Finite Quantification in Hierarchic Theorem Proving

Peter Baumgartner
Joshua Bax

NICTA
Australian National University

Uwe Waldmann

max planck institut informatik

Overall Goal

Theorem Proving in Hierarchic Combinations of Specifications

Foreground Specification (FG)
Axioms: Lists, Arrays
Definitions: Length, isSorted

extends

Background Specification (BG)
- Linear integer arithmetic

$\overset{?}{\models}$

Conjecture

Main issue

Quantifiers: complete theorem proving is theoretically impossible
Problem: incompleteness: "no refutation" $\not\Rightarrow$ "countersatisfiable"

Calculi for Hierarchic Reasoning

SMT: DPLL(T) + instantiation heuristics (CVC4, Z3,...)

Model evolution with LIA constraints [B Tinelli 2008, 2011]

Sequent calculus [Rümmer 2008]

Theory instantiation [Korovin 2006]

LASCA [Korovin Voronkov 2007]

Hierarchic superposition

[Bachmair Ganzinger Waldmann 1994, Althaus Weidenbach Kruglov 2009, Weidenbach Kruglov 2012, B Waldmann 2013]

This work

Recover completeness for finitely quantified fragment

Can be used on top of hierarchic superposition and SMT

Hierarchic Specifications

Background (BG) specification consists of

Sorts, e.g. { **int** }

Operators, e.g. { **0, 1, -1, 2, -2, ..., -, +, >, ≥**} Parameters e.g. { **m, n, α** }

Models, e.g. linear integer arithmetic

Foreground (FG) specification extends BG specification by

New sorts, e.g. { **array** }

New operators, e.g.

{ **read: array × int ↦ int,**

write: array × int × int ↦ array,

a: array }

First-order clauses, e.g. array axiom

$\{ \mathsf{read}(\mathsf{write}(a, i, x), i) \approx x,$

$\mathsf{read}(\mathsf{write}(a, i, x), j) \approx \mathsf{read}(a, j) \vee i \approx j \}$

Finite saturation by superposition

Hierarchic Specifications

Array axioms from above

(1) $\text{read}(\text{write}(a, i, x), i) \approx x$

(2) $\text{read}(\text{write}(a, i, x), j) \approx \text{read}(a, j) \vee i \approx j$

Additional clauses

(3) $\text{read}(\text{a}, i) \leq \text{read}(\text{a}, j) \vee \neg(i < j) \vee i \notin [1..1000] \vee j \notin [1..1000]$

// Array **a** is sorted in the range [1..1000]

(4) $1 \leq \text{m} \wedge \text{m} < 1000$

(5) $\text{read}(\text{a}, \text{m}) < \text{read}(\text{a}, \text{m}+1)$

Can't we directly use superposition?

Contributions of this paper

A general method for model computation on top of HSP/SMT, e.g.

$\mathbf{a} \mapsto$	5	5	5	...	5	6	...	6	6	6
	1	2	3	...	50	51	...	998	999	1000
					m					

Hierarchic Specifications

Models of hierarchic specifications

Must satisfy the FG clauses, and
must leave the interpretation of the BG sorts and operators unchanged (*conservative extension*):

- distinct BG elements may not be identified (*no confusion*), and
- no new elements may be added to BG sorts (*no junk*)

Hierarchic superposition calculus (HSP)

Extension of the superposition calculus for hierarchic specifications

Calls BG-solver to decide BG-unsatisfiability of BG clauses

Complete under assumptions: **sufficient completeness**, compactness

The clause set (1)-(5) is not sufficiently complete

Finite saturation does not mean "satisfiable (wrt hierarchic interpretations)"

Sufficient Completeness

Sufficient Completeness

In every model of the FG clauses, every ground FG term that has a BG sort must be equal to some BG term

Example

(3) $\mathrm{read}(a, i) \leq \mathrm{read}(a, j) \vee \neg(i < j) \vee i \notin [1..1000] \vee j \notin [1..1000]$

(5) $\mathrm{read}(a, m) < \mathrm{read}(a, m{+}1)$

is not sufficiently complete, admits junk:

Domain: { 0, -1, 1, -2, 2, ..., NaN }

Interpret: $\mathrm{read}(a, i) \mapsto \mathrm{NaN}$ $(\mathrm{NaN} < \mathrm{NaN}) \mapsto \mathrm{true}$ $(\mathrm{NaN} \leq \mathrm{NaN}) \mapsto \mathrm{true}$

Consequence

Finite saturation of (1) - (5) under HSP does not mean anything

Next goal: recover sufficient completeness for **finitely quantified clauses**

Finitely Quantified Clauses

Definition

A clause C is *finitely quantified* if for every BG variable x occurring under a BG sorted FG operator, C contains a domain declaration of the form $x \notin [\mathrm{l..u}]$, where l and u are concrete integers.

Examples

$$(3) \quad \mathrm{read}(a, i) \leq \mathrm{read}(a, j) \vee \neg(i < j) \vee i \notin [1..1000] \vee j \notin [1..1000]$$

$$(5) \quad \mathrm{read}(a, m) < \mathrm{read}(a, m{+}1)$$

$$f(i{+}1, f(j, 2) + 1) > \boldsymbol{\alpha} + y \vee y > 0 \vee i \notin [1..1000] \vee j \notin [10..100]$$

(Rationale: using “large” domains is useful enough in practice)

Observation: only finitely many ground instances wrt BG sorted FG terms

Sufficient Completeness for Finitely Quantified Clauses

(3) $\text{read}(a, i) \leq \text{read}(a, j) \vee \neg(i < j) \vee i \notin [1..1000] \vee j \notin [1..1000]$

Alternative 1

Force mapping of relevant read-terms to integers by adding unit clauses

$\text{read}(a, 1) \approx 3$

$\text{read}(a, 2) \approx 5$

...

$\text{read}(a, 999) \approx 4$

$\text{read}(a, 1000) \approx 7$

Properties

Recovers sufficient completeness

Soundness and completeness by exhaustive search through mappings

Practically useless

Sufficient Completeness for Finitely Quantified Clauses

(3) $\mathrm{read}(a, i) \leq \mathrm{read}(a, j) \vee \neg(i < j) \vee i \notin [1..1000] \vee j \notin [1..1000]$

Alternative 2

Force mapping of relevant read-terms to integers by adding unit clauses

$\mathrm{read}(a, 1) \approx \alpha_1$

$\mathrm{read}(a, 2) \approx \alpha_2$

...

$\mathrm{read}(a, 999) \approx \alpha_{999}$

$\mathrm{read}(a, 1000) \approx \alpha_{1000}$

where α_i is a fresh parameter

Properties

Recovers sufficient completeness

Supplants outer loop by BG constraint satisfaction problem

Still practically useless

Sufficient Completeness for Finitely Quantified Clauses

(3) $\mathrm{read}(a, i) \leq \mathrm{read}(a, j) \vee \neg(i < j) \vee i \notin [1..1000] \vee j \notin [1..1000]$

Alternative 3 (taken)

Add unit clauses to express **default interpretation** with **exceptions**

$\mathrm{read}(a, i) \approx \alpha_0 \vee i \notin [1..1000] \setminus \{50, 60\}$

$\mathrm{read}(a, 50) \approx \alpha_{50}$

$\mathrm{read}(a, 60) \approx \alpha_{60}$

where α_i is a fresh parameter

Properties

Recovers sufficient completeness

Basis for procedure in paper

- Start with a default interpretation $\mathrm{read}(a, i) \approx \alpha_0 \vee i \notin [1..1000]$
- Modify by adding exceptions like 50, 60 in a conflict-driven way until model found or unsatisfiable

Next: idea of this method

Our Method - First Round

Given clause set $N[\Delta_x]$, where $\Delta_x = [1..1000]$

(1) $f(x) \not\approx x \lor x \notin [1..1000]$

(2) $f(5) \approx 8$

(3) $f(8) \approx 5$

Current set of exceptions $\Pi_x \subseteq \Delta_x$

Initially $\Pi_x = \{\}$

Finite Domain Transformation $M = FD(N[\Delta_x], \Pi_x)$

(f)	$f(x) \approx \alpha_0 \lor x \notin [1..1000]$	default interpretation for $f(x)$ in (1)
(1f)	$\alpha_0 \not\approx x \lor x \notin [1..1000]$	(f) applied to (1)
(2)	$f(5) \approx 8$	
(3)	$f(8) \approx 5$	

Now use HSP to check satisfiability

Our Method - First Round

Finite Domain Transformation $M = FD(N[\Delta_x], \Pi_x)$

(f)	$f(x) \approx \alpha_0 \vee x \notin [1..1000]$
(1f)	$\alpha_0 \not\approx x \vee x \notin [1..1000]$
(2)	$f(5) \approx 8$
(3)	$f(8) \approx 5$

M is unsatisfiable, take { $f(5) \approx \alpha_0$, $f(8) \approx \alpha_0$, (2), (3) }, HSP detects this

Maximal sub-domain $\Gamma_x = [1..7] \subseteq \Delta_x$ recovers satisfiability ($\alpha_0 \mapsto 8$)

(f)	$f(x) \approx \alpha_0 \vee x \notin [1..7]$
(1f)	$\alpha_0 \not\approx x \vee x \notin [1..7]$
(2)	$f(5) \approx 8$
(3)	$f(8) \approx 5$

Satisfiable

(f)	$f(x) \approx \alpha_0 \vee x \notin [1..8]$
(1f)	$\alpha_0 \not\approx x \vee x \notin [1..8]$
(2)	$f(5) \approx 8$
(3)	$f(8) \approx 5$

Unsatisfiable

Sub-domain [1..7] and critical point 8 can be found by binary search

Repair with 8 as next exception

Our Method - Second Round

Given clause set $N[\Delta_x]$

(1) $f(x) \not\approx x \vee x \notin [1..1000]$

(2) $f(5) \approx 8$

(3) $f(8) \approx 5$

Current set of exception points $\Pi_x \subseteq \Delta_x$

$\Pi_x = \{8\}$

Finite Domain Transformation $M = FD(N[\Delta_x], \Pi_x)$

(f)	$f(x) \approx \alpha_0 \vee x \notin [1..1000] \setminus \{8\}$	default interpretation for $f(x)$ in (1)
(f8)	$f(8) \approx \alpha_8$	f at exception point 8
(1f)	$\alpha_0 \not\approx x \vee x \notin [1..1000] \setminus \{8\}$	(f) applied to (1)
(1f8)	$\alpha_8 \not\approx 8$	(f8) applied to (1)
(2)	$f(5) \approx 8$	
(3)	$f(8) \approx 5$	

Satisfiable with $\alpha_0 \mapsto 8$, $\alpha_8 \mapsto 5$. Done

General Method: checkSAT/find

```
1   algorithm checkSAT(N[Δ_x])
2   // returns "satisfiable" or "unsatisfiable"
3   var Π_x := ∅_x // The current set of exceptions
4   while true {
5     let M = FD(N, Π_x)
6     if M is satisfiable return "satisfiable"
7     if M[∅_x] is unsatisfiable return "unsatisfiable"
8     let (x, d) = find(M)
9     Π_x := Π_x[x ↦ Π_x ∪ {d}]
10  }
```

Tacitly assume these checks are effective

Line 7 example, $\Pi_x = \{8\}$

(1) $f(x) > x \lor x \notin \Delta_x$

(2) $f(5) \approx 8$

(3) $f(8) \approx 5$

$\xrightarrow{\text{FD}}$

$\cancel{M}$ $\quad$ $M[\emptyset_x]$

$\cancel{f(x) \approx \alpha_0 \lor x \not\in \Delta_x \setminus \{8\}}$ $\quad$ $f(5) \approx 8$

$f(8) \approx \alpha_8$ $\quad$ $f(8) \approx 5$

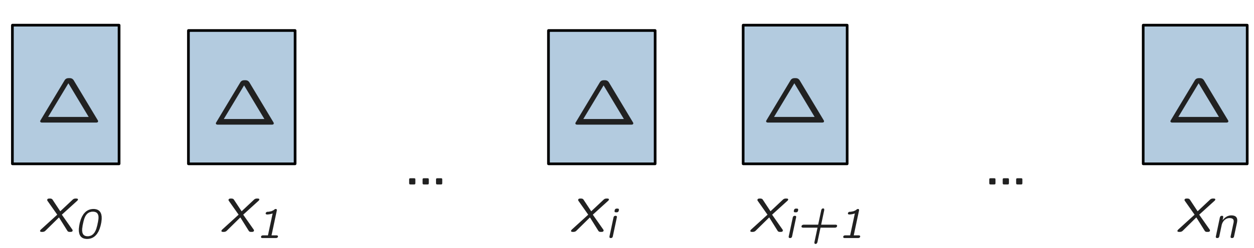
$\cancel{\alpha_0 > x \lor x \not\in \Delta_x \setminus \{8\}}$

$\alpha_8 > 8$

General Method: checkSat/find

```
algorithm find(M[Δ_x])
// returns a pair (x, d) such that x ∈ x and d ∈ Δ_x \ Π_x
let (x_1, ..., x_n) = x
for i = 1 to n {
  if M[∅_(x_1,...,x_i) · Δ_(x_{i+1},...,x_n)] is satisfiable {
     let Γ ⊆ Δ_{x_i} and d ∈ Γ such that
        M[∅_(x_1,...,x_{i-1}) · Γ_{x_i} · Δ_(x_{i+1},...,x_n)] is unsatisfiable and
        M[∅_(x_1,...,x_{i-1}) · (Γ \ {d})_{x_i} · Δ_(x_{i+1},...,x_n)] is satisfiable
     return (x_i, d)
  }
}
```

We know:

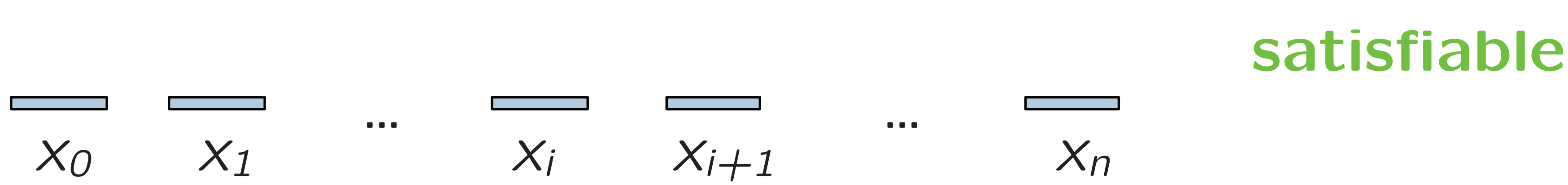


unsatisfiable

General Method: checkSat/find

algorithm find($M[\varDelta_{\mathbf{x}}]$)
// returns a pair (x, d) such that $x \in \mathbf{x}$ and $d \in \varDelta_x \setminus \varPi_x$
let $(x_1, \ldots, x_n) = \mathbf{x}$
for $i = 1$ **to** n $\{$
  if $M[\emptyset_{(x_1,\ldots,x_i)} \cdot \varDelta_{(x_{i+1},\ldots,x_n)}]$ is satisfiable $\{$
    let $\varGamma \subseteq \varDelta_{x_i}$ and $d \in \varGamma$ such that
      $M[\emptyset_{(x_1,\ldots,x_{i-1})} \cdot \varGamma_{x_i} \cdot \varDelta_{(x_{i+1},\ldots,x_n)}]$ is unsatisfiable and
      $M[\emptyset_{(x_1,\ldots,x_{i-1})} \cdot (\varGamma \setminus \{d\})_{x_i} \cdot \varDelta_{(x_{i+1},\ldots,x_n)}]$ is satisfiable
    return (x_i, d)
    $\}$
$\}$

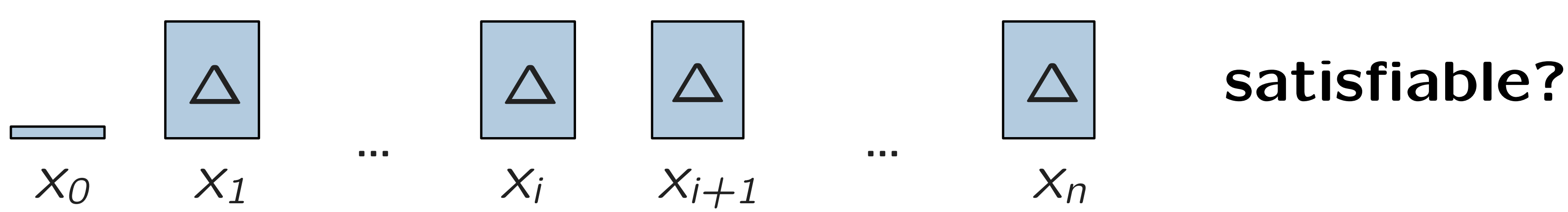
We know:



General Method: checkSat/find

algorithm find($M[\Delta_{\mathbf{x}}]$)
// returns a pair (x, d) such that $x \in \mathbf{x}$ and $d \in \Delta_x \setminus \Pi_x$
let $(x_1, \ldots, x_n) = \mathbf{x}$
for $i = 1$ **to** n {
  if $M[\emptyset_{(x_1,\ldots,x_i)} \cdot \Delta_{(x_{i+1},\ldots,x_n)}]$ is satisfiable {
    let $\Gamma \subseteq \Delta_{x_i}$ and $d \in \Gamma$ such that
      $M[\emptyset_{(x_1,\ldots,x_{i-1})} \cdot \Gamma_{x_i} \cdot \Delta_{(x_{i+1},\ldots,x_n)}]$ is unsatisfiable and
      $M[\emptyset_{(x_1,\ldots,x_{i-1})} \cdot (\Gamma \setminus \{d\})_{x_i} \cdot \Delta_{(x_{i+1},\ldots,x_n)}]$ is satisfiable
    return (x_i, d)
    }
}

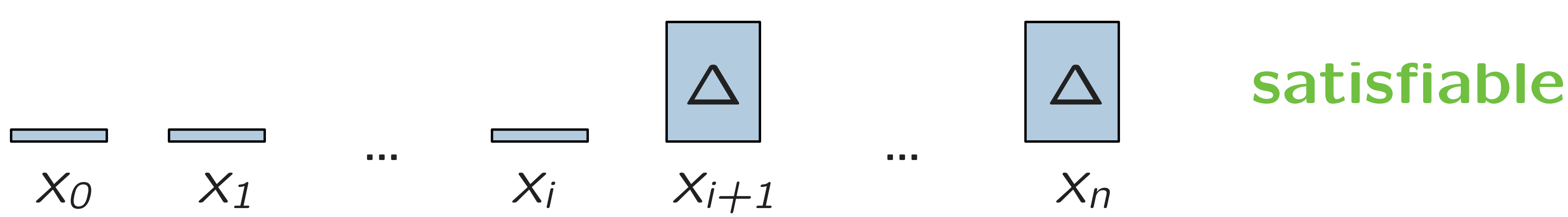
Search:



General Method: checkSat/find

algorithm find($M[\Delta_{\mathbf{x}}]$)
// returns a pair (x, d) such that $x \in \mathbf{x}$ and $d \in \Delta_x \setminus \Pi_x$
let $(x_1, \ldots, x_n) = \mathbf{x}$
for $i = 1$ **to** n {
  if $M[\emptyset_{(x_1,\ldots,x_i)} \cdot \Delta_{(x_{i+1},\ldots,x_n)}]$ is satisfiable {
    let $\Gamma \subseteq \Delta_{x_i}$ and $d \in \Gamma$ such that
      $M[\emptyset_{(x_1,\ldots,x_{i-1})} \cdot \Gamma_{x_i} \cdot \Delta_{(x_{i+1},\ldots,x_n)}]$ is unsatisfiable and
      $M[\emptyset_{(x_1,\ldots,x_{i-1})} \cdot (\Gamma \setminus \{d\})_{x_i} \cdot \Delta_{(x_{i+1},\ldots,x_n)}]$ is satisfiable
    return (x_i, d)
    }
}

Search:

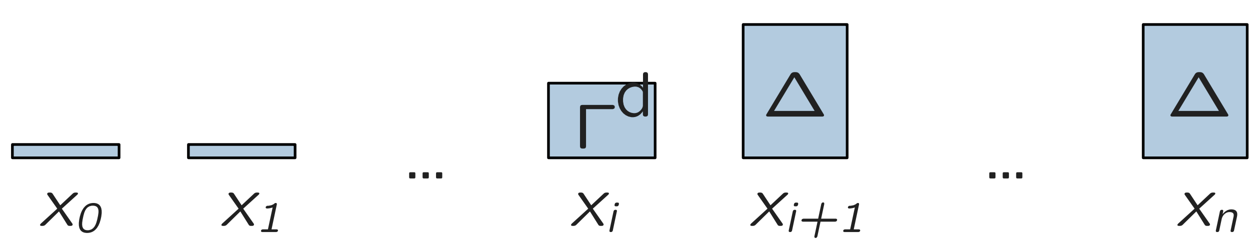


General Method: checkSat/find

algorithm find($M[\Delta_{\mathbf{x}}]$)
// returns a pair (x, d) such that $x \in \mathbf{x}$ and $d \in \Delta_x \setminus \Pi_x$
let $(x_1, \ldots, x_n) = \mathbf{x}$
for $i = 1$ **to** n $\{$
  if $M[\emptyset_{(x_1,\ldots,x_i)} \cdot \Delta_{(x_{i+1},\ldots,x_n)}]$ is satisfiable $\{$
    let $\Gamma \subseteq \Delta_{x_i}$ and $d \in \Gamma$ such that
      $M[\emptyset_{(x_1,\ldots,x_{i-1})} \cdot \Gamma_{x_i} \cdot \Delta_{(x_{i+1},\ldots,x_n)}]$ is unsatisfiable and
      $M[\emptyset_{(x_1,\ldots,x_{i-1})} \cdot (\Gamma \setminus \{d\})_{x_i} \cdot \Delta_{(x_{i+1},\ldots,x_n)}]$ is satisfiable
    return (x_i, d)
  $\}$
$\}$

Use binary search on Δ_x

Search:



unsatisfiable ("just")

Main Result

Assume that HSP decides satisfiability of clause sets $M = \mathrm{FD}(N[\mathbf{\Delta}_x], \mathbf{\Pi}_x)$

Theorem

For any set N of finitely quantified clauses, checkSAT(N) terminates with the correct result "satisfiable" or "unsatisfiable" for N.

Moreover, if the result is "unsatisfiable" then the non-domain restricted version of N is unsatisfiable, which is obtained from N by removing from all clauses in N all domain declarations $x \notin \mathbf{\Delta}_x$. $\square$

(1) $\mathsf{f}(x) > x \cancel{\vee x \notin \mathbf{\Delta}_x}$

(2) $\mathsf{f}(5) \approx 8$

(3) $\mathsf{f}(8) \approx 5$

Some Experiments

Array Example

$$\mathrm{read}(\mathrm{write}(a, i, x), i) \approx x$$

$$\mathrm{read}(\mathrm{write}(a, i, x), j) \approx \mathrm{read}(a, j) \vee i \approx j$$

$$\mathrm{read}(\mathrm{a}, i) \leq \mathrm{read}(\mathrm{a}, j) \vee \neg(i < j) \vee$$
$$\quad i \notin [1..1000] \vee j \notin [1..1000]$$

$$1 \leq \mathrm{m} \wedge \mathrm{m} < 1000$$

$$\mathrm{read}(\mathrm{a}, \mathrm{m}) < \mathrm{read}(\mathrm{a}, \mathrm{m}{+}1)$$

$\lvert\Delta\rvert$	#Iter	#TP	Time
10	3	15	2.3
20	3	17	2.6
50	3	19	2.8
100	3	21	2.8
200	3	23	2.8
500	3	25	2.9
1000	3	27	3.0
2000	3	29	3.0
5000	3	33	3.5

$m = 2$ variable occurrences
$n = 1000$ size of (largest) domain
Each iteration requires about $m + \mathrm{ld}(n) = 2 + 10$ prover calls in find

By contrast, ground instantiation gives $n^m = 10^6$ instances

Some Experiments

Running Example

$$\mathsf{f}(x) \not\approx x \vee x \notin \Delta$$

$$\mathsf{f}(5) \approx 8$$

$$\mathsf{f}(8) \approx 5$$

$\lvert\Delta\rvert$	**#Iter**	**#TP**	**Time**
10	2	5	<1
20	2	6	<1
50	2	8	<1
100	2	9	<1
200	2	10	<1
500	2	11	<1
1000	2	12	<1
2000	2	13	<1
5000	2	15	<1

Some Experiments

#	Problem	$\|\Delta\|$	#Iter	#TP	Time
1	$\mathsf{f}(x) > 1 + y \vee y < 0 \vee x \notin \Delta$	any	1	1	<1
2	$\mathsf{g}(x) \approx x \vee \mathsf{g}(x) \approx x + 1 \vee \neg(x \geq 0)$	10	9	32	5.5
	$\mathsf{g}(x) \approx -x \vee \neg(x < 0)$	20	20	86	55
	$\mathsf{f}(x) < \mathsf{g}(x) \vee x \notin \Delta$				
3	$\mathsf{f}(x_1, x_2, x_3, x_4) > x_1 + x_2 + x_3 + x_4 \vee$ $x_1 \notin \Delta \vee x_2 \notin \Delta \vee x_3 \notin \Delta \vee x_4 \notin \Delta$	any	1	1	<1

Problem 1: default interpretation enough

Problem 2: have sufficient completeness wrt. g, need to treat only f

Problem 3: default interpretation enough

Z3: does not solve problems 1 and 2, solves problem 3 up to $|\Delta| = 60$

Z3: solves running examples above

Conclusions

Presented a method on top of HSP for theorem (dis)proving under finitely quantified variables

Main idea: conflict-driven repair of default interpretation

Requires BG reasoner for EA-fragment

Meant to scale well with domain size

However worst case needs exceptions "everywhere"

Can (sometimes) be used on top of SMT

Eliminate first non-ground definitions by exhaustive superposition

Generalizes instantiation heuristics known from SMT

Return "unsatisfiable" or "satisfiable (over finite domain)"

(Supposing underlying prover terminates)

Future work

Instantiation-based methods as special case of the method here?