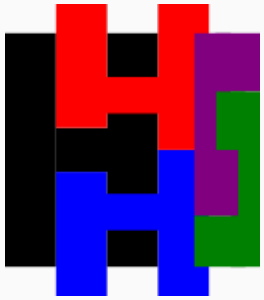


Hardware Model Checking Competition 2019

(10th Edition)

Mathias Preiner Armin Biere

http://fmv.jku.at/hwmcc19

Sponsored by Oski TECHNOLOGY

fmcad.19

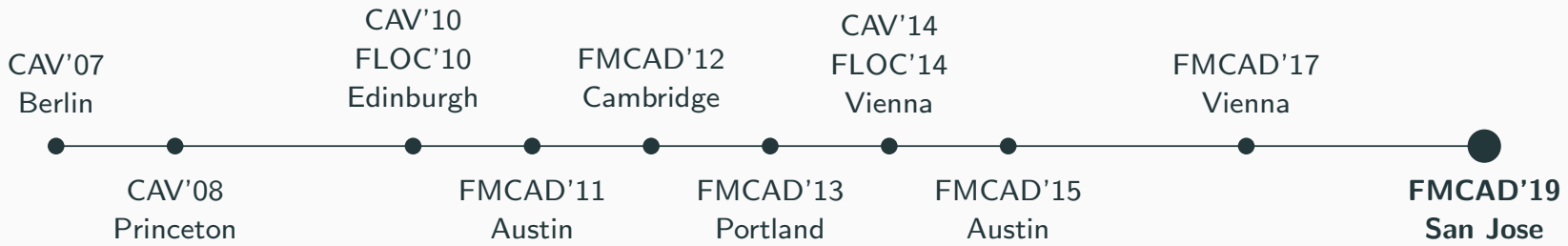
October 22-25, 2019

San Jose, California, USA

Stanford University

JKU JOHANNES KEPLER UNIVERSITÄT LINZ

HWMCC Editions



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)

Word-level model checking mentioned since 2012 in every HWMCC presentation.

This Year

Goal: bootstrap 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
- Oski Technology award $1000
- 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
3 zero 2
4 state 2 cnt
5 init 2 4 3
6 input 2 in
7 add 2 4 6
8 next 2 4 7
9 ones 2
10 eq 1 4 9
11 bad 10
12 constd 2 3
13 ulte 1 6 12
14 constraint 13
```

$cnt = 0$

$cnt' = cnt + in$

$bad(cnt == 7)$

$in \leq 3$

```
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

- **745** new benchmarks with **4177** safety properties
  from Aman Goel (534/537), Makai Mann (162/162), and Clifford Wolf (49/3478)
- 688 BEEM benchmarks translated from BTOR1

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

In total **2352** bit-vector and **2513** bit-vector+array SINGLE benchmarks

Benchmark Selection

- classified all benchmarks into 29 classes
- removed "easy" benchmarks (800 bit-vector, 817 array) solved by all model checkers[1] within 10s wall-clock time
- randomly selected from remaining benchmarks
  - selected $N * frac(N)$ benchmarks per class
    N . . . number of benchmarks in class
  - BEEM benchmarks limited to 15 benchmarks

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

- in total **317** bit-vector and **312** bit-vector+array benchmarks

[1] excluding BMC-only model checkers for unsat

Benchmark Selection: Bit-Vectors

class	selected	unused	removed	total
wolf/2019C/qspiflash	73	361	45	479
goel/industry/cal	44	131	60	235
mann/data-integrity/unsafe/arbitrated_top	27	54	19	100
wolf/2018D/zipcpu	24	24	98	146
wolf/2019A/picorv32	18	36	0	54
wolf/2019C/dspfilters_fastfir_second	17	17	18	52
beem	15	515	158	688
wolf/2019C/vgasim	15	14	93	122
goel/opensource	12	12	119	143
mann/data-integrity/unsafe/shift_register_top	12	12	1	25
mann/data-integrity/unsafe/circular_pointer_top	11	11	3	25
goel/industry/gen	9	9	106	124
wolf/2018A/zipcpu	8	8	34	50
wolf/2018D/picorv32	8	8	10	26
wolf/2019C/zipversa_composecrc_prf	8	8	9	25
goel/industry/mul	5	5	1	11
wolf/2019B/marlann	3	3	3	9
wolf/2018D/VexRiscv	3	3	0	6
goel/crafted	1	1	22	24
mann/unsafe	1	1	1	3
wolf/2018D/ponylink	1	1	0	2
mann/safe	1	1	0	2
mann/unknown	1	0	0	1

Benchmark Selection: Bit-Vectors+Arrays

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

AVR: Abstractly Verifying Reachability

- Aman Goel, Karem Sakallah (University of Michigan)
- AVR proof race: 11 parallel configurations racing for the result
  - 8 variants of IC3+SA
    word-level IC3 using syntax-guided abstraction with add-ons:
    - data abstraction
    - incremental refinement
    - interpolation
    - property-directed word splitting
    - extract/concat handler
    - hybrid abstractions
  - 3 variants of BMC (simple, incremental BMC)
- proof certificates (SMT-LIB), counterexample traces (BTOR2)
- SMT solvers under the hood:
  - Yices 2 (uninterpreted functions)
  - Boolector (with Lingeling/CaDiCaL) (bit-vectors)
  - MathSAT 5 (interpolation support)

github.com/aman-goel/avr

Thanks Yices 2, Boolector, MathSAT 5, Z3, Yosys, Btor2Tools developers

CoNPS-btormc-THP

- Norbert Manthey (hobbyist, former postdoc @ TU Dresden)
- based on BtorMC, no source modification
- uses Lingeling as SAT backend for Boolector
- ConNPS-btormc-no-THP as reference without modifications
- ConNPS-btormc-THP uses huge pages for mapping memory
  - idea based on work from 2009
  - implemented using transparent huge pages
  - statically linked against modified glibc library
  - patches and paper to be published
- single BMC engine

CoSA2: CoreIR Symbolic Analyzer 2

- Makai Mann, Ahmed Irfan, Florian Lonsing, Clark Barrett (Stanford University)
- SMT-based hardware model checker built on solver-agnostic framework smt-switch (`github.com/makaimann/smt-switch`)
- runs 4 engines in parallel:
  - BMC
  - BMC + simple path
  - k-induction
  - interpolation-based
- used SMT solvers:
  - MathSAT 5 (interpolation)
  - Boolector+CaDiCaL (for all other configurations)

Non-Competitive Model Checkers (Submitted by Organizers)

BtorMC

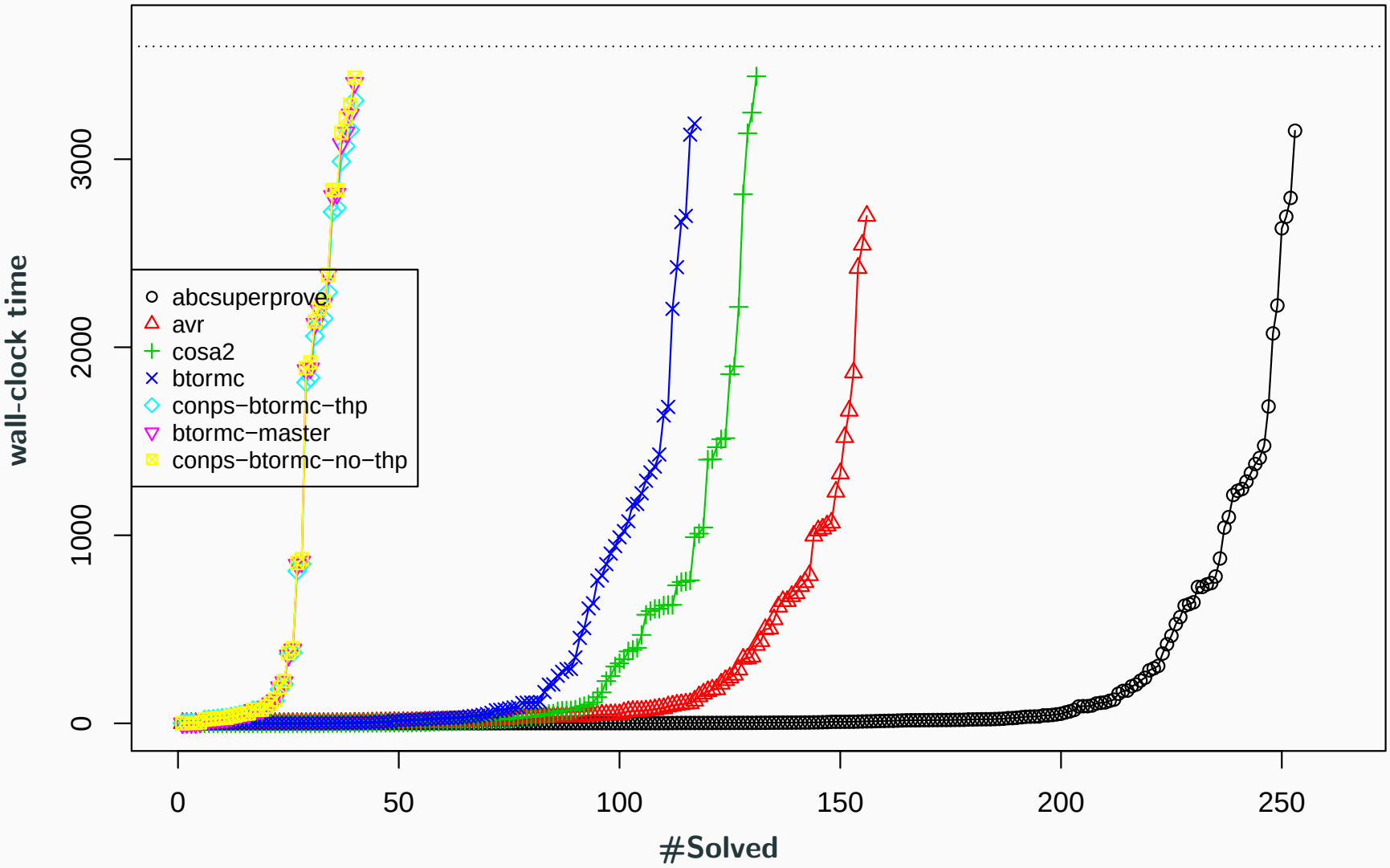
- Aina Niemetz, Mathias Preiner, Armin Biere (Stanford, JKU)
- btormc
  - based on SMT-COMP'19 branch of Boolector+CaDiCaL
  - single k-induction engine (new)
- btormc-master
  - based on master branch of Boolector+Lingeling
  - single BMC engine

abcsuperprove

- Robert K. Brayton, Baruch Sterin, Alan Mishchenko (Berkeley)
- winner of HWMCC'17 SINGLE track

Results

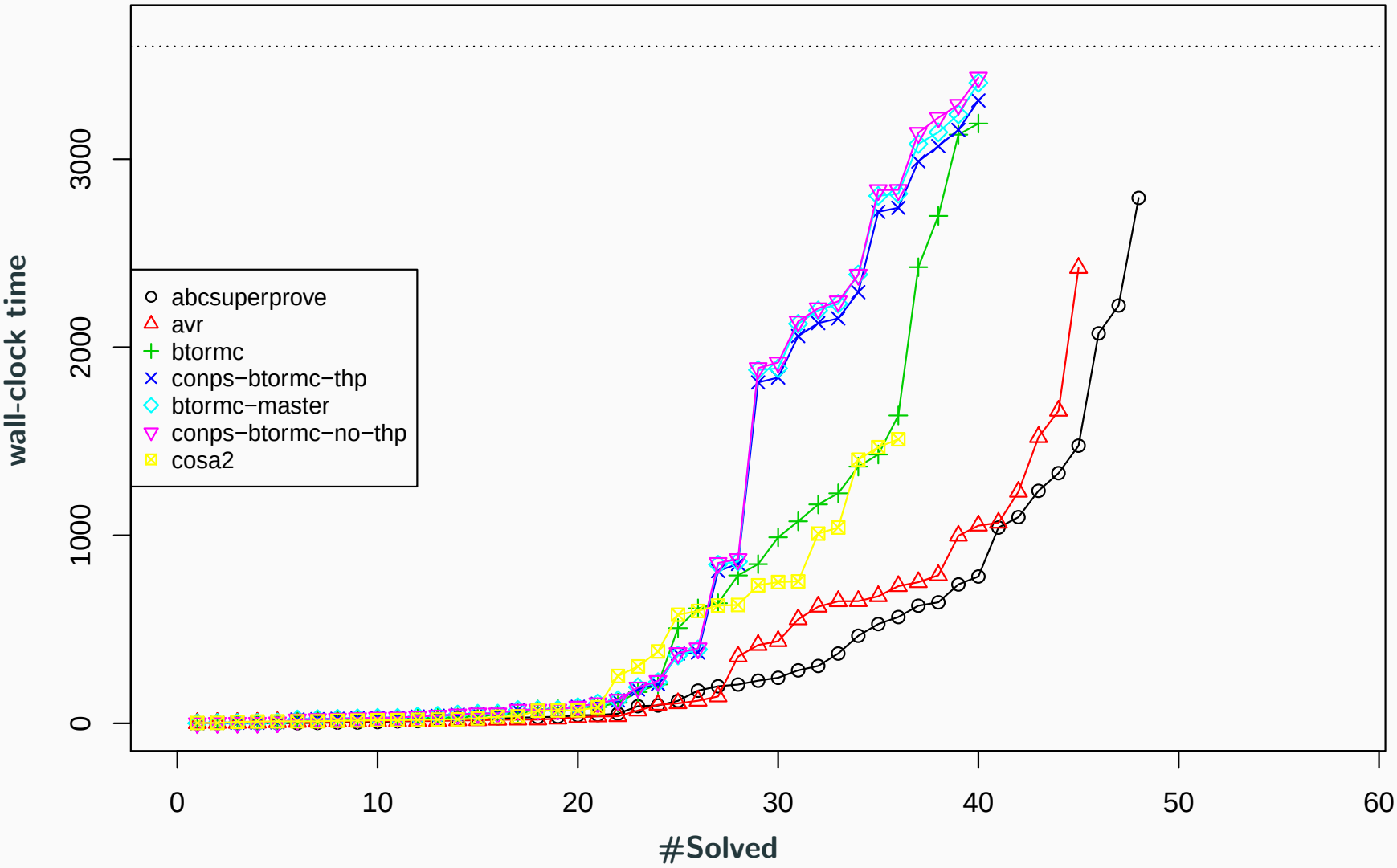
SINGLE: Bit-Vectors



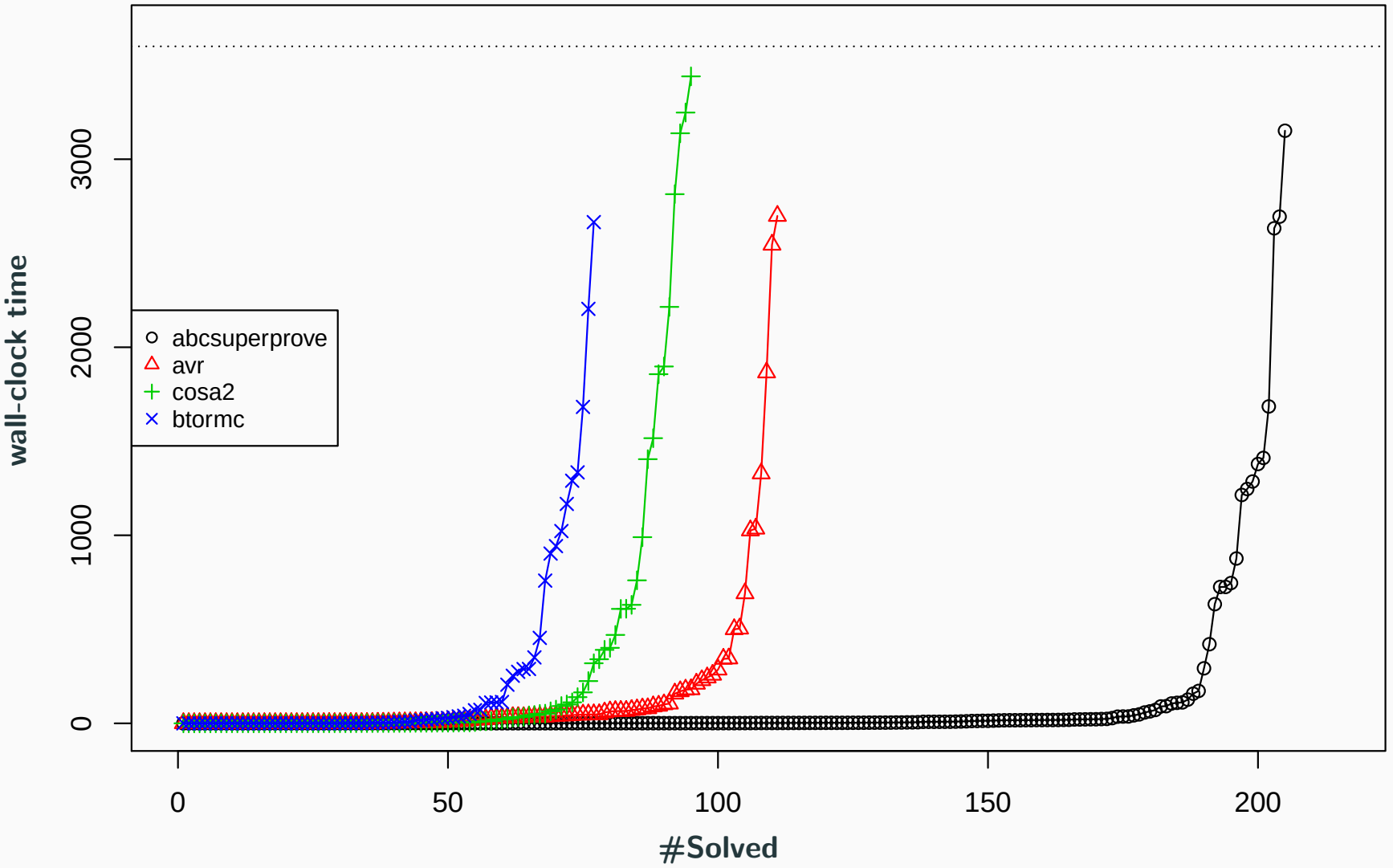
SINGLE: Bit-Vectors

		solved	sat	uns	to	mo	unk	real	time	space	best	uniq
	abcsuperprove	253	48	205	64	0	0	43520	438180	89594	160	73
1	**avr**	**156**	**45**	**111**	**0**	**0**	**161**	**34227**	**92687**	**106867**	**45**	**8**
2	cosa2	131	36	95	185	1	0	41083	91351	75021	13	0
	btormc	117	40	77	200	0	0	41915	41907	33263	50	1
3	conps-btormc-thp	40	40	0	175	0	102	33984	33981	8764	0	0
	btormc-master	40	40	0	176	0	101	34993	34990	7768	1	0
	conps-btormc-no-thp	40	40	0	176	0	101	35393	35389	8660	0	0

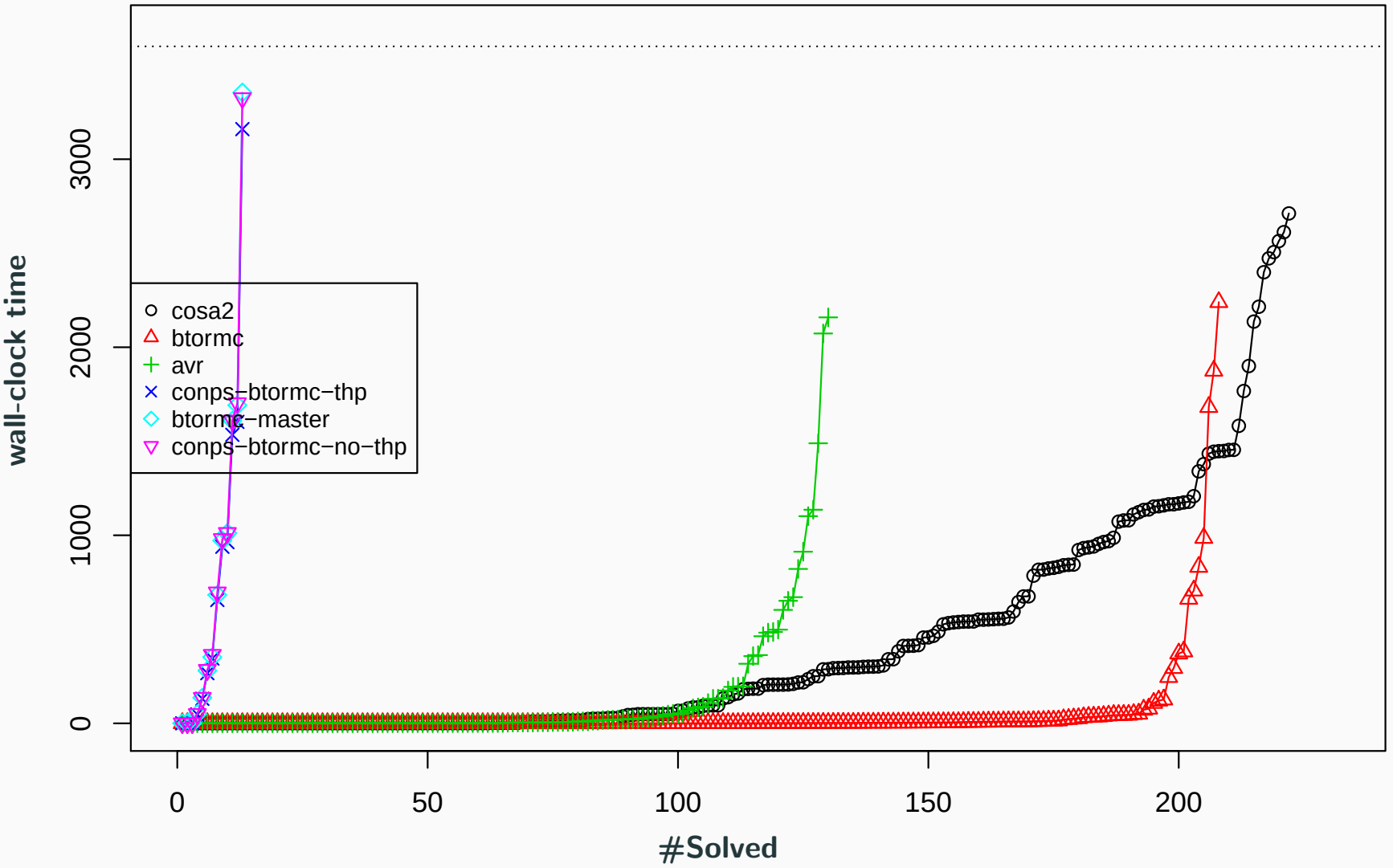
SINGLE SAT: Bit-Vectors



SINGLE UNSAT: Bit-Vectors



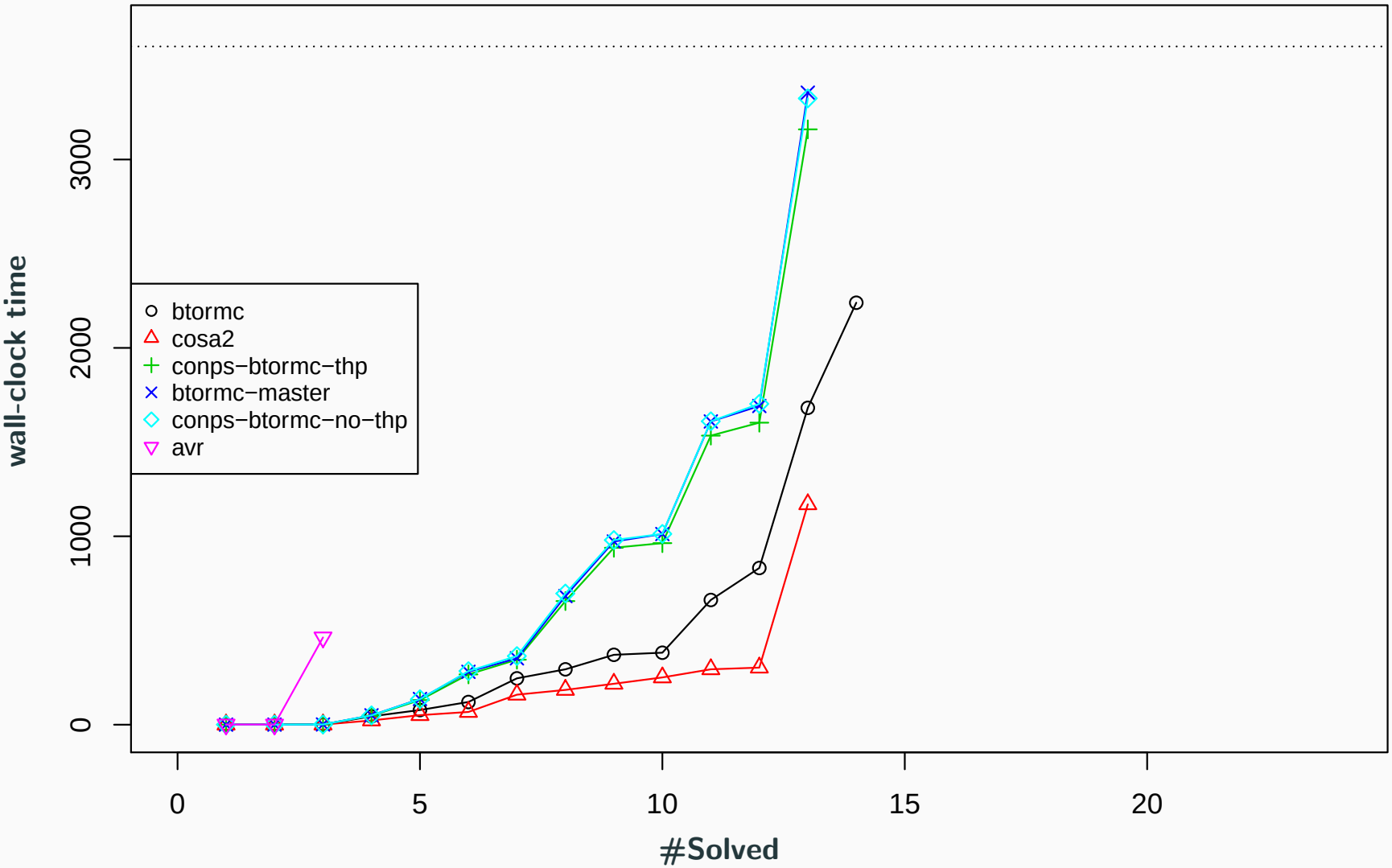
SINGLE: Bit-Vectors+Arrays



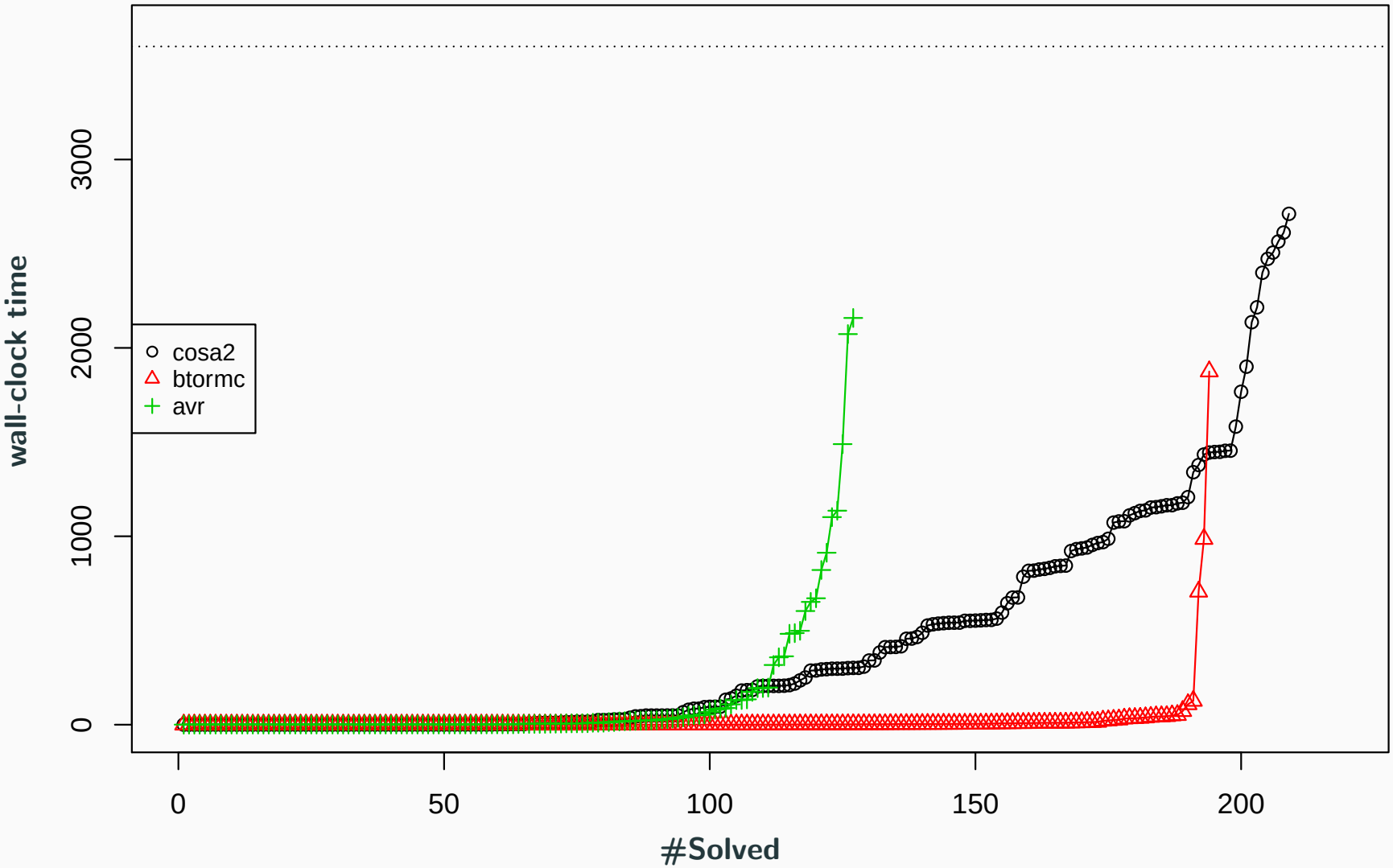
SINGLE: Bit-Vectors+Arrays

		solved	sat	uns	to	mo	unk	real	time	space	best	uniq
1	**cosa2**	**222**	**13**	**209**	**75**	**1**	**14**	**93852**	**198589**	**375490**	**22**	**11**
	btormc	208	14	194	104	0	0	11723	11710	14494	191	3
2	avr	130	3	127	0	0	182	16901	49138	302430	30	17
3	conps-btormc-thp	13	13	0	121	0	178	9648	9647	2683	0	0
	btormc-master	13	13	0	122	0	177	10143	10142	2278	0	0
	conps-btormc-no-thp	13	13	0	122	0	177	10162	10161	2644	0	0

SINGLE SAT: Bit-Vectors+Arrays



SINGLE UNSAT: Bit-Vectors+Arrays



Oski Award for Overall Winner

		solved	sat	uns	to	mo	unk	real	time	space	max	best	uniq
1	**cosa2**	**353**	**49**	**304**	**260**	**2**	**14**	**134935**	**289939**	**450511**	**18275**	**35**	**11**
	btormc	325	54	271	304	0	0	53637	53616	47757	3648	241	4
2	avr	286	48	238	0	0	343	51128	141825	409297	37372	75	25
	abcsuperprove	253	48	205	64	0	0	43520	438180	89594	24444	160	73
3	conps-btormc-thp	53	53	0	296	0	280	43632	43627	11447	2313	0	0
	btormc-master	53	53	0	298	0	278	45136	45131	10046	2102	1	0
	conps-btormc-no-thp	53	53	0	298	0	278	45555	45550	11304	2395	0	0

Note: abcsuperprove only participated in the bit-vector track.

Oski Award for Overall Winner

		solved	sat	uns	to	mo	unk	real	time	space	max	best	uniq
1	**cosa2**	**353**	**49**	**304**	**260**	**2**	**14**	**134935**	**289939**	**450511**	**18275**	**35**	**11**
	btormc	325	54	271	304	0	0	53637	53616	47757	3648	241	4
2	avr	286	48	238	0	0	343	51128	141825	409297	37372	75	25
	abcsuperprove	253	48	205	64	0	0	43520	438180	89594	24444	160	73
3	conps-btormc-thp	53	53	0	296	0	280	43632	43627	11447	2313	0	0
	btormc-master	53	53	0	298	0	278	45136	45131	10046	2102	1	0
	conps-btormc-no-thp	53	53	0	298	0	278	45555	45550	11304	2395	0	0

Note: abcsuperprove only participated in the bit-vector track.

CoSA2

For solving the largest number of benchmarks.

Benchmark Contribution Award

Clifford Wolf

For contributing 3478 SINGLE benchmarks.

Conclusion

Word-level track

- finally word-level track at HWMCC
- 745 new benchmarks with 4177 safety properties
- only 3 model checker submissions

Next Edition

- bit-level *and* word-level tracks
- BTOR2 witnesses required
- DEEP for word-level?
- single-core track?

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