Towards Learning Infinite SMT Models

Mikoláš Janota and Bartosz Piotrowski and Karel Chvalovský

Czech Technical University in Prague

CTU CZECH TECHNICAL UNIVERSITY IN PRAGUE

MŠMT MINISTRY OF EDUCATION, YOUTH AND SPORTS

11 September 2023, Nancy, France

Intro

- A model is infinite iff the universe is infinite.

Intro

- A model is infinite iff the universe is infinite.
- Example: semigroups

$$(\forall xyz)((x * y) * z = x * (y * z))$$

Intro

- A model is infinite iff the universe is infinite.
- Example: semigroups

$$(\forall xyz)((x * y) * z = x * (y * z))$$

- $(\{0, 1\}, + \bmod 2)$ — finite semigroup

Intro

- A model is infinite iff the universe is infinite.
- Example: semigroups

$$(\forall xyz)((x * y) * z = x * (y * z))$$

- $(\{0, 1\}, + \bmod 2)$ — finite semigroup
- $(\mathbb{N}, +)$ — infinite semigroup

Motivation

- Models as counterexamples to:

Motivation

- Models as counterexamples to:
  - incorrect programs

Motivation

- Models as counterexamples to:
  - incorrect programs
  - incorrect theorems

Motivation

- Models as counterexamples to:
  - incorrect programs
  - incorrect theorems
- Structures of interesting properties
  "Find a semigroup not a group!"

Motivation

- Models as counterexamples to:
  - incorrect programs
  - incorrect theorems
- Structures of interesting properties
  "Find a semigroup not a group!"
- Some properties only for infinite models

Motivation

- Models as counterexamples to:
  - incorrect programs
  - incorrect theorems
- Structures of interesting properties
  "Find a semigroup not a group!"
- Some properties only for infinite models
- In Satisfiability Modulo Theories infinite models often required for functions + integers + quantifiers (UFLIA).

SMT Models: Constants

```
(declare-fun c () Int)
(declare-fun d () Int)
(assert (< c d))
(check-sat)
(get-model)
```

$$c < d$$

SMT Models: Constants

```
(declare-fun c () Int)
(declare-fun d () Int)
(assert (< c d))
(check-sat)
(get-model)
```

$$c < d$$

```
:!z3 ex1.smt2
sat
(
  (define-fun d () Int
    1)
  (define-fun c () Int
    0)
)
```

$$c = 0, d = 1$$

SMT Models: Functions

```
(declare-fun f (Int) Int)
(assert (< (f 0) (f 1)))
(check-sat)
(get-model)
```

$$f(0) < f(1)$$

SMT Models: Functions

```
(declare-fun f (Int) Int)
(assert (< (f 0) (f 1)))
(check-sat)
(get-model)
```

$$f(0) < f(1)$$

```
:!z3 ex2.smt2
sat
(
  (define-fun f ((x!0 Int)) Int
    (ite (= x!0 1) 1
      0))
)
```

$$fx \triangleq (1 \text{ if } x = 1 \text{ else } 0)$$

SMT Models: Quantifiers

```
(declare-fun f (Int) Int)
(assert (forall ((x Int))
              (<= (f x) x)))
(check-sat)
(get-model)
```

$$(\forall x)(fx \leq x)$$

SMT Models: Quantifiers

```
(declare-fun f (Int) Int)
(assert (forall ((x Int))
            (<= (f x) x)))
(check-sat)
(get-model)
```

```
:!z3 ex3.smt2
sat
(
  (define-fun f ((x!0 Int)) Int
    x!0)
)
```

$$(\forall x)(fx \leq x)$$

$$fx \triangleq x$$

SMT Models: Quantifiers

```
(declare-fun f (Int) Int)
(assert (forall ((x Int))
            (< (f x) x)))
(check-sat)
(get-model)
```

$$(\forall x)(fx < x)$$

SMT Models: Quantifiers

```
(declare-fun f (Int) Int)
(assert (forall ((x Int))
          (< (f x) x)))
(check-sat)
(get-model)
```

$$(\forall x)(fx < x)$$

```
:!z3 -T:60 ex4.smt2
timeout
```

Not Solved!

Learn infinite models from finite ones?

Background: MBQI

- Model-Based Guided Quantifier Instantiation
  [Ge and de Moura, 2009]

Background: MBQI

- Model-Based Guided Quantifier Instantiation [Ge and de Moura, 2009]

For $\forall x \phi$ construct a sequence of:

- candidate models M_i

Background: MBQI

- Model-Based Guided Quantifier Instantiation
  [Ge and de Moura, 2009]

For $\forall x \phi$ construct a sequence of:

- candidate models M_i
- counterexample instantiations σ_i

Background: MBQI

- Model-Based Guided Quantifier Instantiation [Ge and de Moura, 2009]

For $\forall x \phi$ construct a sequence of:

- candidate models M_i
- counterexample instantiations σ_i
- s.t. $M_i \models \bigwedge_{j \in 1..i-1} \phi[x/\sigma_j]$

Background: MBQI

- Model-Based Guided Quantifier Instantiation
  [Ge and de Moura, 2009]

For $\forall x \phi$ construct a sequence of:

- candidate models M_i
- counterexample instantiations σ_i
- s.t. $M_i \models \bigwedge_{j \in 1..i-1} \phi[x/\sigma_j]$
- s.t. $M_i \not\models \phi[x/\sigma_i]$

Example

$$(\forall x)(fx > x)$$

$\bigwedge_{j\in 1..i-1} \phi[x/\sigma_j]$	M_i	σ_i
true	$fx \triangleq 0$	$x \mapsto 0$

Example

$$(\forall x)(fx > x)$$

$\bigwedge_{j\in 1..i-1} \phi[x/\sigma_j]$	M_i	σ_i
$f(0) > 0$		

Example

$$(\forall x)(fx > x)$$

$\bigwedge_{j \in 1..i-1} \phi[x/\sigma_j]$	M_i	σ_i
$f(0) > 0$	$fx \triangleq 1$	$x \mapsto 1$

Example

$$(\forall x)(fx > x)$$

$\bigwedge_{j\in 1..i-1} \phi[x/\sigma_j]$	M_i	σ_i
$f(0) > 0$	$fx \triangleq (x = 0\,?\,1 : 2)$	$x \mapsto 2$
$f(1) > 1$		

Example

$$(\forall x)(fx > x)$$

$\bigwedge_{j\in 1..i-1}\phi[x/\sigma_j]$	M_i	σ_i
$f(0) > 0$	$fx \triangleq (x = 0\,?\,1 : (x = 1\,?\,2 : 3))$	
$f(1) > 1$		
$f(2) > 2$		

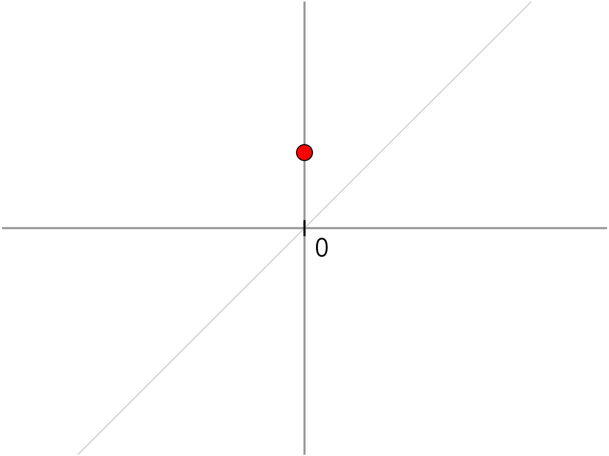
Example

$$(\forall x)(fx > x)$$

$\bigwedge_{j\in 1..i-1} \phi[x/\sigma_j]$	M_i	σ_i
$f(0) > 0$	$fx \triangleq (x = 0\,?\,1$ … $1\,?\,2\,:\,3))$	
$f(1) > 1$		
$f(2) > 2$		

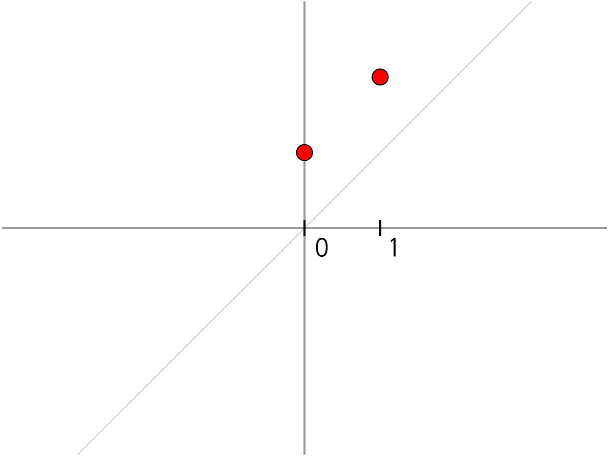
Example: Generalization

$$(\forall x)(fx > x)$$



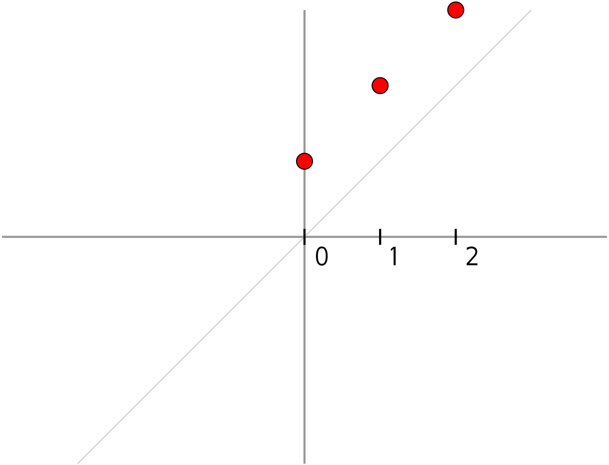
Example: Generalization

$$(\forall x)(fx > x)$$



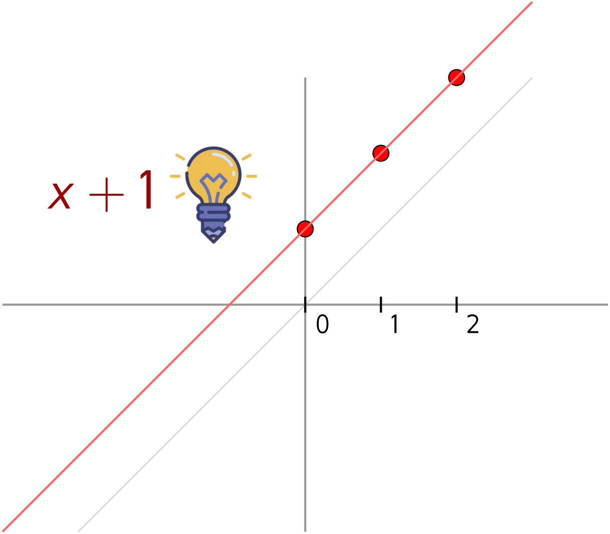
Example: Generalization

$$(\forall x)(fx > x)$$



Example: Generalization

$$(\forall x)(fx > x)$$



Generalization for Functions

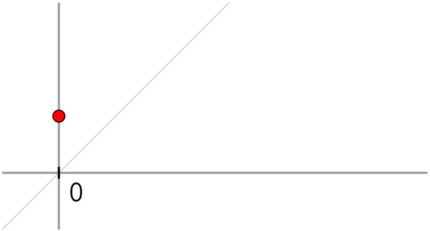
- Sort points lexicographically
- Keep the same hyper-plane as long as possible
- Otherwise start a new hyper-plane.

Generalization for Functions

- Sort points lexicographically
- Keep the same hyper-plane as long as possible
- Otherwise start a new hyper-plane.
- For LIA: linear Diophantine equations, solvable in polynomial time

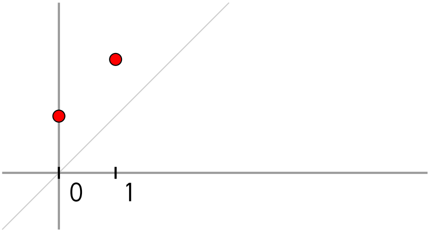
Generalization for Functions

- Sort points lexicographically
- Keep the same hyper-plane as long as possible
- Otherwise start a new hyper-plane.
- For LIA: linear Diophantine equations, solvable in polynomial time



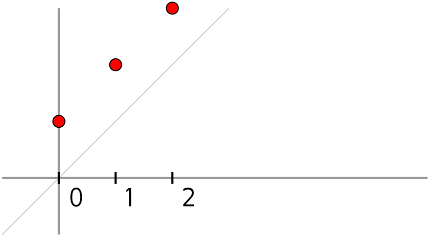
Generalization for Functions

- Sort points lexicographically
- Keep the same hyper-plane as long as possible
- Otherwise start a new hyper-plane.
- For LIA: linear Diophantine equations, solvable in polynomial time



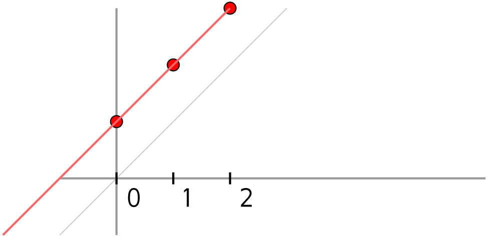
Generalization for Functions

- Sort points lexicographically
- Keep the same hyper-plane as long as possible
- Otherwise start a new hyper-plane.
- For LIA: linear Diophantine equations, solvable in polynomial time



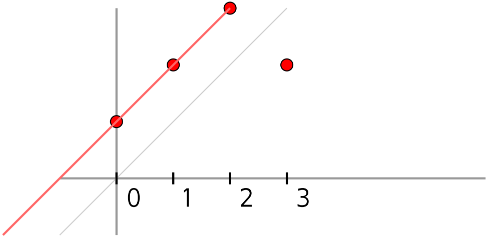
Generalization for Functions

- Sort points lexicographically
- Keep the same hyper-plane as long as possible
- Otherwise start a new hyper-plane.
- For LIA: linear Diophantine equations, solvable in polynomial time



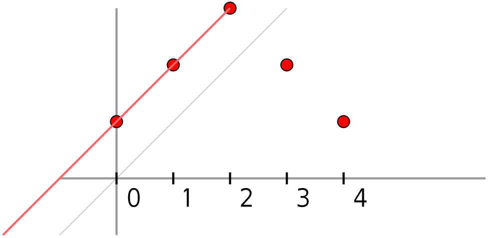
Generalization for Functions

- Sort points lexicographically
- Keep the same hyper-plane as long as possible
- Otherwise start a new hyper-plane.
- For LIA: linear Diophantine equations, solvable in polynomial time



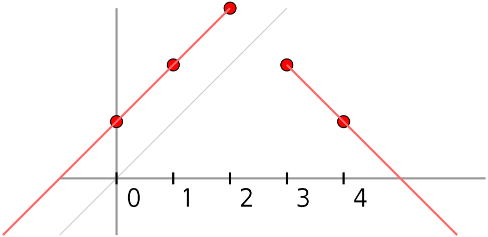
Generalization for Functions

- Sort points lexicographically
- Keep the same hyper-plane as long as possible
- Otherwise start a new hyper-plane.
- For LIA: linear Diophantine equations, solvable in polynomial time



Generalization for Functions

- Sort points lexicographically
- Keep the same hyper-plane as long as possible
- Otherwise start a new hyper-plane.
- For LIA: linear Diophantine equations, solvable in polynomial time

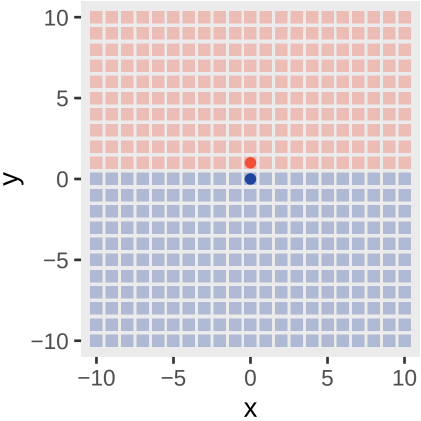


Generalization for Predicates

- Split recursively by hyper-planes
- until all positive or all negative

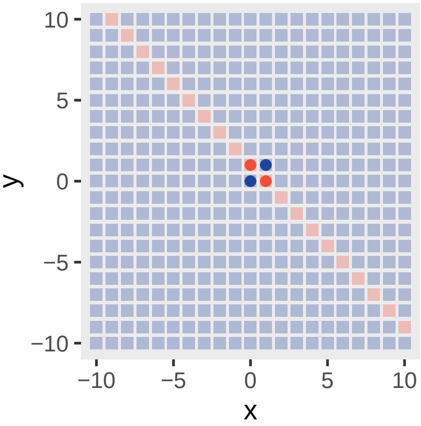
Generalization for Predicates

- Split recursively by hyper-planes
- until all positive or all negative



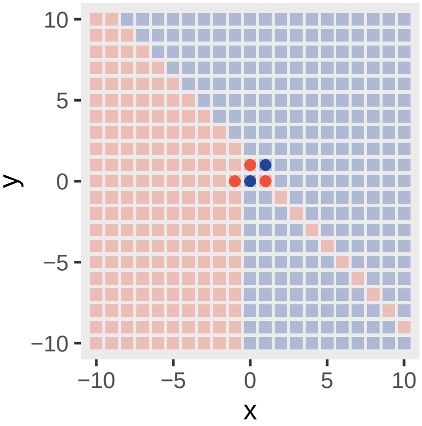
Generalization for Predicates

- Split recursively by hyper-planes
- until all positive or all negative



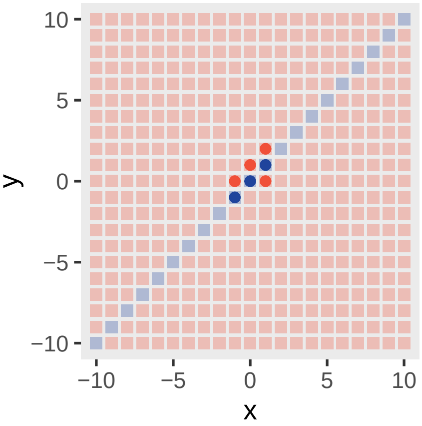
Generalization for Predicates

- Split recursively by hyper-planes
- until all positive or all negative



Generalization for Predicates

- Split recursively by hyper-planes
- until all positive or all negative



Results UFLIA

- Implemented in cvc5
- Run on [Janota et al., 2023]

solver	SAT	UNSAT	total
standard MBQI	18843	7863	26706
ours smart MBQI	31977	7863	39840
Z3	28380	7482	35862

Summary

- Guessing infinite models for MBQI,

Summary

- Guessing infinite models for MBQI,
- Currently for UFLIA

Summary

- Guessing infinite models for MBQI,
- Currently for UFLIA
- Fast: without losing performance on UNSAT.

Summary

- Guessing infinite models for MBQI,
- Currently for UFLIA
- Fast: without losing performance on UNSAT.

What next?

- Tighter integration with ground theory solver?

Summary

- Guessing infinite models for MBQI,
- Currently for UFLIA
- Fast: without losing performance on UNSAT.

What next?

- Tighter integration with ground theory solver?
- More theories?

Summary

- Guessing infinite models for MBQI,
- Currently for UFLIA
- Fast: without losing performance on UNSAT.

What next?

- Tighter integration with ground theory solver?
- More theories?
- Non-linear shapes?

Ge, Y. and de Moura, L. M. (2009).
Complete instantiation for quantified formulas in satisfiabiliby modulo theories.
In *Computer Aided Verification CAV*, pages 306–320.

Janota, M., Brown, C. E., and Kaliszyk, C. (2023).
A benchmark for infinite models in smt.
In *8th Conference on Artificial Intelligence and Theorem Proving, AITP 2023*.