PolySAT
A Word-level Solver for Large Bitvectors

Jakob Rath[1] Nikolaj Bjørner[2] Laura Kovács[1]
Clemens Eisenhofer[1] Daniela Kaufmann[1]

[1]TU Wien

[2]Microsoft Research

Background: Satisfiability Modulo Theories (SMT)

Problem Statement:

Is φ satisfiable?

where φ is a formula in classical first-order logic with equality and certain theories (e.g., fragments of integer arithmetic).

Background: Satisfiability Modulo Theories (SMT)

Problem Statement:

Is φ satisfiable?

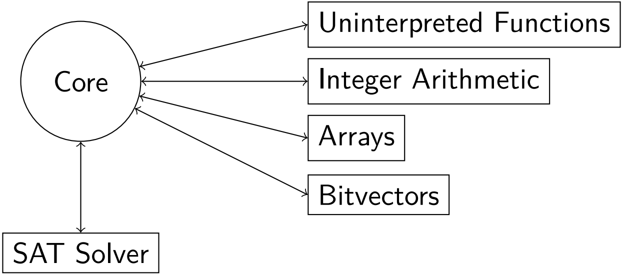
where φ is a formula in classical first-order logic with equality and certain theories (e.g., fragments of integer arithmetic).

Example

$$f(x+1) \neq f(x) \wedge 2x + 5y \leq 10 \wedge (x = y \vee f(x) = f(y))$$

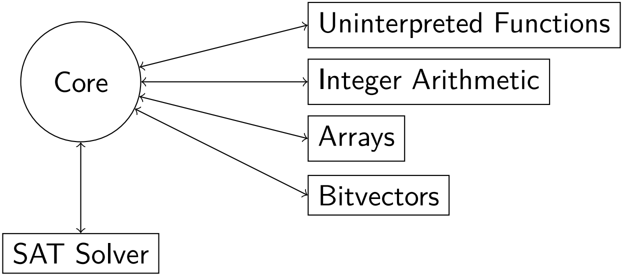
Background: SMT Solving

SMT Solver: fully automated system to determine satisfiability



Background: SMT Solving

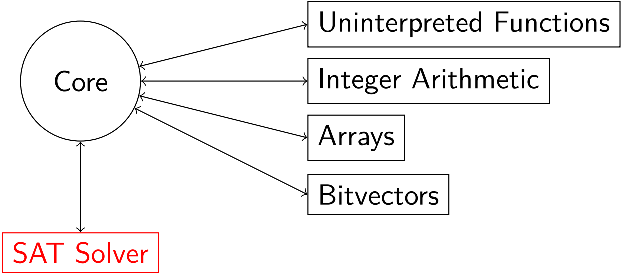
SMT Solver: fully automated system to determine satisfiability



$$f(x+1) \neq f(x) \land 2x + 5y \leq 10 \land (x = y \lor f(x) = f(y))$$

Background: SMT Solving

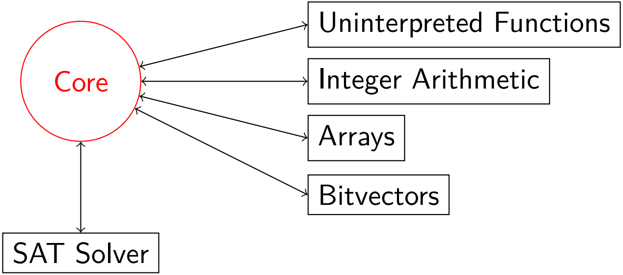
SMT Solver: fully automated system to determine satisfiability



$$f(x+1) \neq f(x) \wedge 2x + 5y \leq 10 \wedge (x = y \vee f(x) = f(y))$$

Background: SMT Solving

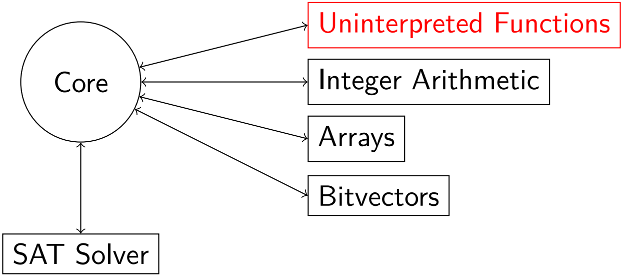
SMT Solver: fully automated system to determine satisfiability



$$f(x+1) \neq f(x) \land 2x + 5y \leq 10 \land (x = y \lor f(x) = f(y))$$

Background: SMT Solving

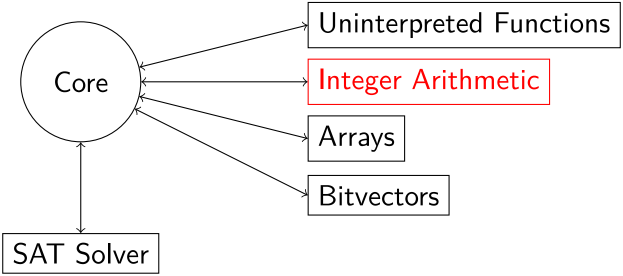
SMT Solver: fully automated system to determine satisfiability



$$f(x+1) \neq f(x) \land 2x + 5y \leq 10 \land (x = y \lor f(x) = f(y))$$

Background: SMT Solving

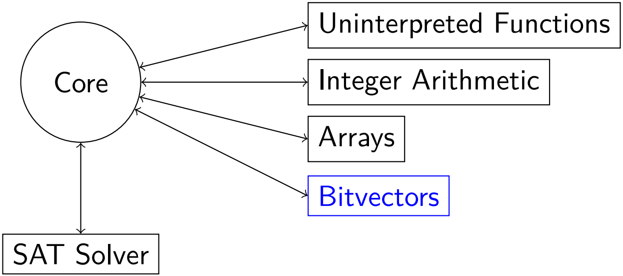
SMT Solver: fully automated system to determine satisfiability



$$f(x+1) \neq f(x) \land 2x + 5y \leq 10 \land (x = y \lor f(x) = f(y))$$

Background: SMT Solving

SMT Solver: fully automated system to determine satisfiability



$$f(x+1) \neq f(x) \wedge 2x + 5y \leq 10 \wedge (x = y \vee f(x) = f(y))$$

Our focus: theory solver for bitvectors!

PolySAT: a Word-level Solver for Large Bitvectors

Bitvectors?

1. Sequence of bits, e.g., `01011`
2. Fixed-width machine integers, e.g., `uint32_t`, `int64_t`
3. Modular arithmetic: $\mathbb{Z}/2^k\mathbb{Z}$

PolySAT: a Word-level Solver for Large Bitvectors

Bitvectors?

1. Sequence of bits, e.g., `01011`
2. Fixed-width machine integers, e.g., `uint32_t`, `int64_t`
3. Modular arithmetic: $\mathbb{Z}/2^k\mathbb{Z}$

Examples:

- $2x^2y + z = 3$
- $x + 3 \leq x + y$
- $z = x \,\&\, y$
- $x[3{:}0] = 0$

PolySAT: a Word-level Solver for Large Bitvectors

Bitvectors?

1. Sequence of bits, e.g., `01011`
2. Fixed-width machine integers, e.g., `uint32_t`, `int64_t`
3. Modular arithmetic: $\mathbb{Z}/2^k\mathbb{Z}$

Examples:

- $2x^2y + z = 3$
- $x + 3 \leq x + y$
- $z = x \,\&\, y$
- $x[3{:}0] = 0$

Natural target for many program verification tasks!

Bitvector Pitfalls

$\mathbb{Z}/2^k\mathbb{Z}$ is a finite commutative ring, but not a field.

Ordering: representatives $\{0, 1, \ldots, 2^k - 1\}$ (unsigned bitvectors)

Bitvector Pitfalls

$\mathbb{Z}/2^k\mathbb{Z}$ is a finite commutative ring, but not a field.

Ordering: representatives $\{0, 1, \ldots, 2^k - 1\}$ (unsigned bitvectors)

$x, y \geq 0 \;\not\Rightarrow\; xy \geq x$	Overflow/wraparound: $3 \cdot 6 = 2 \mod 2^4$
$x, y \neq 0 \;\not\Rightarrow\; xy \neq 0$	Zero divisors: $6 \cdot 8 = 0 \mod 2^4$
$x \leq y \;\not\Rightarrow\; x - y \leq 0$	Usual inequality normalization fails

Bitvector Pitfalls

$\mathbb{Z}/2^k\mathbb{Z}$ is a finite commutative ring, but not a field.

Ordering: representatives $\{0, 1, \ldots, 2^k - 1\}$ (unsigned bitvectors)

$$x, y \geq 0 \not\Rightarrow xy \geq x \qquad \text{Overflow/wraparound: } 3 \cdot 6 = 2 \mod 2^4$$

$$x, y \neq 0 \not\Rightarrow xy \neq 0 \qquad \text{Zero divisors: } 6 \cdot 8 = 0 \mod 2^4$$

$$x \leq y \not\Rightarrow x - y \leq 0 \qquad \text{Usual inequality normalization fails}$$

Example

$x + 3 \leq x + y \mod 2^3$

- For $x = 0$: $3 \leq y \iff y \in \{3, 4, 5, 6, 7\}$
- For $x = 2$: $5 \leq 2 + y \iff y \in \{3, 4, 5\}$

Bitvector Pitfalls

$\mathbb{Z}/2^k\mathbb{Z}$ is a finite commutative ring, but not a field.

Ordering: representatives $\{0, 1, \ldots, 2^k - 1\}$ (unsigned bitvectors)

$x, y \geq 0 \;\not\Rightarrow\; xy \geq x$	Overflow/wraparound: $3 \cdot 6 = 2 \mod 2^4$
$x, y \neq 0 \;\not\Rightarrow\; xy \neq 0$	Zero divisors: $6 \cdot 8 = 0 \mod 2^4$
$x \leq y \;\not\Rightarrow\; x - y \leq 0$	Usual inequality normalization fails

Example

$x + 3 \leq x + y \mod 2^3$

- For $x = 0$: $3 \leq y \iff y \in \{$
- For $x = 2$: $5 \leq 2 + y \iff y \in \{$
- $x + 3 \leq -y + 2 \mod 2^3$

$$
\begin{aligned}
&p \leq q \\
&p \leq p - q - 1 \\
&q - p \leq q \\
&q - p \leq -p - 1 \\
&-q - 1 \leq -p - 1 \\
&-q - 1 \leq p - q - 1
\end{aligned}
$$

Solving Approaches

- Bit-blasting

  Translate into boolean formula and use SAT solver

[1] Yoni Zohar et al.: *Bit-Precise Reasoning via Int-Blasting*

[2] S. Graham-Lengrand, D. Jovanović, B. Dutertre: *Solving bitvectors with MCSAT: explanations from bits and pieces*

Solving Approaches

- Bit-blasting

  Translate into boolean formula and use SAT solver

- Int-blasting[1]

  Translate into integer arithmetic

  Bound constraints: $0 \leq x < 2^k$
  Modulo operations: $x \cdot y \bmod 2^k$

[1] Yoni Zohar et al.: *Bit-Precise Reasoning via Int-Blasting*

[2] S. Graham-Lengrand, D. Jovanović, B. Dutertre: *Solving bitvectors with MCSAT: explanations from bits and pieces*

Solving Approaches

- Bit-blasting

  Translate into boolean formula and use SAT solver

- Int-blasting[1]

  Translate into integer arithmetic

  Bound constraints: $0 \leq x < 2^k$
  Modulo operations: $x \cdot y \bmod 2^k$

- MCSAT-based approaches[2]

  Search for assignment to bitvector variables

  $\rightsquigarrow$ PolySAT

[1] Yoni Zohar et al.: *Bit-Precise Reasoning via Int-Blasting*

[2] S. Graham-Lengrand, D. Jovanović, B. Dutertre: *Solving bitvectors with MCSAT: explanations from bits and pieces*

PolySAT Overview

- Theory solver for bitvector arithmetic
  - Input: conjunction of bitvector constraints
  - Output: SAT or UNSAT
- Based on modular integer arithmetic ($\mathbb{Z}/2^k\mathbb{Z}$)

PolySAT Overview

- Theory solver for bitvector arithmetic
  - Input: conjunction of bitvector constraints
  - Output: SAT or UNSAT
- Based on modular integer arithmetic ($\mathbb{Z}/2^k\mathbb{Z}$)
- Search for a model of the input constraints
  - Incrementally assign bitvector variables
  - Keep track of viable values for variables
  - Add lemmas on demand

Bitvector Language

Arithmetic	$x + y$, $x \cdot y$, div, mod, ...
Equations	$x = y$
Inequalities	$x \leq y$ with $x, y \in \{0, 1, \ldots, 2^k - 1\}$
Inequalities (signed)	$x \leq_s y$ with $x, y \in \{-2^{k-1}, \ldots, 2^{k-1} - 1\}$
Bit-wise	and, or, xor, not, ...
Structural	$shift$, $concat$, $extract$, ...

Bitvector Constraints in PolySAT

Inequalities	$p \leq q$	(polynomials p, q)
Overflow	$\Omega^*(p, q)$	
Bit-wise	$r = p \,\&\, q$	
Structural	$r = p \ll q, \quad r = p \gg q, \quad x = y[h{:}l]$	
Clauses	Disjunction of constraint literals	

Bitvector Constraints in PolySAT

Inequalities	$p \leq q$	(polynomials p, q)
Overflow	$\Omega^*(p, q)$	
Bit-wise	$r = p \,\&\, q$	
Structural	$r = p \ll q, \quad r = p \gg q, \quad x = y[h{:}l]$	
Clauses	Disjunction of constraint literals	
By Reduction:		
Equations	$p = q \iff p - q \leq 0$	

Bitvector Constraints in PolySAT

Inequalities	$p \le q$	(polynomials p, q)
Overflow	$\Omega^*(p, q)$	
Bit-wise	$r = p \mathbin{\&} q$	
Structural	$r = p \ll q, \quad r = p \gg q, \quad x = y[h{:}l]$	
Clauses	Disjunction of constraint literals	
By Reduction:		
Equations	$p = q \iff p - q \le 0$	
Inequalities (signed)	$p \le_s q \iff p + 2^{k-1} \le q + 2^{k-1}$	

Bitvector Constraints in PolySAT

Inequalities	$p \le q$	(polynomials p, q)
Overflow	$\Omega^*(p, q)$	
Bit-wise	$r = p \,\&\, q$	
Structural	$r = p \ll q, \quad r = p \gg q, \quad x = y[h{:}l]$	
Clauses	Disjunction of constraint literals	
By Reduction:		
Equations	$p = q \iff p - q \le 0$	
Inequalities (signed)	$p \le_s q \iff p + 2^{k-1} \le q + 2^{k-1}$	
Bit-wise negation	$\sim p = -p - 1$	

Bitvector Constraints in PolySAT

Inequalities	$p \leq q$	(polynomials p, q)
Overflow	$\Omega^*(p, q)$	
Bit-wise	$r = p \mathbin{\&} q$	
Structural	$r = p \ll q, \quad r = p \gg q, \quad x = y[h{:}l]$	
Clauses	Disjunction of constraint literals	
By Reduction:		
Equations	$p = q \iff p - q \leq 0$	
Inequalities (signed)	$p \leq_s q \iff p + 2^{k-1} \leq q + 2^{k-1}$	
Bit-wise negation	$\sim p = -p - 1$	
Bit-wise or	$p \mid q = p + q - (p \mathbin{\&} q)$	

Bitvector Constraints in PolySAT

Inequalities	$p \leq q$	(polynomials p, q)
Overflow	$\Omega^*(p, q)$	
Bit-wise	$r = p \,\&\, q$	
Structural	$r = p \ll q, \quad r = p \gg q, \quad x = y[h{:}l]$	
Clauses	Disjunction of constraint literals	

By Reduction:

Equations	$p = q \iff p - q \leq 0$	
Inequalities (signed)	$p \leq_s q \iff p + 2^{k-1} \leq q + 2^{k-1}$	
Bit-wise negation	$\sim p = -p - 1$	
Bit-wise or	$p \mid q = p + q - (p \,\&\, q)$	
Quotient/remainder	$q := \mathtt{bvudiv}(a, b)$, $r := \mathtt{bvurem}(a, b)$	

- $a = bq + r$
- $\neg\Omega^*(b, q)$
- $\neg\Omega^+(bq, r)$ (e.g., $bq \leq -r - 1$)
- $b \neq 0 \rightarrow r < b$
- $b = 0 \rightarrow q + 1 = 0$

PolySAT Solving Loop

Modified CDCL loop, similar to MCSAT[3]

- Assign boolean values to constraint literals ($p \leq q$ vs. $p > q$)
- Assign integer values to bitvector variables ($x \mapsto 3$)

[3] L. de Moura, D. Jovanović: *A Model-Constructing Satisfiability Calculus*

PolySAT Solving Loop

Modified CDCL loop, similar to MCSAT[3]

- Assign boolean values to constraint literals ($p \leq q$ vs. $p > q$)
- Assign integer values to bitvector variables ($x \mapsto 3$)

Main components:

- Trail Γ records assignments and reasons
- For each variable x, keep track of viable values V_x
- Conflict $\mathcal{C}$: set of constraints that contradicts Γ
- Conflict analysis extracts lemmas from $\mathcal{C}$

[3] L. de Moura, D. Jovanović: *A Model-Constructing Satisfiability Calculus*

Example: Polynomial Equations

$$
\begin{aligned}
C_1: \quad & x^2y + 3y + 7 = 0 \mod 2^4 \\
C_2: \quad & 2y + z + 8 = 0 \mod 2^4 \\
C_3: \quad & 3x + 4yz + 2z^2 + 1 = 0 \mod 2^4
\end{aligned}
$$

1. $\Gamma = C_1 C_2 C_3$

Example: Polynomial Equations

$$
\begin{aligned}
C_1: && x^2y + 3y + 7 = 0 \quad \text{mod } 2^4 \\
C_2: && 2y + z + 8 = 0 \quad \text{mod } 2^4 \\
C_3: && 3x + 4yz + 2z^2 + 1 = 0 \quad \text{mod } 2^4
\end{aligned}
$$

1. $\Gamma = C_1 C_2 C_3$
2. $\Gamma = C_1 C_2 C_3 (x \mapsto 0)^\delta$ decide x

Example: Polynomial Equations

$$
\begin{aligned}
C_1: &\quad x^2y + 3y + 7 = 0 \mod 2^4\\
C_2: &\quad 2y + z + 8 = 0 \mod 2^4\\
C_3: &\quad 3x + 4yz + 2z^2 + 1 = 0 \mod 2^4
\end{aligned}
$$

1. $\Gamma = C_1 C_2 C_3$
2. $\Gamma = C_1 C_2 C_3 (x \mapsto 0)^\delta$ decide x
   $\rightsquigarrow C_1|_\Gamma : 3y + 7 = 0 \quad \Rightarrow y = 3$

Example: Polynomial Equations

$$\begin{aligned} C_1: &\quad x^2y + 3y + 7 = 0 \mod 2^4 \\ C_2: &\quad 2y + z + 8 = 0 \mod 2^4 \\ C_3: &\quad 3x + 4yz + 2z^2 + 1 = 0 \mod 2^4 \end{aligned}$$

1. $\Gamma = C_1 C_2 C_3$
2. $\Gamma = C_1 C_2 C_3 (x \mapsto 0)^\delta$ decide x
   $\rightsquigarrow C_1|_\Gamma : 3y + 7 = 0 \quad \Rightarrow y = 3$
3. $\Gamma = C_1 C_2 C_3 (x \mapsto 0)^\delta (y \mapsto 3)^{C_1, x}$ propagate y

Example: Polynomial Equations

$$C_1: \quad x^2y + 3y + 7 = 0 \mod 2^4$$
$$C_2: \quad 2y + z + 8 = 0 \mod 2^4$$
$$C_3: \quad 3x + 4yz + 2z^2 + 1 = 0 \mod 2^4$$

1. $\Gamma = C_1 C_2 C_3$
2. $\Gamma = C_1 C_2 C_3 (x \mapsto 0)^\delta$ decide x
   $\rightsquigarrow C_1|_\Gamma : 3y + 7 = 0 \quad \Rightarrow y = 3$
3. $\Gamma = C_1 C_2 C_3 (x \mapsto 0)^\delta (y \mapsto 3)^{C_1,x}$ propagate y
   $\rightsquigarrow C_2|_\Gamma : z + 14 = 0 \quad \Rightarrow z = 2$

Example: Polynomial Equations

$$C_1: \quad x^2 y + 3y + 7 = 0 \mod 2^4$$

$$C_2: \quad 2y + z + 8 = 0 \mod 2^4$$

$$C_3: \quad 3x + 4yz + 2z^2 + 1 = 0 \mod 2^4$$

1. $\Gamma = C_1 C_2 C_3$
2. $\Gamma = C_1 C_2 C_3 (x \mapsto 0)^\delta$ — decide x

   $\rightsquigarrow C_1|_\Gamma : 3y + 7 = 0 \quad \Rightarrow y = 3$
3. $\Gamma = C_1 C_2 C_3 (x \mapsto 0)^\delta (y \mapsto 3)^{C_1, x}$ — propagate y

   $\rightsquigarrow C_2|_\Gamma : z + 14 = 0 \quad \Rightarrow z = 2$
4. $\Gamma = C_1 C_2 C_3 (x \mapsto 0)^\delta (y \mapsto 3)^{C_1, x} (z \mapsto 2)^{C_2, y}$ — propagate z

Example: Polynomial Equations

$$C_1: \quad x^2y + 3y + 7 = 0 \mod 2^4$$
$$C_2: \quad 2y + z + 8 = 0 \mod 2^4$$
$$C_3: \quad 3x + 4yz + 2z^2 + 1 = 0 \mod 2^4$$

1. $\Gamma = C_1 C_2 C_3$
2. $\Gamma = C_1 C_2 C_3 (x \mapsto 0)^\delta$ — decide x
   $\rightsquigarrow C_1|_\Gamma : 3y + 7 = 0 \quad \Rightarrow y = 3$
3. $\Gamma = C_1 C_2 C_3 (x \mapsto 0)^\delta (y \mapsto 3)^{C_1, x}$ — propagate y
   $\rightsquigarrow C_2|_\Gamma : z + 14 = 0 \quad \Rightarrow z = 2$
4. $\Gamma = C_1 C_2 C_3 (x \mapsto 0)^\delta (y \mapsto 3)^{C_1, x} (z \mapsto 2)^{C_2, y}$ — propagate z
   $\rightsquigarrow C_3|_\Gamma : 1 = 0$
   Conflict: $\mathcal{C} = \{C_3, x = 0, y = 3, z = 2\}$

Example: Polynomial Equations (conflict)

$$\Gamma = C_1 C_2 C_3 (x \mapsto 0)^{\delta} (y \mapsto 3)^{C_1, x} (z \mapsto 2)^{C_2, y}$$
$$\mathcal{C} = \{C_3, x = 0, y = 3, z = 2\}$$

Follow dependencies of $\mathcal{C}$ according to Γ:

$$\mathcal{C}' = \{C_3, x = 0, y = 3, C_2\}$$

Example: Polynomial Equations (conflict)

$$\Gamma = C_1 C_2 C_3 (x \mapsto 0)^{\delta} (y \mapsto 3)^{C_1,x} (z \mapsto 2)^{C_2,y}$$

$$\mathcal{C} = \{C_3, x = 0, y = 3, z = 2\}$$

Follow dependencies of $\mathcal{C}$ according to Γ:

$$\mathcal{C}' = \{C_3, x = 0, y = 3, C_2\}$$

$$\begin{aligned} C_3: \quad & 3x + 4yz + 2z^2 + 1 = 0 \\ C_2: \quad & 2y + z + 8 = 0 \qquad \mid \cdot 2z \\ C_3 - 2z \cdot C_2: \quad & 3x + 1 = 0 \end{aligned}$$

Example: Polynomial Equations (conflict)

$$\Gamma = C_1 C_2 C_3 (x \mapsto 0)^{\delta} (y \mapsto 3)^{C_1,x} (z \mapsto 2)^{C_2,y}$$
$$\mathcal{C} = \{C_3, x = 0, y = 3, z = 2\}$$

Follow dependencies of $\mathcal{C}$ according to Γ:

$$\mathcal{C}' = \{C_3, x = 0, y = 3, C_2\}$$

$$\begin{aligned} C_3: &\quad 3x + 4yz + 2z^2 + 1 = 0 \\ C_2: &\quad 2y + z + 8 = 0 \quad \mid \cdot 2z \\ C_3 - 2z \cdot C_2: &\quad 3x + 1 = 0 \end{aligned}$$

Lemma:

$$C_3 \wedge C_2 \rightarrow 3x + 1 = 0$$

Example: Polynomial Equations

Constraints:

$$
\begin{aligned}
C_1: &\quad x^2y + 3y + 7 = 0 \mod 2^4 \\
C_2: &\quad 2y + z + 8 = 0 \mod 2^4 \\
C_3: &\quad 3x + 4yz + 2z^2 + 1 = 0 \mod 2^4 \\
C_4: &\quad 3x + 1 = 0 \mod 2^4
\end{aligned}
$$

Example: Polynomial Equations

Constraints:

$$
\begin{aligned}
C_1: &\quad x^2y + 3y + 7 = 0 \mod 2^4 \\
C_2: &\quad 2y + z + 8 = 0 \mod 2^4 \\
C_3: &\quad 3x + 4yz + 2z^2 + 1 = 0 \mod 2^4 \\
C_4: &\quad 3x + 1 = 0 \mod 2^4
\end{aligned}
$$

Continued:

5. $\Gamma = C_1 C_2 C_3 C_4^{C_2, C_3}$ backjump, propagate lemma

Example: Polynomial Equations

Constraints:

$$\begin{aligned} C_1: && x^2y + 3y + 7 = 0 \quad \text{mod } 2^4 \\ C_2: && 2y + z + 8 = 0 \quad \text{mod } 2^4 \\ C_3: && 3x + 4yz + 2z^2 + 1 = 0 \quad \text{mod } 2^4 \\ C_4: && 3x + 1 = 0 \quad \text{mod } 2^4 \end{aligned}$$

Continued:

5. $\Gamma = C_1 C_2 C_3 C_4^{C_2, C_3}$ backjump, propagate lemma
6. $\Gamma = C_1 C_2 C_3 C_4^{C_2, C_3} (x \mapsto 5)^{C_4}$ propagate x

Example: Polynomial Equations

Constraints:

$$
\begin{aligned}
C_1 &: & x^2 y + 3y + 7 = 0 &\quad \text{mod } 2^4 \\
C_2 &: & 2y + z + 8 = 0 &\quad \text{mod } 2^4 \\
C_3 &: & 3x + 4yz + 2z^2 + 1 = 0 &\quad \text{mod } 2^4 \\
C_4 &: & 3x + 1 = 0 &\quad \text{mod } 2^4
\end{aligned}
$$

Continued:

5. $\Gamma = C_1 C_2 C_3 C_4^{C_2, C_3}$ backjump, propagate lemma
6. $\Gamma = C_1 C_2 C_3 C_4^{C_2, C_3} (x \mapsto 5)^{C_4}$ propagate x
   $\rightsquigarrow C_1|_\Gamma : 12y + 7 = 0$
   Conflict due to parity!

Example: Polynomial Equations

Constraints:

$$\begin{aligned}
C_1 &: & x^2y + 3y + 7 = 0 &\quad \text{mod } 2^4 \\
C_2 &: & 2y + z + 8 = 0 &\quad \text{mod } 2^4 \\
C_3 &: & 3x + 4yz + 2z^2 + 1 = 0 &\quad \text{mod } 2^4 \\
C_4 &: & 3x + 1 = 0 &\quad \text{mod } 2^4
\end{aligned}$$

Continued:

5. $\Gamma = C_1\, C_2\, C_3\, C_4^{C_2, C_3}$ backjump, propagate lemma
6. $\Gamma = C_1\, C_2\, C_3\, C_4^{C_2, C_3} (x \mapsto 5)^{C_4}$ propagate x
   $\rightsquigarrow C_1|_\Gamma : 12y + 7 = 0$
   Conflict due to parity!
7. Unsatisfiable.

How to choose values?

For each variable x, keep track of viable values V_x:

- choose a value from V_x for decisions
- propagate $x \mapsto v$ when $V_x = \{v\}$ is a singleton set
- conflict if $V_x = \emptyset$

How to choose values?

For each variable x, keep track of viable values V_x:

- choose a value from V_x for decisions
- propagate $x \mapsto v$ when $V_x = \{v\}$ is a singleton set
- conflict if $V_x = \emptyset$

Currently:

- V_x represented as set of intervals
- when x appears only linearly, extract a forbidden interval
- additionally, keep track of fixed bits of x (e.g., $2^4 x = 2^4\, 5$)
- bit-blasting as fallback
  (only a single bitvector variable)

Intervals

We use half-open intervals:

- Usual notation $[\ell; u[$
- but wrap around if $\ell > u$

Intervals

We use half-open intervals:

- Usual notation $[\ell; u[$
- but wrap around if $\ell > u$

Examples mod 2^4:

$$[2; 5[= \{2, 3, 4\}$$
$$[13; 2[= \{13, 14, 15, 0, 1\}$$
$$[0; 0[= \emptyset$$

Note:

$$p \in [\ell; u[\quad \iff \quad p - \ell < u - \ell$$

Forbidden Intervals

p, q, r, s: polynomials, evaluable in current trail Γ
x: variable, unassigned in Γ

$$px + r \leq qx + s$$

[4] S. Graham-Lengrand, D. Jovanović, B. Dutertre: *Solving bitvectors with MCSAT: explanations from bits and pieces*

Forbidden Intervals

p, q, r, s: polynomials, evaluable in current trail Γ
x: variable, unassigned in Γ

$$px + r \leq qx + s$$

$\widehat{p}$	$\widehat{q}$	Interval	
0	1	$x \notin [-s; r - s[$	if $r \neq 0$
1	0	$x \notin [s - r + 1; -r[$	if $s \neq -1$
1	1	$x \notin [-s; -r[$	if $r \neq s$
		Lemmas from intervals[4]	

[4] S. Graham-Lengrand, D. Jovanović, B. Dutertre: *Solving bitvectors with MCSAT: explanations from bits and pieces*

Forbidden Intervals

p, q, r, s: polynomials, evaluable in current trail Γ
x: variable, unassigned in Γ

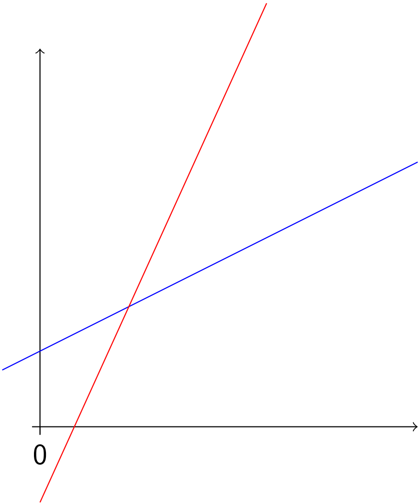
$$px + r \leq qx + s$$

$\widehat{p}$	$\widehat{q}$	Interval	
0	1	$x \notin [-s; r - s[$	if $r \neq 0$
1	0	$x \notin [s - r + 1; -r[$	if $s \neq -1$
1	1	$x \notin [-s; -r[$	if $r \neq s$
		Lemmas from intervals[4]	
$\{0, n\}$	$\{0, n\}$	Set of intervals ("equal coeff.")	
n	m	Set of intervals ("disequal coeff.")	
		Intervals from fixed bits	
		Combination with value selection	
		Fallback to bit-blasting	

[4] S. Graham-Lengrand, D. Jovanović, B. Dutertre: *Solving bitvectors with MCSAT: explanations from bits and pieces*

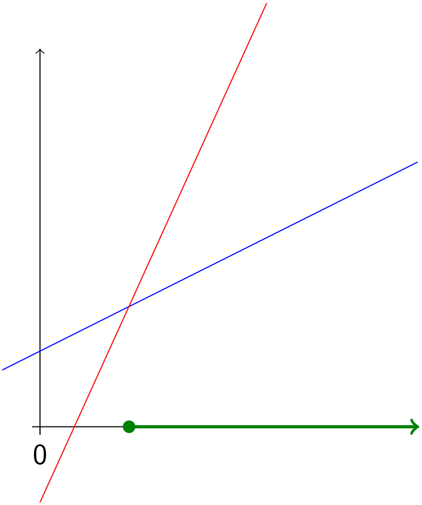
Forbidden Intervals (disequal coefficients)

$$px + r \leq qx + s \quad \text{with } \widehat{p} \neq \widehat{q}$$



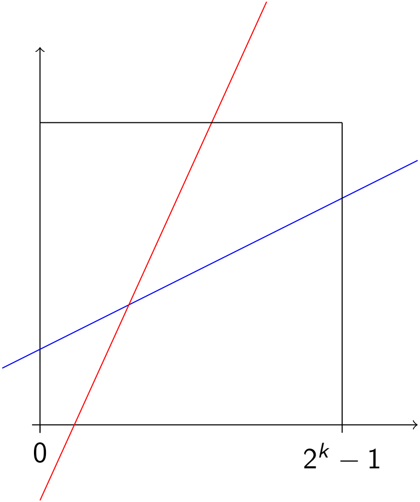
Forbidden Intervals (disequal coefficients)

$px + r \leq qx + s \quad \text{with } \widehat{p} \neq \widehat{q}$



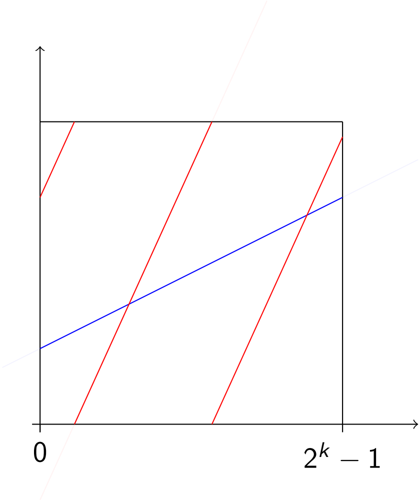
Forbidden Intervals (disequal coefficients)

$$px + r \le qx + s \quad \text{with } \widehat{p} \neq \widehat{q}$$



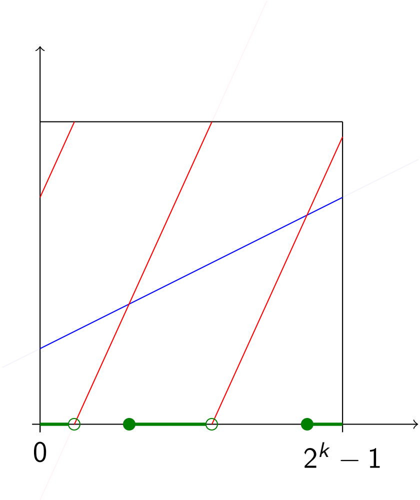
Forbidden Intervals (disequal coefficients)

$$\color{blue}{px + r} \leq \color{red}{qx + s} \quad \text{with } \widehat{p} \neq \widehat{q}$$



Forbidden Intervals (disequal coefficients)

$px + r \leq qx + s \quad \text{with } \widehat{p} \neq \widehat{q}$



choice based on current value $\widehat{x}$

Conflict Resolution Strategy

1. Track the conflict's cone of influence while backtracking over the trail Γ
2. Conflict resolution plugins derive lemmas from constraints in the conflict
3. For now, accumulate lemmas from conflict plugins
   - New (often simpler) constraints improve propagation
   - Easy to experiment with new types of lemmas
4. When reaching the first relevant decision, learn lemmas and resume search

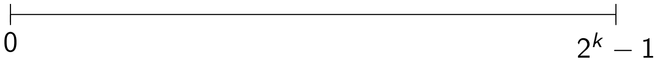
Forbidden Interval Lemma

- Assume conflict $V_x = \emptyset$
- Forbidden intervals:
  $C_i \implies x \notin [\ell_i; u_i[$ if c_i

S. Graham-Lengrand, D. Jovanović, B. Dutertre: *Solving bitvectors with MCSAT: explanations from bits and pieces*

Forbidden Interval Lemma

- Assume conflict $V_x = \emptyset$
- Forbidden intervals:
  $C_i \implies x \notin [\ell_i; u_i[$ if c_i
- Concrete intervals cover the domain: $\bigcup_i [\widehat{\ell_i}; \widehat{u_i}[= [0; 2^k[$



S. Graham-Lengrand, D. Jovanović, B. Dutertre: *Solving bitvectors with MCSAT: explanations from bits and pieces*

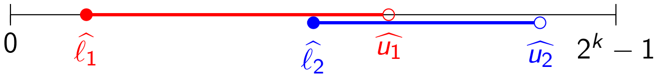
Forbidden Interval Lemma

- Assume conflict $V_x = \emptyset$
- Forbidden intervals:
  $C_i \implies x \notin [\ell_i; u_i[$ if c_i
- Concrete intervals cover the domain: $\bigcup_i [\widehat{\ell_i}; \widehat{u_i}[= [0; 2^k[$

S. Graham-Lengrand, D. Jovanović, B. Dutertre: *Solving bitvectors with MCSAT: explanations from bits and pieces*

Forbidden Interval Lemma

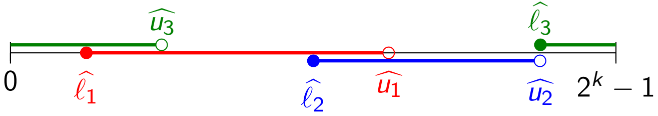
- Assume conflict $V_x = \emptyset$
- Forbidden intervals:
  $C_i \implies x \notin [\ell_i; u_i[$ if c_i
- Concrete intervals cover the domain: $\bigcup_i [\widehat{\ell_i}; \widehat{u_i}[= [0; 2^k[$



S. Graham-Lengrand, D. Jovanović, B. Dutertre: *Solving bitvectors with MCSAT: explanations from bits and pieces*

Forbidden Interval Lemma

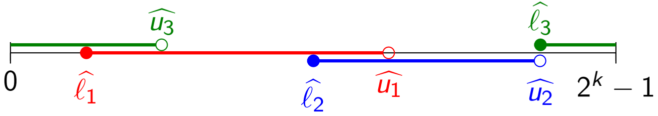
- Assume conflict $V_x = \emptyset$
- Forbidden intervals:
  $C_i \implies x \notin [\ell_i; u_i[$ if c_i
- Concrete intervals cover the domain: $\bigcup_i [\widehat{\ell_i}; \widehat{u_i}[= [0; 2^k[$



S. Graham-Lengrand, D. Jovanović, B. Dutertre: *Solving bitvectors with MCSAT: explanations from bits and pieces*

Forbidden Interval Lemma

- Assume conflict $V_x = \emptyset$
- Forbidden intervals:
  $C_i \implies x \notin [\ell_i; u_i[$ if c_i
- Concrete intervals cover the domain: $\bigcup_i [\widehat{\ell_i}; \widehat{u_i}[= [0; 2^k[$



- Use symbolic intervals to express the overlap condition:

$$u_1 \in [\ell_2; u_2[\ \wedge \ u_2 \in [\ell_3; u_3[\ \wedge \ u_3 \in [\ell_1; u_1[$$

S. Graham-Lengrand, D. Jovanović, B. Dutertre: *Solving bitvectors with MCSAT: explanations from bits and pieces*

Conflict Resolution Plugins

Forbidden Intervals Lemma			
Superposition	$p(x) = 0 \wedge q(x) = 0$	$\Longrightarrow$	$rp(x) + sq(x) = 0$
	choose r, s to eliminate highest power of x		
Overflow	$\Omega^*(p, q) \wedge \neg\Omega^*(p, r)$	$\Longrightarrow$	$q > r$
Inequality	$px < qx$	$\Longrightarrow$	$\Omega^*(x, p) \vee p < q$
	$px \leq qx$	$\Longrightarrow$	$\Omega^*(x, p) \vee p \leq q$
			$\vee\ x = 0$
Bit-wise *and*	$x = p \,\&\, q$	$\Longrightarrow$	$x \leq p$
	$x = p \,\&\, q \wedge p = q$	$\Longrightarrow$	$x = p$
	$x = p \,\&\, q \wedge p = 2^n - 1$	$\Longrightarrow$	$2^{n-k}x = 2^{n-k}q$
Bounds	$C(x, y) \wedge x \in [x_l; x_h]$	$\Longrightarrow$	$y \in [y_l; y_h]$
	$\Omega^*(p, q) \wedge p \leq b_1$	$\Longrightarrow$	$q \geq b_2$
	$axy + bx + cy + d \leq \dots$	$\Longrightarrow$	$\dots$
...	...		...

Conclusion

PolySAT

- Bit-vector solver in Z3
- Word-level reasoning

Thank you!