Hardware Model Checking Competition 2020

(11th Edition)

Mathias Preiner Armin Biere Nils Froleyks

http://fmv.jku.at/hwmcc20

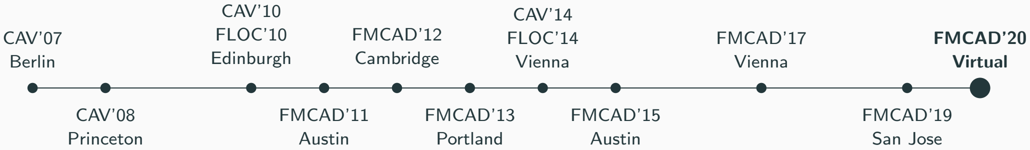
fmcad.20

September 21-24, 2020

Stanford University

JOHANNES KEPLER UNIVERSITÄT LINZ

HWMCC Editions



CAV'07 Berlin

CAV'08 Princeton

CAV'10 FLOC'10 Edinburgh

FMCAD'11 Austin

FMCAD'12 Cambridge

FMCAD'13 Portland

CAV'14 FLOC'14 Vienna

FMCAD'15 Austin

FMCAD'17 Vienna

FMCAD'19 San Jose

FMCAD'20 Virtual

Previous Years

- AIGER format (`http://fmv.jku.at/aiger`)
- Tracks
  - **SINGLE** safety (bad state) property track
  - how **DEEP** model checkers go on unsolved SINGLE instances (Oski Technology award $500)
  - **LIVENESS** track (single "justice" property)
- BTOR2 format (`https://github.com/boolector/btor2tools`)
  - **HWMCC'19** first year with **word-level** tracks

This Year

Goal: establish word-level track as part of HWMCC

- collect large set of publicly available word-level benchmarks
- encourage researchers to work on novel model checking engines
- provide a platform for comparison

Word-level Track(s)

- BTOR2 format (`https://github.com/Boolector/btor2tools`)
- **SINGLE** safety (bad state) property track
  - subtracks: bit-vectors, bit-vectors+arrays
  - BTOR2 witnesses optional
- Intel Xeon E5-2620 v4 2.10GHz, 16 cores (32 threads),
  Limits: 120 GB memory, 1h wall-clock time

Word-Level Format BTOR

BTOR 1.0 [BPR'08]

- word-level generalization of the initial AIGER format
- format for quantifier-free formulas over bit-vectors and arrays
- sequential extensions

BTOR 2.0 [CAV'18]

- lifts features from the AIGER 1.9 format to word-level
  - supports invariant and fairness constraints
  - supports safety and liveness properties
  - initialization of registers/memories
- witness format
- tool suite: libbtor2parser, btorsim, btor2aiger, btorsplit, ...

BTOR 2.0

BTOR2 Format

```
⟨num⟩        ::=  positive unsigned integer (greater than zero)
⟨uint⟩       ::=  unsigned integer (including zero)
⟨string⟩     ::=  sequence of whitespace and printable characters without '\n'
⟨symbol⟩     ::=  sequence of printable characters without '\n'
⟨comment⟩    ::=  ';' ⟨string⟩
⟨nid⟩        ::=  ⟨num⟩
⟨sid⟩        ::=  ⟨num⟩
⟨const⟩      ::=  'const' ⟨sid⟩ [0-1]+
⟨constd⟩     ::=  'constd' ⟨sid⟩ ['-']⟨uint⟩
⟨consth⟩     ::=  'consth' ⟨sid⟩ [0-9a-fA-F]+
⟨input⟩      ::=  ( 'input' | 'one' | 'ones' | 'zero' ) ⟨sid⟩ | ⟨const⟩ | ⟨constd⟩ | ⟨consth⟩
⟨state⟩      ::=  'state' ⟨sid⟩
⟨bitvec⟩     ::=  'bitvec' ⟨num⟩
⟨array⟩      ::=  'array' ⟨sid⟩ ⟨sid⟩
⟨node⟩       ::=  ⟨sid⟩ 'sort' ( ⟨array⟩ | ⟨bitvec⟩ )
                | ⟨nid⟩ ( ⟨input⟩ | ⟨state⟩ )
                | ⟨nid⟩ ⟨opidx⟩ ⟨sid⟩ ⟨nid⟩ ⟨uint⟩ [⟨uint⟩]
                | ⟨nid⟩ ⟨op⟩ ⟨sid⟩ ⟨nid⟩ [⟨nid⟩ [⟨nid⟩]]
                | ⟨nid⟩ ( 'init' | 'next' ) ⟨sid⟩ ⟨nid⟩ ⟨nid⟩
                | ⟨nid⟩ ( 'bad' | 'constraint' | 'fair' | 'output' ) ⟨nid⟩
                | ⟨nid⟩ 'justice' ⟨num⟩ ( ⟨nid⟩ )+
⟨line⟩       ::=  ⟨comment⟩ | ⟨node⟩ [ ⟨symbol⟩ ] [ ⟨comment⟩ ]
⟨btor⟩       ::=  ( ⟨line⟩'\n' )+
```

Witness Format

```
⟨binary-string⟩      ::=  [0-1]+
⟨bv-assignment⟩      ::=  ⟨binary-string⟩
⟨array-assignment⟩   ::=  '[' ⟨binary-string⟩ ']' ⟨binary-string⟩
⟨assignment⟩         ::=  ⟨uint⟩ ( ⟨bv-assignment⟩ | ⟨array-assignment⟩ ) [⟨symbol⟩]
⟨model⟩              ::=  ( ⟨comment⟩'\n' | ⟨assignment⟩'\n' )+
⟨state part⟩         ::=  '#' ⟨uint⟩ '\n' ⟨model⟩
⟨input part⟩         ::=  '@' ⟨uint⟩ '\n' ⟨model⟩
⟨frame⟩              ::=  [ ⟨state part⟩ ] ⟨input part⟩
⟨prop⟩               ::=  ( 'b' | 'j' )⟨uint⟩
⟨header⟩             ::=  'sat\n' ( ⟨prop⟩ )+ '\n'
⟨witness⟩            ::=  ( ⟨comment⟩'\n' )+ | ⟨header⟩ ( ⟨frame⟩ )+ '.'
```

https://github.com/Boolector/btor2tools

BTOR 2.0 Example

```
1 sort bitvec 1
2 sort bitvec 3
```

$$\left.\begin{array}{l}\texttt{3 zero 2}\\ \texttt{4 state 2 cnt}\\ \texttt{5 init 2 4 3}\end{array}\right\} cnt = 0$$

$$\left.\begin{array}{l}\texttt{6 input 2 in}\\ \texttt{7 add 2 4 6}\\ \texttt{8 next 2 4 7}\end{array}\right\} cnt' = cnt + in$$

$$\left.\begin{array}{l}\texttt{9 constd 2 7}\\ \texttt{10 eq 1 4 9}\\ \texttt{11 bad 10}\end{array}\right\} bad(cnt == 7)$$

$$\left.\begin{array}{l}\texttt{12 constd 2 3}\\ \texttt{13 ulte 1 6 12}\\ \texttt{14 constraint 13}\end{array}\right\} in \leq 3$$

```
sat
b0
#0
@0
0 011 in@0
@1
0 010 in@1
@2
0 010 in@2
@3
0 000 in@3
.
```

Benchmarks

Submissions

- **10** new benchmarks (4 bit-vector, 6 bit-vector+arrays) with **36** safety properties submitted by Makai Mann
- **30** new bit-level benchmarks submitted by Gianpiero Cabodi, not considered for this year since only bit-level

Bit-blasting BTOR2 to AIGER (btor2aiger)

- bit-blasted all bit-vector benchmarks to AIGER
- no array support yet
- uses Boolector to synthesize AIGs
- uses AIGER library for constructing AIGER benchmarks

Benchmark Selection

- selected from **2319** BV and **2518** BV+arrays SINGLE benchmarks
- divided all benchmarks into 30 classes
- removed "easy" benchmarks (800 bit-vector, 817 array)
  solved by all model HWMCC'19 checkers[1] within 10s wall-clock time
- randomly selected from remaining benchmarks
  - □ selected $N * \mathit{frac}(N)$ benchmarks per class
    N . . . number of benchmarks in class
  - □ BEEM benchmarks limited to 15 benchmarks

$$\mathit{frac}(N) = \begin{cases} 1/2 & \text{if } N < 50 \\ 1/3 & \text{if } N < 100 \\ 1/4 & \text{if } N < 200 \\ 1/5 & \text{if } N < 300 \\ 1/6 & \text{else} \end{cases}$$

- in total **324** bit-vector and **315** bit-vector+array benchmarks

[1] excluding BMC-only model checkers for unsat

Benchmark Selection: Bit-Vectors

class	selected	unused	removed	total
2019/wolf/2019C/qspiflash	73	361	45	479
2019/goel/industry/cal	44	131	58	233
2019/mann/data-integrity/unsafe/arbitrated_top	27	54	19	100
2019/wolf/2018D/zipcpu	24	24	98	146
2019/wolf/2019A/picorv32	18	36	0	54
2019/wolf/2019C/dspfilters_fastfir_second	17	17	18	52
2019/beem	15	509	156	680
2020/mann	15	15	0	30
2019/wolf/2019C/vgasim	15	14	93	122
2019/mann/data-integrity/unsafe/shift_register_top	12	12	1	25
2019/goel/opensource	12	11	117	140
2019/mann/data-integrity/unsafe/circular_pointer_top	11	11	3	25
2019/goel/industry/gen	9	9	106	124
2019/wolf/2018D/picorv32	8	8	10	26
2019/wolf/2019C/zipversa_composecrc_prf	8	8	9	25
2019/goel/industry/mul	5	5	1	11
2019/wolf/2019B/marlann	3	3	3	9
2019/wolf/2018D/VexRiscv	3	3	0	6
2019/goel/crafted	1	1	22	24
2019/mann/unsafe	1	1	1	3
2019/wolf/2018D/ponylink	1	1	0	2
2019/mann/safe	1	1	0	2
2019/mann/unknown	1	0	0	1

30% overlap with HWMCC'19 benchmarks (98 in total)

Benchmark Selection: Bit-Vectors+Arrays

class	**selected**	**unused**	**removed**	**total**
2019/wolf/2019C/dblclockfft_butterfly	129	643	427	1199
2019/wolf/2019C/zipcpu_zipcpu_piped	74	370	86	530
2019/wolf/2019C/zipcpu_zipcpu_dcache	63	311	253	627
2019/wolf/2019A/picorv32	18	36	0	54
2019/wolf/2018A/zipcpu	9	8	40	57
2019/wolf/2018A/picorv32	7	6	10	23
2019/wolf/2019B/marlann	5	4	0	9
2019/wolf/2018A/VexRiscv	3	3	0	6
2020/mann	3	2	0	5
2019/mann/unsafe	1	1	1	3
2019/wolf/2018A/ponylink	1	1	0	2
2019/mann/safe	1	1	0	2
2019/mann/unknown	1	0	0	1

19% overlap with HWMCC'19 benchmarks (58 in total)

AVR: Abstractly Verifying Reachability

https://github.com/aman-goel/avr

Aman Goel, Karem Sakallah (Univ. of Michigan)

- AVR proof race: 16 parallel configurations racing proof/counterexample
  - □ 11 variants of IC3+SA
    word-level IC3 with syntax-guided abstraction, plus add-ons:
    - data abstraction
    - incremental refinement
    - interpolation
    - property-directed word splitting
    - extract/concat handler
    - hybrid abstractions
  - □ 2 variants of BMC, plus data abstraction **(new)**
  - □ 3 variants of K-induction **(new)**
- Support for Arrays **(new)**
- Arrays + data / hybrid abstractions **(new)**
- Supports Verilog, VMT and BTOR2 frontends
- Inductive invariants (SMT-LIB), counterexample traces (BTOR2)

Thanks to Yices 2, Boolector, MathSAT 5, Yosys, Btor2Tools and Cadence JasperGold teams

[TACAS'20] AVR at a high level, [NFM'19] Syntax-guided abstraction technique of AVR
[SOSP'19] Using AVR for verifying distributed protocols

AVY

Yakir Vizel (Technion), Arie Gurfinkel (Univ. of Waterloo)

- Based on a model checking algorithm that combines interpolation and PDR
- Interpolants are extracted from BMC queries
- PDR is used to generalize the interpolants
- Original paper CAV 2014
- Other improvements: FMCAD 2014, CAV 2015
- The latest version includes interpolants that are extracted from k-induction proofs
- K-AVY paper appeared in CAV 2019
- The tool executes AVY and k-AVY in various configurations, BMC and PDR
- Implementation uses ABCs infrastructure, MiniSAT, Glucose and Muser

nuXmv

https://nuxmv.fbk.eu

Alberto Griggio, Martin Jonas (Fondazione Bruno Kessler)

- Portfolio approach (*n* engines in parallel, no communication)
- **BV category**:
  - SAT-based IC3 and BMC
  - SMT-based IC3 with implicit abstraction
- **BV+Arrays category**:
  - SMT-based IC3 with implicit abstraction, k-induction, BMC
- **SAT solver**: CaDiCaL
- **SMT solver**: MathSAT

Pono

https://github.com/upscale-project/pono

Makai Mann, Ahmed Irfan, Florian Lonsing, Yahan Yang, Clark Barrett (Stanford Univ.)

- Lightweight, adaptable SMT-based model checker
  - Built on solver-agnostic SMT API, smt-switch
- Competition portfolio configuration
  - BMC
  - K-Induction
  - Interpolation-based
  - Model-based IC3 (BV only)
  - IC3 with Interpolant Generalization (BV only)
  - Counterexample-Guided Prophecy built on Interpolation-based MC (Arrays only)
- SMT Solvers
  - Boolector, MathSAT5 and CVC4
  - Many thanks to the SMT solver developers!

Further submissions

Competitive Bit-Level Model Checkers

- **ABC**
  - Robert K. Brayton, Baruch Sterin, Alan Mishchenko (UC Berkeley)
- **pdtrav**
  - Gianpiero Cabodi et.al. (Politecnico di Torino)

Non-Competitive Model Checkers (submitted by organizers)

- **BtorMC**
  - Aina Niemetz, Mathias Preiner, Armin Biere (Stanford, JKU)
  - compiled from Boolector repository @ 95859db with CaDiCaL
- **CoSA2**
  - Makai Mann, Ahmed Irfan, Florian Lonsing, Clark Barrett (Stanford Univ.)
- **camical**[2]: BMC for sanity checking
- **ABC17**[2]: HWMCC'17 winner, similar performance to ABC this year
- **nmtip**[2]: withdrawn due to some issues in testing phase

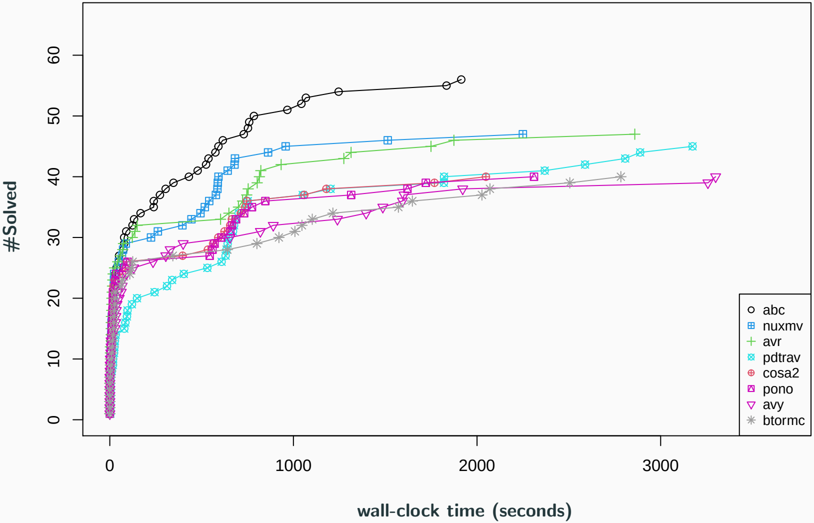
[2] Not shown in results but log files provided

Ranking

- **3 categories**
  - □ bit-vectors
  - □ bit-vectors+arrays
  - □ combined
- each category divided into
  - □ sat
  - □ unsat
  - □ all
- **9 rankings** each with 1 gold, 1 silver, 1 bronze (27 “medals” in total)

Results

Bit-Vectors (sat)

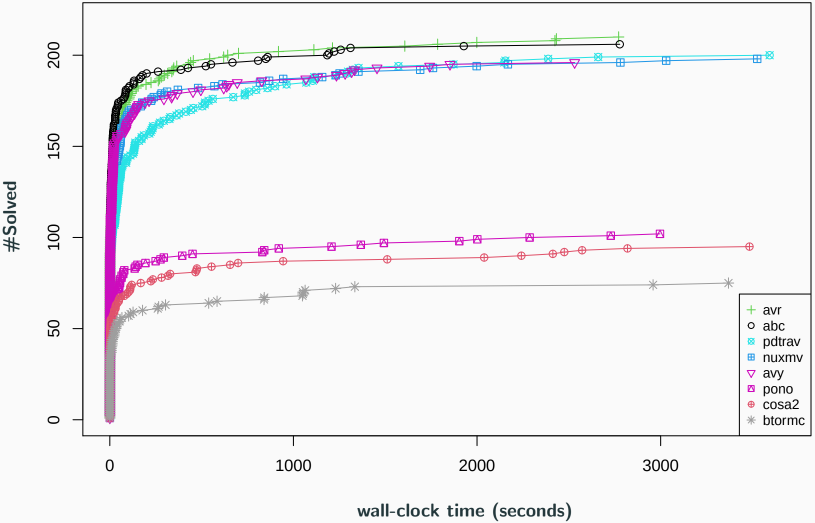


Bit-Vectors (sat)

			total	slvd	sat	to	unk	real	time	best	uniq
1	**abc**	**●**	**64**	**56**	**56**	**8**	**0**	**17368**	**200348**	**20**	**6**
2	nuxmv		64	47	47	17	0	13340	53240	9	0
3	avr		64	47	47	0	17	17452	251887	2	1
	pdtrav	●	64	45	45	16	3	29746	166427	6	3
	cosa2		64	40	40	24	0	12702	51132	9	0
	pono		64	40	40	24	0	14025	70219	2	0
	avy	●	64	40	40	24	0	20110	199809	10	0
	btormc		64	40	40	24	0	20415	20412	6	0

● . . . run on bit-blasted AIGER benchmark

Bit-Vectors (unsat)

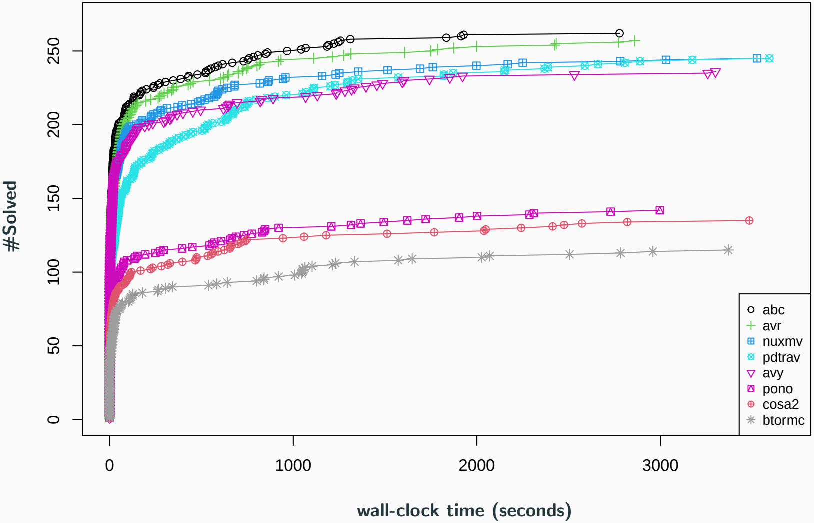


Bit-Vectors (unsat)

			total	slvd	unsat	to	mo	unk	real	time	best	uniq
1	**avr**		**225**	**210**	**210**	**0**	**0**	**15**	**26893**	**409198**	**28**	**10**
2	abc	•	225	206	206	19	0	0	19137	228731	67	0
3	pdtrav	•	225	200	200	25	0	0	45145	263503	8	1
	nuxmv		225	198	198	27	0	0	31397	125205	21	2
	avy	•	225	196	196	29	0	0	23945	237719	44	0
	pono		225	102	102	106	16	1	21918	103916	9	0
	cosa2		225	95	95	130	0	0	26547	99951	3	0
	btormc		225	75	75	149	0	1	17797	17791	45	1

• . . . run on bit-blasted AIGER benchmark

Bit-Vectors (all)

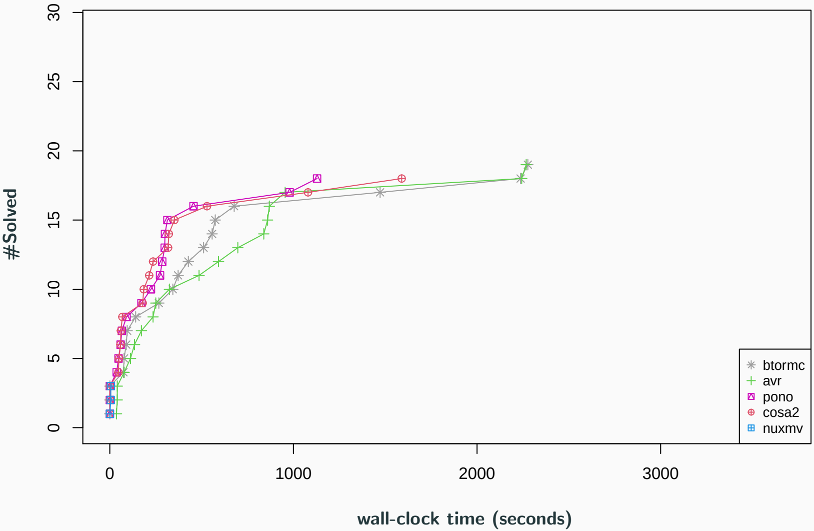


Bit-Vectors (all)

			total	slvd	sat	unsat	to	mo	unk	real	time	best	uniq
1	**abc**	●	**324**	**262**	**56**	**206**	**62**	**0**	**0**	**36505**	**429078**	**87**	**6**
2	avr		324	257	47	210	0	0	67	44345	661085	30	11
3	nuxmv		324	245	47	198	79	0	0	44737	178444	30	2
	pdtrav	●	324	245	45	200	76	0	3	74891	429930	14	4
	avy	●	324	236	40	196	88	0	0	44055	437528	54	0
	pono		324	142	40	102	165	16	1	35943	174134	11	0
	cosa2		324	135	40	95	189	0	0	39249	151083	12	0
	btormc		324	115	40	75	208	0	1	38212	38203	51	1

● . . . run on bit-blasted AIGER benchmark

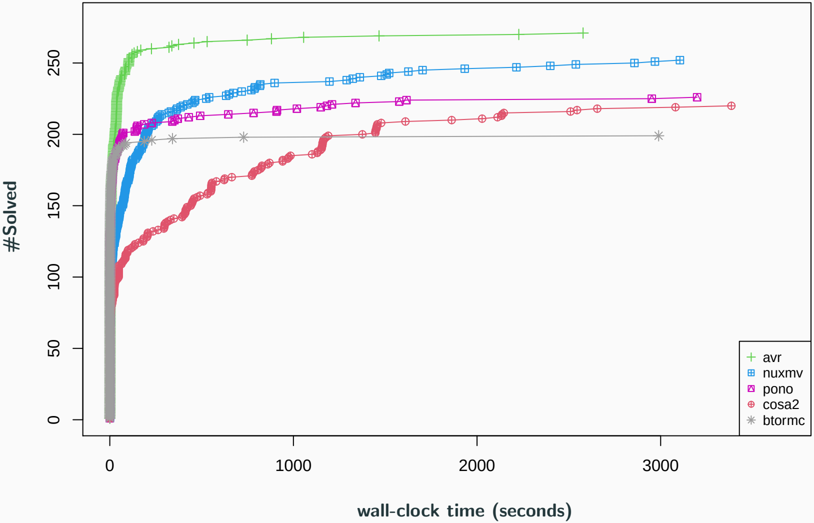
Bit-Vectors+Arrays (sat)



Bit-Vectors+Arrays (sat)

		total	slvd	sat	to	unk	real	time	best
	btormc	19	19	19	0	0	10193	10192	0
1	**avr**	**19**	**19**	**19**	**0**	**0**	**11232**	**157865**	**1**
2	pono	19	18	18	0	1	4733	19048	10
	cosa2	19	18	18	1	0	5287	21277	8
3	nuxmv	19	3	3	16	0	11	30	0

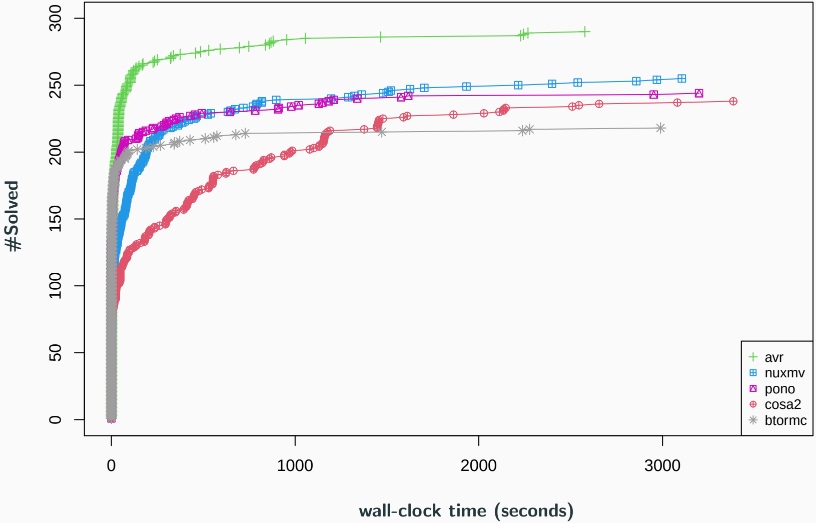
Bit-Vectors+Arrays (unsat)



Bit-Vectors+Arrays (unsat)

		total	slvd	unsat	to	mo	unk	real	time	best	uniq
1	**avr**	**274**	**271**	**271**	**0**	**0**	**3**	**15878**	**230699**	**75**	**12**
2	nuxmv	274	252	252	22	0	0	54916	161218	2	0
3	pono	274	226	226	38	2	8	23112	95755	23	1
	cosa2	274	220	220	43	1	10	91618	365311	1	0
	btormc	274	199	199	75	0	0	5417	5403	173	1

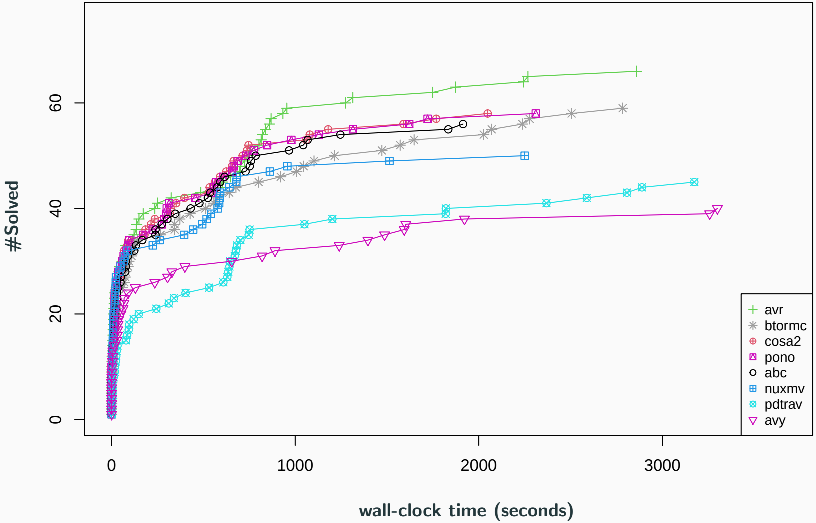
Bit-Vectors+Arrays (all)



Bit-Vectors+Arrays (all)

		total	slvd	sat	unsat	to	mo	unk	real	time	best	uniq
1	**avr**	**315**	**290**	**19**	**271**	**0**	**0**	**25**	**27110**	**388564**	**76**	**12**
2	nuxmv	315	255	3	252	60	0	0	54927	161248	2	0
3	pono	315	244	18	226	54	6	11	27845	114803	33	1
	cosa2	315	238	18	220	63	1	13	96905	386589	9	0
	btormc	315	218	19	199	97	0	0	15611	15595	173	1

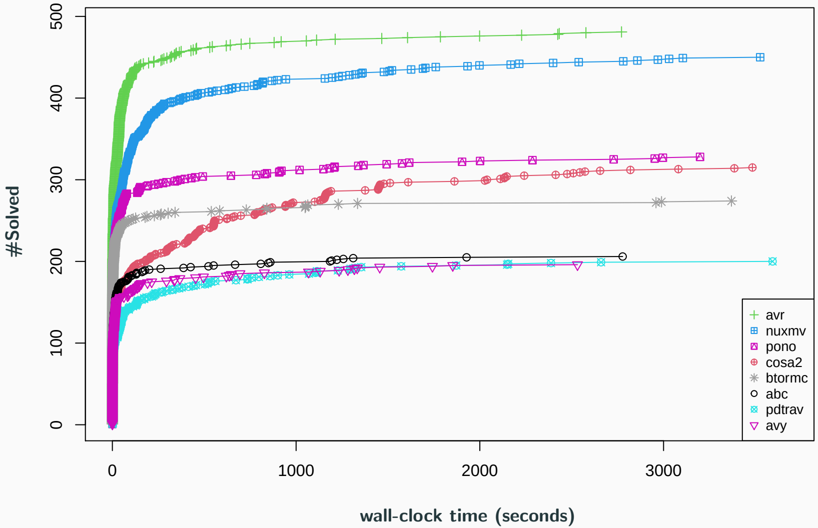
Combined (sat)



Combined (sat)

			total	slvd	sat	to	unk	real	time	best	uniq
1	**avr**		**83**	**66**	**66**	**0**	**17**	**28685**	**409752**	**3**	**1**
	btormc		83	59	59	24	0	30608	30603	6	0
	cosa2		83	58	58	25	0	17989	72410	17	0
2	pono		83	58	58	24	1	18759	89266	12	0
3	abc	●	64	56	56	8	0	17368	200348	20	6
	nuxmv		83	50	50	33	0	13351	53270	9	0
	pdtrav	●	64	45	45	16	3	29746	166427	6	3
	avy	●	64	40	40	24	0	20110	199809	10	0

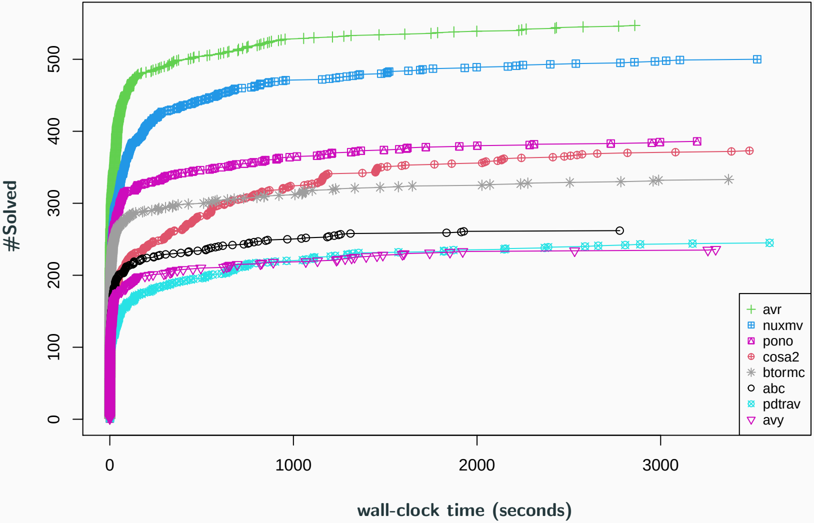
Combined (unsat)



Combined (unsat)

			total	slvd	unsat	to	mo	unk	real	time	best	uniq
1	**avr**		**499**	**481**	**481**	**0**	**0**	**18**	**42771**	**639897**	**103**	**22**
2	nuxmv		499	450	450	49	0	0	86313	286423	23	2
3	pono		499	328	328	144	18	9	45030	199671	32	1
	cosa2		499	315	315	173	1	10	118164	465262	4	0
	btormc		499	274	274	224	0	1	23214	23194	218	2
	abc	●	225	206	206	19	0	0	19137	228731	67	0
	pdtrav	●	225	200	200	25	0	0	45145	263503	8	1
	avy	●	225	196	196	29	0	0	23945	237719	44	0

Combined (all)



Combined (all)

			total	slvd	sat	unsat	to	mo	unk	real	time	best	uniq
1	**avr**		**639**	**547**	**66**	**481**	**0**	**0**	**92**	**71456**	**1049649**	**106**	**23**
2	nuxmv		639	500	50	450	139	0	0	99664	339693	32	2
3	pono		639	386	58	328	219	22	12	63789	288937	44	1
	cosa2		639	373	58	315	252	1	13	136153	537672	21	0
	btormc		639	333	59	274	305	0	1	53822	53797	224	2
	abc	●	324	262	56	206	62	0	0	36505	429078	87	6
	pdtrav	●	324	245	45	200	76	0	3	74891	429930	14	4
	avy	●	324	236	40	196	88	0	0	44055	437528	54	0

Results Summary

	gold	silver	bronze
avr	7	1	1
abc	2	1	1
nuxmv		5	2
pono		2	4
pdtrav			1

Congratulations to the winners!

Conclusion

Submissions

- 3 word-level and 3 bit-level model checkers
- only 10 new word-level benchmarks with 36 safety properties

Next Edition

- BTOR2 witnesses required
- (maybe) bit-blasting of array benchmarks

Thanks to all submitters!

References i

- Robert Brummayer and Armin Biere and Florian Lonsing BTOR: Bit-Precise Modelling of Word-Level Problems for Model Checking. Workshop on Bit-Precise Reasoning, 2018
- Aina Niemetz and Mathias Preiner and Clifford Wolf and Armin Biere Btor2 , BtorMC and Boolector 3.0. CAV, Pages 587–595, 2018
- Armin Biere and Keijo Heljanko and Siert Wieringa AIGER 1.9 and Beyond. FMV Technical Report 11/2, 2011