From Local Search\textsuperscript{1} to Quantifier-Elimination\textsuperscript{2} for Bit-Vectors in SMT

Aina Niemetz
Stanford University

joint work with Clark Barrett\textsuperscript{2*}, Armin Biere\textsuperscript{1○}, Mathias Preiner\textsuperscript{12*}, Andrew Reynolds\textsuperscript{2†} and Cesare Tinelli\textsuperscript{2†}

* Stanford University † The University of Iowa ○ Johannes Kepler University Linz

Theory and Practice of Satisfiability Solvers
August 27-31, 2018
Oaxaca, Mexico
Theory of Fixed-Size Bit-Vectors

- constants, variables: 00000010, 2[8], x[32], y[2]
- bit-vector operators: =, <, >, ∼, &, ≪, ≫, ◦, [:], ...
- arithmetic operators modulo 2^n (overflow semantics!)

Bit-Blasting

- current state-of-the-art for quantifier-free bit-vector formulas
- rewriting + simplifications + eager reduction to SAT

- efficient in practice
- may suffer from an exponential blow-up in the formula size
- may not scale well for increasing bit-widths
Example \( x[8] \times y[8] = z[8] \)
Example

\[ x_{[32]} * y_{[32]} = z_{[32]} \]
without bit-blasting, no SAT solver (orthogonal approach)

- assume satisfiability, start with initial assignment
- propagate target values towards inputs
- iteratively improve current state until solution is found

not able to determine unsatisfiability

Probabilistically Approximately Complete (PAC) [Hoos, AAAI’99]
if there exists a non-deterministic choice of moves that lead to a solution
choose controlling / essential input, else choose random input
Value Selection

- produces target value **without** changing the value of other inputs
  - **unconditional** inverse not always possible

- **less strict** notion
  - produces target value **after** changing the value of other inputs

- using **only** inverse values without further randomization is **incomplete**
Why is using only inverse values incomplete?

Results

Implemented in Boolector

- **Bb** bit-blasting engine
- **Bb+Pw-10k** sequential portfolio combination
  - 16436 benchmarks from QF_BV (SMT-LIB) (sat + unknown)
  - 1200s time limit
unconditional inverse value computation not possible in general

if the current assignment does not satisfy the **invertibility** condition for a bit-vector operator we choose a **consistent** value

Can we utilize the concept of invertibility conditions for something else?
• unconditional inverse value computation not possible in general
  ▶ if the current assignment does not satisfy the invertibility condition for a bit-vector operator we choose a consistent value

Can we utilize the concept of invertibility conditions for something else?
  ▶ Yes! For quantified bit-vector formulas!
Motivation

Example \( \psi = \forall x . (x + s \not\approx t) \quad x, s, t \ldots \) bit-vectors of size \( N \)

State of the Art in SMT: Quantifier instantiation-based techniques

- Find conflicting ground instances of the formula
  - Crucial to find good instantiation candidates

  - Naive: Enumerate values for \( x \) \( (2^N \) possible instantiations)\)
  
  - Better: Instantiate with symbolic term \( t - s \)

\[ (t - s) + s \not\approx t \]

\( \text{UNSAT} \)

- Idea: Compute symbolic inverses of bit-vector operators [CAV’18]
Symbolic Inverses

- **unconditional inverses** do not always exist

**Example** \( x \cdot s \approx t \), \( x, s, t \) \ldots bit-vectors of size \( N \)

- solve for \( x \)
- no inverse for, e.g., \( x \cdot 2 \approx 3 \)

- **invertibility condition**: \( ((-s \mid s) \& t) \approx t \)
- identifies condition under which \( x \cdot s \approx t \) is invertible:
  \[ ((-s \mid s) \& t) \approx t \iff x \cdot s \approx t \]
- independent from the bit-width
Invertibility Conditions

- 162 invertibility conditions for:
  - Operators: $\diamond \in \{\&, |, \ll, \gg, \gg_a, \cdot, \mod, \div, \circ\}$
  - Relations: $\triangledown \in \{\approx, \not\approx, <_u, \leq_u, >_u, \geq_u, <_s, \leq_s, >_s, \geq_s\}$
- 83 manually, 79 synthesized with SyGuS (syntax-guided synthesis)

► **SyGuS problem:**
$$\exists C \forall s \forall t. ( (\exists x. x \diamond s \triangleleft t) \iff C(s, t) )$$

► **Expand innermost** (4-bit):
$$\exists (4\text{-bit}): \exists C \forall s \forall t. ( \bigvee_{i=0}^{15} i \diamond s \triangleleft t ) \iff C(s, t)$$

- Synthesized 118 conditions (out of 140) with CVC4
- Verified correctness of 94.6% the 162 ICs for bit-width 1 to 65 (with Boolector, CVC4, Q3B, Z3)
Hilbert choice function $\varepsilon x. \varphi[x]$

- represents a solution for $\varphi[x]$ if there is one
- and an arbitrary value otherwise
- $\exists x. \varphi[x] \iff \varphi[\varepsilon x. \varphi[x]]$

Embed invertibility conditions into Hilbert choice functions

- bit-vector literal: $e[x] := x \oplus s \otimes t$
- invertibility condition: $C(s, t) \iff e[x]$
- symbolic term: $\varepsilon y. (C(s, t) \Rightarrow e[y])$

- choice functions express all conditional solutions in one symbolic term
Example: $\forall x. (s_2 + x) \cdot s_1 >_u t$

1. Pick variable to solve for ($x$)
2. Compute inverse/invertibility conditions along path to $x$
Example: $\forall x. (s_2 + x) \cdot s_1 >_u t$

1. Pick variable to solve for ($x$)

2. Compute inverse/invertibility conditions along path to $x$

3. $x' \cdot s_1 >_u t$
   - $IC_{x'} = t <_u -s \mid s$
   - $x' = \varepsilon y. (IC_{x'} \Rightarrow y \cdot s_1 >_u t)$
Example: $\forall x. (s_2 + x) \cdot s_1 >_u t$

1. Pick variable to solve for ($x$)

2. Compute inverse/invertibility conditions along path to $x$

3. $x' \cdot s_1 >_u t$
   - $IC_{x'} = t <_u -s \mid s$
   - $x' = \varepsilon y. (IC_{x'} \Rightarrow y \cdot s_1 >_u t)$

4. $s_2 + x \approx x'$
   - $IC_x = \top$
   - $x = x' - s_2$

Instantiation for $x$: $\varepsilon y. (t <_u -s \mid s \Rightarrow s_1 \cdot y >_u t) - s_2$
Multiple Variable Occurrences

Non-linear constraints (multiple occurrences of a variable)

- Try to linearize with rewriting/normalization
  e.g., \( x + x + s \approx t \rightarrow 2 \cdot x + s \approx t \)

- Else: Replace all but one occurrence with value in current model \( \mathcal{I} \)
  e.g., \( x \cdot x + s \approx t \rightarrow x \cdot x^{\mathcal{I}} + s \approx t \)

▸ Future work: Use SyGuS to synthesize ICs for non-linear cases

Unit linear invertible formulas

- If \( \forall x. \varphi[x] \) is linear in \( x \) (only one occurrence of \( x \))

▸ Quantifier elimination: reduce to quantifier-free bit-vector formula
## Experiments

<table>
<thead>
<tr>
<th></th>
<th>CVC4_{base}</th>
<th>Q3B</th>
<th>Boolector</th>
<th>Z3</th>
<th>CVC4_{ic}</th>
</tr>
</thead>
<tbody>
<tr>
<td>keymaera (4035)</td>
<td>3823</td>
<td>3805</td>
<td>4025</td>
<td>4031</td>
<td>3993</td>
</tr>
<tr>
<td>psyco (194)</td>
<td>194</td>
<td>99</td>
<td>193</td>
<td>193</td>
<td>190</td>
</tr>
<tr>
<td>scholl (374)</td>
<td>239</td>
<td>214</td>
<td>289</td>
<td>271</td>
<td>246</td>
</tr>
<tr>
<td>tptp (73)</td>
<td>73</td>
<td>73</td>
<td>72</td>
<td>73</td>
<td>73</td>
</tr>
<tr>
<td>uauto (284)</td>
<td>112</td>
<td>256</td>
<td>180</td>
<td>190</td>
<td>274</td>
</tr>
<tr>
<td>wintersteiger (191)</td>
<td>168</td>
<td>184</td>
<td>154</td>
<td>162</td>
<td>168</td>
</tr>
<tr>
<td><strong>Total (5151)</strong></td>
<td>4609</td>
<td>4631</td>
<td>4913</td>
<td>4920</td>
<td><strong>4944</strong></td>
</tr>
</tbody>
</table>

Limits: 300 seconds CPU time limit, 100G memory limit

**CVC4_{ic}** won division BV at SMT-COMP 2018
Conclusion

- Propagation-based local search approach implemented in Boolector
  https://github.com/boolector/boolector

- Quantifier elimination approach implemented in CVC4
  https://github.com/cvc4/cvc4
