LEARNING TO INSTANTIATE QUANTIFIERS

Armin Biere[1] joint work with Mathias Preiner[1,2], Aina Niemetz[1,2]

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

JOHANNES KEPLER UNIVERSITÄT LINZ

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: $0011,\ 00000011,\ 3_{[8]}, \ldots$
- Variables: $x_{[16]},\ y_{[9]}, \ldots$
- Operators:
  - bitwise: $\sim,\ \&,\ |,\ \oplus,\ \ll,\ \gg, \ldots$
  - arithmetic: $+,\ -,\ *,\ /, \ldots$
  - predicates: $=,\ <,\ \leq, \ldots$
  - string operations: concat, extract, extension, ...

Example with Quantifiers $\qquad \forall x_{[4]}\, \exists y_{[4]}\ .\ (x \,\&\, 1100) + y = 0000$

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 φ_S

$\mathbf{x}$	$\mathbf{x + f(x) = 0}$
0	$0 + f(0) = 0$
1	$1 + f(1) = 0$
2	$2 + f(2) = 0$
$\vdots$	$\vdots$
2^{32}-1	$\ldots$

Counterexample-Guided Model Synthesis

Example $\varphi \coloneqq \forall x_{[32]} \, \exists y_{[32]} \, . \, x + y = 0$

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

Ground Instances of φ_S

$\mathbf{x}$	$\mathbf{x + f(x) = 0}$
0	$0 + f(0) = 0$
1	$1 + f(1) = 0$
2	$2 + f(2) = 0$
$\vdots$	$\vdots$
2^{32}-1	$\ldots$

Function Table f

$\mathbf{x}$	$\mathbf{f(x)}$
0	0
1	-1
2	-2
$\vdots$	$\vdots$
2^{32}-1	-(2^{32}-1)

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 φ_S

$\mathbf{x}$	$\mathbf{x + f(x) = 0}$
0	$0 + f(0) = 0$
1	$1 + f(1) = 0$
2	$2 + f(2) = 0$
⋮	⋮
2^{32}-1	...

Function Table f

$\mathbf{x}$	$\mathbf{f(x)}$
0	0
1	-1
2	-2
⋮	⋮
2^{32}-1	-(2^{32}-1)

Goal

$f := \lambda x. -x$

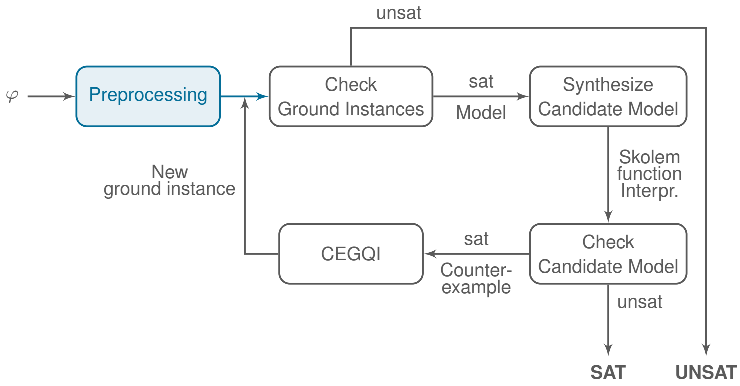
$\downarrow$

$\forall x_{[32]} \, . \, x + -x = 0$ ✓

How?

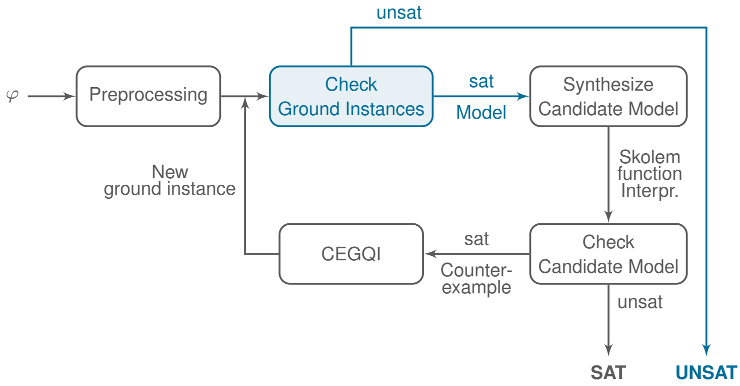
Synthesize + Refine

Workflow



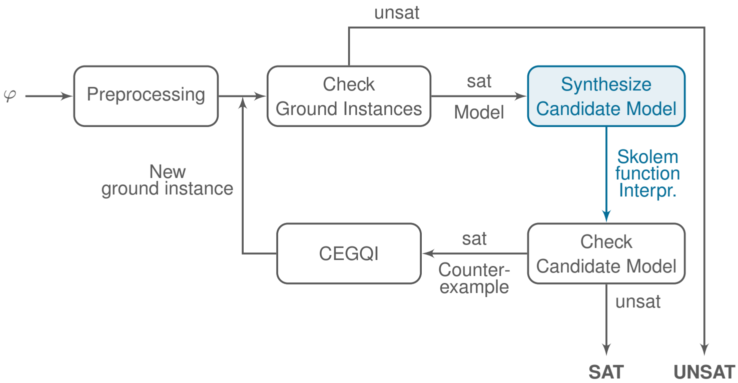
unsat
φ
Preprocessing
Check
Ground Instances
sat
Model
Synthesize
Candidate Model
Skolem
function
Interpr.
New
ground instance
CEGQI
sat
Counter-
example
Check
Candidate Model
unsat
SAT
UNSAT

Workflow



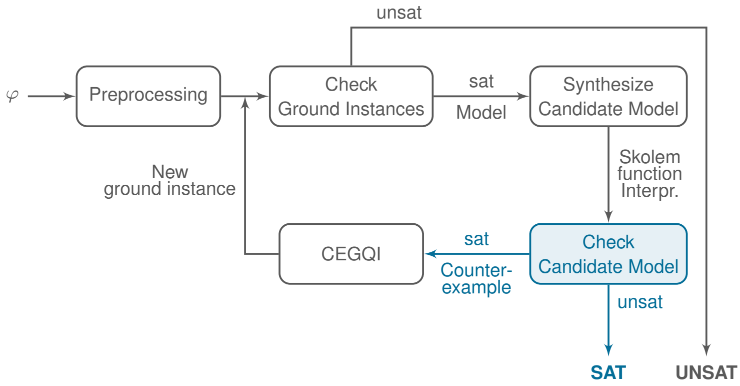
unsat
φ
Preprocessing
Check
Ground Instances
sat
Model
Synthesize
Candidate Model
Skolem
function
Interpr.
New
ground instance
CEGQI
sat
Counter-
example
Check
Candidate Model
unsat
SAT
UNSAT

Workflow



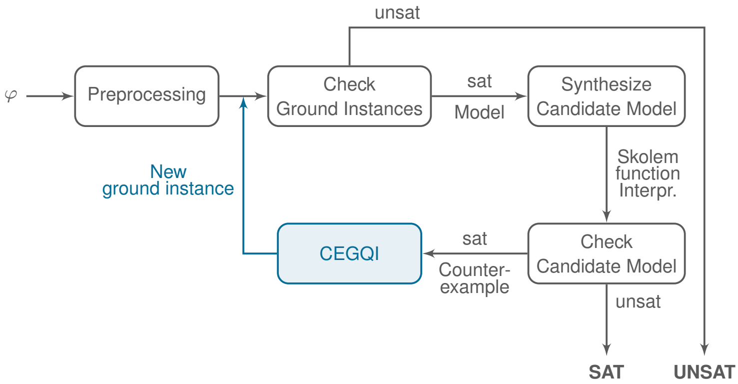
unsat
φ
Preprocessing
Check
Ground Instances
sat
Model
Synthesize
Candidate Model
Skolem
function
Interpr.
New
ground instance
CEGQI
sat
Counter-
example
Check
Candidate Model
unsat
SAT
UNSAT

Workflow

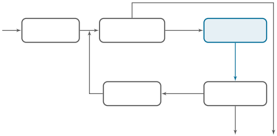


unsat
φ
Preprocessing
Check
Ground Instances
sat
Model
Synthesize
Candidate Model
Skolem
function
Interpr.
New
ground instance
CEGQI
sat
Counter-
example
Check
Candidate Model
unsat
SAT
UNSAT

Workflow


unsat
φ
Preprocessing
Check
Ground Instances
sat
Model
Synthesize
Candidate Model
Skolem
function
Interpr.
New
ground instance
CEGQI
sat
Counter-
example
Check
Candidate Model
unsat
SAT
UNSAT

Synthesis of Candidate Models

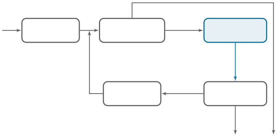


Enumerative Learning [Alur'13]

- 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	$\boxed{x}$ $\boxed{y}$ $\boxed{z}$ $\boxed{0}$ $\boxed{1}$ $\boxed{2}$ $\cdots$
2	$\cdots$ $\boxed{x+y}$ $\boxed{x+z}$ $\boxed{y+z}$ $\boxed{x*y}$ $\cdots$ $\boxed{x=y}$ $\cdots$
3	$\cdots$ $\boxed{(x+y)*x}$ $\boxed{(x+y)*2}$ $\cdots$ $\boxed{x<(x*y)}$ $\boxed{y<(x*y)}$ $\cdots$
4	$\cdots$ $\boxed{(x+y)\&(x*y)}$ $\cdots$ $\boxed{\text{ite}(x=y,z,x)}$ $\cdots$
$\vdots$	

Example: Synthesis



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

$$\varphi := \forall x\, y\, \exists z\, .\, (x < y \rightarrow z = x) \land (x \geq y \rightarrow z = y)$$

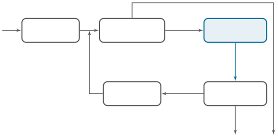
$$\varphi_S := \forall x\, y\, .\, (x < y \rightarrow f_z(x, y) = x) \land (x \geq y \rightarrow f_z(x, y) = y)$$

Inputs for f_z $\quad \{\, x,\ y \,\}$

Operators $\quad \{\, =,\ <,\ \geq,\ \land,\ \rightarrow,\ ite \,\}$

Ground Inst. G $\quad \{\, f_z(0,0) = 0,$
$\quad f_z(0,1) = 0,$
$\quad f_z(2,1) = 1 \,\}$

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	$(x = y \land x < y), \; \ldots, \; (x = y \rightarrow x < y), \; \ldots, \; ite(x < y, x, y)$

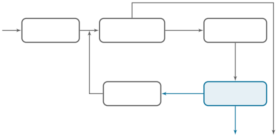
Signature Computation

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

Signature of Candidate $ite(x < y, x, y)$

$$\underbrace{ite(0 < 0, 0, 0) = 0}_{\top}, \; \underbrace{ite(0 < 1, 0, 1) = 0}_{\top}, \; \underbrace{ite(2 < 1, 2, 1) = 1}_{\top}$$

Example: Check Candidate Model



Candidate Model $\{f_z \coloneqq \lambda\, x\, y\ .\ ite(x < y, x, y)\}$

Check

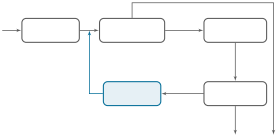
$$\neg\varphi_S[\lambda\, x\, y\ .\ ite(x < y, x, y)/f_z]$$
$$\equiv \exists\, x\, y\ .\ (x < y \land ite(x < y, x, y) \neq x) \lor (x \geq y \land ite(x < y, x, y) \neq y)$$

SMT Solver Check

$$\underbrace{(a < b \land ite(a < b, a, b) \neq a)}_{\perp} \lor \underbrace{(a \geq b \land ite(a < b, a, b) \neq b)}_{\perp}$$

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

Example: Refinement



Assume Candidate Model $\{f_z := \lambda\, x\, y\, .\, x\}$

SMT Solver Check

$$\underbrace{(a < b \wedge a \neq a)}_{\perp} \vee \underbrace{(a \geq b \wedge a \neq b)}_{\top}$$

▷ Solver returns sat, candidate model is invalid

▷ Solver produces counterexample $\{\, a = 1, b = 0 \,\}$

Add New Instance of φ_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 CEGMS($\neg\varphi$) sat $\rightsquigarrow$ CEGMS(φ) unsat

CEGMS($\neg\varphi$) unsat $\rightsquigarrow$ CEGMS(φ) sat

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

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

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

Experiments

	SMT-LIB (191)				**New**[1] (4838)			
	Solved	Sat	Unsat	Time [s]	Solved	Sat	Unsat	Time [s]
Boolector	142	51	91	59529	4527	465	4062	389020
Boolector+s	164	72	92	32996	4526	467	4059	390613
Boolector+d	162	67	95	35877	4572	**518**	4054	342412
Boolector+ds	**172**	**77**	**95**	**24163**	**4704**	517	**4187**	**187411**

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

	SMT-LIB (191)				**New** (4838)			
	Solved	Sat	Unsat	Time [s]	Solved	Sat	Unsat	Time [s]
Boolector+ds	172	77	**95**	24163	4704	**517**	4187	187411
CVC4	145	64	81	57652	4362	339	4023	580402
Q3B	**187**	**93**	94	**9086**	4367	327	4040	581252
Z3	161	69	92	37534	4732	476	**4256**	**133241**

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
- $(x_i \ op \ x_j)$
- $(c \ op \ x_i)$
- $\sim(c * x_i))$
- $(x_i + (c + \sim x_j))$

x_i ...universal variables, c ...constant value, 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

[Alur'17] Rajeev Alur and Arjun Radhakrishna and Abhishek Udupa. Scaling Enumerative Program Synthesis via Divide and Conquer. TACAS, 2017

[Alur'13] Abhishek Udupa and Arun Raghavan and Jyotirmoy V. Deshmukh and Sela Mador-Haim and Milo M. K. Martin and Rajeev Alur. TRANSIT: specifying protocols with concolic snippets. SIGPLAN, 2013

[Strejcek'16] Martin Jonás and Jan Strejcek. Solving Quantified Bit-Vector Formulas Using Binary Decision Diagrams. SAT, Pages 267-283. 2016

[de Moura'09] Yeting Ge and Leonardo Mendonça de Moura. Complete Instantiation for Quantified Formulas in Satisfiabiliby Modulo Theories. CAV, Pages 306-320. 2009

[Reynolds'15] Andrew Reynolds and Morgan Deters and Viktor Kuncak and Cesare Tinelli and Clark W. Barrett. Counterexample-Guided Quantifier Instantiation for Synthesis in SMT. CAV, Pages 198-216. 2015