Towards Learning Infinite SMT Models

Mikoláš Janota and Bartosz Piotrowski
and Karel Chvalovský

Czech Technical University in Prague

11 September 2023, Nancy, France
A model is infinite iff the universe is infinite.
A model is infinite iff the universe is infinite.

Example: semigroups

\[(\forall xyz)((x \ast y) \ast z = x \ast (y \ast z))\]
A model is infinite iff the universe is infinite.

**Example:** semigroups

\[(\forall xyz)((x \ast y) \ast z = x \ast (y \ast z))\]

\[\{0, 1\}, + \text{ mod } 2\] — finite semigroup
A model is **infinite** iff the universe is infinite.

**Example:** semigroups

\[(\forall xyz)((x \ast y) \ast z = x \ast (y \ast z))\]

- \([\{0,1\}, + \mod 2]\) — finite semigroup
- \([\mathbb{N}, +]\) — infinite semigroup
Motivation

- Models as counterexamples to:
  - Incorrect programs
  - Incorrect theorems
  - Structures of interesting properties

"Find a semigroup not a group!"

Some properties only for finite models

In Satisfiability Modulo Theories finite models often required for functions + integers + quantifiers (UFLIA).

Janota
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

\[
\begin{align*}
\text{(declare-fun } c \text{ () Int)} \\
\text{(declare-fun } d \text{ () Int)} \\
\text{(assert } (< c d)) \\
\text{(check-sat)} \\
\text{(get-model)} \\
\end{align*}
\]

\[
c < d
\]

\[
\begin{align*}
\text{!z3 ex1.smt2} \\
\text{sat} \\
\text{ (} \\
\text{ (define-fun } d \text{ () Int } 1) \\
\text{ (define-fun } c \text{ () Int } 0) \\
\text{)} \\
\end{align*}
\]

\[
c = 0, d = 1
\]
SMT Models: Functions

\[(\text{declare-fun } f \ (\text{Int}) \ \text{Int})\]
\[(\text{assert } (\lt (f \ 0) (f \ 1)))\]
\[(\text{check-sat})\]
\[(\text{get-model})\]

\[f(0) < f(1)\]
SMT Models: Functions

\[
\begin{align*}
    f(0) & < f(1) \\
    f_x & \triangleq (1 \text{ if } x = 1 \text{ else } 0)
\end{align*}
\]
SMT Models: Quantifiers

\[(\forall x)(fx \leq x)\]
SMT Models: Quantifiers

\((\forall x)(f x \leq x)\)

\[ f x \triangleq x \]
SMT Models: Quantifiers

(∀x)(fx < x)
SMT Models: Quantifiers

(∀x)(fx < x)

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

:!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 $\sigma_i$
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 $\sigma_i$
- 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 $\sigma_i$

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

\[(\forall x)(fx > x)\]

\[\Lambda_{j \in 1..i-1} \phi[x/\sigma_j] \quad M_i \quad \sigma_i\]

\[\text{true} \quad fx \triangleq 0 \quad x \mapsto 0\]
Example

\[(\forall x)(fx > x)\]

\[\bigwedge_{j \in 1..i-1} \phi[x/\sigma_j] \quad M_i \quad \sigma_i\]

\[f(0) > 0\]
Example

\[(\forall x)(fx > x)\]

\[\bigwedge_{j \in 1..i-1} \phi[x/\sigma_j]\]

\[f(0) > 0\]

\[f_x \triangleq 1\]

\[x \mapsto 1\]
Example

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

\(\wedge_{j \in 1..i-1} \phi[x/\sigma_j]\)

\(f(0) > 0\)

\(f(1) > 1\)

\(f_x \triangleq (x = 0 ? 1 : 2)\)

\(x \mapsto 2\)
Example

\[(\forall x)(fx > x)\]

\[\bigwedge_{j \in 1..i-1} \phi[x/\sigma_j] \quad \text{Mi} \quad \sigma_i\]

\[
\begin{align*}
  f(0) & > 0 & fx & \triangleq (x = 0 \ ? 1 \ : (x = 1 \ ? 2 \ : 3)) \\
  f(1) & > 1 \\
  f(2) & > 2 
\end{align*}
\]
Example

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

\[ \land_{j \in 1..i-1} \phi[x/\sigma_j] \]

\[ f(0) > 0 \quad fx \triangleq (x = 0 \ ? 1 \ ? 2 : 3)) \]

\[ f(1) > 1 \]

\[ f(2) > 2 \]

Déjà Vu
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

![Diagram showing generalization process]
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]

<table>
<thead>
<tr>
<th>solver</th>
<th>SAT</th>
<th>UNSAT</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>standard MBQI</td>
<td>18843</td>
<td>7863</td>
<td>26706</td>
</tr>
<tr>
<td>ours smart MBQI</td>
<td>31977</td>
<td>7863</td>
<td>39840</td>
</tr>
<tr>
<td>Z3</td>
<td>28380</td>
<td>7482</td>
<td>35862</td>
</tr>
</tbody>
</table>
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?