Reasoning About Set Comprehensions

Edmund S. L. Lam  Iliano Cervesato
sllam@qatar.cmu.edu  iliano@cmu.edu

Carnegie Mellon University

Supported by grant NPRP 09-667-1-100, Effective Programming for Large Distributed Ensembles

SMT’14  Vienna, Austria, July 2014
Outline

1. Introduction

2. Encoding $SC(LIA)$ into $U+LIA$

3. Implementation and Future Work
Automated support for reasoning about sets (multisets)
- Cardinality constraints
  [Piskac and Kuncak, 2010, Suter et al., 2011]
- Aggregate constraints [Leino and Monahan, 2009]

But what about set comprehensions?
- Is \( \{10, 20, 30\} \models \{x \mid x < 4\}_{x \in X} \) satisfiable?

- Is \( \{x \mid x < 4\}_{x \in X} \cap \{x \mid x \geq 4\}_{x \in X} \neq \emptyset \) satisfiable?

We want automated support for reasoning about set comprehensions as well!
Automated support for reasoning about sets (multisets)
- Cardinality constraints
  - [Piskac and Kuncak, 2010, Suter et al., 2011]
- Aggregate constraints [Leino and Monahan, 2009]

But what about set comprehensions?
- Is \( \{10, 20, 30\} \equiv \{x \cdot 10 \mid x < 4\}_{x \in X} \) satisfiable?
  - Yes! Possible solutions: \( X = \{1, 2, 3\} \) or \( X = \{1, 2, 3, 4\} \) or . . .
- Is \( \{x \mid x < 4\}_{x \in X} \cap \{x \mid x \geq 4\}_{x \in X} \neq \emptyset \) satisfiable?

We want automated support for reasoning about set comprehensions as well!
Motivation

- Automated support for reasoning about sets (multisets)
  - Cardinality constraints
    [Piskac and Kuncak, 2010, Suter et al., 2011]
  - Aggregate constraints [Leino and Monahan, 2009]

- But what about set comprehensions?
  - Is \( \{10, 20, 30\} \subseteq \{x \times 10 \mid x < 4\}_{x \in X} \) satisfiable?
    Yes! Possible solutions: \( X = \{1, 2, 3\} \) or \( X = \{1, 2, 3, 4\} \) or ... 
  - Is \( \{x \mid x < 4\}_{x \in X} \cap \{x \mid x \geq 4\}_{x \in X} \neq \emptyset \) satisfiable?
    No! No such \( X \) exists

- We want automated support for reasoning about set comprehensions as well!
This work, at a Glance

Reasoning about set comprehensions:

- Source language: set comprehensions over some base theory
  \( Th \rightarrow SC(Th) \)

- We encode formulas of \( SC(Th) \) into formulas of \( Th \), plus an uninterpreted domain \( U \rightarrow U+Th \)
  - Uninterpreted domain \( U \) represents the domain of sets of \( Th \)
  - \( U+Th \) formulas are fed to an off-the-shelf SAT checker (e.g., Z3)

For simplicity, we demonstrate this encoding for \( Th = LIA \) (Linear Integer Arithmetic’s)
Outline

1. Introduction
2. Encoding SC(LIA) into U+LIA
3. Implementation and Future Work
**SC(LIA) and U+LIA**

**SC(LIA):** Set Comprehensions over Linear Integer Arithmetic

<table>
<thead>
<tr>
<th>Arithmetic Term</th>
<th>$t ::= x \mid v \mid t \text{ op } t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic Formula</td>
<td>$T ::= t \doteq t \mid t &lt; t \mid \neg T \mid T \land T$</td>
</tr>
<tr>
<td>Set Term</td>
<td>$s ::= X \mid {t} \mid {t \mid T}_{x \in s} \mid s \cup s \mid s \cap s \mid s \setminus s$</td>
</tr>
<tr>
<td>Set Formula</td>
<td>$S ::= t \in s \mid s \doteq s \mid s \subseteq s \mid \neg S \mid S \land S$</td>
</tr>
</tbody>
</table>

**U+LIA:** Linear Integer Arithmetic and Uninterpreted Sets

<table>
<thead>
<tr>
<th>Arithmetic Term</th>
<th>$t ::= x \mid v \mid t \text{ op } t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic Formula</td>
<td>$T ::= t \doteq t \mid t &lt; t$</td>
</tr>
<tr>
<td>Uninterpreted Set Term</td>
<td>$s ::= X$</td>
</tr>
<tr>
<td>Uninterpreted Set Formula</td>
<td>$S ::= t \in s$</td>
</tr>
<tr>
<td>Formula</td>
<td>$F, C ::= S \mid T \mid \neg F \mid F \land F \mid \exists x . F \mid \forall x . F$</td>
</tr>
</tbody>
</table>

- **Set comprehensions:** $\{t_x \mid T_x\}_{x \in s}$
  - $t_x$: range pattern
  - $T_x$: guard condition
  - $s$: comprehension domain
- Scope of $x$ is $t_x$ and $T_x$
Encoding $\text{SC}(\text{LIA})$ into $\text{U+LIA}$ — an Example

- $\llbracket S \rrbracket = F$ is the encoding in $\text{U+LIA}$ of $\text{SC}(\text{LIA})$ formula $S$

An example:

$\llbracket \{10, 20, 30\} \rrbracket \equiv \{x \times 10 \mid x < 4\}_{x \in X} \|

= \{\ldots\}$
Encoding \( SC(LIA) \) into \( U+LIA \) — an Example

- \( \lbrack S \rbrack = F \) is the encoding in \( U+LIA \) of \( SC(LIA) \) formula \( S \)
- An example:
  \[
  \lbrack \{10, 20, 30\} \rbrack = \{x \ast 10 \mid x < 4\}_{x \in X}
  \]
  \[
  \forall y. y \hat{\in} X_2 \leftrightarrow (y \hat{=} 10 \lor y \hat{=} 20 \lor y \hat{=} 30) \quad - \quad F_1 : X_2 = \{10, 20, 30\}
  \]

- Encode set term \( \{10, 20, 30\} \) as uninterpreted variable \( X_2 \)
- Relation \( \hat{\in} \) is treated as an uninterpreted binary predicate
- Formula \( F_1 \) provides the interpretation of \( X_2 \) and \( \hat{\in} \)
Encoding \( SC(LIA) \) into \( U+LIA \) — an Example

- \( \llbracket S \rrbracket = F \) is the encoding in \( U+LIA \) of \( SC(LIA) \) formula \( S \)

An example:

\[
\llbracket \{10, 20, 30\} \rrbracket = \{x * 10 \mid x < 4\}_{x \in X}
\]

\[
= \begin{cases} 
\forall y. y \in X_2 \leftrightarrow (y = 10 \lor y = 20 \lor y = 30) & - F_1 : X_2 = \{10, 20, 30\} \\
\forall x. (x \times 10 \in X_3) \leftrightarrow (x \in X \land x < 4) & - F_2 : X_3 = \{x \times 10 \mid x < 4\}_{x \in X}
\end{cases}
\]

- Same for \( \{x \times 10 \mid x < 4\}_{x \in X} \) with \( X_3 \) and \( F_2 \)

- Given \( \{t_x \mid T_x\}_{x \in S} \), we encode with \( X_3 \)

\[
\forall x. (t_x \in X_3) \leftrightarrow (x \in s \land T_x)
\]

- This is a special case though . . .
Encoding $SC(LIA)$ into $U+LIA$ — an Example

- $\models S = F$ is the encoding in $U+LIA$ of $SC(LIA)$ formula $S$

- An example:

$\models \{10, 20, 30\} = \{x \ast 10 \mid x < 4\}_{x \in X}$

$= \begin{cases} 
\forall y. y \in X_2 \leftrightarrow (y \doteq 10 \lor y \doteq 20 \lor y \doteq 30) & - F_1 : X_2 = \{10, 20, 30\} \\
\forall x. (x \ast 10 \in X_3) \leftrightarrow (x \in X \land x < 4) & - F_2 : X_3 = \{x \ast 10 \mid x < 4\}_{x \in X} \\
\forall z. z \in X_2 \leftrightarrow z \in X_3 & - F_3 : X_2 = X_3 
\end{cases}$

- Finally, $F_3$ states that $X_2$ and $X_3$ are extensionally equal
Encoding \( SC(LIA) \) into \( U+LIA \) — an Example

- \( \models S \models F \) is the encoding in \( U+LIA \) of \( SC(LIA) \) formula \( S \)

An example:

\[
\begin{align*}
\models \& \{10, 20, 30\} &\models \{x \times 10 \mid x < 4\}_{x \in X} \models \\
& \begin{cases}
\forall y. \ y \in X_2 \leftrightarrow (y \models 10 \lor y \models 20 \lor y \models 30) \\
\forall x. \ (x \times 10 \in X_3) \leftrightarrow (x \in X \land x < 4) \\
\forall z. \ z \in X_2 \leftrightarrow z \in X_3
\end{cases} \\
& = \\
& \begin{cases}
F_1 : X_2 = \{10, 20, 30\} \\
F_2 : X_3 = \{x \times 10 \mid x < 4\}_{x \in X} \\
F_3 : X_2 = X_3
\end{cases}
\end{align*}
\]

- \( \{10, 20, 30\} \models \{x \times 10 \mid x < 4\}_{x \in X} \) is satisfiable
- iff \( F_1 \land F_2 \land F_3 \) is satisfiable (i.e., \( M \models F_1 \land F_2 \land F_3 \))
- \( M \models F_1 \land F_2 \land F_3 \) can be checked by many off-the-shelf SMT solvers (e.g., Z3)

\( M \models F_1 \land F_2 \land F_3 \)
Set Comprehension Encoding (Special Case)

- This was a special case

Encode \( \{ t_x \mid T_x \}_{x \in s} \) as \( \forall x. (t_x \in X_3) \leftrightarrow (x \in s \land T_x) \)

- Here’s why:

\[
\{0, 2\} \equiv \{x \% 3 \mid T\}_{x \in \{3, 6, 8\}} \\
= \begin{cases} 
\forall y. y \in X_2 \leftrightarrow (y \equiv 0 \lor y \equiv 2) \\
\forall x. (x \% 3 \in X_3) \leftrightarrow (x \in X_4) \\
\forall z. z \in X_4 \leftrightarrow (z \equiv 3 \lor z \equiv 6 \lor z \equiv 8) \\
\forall w. w \in X_2 \leftrightarrow w \in X_3 
\end{cases}
\]

- \( F_1 : X_2 = \{0, 2\} \)
- \( F_2 : X_3 = \{x \% 3 \mid T\}_{x \in X_4} \)
- \( F_3 : X_4 = \{3, 6, 8\} \)
- \( F_3 : X_2 = X_3 \)
This was a special case

Encode \( \{ t_x \mid T_x \}_{x \in s} \) as \( \forall x. (t_x \in X_3) \iff (x \in s \land T_x) \)

Here’s why:

\[
\prod \{0, 2\} \equiv \{ x \% 3 \mid \top \}_{x \in \{3, 6, 8\}}
\]

\[
= \begin{cases} 
\forall y. y \in X_2 \iff (y \div 0 \lor y \div 2) & - F_1 : X_2 = \{0, 2\} \\
\forall x. (x \% 3 \in X_3) \iff (x \in X_4) & - F_2 : X_3 = \{x \% 3 \mid \top\}_{x \in X_4} \\
\forall z. z \in X_4 \iff (z \div 3 \lor z \div 6 \lor z \div 8) & - F_3 : X_4 = \{3, 6, 8\} \\
\forall w. w \in X_2 \iff w \in X_3 & - F_3 : X_2 = X_3
\end{cases}
\]

We expect \( \{0, 2\} \equiv \{ x \% 3 \mid \top \}_{x \in \{3, 6, 8\}} \) to be satisfiable . . .

but \( F_1 \land F_2 \land F_3 \land F_4 \) is not!
Set Comprehension Encoding (Special Case)

- This was a special case

Encode \( \{ t_x \mid T_x \}_{x \in s} \) as \( \forall x. (t_x \in X_3) \leftrightarrow (x \in s \land T_x) \)

- Here’s why:

\[
\prod \{0, 2\} \equiv \{ x \% 3 \mid T \}_{x \in \{3, 6, 8\}}
\]

\[
= \begin{cases} 
\forall y. y \in X_2 \leftrightarrow (y \div 0 \lor y \div 2) & - F_1 : X_2 = \{0, 2\} \\
\forall x. (x \% 3 \dot{\in} X_3) \leftrightarrow (x \dot{\in} X_4) & - F_2 : X_3 = \{ x \% 3 \mid T \}_{x \in X_4} \\
\forall z. z \dot{\in} X_4 \leftrightarrow (z \div 3 \lor z \div 6 \lor z \div 8) & - F_3 : X_4 = \{3, 6, 8\} \\
\forall w. w \dot{\in} X_2 \leftrightarrow w \dot{\in} X_3 & - F_3 : X_2 = X_3
\end{cases}
\]

- The problem: \( F_2 \) is “malfunctioning” on the \( \rightarrow \) case
- A counterexample \( 9 \% 3 = 0 \), but \( 0 \notin X_3 \not\rightarrow 9 \notin X_4 \)
Set Comprehension Encoding (In General)

- Encode comprehensions with $\forall x. (t_x \in X) \leftrightarrow (x \in s \land T_x)$ on work if $t_x$ is injective.

- In general, comprehension patterns are encoding with two $U+LIA$ formulas

$$\prod\{t_x \mid T_x\}_{x \in X} = \begin{cases} X' \\
\text{such that} \\
\forall x. (x \in X \land T_x) \rightarrow t_x \in X' \\
\forall z. z \in X' \rightarrow \exists x. (z \equiv t_x \land x \in X \land T_x) \\
\text{− } F_{max} \\
\text{− } F_{rg}
\end{cases}$$

- $F_{max}$ enforces maximality: Every domain value in $X$ has a corresponding value in $X'$

- $F_{rg}$ enforces range restriction: Every member of $X'$ has a corresponding value in $X$
Given a $SC(LIA)$ formula $S$, is $\mathcal{M} \models \llbracket S \rrbracket$ decidable?
- Most likely not.
- $U+LIA$ is not decidable [Halpern, 1991].

Nonetheless still useful:
- Compiler optimization for CHR with comprehensions [Lam and Cervesato, 2014]

See paper for details!
Outline

1. Introduction
2. Encoding \textit{SC(LIA)} into \textit{U+LIA}
3. Implementation and Future Work
Implementation

- A lightweight Python library:
  - Built on top of Z3 SMT Solver [De Moura and Bjørner, 2008]
  - Simple combinator library to write $SC(Th_{Z3})$ formulas, where $Th_{Z3}$ consist of Z3 base types.
  - Translates $SC(Th_{Z3})$ formulas to $U+Th_{Z3}$ formulas, which are SAT checked by Z3

- Available for download at:
  
  https://github.com/sllam/pysetcomp
Future Work

- Set comprehension is great, but what about *multiset* comprehensions?
- Possible approach: Multisets as arrays (map elements to multiplicity)

\[
M = X_1 \triangleq \{x \times 10 \mid x \leq 3\} \cup X_2
\]

\[
\prod M = \left\{ \begin{array}{l}
\forall x, m. \ (X_2[x] = m \land m > 0 \land x \leq 3) \rightarrow X_1[x \times 10] = m \\
\forall z, m. \ (X_1[z] = m \land m > 0) \rightarrow \exists x. (z = x \times 10 \land x \leq 3 \land X_2[x] = m)
\end{array} \right.
\]

- Future work:
  - “Multisets as arrays” works only for injective functions
  - Requires a *reduce* sum on array values
Conclusion

- We have developed a framework for automated reasoning about formulas on set comprehensions over some base term theory $Th$ (i.e., $SC(Th)$).
- Encodes $SC(Th)$ into $U+Th$ formulas, which can be SAT checked by off-the-shelf SMT solvers.
- Implemented a light-weight Python library, built on top of Z3.
- Available for download at: https://github.com/sllam/pysetcomp
Z3: An Efficient SMT Solver.

Presburger Arithmetic with Unary Predicates is $\Pi^1_1$ Complete.

Lam, E. S. L. and Cervesato, I. (2014).
Optimized Compilation of Multiset Rewriting with Comprehensions (Full-Version).

Reasoning about Comprehensions with First-Order SMT Solvers.

MUNCH — Automated Reasoner for Sets and Multisets.
In IJCAR’10, volume 6173 of Lecture Notes in Computer Science, pages 149–155. Springer.

Sets with Cardinality Constraints in Satisfiability Modulo Theories.
In Verification, Model Checking, and Abstract Interpretation, pages 403–418. Springer.