 \cup [N2 \rightarrow \{a\}]$$



Program proven safe

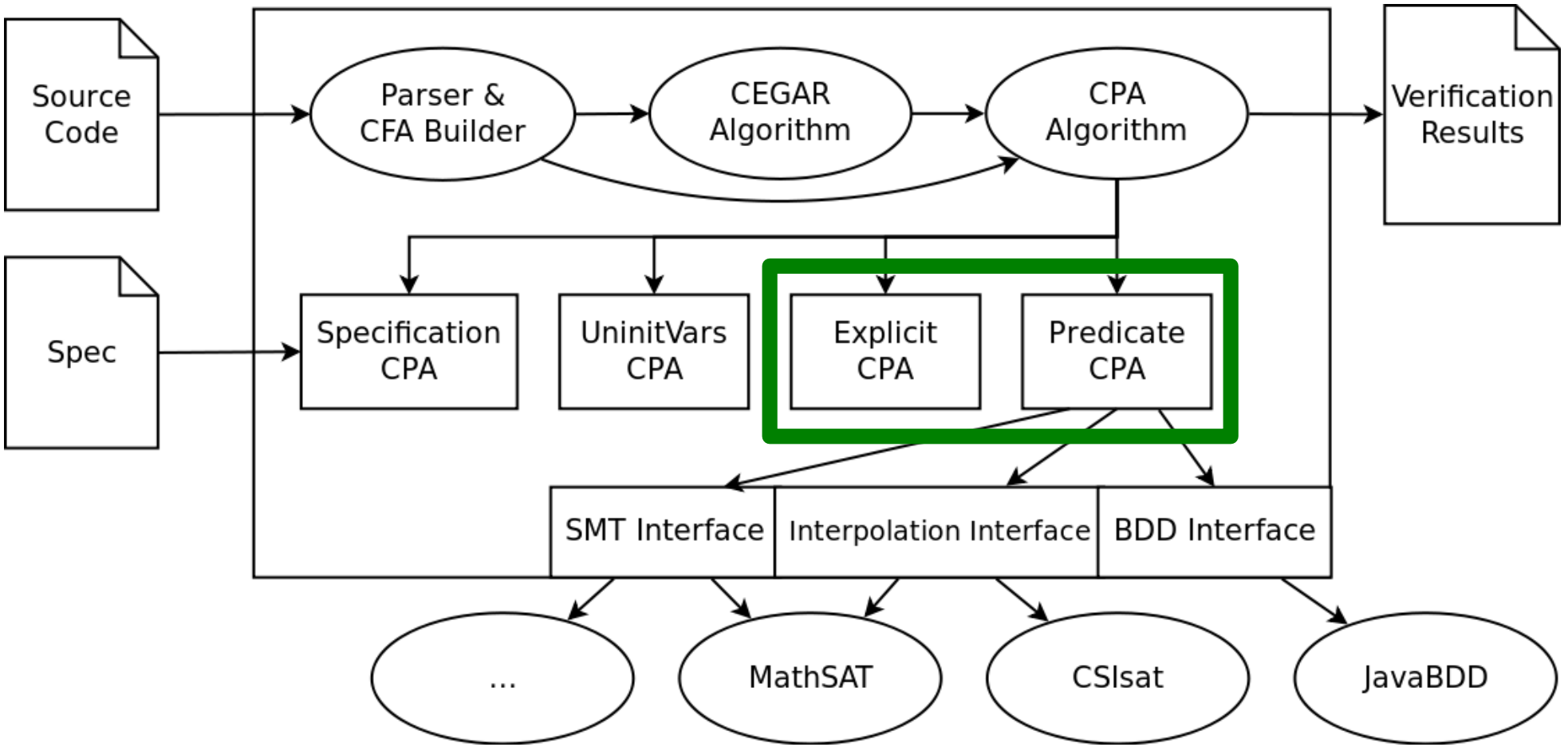
What do we have so far?

- ✔ Abstraction
- ✔ CEGAR
- ✔ Interpolation



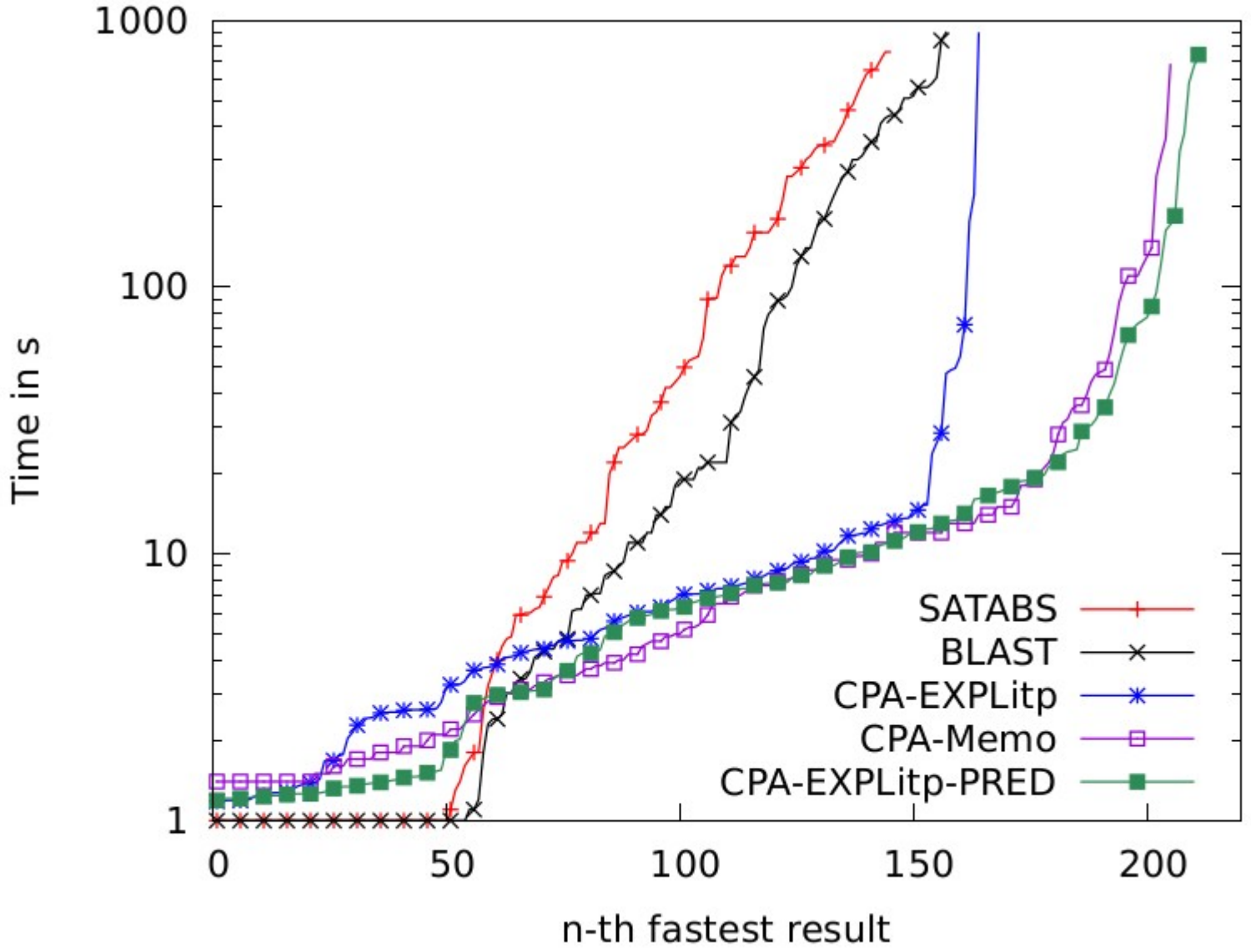
- ✗ Precise state representation
  - ✗ Inequalities [a != b]
  - ✗ Intervals [a < b]

CPAchecker: Architecture



- Add auxiliary predicate analysis
- Refinement of both domains based on (lack of) expressiveness
- Predicate analysis tracks only what is beyond explicit domain

Comparison with SV-COMP Run Times



Comparison with SV-COMP Scores

	Results taken from SV-COMP'12			Results from our experiments		
	BLAST 2.7	SATABS	CPA-Memo	CPA-Expl	CPA-Expl-Itp	CPA-Expl-Itp-Pred
ControlFlow	71	75	140	124	123	141
Drivers32	72	71	51	53	53	71
Drivers64	55	32	49	5	33	37
Heap	--	--	4	1	1	8
SystemC	33	57	36	34	34	61
Overall	231	236	**280**	217	**244**	**318**

Conclusion

- We defined abstraction, CEGAR and Craig interpolation for the explicit domain
- Results are very encouraging
  - Valid methods to lower size of reached set
  - ➢ Circumvent state-space explosion
- ExplicitCPA proofs to be competitive
- Especially in combination with auxiliary predicate analysis

Questions ?