Speeding Up Quantified Bit-Vector SMT Solvers by Bit-Width Reductions and Extensions

Martin Jonáš, Jan Strejček

Fondazione Bruno Kessler, Italy
Faculty of Informatics, Masaryk University, Czech Republic

Theory of Bit-Vectors

The theory of bit-vectors describes bounded integers (or vectors of bits of fixed size) with:

- bitwise operations,
- arithmetic operations,
- signed and unsigned comparison.

Theory of Bit-Vectors

The theory of bit-vectors describes bounded integers (or vectors of bits of fixed size) with:

- bitwise operations,
- arithmetic operations,
- signed and unsigned comparison.

In many software verification applications, quantifiers are necessary. For example in

- invariant generation,
- ranking function synthesis,
- cycle summarization,
- symbolic state equality test.

Bit-with Reductions

In the formula

$$\forall x^{[32]} \exists y^{[32]} \, (x^{[32]} + y^{[32]} = 0^{[32]})$$

- $x^{[32]}$ and $y^{[32]}$ are variables of bit-width 32,
- $0^{[32]}$ is a constant of bit-width 32.

Bit-with Reductions

In the formula

$$\forall x^{[32]} \exists y^{[32]} \, (x^{[32]} + y^{[32]} = 0^{[32]})$$

- $x^{[32]}$ and $y^{[32]}$ are variables of bit-width 32,
- $0^{[32]}$ is a constant of bit-width 32.

Reduction to 4 bits

$$\forall x^{[4]} \exists y^{[4]} \, (x^{[4]} + y^{[4]} = 0^{[4]})$$

Bit-with Reductions

In the formula

$$\forall x^{[32]} \exists y^{[32]} \, (x^{[32]} + y^{[32]} = 0^{[32]})$$

- $x^{[32]}$ and $y^{[32]}$ are variables of bit-width 32,
- $0^{[32]}$ is a constant of bit-width 32.

Reduction to 4 bits

$$\forall x^{[4]} \exists y^{[4]} \, (x^{[4]} + y^{[4]} = 0^{[4]})$$

Observation
Performance of the solvers for quantified bit-vector formulas usually depends on the bit-widths used in the formula.

Bit-width Reductions and Satisfiability

Observation from our LPAR 2018 paper
Vast majority of quantified bit-vectors does not change their satisfiability from very low bit-widths.

	Total	Different answer for some bit-width			
		≥ 1b	≥ 2b	≥ 4b	≥ 8b
Count	4905	216	95	32	14
%	100	4.4	1.9	0.65	0.29

Bit-width Reductions and Satisfiability

Observation from our LPAR 2018 paper

Vast majority of quantified bit-vectors does not change their satisfiability from very low bit-widths.

		Different answer for some bit-width			
	Total	$\geq 1b$	$\geq 2b$	$\geq 4b$	$\geq 8b$
Count	4905	216	95	32	14
%	100	4.4	1.9	0.65	0.29

Challenges

- Can the safe bit-width be computed from the formula?
- Can the observation be leveraged to speed-up SMT solvers?

Outline of the Presentation

1. How to decide satisfiability using bit-width reductions
2. How to decide unsatisfiability using bit-width reductions
3. Our implementation in a single algorithm
4. Experimental evaluation

Symbolic Models of Quantified Formulas

Consider the formula

$$\forall x^{[32]} \exists y^{[32]} \, (x^{[32]} + y^{[32]} = 0^{[32]})$$

It is satisfiable. What is its symbolic model?

Symbolic Models of Quantified Formulas

Consider the formula

$$\forall x^{[32]} \exists y^{[32]} \, (x^{[32]} + y^{[32]} = 0^{[32]})$$

It is satisfiable. What is its symbolic model? The Skolem function

$$y^{[32]} \mapsto -x^{[32]}$$

Symbolic Models of Quantified Formulas

Consider the formula

$$\forall x^{[32]} \exists y^{[32]} \, (x^{[32]} + y^{[32]} = 0^{[32]})$$

It is satisfiable. What is its symbolic model? The Skolem function

$$y^{[32]} \mapsto -x^{[32]}$$

because

$$\forall x^{[32]} \, (x^{[32]} + (-x^{[32]}) = 0^{[32]})$$

is satisfiable.

Deciding Satisfiability With Reductions

$$\forall x^{[32]} \exists y^{[32]} \, (x^{[32]} + y^{[32]} = 0^{[32]})$$

Deciding Satisfiability With Reductions

$$\forall x^{[32]} \exists y^{[32]} \, (x^{[32]} + y^{[32]} = 0^{[32]}) \xrightarrow{\text{reduce}} \forall x^{[4]} \exists y^{[4]} \, (x^{[4]} + y^{[4]} = 0^{[4]})$$

Deciding Satisfiability With Reductions

$$\forall x^{[32]} \exists y^{[32]} \, (x^{[32]} + y^{[32]} = 0^{[32]}) \xrightarrow{\text{reduce}} \forall x^{[4]} \exists y^{[4]} \, (x^{[4]} + y^{[4]} = 0^{[4]})$$

$\downarrow$ solve

SAT: $y^{[4]} \mapsto -x^{[4]}$

Deciding Satisfiability With Reductions

$$\forall x^{[32]} \exists y^{[32]} \, (x^{[32]} + y^{[32]} = 0^{[32]}) \xrightarrow{\text{reduce}} \forall x^{[4]} \exists y^{[4]} \, (x^{[4]} + y^{[4]} = 0^{[4]})$$

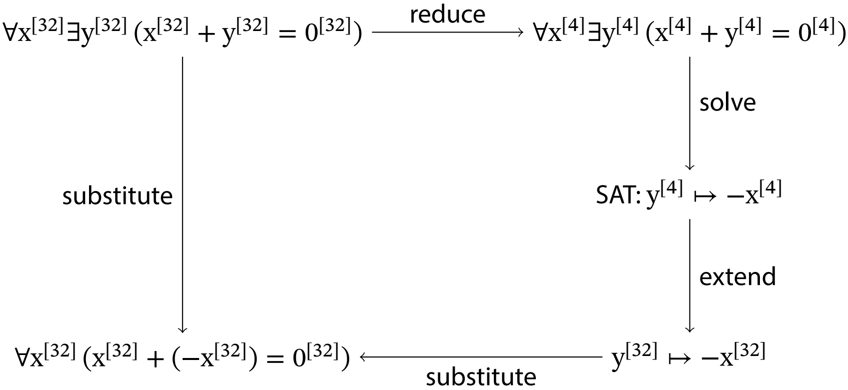
$$\Big\downarrow \text{ solve}$$

SAT: $y^{[4]} \mapsto -x^{[4]}$

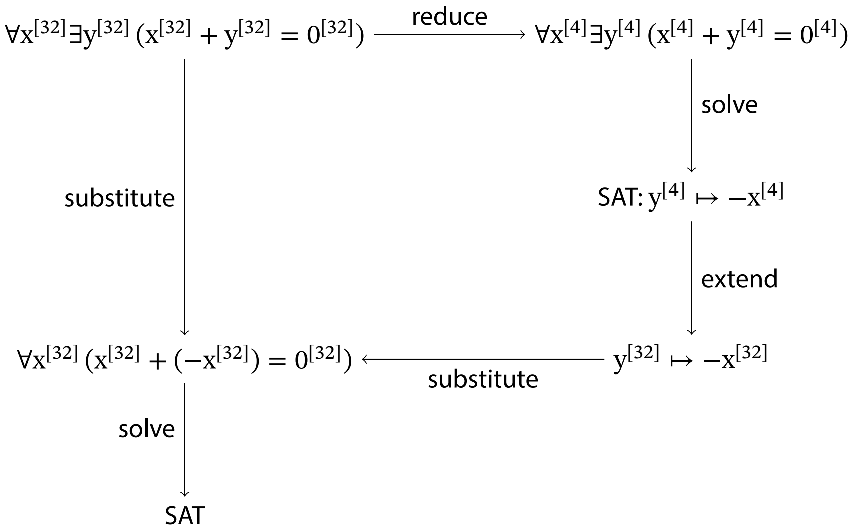
$$\Big\downarrow \text{ extend}$$

$$y^{[32]} \mapsto -x^{[32]}$$

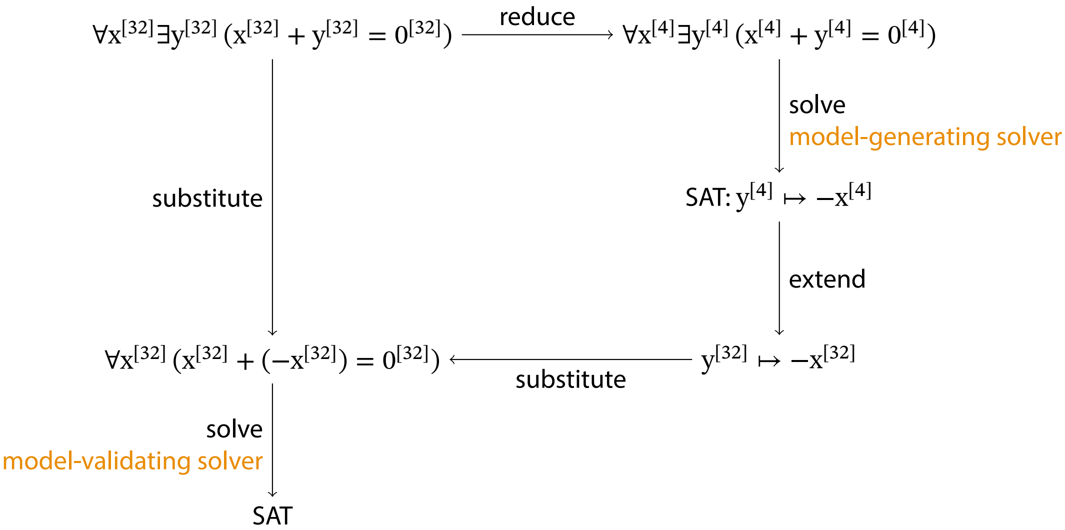
Deciding Satisfiability With Reductions



Deciding Satisfiability With Reductions



Deciding Satisfiability With Reductions



Symbolic Countermodels of Quantified Formulas

Consider the formula

$$\forall x^{[32]}\,(x^{[32]} + y^{[32]} = 0^{[32]})$$

It is unsatisfiable. What is its symbolic countermodel?

Symbolic Countermodels of Quantified Formulas

Consider the formula

$$\forall x^{[32]} \left(x^{[32]} + y^{[32]} = 0^{[32]}\right)$$

It is unsatisfiable. What is its symbolic countermodel? The Herbrand function

$$x^{[32]} \mapsto -y^{[32]} + 1^{[32]}$$

Symbolic Countermodels of Quantified Formulas

Consider the formula

$$\forall x^{[32]} \, (x^{[32]} + y^{[32]} = 0^{[32]})$$

It is unsatisfiable. What is its symbolic countermodel? The Herbrand function

$$x^{[32]} \mapsto -y^{[32]} + 1^{[32]}$$

because

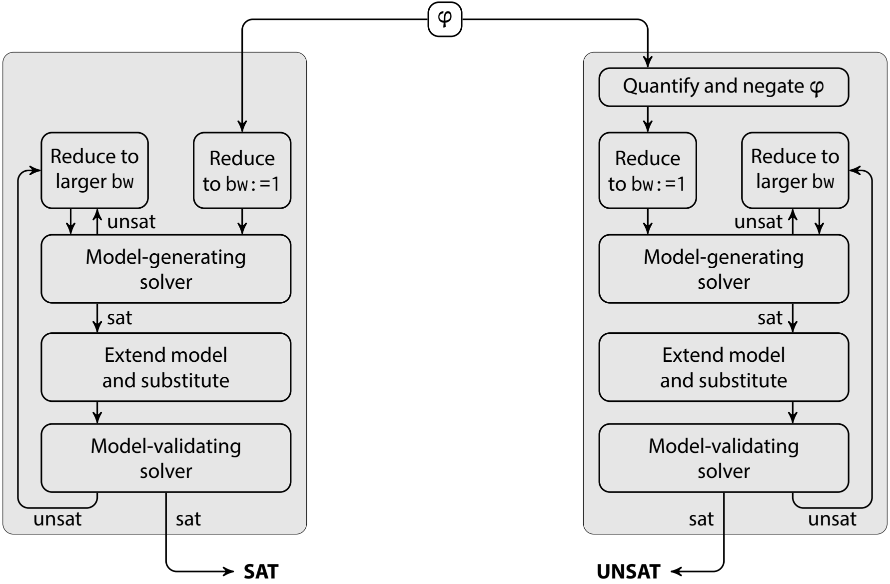
$$(-y^{[32]} + 1^{[32]}) + y^{[32]} = 0^{[32]}$$

is unsatisfiable.

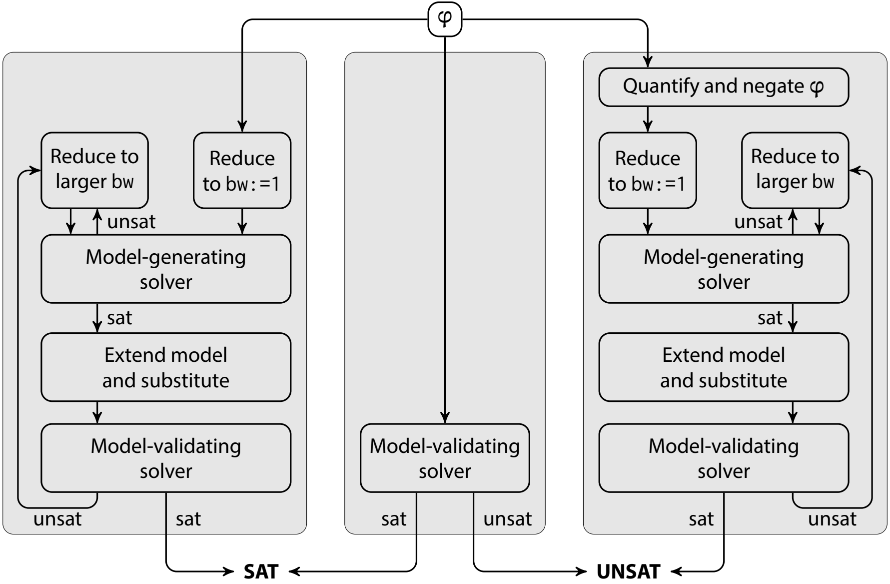
Overall Algorithm

1. Reduce the formula.
2. Try to solve its satisfiability.
3. Get a symbolic model/countermodel.
4. Extend the symbolic model/countermodel to the original bit-width.
5. Check whether it is a symbolic model/countermodel of the original formula.
6. If unsuccessful, increase the reduction bit-width and repeat.

Overall Algorithm – Practical Implementation



Overall Algorithm – Practical Implementation



Experimental Evaluation

Implementation

- reductions, extensions, and the solving algorithm
- in C++, using Z3 API

Experimental Evaluation

Implementation

- reductions, extensions, and the solving algorithm
- in C++, using Z3 API

Benchmarks

- 5741 quantified BV formulas from SMT-LIB
- 8 benchmark families

Experimental Evaluation

Implementation

- reductions, extensions, and the solving algorithm
- in C++, using Z3 API

Benchmarks

- 5741 quantified BV formulas from SMT-LIB
- 8 benchmark families

Model-generating solver

- Boolector

Experimental Evaluation

Implementation

- reductions, extensions, and the solving algorithm
- in C++, using Z3 API

Benchmarks

- 5741 quantified BV formulas from SMT-LIB
- 8 benchmark families

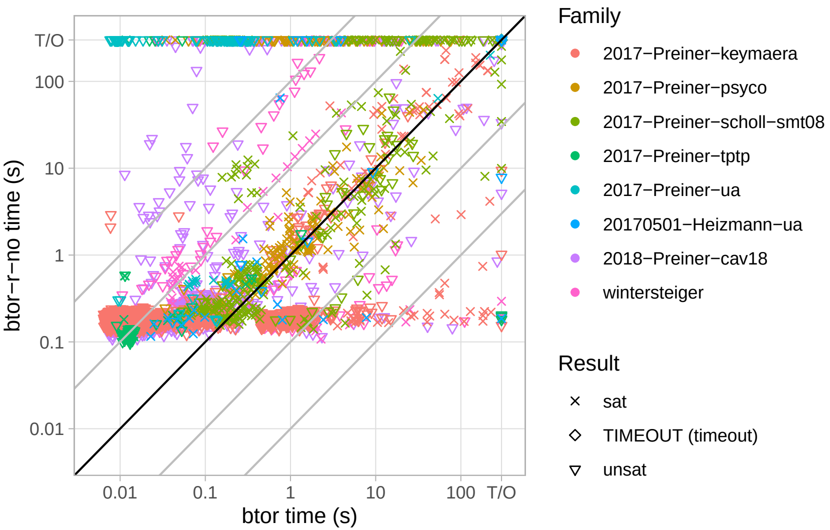
Model-generating solver

- Boolector

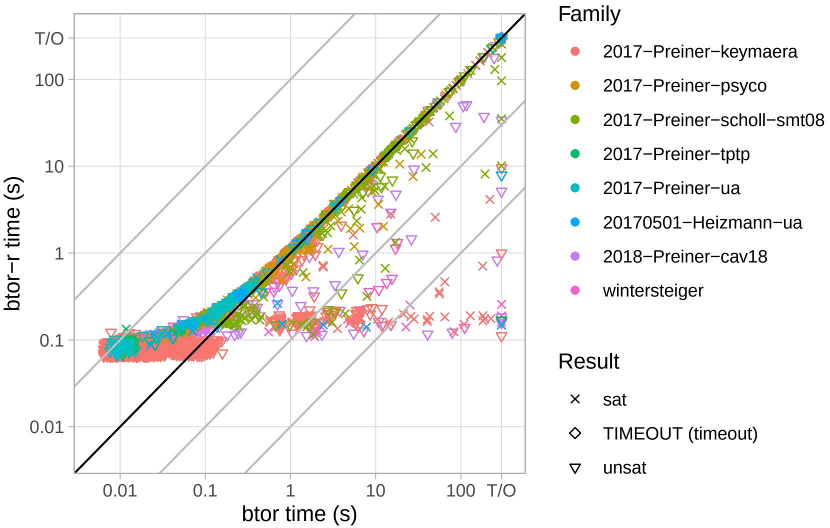
Model-validating solver

- Boolector
- CVC4
- Q3B

Effect of Reductions on Boolector



Effect of Reductions on Boolector



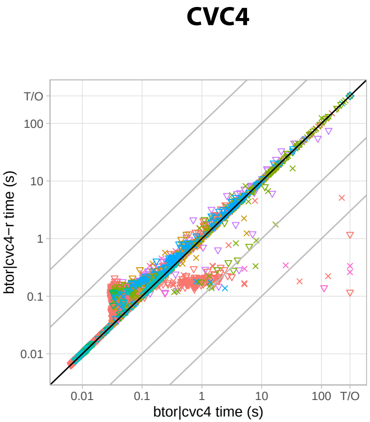
Effect of Reductions on Boolector

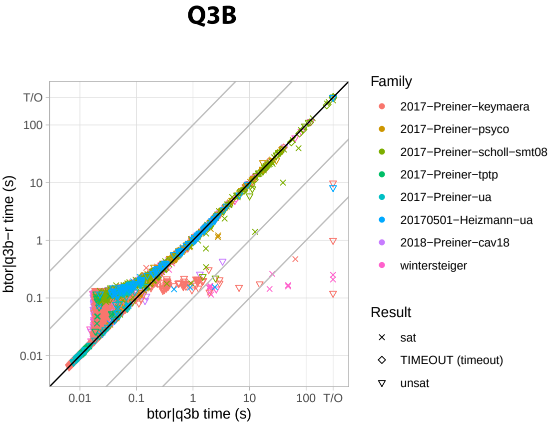
Reducing solver solved 449 formulas faster than Boolector itself.

Used bit-widths for these formulas:

Reduced bit-width	1	2	4	8	16
Count	185	119	122	17	6

Effect on CVC4 and Q3B





Newly Solved Formulas

With reductions, all the solvers were able to solve some previously unsolved formulas:

- Boolector – 22 formulas
- CVC4 – 4 formulas
- Q3B – 7 formulas

Conclusions

We have developed a technique that

- solves satisfiability of formulas with reduced bit-widths,
- extends the models/countermodels to the original bit-width,
- verifies the extended models/countermodels.

We have shown that this technique

- can improve performance of state-of-the-art SMT solvers,
- allows solving previously unsolved formulas.

Conclusions

We have developed a technique that

- solves satisfiability of formulas with reduced bit-widths,
- extends the models/countermodels to the original bit-width,
- verifies the extended models/countermodels.

We have shown that this technique

- can improve performance of state-of-the-art SMT solvers,
- allows solving previously unsolved formulas.

Thank you for your attention.