Complete Instantiation of Quantified Formulas in Satisfiability Modulo Theories

Yeting Ge[1] Leonardo de Moura[2]

[1]New York University

[2]Microsoft Research

7th International Workshop on Satisfiability Modulo Theories

Aug 3, 2009

Montreal, Canada

Quantified SMT formulas

- Traditional SMT solvers only deal with quantifier free formulas
- Quantified SMT formulas are useful

Quantified SMT formulas

- Traditional SMT solvers only deal with quantifier free formulas
- Quantified SMT formulas are useful
- Unsupported/customized theories
  - Type system in ESC/Java, Boogie
  - $\forall x_1, x_2, x_3 : (subtype(x_1, x_2) \wedge subtype(x_2, x_3) \rightarrow subtype(x_1, x_3))$

Quantified SMT formulas

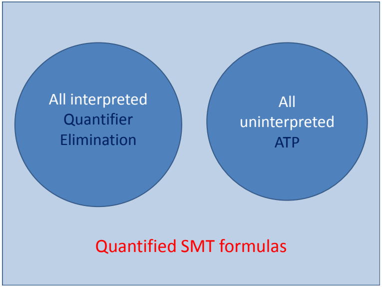
- Traditional SMT solvers only deal with quantifier free formulas
- Quantified SMT formulas are useful
- Unsupported/customized theories
  - Type system in ESC/Java, Boogie
  - $\forall x_1, x_2, x_3 : (subtype(x_1, x_2) \wedge subtype(x_2, x_3) \rightarrow subtype(x_1, x_3))$
- User assertions/invariants
  - $\forall x, y : (x \leq y \rightarrow read(a, x) \leq read(a, y))$

Quantified SMT formulas

- Traditional SMT solvers only deal with quantifier free formulas
- Quantified SMT formulas are useful
- Unsupported/customized theories
  - Type system in ESC/Java, Boogie
  - $\forall x_1, x_2, x_3 : (subtype(x_1, x_2) \land subtype(x_2, x_3) \rightarrow subtype(x_1, x_3))$
- User assertions/invariants
  - $\forall x, y : (x \leq y \rightarrow read(a, x) \leq read(a, y))$
- Many more......
  - Heaps, linked lists,...

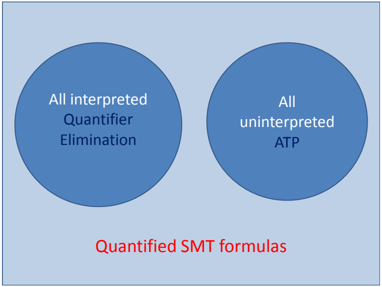
Quantifier Reasoning in SMT: a long-standing challenge

- Mixed uninterpreted and interpreted symbols



Quantifier Reasoning in SMT: a long-standing challenge

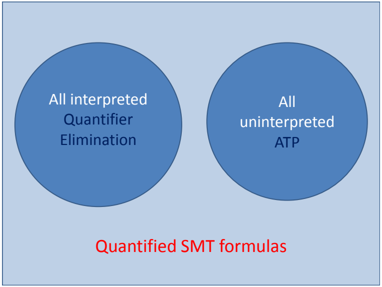
- Mixed uninterpreted and interpreted symbols



- Difficult for a general solution
  - Uninterpreted functions + arithmetic : undecidable

Quantifier Reasoning in SMT: a long-standing challenge

- Mixed uninterpreted and interpreted symbols



- Difficult for a general solution
  - Uninterpreted functions + arithmetic : undecidable
- Solutions
  - Theory resolution, SMT+ATP,...

Instantiation Based Quantifier Reasoning in SMT

The big idea: Given quantified formula $\forall x : P$

Instantiation Based Quantifier Reasoning in SMT

The big idea: Given quantified formula $\forall x : P$

1. Select some ground terms

Instantiation Based Quantifier Reasoning in SMT

The big idea: Given quantified formula $\forall x : P$

1. Select some ground terms
2. Instantiate $\forall x : P$ using ground terms from step 1
   Let P' be the conjunction of instantiations

Instantiation Based Quantifier Reasoning in SMT

The big idea: Given quantified formula $\forall x : P$

1. Select some ground terms
2. Instantiate $\forall x : P$ using ground terms from step 1
   Let P' be the conjunction of instantiations
3. Check P'
   - If P' is unsatisfiable, then $\forall x : P$ is unsatisfiable
   - P' is quantifier free

Instantiation Based Quantifier Reasoning in SMT

The big idea: Given quantified formula $\forall x : P$

1. Select some ground terms
2. Instantiate $\forall x : P$ using ground terms from step 1
   Let P' be the conjunction of instantiations
3. Check P'
   - If P' is unsatisfiable, then $\forall x : P$ is unsatisfiable
   - P' is quantifier free

Example

$f(a) < 1 \wedge (\forall x : f(x) > 2)$

- Select a as the ground term for instantiation
- Instantiate $\forall x : f(x) > 2$ with x substituted by a
- $f(a) < 1 \wedge f(a) > 2$, contradiction

Instantiation Based Quantifier Reasoning in SMT

The big idea: Given quantified formula $\forall x : P$

1. Select some ground terms
2. Instantiate $\forall x : P$ using ground terms from step 1
   Let P' be the conjunction of instantiations
3. Check P'
   - If P' is unsatisfiable, then $\forall x : P$ is unsatisfiable
   - P' is quantifier free

Example

$f(a) < 1 \wedge (\forall x : f(x) > 2)$

- Select a as the ground term for instantiation
- Instantiate $\forall x : f(x) > 2$ with x substituted by a
- $f(a) < 1 \wedge f(a) > 2$, contradiction

What if P' is satisfiable?

Incomplete vs Complete Instantiation

- Instantiation based methods are attractive
- Acceptable performance (E-matching,...)
- Problem:Incompleteness
  - If P' is satisfiable, we can say nothing about the satisfiability of $\forall x : P$

In this talk, we will introduce a series of new fragments that can be solved by complete instantiation

Incomplete vs Complete Instantiation

- Instantiation based methods are attractive
- Acceptable performance (E-matching,...)
- Problem:Incompleteness
  - If P' is satisfiable, we can say nothing about the satisfiability of $\forall x : P$

Can we have a complete method based on instantiation?

Ideally, given F, we would like to have a F^* such that :

- F^* is quantifier free
- F^* is the conjunction of instantiations of F
- F and F^* are equi-satisfiable

In this talk, we will introduce a series of new fragments that can be solved by complete instantiation

Incomplete vs Complete Instantiation

- Instantiation based methods are attractive
- Acceptable performance (E-matching,...)
- Problem:Incompleteness
  - If P' is satisfiable, we can say nothing about the satisfiability of $\forall x : P$

Can we have a complete method based on instantiation?

Ideally, given F, we would like to have a F^* such that :

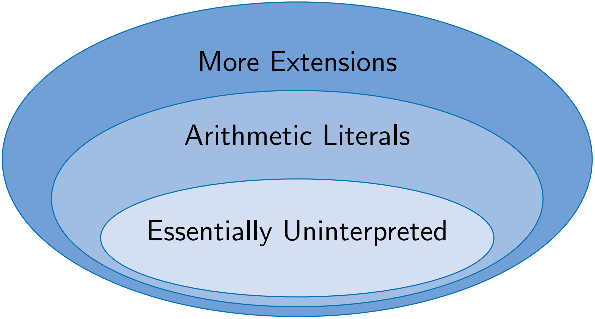
- F^* is quantifier free
- F^* is the conjunction of instantiations of F
- F and F^* are equi-satisfiable

Of course, only possible for formulas in some fragments

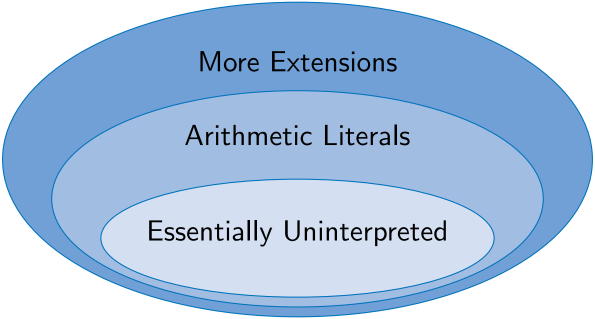
- Array property fragment by Bradley et al
- Linked list by Scott et al

In this talk, we will introduce a series of new fragments that can be solved by complete instantiation

New Fragments for Complete Instantiation



New Fragments for Complete Instantiation



Two key issues for complete instantiation:

- What to instantiate? How to collect ground terms for instantiation?
- Why complete? How to prove?

Notations

- $x, y, x_1, y_1, \ldots$ denotes variables
- $a, b, c, \ldots$ are constants
- $f, g, h, \ldots$ are uninterpreted functions
- $+, -, *, <, \leq, \ldots$ are interpreted arithmetic symbols
- t^M denotes the interpretation of term t in model M

Essentially Uninterpreted Formulas

Definition (Essentially Uninterpreted)

Variables only appear as arguments of uninterpreted functions/predicates

For complete instantiation, two key issues:

- What to instantiate?
- Why complete?

Essentially Uninterpreted Formulas

Definition (Essentially Uninterpreted)

Variables only appear as arguments of uninterpreted functions/predicates

Example

- $f(x) + b > c$, YES

For complete instantiation, two key issues:

- What to instantiate?
- Why complete?

Essentially Uninterpreted Formulas

Definition (Essentially Uninterpreted)

Variables only appear as arguments of uninterpreted functions/predicates

Example

- $f(x) + b > c$, YES
- $f(x + y) > c$, NO

For complete instantiation, two key issues:

- What to instantiate?
- Why complete?

Essentially Uninterpreted Formulas

Definition (Essentially Uninterpreted)

Variables only appear as arguments of uninterpreted functions/predicates

Example

- $f(x) + b > c$, YES
- $f(x + y) > c$, NO
- A formula in pure first order logic is an EU formula

For complete instantiation, two key issues:

- What to instantiate?
- Why complete?

Motivation Example

Is $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$ satisfiable?

Motivation Example

Is $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$ satisfiable?

Yes.

Motivation Example

Is $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$ satisfiable?

Yes.
Why?

Motivation Example

Is $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$ satisfiable?

Yes.

Why?

My SMT solver finds M, a model for $P(f(b)) \wedge Q(f(a)) \wedge P(f(a))$.

Motivation Example

Is $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$ satisfiable?

Yes.
Why?
My SMT solver finds M, a model for $P(f(b)) \wedge Q(f(a)) \wedge P(f(a))$.
So?

Motivation Example

Is $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$ satisfiable?

Yes.

Why?

My SMT solver finds M, a model for $P(f(b)) \wedge Q(f(a)) \wedge P(f(a))$.

So?

Then I construct a M^{π} for $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$.

Motivation Example

Is $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$ satisfiable?

Yes.

Why?

My SMT solver finds M, a model for $P(f(b)) \wedge Q(f(a)) \wedge P(f(a))$.

So?

Then I construct a M^{π} for $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$.

From M?

Motivation Example

Is $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$ satisfiable?

Yes.

Why?

My SMT solver finds M, a model for $P(f(b)) \wedge Q(f(a)) \wedge P(f(a))$.

So?

Then I construct a M^{π} for $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$.

From M?

Yes.

Motivation Example

Is $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$ satisfiable?

Yes.

Why?

My SMT solver finds M, a model for $P(f(b)) \wedge Q(f(a)) \wedge P(f(a))$.

So?

Then I construct a M^{π} for $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$.

From M?

Yes.

Motivation Example

Is $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$ satisfiable?

Motivation Example

Is $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$ satisfiable?

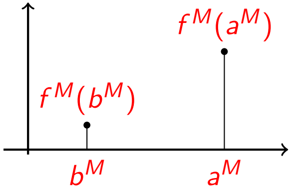
- We use $f^M(a^M)$ to denote the interpretation of $f(a)$ in model M

Motivation Example

Is $P(f(b)) \land Q(f(a)) \land \forall x : P(f(x))$ satisfiable?

- We use $f^M(a^M)$ to denote the interpretation of $f(a)$ in model M

M:

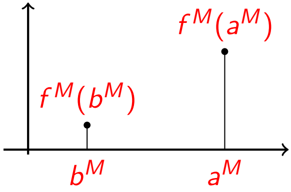


Motivation Example

Is $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$ satisfiable?

- We use $f^M(a^M)$ to denote the interpretation of $f(a)$ in model M
- One solution of M^π is to let $f^{M^\pi}(e)$ be $f^M(a^M)$ for every element e except b^M in the domain (Other solutions possible)

M:

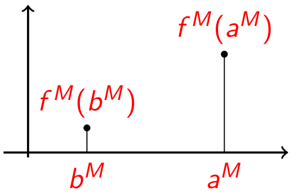


Motivation Example

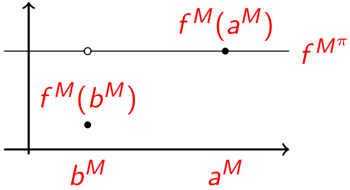
Is $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$ satisfiable?

- We use $f^M(a^M)$ to denote the interpretation of $f(a)$ in model M
- One solution of M^π is to let $f^{M^\pi}(e)$ be $f^M(a^M)$ for every element e except b^M in the domain (Other solutions possible)

M:



M^π:

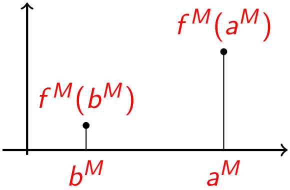


Motivation Example

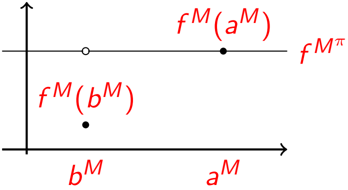
Is $P(f(b)) \wedge Q(f(a)) \wedge \forall x : P(f(x))$ satisfiable?

- We use $f^M(a^M)$ to denote the interpretation of $f(a)$ in model M
- One solution of M^π is to let $f^{M^\pi}(e)$ be $f^M(a^M)$ for every element e except b^M in the domain (Other solutions possible)

M:



M^π:



- We have $P^{M^\pi}(f^{M^\pi}(e))$ holds for all e in the domain

More notations

From now on, assume:

- A formula is a set of CNF clauses
- A clause is universally quantified at outermost scope
- $t[x_1, x_2, ..., x_n]$ means term t may contains variables $x_1, x_2, ..., x_n$
- $t[x/t_1]$ is the result of substituting t_1 for all free occurrences of x
- $t[x_1/s_1, x_2/s_s, ..., x_n/s_n]$ with the obvious meaning
- $t[S_1, S_2, ..., S_n]$ denotes the set $\{t[x_1/s_1, x_2/s_s, ..., x_n/s_n] \mid s_i \in S_i\}$, where S_i are sets of terms

Rules for Collecting Ground Terms for EU Formulas

- S_i and A_f are sets of ground terms
- Details skipped
- S_i contains ground terms for instantiating variable x_i
- A_f contains all ground terms that can appear as argument of f in the result of instantiation

Rules for Collecting Ground Terms for EU Formulas

- S_i and A_f are sets of ground terms
- Details skipped
- S_i contains ground terms for instantiating variable x_i
- A_f contains all ground terms that can appear as argument of f in the result of instantiation
- Suppose $f(t)$ appears in the quantified formula :

t is a ground term	A_f includes t
t is $s[x_1, ..., x_n]$	A_f contains $s[S_1, ..., S_n]$
t is variable x_j	A_f equals to S_j

EU Example

F

$$g(x_1) \leq 0$$
$$g(f(x_2)) + 1 \leq f(x_2)$$
$$f(a) = 0$$

Constraints —— Rules used

Constraints	Rules used
$A_g = S_1$	x_1 is a variable
$f[S_2] \subseteq A_g$	$f(x_2)$ contains variable x_2
$A_f = S_2$	x_2 is a variable
$a \in A_f$	a is ground

EU Example

F

$$g(x_1) \leq 0$$
$$g(f(x_2)) + 1 \leq f(x_2)$$
$$f(a) = 0$$

Constraints —— Rules used

Constraints	Rules used
$A_g = S_1$	x_1 is a variable
$f[S_2] \subseteq A_g$	$f(x_2)$ contains variable x_2
$A_f = S_2$	x_2 is a variable
$a \in A_f$	a is ground

EU Example

F

$g(x_1) \leq 0$
$g(f(x_2)) + 1 \leq f(x_2)$
$f(a) = 0$

Constraints —— Rules used

Constraints	Rules used
$A_g = S_1$	x_1 is a variable
$f[S_2] \subseteq A_g$	$f(x_2)$ contains variable x_2
$A_f = S_2$	x_2 is a variable
$a \in A_f$	a is ground

EU Example

F

$g(x_1) \leq 0$
$g(f(x_2)) + 1 \leq f(x_2)$
$f(a) = 0$

Constraints —— Rules used

$A_g = S_1$	x_1 is a variable
$f[S_2] \subseteq A_g$	$f(x_2)$ contains variable x_2
$A_f = S_2$	x_2 is a variable
$a \in A_f$	a is ground

EU Example

F

$$g(x_1) \leq 0$$
$$g(f(x_2)) + 1 \leq f(x_2)$$
$$f(a) = 0$$

Constraints —— Rules used

$A_g = S_1$	x_1 is a variable
$f[S_2] \subseteq A_g$	$f(x_2)$ contains variable x_2
$A_f = S_2$	x_2 is a variable
$a \in A_f$	a is ground

EU Example

F

$$g(x_1) \leq 0$$
$$g(f(x_2)) + 1 \leq f(x_2)$$
$$f(a) = 0$$

Constraints —— Rules used

$A_g = S_1$	x_1 is a variable
$f[S_2] \subseteq A_g$	$f(x_2)$ contains variable x_2
$A_f = S_2$	x_2 is a variable
$a \in A_f$	a is ground

Ground terms

$$A_g = \{ \quad \}$$
$$A_f = \{a\}$$
$$S_1 = \{ \quad \}$$
$$S_2 = \{ \}$$

EU Example

F

$g(x_1) \leq 0$
$g(f(x_2)) + 1 \leq f(x_2)$
$f(a) = 0$

Constraints —— Rules used

$A_g = S_1$	x_1 is a variable
$f[S_2] \subseteq A_g$	$f(x_2)$ contains variable x_2
$A_f = S_2$	x_2 is a variable
$a \in A_f$	a is ground

Ground terms

$A_g = \{ \quad \}$
$A_f = \{a\}$
$S_1 = \{ \quad \}$
$S_2 = \{ \}$

EU Example

F

$$g(x_1) \leq 0$$
$$g(f(x_2)) + 1 \leq f(x_2)$$
$$f(a) = 0$$

Constraints —— Rules used

Constraints	Rules used
$A_g = S_1$	x_1 is a variable
$f[S_2] \subseteq A_g$	$f(x_2)$ contains variable x_2
$A_f = S_2$	x_2 is a variable
$a \in A_f$	a is ground

Ground terms

$$A_g = \{ \quad \}$$
$$A_f = \{a\}$$
$$S_1 = \{ \quad \}$$
$$S_2 = \{a\}$$

EU Example

F

$$g(x_1) \leq 0$$
$$g(f(x_2)) + 1 \leq f(x_2)$$
$$f(a) = 0$$

Constraints —— Rules used

Constraints	Rules used
$A_g = S_1$	x_1 is a variable
$f[S_2] \subseteq A_g$	$f(x_2)$ contains variable x_2
$A_f = S_2$	x_2 is a variable
$a \in A_f$	a is ground

Ground terms

$$A_g = \{ \quad \}$$
$$A_f = \{a\}$$
$$S_1 = \{ \quad \}$$
$$S_2 = \{a\}$$

EU Example

F

$g(x_1) \leq 0$
$g(f(x_2)) + 1 \leq f(x_2)$
$f(a) = 0$

Constraints —— Rules used

Constraints	Rules used
$A_g = S_1$	x_1 is a variable
$f[S_2] \subseteq A_g$	$f(x_2)$ contains variable x_2
$A_f = S_2$	x_2 is a variable
$a \in A_f$	a is ground

Ground terms

$A_g = \{f(a)\}$
$A_f = \{a\}$
$S_1 = \{\quad\}$
$S_2 = \{a\}$

EU Example

F

$$g(x_1) \leq 0$$
$$g(f(x_2)) + 1 \leq f(x_2)$$
$$f(a) = 0$$

Constraints —— Rules used

$A_g = S_1$	x_1 is a variable
$f[S_2] \subseteq A_g$	$f(x_2)$ contains variable x_2
$A_f = S_2$	x_2 is a variable
$a \in A_f$	a is ground

Ground terms

$$A_g = \{f(a)\}$$
$$A_f = \{a\}$$
$$S_1 = \{\quad\}$$
$$S_2 = \{a\}$$

EU Example

F

$$g(x_1) \leq 0$$
$$g(f(x_2)) + 1 \leq f(x_2)$$
$$f(a) = 0$$

Constraints —— Rules used

$A_g = S_1$	x_1 is a variable
$f[S_2] \subseteq A_g$	$f(x_2)$ contains variable x_2
$A_f = S_2$	x_2 is a variable
$a \in A_f$	a is ground

Ground terms

$$A_g = \{f(a)\}$$
$$A_f = \{a\}$$
$$S_1 = \{f(a)\}$$
$$S_2 = \{a\}$$

EU Example

F

$$g(x_1) \leq 0$$
$$g(f(x_2)) + 1 \leq f(x_2)$$
$$f(a) = 0$$

Constraints —— Rules used

Constraints	Rules used
$A_g = S_1$	x_1 is a variable
$f[S_2] \subseteq A_g$	$f(x_2)$ contains variable x_2
$A_f = S_2$	x_2 is a variable
$a \in A_f$	a is ground

Ground terms

$$A_g = \{f(a)\}$$
$$A_f = \{a\}$$
$$S_1 = \{f(a)\}$$
$$S_2 = \{a\}$$

- The ground terms are collected incrementally

EU Example

F

$$g(x_1) \leq 0$$
$$g(f(x_2)) + 1 \leq f(x_2)$$
$$f(a) = 0$$

Constraints —— Rules used

Constraints	Rules used
$A_g = S_1$	x_1 is a variable
$f[S_2] \subseteq A_g$	$f(x_2)$ contains variable x_2
$A_f = S_2$	x_2 is a variable
$a \in A_f$	a is ground

F^*

$$g(f(a)) \leq 0$$
$$g(f(a)) + 1 \leq f(a)$$
$$f(a) = 0$$

Ground terms

$$A_g = \{f(a)\}$$
$$A_f = \{a\}$$
$$S_1 = \{f(a)\}$$
$$S_2 = \{a\}$$

- The ground terms are collected incrementally

EU Example (Cont.)

F

$g(x_1) \leq 0$
$g(f(x_2)) + 1 \leq f(x_2)$
$f(a) = 0$

F^*

$g(f(a)) \leq 0$
$g(f(a)) + 1 \leq f(a)$
$f(a) = 0$

EU Example (Cont.)

F

$g(x_1) \leq 0$
$g(f(x_2)) + 1 \leq f(x_2)$
$f(a) = 0$

F^*

$g(f(a)) \leq 0$
$g(f(a)) + 1 \leq f(a)$
$f(a) = 0$

M

Let $a = 2$
$f(a) = 0$
$g(0) = -1$

EU Example (Cont.)

F

$g(x_1) \leq 0$
$g(f(x_2)) + 1 \leq f(x_2)$
$f(a) = 0$

F^*

$g(f(a)) \leq 0$
$g(f(a)) + 1 \leq f(a)$
$f(a) = 0$

M^π

Let $a = 2$
$f = \lambda x.0$
$g = \lambda x. - 1$

M

Let $a = 2$
$f(a) = 0$
$g(0) = -1$

Proof of Completeness

- Details skipped
- The big idea:
  - Construct M^π from M by defining interpretation for uninterpreted functions (projection)
  - Assume F^* is satisfiable but F is not
  - Deduce contradiction

Theorem

Given F an essentially uninterpreted formula, F and F^* are equi-satisfiable

Finite F^*

- If F^* is finite, then we have a decision procedure
- When F^* is finite?
  - The set $\{S_i\}$ is stratified
  - Details skipped
  - Better than sorts stratification
    - $f(a) = a \wedge g(f(x)) = f(x)$
    - a and $f(a)$ must be in the same sort

Herbrand Theorem and Finite F^*

Let's assume Q and P are uninterpreted predicates
$Q(f(a)) \wedge \forall x : P(f(x))$

- Herbrand universe
  - $\{a, f(a), f(f(a)), ...\}$

Herbrand Theorem and Finite F^*

Let's assume Q and P are uninterpreted predicates
$Q(f(a)) \wedge \forall x : P(f(x))$

- Herbrand universe
  - $\{a, f(a), f(f(a)), ...\}$
- In the standard Herbrand Theorem, we need to check the satisfiability of
  $Q(f(a)) \wedge P(f(a)) \wedge P(f(f(a))) \wedge P(f(f(f(a))))...$

Herbrand Theorem and Finite F^*

Let's assume Q and P are uninterpreted predicates
$Q(f(a)) \wedge \forall x : P(f(x))$

- Herbrand universe
  - $\{a, f(a), f(f(a)), ...\}$
- In the standard Herbrand Theorem, we need to check the satisfiability of
  $Q(f(a)) \wedge P(f(a)) \wedge P(f(f(a))) \wedge P(f(f(f(a))))...$
- In our theorem, we only need to check $Q(f(a)) \wedge P(f(a))$

Herbrand Theorem and Finite F^*

Let's assume Q and P are uninterpreted predicates
$Q(f(a)) \wedge \forall x : P(f(x))$

- Herbrand universe
  - $\{a, f(a), f(f(a)), ...\}$
- In the standard Herbrand Theorem, we need to check the satisfiability of $Q(f(a)) \wedge P(f(a)) \wedge P(f(f(a))) \wedge P(f(f(f(a))))...$
- In our theorem, we only need to check $Q(f(a)) \wedge P(f(a))$
- Do we have a new decidable class (stratified) in pure first order logic?

Herbrand Theorem and Finite F^*

Let's assume Q and P are uninterpreted predicates
$Q(f(a)) \wedge \forall x : P(f(x))$

- Herbrand universe
  - $\{a, f(a), f(f(a)), ...\}$
- In the standard Herbrand Theorem, we need to check the satisfiability of
  $Q(f(a)) \wedge P(f(a)) \wedge P(f(f(a))) \wedge P(f(f(f(a))))...$
- In our theorem, we only need to check $Q(f(a)) \wedge P(f(a))$
- Do we have a new decidable class (stratified) in pure first order logic?
- Yes, we have

Refinement: Lazy construction of F^*

- F^* may be very big (even infinite)
- By following the rules for collecting ground terms, incrementally construct sequence $F^0 \subset F^1 \subset \ldots$
- If F^k is unsatisfiable, then return unsatisfiable
- If F^k is satisfiable, construct candidate model M^{π^k}
  - If M^{π^k} is a model for all quantified formulas, return satisfiable
  - If M^{π^k} is not a model for all quantified formulas, continue

Refinement: Model Checking Example

How to check if M is a model of a quantified formula $\forall P$?

- Model M
  $h = \lambda x.$ IF $(x = 2)$ THEN 0 ELSE 1
  $g = \lambda x, y.$ IF $(x = 0$ AND $y = 2)$ THEN -1 ELSE 0
- Quantified Formula
  $\forall x1, x2 : g(x1, x2) = 0 \vee h(x2) = 0$

Refinement: Model Checking Example

How to check if M is a model of a quantified formula $\forall P$?

- Model M
  $h = \lambda x.$ IF $(x = 2)$ THEN 0 ELSE 1
  $g = \lambda x, y.$ IF $(x = 0$ AND $y = 2)$ THEN -1 ELSE 0
- Quantified Formula
  $\forall x1, x2 : g(x1, x2) = 0 \lor h(x2) = 0$
- Plug in the model
  $\forall x1, x2 :$ ((IF $(x1 = 0 \land x2 = 2)$ THEN -1 ELSE 0) $=$ 0) $\land$ (IF $(x2 = 2)$ THEN 0 ELSE 1) $= 0$

Refinement: Model Checking Example

How to check if M is a model of a quantified formula $\forall P$?

- Model M
  $h = \lambda x.$ IF $(x = 2)$ THEN 0 ELSE 1
  $g = \lambda x, y.$ IF $(x = 0$ AND $y = 2)$ THEN -1 ELSE 0
- Quantified Formula
  $\forall x1, x2 : g(x1, x2) = 0 \vee h(x2) = 0$
- Plug in the model
  $\forall x1, x2 :$ ((IF $(x1 = 0 \wedge x2 = 2)$ THEN -1 ELSE 0$) =$ 0$) \wedge$ (IF $(x2 = 2)$ THEN 0 ELSE 1$) = 0$
- Check if valid (Send it to a SMT solver)

Refinement: Model Checking Example

How to check if M is a model of a quantified formula $\forall P$?

- Model M
  $h = \lambda x.\ \text{IF } (x = 2) \text{ THEN } 0 \text{ ELSE } 1$
  $g = \lambda x, y.\ \text{IF } (x = 0 \text{ AND } y = 2) \text{ THEN } -1 \text{ ELSE } 0$
- Quantified Formula
  $\forall x1, x2 : g(x1, x2) = 0 \vee h(x2) = 0$
- Plug in the model
  $\forall x1, x2 : ((\text{ IF } (x1 = 0 \wedge x2 = 2) \text{ THEN } -1 \text{ ELSE } 0) = 0) \wedge (\text{IF } (x2 = 2) \text{ THEN } 0 \text{ ELSE } 1) = 0$
- Check if valid (Send it to a SMT solver)
- The above formula is valid and we conclude that M is indeed a model for the quantified formula

Refinement: Model Based Instantiation

- Model checking can be used to select ground terms for instantiation

Example

F	$f(x) \leq 0,\ f(a) = 1, f(b) = -1$
F^0	$f(a) = 1, f(b) = -1$
M^{π^0}	$\{a = 2, b = 3, f = \lambda x.(\text{IF } x = 2 \text{ THEN } 1 \text{ ELSE } -1)\}$

Refinement: Model Based Instantiation

- Model checking can be used to select ground terms for instantiation

Example

F	$f(x) \leq 0,\ f(a) = 1, f(b) = -1$
F^0	$f(a) = 1, f(b) = -1$
M^{π^0}	$\{a = 2, b = 3, f = \lambda x.(\text{IF } x = 2 \text{ THEN } 1 \text{ ELSE } -1)\}$

- Model Checking: $\neg((\text{ IF } s = 2 \text{ THEN } 1 \text{ ELSE } -1) < 0)$
- Satisfiable, with $s = 2$

Refinement: Model Based Instantiation

- Model checking can be used to select ground terms for instantiation

Example

F	$f(x) \leq 0,\ f(a) = 1, f(b) = -1$
F^0	$f(a) = 1, f(b) = -1$
M^{π^0}	$\{a = 2, b = 3, f = \lambda x.(\text{IF } x = 2 \text{ THEN } 1 \text{ ELSE } -1)\}$

- Model Checking: $\neg((\text{ IF } s = 2 \text{ THEN } 1 \text{ ELSE } -1) < 0)$
- Satisfiable, with $s = 2$
- Because $a = 2$ in M^{π^0}
- Instantiate x with a

Refinement: Model Based Instantiation

- Model checking can be used to select ground terms for instantiation

Example

F	$f(x) \leq 0,\ f(a) = 1, f(b) = -1$
F^0	$f(a) = 1, f(b) = -1$
M^{π^0}	$\{a = 2, b = 3, f = \lambda x.(\text{IF } x = 2 \text{ THEN } 1 \text{ ELSE } -1)\}$

- Model Checking: $\neg((\text{ IF } s = 2 \text{ THEN } 1 \text{ ELSE } -1) < 0)$
- Satisfiable, with $s = 2$
- Because $a = 2$ in M^{π^0}
- Instantiate x with a

F^1	$f(a) = 1, f(b) = -1,\ f(a) < 0$

- Unsatisfiable

Infinite F^* and Refutation Complete

When F^* is infinite, is the procedure refutation complete?

F : unsatisfiable, x ranges over integers

$f(x_1) < f(f(x_1))$	f is always increasing
$f(x_2) < a$	f has a up limit
$1 < f(0)$	f has a bottom

F^* : every finite subset is satisfiable

$f(0) < f(f(0)), f(f(0)) < f(f(f(0))), \ldots$

$f(0) < a, f(f(0)) < a, \ldots$

$1 < f(0)$

Infinite F^* and Refutation Complete

When F^* is infinite, is the procedure refutation complete?

- Not always

F : unsatisfiable, x ranges over integers

$f(x_1) < f(f(x_1))$	f is always increasing
$f(x_2) < a$	f has a up limit
$1 < f(0)$	f has a bottom

F^* : every finite subset is satisfiable

$f(0) < f(f(0)), f(f(0)) < f(f(f(0))), \ldots$

$f(0) < a, f(f(0)) < a, \ldots$

$1 < f(0)$

Infinite F^* and Refutation Complete

When F^* is infinite, is the procedure refutation complete?

- Not always

F : unsatisfiable, x ranges over integers

$f(x_1) < f(f(x_1))$	f is always increasing
$f(x_2) < a$	f has a up limit
$1 < f(0)$	f has a bottom

F^* : every finite subset is satisfiable

$f(0) < f(f(0)), f(f(0)) < f(f(f(0))), \ldots$

$f(0) < a, f(f(0)) < a, \ldots$

$1 < f(0)$

- Refutation complete, if we assume the background theory is a (potentially infinite) set of sentences

Infinite F^* and Refutation Complete

When F^* is infinite, is the procedure refutation complete?

- Not always

F : unsatisfiable, x ranges over integers

$f(x_1) < f(f(x_1))$	f is always increasing
$f(x_2) < a$	f has a up limit
$1 < f(0)$	f has a bottom

F^* : every finite subset is satisfiable

$f(0) < f(f(0)), f(f(0)) < f(f(f(0))), \ldots$
$f(0) < a, f(f(0)) < a, \ldots$
$1 < f(0)$

- Refutation complete, if we assume the background theory is a (potentially infinite) set of sentences
- Refutation incomplete, if the background theory is a class of structures (Why? Compactness no longer holds)

Arithmetic Literals

Definition (Arithmetic Literals)

Variables in a CNF clause may appear in literals of the form:
$\neg(x_i \leq x_j), \neg(x_i \leq t), \neg(t \leq x_i), x_i = t$

Arithmetic Literals

Definition (Arithmetic Literals)

Variables in a CNF clause may appear in literals of the form:
$\neg(x_i \leq x_j), \neg(x_i \leq t), \neg(t \leq x_i), x_i = t$

Details skipped, we have similar rules for arithmetic literals

Arithmetic Literals

Definition (Arithmetic Literals)

Variables in a CNF clause may appear in literals of the form:
$\neg(x_i \leq x_j), \neg(x_i \leq t), \neg(t \leq x_i), x_i = t$

Details skipped, we have similar rules for arithmetic literals

Example

$\neg(x_i \leq x_j) \vee A[x_i] \leq A[x_j]$

More Extensions

- Offsets
  - $x_i + 2$
  - $\neg(0 \leq x_i) \vee \neg(x_i \leq n) \vee f(x_i) = g(x_i + 2)$
- Modular equalities
  - $\neg(x_i =_n t)$, means $x_i = t + n * c$, n is an integer and c is a constant
  - $\neg(x_1 =_4 0) \vee (star(x_1) = e)$
- Pseudo-macros

Related Works

- Array property fragment
  - No nested application of uninterpreted functions
  - $P(f(g(x)))$, NO
- "What else is decidable" by Habermehl et al
  - Literals of the form $x_i - x_j \leq c$ are allowed
  - In a clause, at most one literal of the form $f(x_i) - g(x_j) \leq c$ is allowed
  - No other type literal allowed, no nested applications
  - Proof based on a customized automaton, implementation unknown
- Local Theories
  - Certain quantified formulas can be added upon other decidable fragments

Summary

- Several new fragments that can be decided by complete instantiation
- Model checking and model based instantiation
- Conditions for refutation complete
- Z3 was the only solver could return SAT for satisfiable quantified formulas in SMT COMP 2008

Future works:

- Efficient implementation
- More fragments for complete instantiation

Thank you

- Questions?