Towards Bit-Width-Independent Proofs in SMT Solvers

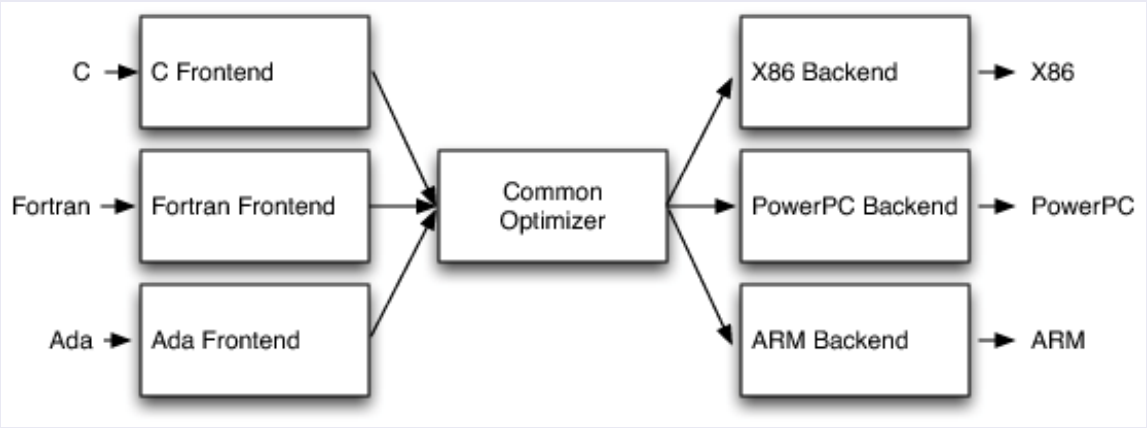
Aina Niemetz[1] Mathias Preiner[1] Andrew Reynolds[2] Yoni Zohar[1]
Clark Barrett[1] Cesare Tinelli[2]

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

```
1  Name: AddSub:1604
2  Pre: C == width(%a) - 1
3  %a = ashr %x, C
4  %r = sub 0, %a
5    =>
6  %r = lshr %x, C
```

```
1  Name: muldivrem:876
2  Pre: C u>= 1<<(width(C)-1)
3  %r = udiv %x, C
4    =>
5  %c = icmp ult %x, C
6  %r = select %c, 0, 1
```

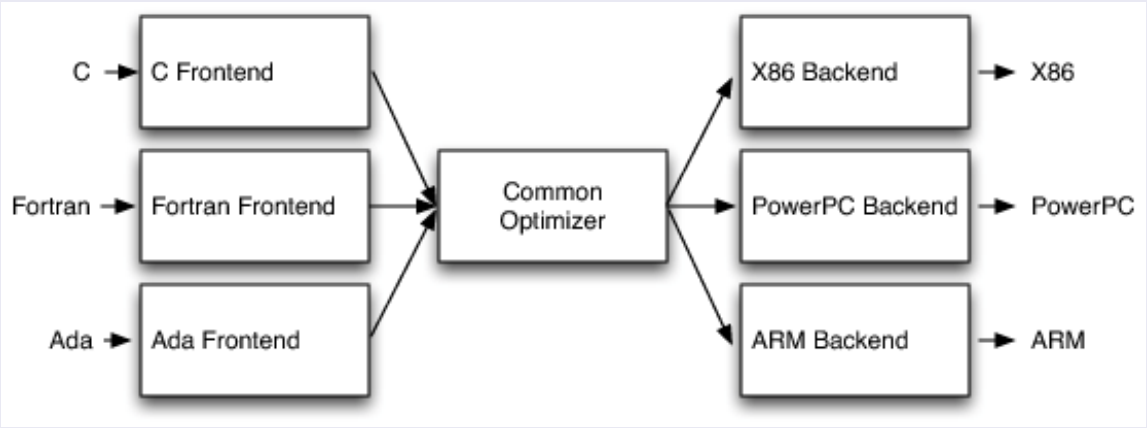
```
1  Name: AndOrXor:1733
2  %cmp1 = icmp ne %A, 0
3  %cmp2 = icmp ne %B, 0
4  %r = or %cmp1, %cmp2
5    =>
6  %C = or %A, %B
7  %r = icmp ne %C, 0
```

AndOrXor:1733 is correct IFF $(A \neq 0 \vee B \neq 0) \Leftrightarrow (A \mid B \neq 0)$ is VALID

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

```
1  Name: AddSub:1604
2  Pre: C == width(%a) - 1
3  %a = ashr %x, C
4  %r = sub 0, %a
5    =>
6  %r = lshr %x, C
```

```
1  Name: muldivrem:876
2  Pre: C u>= 1<<(width(C)-1)
3  %r = udiv %x, C
4    =>
5  %c = icmp ult %x, C
6  %r = select %c, 0, 1
```

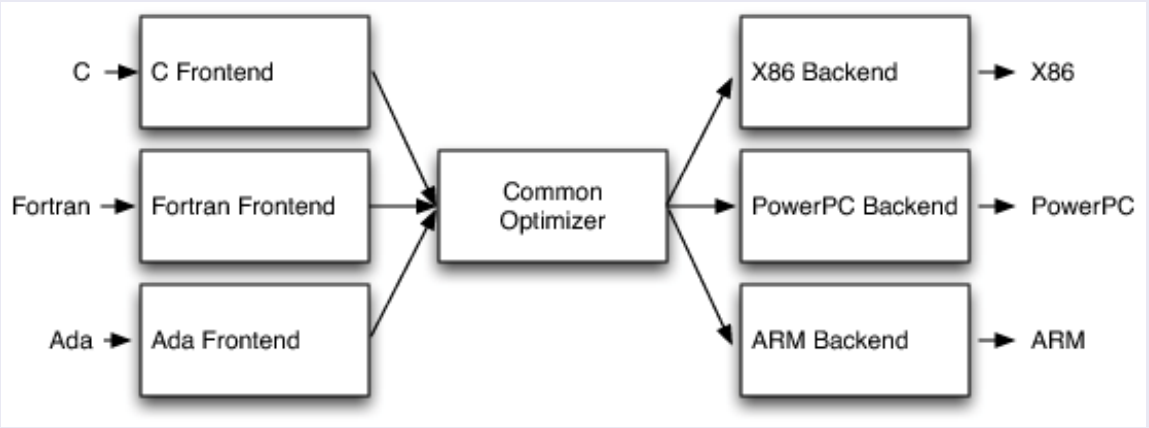
```
1  Name: AndOrXor:1733
2  %cmp1 = icmp ne %A, 0
3  %cmp2 = icmp ne %B, 0
4  %r = or %cmp1, %cmp2
5    =>
6  %C = or %A, %B
7  %r = icmp ne %C, 0
```

AndOrXor:1733 is correct IFF $(A \neq 0 \vee B \neq 0) \Leftrightarrow (A \mid B \neq 0)$ is VALID

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

```
1  Name: AddSub:1604
2  Pre: C == width(%a) - 1
3  %a = ashr %x, C
4  %r = sub 0, %a
5    =>
6  %r = lshr %x, C
```

```
1  Name: muldivrem:876
2  Pre: C u>= 1<<(width(C)-1)
3  %r = udiv %x, C
4    =>
5  %c = icmp ult %x, C
6  %r = select %c, 0, 1
```

```
1  Name: AndOrXor:1733
2  %cmp1 = icmp ne %A, 0
3  %cmp2 = icmp ne %B, 0
4  %r = or %cmp1, %cmp2
5    =>
6  %C = or %A, %B
7  %r = icmp ne %C, 0
```

AndOrXor:1733 is correct IFF $(A \neq 0 \vee B \neq 0) \Leftrightarrow (A \mid B \neq 0)$ is VALID

Our Goal: proving validity for every bit-width

Outline

Our Goal: proving validity for every bit-width

How to express?

How to solve?

Case Studies

Outline

Our Goal: proving validity for every bit-width

How to express? How to solve? Case Studies

Bit-vectors in SMT-LIB 2

Many-sorted First-order Logic

- Sorts: $\sigma_1, \sigma_2, \ldots$
- Sorted equality, functions, predicates

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

What Do We Need?

- Variables ranging over bit-vectors of unspecified bit-width
- Constants with unspecified bit-width $0 \ldots 0$

$$(x \neq_k 0 \ldots 0 \vee y \neq_k 0 \ldots 0) \Leftrightarrow (x \mid_k y \neq_k 0 \ldots 0)$$

- No such thing as "σ_k"
- Many-sorted first-order logic does not seem like a natural fit

Language for Bit-vectors of Parametric Width

Language

- Unsorted equal, functions, predicates
- Bit-vector variables: $X = \{x_1, \ldots\}$
- Bit-vector constants: $Z = \{z_1, \ldots\}$
- Integer terms: $N = \{0, n + m, \ldots\}$

Auxiliary Maps

Pair of maps: $\omega = \langle \omega^b, \omega^N \rangle$

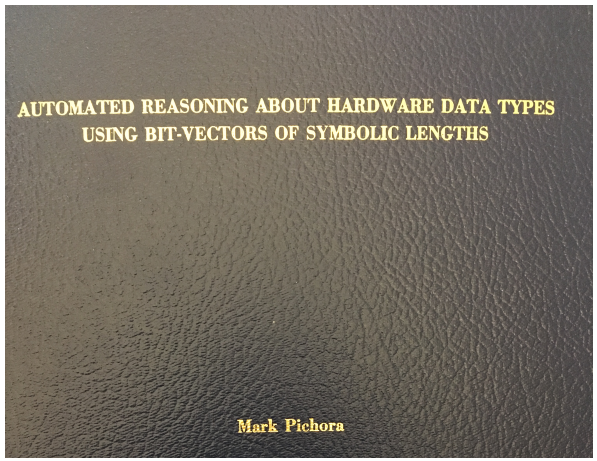
- $\omega^b : X \cup Z \to N$ symbolic bit-width
- $\omega^N : Z \to N$ symbolic value

Not every ω can be used

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

$$(x_1 \neq z_0 \vee x_2 \neq z_0) \Leftrightarrow (x_1 \mid x_2 \neq z_0)$$

$$(z_0 \ \& \ x_1) <_u x_2$$



Language for Bit-vectors of Parametric Width

Language

- Unsorted equal, functions, predicates
- Bit-vector variables: $X = \{x_1, \ldots\}$
- Bit-vector constants: $Z = \{z_1, \ldots\}$
- Integer terms: $N = \{0, n + m, \ldots\}$

Auxiliary Maps

Pair of maps: $\omega = \langle \omega^b, \omega^N \rangle$

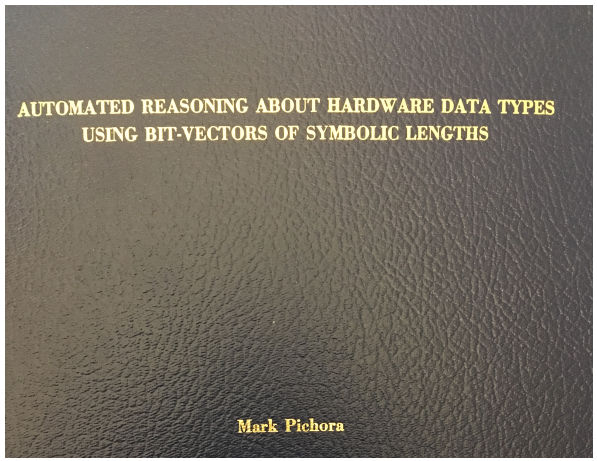
- $\omega^b : X \cup Z \to N$ symbolic bit-width
- $\omega^N : Z \to N$ symbolic value

Not every ω can be used

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

$$(x_1 \neq z_0 \vee x_2 \neq z_0) \Leftrightarrow (x_1 \mid x_2 \neq z_0)$$
with
$$\omega^b(x_1) = \omega^b(x_2) = \omega^b(z_0) = k$$
$$\omega^N(z_0) = 0$$

$$(z_0 \ \& \ x_1) <_{\mathrm{u}} x_2$$
with
$$\omega^b(x_1) = \omega^b(x_2) = \omega^b(z_0) = k$$
$$\omega^N(z_0) = k$$



Language for Bit-vectors of Parametric Width

Language

- Unsorted equal, functions, predicates
- Bit-vector variables: $X = \{x_1, \ldots\}$
- Bit-vector constants: $Z = \{z_1, \ldots\}$
- Integer terms: $N = \{0, n + m, \ldots\}$

Auxiliary Maps

Pair of maps: $\omega = \langle \omega^b, \omega^N \rangle$

- $\omega^b : X \cup Z \to N$ symbolic bit-width
- $\omega^N : Z \to N$ symbolic value

Not every ω can be used

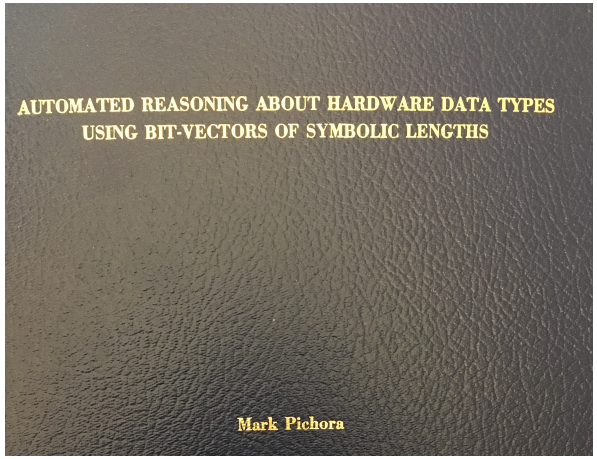
- Validity: always w.r.t. a given ω
- considering all integer interpretations
- Variant of [Pichora 2003]

$$(x_1 \neq z_0 \vee x_2 \neq z_0) \Leftrightarrow (x_1 \mid x_2 \neq z_0)$$
with
$$\omega^b(x_1) = \omega^b(x_2) = \omega^b(z_0) = k$$
$$\omega^N(z_0) = 0$$

$$(z_0 \ \& \ x_1) <_u x_2$$
with
$$\omega^b(x_1) = \omega^b(x_2) = \omega^b(z_0) = k$$
$$\omega^N(z_0) = k$$

Bad ω

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



Outline

Our Goal: proving validity for every bit-width

How to express? How to solve? Case Studies

Solving Bit-vector Formulas with Parametric Width

Possibilities

- Bit-blasting
- 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
- Need also UF and quantifiers
  - Strings with UF and quantifiers are not well-supported

Solving Bit-vector Formulas with Parametric Width

Possibilities

- ~~Bit-blasting~~
- 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
- Need also UF and quantifiers
  - Strings with UF and quantifiers are not well-supported

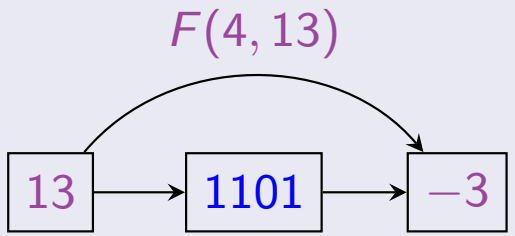
Translation

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

$$\begin{array}{lcl} x & \mapsto & x \\ z & \mapsto & \omega^N(z) \mod 2^k \end{array}$$

$$k = \omega^b(x)$$

$$\begin{array}{lcl} = & \mapsto & = \\ x <_u y & \mapsto & x < y \\ x <_s y & \mapsto & F(k,x) < F(k,y) \end{array}$$



$$F(k,x) = 2 \cdot (x \mod 2^{k-1}) - x$$

$$\begin{array}{lcl} x + y & \mapsto & (x + y) \mod 2^k \\ x \text{ div } y & \mapsto & y = 0 \ ? \ 2^k - 1 \ : \ x \div y \\ \sim x & \mapsto & 2^k - 1 - x \\ x \ll y & \mapsto & (x \cdot 2^y) \mod 2^k \\ x \circ y & \mapsto & x \cdot 2^k + y \\ x \mid y & \mapsto & \Sigma_{i=0}^{k} 2^i \cdot \max(x[i], y[i]) \end{array}$$

$$\begin{array}{lcl} x \cdot y & \mapsto & (x \cdot y) \mod 2^k \\ x \bmod y & \mapsto & y = 0 \ ? \ x \ : \ x \mod y \\ -x & \mapsto & (2^k - x) \mod 2^k \\ x \gg y & \mapsto & (x \div 2^y) \mod 2^k \\ x \ \& \ y & \mapsto & \Sigma_{i=0}^{k} 2^i \cdot \min(x[i], y[i]) \\ x \oplus y & \mapsto & \Sigma_{i=0}^{k} 2^i \cdot |x[i] - y[i]| \end{array}$$

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

Translation

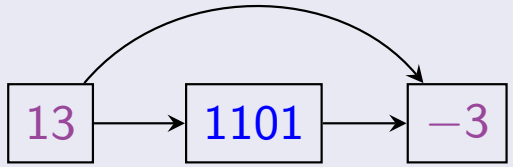
$$Tr: \quad BV \quad \mapsto \quad NIA$$

$$\begin{array}{lcl} x & \mapsto & x \\ z & \mapsto & \omega^N(z) \mod 2^k \end{array}$$

$$k = \omega^b(x)$$

$$\begin{array}{lcl} = & \mapsto & = \\ x <_u y & \mapsto & x < y \\ x <_s y & \mapsto & F(k,x) < F(k,y) \end{array}$$

$F(4, 13)$



$$F(k,x) = 2 \cdot (x \mod 2^{k-1}) - x$$

$$\begin{array}{lcl} x + y & \mapsto & (x + y) \mod 2^k \\ x \text{ div } y & \mapsto & y = 0 \ ? \ 2^k - 1 \ : \ x \div y \\ \sim x & \mapsto & 2^k - 1 - x \\ x \ll y & \mapsto & (x \cdot 2^y) \mod 2^k \\ x \circ y & \mapsto & x \cdot 2^k + y \\ x \mid y & \mapsto & \Sigma_{i=0}^{k} 2^i \cdot \max(x[i], y[i]) \end{array}$$

$$\begin{array}{lcl} x \cdot y & \mapsto & (x \cdot y) \mod 2^k \\ x \text{ mod } y & \mapsto & y = 0 \ ? \ x \ : \ x \mod y \\ -x & \mapsto & (2^k - x) \mod 2^k \\ x \gg y & \mapsto & (x \div 2^y) \mod 2^k \\ x \ \& \ y & \mapsto & \Sigma_{i=0}^{k} 2^i \cdot \min(x[i], y[i]) \\ x \oplus y & \mapsto & \Sigma_{i=0}^{k} 2^i \cdot |x[i] - y[i]| \end{array}$$

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

Translation

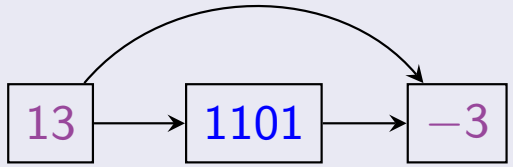
$$Tr: \quad BV \quad \mapsto \quad NIA$$

$$\begin{array}{lcl} x & \mapsto & x \\ z & \mapsto & \omega^N(z) \mod 2^k \end{array}$$

$$k = \omega^b(x)$$

$$\begin{array}{lcl} = & \mapsto & = \\ x <_u y & \mapsto & x < y \\ x <_s y & \mapsto & F(k,x) < F(k,y) \end{array}$$

$F(4, 13)$



$$F(k,x) = 2 \cdot (x \mod 2^{k-1}) - x$$

$$\begin{array}{lcl} x + y & \mapsto & (x + y) \mod 2^k \\ x \text{ div } y & \mapsto & y = 0 \ ? \ 2^k - 1 \ : \ x \div y \\ \sim x & \mapsto & 2^k - 1 - x \\ x \ll y & \mapsto & (x \cdot 2^y) \mod 2^k \\ x \circ y & \mapsto & x \cdot 2^k + y \\ x \mid y & \mapsto & \Sigma_{i=0}^{k} 2^i \cdot \max(x[i], y[i]) \end{array}$$

$$\begin{array}{lcl} x \cdot y & \mapsto & (x \cdot y) \mod 2^k \\ x \text{ mod } y & \mapsto & y = 0 \ ? \ x \ : \ x \mod y \\ -x & \mapsto & (2^k - x) \mod 2^k \\ x \gg y & \mapsto & (x \div 2^y) \mod 2^k \\ x \ \& \ y & \mapsto & \Sigma_{i=0}^{k} 2^i \cdot \min(x[i], y[i]) \\ x \oplus y & \mapsto & \Sigma_{i=0}^{k} 2^i \cdot |x[i] - y[i]| \end{array}$$

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

Translation

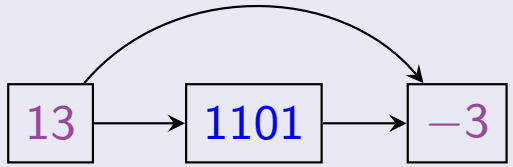
$$Tr: \quad BV \quad \mapsto \quad UFNIA$$

$$\begin{array}{lcl} x & \mapsto & x \\ z & \mapsto & \omega^N(z) \mod p2(k) \end{array}$$

$$k = \omega^b(x)$$

$$\begin{array}{lcl} = & \mapsto & = \\ x <_u y & \mapsto & x < y \\ x <_s y & \mapsto & F(k,x) < F(k,y) \end{array}$$

$F(4, 13)$



$$F(k,x) = 2 \cdot (x \mod p2(k-1)) - x$$

$$\begin{array}{lcl} x + y & \mapsto & (x + y) \mod p2(k) \\ x \text{ div } y & \mapsto & y = 0 \ ? \ p2(k) - 1 \ : \ x \div y \\ \sim x & \mapsto & p2(k) - 1 - x \\ x \ll y & \mapsto & (x \cdot p2(y)) \mod p2(k) \\ x \circ y & \mapsto & x \cdot p2(k) + y \\ x \mid y & \mapsto & |^{\mathbb{N}}(k,x,y) \end{array}$$

$$\begin{array}{lcl} x \cdot y & \mapsto & (x \cdot y) \mod p2(k) \\ x \bmod y & \mapsto & y = 0 \ ? \ x \ : \ x \mod y \\ -x & \mapsto & (p2(k) - x) \mod p2(k) \\ x \gg y & \mapsto & (x \div p2(y)) \mod p2(k) \\ x \ \& \ y & \mapsto & \&^{\mathbb{N}}(k,x,y) \\ x \oplus y & \mapsto & \oplus^{\mathbb{N}}(k,x,y) \end{array}$$

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

Axiomatizations

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

Axiomatization modes: full, partial, combined, qf

full

$p2 \quad p2(0) = 1 \wedge \forall k.k > 0 \Rightarrow p2(k) = 2 \cdot p2(k-1)$

$$\begin{array}{lll} \&^{\mathbb{N}} & \forall k, x, y. & \\ & k = 1 \Rightarrow \quad \&^{\mathbb{N}}(k, x, y) = & \min(x[0], y[0]) \wedge \\ & k > 1 \Rightarrow \quad \&^{\mathbb{N}}(k, x, y) = & \&^{\mathbb{N}}(k-1, x[k-2:0], y[k-2:0]) + \\ & & p2(k-1) \cdot \min(x[k-1], y[k-1]) \end{array}$$

$$\begin{array}{lll} x[k-2:0] & := & x \mod p2(k-1) \\ x[k-1] & := & (x \div p2(k-1)) \mod 2 \\ x[0] & := & x \mod 2 \end{array}$$

Axiomatizations

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

Axiomatization modes: full, partial, combined, qf

partial – $p2$

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

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

Axiomatization modes: full, partial, combined, qf

partial – $\&^{\mathbb{N}}$

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

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

Axiomatization modes: full, partial, combined, qf

- combined = full + partial
- qf = some base cases (quantifier free)

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

Outline

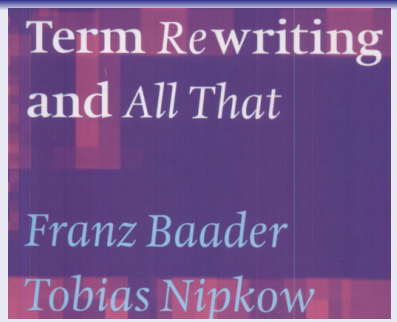
Our Goal: proving validity for every bit-width

How to express? How to solve? Case Studies

Case Studies

3 Application Domains

Invertibility Conditions



Rewriting Rules

Compiler Optimizations

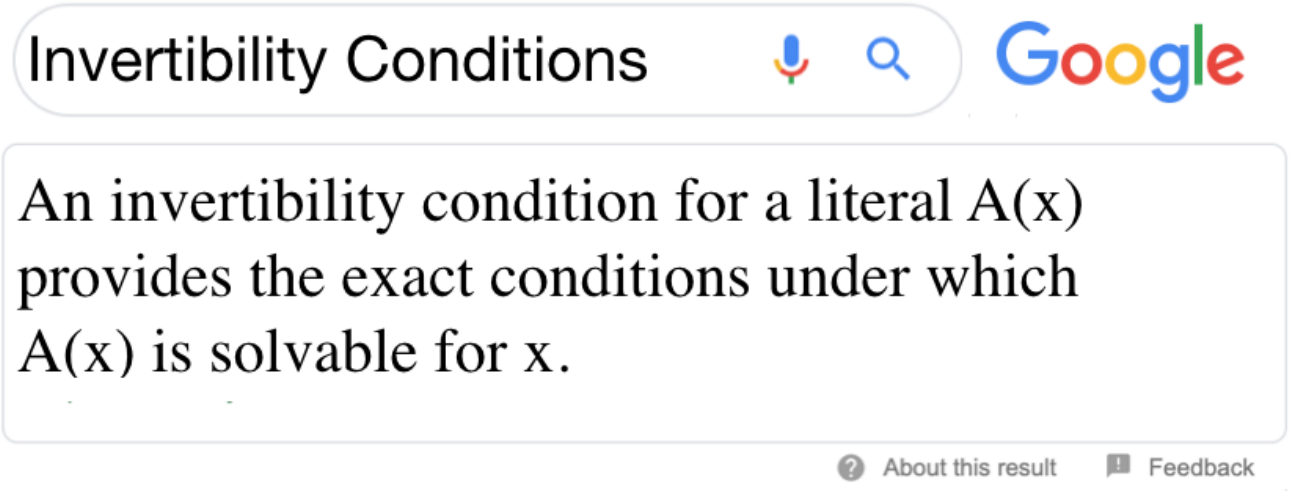
Benchmarks Generation

- Abstracted each set of problems to a parametric bit-width problem
- Translated to integers using the four approaches
- Submitted translations to SMT-COMP UFNIA and QF_UFNIA divisions

Evaluation

- Participants of SMT-COMP 2018 UFNIA division: *CVC4*, *Z3*, *Vampire*
- Limits: 5 minutes run-time, 4GB memory
- Each problem has 4 translations and 3 solvers to run with
- Original problems are UNSAT

Invertibility Conditions [Niemetz et. al 2018]



Example

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

- 160 Invertibility conditions were found in [Niemetz et. al 2018].
- Many of them were synthesized for bit-width 4.
- All were 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 Essentially 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

$$\begin{array}{lcl} true & \Leftrightarrow & \exists x.x + s = t \\ (t \neq 0 \vee s \neq 0) & \Leftrightarrow & \exists x.x \ \& \ s \neq t \end{array}$$

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 Essentially 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

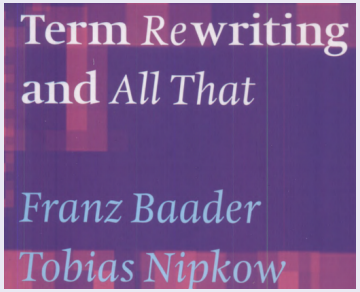
$\ell[x]$	$=$	$\neq$	$<_u$	$>_u$	$\leq_u$	$\geq_u$	$<_s$	$>_s$	$\leq_s$	$\geq_s$
$-x \bowtie t$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$\sim x \bowtie t$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$x \,\&\, s \bowtie t$	→	✓	✓	✓	✓	✓	→	→	✗	→
$x \mid s \bowtie t$	→	✓	✓	✓	✓	✓	→	✗	→	✗
$x \ll s \bowtie t$	→	←	✓	→	✓	→	→	✗	←	✗
$s \ll x \bowtie t$	✓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

- 110 out of 160 invertibility conditions verified for any bit-width
- $\Leftarrow$: 19 and $\Rightarrow$: 17
- 8 $\Rightarrow$-directions were proved only when using conditional inverses
- qf mode proved 40

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

```
1  Name: AndOrXor:1733
2  %cmp1 = icmp ne %A, 0
3  %cmp2 = icmp ne %B, 0
4  %r = or %cmp1, %cmp2
5    =>
6  %C = or %A, %B
7  %r = icmp ne %C, 0
```

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

- We translated 160 correctness conditions to UFNIA
- Verified 88 of them for every bit-width
- Axiomatization modes performed similarly

Required axioms

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

Conclusion

We Have Seen

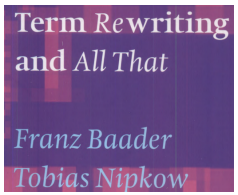
- Solving parametric bit-vector formulas is useful, and possible!
- Translation to integers + UF + quantifiers

Why Is This Possible?

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

Future Work

- Satisfiable Benchmarks
- UFNIA proofs for SMTCoq
- Stronger axioms



Conclusion

We Have Seen

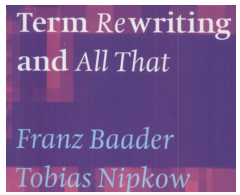
- Solving parametric bit-vector formulas is useful, and possible!
- Translation to integers + UF + quantifiers

Why Is This Possible?

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

Future Work

- Satisfiable Benchmarks
- UFNIA proofs for SMTCoq
- 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: $\sigma_1, \ldots, \sigma_{(2 \cdot k+3)}, \ldots$
  - Variables of sort $\sigma_{2 \cdot k}$ and σ_{k+k} are not comparable
- Option 2: A sort for every normalized integer term: $\sigma_1, \ldots, \sigma_{(2 \cdot k+3),\ldots}$
  - σ_5 and $\sigma_{[k]}$ have disjoint domains in all interpretations