Scalable Bit-Blasting with Abstractions

Aina Niemetz[⋆], Mathias Preiner[⋆] and Yoni Zohar[◇]

[⋆] Stanford University [◇] Bar-Ilan University

CAV, July 25, 2024

Stanford University

CENTAUR
Stanford | Center for Automated Reasoning

Bar-Ilan University
אוניברסיטת בר־אילן

Satisfiability Modulo Theories (SMT) for Theory of Fixed-Size Bit-Vectors

Theory of Fixed-Size Bit-Vectors

- **constants, variables:** 010, $2_{[3]}$, $x_{[3]}$
- **bit-vector** operators: $<_u$, $>_s$, $\sim$, $\&$, $\gg$, $\gg_a$, $\circ$, $[:]$, $+$, $\cdot$, $\div$, $\ldots$
- **arithmetic** operators modulo 2^n (**overflow semantics!**)

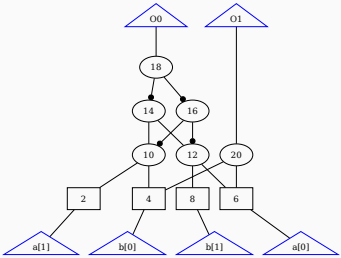
State of The Art for **quantifier-free** bit-vector formulas

- **Bit-Blasting**
  - » rewriting + simplifications + **reduction** to SAT
- **efficient** in practice (state-of-the-art SAT solvers)
- significant **increase** in formula size

Scalability of Bit-Blasting

- **does not generally scale well** with **increasing** bit-width
- ▷ especially with **arithmetic** operators
  - » large and complex Boolean circuits
- potential **bottleneck** for SAT solver
  - » in practice already as low as 32 bits
- especially severe for applications with **large bit-vectors**
  - » **smart contract verification:** 256 bits, heavy use of $\{\cdot, \div, \mathrm{mod}\}$
  - » **floating-point arithmetic** when word-blasting

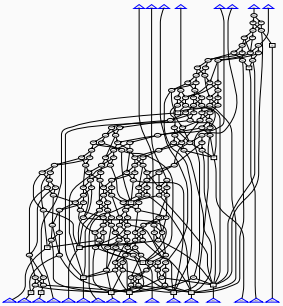
2-bit Multiplier (AIG circuit)



Scalability of Bit-Blasting

- **does not generally scale well** with **increasing** bit-width
- ▷ especially with **arithmetic** operators
  - » large and complex Boolean circuits
- potential **bottleneck** for SAT solver
  - » in practice already as low as 32 bits
- especially severe for applications with **large bit-vectors**
  - » **smart contract verification:** 256 bits, heavy use of $\{\cdot, \div, \text{mod}\}$
  - » **floating-point arithmetic** when word-blasting

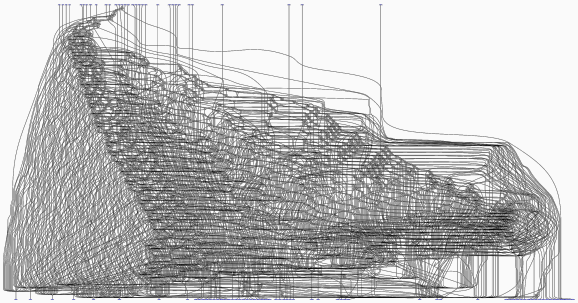
8-bit Multiplier (AIG circuit)



Scalability of Bit-Blasting

- **does not generally scale well** with **increasing** bit-width
- ▷ especially with **arithmetic** operators
  - » large and complex Boolean circuits
- potential **bottleneck** for SAT solver
  - » in practice already as low as 32 bits
- especially severe for applications with **large bit-vectors**
  - » **smart contract verification:** 256 bits, heavy use of $\{\cdot, \div, \mathrm{mod}\}$
  - » **floating-point arithmetic** when word-blasting

32-bit Multiplier (AIG circuit)



Contribution

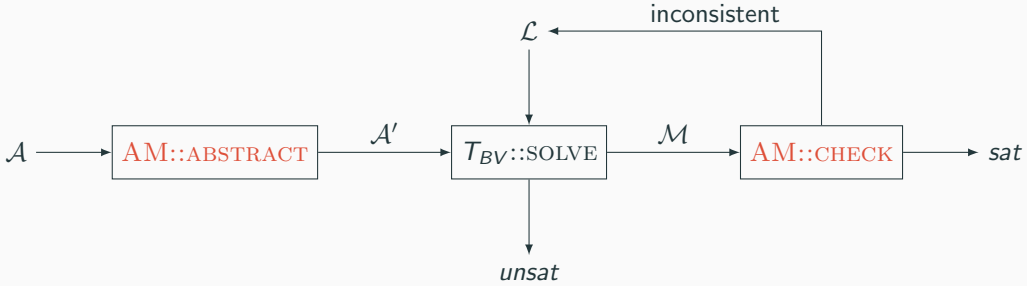
- a **CEGAR-style** abstraction-refinement framework for **theory BV**
  - based on **bit-blasting**
  - implemented in our SMT solver **Bitwuzla**
  - significantly **improves** performance on **large** bit-vectors

- **abstraction refinement scheme** for bit-vector operators $\{\cdot, \div, \mathrm{mod}\}$
  - **most expensive** operators for bit-blasting
  - 70 lemmas total

- abduction-based framework for **lemma synthesis**

- lemma **scoring** scheme

Abstraction-Refinement Loop

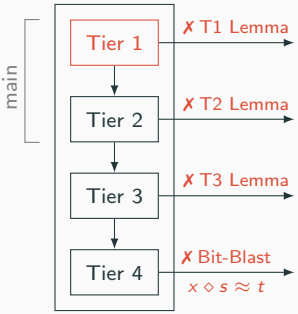


- abstract each relevant $\{\cdot, \div, \mathrm{mod}\}$ term with **fresh constant**
- **over-approximation**

- **check consistency** with true semantics of operators
  - » consistent: ✓
  - » inconsistent: refine abstraction

4-Tiered Refinement Scheme

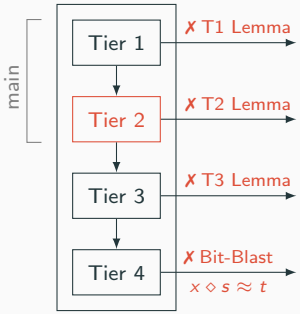
- **Tier 1** **Hand-Crafted Lemmas**
  - ▷ **basic** properties of the abstracted operator
    - e.g., abstraction t for $x \cdot s$ is odd iff x and s are odd
      » $t[0] \approx (x[0] \;\&\; s[0])$
  - ▷ properties described by **invertibility condition** IC
    - $\exists x.\, x \diamond s \approx t \Leftrightarrow IC[s, t]$
    - e.g., $x \cdot s$ has at least as many trailing zeros as x or s
      » $((-s \mid s) \;\&\; t) \approx t$
      » $((-x \mid x) \;\&\; t) \approx t$
  - ▷ **verified** up to bit-width 512 (17 lemmas)



processed in order

4-Tiered Refinement Scheme

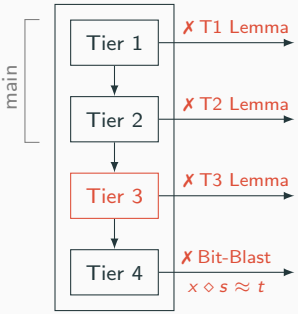
- **Tier 2 Synthesized Lemmas**
  - synthesized via **syntax-restricted abduction** with **cvc5** (offline)
    - **abduction:** A, B, find C s.t. $A \wedge C \Rightarrow B$ valid, $A \wedge C$ sat
    - find **lemma** ℓ such that $\top \wedge \neg\ell \Rightarrow x \diamond s \not\approx t$
  - e.g., for $x \cdot s \approx t$
    - $x \not\approx (1 \oplus (x \ll (s \oplus t)))$
  - lemmas **filtered** based on **lemma score**
  - **with respect to hand-crafted (tier 1) lemmas**
  - **verified** up to bit-width 512 (53 lemmas)



processed in order

4-Tiered Refinement Scheme

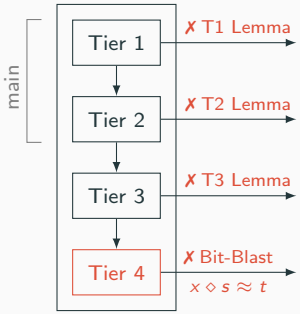
- **Tier 3** **Value Instantiation** Lemmas
  - rule out **current inconsistent model value**
  - only added if none in tiers 1–2 are violated
  - **limited** fallback strategy
    - » limited to $w/8$ instantiations per term
    - » e.g., 4 instantiations for 32 bit
  - e.g., for $x \cdot s$ and $\mathcal{M} = \{x_{[32]} = 3, s_{[32]} = 6, t_{[32]} = 1\}$, we add lemma $(x = 3 \wedge s = 6) \Rightarrow t = 18$



processed in order

4-Tiered Refinement Scheme

- **Tier 4 Bit-blasting** lemmas
  - **last resort**
  - add lemma to **enforce bit-blasting** of the abstracted term
  - e.g., lemma $t \approx x \cdot s$ for $x \cdot s$



processed in order

Lemma Score

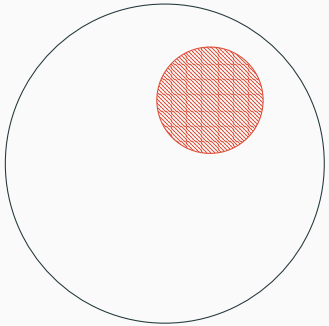
- **Metric for quality of a lemma**

▷ Given abstracted term $x \diamond s$ and lemma $\ell[x, s, t]$ s.t. $x \diamond s \approx t \Rightarrow \ell$, then
$\textsc{Score}(\ell, w) :=$ num. triplets (v^x, v^s, v^t) where $\ell[v^x, v^s, v^t] = \top$.

Example. $x \cdot s$ with $w = 4$

▷ Worst score: $2^4 \times 2^4 \times 2^4 = 4096$ $(\ell = \top)$

▷ Best score: $2^4 \times 2^4 = 256$ $(\ell = x \cdot s \approx t)$



Lemma Score

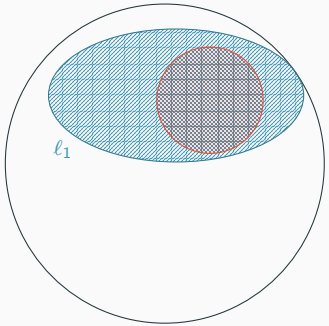
- **Metric for quality of a lemma**

▷ Given abstracted term $x \diamond s$ and lemma $\ell[x, s, t]$ s.t. $x \diamond s \approx t \Rightarrow \ell$, then
$\textsc{Score}(\ell, w) :=$ num. triplets (v^x, v^s, v^t) where $\ell[v^x, v^s, v^t] = \top$.

Example. $x \cdot s$ with $w = 4$

▷ Worst score: $2^4 \times 2^4 \times 2^4 = 4096$ ($\ell = \top$)

▷ Best score: $2^4 \times 2^4 = 256$ ($\ell = x \cdot s \approx t$)



ℓ_1: 2416

Lemma Score

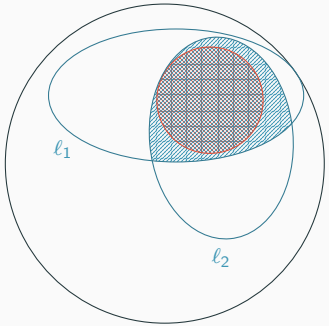
- **Metric for quality of a lemma**

▷ Given abstracted term $x \diamond s$ and lemma $\ell[x, s, t]$ s.t. $x \diamond s \approx t \Rightarrow \ell$, then
$\textsc{Score}(\ell, w) :=$ num. triplets (v^x, v^s, v^t) where $\ell[v^x, v^s, v^t] = \top$.

Example. $x \cdot s$ with $w = 4$

▷ Worst score: $2^4 \times 2^4 \times 2^4 = 4096$ ($\ell = \top$)

▷ Best score: $2^4 \times 2^4 = 256$ ($\ell = x \cdot s \approx t$)



ℓ_1: 2416
ℓ_2: 2791

Lemma Score

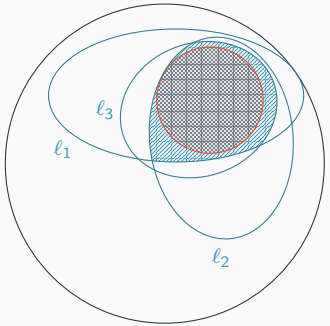
- **Metric for quality of a lemma**

▷ Given abstracted term $x \diamond s$ and lemma $\ell[x, s, t]$ s.t. $x \diamond s \approx t \Rightarrow \ell$, then
$\textsc{Score}(\ell, w) :=$ num. triplets (v^x, v^s, v^t) where $\ell[v^x, v^s, v^t] = \top$.

Example. $x \cdot s$ with $w = 4$

▷ Worst score: $2^4 \times 2^4 \times 2^4 = 4096$ ($\ell = \top$)

▷ Best score: $2^4 \times 2^4 = 256$ ($\ell = x \cdot s \approx t$)



ℓ_1: 2416
ℓ_2: 2791
ℓ_3: 1961

Lemma Score

- **Metric for quality of a lemma**

▷ Given abstracted term $x \diamond s$ and lemma $\ell[x, s, t]$ s.t. $x \diamond s \approx t \Rightarrow \ell$, then
$\textsc{Score}(\ell, w) :=$ num. triplets (v^x, v^s, v^t) where $\ell[v^x, v^s, v^t] = \top$.

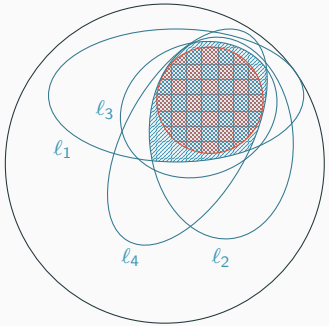
Example. $x \cdot s$ with $w = 4$

▷ Worst score: $2^4 \times 2^4 \times 2^4 = 4096$ ($\ell = \top$)

▷ Best score: $2^4 \times 2^4 = 256$ ($\ell = x \cdot s \approx t$)

▷ **Score for hand-crafted $\{\ell_1, \ldots, \ell_4\}$: 704**

» **rule out 88% of incorrect triplets**



ℓ_1: 2416
ℓ_2: 2791
ℓ_3: 1961
ℓ_4: 2048

Lemma Score

- **Metric for quality of a lemma**

▷ Given abstracted term $x \diamond s$ and lemma $\ell[x, s, t]$ s.t. $x \diamond s \approx t \Rightarrow \ell$, then
$\textsc{Score}(\ell, w) :=$ num. triplets (v^x, v^s, v^t) where $\ell[v^x, v^s, v^t] = \top$.

Example. $x \cdot s$ with $w = 4$

▷ Worst score: $2^4 \times 2^4 \times 2^4 = 4096$ ($\ell = \top$)

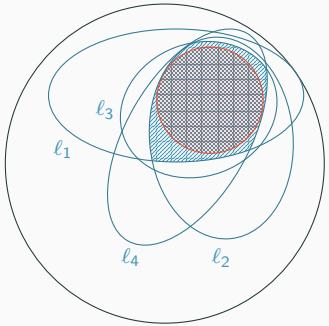
▷ Best score: $2^4 \times 2^4 = 256$ ($\ell = x \cdot s \approx t$)

▷ **Score for hand-crafted $\{\ell_1, \ldots, \ell_4\}$: 704**

» **rule out 88% of incorrect triplets**

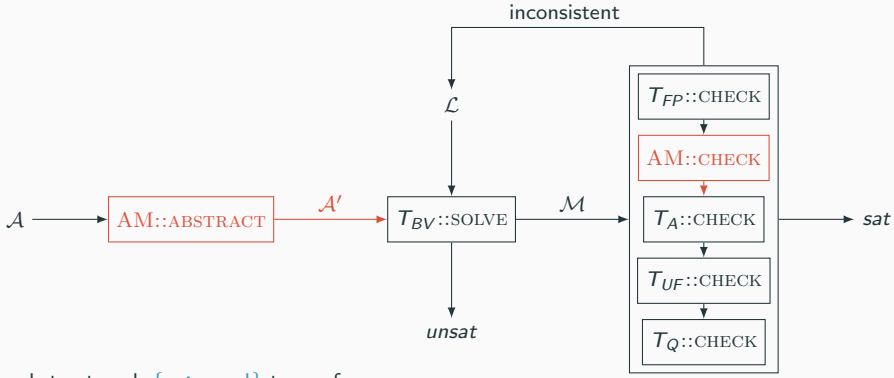
- **Overall scores** (tiers 1+2, w = 4)

» $x \cdot s$ 490 ≅ 94% (19 lemmas)
» $x \div s$ 396 ≅ 96% (36 lemmas)
» $x \bmod s$ 400 ≅ 96% (15 lemmas)



ℓ_1: 2416
ℓ_2: 2791
ℓ_3: 1961
ℓ_4: 2048

Bitwuzla: Lemmas on Demand Loop with Abstraction Module



- abstract each $\{\cdot, \div, \text{mod}\}$ term of size ≥ 32 with **fresh constant**
- optional: **assertion abstraction** [?]
  - » interleaved with term abstraction
  - » effective if unsat core very small

- order **not arbitrary**
  - » T_{FP} word-blasted to T_{BV}
  - » T_A, T_{UF} and T_Q require consistent T_{BV} abstraction

Evaluation

- implemented in **Bitwuzla**
- **Benchmarks**
  - **smart contract verification**
    - » *certora* (Certora Prover)
    - » *ethereum* (hevm, Ethereum Foundation)
    - 256 bit bit-vectors
  - **crafted benchmarks**
    - » *syrew*
    - controlled set to evaluate effectiveness
    - equivalence checks for each operator
    - enumerated by SyGuS (cvc5) for $w = 4$
    - instantiated for $w = \{16, 32, \ldots, 8192\}$
  - **translation validation of ZK proofs**
    - » *ff*
    - 510 bit bit-vectors
  - **SMT-LIB**
    - » *smtlib*
    - all quantifier-free and quantified logics supported by Bitwuzla (24 in total)
    - combinations of BV, FP, UF, Arrays

Evaluation

▶ Configurations

▷ Base Configuration

- Bitwuzla [0.3.2]

▷ Abstraction

- Bitwuzla-abstr [Bitwuzla + abstraction]
- $\{\cdot, \div, \mathrm{mod}\}$ of size ≥ 32

▷ Other Configurations

- cvc5 [1.1.0]
- Z3 [4.12.4]

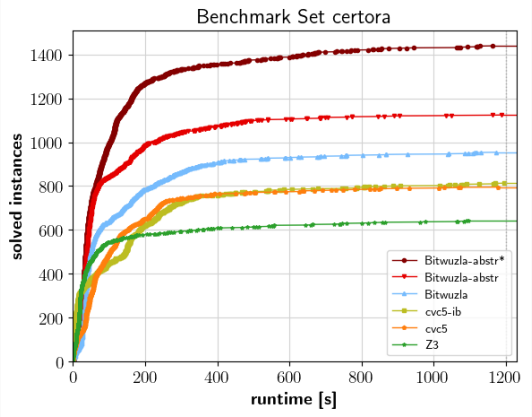
» support all logics supported by Bitwuzla

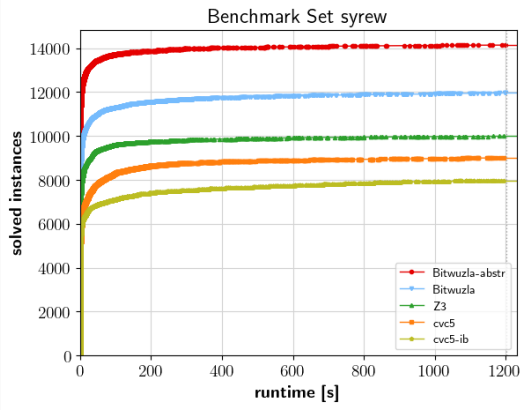
- cvc5-ib [cvc5 int-blasting]

» translation from BV to NIA

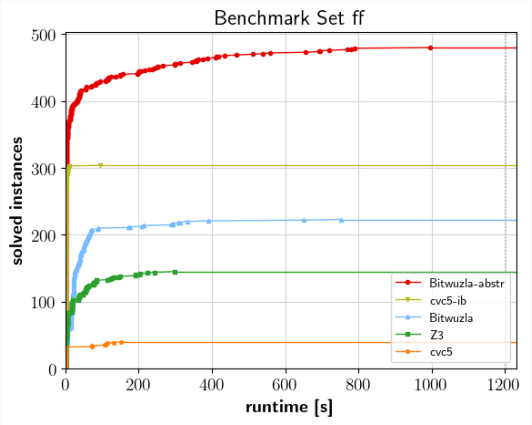
» motivated by large bit-vectors

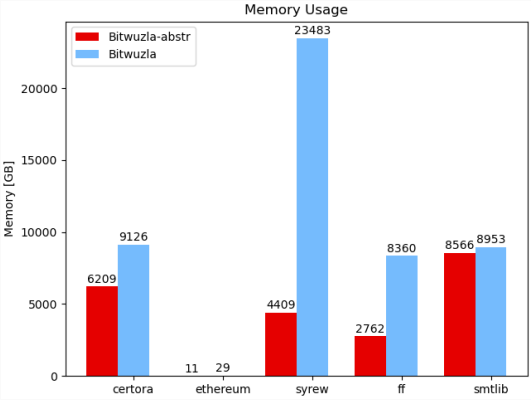
Evaluation


Benchmark Set certora
solved instances
runtime [s]
0
200
400
600
800
1000
1200
1400
Bitwuzla-abstr*
Bitwuzla-abstr
Bitwuzla
cvc5-ib
cvc5
Z3


Benchmark Set syrew
solved instances
runtime [s]
0
2000
4000
6000
8000
10000
12000
14000
Bitwuzla-abstr
Bitwuzla
Z3
cvc5
cvc5-ib

Evaluation: Memory Usage





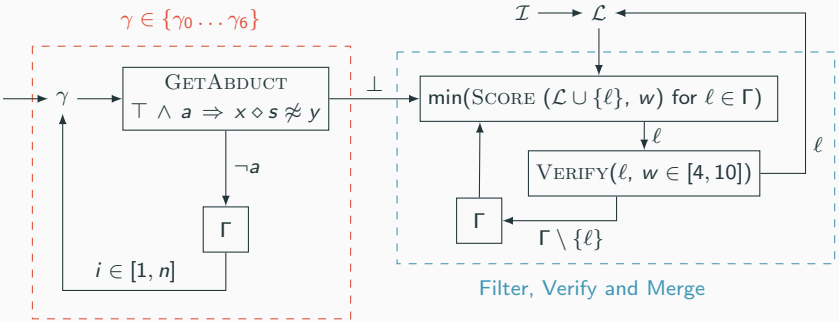
Evaluation: Abstraction Statistics of Solved Instances

- **80% solved without bit-blasting of abstracted terms**
  - » Tier 1–3 refinement lemmas were sufficent to solve the problem
- 20% required bit-blasting of abstracted terms
  - » Only 37% of multiplication, 13% of division, 2% of remainder terms bit-blasted
- On average 37 refinement iterations (median 4)
- 24,033 instances solved with abstracted terms

Conclusion

- A **CEGAR-style** abstraction-refinement for **theory BV** based on **bit-blasting**
  - » integration independent of solver architecture
- **significantly improves scalability of bit-blasting**
  - » for majority of solved instances full arithmetic circuit not required
- Implemented in **Bitwuzla**
- Artifact: `https://doi.org/10.5281/zenodo.10913320`
- `https://bitwuzla.github.io`

Lemma Synthesis



- **verified lemmas for bit-widths [1, 256]**
  - Bitwuzla, cvc5, Yices, Z3
  - 8 hours time limit, 8 GB memory limit
  - 16,896 benchmark, 6348 CPU hours

Evaluation: Improvements for FP Arithmetic on Set smtlib

- ▷ Bitwuzla employs word-blasting for FP logics
  - Reduction of FP to bit-vectors with SymFPU [**?**]
- ▷ Operations on unpacked formats require full precision arithmetic
  - Float32 `fp.mul`: 48-bit `bvmul`
  - Float64 `fp.mul`: 106-bit `bvmul`

Bitwuzla vs. Bitwuzla-abstr

Logic	Solved (Diff)	Time [s] (Common)
FP	+5	−16%
BVFP	0	−45%
QF_ABVFP	+1	−33%
QF_ABVFPLRA	0	−23%
QF_BVFP	+1	−45%
QF_BVFPLRA	+9	−46%
QF_FP	+23	−13%
QF_FPLRA	+1	−7%
QF_BV/float	+11	−16%

Evaluation

Benchmarks	Solver	Solved	Timeout	Memout	Time [s]	Mem [GB]
certora$_1$ (850)	Abstr-ta	573	231	46	448k	2,492
	Abstr-a	386	140	324	681k	5,201
	Bitwuzla-abstr	258	155	437	760k	4,807
	cvc5-ib	147	674	0	879k	667
	Bitwuzla	111	86	653	915k	6,182
	cvc5	90	113	610	923k	6,064
	Z3	30	447	373	989k	4,944
certora$_2$ (1,138)	Abstr-ta	866	264	8	370k	1,024
	Bitwuzla-abstr	866	263	9	384k	1,402
	Abstr-a	844	269	25	433k	2,661
	Bitwuzla	843	266	29	439k	2,944
	cvc5	705	223	210	603k	4,027
	cvc5-ib	666	472	0	643k	106
	Z3	612	492	34	679k	1,866
ethereum (3,173)	Bitwuzla-abstr	3,173	0	0	407	11
	Bitwuzla	3,173	0	0	720	29
	Z3	3,169	4	0	6k	107
	cvc5	3,158	0	1	18k	36
	cvc5-ib	3,141	20	0	39k	21

Evaluation

Benchmarks	Solver	Solved	Timeout	Memout	Time [s]	Mem [GB]
syrew (15,000)	Bitwuzla-abstr	14,142	583	276	1,225k	4,409
	Bitwuzla	11,961	744	2,296	3,955k	23,483
	Z3	9,992	833	4,175	6,198k	39,506
	cvc5	9,003	797	5,200	7,498k	48,421
	cvc5-ib	7,974	5,137	1,632	8,836k	19,850
ff (1,224)	cvc5-ff	973	129	122	313k	1,364
	Bitwuzla-abstr	480	729	15	913k	2,762
	cvc5-ib	304	822	98	1,104k	1,074
	Bitwuzla	223	71	930	1,211k	8,360
	Z3	145	56	1,023	1,299k	8,893
	cvc5	40	0	1,184	1,422k	9,523
smtlib (155,269)	Bitwuzla-abstr	148,554	1,944	152	8,770k	8,566
	Bitwuzla	148,492	1,966	193	8,748k	8,953
	Z3	145,121	4,846	565	13,528k	18,278
	cvc5	144,829	3,775	285	13,513k	11,029
	cvc5-ib	127,144	24,479	194	39,647k	15,233

36 unique

Limits: 1200 seconds CPU time limit, 8GB memory limit