Model Finding for Recursive Functions in SMT

Andrew Reynolds

Jasmin Christian Blanchette

Cesare Tinelli **SMT** July 18, 2015

Recursive Functions

• Recursive function definitions:

f(x:Int) := if $x \le 0$ then 0 else f(x-1)+x

- Are useful in applications:
 - Software verification
 - Theorem Proving
- Often, interested in finding models for
 - Conjectures $(\exists x.) P(f, x)$ in the presence of recursive functions f
 - This poses a challenge to current SMT solvers

Recursive Functions

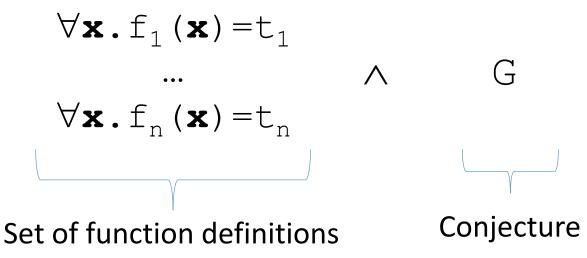
• Recursive function definitions:

f(x:Int) := if $x \le 0$ then 0 else f(x-1)+x

• Can be expressed in SMT as quantified formulas:

 $\forall x: Int. f(x) = ite(x \le 0, 0, f(x-1) + x)$

• SMT solver must handle inputs of the form:



Recursive Functions

- In this talk:
 - Existing techniques for quantified formulas in SMT
 - Limited in their ability to find models when recursive functions are present
 - A satisfiability-preserving translation A for function definitions
 - Allows us to use existing techniques for model finding
 - Evaluation of translation A on benchmarks from theorem proving/verification

Existing Techniques for Quantified Formulas in SMT

- Heuristic Techniques for UNSAT:
 - E-matching [Detlefs et al 2003, Ge et al 2007, de Moura/Bjorner 2007]
- Limited Techniques for SAT:
 - Local theory extensions [Sofronie-Stokkermans 2005]
 - Array fragments [Bradley et al 2006, Alberti et al 2014]
 - Complete Instantiation [Ge/de Moura 2009]
 - Implemented in Z3
 - Finite Model Finding [Reynolds et al 2013]
 - Implemented in CVC4

Existing Techniques for Quantified Formulas in SMT

- Heuristic Techniques for UNSAT:
 - E-matching [Detlefs et al 2003, Ge et al 2007, de Moura/Bjorner 2007]
- Limited Techniques for SAT:
 - Local theory extensions [Sofronie-Stokkermans 2005]
 - Array fragments [Bradley et al 2006, Alberti et al 2014]
 - Complete Instantiation [Ge/de Moura 2009]
 - Implemented in Z3
 - Finite Model Finding [Reynolds et al 2013]
 - Implemented in CVC4

- Focus of next slides

Complete Instantiation in Z3

• Complete method for \forall in essentially uninterpreted fragment

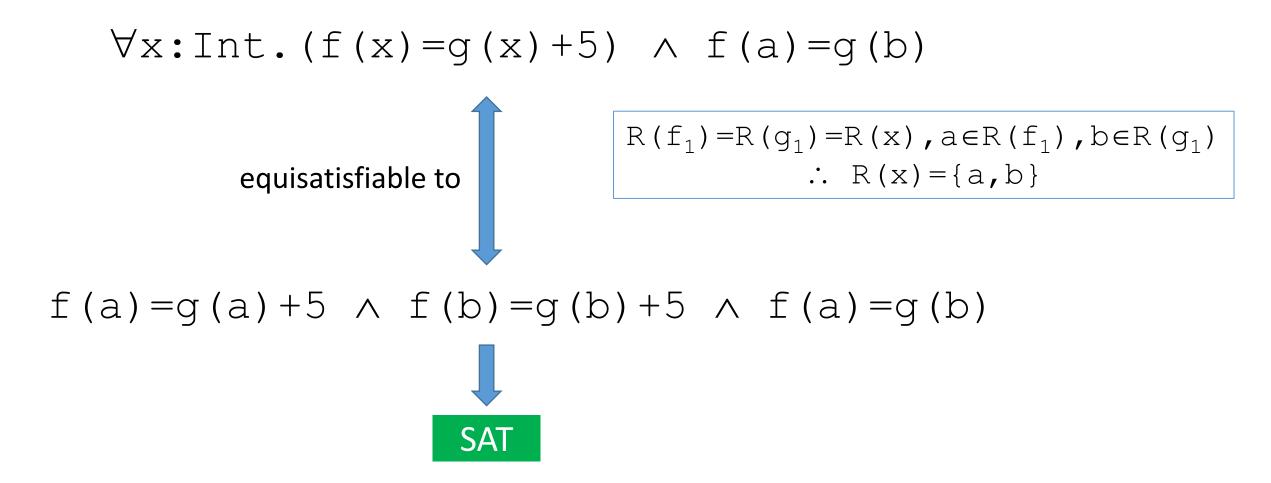
 $\forall x: Int. (f(x) = g(x) + 5) \land f(a) = g(b)$ All occurrences of x are children of UF

Complete Instantiation in Z3

$\forall x: Int. (f(x) = g(x) + 5) \land f(a) = g(b)$

Relevant domain R(x) of variable x is $\{a, b\}$

Complete Instantiation in Z3



Finite Model Finding in CVC4

• Finite Model-complete method for finite/uninterpreted \forall

$$\forall xy: U. (x \neq y \Rightarrow f(x) \neq f(y)) \land a \neq b$$

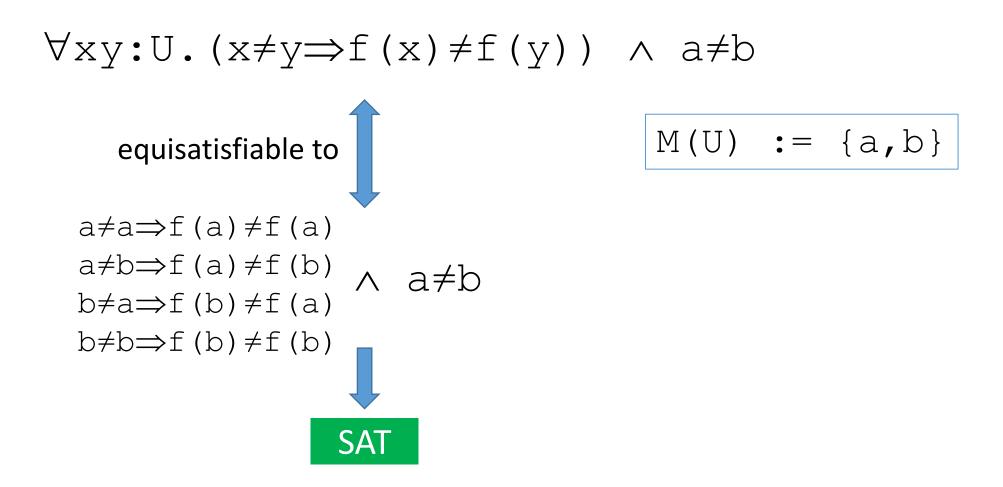
All variables have finite/uninterpreted sort **U**

Finite Model Finding in CVC4

$\forall xy: U. (x \neq y \Rightarrow f(x) \neq f(y)) \land a \neq b$

$$M(U) := \{a, b\}$$
Model interprets U as the set $M(U) = \{a, b\}$

Finite Model Finding in CVC4



...Both fail on most Recursive Function Definitions!

• Example:

 $\forall x: Int. (f(x) = ite(x \le 0, 0, f(x-1) + x)) \land f(k) > 100$

...Both fail on most Recursive Function Definitions!

• Example:

 $\forall x: Int. (f(x) = ite(x \le 0, 0, f(x-1) + x)) \land f(k) > 100$

- Complete instantiation:
 - Fails, since body has subterm f(x-1) + x with unshielded variable x
 - $R(x) = \{k, k-1, k-2, k-3, ...\}$

...Both fail on most Recursive Function Definitions!

• Example:

 $\forall x: Int. (f(x) = ite(x \le 0, 0, f(x-1) + x)) \land f(k) > 100$

- Complete instantiation:
 - Fails, since body has subterm f(x-1) + x with unshielded variable x
 - $R(x) = \{k, k-1, k-2, k-3, ...\}$
- Finite Model Finding:
 - Fails, since quantification is over infinite type Int
 - M(Int) = {..., -3, -2, -1, 0, 1, 2, 3, ...}

Running example

∀x:Int.(f(x)=ite(x≤0,0,f(x-1)+x)) ∧ f(k)>100

- Function f
 - Returns the sum of all positive integers up to x, when x is non-negative
- Formula is satisfiable
 - By models interpreting k as an integer ≥ 14

Can we make the problem easier?

$$\forall x: Int. (f(x) = ite(x \le 0, 0, f(x-1) + x)) \land f(k) > 100$$

- What if we assume function definitions in Φ are *well-behaved*?
 - E.g. we know that \pm is terminating
- Introduce translation A, which:
 - Restricts quantification to subset of the domain of function definitions
 - Under right assumptions, preserves satisfiability
- Use existing techniques for model finding in Z3, CVC4 on $\mathbb{A}\left(\Phi\right)$

Translation A: Part 1

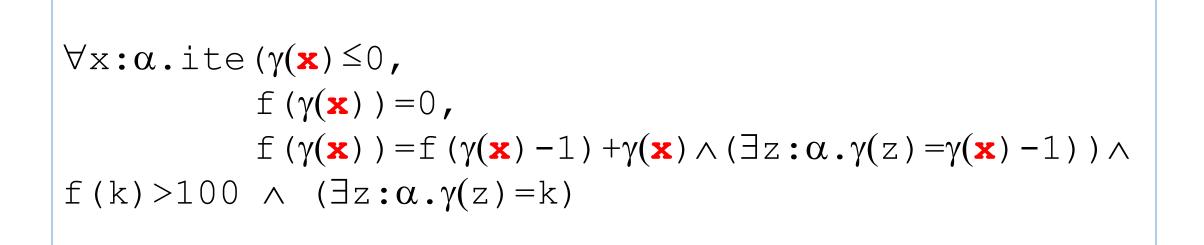
$$\begin{aligned} \forall x : \boldsymbol{\alpha} \text{.ite} (\boldsymbol{\gamma}(x) \leq 0, \\ & \text{f}(\boldsymbol{\gamma}(x)) = 0, \\ & \text{f}(\boldsymbol{\gamma}(x)) = \text{f}(\boldsymbol{\gamma}(x) - 1) + \boldsymbol{\gamma}(x)) \wedge \\ & \text{f}(k) > 100 \end{aligned}$$

- Introduce uninterpreted sort α
 - Conceptually, α represents the set of relevant arguments of ${\rm f}$
 - Restrict the domain of function definition quantification to $\boldsymbol{\alpha}$
- Introduce uninterpreted function $\gamma : \alpha \rightarrow \text{Int}$
 - Maps between abstract and concrete domains

Translation A: Part 2

$$\begin{aligned} \forall \mathbf{x} : \boldsymbol{\alpha} . \text{ ite } (\gamma(\mathbf{x}) \leq 0, \\ & \text{ f } (\gamma(\mathbf{x})) = 0, \\ & \text{ f } (\gamma(\mathbf{x})) = \text{ f } (\gamma(\mathbf{x}) - 1) + \gamma(\mathbf{x}) \wedge (\exists \mathbf{z} : \boldsymbol{\alpha} . \gamma(\mathbf{z}) = \gamma(\mathbf{x}) - 1)) \wedge \\ & \text{ f } (\mathbf{k}) > 100 \wedge (\exists \mathbf{z} : \boldsymbol{\alpha} . \gamma(\mathbf{z}) = \mathbf{k}) \end{aligned}$$

- Add appropriate constraints regarding α , γ
 - Each relevant concrete value must be mapped to by some abstract value



• \forall is essentially uninterpreted

$$\begin{aligned} \forall \mathbf{x} : \boldsymbol{\alpha} \text{.ite} \left(\gamma(\mathbf{x}) \leq 0 \right), \\ & \text{f} \left(\gamma(\mathbf{x}) \right) = 0 , \\ & \text{f} \left(\gamma(\mathbf{x}) \right) = \text{f} \left(\gamma(\mathbf{x}) - 1 \right) + \gamma(\mathbf{x}) \land \left(\exists z : \boldsymbol{\alpha} \cdot \gamma(z) = \gamma(\mathbf{x}) - 1 \right) \right) \land \\ & \text{f} \left(\mathbf{k} \right) > 100 \land \left(\exists z : \boldsymbol{\alpha} \cdot \gamma(z) = \mathbf{k} \right) \end{aligned}$$

• \forall is essentially uninterpreted, and over finite/uninterpreted sorts

$$\begin{aligned} \forall \mathbf{x} : \mathbf{\alpha} . \text{ ite } (\gamma(\mathbf{x}) \leq 0, \\ & \text{ f } (\gamma(\mathbf{x})) = 0, \\ & \text{ f } (\gamma(\mathbf{x})) = \text{ f } (\gamma(\mathbf{x}) - 1) + \gamma(\mathbf{x}) \land (\exists z : \alpha . \gamma(z) = \gamma(\mathbf{x}) - 1)) \land \\ & \text{ f } (k) > 100 \land (\exists z : \alpha . \gamma(z) = k) \end{aligned}$$

∀ is essentially uninterpreted, and over finite/uninterpreted sorts
 ⇒Both Z3 (complete instantiation) and CVC4 (finite model finding)
 find model for this benchmark in <.1 second

$$\begin{aligned} \forall x : \alpha . \text{ite} (\gamma(x) \leq 0, \\ & f(\gamma(x)) = 0, \\ & f(\gamma(x)) = f(\gamma(x) - 1) + \gamma(x) \land (\exists z : \alpha . \gamma(z) = \gamma(x) - 1)) \land \\ & f(k) > 100 \land (\exists z : \alpha . \gamma(z) = k) \end{aligned}$$

- Formula is satisfied by a model **M** where:
 - M(k) := 14, M(f) := λ x.ite(x=14, 105, ite(x=13, 91, ... ite(x=1, 1, 0) ...))

 \Rightarrow *M* is correct only for relevant inputs of original formula, and not e.g. f(15) = 0

• Nevertheless, A is satisfiability-preserving under right assumptions

Translation A : Properties

- Translation A is:
 - Refutation sound
 - When A $(\Phi)\,$ is unsatisfiable, Φ is unsatisfiable
 - Model sound, when function definitions are admissible
 - When A (Φ) is satisfiable, Φ is satisfiable

Translation A : Properties

- Translation A is:
 - Refutation sound
 - When A $(\Phi)\,$ is unsatisfiable, Φ is unsatisfiable
 - Model sound, when function definitions are *admissible*
 - When A (Φ) is satisfiable, Φ is satisfiable

Focus of next slides

Admissible Function Definitions

• Given a function definition:

 $\forall x \cdot f(x) = t[x]$

• A set of ground formulas G is *closed under function expansion wrt* \pm if:

$$G \models \{f(k)=t[k] \mid f(k) \in terms(G)\}$$

- A function definition $\forall x . f(x) = t$ is *admissible* if:
 - For all G that is closed under function expansion wrt f:

G is sat \Rightarrow G $\land \forall x.f(x) = t[x]$ is also sat

Admissible Function Definitions

- Examples of admissible definitions:
 - Terminating functions: $\forall x . f(x) = ite(x \le 0, 0, f(x-1) + x)$
 - f is well-founded (terminating)
 - Consistent definitions: $\forall x . f(x) = f(x)$
 - \pm is essentially unconstrained
 - ...even: $\forall x . f(x) = f(x-1) + 1$

Inadmissible Function Definitions

- Examples of inadmissible definitions:
 - Inconsistent definitions: $\forall x . f(x) = f(x) + 1$
 - T is closed under function expansion wrt ${\rm f}$
 - But no model for $T \land \forall x . f(x) = f(x) + 1$
 - Others: { $\forall x.f(x) = f(x) + g(x)$, $\forall x.g(x) = g(x)$ }
 - Although has model where f and g are $\lambda\texttt{x}$. 0,
 - g(0)=1 is closed under function expansion wrt f, g
 - But no model for $g(0) = 1 \land \forall x.f(x) = f(x) + g(x) \land \forall x.g(x) = g(x)$

Evaluation

- Considered two sets of benchmarks:
 - Isa
 - Challenge problems for inductive theorem provers
 - Purely datatypes + recursive functions
 - Leon
 - Taken from Leon verification tool (EPFL)
 - Many theories: datatypes + recursive functions + bitvectors + arrays + sets + arithmetic
- Consider mutated forms of these benchmarks (Isa-mut, Leon-mut)
 - Obtained by swapping subterms in conjectures
 - High likelihood to have models
- All benchmarks considered with/without translation A

Evaluation : solved SAT benchmarks

	Z3	CVC4f	
	$arphi ~~ \mathcal{A}(arphi)$	$arphi \mathcal{A}(arphi)$	Total
Isa	0 0	0 0	79
Leon	0 2	0 9	166
Isa-Mut	0 35	0 153	213
Leon-Mut	11 75	6 169	427
Total	11 112	6 331	885

- Translation increases ability of SMT solvers for finding models:
 - Z3: 11 -> 112
 - CVC4: 6 -> 331
- Finds counterexamples to verification conditions of interest in Leon

Evaluation : solved UNSAT benchmarks

	Z3	CVC4f	
	$arphi ~~ \mathcal{A}(arphi)$	$arphi \mathcal{A}(arphi)$	Total
Isa	14 15	15 15	79
Leon	73 78	80 76	166
Isa-Mut	17 18	18 18	213
Leon-Mut	83 98	104 95	427
Total	187 209	217 204	885

- Translation has mixed impact on UNSAT benchmarks:
 - Z3 : 187 -> 209
 - CVC4 : 217 -> 204

• CVC4 supports SMT LIB version 2.5 command:

```
...
(define-fun-rec f ((x Int)) Int
    (ite (<= x 0) 0 (+ (f (- x 1)) x)))
(assert (> (f k) 100))
(check-sat)
```

• Input (without A) is equivalent to:

...
(assert (forall ((x Int))
 (= (f x) (ite (<= x 0) 0 (+ (f (- x 1)) x))))
(assert (> (f k) 100))
(check-sat)

• Input (with A) is equivalent to:

 \Rightarrow Enabled as preprocessor by command line parameter "--fmf-fun"

• Model (with A) outputted is:

• Gives model that is correct for relevant inputs of function \pm

Summary

- Translation A:
 - Increases ability of SMT solvers for model finding recursive functions
 - Complete instantiation in Z3
 - Finite Model Finding in CVC4
 - Is model-sound for admissible function definitions
 - Implemented as a preprocessor in CVC4 "--fmf-fun"
 - Responsibility on user to show function definitions are admissible

Future Work

- Increase scope of evaluation
 - Comparison against existing counterexample generators (Leon, Nitpick, ...)
- Use of CVC4 as backend
 - To Leon verification system
 - To Isabelle proof assistant
- Identify additional sufficient conditions for admissibility
 - E.g. productive corecursive functions

Thanks!

- CVC4:
 - Available at http://cvc4.cs.nyu.edu/downloads/

- To use translation A as a preprocessor:
 - Use command line option "--fmf-fun"

