LEARNING TO INSTANTIATE QUANTIFIERS

Armin Biere¹  joint work with  Mathias Preiner¹,², Aina Niemetz¹,²
TACAS’17, SMT’17, PhD Thesis Mathias Preiner in 2017

¹ Johannes Kepler University Linz  ² Stanford University

Verification Seminar
Department of Computer Science
University of Oxford
18th December 2017
Introduction

- **Counterexample-Guided**
  Combine counterexample-guided quantifier instantiation with . . .

- **Synthesis**
  . . . syntax-guided synthesis to synthesize . . .

- **Model**
  . . . interpretations for Skolem functions.

- **Quantified Bit-Vectors**
Fixed-Size Bit-Vectors

**Bit-Vector**: vector of bits of a fixed size

- **Constant values**: 0011, 00000011, 3\[8], ...
- **Variables**: \(x[16], y[9], \ldots\)
- **Operators**:
  - *bitwise*: \(\sim, \&, |, \oplus, <<, >>, \ldots\)
  - *arithmetic*: +, −, ∗, /, ...
  - *predicates*: =, <, ≤, ...
  - *string operations*: concat, extract, extension, ...

**Example with Quantifiers**
\[ \forall x[4] \exists y[4]. (x \& 1100) + y = 0000 \]
Quantified Bit-Vectors

State-of-the-Art

- **Z3**: Model-based quantifier instantiation (MBQI) [de Moura’09]
  - combined with E-matching, symbolic quantifier instantiation

- **CVC4**: Counterexample-guided quantifier instantiation (CEGQI) [Reynolds’15]
  - concrete models and counterexamples only

- **Q3B**: BDD-based approach [Strejcek’16]
  - relies on simplifications, approximation techniques, variable ordering

Our approach

**Counterexample-Guided Model Synthesis (CEGMS)**

- Combines synthesis with variant of CEGQI
Counterexample-Guided Model Synthesis

Example  \( \varphi := \forall x_{[32]} \exists y_{[32]} . x + y = 0 \)

Skolem  \( \varphi_S := \forall x_{[32]} . x + f(x) = 0 \)

Ground Instances of \( \varphi_S \)

<table>
<thead>
<tr>
<th>x</th>
<th>x + f(x) = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 + f(0) = 0</td>
</tr>
<tr>
<td>1</td>
<td>1 + f(1) = 0</td>
</tr>
<tr>
<td>2</td>
<td>2 + f(2) = 0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(2^{32}-1)</td>
<td>...</td>
</tr>
</tbody>
</table>
Counterexample-Guided Model Synthesis

Example \( \varphi := \forall x_{[32]} \exists y_{[32]} \cdot x + y = 0 \)

Skolem \( \varphi_S := \forall x_{[32]} \cdot x + f(x) = 0 \)

<table>
<thead>
<tr>
<th>Ground Instances of ( \varphi_S )</th>
<th>( x )</th>
<th>( x + f(x) = 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0 + f(0) = 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1 + f(1) = 0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2 + f(2) = 0</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>( 2^{32} - 1 )</td>
<td>( 2^{32} - 1 )</td>
<td>( \vdots )</td>
</tr>
</tbody>
</table>

Function Table \( f \)

<table>
<thead>
<tr>
<th>( x )</th>
<th>( f(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>-2</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>( 2^{32} - 1 )</td>
<td>( -(2^{32} - 1) )</td>
</tr>
</tbody>
</table>
Counterexample-Guided Model Synthesis

Example \( \varphi := \forall x_{[32]} \exists y_{[32]} . x + y = 0 \)

Skolem \( \varphi_S := \forall x_{[32]} . x + f(x) = 0 \)

Ground Instances of \( \varphi_S \)

<table>
<thead>
<tr>
<th>( x )</th>
<th>( x + f(x) = 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( 0 + f(0) = 0 )</td>
</tr>
<tr>
<td>1</td>
<td>( 1 + f(1) = 0 )</td>
</tr>
<tr>
<td>2</td>
<td>( 2 + f(2) = 0 )</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>( 2^{32} - 1 )</td>
<td>( \ldots )</td>
</tr>
</tbody>
</table>

Function Table \( f \)

<table>
<thead>
<tr>
<th>( x )</th>
<th>( f(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>-2</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>( 2^{32} - 1 )</td>
<td>( -(2^{32} - 1) )</td>
</tr>
</tbody>
</table>

Goal

\[ f := \lambda x. -x \]

\[ \forall x_{[32]} . x - x = 0 \checkmark \]

How?

Synthesize + Refine
Workflow

Preprocessing → Check Ground Instances → Synthesize Candidate Model → Check Candidate Model → CEGQI → New ground instance

- \( \varphi \) → Preprocessing
- Check Ground Instances → sat → CEGQI → sat → Check Candidate Model
- Synthesize Candidate Model → sat → Check Candidate Model
- CEGQI → Counterexample
- New ground instance → Check Candidate Model

- unsat → SAT
- unsat → UNSAT

- Skolem function Interpr.
Workflow

Preprocessing $\phi$ → Check Ground Instances → Synthesize Candidate Model → Check Candidate Model

- CEGQR
- SAT
- UNSAT

New ground instance

Skolem function Interpr.

Counter-example

unsat → sat → unsat
Workflow

- Preprocessing
- Check Ground Instances
- Synthesize Candidate Model
- Check Candidate Model

φ → Preprocessing → Check Ground Instances → Synthesize Candidate Model → Check Candidate Model

unsat → New ground instance → sat → CEGQI → sat → Counter-example → unsat → Skolem function Interpr.

SAT → UNSAT
Workflow

1. Preprocessing
2. Check Ground Instances
3. Synthesize Candidate Model
4. Check Candidate Model

- $\varphi \rightarrow$ Preprocessing
- New ground instance
- CEGQI
- SAT
- UNSAT

- Check Ground Instances $\rightarrow$ sat
- Synthesize Model $\rightarrow$ sat
- Skolem function Interpreter
- Check Candidate Model $\rightarrow$ unsat
- Counterexample

Decision:
- SAT
- UNSAT
Workflow

1. Preprocessing
2. Check Ground Instances
3. Synthesize Candidate Model
4. Check Candidate Model
5. Skolem function Interpr.
6. CEGQI

- $\varphi$ → Preprocessing → Check Ground Instances → Synthesize Candidate Model → Check Candidate Model → SAT/UNSAT
- New ground instance
- sat → Unsat
- Unsat → sat → Counter-example
- Unsat → Unsat

5/16
Synthesis of Candidate Models

Enumerative Learning [Alur’13]

- enumerate expressions based on a syntax/grammar
- check if expressions conform to some set of test cases
  - generate expression signature
  - discard expressions with same signature (pruning)
- return expression if signature matches target signature
  - candidate expressions must satisfy set of ground instances

Size | Enumerated Expressions
--- | ---
1 | \( x \), \( y \), \( z \), 0, 1, 2, \( \ldots \)
2 | \( \ldots \), \( x + y \), \( x + z \), \( y + z \), \( x * y \), \( \ldots \), \( x = y \), \( \ldots \)
3 | \( \ldots \), \( (x + y) * x \), \( (x + y) * 2 \), \( \ldots \), \( x < (x * y) \), \( y < (x * y) \), \( \ldots \)
4 | \( \ldots \), \( (x + y) \& (x * y) \), \( \ldots \), \( \text{ite}(x = y, z, x) \), \( \ldots \)
\( \ldots \)
Example: Synthesis

Example: \( z = \min(x, y) \)

\[ \varphi := \forall x \, y \exists z \cdot (x < y \rightarrow z = x) \land (x \geq y \rightarrow z = y) \]

\[ \varphi_S := \forall x \, y \cdot (x < y \rightarrow f_z(x, y) = x) \land (x \geq y \rightarrow f_z(x, y) = y) \]

Inputs for \( f_z \) \{ \( x, y \) \}

Operators \{ =, <, \geq, \land, \rightarrow, \text{ite} \}

Ground Inst. \( G \) \{ \[ f_z(0, 0) = 0, \] \[ f_z(0, 1) = 0, \] \[ f_z(2, 1) = 1 \] \}
Example: Synthesis cont.

Size | Enumerated Expressions
---|---
1 | \(x, y\)
2 | \(x = y, y = x, x < y, y < x, x \geq y, y \geq x\)
3 | -
4 | \((x = y \land x < y), \ldots, (x = y \rightarrow x < y), \ldots, \text{ite}(x < y, x, y)\)

Signature Computation

- substitute \(f_z\) in \(G := \{g_1, \ldots, g_n\}\) by current expression \(\lambda xy \cdot t[x, y]\)
- evaluate resulting \(g_1', \ldots, g_n'\)
- obtain vector of \(n\) Boolean values (= signature)

Signature of Candidate \(\text{ite}(x < y, x, y)\)

\[
\begin{align*}
\text{ite}(0 < 0, 0, 0) &= 0, & \text{ite}(0 < 1, 0, 1) &= 0, & \text{ite}(2 < 1, 2, 1) &= 1
\end{align*}
\]
Example: Check Candidate Model

Candidate Model
\[ \{ f_z := \lambda x y . \text{ite}(x < y, x, y) \} \]

Check

\[ \neg \varphi_S[\lambda x y . \text{ite}(x < y, x, y)/f_z] \]
\[ \equiv \exists x y . (x < y \land \text{ite}(x < y, x, y) \neq x) \lor (x \geq y \land \text{ite}(x < y, x, y) \neq y) \]

SMT Solver Check

\[ (a < b \land \text{ite}(a < b, a, b) \neq a) \lor (a \geq b \land \text{ite}(a < b, a, b) \neq b) \]

- unsat: candidate model is valid
- sat: found counterexample, refine
Example: Refinement

Assume Candidate Model \( \{ f_z := \lambda x y . x \} \)

SMT Solver Check

\[ (a < b \wedge a \neq a) \lor (a \geq b \wedge a \neq b) \]

- Solver returns \textit{sat}, candidate model is invalid
- Solver produces counterexample \( \{ a = 1, b = 0 \} \)

Add New Instance of \( \varphi_S \) to \( G \)

\[ G := G \cup \{ \varphi_S[1/x, 0/y] \} \]
Dual Counterexample-Guided Model Synthesis

Idea
Find instantiation for $\forall$-variables s.t. formula is unsatisfiable.

How
Apply CEGMS to the dual formula $\neg \varphi$

Duality
$\text{CEGMS}(\neg \varphi)$ sat $\iff$ CEGMS($\varphi$) unsat

$\text{CEGMS}(\neg \varphi)$ unsat $\iff$ CEGMS($\varphi$) sat

Original
$\varphi := \exists a \ b \ c \ \forall x . \underbrace{(a * c) + (b * c)}_{\text{unsat with } \varphi[a+b/x]} \neq \underbrace{(x * c)}_{\text{unsat with } \varphi[a+b/x]}$

Dual
$\neg \varphi := \forall a \ b \ c \ \exists x . \underbrace{(a * c) + (b * c)}_{\text{sat with } \neg \varphi[a+b/x]} = \underbrace{(x * c)}_{\text{sat with } \neg \varphi[a+b/x]}$

- Dual CEGMS finds non-ground quantifier instantiations
- CEGMS($\varphi$) and CEGMS(\neg \varphi) can be executed in parallel
## Experiments

<table>
<thead>
<tr>
<th>SMT-LIB (191)</th>
<th>New(^1) (4838)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solved</td>
<td>Sat</td>
</tr>
<tr>
<td><strong>Boolector</strong></td>
<td>142</td>
</tr>
<tr>
<td><strong>Boolector+s</strong></td>
<td>164</td>
</tr>
<tr>
<td><strong>Boolector+d</strong></td>
<td>162</td>
</tr>
<tr>
<td><strong>Boolector+ds</strong></td>
<td><strong>172</strong></td>
</tr>
</tbody>
</table>

**Boolector . . . CEGQI only** \(+s . . . \) synthesis \(+d . . . \) dual (parallel)

**Limits** 1200 seconds CPU time, 7GB memory

---

\(^1\) LIA, LRA, NIA, NRA SMT-LIB benchmarks translated to BV
## Experiments

<table>
<thead>
<tr>
<th></th>
<th>SMT-LIB (191)</th>
<th>New (4838)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Solved</td>
<td>Sat</td>
</tr>
<tr>
<td>Boolector+ds</td>
<td>172</td>
<td>77</td>
</tr>
<tr>
<td>CVC4</td>
<td>145</td>
<td>64</td>
</tr>
<tr>
<td>Q3B</td>
<td><strong>187</strong></td>
<td><strong>93</strong></td>
</tr>
<tr>
<td>Z3</td>
<td>161</td>
<td>69</td>
</tr>
</tbody>
</table>

**Limits** 1200 seconds CPU time, 7GB memory
Experiments

Synthesis Overhead (Runtime)

- up to 75% on solved benchmarks
- up to 98% on unsolved benchmarks

Refinement Iterations

- up to 300 iterations on solved benchmarks
- up to 9400 iterations on unsolved benchmarks

Synthesized Terms

- $c$
- $x_i$
- $(x_i \text{ op } x_j)$
- $(c \text{ op } x_i)$
- $\sim(c \times x_i)$
- $(x_i + (c + \sim x_j))$

$x_i$ ... universal variables, \hspace{1em} $c$ ... constant value, \hspace{1em} $\text{op}$ ... bit-vector operator
Conclusion

- **simple** approach for solving quantified bit-vectors
  - only requires two instances of ground theory solvers
  - enumerative learning algorithm straightforward to implement

- **competitive** with the state-of-the-art in solving BV
  - no simplification techniques yet
  - no E-matching or other quantifier instantiation heuristics

- **future directions**
  - improve synthesis approach
    - employ divide and conquer approach from [Alur’17]
    - employ other synthesis approaches?
  - generalize counterexamples via synthesis
  - model reconstruction from unsatisfiable dual formulas
  - useful for other theories?
References I


