Towards Bit-Width-Independent Proofs in SMT Solvers

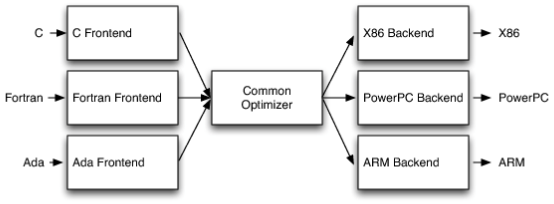
Aina Niemetz[1] Mathias Preiner[1] Andrew Reynolds[2] Yoni Zohar[1]
Clark Barrett[1] Cesare Tinelli[2]

1. Stanford University, Stanford, USA

2. The University of Iowa, Iowa City, USA

Why Bit-width Independence?

LLVM [Image from Lattner 2012]



Alive [Lopes et al. 2015]

Language + tool for:

- Writing optimizations
- **Verifying them**
- Generating code

```
Name: AndOrXor:1733
%cmp1 = icmp ne %A, 0
%cmp2 = icmp ne %B, 0
%r = or %cmp1, %cmp2
  =>
%C = or %A, %B
%r = icmp ne %C, 0
```

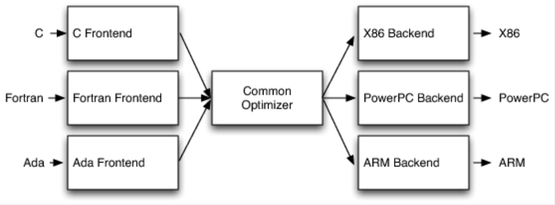
$\Longleftrightarrow$

$$(A \neq 0 \vee B \neq 0) \Leftrightarrow (A \,|\, B \neq 0)$$

Alive proves validity up to a **certain** bit-width

Why Bit-width Independence?

LLVM [Image from Lattner 2012]



Alive [Lopes et al. 2015]

Language + tool for:

- Writing optimizations
- **Verifying them**
- Generating code

```
Name: AndOrXor:1733
%cmp1 = icmp ne %A, 0
%cmp2 = icmp ne %B, 0
%r = or %cmp1, %cmp2
  =>
%C = or %A, %B
%r = icmp ne %C, 0
```

$\Longleftrightarrow$

$$(A \neq 0 \vee B \neq 0) \Leftrightarrow (A \,|\, B \neq 0)$$

Our Goal: proving validity for **every** bit-width

Goal:
proving validity for **every** bit-width

Express

Solve

Examples

Goal:

proving validity for **every** bit-width

Express

Solve

Examples

Bit-vectors Representations

Fixed-width Bit-vectors

- Many-sorted First-order Logic
- Sorts: $\mathbf{BV}[1], \mathbf{BV}[2], \ldots$
- Sorted equality, functions, predicates
- Used in SMT-LIB 2

$$(x \neq_3 000 \vee y \neq_3 000) \Leftrightarrow (x \mid_3 y \neq_3 000)$$

Arbitrary-width Bit-vectors

- Variables range over bit-vectors of arbitrary width
- Bit-width can be quantified
- Many-sorted first-order logic does not seem like a natural fit

$$\forall k.(x \neq_k 0\ldots0 \vee y \neq_k 0\ldots0) \Leftrightarrow (x \mid_k y \neq_k 0\ldots0)$$

Language for Bit-vectors of Parametric Width

Language

- **Unsorted** functions & predicates
- Bit-vector variables: $X = \{x_0, x_1, \ldots\}$
- Bit-vector constants: $C = \{c_0, c_1, \ldots\}$

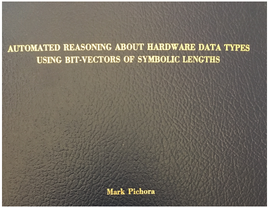
$(x_1 \neq c_0 \vee x_2 \neq c_0) \Leftrightarrow (x_1 \mid x_2 \neq c_0)$

Auxiliary Maps

$t(N) = \{0, n + m, ...\}$

- $\omega^{\mathrm{b}} : X \cup C \to t(N)$ **symbolic bit-width**
- $\omega^{\mathrm{N}} : C \to t(N)$ **symbolic value**

- **Validity: always w.r.t. a given** ω
- considering all integer interpretations
- Variant of [Pichora 2003]



Language for Bit-vectors of Parametric Width

Language

- **Unsorted** functions & predicates
- Bit-vector variables: $X = \{x_0, x_1, \ldots\}$
- Bit-vector constants: $C = \{c_0, c_1, \ldots\}$

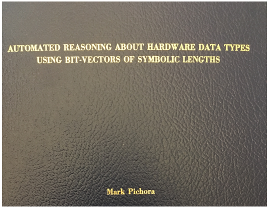
$$(x_1 \neq c_0 \vee x_2 \neq c_0) \Leftrightarrow (x_1 \mid x_2 \neq c_0)$$
with
$$\omega^{\mathrm{b}}(x_1) = \omega^{\mathrm{b}}(x_2) = \omega^{\mathrm{b}}(c_0) = k$$
$$\omega^{\mathrm{N}}(c_0) = 0$$

Auxiliary Maps

$t(N) = \{0, n + m, ...\}$

- $\omega^{\mathrm{b}} : X \cup C \rightarrow t(N)$ **symbolic bit-width**
- $\omega^{\mathrm{N}} : C \rightarrow t(N)$ **symbolic value**

- **Validity: always w.r.t. a given** ω
- considering all integer interpretations
- Variant of [Pichora 2003]



Language for Bit-vectors of Parametric Width

Language

- **Unsorted** functions & predicates
- Bit-vector variables: $X = \{x_0, x_1, \ldots\}$
- Bit-vector constants: $C = \{c_0, c_1, \ldots\}$

$$(x_1 \neq c_0 \vee x_2 \neq c_0) \Leftrightarrow (x_1 \mid x_2 \neq c_0)$$
with
$$\omega^{\mathrm{b}}(x_1) = \omega^{\mathrm{b}}(x_2) = \omega^{\mathrm{b}}(c_0) = k$$
$$\omega^{\mathrm{N}}(c_0) = 0$$

Auxiliary Maps

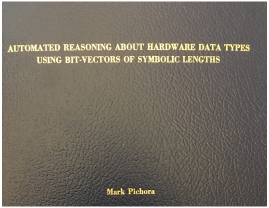
$t(N) = \{0, n + m, \ldots\}$

- $\omega^{\mathrm{b}} : X \cup C \rightarrow t(N)$ **symbolic bit-width**
- $\omega^{\mathrm{N}} : C \rightarrow t(N)$ **symbolic value**

Bad ω

$\omega^{\mathrm{b}}(x_1) = k, \omega^{\mathrm{b}}(x_2) = k + 1$

- **Validity: always w.r.t. a given ω**
- considering all integer interpretations
- Variant of [Pichora 2003]



Goal:

proving validity for **every** bit-width

Express

Solve

Examples

Solving Bit-vector Formulas with Parametric Width

Possibilities

- Bit-blasting (infinite SAT-instance)
- Specialized solver
- Translation to strings
- **Translation to integers**

From Bit-vectors to Integers

- Semantics for many operators is already built-in (exceptions: $\&, |, \ldots$)
- Benefit from advancements in integer-solving

Solving Bit-vector Formulas with Parametric Width

Possibilities

- ~~Bit-blasting~~ (infinite SAT-instance)
- Specialized solver
- Translation to strings
- **Translation to integers**

From Bit-vectors to Integers

- Semantics for many operators is already built-in (exceptions: $\&, |, \ldots$)
- Benefit from advancements in integer-solving

Translation

$$Tr: \quad BV \quad \mapsto \quad NIA$$

$$
\begin{array}{lll}
x & \mapsto & x \\
c & \mapsto & \omega^{\mathrm{N}}(c) \mod 2^k \\
k = \omega^{\mathrm{b}}(x) & &
\end{array}
$$

$$
\begin{array}{lll}
= & \mapsto & = \\
x <_{\mathrm{u}} y & \mapsto & x < y \\
x <_{\mathrm{s}} y & \mapsto & to\_s(k, x) < to\_s(k, y)
\end{array}
$$

$$
\begin{array}{lll}
x + y & \mapsto & (x + y) \mod 2^k \\
x \ll y & \mapsto & (x \cdot 2^y) \mod 2^k
\end{array}
$$

$\cdot$, div, mod, $\sim$, $-$, $\gg$, $\circ$ are handled similarly

$$
\begin{array}{lll}
x \ \& \ y & \mapsto & \Sigma_{i=0}^{k} 2^i \cdot \min(x[i], y[i]) \\
x \mid y & \mapsto & \Sigma_{i=0}^{k} 2^i \cdot \max(x[i], y[i]) \\
x \oplus y & \mapsto & \Sigma_{i=0}^{k} 2^i \cdot |x[i] - y[i]|
\end{array}
$$

$$\varphi \quad \mapsto \quad Tr(\varphi)$$

Translation

$$Tr: \quad BV \mapsto NIA$$

$$\begin{aligned} x &\mapsto x \\ c &\mapsto \omega^{\mathrm{N}}(c) \mod 2^k \\ k &= \omega^{\mathrm{b}}(x) \end{aligned}$$

$$\begin{aligned} = &\mapsto = \\ x <_{\mathrm{u}} y &\mapsto x < y \\ x <_{\mathrm{s}} y &\mapsto to\_s(k,x) < to\_s(k,y) \end{aligned}$$

$$\begin{aligned} x + y &\mapsto (x+y) \mod 2^k \\ x \ll y &\mapsto (x \cdot 2^y) \mod 2^k \end{aligned}$$

$\cdot$, div, mod, $\sim$, $-$, $\gg$, $\circ$ are handled similarly

$$\begin{aligned} x \,\&\, y &\mapsto \Sigma_{i=0}^{k} 2^i \cdot \min(x[i], y[i]) \\ x \mid y &\mapsto \Sigma_{i=0}^{k} 2^i \cdot \max(x[i], y[i]) \\ x \oplus y &\mapsto \Sigma_{i=0}^{k} 2^i \cdot |x[i] - y[i]| \end{aligned}$$

$$\varphi \mapsto Tr(\varphi) \wedge \bigwedge(0 \leq x < 2^k)$$

Translation

$$Tr: \quad BV \mapsto NIA$$

$$\begin{array}{lll} x & \mapsto & x \\ c & \mapsto & \omega^{\mathrm{N}}(c) \mod 2^k \\ k = \omega^{\mathrm{b}}(x) & & \end{array}$$

$$\begin{array}{lll} = & \mapsto & = \\ x <_{\mathrm{u}} y & \mapsto & x < y \\ x <_{\mathrm{s}} y & \mapsto & to\_s(k, x) < to\_s(k, y) \end{array}$$

$$\begin{array}{lll} x + y & \mapsto & (x + y) \mod 2^k \\ x \ll y & \mapsto & (x \cdot 2^y) \mod 2^k \end{array}$$

$\cdot$, div, mod, $\sim$, $-$, $\gg$, $\circ$ are handled similarly

$$\begin{array}{lll} x \,\&\, y & \mapsto & \Sigma_{i=0}^{k} 2^i \cdot \min(x[i], y[i]) \\ x \mid y & \mapsto & \Sigma_{i=0}^{k} 2^i \cdot \max(x[i], y[i]) \\ x \oplus y & \mapsto & \Sigma_{i=0}^{k} 2^i \cdot |x[i] - y[i]| \end{array}$$

$$\varphi \mapsto Tr(\varphi) \wedge \bigwedge(0 \leq x < 2^k)$$

Translation

$$Tr: \quad BV \quad \mapsto \quad UF\,NIA$$

$$\begin{array}{lll} x & \mapsto & x \\ c & \mapsto & \omega^{\mathrm{N}}(c) \mod p2(k) \\ k = \omega^{\mathrm{b}}(x) & & \end{array}$$

$$\begin{array}{lll} = & \mapsto & = \\ x <_{\mathrm{u}} y & \mapsto & x < y \\ x <_{\mathrm{s}} y & \mapsto & to\_s(k, x) < to\_s(k, y) \end{array}$$

$$\begin{array}{lll} x + y & \mapsto & (x + y) \mod p2(k) \\ x \ll y & \mapsto & (x \cdot p2(y)) \mod p2(k) \end{array}$$

$\cdot$, div, mod, $\sim$, $-$, $\gg$, $\circ$ are handled similarly

$$\begin{array}{lll} x \,\&\, y & \mapsto & \&^{\mathbb{N}}(k, x, y) \\ x \mid y & \mapsto & |^{\mathbb{N}}(k, x, y) \\ x \oplus y & \mapsto & \oplus^{\mathbb{N}}(k, x, y) \end{array}$$

$$\varphi \quad \mapsto \quad \mathrm{Tr}(\varphi) \;\wedge\; \bigwedge(0 \leq x < p2(k)) \;\wedge\; Axioms$$

Axiomatizations

$$\varphi \quad \mapsto \quad \mathrm{Tr}(\varphi) \;\wedge\; \bigwedge(0 \leq x < p2(k)) \;\wedge\; Axioms(p2, \&^{\mathbb{N}}, |^{\mathbb{N}}, \oplus^{\mathbb{N}})$$

Axiomatization Modes

- full
- partial
- combined
- qf

Axiomatizations

$$\varphi \quad \mapsto \quad \mathrm{Tr}(\varphi) \;\wedge\; \bigwedge(0 \le x < p2(k)) \;\wedge\; \mathit{Axioms}(p2, \&^{\mathbb{N}}, |^{\mathbb{N}}, \oplus^{\mathbb{N}})$$

Axiomatization Mode: full

power of two	$p2(0) = 1 \wedge \forall k. k > 0 \Rightarrow p2(k) = 2 \cdot p2(k-1)$
bit-wise and	$\forall k, x, y.$ $k = 1 \Rightarrow \quad \&^{\mathbb{N}}(k, x, y) = \quad \min(x[0], y[0]) \wedge$ $k > 1 \Rightarrow \quad \&^{\mathbb{N}}(k, x, y) = \quad \&^{\mathbb{N}}(k-1, x[k-2:0], y[k-2:0]) +$ $p2(k-1) \cdot \min(x[k-1], y[k-1])$
bit-extraction	$x[k-2:0] \;:=\; x \mod p2(k-1)$ $x[k-1] \;:=\; (x \div p2(k-1)) \mod 2$ $x[0] \;:=\; x \mod 2$

Axiomatizations

$$\varphi \quad \mapsto \quad \mathrm{Tr}(\varphi) \;\wedge\; \bigwedge(0 \leq x < p2(k)) \;\wedge\; \mathit{Axioms}(p2, \&^{\mathbb{N}}, |^{\mathbb{N}}, \oplus^{\mathbb{N}})$$

Axiomatization Mode: partial

base cases	$p2(0) = 1 \wedge p2(1) = 2 \wedge p2(2) = 4 \wedge p2(3) = 8$
weak monotonicity	$\forall i \forall j.\, i \leq j \Rightarrow p2(i) \leq p2(j)$
strong monotonicity	$\forall i \forall j.\, i < j \Rightarrow p2(i) < p2(j)$
modularity	$\forall i \forall j \forall x.\, (x \cdot p2(i)) \mod p2(j) \neq 0 \Rightarrow i < j$
never even	$\forall i \forall x.\, p2(i) - 1 \neq 2 \cdot x$
always positive	$\forall i.\, p2(i) \geq 1$
div 0	$\forall i.\, i \div p2(i) = 0$

Axiomatizations

$$\varphi \quad \mapsto \quad \mathrm{Tr}(\varphi) \;\wedge\; \bigwedge(0 \leq x < p2(k)) \;\wedge\; Axioms(p2, \&^{\mathbb{N}}, |^{\mathbb{N}}, \oplus^{\mathbb{N}})$$

Axiomatization Mode: partial

base case	$\forall x \forall y.\, \&^{\mathbb{N}}(1, x, y) = \min(x[0], y[0])$
max	$\forall k \forall x.\, \&^{\mathbb{N}}(k, x, p2(k) - 1) = x$
min	$\forall k \forall x.\, \&^{\mathbb{N}}(k, x, 0) = 0$
idempotence	$\forall k \forall x.\, \&^{\mathbb{N}}(k, x, x) = x$
contradiction	$\forall k \forall x.\, \&^{\mathbb{N}}(k, x, p2(k) - 1 - x) = 0$
symmetry	$\forall k \forall x \forall y.\, \&^{\mathbb{N}}(k, x, y) = \&^{\mathbb{N}}(k, y, x)$
difference	$\forall k \forall x \forall y \forall z.\, x \neq y \Rightarrow \&^{\mathbb{N}}(k, x, z) \neq y \vee \&^{\mathbb{N}}(k, y, z) \neq x$
range	$\forall k \forall x \forall y.\, 0 \leq \&^{\mathbb{N}}(k, x, y) \leq \min(x, y)$

Axiomatizations

$$\varphi \quad \mapsto \quad \mathrm{Tr}(\varphi) \;\wedge\; \bigwedge(0 \leq x < p2(k)) \;\wedge\; \mathit{Axioms}(p2, \&^{\mathbb{N}}, |^{\mathbb{N}}, \oplus^{\mathbb{N}})$$

Axiomatization Mode: combined

combined = full + partial

Axiomatization Mode: qf

qf = some base cases (quantifier free)

Axiomatizations

$$\varphi \quad \mapsto \quad \mathrm{Tr}(\varphi) \;\wedge\; \bigwedge(0 \leq x < p2(k)) \;\wedge\; \mathit{Axioms}(p2, \&^{\mathbb{N}}, |^{\mathbb{N}}, \oplus^{\mathbb{N}})$$

Correctness

- full and combined translations are sound and complete.
- partial and qf translations are sound

Effectiveness

- combined $>$ partial $>$ full $>$ qf
- combined and full can be used for a SAT result
- qf can be used with more solvers

Goal:
proving validity for **every** bit-width

Express

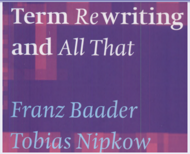
Solve

Examples

Case Studies

3 Application Domains

Invertibility Conditions



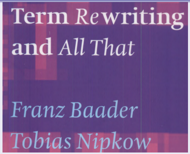
Rewriting Rules

Compiler Optimizations

Case Studies

3 Application Domains

Invertibility Conditions



Rewriting Rules

LLVM COMPILER INFRASTRUCTURE

Compiler Optimizations

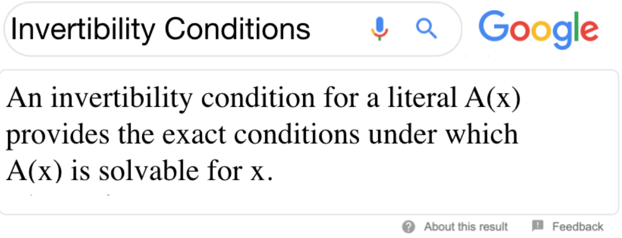
Benchmarks Generation

- Abstracted each set of problems to a parametric bit-width problem
- Translated to integers using the four approaches
- Submitted translations to SMT-LIB

Evaluation

- Participants of SMT-COMP 2018 UFNIA division: *CVC4*, *Z3*, *Vampire*
- Limits: 5 minutes run-time, 4GB memory
- **Original problems are UNSAT**

Invertibility Conditions for Bit-vectors [Niemetz et. al 2018]



Example

- $true \Leftrightarrow \exists x. x + s = t$
- $(t \neq 0 \vee s \neq 0) \Leftrightarrow \exists x. x \,\&\, s \neq t$

- We translated 160 Invertibility conditions from [Niemetz et. al 2018]
- Excluded conditions that involve $\circ$
- All were already verified up to 65 bits
- They are used for arbitrary bit-width in CVC4 for quantifier instantiation

Verifying Invertibility Conditions

Goal: Prove Validity of $IC \Leftrightarrow \exists x.\ell[x]$ for every bit-width.

$\Leftarrow$: Prove that $\exists x.\ell[x] \wedge \neg IC$ is UNSAT **QF (modulo axioms)**

$\Rightarrow$: Quantifier cannot be eliminated in the general case.

Conditional Inverses

- We used SyGuS to synthesize **conditional inverses**.
- A conditional inverse for $\ell[x]$ is a term α such that $\exists x.\ell[x] \Leftrightarrow \ell[\alpha]$
- $(\Rightarrow')$: $IC \Rightarrow \ell[\alpha]$ **Quantifier Eliminated**.
- We found 131 Conditional inverses.

Example

$$true \quad \Leftrightarrow \quad \exists x. x + s = t$$

$$(t \neq 0 \vee s \neq 0) \quad \Leftrightarrow \quad \exists x. x \,\&\, s \neq t$$

Verifying Invertibility Conditions

Goal: Prove Validity of $IC \Leftrightarrow \exists x.\ell[x]$ for every bit-width.

$\Leftarrow$: Prove that $\exists x.\ell[x] \wedge \neg IC$ is UNSAT **QF (modulo axioms)**

$\Rightarrow$: Quantifier cannot be eliminated in the general case.

Conditional Inverses

- We used SyGuS to synthesize **conditional inverses**.
- A conditional inverse for $\ell[x]$ is a term α such that $\exists x.\ell[x] \Leftrightarrow \ell[\alpha]$
- $(\Rightarrow')$: $IC \Rightarrow \ell[\alpha]$ **Quantifier Eliminated**.
- We found 131 Conditional inverses.

Example

$$(t - s) + s = t \quad \Leftrightarrow \quad \exists x.x + s = t$$
$$\sim t \; \& \; s \neq t \quad \Leftrightarrow \quad \exists x.x \; \& \; s \neq t$$

Invertibility Conditions: Results

$\ell[x]$	$=$	$\neq$	$<_u$	$>_u$	$\leq_u$	$\geq_u$	$<_s$	$>_s$	$\leq_s$	$\geq_s$
$-x \bowtie t$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$\sim x \bowtie t$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$x \mathbin{\&} s \bowtie t$	→	✓	✓	✓	✓	✓	→	→	✗	→
$x \mid s \bowtie t$	→	✓	✓	✓	✓	✓	→	✗	→	✗
$x \ll s \bowtie t$	→	←	✓	→	✓	→	→	✗	←	✗
$s \ll x \bowtie t$	✓	✓	✓	✓	✓	✓	←	✓	←	✓
$x \gg s \bowtie t$	✓	✓	✓	→	✓	✓	✓	→	✓	→
$s \gg x \bowtie t$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$x \gg_a s \bowtie t$	✗	✓	✓	✓	✓	✓	→	✓	→	✓
$s \gg_a x \bowtie t$	✓	✓	←	←	←	←	←	✗	←	✓
$x + s \bowtie t$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$x \cdot s \bowtie t$	✗	←	✓	✗	✓	✗	✗	✗	←	✗
$x \text{ div } s \bowtie t$	✓	✓	✓	✓	✓	←	✓	✓	✓	✓
$s \text{ div } x \bowtie t$	✓	←	✓	✓	✓	✓	✓	←	✓	←
$x \bmod s \bowtie t$	✓	✓	✓	✓	✓	✓	✗	✓	←	✓
$s \bmod x \bowtie t$	→	✓	✓	✓	✓	✓	✓	←	✓	←

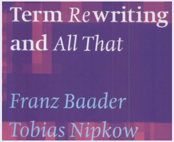
- 110 out of 160 invertibility conditions verified for any bit-width
- →: 8 were proved only when using conditional inverses
- qf mode proved 40

new 11 more conditions were proven in Coq [Ekici et al. 2019]

Rewriting Rules for Fixed-width Bit-vectors

Rewriting in Bit-vector Solvers

- Bit-vector formulas are **rewritten** before bit-blasting
- Rewrites are Implemented for arbitrary bit-width
- Their verification is crucial for soundness



Evaluation

- We synthesized ~ 2000 "Rewrite Candidates"
  - pairs $\langle A, B \rangle$ of bit-vector formulas/terms that are equivalent for bit-width 4
- Proven rewrites were added as axioms
- Fixpoint was reached after 1 round for formulas and 2 rounds for terms

	Generated	Proved
Formula	435	409
Term	1575	878 (935)

Compiler Optimizations with Alive

```
Name: AndOrXor:1733
%cmp1 = icmp ne %A, 0
%cmp2 = icmp ne %B, 0
%r = or %cmp1, %cmp2
  =>
%C = or %A, %B
%r = icmp ne %C, 0
```

$$(A \neq 0 \lor B \neq 0) \Leftrightarrow (A \mid B \neq 0)$$

- We translated 160 correctness conditions to UFNIA
- Verified 88 of them for arbitrary bit-width
- combined mode was best, qf mode was very good

Required axioms

$$\forall k \forall x. |^{\mathbb{N}}(k, 0, x) = x \qquad \forall k \forall x \forall y. \max(x, y) \leq |^{\mathbb{N}}(k, x, y)$$

Conclusion

We Have Seen

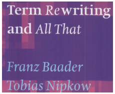
- Proving parametric bit-vector formulas is useful, and possible!
- Translation to integers + UF + quantifiers

Why Is This Possible?

- Advances in non-linear arithmetic and quantifier solving
- Features of case studies: Real & Rely on basic properties

Future Work

- Satisfiable Benchmarks
- Stronger axioms



Conclusion

We Have Seen

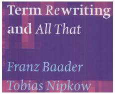
- Proving parametric bit-vector formulas is useful, and possible!
- Translation to integers + UF + quantifiers

Why Is This Possible?

- Advances in non-linear arithmetic and quantifier solving
- Features of case studies: Real & Rely on basic properties

Future Work

- Satisfiable Benchmarks
- Stronger axioms



Thank You !

Many-sorted Logic for Parametric Bit-vectors

Many-sorted First-order Logic?

- Option 1: One sort for all bit-widths $0010 + 111 = ?$ $000 \circ 00 \stackrel{?}{=} 0$
  - No type-checking $\Rightarrow$ more errors
- Option 2.0: A sort for every integer term: $\mathbf{BV}[1], \ldots, \mathbf{BV}[(2 \cdot k + 3)], \ldots$
  - Variables of sort $\mathbf{BV}[2 \cdot k]$ and $\mathbf{BV}[k + k]$ are not comparable
- Option 2: A sort for every normalized integer term: $\mathbf{BV}[1], \ldots, \mathbf{BV}[(2 \cdot k + 3), \ldots]$
  - $\mathbf{BV}[5]$ and $\mathbf{BV}[[k]]$ have disjoint domains in all interpretations