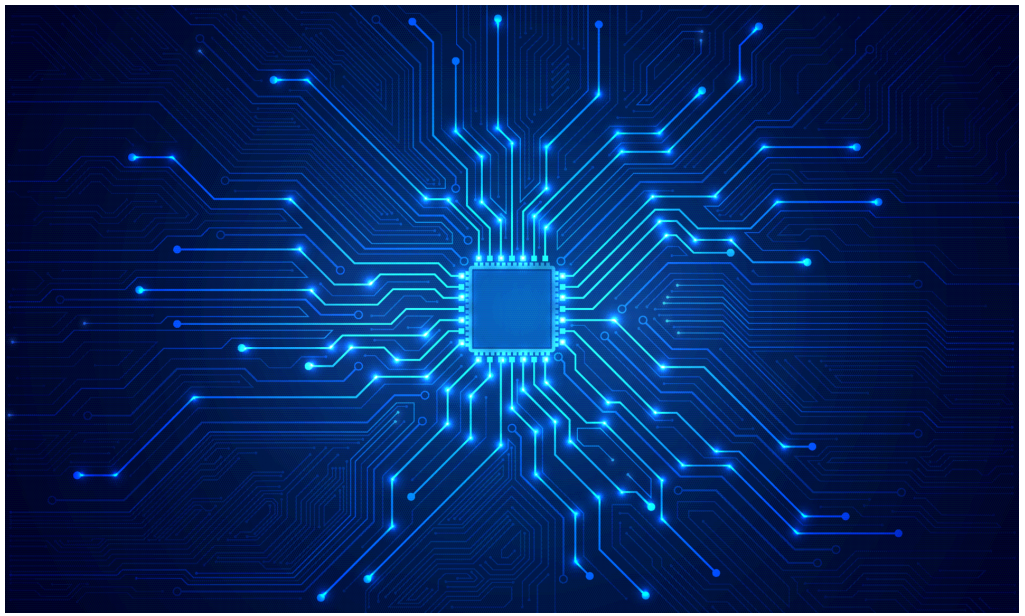
Bit-precise Reasoning with Parametric Bit-vectors

Zvika Berger	Bar-Ilan Univesity
Yoni Zohar	Bar-Ilan Univesity
Aina Niemetz	Stanford University
Mathias Preiner	Stanford University
Andrew Reynolds	The University of Iowa
Clark Barrett	Stanford University
Cesare Tinelli	The University of Iowa

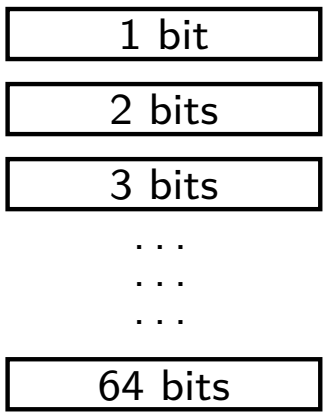


CENTAUR Annual Meeting 2025
Based on SAT'25 paper

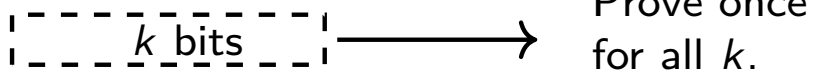
Demo

Motivation: Parametric Bit-Vectors

Fixed Bit-Width



Parametric Bit-Width



1. "Once and for all" verification
2. Encoding limitations (e.g., Rachel's talk)

Parametric Bit-Vectors

- Preliminary work was introduced in [CADE'19][1].
- Based on int-blasting.
- Limitations:
  - Theory was limited and unnatural.
  - Eager translation with quantifiers.
  - Unable to reason about bit-widths.
  - Supports only a single parametric bit-width.
  - Ad hoc evaluation (no standalone tool).

- We fix these limitations!

[1] Niemetz Aina, et al. Towards bit-width-independent proofs in SMT solvers. CADE'19.

Contributions

1. Theory
2. Solver
3. Evaluation

Outline

Parametric Bit-Vectors Theory

- $T_{PBV} = (\Sigma_{PBV}, I_{PBV})$
- Σ_{PBV} = linear arithmetic + the operators below.
- I_{PBV} = as expected; arbitrary when bit-widths don't match.

Symbol	**SMT-LIB Syntax**	**Sort**
$\approx_{\text{PBV}}$, $\not\approx_{\text{PBV}}$	`=`, `distinct`	PBV $\times$ PBV $\to$ Bool
$<_{\text{u}}$, $>_{\text{u}}$, $<_{\text{s}}$, $>_{\text{s}}$	`bvult`, `bvugt`, `bvslt`, `bvsgt`	PBV $\times$ PBV $\to$ Bool
$\leq_{\text{u}}$, $\geq_{\text{u}}$, $\leq_{\text{s}}$, $\geq_{\text{s}}$	`bvule`, `bvuge`, `bvsle`, `bvsge`	PBV $\times$ PBV $\to$ Bool
$\sim$, $-^{\mathcal{B}}$	`bvnot`, `bvneg`	PBV $\to$ PBV
&, $\mid$, $\oplus$	`bvand`, `bvor`, `bvxor`	PBV $\times$ PBV $\to$ PBV
$\ll$, $\gg$, $\gg_a$	`bvshl`, `bvlshr`, `bvashr`	PBV $\times$ PBV $\to$ PBV
$+^{\mathcal{B}}$, $-^{\mathcal{B}}$	`bvadd`, `bvsub`	PBV $\times$ PBV $\to$ PBV
$\cdot^{\mathcal{B}}$, $\text{mod}^{\mathcal{B}}$, $\text{div}^{\mathcal{B}}$	`bvmul`, `bvurem`, `bvudiv`	PBV $\times$ PBV $\to$ PBV
$\llcorner\llcorner : \llcorner\rfloor$	`pextract`	PBV $\times$ Int $\times$ Int $\to$ PBV
$\circ$	`concat`	PBV $\times$ PBV $\to$ PBV
ext_z	`pzero_extend`	Int $\times$ PBV $\to$ PBV
ext_s	`psign_extend`	Int $\times$ PBV $\to$ PBV
$\mid\llcorner\mid$	`bvsize`	PBV $\to$ Int
to-pbv	`int_to_pbv`	Int $\times$ Int $\to$ PBV

Parametric Bit-Vectors Theory

- $T_{PBV} = (\Sigma_{PBV}, I_{PBV})$
- Σ_{PBV} = linear arithmetic + the operators below.
- I_{PBV} = as expected; arbitrary when bit-widths don't match.

Symbol	**SMT-LIB Syntax**	**Sort**
$\approx_{\text{PBV}}$, $\not\approx_{\text{PBV}}$	`=`, `distinct`	PBV $\times$ PBV $\to$ Bool
$<_{\text{u}}$, $>_{\text{u}}$, $<_{\text{s}}$, $>_{\text{s}}$	`bvult`, `bvugt`, `bvslt`, `bvsgt`	PBV $\times$ PBV $\to$ Bool
$\leq_{\text{u}}$, $\geq_{\text{u}}$, $\leq_{\text{s}}$, $\geq_{\text{s}}$	`bvule`, `bvuge`, `bvsle`, `bvsge`	PBV $\times$ PBV $\to$ Bool
$\sim$, $-^{\mathcal{B}}$	`bvnot`, `bvneg`	PBV $\to$ PBV
&, $\mid$, $\oplus$	`bvand`, `bvor`, `bvxor`	PBV $\times$ PBV $\to$ PBV
$\ll$, $\gg$, $\gg_a$	`bvshl`, `bvlshr`, `bvashr`	PBV $\times$ PBV $\to$ PBV
$+^{\mathcal{B}}$, $-^{\mathcal{B}}$	`bvadd`, `bvsub`	PBV $\times$ PBV $\to$ PBV
$\cdot^{\mathcal{B}}$, $\text{mod}^{\mathcal{B}}$, $\text{div}^{\mathcal{B}}$	`bvmul`, `bvurem`, `bvudiv`	PBV $\times$ PBV $\to$ PBV
$\llcorner\llcorner : \llcorner\rfloor$	`pextract`	PBV $\times$ Int $\times$ Int $\to$ PBV
$\circ$	`concat`	PBV $\times$ PBV $\to$ PBV
ext_z	`pzero_extend`	Int $\times$ PBV $\to$ PBV
ext_s	`psign_extend`	Int $\times$ PBV $\to$ PBV
$\mid\llcorner\mid$	`bvsize`	PBV $\to$ Int
to-pbv	`int_to_pbv`	Int $\times$ Int $\to$ PBV

Parametric Bit-Vectors Theory

- $T_{PBV} = (\Sigma_{PBV}, I_{PBV})$
- Σ_{PBV} = linear arithmetic + the operators below.
- I_{PBV} = as expected; arbitrary when bit-widths don't match.

Symbol	**SMT-LIB Syntax**	**Sort**
$\approx_{\text{PBV}}$, $\not\approx_{\text{PBV}}$	`=`, `distinct`	PBV $\times$ PBV $\to$ Bool
$<_{\text{u}}$, $>_{\text{u}}$, $<_{\text{s}}$, $>_{\text{s}}$	`bvult`, `bvugt`, `bvslt`, `bvsgt`	PBV $\times$ PBV $\to$ Bool
$\leq_{\text{u}}$, $\geq_{\text{u}}$, $\leq_{\text{s}}$, $\geq_{\text{s}}$	`bvule`, `bvuge`, `bvsle`, `bvsge`	PBV $\times$ PBV $\to$ Bool
$\sim$, $-^{\mathcal{B}}$	`bvnot`, `bvneg`	PBV $\to$ PBV
&, $\mid$, $\oplus$	`bvand`, `bvor`, `bvxor`	PBV $\times$ PBV $\to$ PBV
$\ll$, $\gg$, $\gg_a$	`bvshl`, `bvlshr`, `bvashr`	PBV $\times$ PBV $\to$ PBV
$+^{\mathcal{B}}$, $-^{\mathcal{B}}$	`bvadd`, `bvsub`	PBV $\times$ PBV $\to$ PBV
$\cdot^{\mathcal{B}}$, $\text{mod}^{\mathcal{B}}$, $\text{div}^{\mathcal{B}}$	`bvmul`, `bvurem`, `bvudiv`	PBV $\times$ PBV $\to$ PBV
$\_[\_:\_]$	`pextract`	PBV $\times$ Int $\times$ Int $\to$ PBV
$\circ$	`concat`	PBV $\times$ PBV $\to$ PBV
ext_z	`pzero_extend`	Int $\times$ PBV $\to$ PBV
ext_s	`psign_extend`	Int $\times$ PBV $\to$ PBV
$\lvert\_\rvert$	`bvsize`	PBV $\to$ Int
to-pbv	`int_to_pbv`	Int $\times$ Int $\to$ PBV

Satisfiability

- $\mathcal{A}$ is a T_{PBV}-interpretation if it is in I_{PBV}
- φ is T_{PBV}-satisfiable if $\exists$ a T_{PBV}-interpretation that satisfies it
- Not enough!

Example

$$\varphi = x \approx x +^{\mathcal{B}} (x \circ x)$$

A satisfying T_{PBV}-interpretation:

- $x \mapsto 0$
- $x \circ x \mapsto 00$
- $x +^{\mathcal{B}} (x \circ x) \mapsto 0$ (different widths: arbitrary)

Standard Interpretations

Encode well-sortedness as a formula.

function $\textsc{Bw}(e)$
  match e:

x	$\to$	$\lvert x \rvert$
$t[i:j]$	$\to$	$i - j + 1$
$t_1 \circ t_2$	$\to$	$\textsc{Bw}(t_1) + \textsc{Bw}(t_2)$
$\vdots$		

end function

function $\textsc{Adm}(e)$
  match e:

x	$\to$	$\textsc{Bw}(e) > 0$
$t[i:j]$	$\to$	$0 \leq j \leq i < \textsc{Bw}(t) \wedge \textsc{Adm}(t)$
$t_1 \circ t_2$	$\to$	$\textsc{Adm}(t_1) \wedge \textsc{Adm}(t_2)$
$\bullet(t_1, \ldots, t_n)$	$\to$	$\bigwedge_{i=1}^{n} \textsc{Adm}(t_i)$
$\vdots$		

end function

Satisfiability and Admissible Satisfiability

- φ is T_{PBV}-satisfiable if $\exists$ T_{PBV}-interpretation that satisfies φ.
- φ is admissibly T_{PBV}-satisfiable if $\exists$ T_{PBV}-interpretation that satisfies $\varphi \wedge \textsc{Adm}(\varphi)$.

Example

$$\varphi = x \approx x +^{\mathcal{B}} (x \circ x)$$

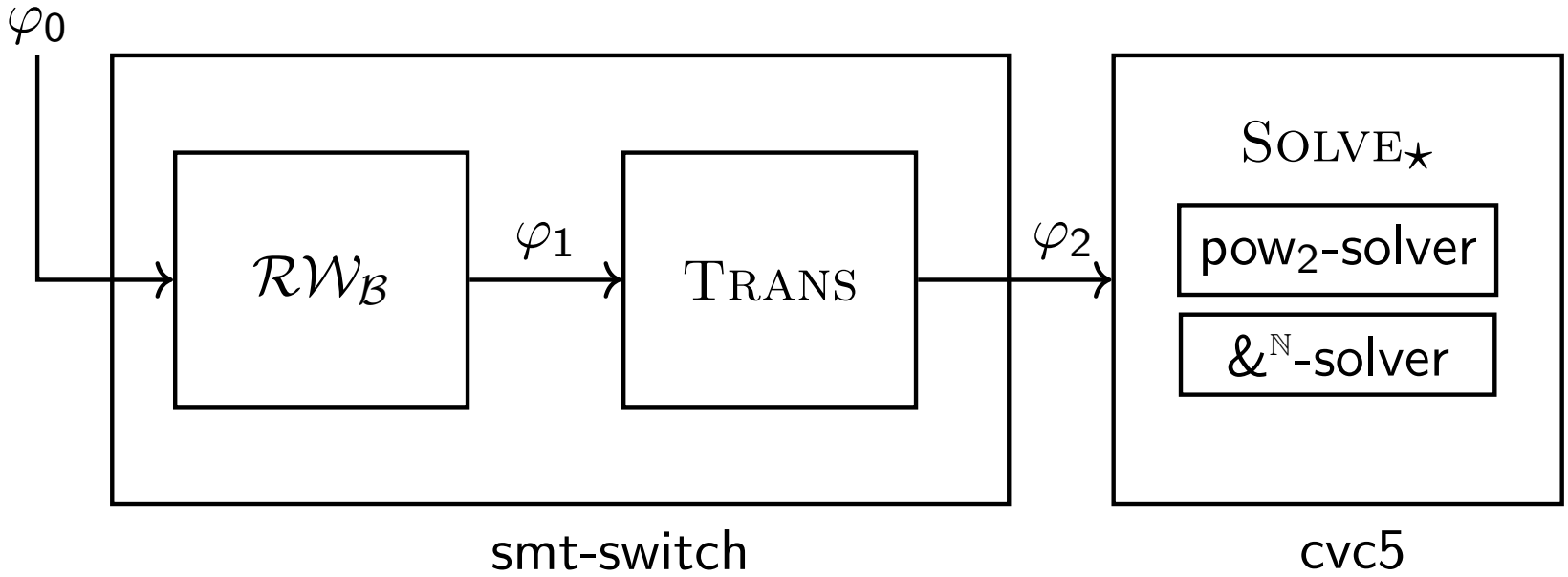
$\textsc{Adm}(\varphi)$ includes the following constraints:

$$|x| > 0 \wedge |x| \approx |x| + |x|$$

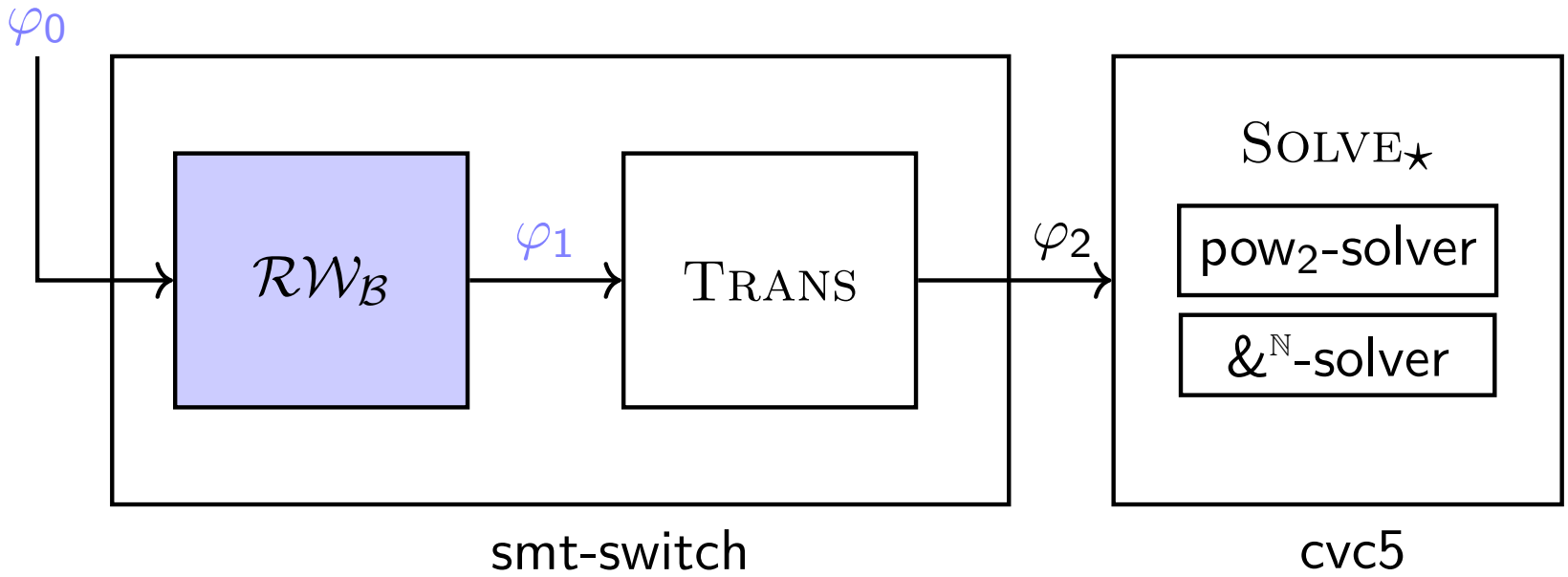
Therefore, φ is not admissibly T_{PBV}-satisfiable.

Outline

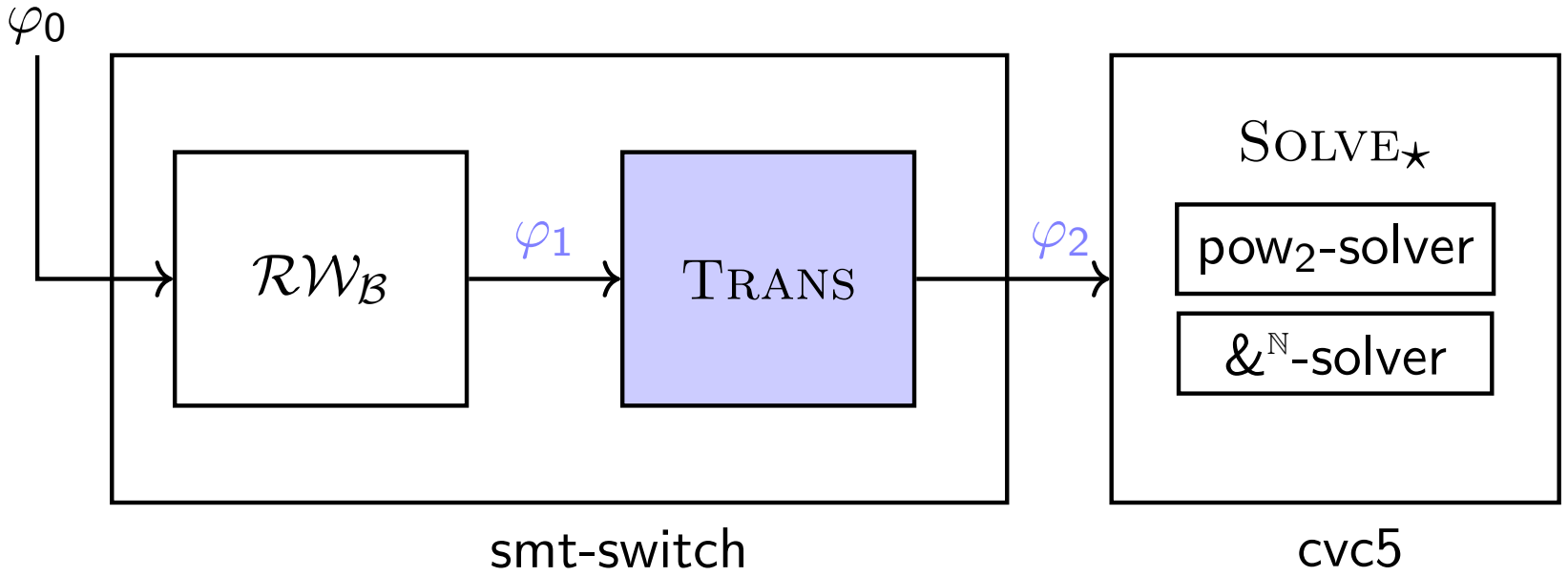
PBV Solver - Architecture



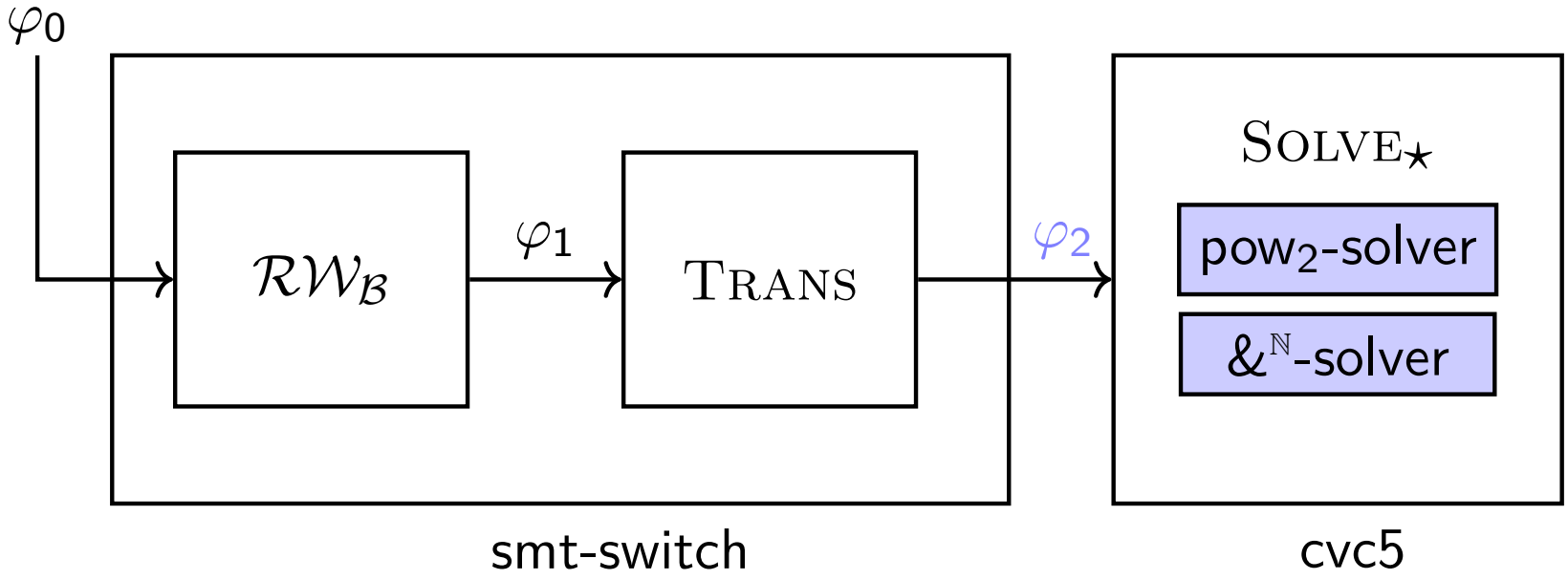
PBV Solver - Architecture



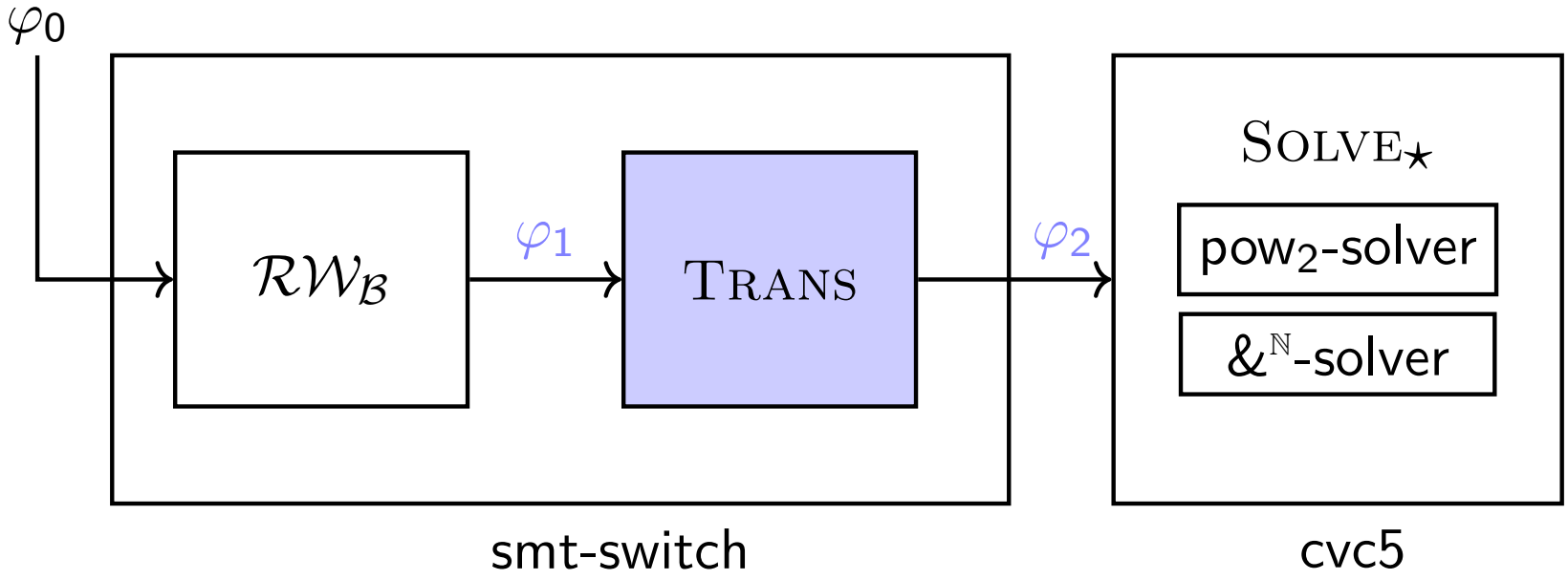
PBV Solver - Architecture



PBV Solver - Architecture



PBV Solver - Trans



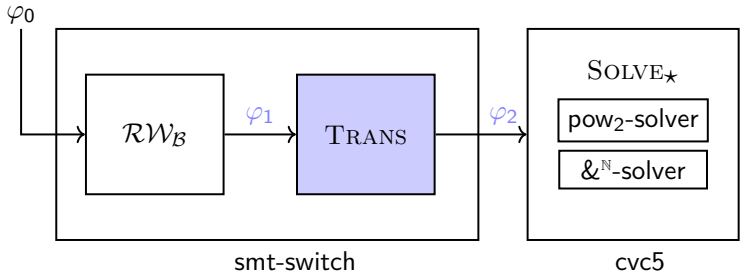
Translation Function Conv

function $\textsc{Conv}(e)$[a]

match e:

z	$\to z$
$\lvert t \rvert$	$\to \kappa(t)$
$\text{to-pbv}(k, t)$	$\to \textsc{Conv}(t) \bmod \text{pow}_2(k)$
x	$\to \chi(x)$
$t_1 \approx t_2$	$\to \textsc{Conv}(t_1) \approx \textsc{Conv}(t_2)$
$\bowtie_u (t_1, t_2)$	$\to \textsc{Conv}(t_1) \bowtie \textsc{Conv}(t_2)$
$\bowtie_s (t_1, t_2)$	$\to \text{uts}(\kappa(t_1), \textsc{Conv}(t_1)) \bowtie \text{uts}(\kappa(t_2), \textsc{Conv}(t_2))$
$t_1 +^{\mathcal{B}} t_2$	$\to (\textsc{Conv}(t_1) + \textsc{Conv}(t_2)) \bmod \text{pow}_2(\kappa(t_1))$
$t_1 -^{\mathcal{B}} t_2$	$\to (\textsc{Conv}(t_1) - \textsc{Conv}(t_2)) \bmod \text{pow}_2(\kappa(t_1))$
$t_1 \cdot^{\mathcal{B}} t_2$	$\to (\textsc{Conv}(t_1) \cdot \textsc{Conv}(t_2)) \bmod \text{pow}_2(\kappa(t_1))$
$t_1 \,\text{div}^{\mathcal{B}}\, t_2$	$\to \mathit{ite}(\textsc{Conv}(t_2) \approx 0, \text{pow}_2(\kappa(t_1)) - 1, \textsc{Conv}(t_1) \text{ div } \textsc{Conv}(t_2))$
$t_1 \,\text{mod}^{\mathcal{B}}\, t_2$	$\to \mathit{ite}(\textsc{Conv}(t_2) \approx 0, \textsc{Conv}(t_1), \textsc{Conv}(t_1) \bmod \textsc{Conv}(t_2))$
$\sim t$	$\to \text{pow}_2(\kappa(t)) - (\textsc{Conv}(t) + 1)$
$-^{\mathcal{B}} t$	$\to (\text{pow}_2(\kappa(t)) - \textsc{Conv}(t)) \bmod \text{pow}_2(\kappa(t))$
$t_1 \ll t_2$	$\to (\textsc{Conv}(t_1) \cdot \text{pow}_2(\textsc{Conv}(t_2))) \bmod \text{pow}_2(\kappa(t_1))$
$t_1 \gg t_2$	$\to \textsc{Conv}(t_1) \text{ div } \text{pow}_2(\textsc{Conv}(t_2))$
$t_1 \,\&\, t_2$	$\to \&^{\mathbb{N}}(\kappa(t_1), \textsc{Conv}(t_1), \textsc{Conv}(t_2))$
$t_1 \mid t_2$	$\to \textsc{Conv}(t_1) + \textsc{Conv}(t_2) - \&^{\mathbb{N}}(\kappa(t_1), \textsc{Conv}(t_1), \textsc{Conv}(t_2))$
$t_1 \oplus t_2$	$\to \textsc{Conv}(t_1) + \textsc{Conv}(t_2) - 2 \cdot \&^{\mathbb{N}}(\kappa(t_1), \textsc{Conv}(t_1), \textsc{Conv}(t_2))$
$t_1 \circ t_2$	$\to \textsc{Conv}(t_1) \cdot \text{pow}_2(\kappa(t_2)) + \textsc{Conv}(t_2)$
$t[i:j]$	$\to (\textsc{Conv}(t) \text{ div } \text{pow}_2(j)) \bmod \text{pow}_2(i - j + 1)$
$\mathit{ext}_z(n, t)$	$\to \textsc{Conv}(t)$
$\mathit{ext}_s(n, t)$	$\to \mathit{ite}(\textsc{Conv}(t[\kappa(t) - 1]) \approx 1, (\text{pow}_2(n) - 1) \cdot \text{pow}_2(\kappa(t)) + \textsc{Conv}(t), \textsc{Conv}(t))$
$\bullet(t_1, \ldots, t_n)$	$\to \bullet(\textsc{Conv}(t_1), \ldots, \textsc{Conv}(t_n))$

end function



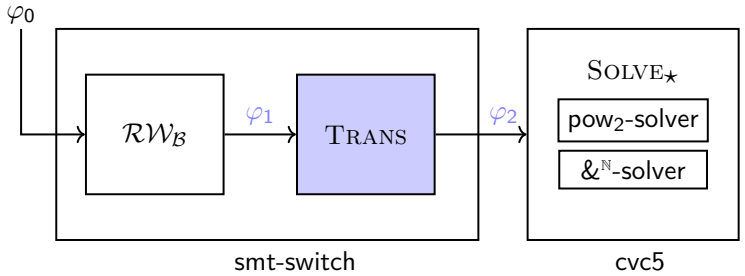
[a] Based on CADE'19. Changes in blue.

Translation Function Conv

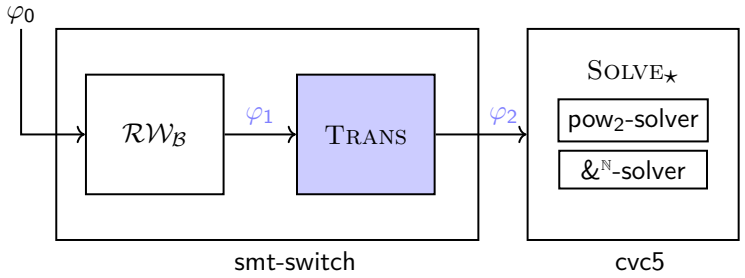
function $\textsc{Conv}(e)$
match e:

$$
\begin{array}{ll}
z & \to z \\
|t| & \to \kappa(t) \\
\text{to-pbv}(k, t) & \to \textsc{Conv}(t) \bmod \text{pow}_2(k) \\
x & \to \chi(x) \\
t_1 \approx t_2 & \to \textsc{Conv}(t_1) \approx \textsc{Conv}(t_2) \\
\bowtie_u (t_1, t_2) & \to \textsc{Conv}(t_1) \bowtie \textsc{Conv}(t_2) \\
\bowtie_s (t_1, t_2) & \to \text{uts}(\kappa(t_1), \textsc{Conv}(t_1)) \bowtie \text{uts}(\kappa(t_2), \textsc{Conv}(t_2)) \\
t_1 +^{\mathcal{B}} t_2 & \to (\textsc{Conv}(t_1) + \textsc{Conv}(t_2)) \bmod \text{pow}_2(\kappa(t_1)) \\
t_1 \text{div}^{\mathcal{B}} t_2 & \to ite(\textsc{Conv}(t_2) \approx 0, \text{pow}_2(\kappa(t_1)) - 1, \textsc{Conv}(t_1) \text{ div } \textsc{Conv}(t_2)) \\
t_1 \bmod^{\mathcal{B}} t_2 & \to ite(\textsc{Conv}(t_2) \approx 0, \textsc{Conv}(t_1), \textsc{Conv}(t_1) \bmod \textsc{Conv}(t_2)) \\
\sim t & \to \text{pow}_2(\kappa(t)) - (\textsc{Conv}(t) + 1) \\
-^{\mathcal{B}} t & \to (\text{pow}_2(\kappa(t)) - \textsc{Conv}(t)) \bmod \text{pow}_2(\kappa(t)) \\
t_1 \ll t_2 & \to (\textsc{Conv}(t_1) \cdot \text{pow}_2(\textsc{Conv}(t_2))) \bmod \text{pow}_2(\kappa(t_1)) \\
t_1 \gg t_2 & \to \textsc{Conv}(t_1) \text{ div } \text{pow}_2(\textsc{Conv}(t_2)) \\
t_1 \,\&\, t_2 & \to \&^{\mathbb{N}}(\kappa(t_1), \textsc{Conv}(t_1), \textsc{Conv}(t_2)) \\
t_1 \mid t_2 & \to \textsc{Conv}(t_1) + \textsc{Conv}(t_2) - \&^{\mathbb{N}}(\kappa(t_1), \textsc{Conv}(t_1), \textsc{Conv}(t_2)) \\
t_1 \oplus t_2 & \to \textsc{Conv}(t_1) + \textsc{Conv}(t_2) - 2 \cdot \&^{\mathbb{N}}(\kappa(t_1), \textsc{Conv}(t_1), \textsc{Conv}(t_2)) \\
t_1 \circ t_2 & \to \textsc{Conv}(t_1) \cdot \text{pow}_2(\kappa(t_2)) + \textsc{Conv}(t_2) \\
t[i:j] & \to (\textsc{Conv}(t) \text{ div } \text{pow}_2(j)) \bmod \text{pow}_2(i - j + 1) \\
ext_z(n, t) & \to \textsc{Conv}(t) \\
ext_s(n, t) & \to ite(\textsc{Conv}(t[\kappa(t) - 1]) \approx 1, (\text{pow}_2(n) - 1) \cdot \text{pow}_2(\kappa(t)) + \textsc{Conv}(t), \textsc{Conv}(t)) \\
\bullet(t_1, \ldots, t_n) & \to \bullet(\textsc{Conv}(t_1), \ldots, \textsc{Conv}(t_n))
\end{array}
$$

end function



Translation Function Conv



function Conv(e)
  match e:

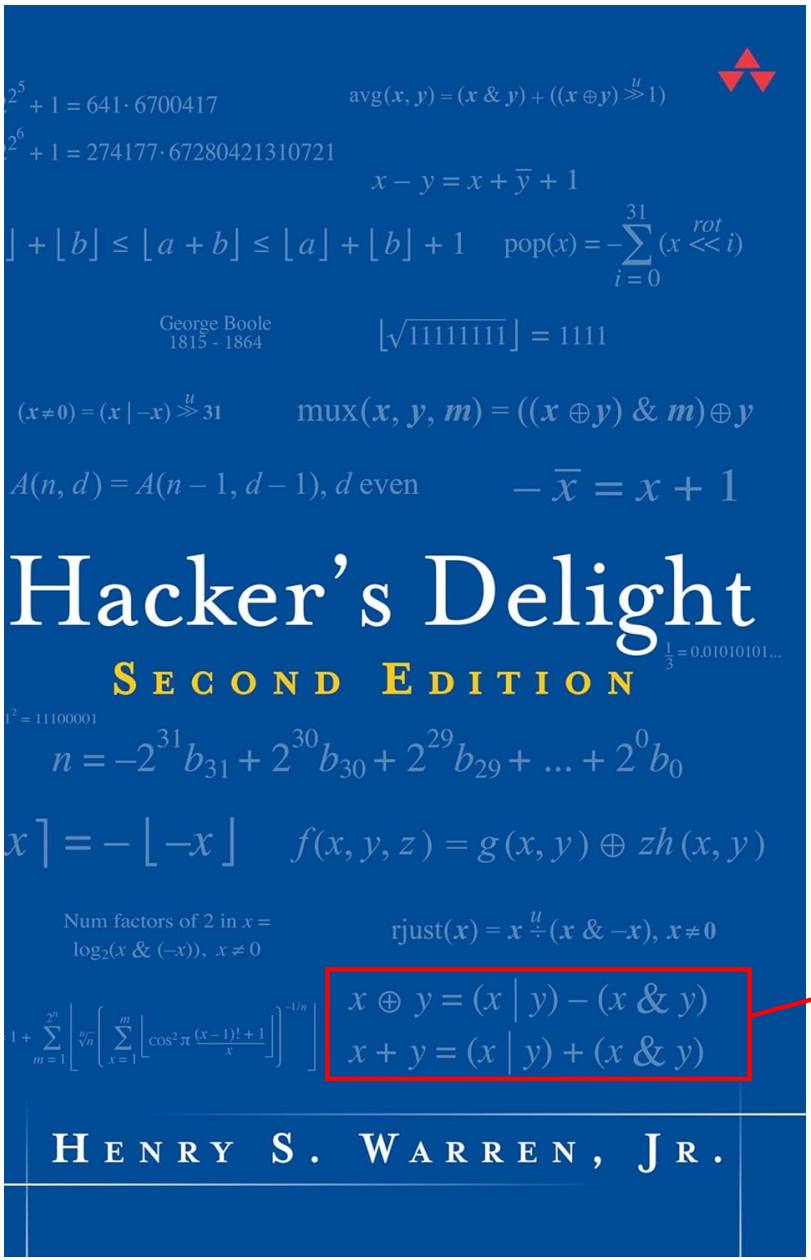
$$
\begin{array}{ll}
z & \to z \\
|t| & \to \kappa(t) \\
\text{to-pbv}(k, t) & \to \textsc{Conv}(t) \bmod \text{pow}_2(k) \\
x & \to \chi(x) \\
t_1 \approx t_2 & \to \textsc{Conv}(t_1) \approx \textsc{Conv}(t_2) \\
\bowtie_u (t_1, t_2) & \to \textsc{Conv}(t_1) \bowtie \textsc{Conv}(t_2) \\
\bowtie_s (t_1, t_2) & \to \text{uts}(\kappa(t_1), \textsc{Conv}(t_1)) \bowtie \text{uts}(\kappa(t_2), \textsc{Conv}(t_2)) \\
t_1 +^{\mathcal{B}} t_2 & \to (\textsc{Conv}(t_1) + \textsc{Conv}(t_2)) \bmod \text{pow}_2(\kappa(t_1)) \\
\end{array}
$$

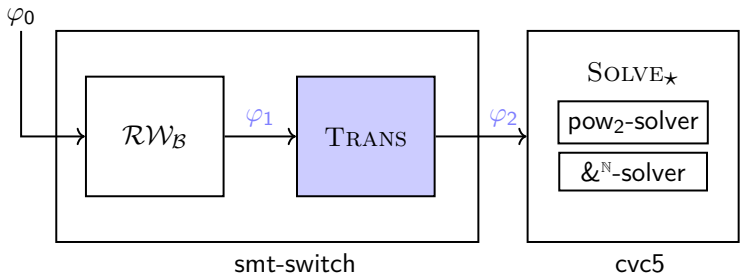
$$
\begin{array}{ll}
t_1 \ \& \ t_2 & \to \ \&^{\mathbb{N}}(\kappa(t_1), \textsc{Conv}(t_1), \textsc{Conv}(t_2)) \\
t_1 \mid t_2 & \to \ \textsc{Conv}(t_1) + \textsc{Conv}(t_2) - \&^{\mathbb{N}}(\kappa(t_1), \textsc{Conv}(t_1), \textsc{Conv}(t_2)) \\
t_1 \oplus t_2 & \to \ \textsc{Conv}(t_1) + \textsc{Conv}(t_2) - 2 \cdot \&^{\mathbb{N}}(\kappa(t_1), \textsc{Conv}(t_1), \textsc{Conv}(t_2))
\end{array}
$$

$$
\begin{array}{ll}
-^{\mathcal{B}} t & \to (\text{pow}_2(\kappa(t)) - \textsc{Conv}(t)) \bmod \text{pow}_2(\kappa(t)) \\
t_1 \ll t_2 & \to (\textsc{Conv}(t_1) \cdot \text{pow}_2(\textsc{Conv}(t_2))) \bmod \text{pow}_2(\kappa(t_1)) \\
t_1 \gg t_2 & \to \textsc{Conv}(t_1) \text{ div } \text{pow}_2(\textsc{Conv}(t_2)) \\
t_1 \ \& \ t_2 & \to \ \&^{\mathbb{N}}(\kappa(t_1), \textsc{Conv}(t_1), \textsc{Conv}(t_2)) \\
t_1 \mid t_2 & \to \ \textsc{Conv}(t_1) + \textsc{Conv}(t_2) - \&^{\mathbb{N}}(\kappa(t_1), \textsc{Conv}(t_1), \textsc{Conv}(t_2)) \\
t_1 \oplus t_2 & \to \ \textsc{Conv}(t_1) + \textsc{Conv}(t_2) - 2 \cdot \&^{\mathbb{N}}(\kappa(t_1), \textsc{Conv}(t_1), \textsc{Conv}(t_2)) \\
t_1 \circ t_2 & \to \textsc{Conv}(t_1) \cdot \text{pow}_2(\kappa(t_2)) + \textsc{Conv}(t_2) \\
t[i:j] & \to (\textsc{Conv}(t) \text{ div } \text{pow}_2(j)) \bmod \text{pow}_2(i - j + 1) \\
ext_z(n, t) & \to \textsc{Conv}(t) \\
ext_s(n, t) & \to ite(\textsc{Conv}(t[\kappa(t) - 1]) \approx 1, (\text{pow}_2(n) - 1) \cdot \text{pow}_2(\kappa(t)) + \textsc{Conv}(t), \textsc{Conv}(t)) \\
\bullet(t_1, \ldots, t_n) & \to \bullet(\textsc{Conv}(t_1), \ldots, \textsc{Conv}(t_n))
\end{array}
$$

end function

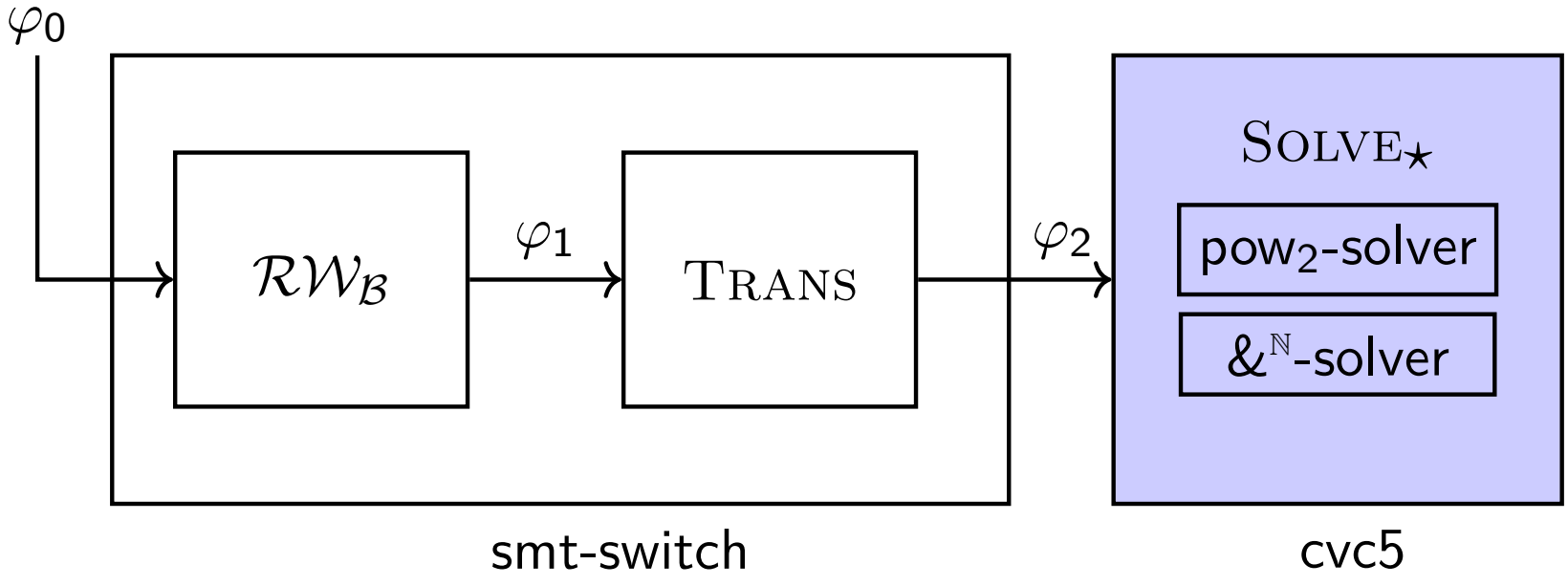
Improved Bitwise-Or Translation



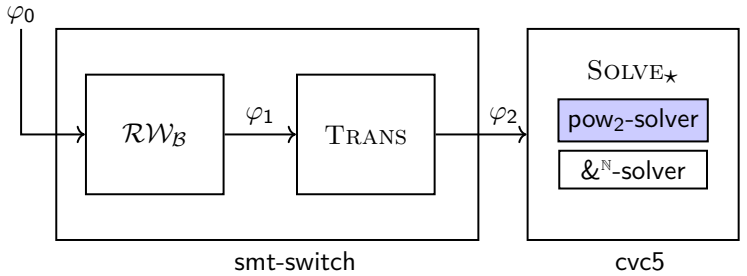


$$x \oplus y = (x \mid y) - (x \,\&\, y)$$
$$x + y = (x \mid y) + (x \,\&\, y)$$

PBV Solver - $\textsc{Solve}_\star$

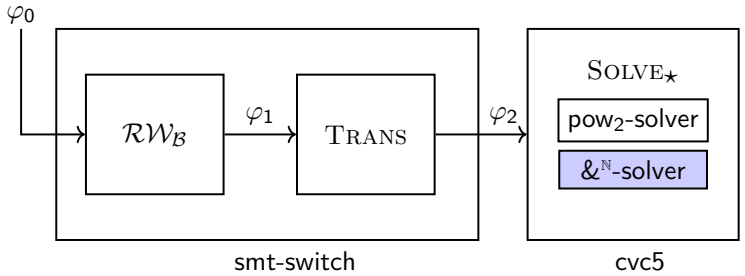


pow_2-solver



Name	**Lemma**	**Source**
	$\text{pow}_2(x) \in S$	
positive	$0 \leq x \Rightarrow \text{pow}_2(x) > 0$	[CADE'19]
even	$x \geq 1 \Rightarrow \text{pow}_2(x) \bmod 2 \approx 0$	[CADE'19]
div	$x \geq 0 \Rightarrow x \text{ div } \text{pow}_2(x) \approx 0$	[CADE'19]
neg	$x < 0 \Rightarrow \text{pow}_2(x) \approx 0$	new
bound	$(x \geq v \wedge v \geq 7) \Rightarrow v \cdot x + v^2 < \text{pow}_2(x)$	new
value	$(0 \leq x \wedge x \approx v) \Rightarrow \text{pow}_2(x) \approx 2^v$	new
	$\text{pow}_2(x), \text{pow}_2(y) \in S$	
monotonicity	$(0 \leq x \wedge x < y) \Rightarrow \text{pow}_2(x) < \text{pow}_2(y)$	[CADE'19]

$\&^{\mathbb{N}}$-solver



Name	Lemma	Source
	$\&^{\mathbb{N}}(k,x,y) \in S$	
base	$(k > 0 \wedge x \approx 1 \wedge y \approx 1) \Rightarrow \&^{\mathbb{N}}(k,x,y) \approx 1$	[CADE'19]
max	$(k > 0 \wedge \langle x \rangle_k \wedge y \approx \mathsf{pow}_2(k) - 1) \Rightarrow \&^{\mathbb{N}}(k,x,y) \approx x$	[CADE'19]
min	$y \approx 0 \Rightarrow \&^{\mathbb{N}}(k,x,y) \approx 0$	[CADE'19]
idem	$(k > 0 \wedge \langle x \rangle_k \wedge x \approx y) \Rightarrow \&^{\mathbb{N}}(k,x,y) \approx x$	[CADE'19]
contra	$(x + y) \bmod \mathsf{pow}_2(k) \approx \mathsf{pow}_2(k) - 1 \Rightarrow \&^{\mathbb{N}}(k,x,y) \approx 0$	[CADE'19]
range	$0 \leq x \wedge 0 \leq y \Rightarrow 0 \leq \&^{\mathbb{N}}(k,x,y) \leq \mathsf{min}(x,y)$	[CADE'19]
empty	$k \leq 0 \Rightarrow \&^{\mathbb{N}}(k,x,y) \approx 0$	new
lsb	$x \bmod 2 \approx 0 \Rightarrow \&^{\mathbb{N}}(k,x,y) \bmod 2 \approx 0$	new
one	$(k > 0 \wedge y \approx 1) \Rightarrow \&^{\mathbb{N}}(k,x,y) \approx x \bmod 2$	new
sum$_{\geq}$	$(k \geq v \wedge v > 0 \wedge \langle x \rangle_v \wedge \langle y \rangle_v) \Rightarrow \&^{\mathbb{N}}(k,x,y) \approx \sum_{i=0}^{v-1} \mathsf{ex}_i(x) \cdot \mathsf{ex}_i(y) \cdot 2^i$	new
	$\&^{\mathbb{N}}(k,x,z) \in S,\ \&^{\mathbb{N}}(k,y,w) \in S$	
sym	$(x \approx w \wedge y \approx z) \Rightarrow \&^{\mathbb{N}}(k,x,y) \approx \&^{\mathbb{N}}(k,z,w)$	[CADE'19]
diff	$(k > 0 \wedge z \approx w \wedge x \not\approx y \wedge \langle x \rangle_k \wedge \langle y \rangle_k \wedge \langle z \rangle_k) \Rightarrow (\&^{\mathbb{N}}(k,x,z) \not\approx y \vee \&^{\mathbb{N}}(k,y,w) \not\approx x)$	[CADE'19]

Outline

Benchmark Sets

Compiler Optimizations

- *alive* (200 benchmarks)[2]

Crafted benchmarks

- *rewrite* (2006 benchmarks)[4]
- *syrew* (1500 benchmarks)[3]

SMT solvers internals

- *lemmas* (70 benchmarks)[5]
- *ic* (180 benchmarks)[4]
- *icfb* (46 benchmarks)[5]

Mutated Benchmarks

- *mut* (9442 benchmarks):

[4] Niemetz Aina, et al. Towards bit-width-independent proofs in SMT solvers. CADE'19.

[5] Niemetz Aina, et al. Scalable bit-blasting with abstractions. CAV'24.

[6] Niemetz Aina, et al. On solving quantified bit-vector constraints using invertibility conditions. CAV'18

[7] Niemetz Aina, et al. Ternary Propagation-Based Local Search for More Bit-Precise Reasoning, FMCAD'20.

Evaluation Configuration

Attribute	BASELINE	EAGER	PBV
Multiple Bit-widths	✗	✓	✓
Lazy pow_2	✗	✗	✓
Lazy $\&^{\mathbb{N}}$	✗	✗	✓
$\mid$*-elimination*	✗	✓	✓
$\oplus$*-elimination*	✗	✓	✓
$\gg$ *without* mod	✗	✓	✓
New lemmas for $\&^{\mathbb{N}}$	✗	✓	✓
New lemmas for pow_2	✗	✓	✓
$\mathcal{RW}_{\mathcal{B}}$	✗	✓	✓

Evaluation Configuration

Attribute	BASELINE	EAGER	PBV
Multiple Bit-widths	✗	✓	✓
Lazy pow_2	✗	✗	✓
Lazy $\&^{\mathbb{N}}$	✗	✗	✓
$\lvert$*-elimination*	✗	✓	✓
$\oplus$*-elimination*	✗	✓	✓
$\gg$ *without* mod	✗	✓	✓
New lemmas for $\&^{\mathbb{N}}$	✗	✓	✓
New lemmas for pow_2	✗	✓	✓
$\mathcal{RW}_{\mathcal{B}}$	✗	✓	✓

Evaluation Configuration

Attribute	BASELINE	EAGER	PBV
Multiple Bit-widths	✗	✓	✓
Lazy pow_2	✗	✗	✓
Lazy $\&^{\mathbb{N}}$	✗	✗	✓
$\mid$*-elimination*	✗	✓	✓
$\oplus$*-elimination*	✗	✓	✓
$\gg$ *without* mod	✗	✓	✓
New lemmas for $\&^{\mathbb{N}}$	✗	✓	✓
New lemmas for pow_2	✗	✓	✓
$\mathcal{RW}_{\mathcal{B}}$	✗	✓	✓

Evaluation Configuration

Attribute	BASELINE	EAGER	PBV
Multiple Bit-widths	✗	✓	✓
Lazy pow_2	✗	✗	✓
Lazy $\&^{\mathbb{N}}$	✗	✗	✓
$\mid$*-elimination*	✗	✓	✓
$\oplus$*-elimination*	✗	✓	✓
$\gg$ *without* mod	✗	✓	✓
New lemmas for $\&^{\mathbb{N}}$	✗	✓	✓
New lemmas for pow_2	✗	✓	✓
$\mathcal{RW}_{\mathcal{B}}$	✗	✓	✓

Evaluation Configuration

Attribute	BASELINE	EAGER	PBV
Multiple Bit-widths	✗	✓	✓
Lazy pow_2	✗	✗	✓
Lazy $\&^{\mathbb{N}}$	✗	✗	✓
$\lvert$*-elimination*	✗	✓	✓
$\oplus$*-elimination*	✗	✓	✓
$\gg$ *without* mod	✗	✓	✓
New lemmas for $\&^{\mathbb{N}}$	✗	✓	✓
New lemmas for pow_2	✗	✓	✓
$\mathcal{RW}_{\mathcal{B}}$	✗	✓	✓

Evaluation Configuration

Attribute	BASELINE	EAGER	PBV
Multiple Bit-widths	✗	✓	✓
Lazy pow_2	✗	✗	✓
Lazy $\&^{\mathbb{N}}$	✗	✗	✓
$\mid$*-elimination*	✗	✓	✓
$\oplus$*-elimination*	✗	✓	✓
$\gg$ *without* mod	✗	✓	✓
New lemmas for $\&^{\mathbb{N}}$	✗	✓	✓
New lemmas for pow_2	✗	✓	✓
$\mathcal{RW}_{\mathcal{B}}$	✗	✓	✓

Evaluation Results

Benchmarks	#	BASELINE	EAGER	PBV
alive	200	71	93	**107**
ic	180	43	58	**77**
rewrite	2006	658	1221	**1331**
syrew	1500	558	720	**912**
lemmas	70	12	14	**23**
icfb	46	1	9	**12**
mut	9441	669	1084	**4863**
sat		0	0	**3641**
unsat		2012	3199	**3684**
total	13443	2012	3199	**7325**

Contributions on Benchmark Sets

- Improvements over [CADE'19]:
  - 45 benchmarks in *alive*.
  - 12 benchmarks in *ic*.
- First parametric verification of *lemmas* and *icfb*:
  - 23 benchmarks in *lemmas*.
  - 12 benchmarks in *icfb*.
  - Found a known typo in *icfb* from [FMCAD'20].

Summary and Future Work

- **Summary:**
  - New theory for parametric bit-vectors
  - Lazy algorithm for parametric bit-vectors
  - Major improvement in performance
- **Future Work:**
  - Improve modular reasoning in cvc5 (Liza's talk!)
  - cvc5 integration + proof production
  - Complete Isabelle integration

Summary and Future Work

- **Summary:**
  - New theory for parametric bit-vectors
  - Lazy algorithm for parametric bit-vectors
  - Major improvement in performance

- **Future Work:**
  - Improve modular reasoning in cvc5 (Liza's talk!)
  - cvc5 integration + proof production
  - Complete Isabelle integration

Thank you