Reasoning About Set Comprehensions

Edmund S. L. Lam — sllam@qatar.cmu.edu

Iliano Cervesato — 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

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 $\{10, 20, 30\} \doteq \{x * 10 \mid x < 4\}_{x \dot{\in} X}$ satisfiable?
  - Is $\{x \mid x < 4\}_{x \dot{\in} X} \cap \{x \mid x \geq 4\}_{x \dot{\in} X} \neq \varnothing$ 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 $\{10, 20, 30\} \doteq \{x * 10 \mid x < 4\}_{x \dot{\in} X}$ satisfiable?
      Yes! Possible solutions: $X = \{1, 2, 3\}$ or $X = \{1, 2, 3, 4\}$ or $\ldots$
    - Is $\{x \mid x < 4\}_{x \dot{\in} X} \cap \{x \mid x \geq 4\}_{x \dot{\in} X} \neq \varnothing$ 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 $\{10, 20, 30\} \doteq \{x * 10 \mid x < 4\}_{x \dot\in X}$ satisfiable?
    Yes! Possible solutions: $X = \{1, 2, 3\}$ or $X = \{1, 2, 3, 4\}$ or $\ldots$
  - Is $\{x \mid x < 4\}_{x \dot\in X} \cap \{x \mid x \geq 4\}_{x \dot\in X} \neq \varnothing$ 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 — $SC(Th)$
- We encode formulas of $SC(Th)$ into formulas of Th, plus an uninterpreted domain U — 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

$SC(LIA)$ and $U{+}LIA$

$\boxed{SC(LIA)\text{: Set Comprehensions over Linear Integer Arithmetic}}$

Arithmetic Term	t	$::=$	$x \mid v \mid t \ op \ t$
Arithmetic Formula	T	$::=$	$t \doteq t \mid t < t \mid \neg T \mid T \wedge T$
Set Term	s	$::=$	$X \mid \{\bar{t}\} \mid \boxed{\{t \mid T\}_{x \dot{\in} s}} \mid s \cup s \mid s \cap s \mid s \setminus s$
Set Formula	S	$::=$	$t \dot{\in} s \mid s \doteq s \mid s \subseteq s \mid \neg S \mid S \wedge S$

$\boxed{U{+}LIA\text{: Linear Integer Arithmetic and Uninterpreted Sets}}$

Arithmetic Term	t	$::=$	$x \mid v \mid t \ op \ t$
Arithmetic Formula	T	$::=$	$t \doteq t \mid t < t$
Uninterpreted Set Term	s	$::=$	X
Uninterpreted Set Formula	S	$::=$	$t \dot{\in} s$
Formula	F, C	$::=$	$S \mid T \mid \neg F \mid F \wedge F \mid \exists x.F \mid \forall x.F$

- Set comprehensions: $\{t_x \mid T_x\}_{x \dot{\in} s}$
  - t_x: range pattern
  - T_x: guard condition
  - s: comprehension domain
- Scope of x is t_x and T_x

Encoding $SC(LIA)$ into U+LIA — an Example

- $[\![S]\!] = F$ is the encoding in U+LIA of $SC(LIA)$ formula S
- An example:

$$[\![\{10, 20, 30\} \doteq \{x * 10 \mid x < 4\}_{x \dot{\in} X}]\!]$$

$$= \left\{ \right.$$

Encoding $SC(LIA)$ into U+LIA — an Example

- $[\![S]\!] = F$ is the encoding in U+LIA of $SC(LIA)$ formula S
- An example:

$$[\![\{10, 20, 30\} \doteq \{x * 10 \mid x < 4\}_{x \dot{\in} X}]\!] = \begin{cases} \forall y.\ y \dot{\in} X_2 \leftrightarrow (y \doteq 10 \vee y \doteq 20 \vee y \doteq 30) \quad - \quad F_1 : X_2 = \{10, 20, 30\} \\ \\ \end{cases}$$

- Encode set term $\{10, 20, 30\}$ as uninterpreted variable X_2
- Relation $\dot{\in}$ is treated as an uninterpreted binary predicate
- Formula F_1 provides the interpretation of X_2 and $\dot{\in}$

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

- $[\![S]\!] = F$ is the encoding in $U{+}LIA$ of $SC(LIA)$ formula S
- An example:

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

- Same for $\{x * 10 \mid x < 4\}_{x \dot\in X}$ with X_3 and F_2
- Given $\{t_x \mid T_x\}_{x \dot\in s}$, we encode with X_3
  $$\forall x.\ (t_x \dot\in X_3) \leftrightarrow (x \dot\in s \land T_x)$$
- This is a special case though ...

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

- $[\![S]\!] = F$ is the encoding in $U{+}LIA$ of $SC(LIA)$ formula S
- An example:

$$[\![\{10,20,30\} \doteq \{x * 10 \mid x < 4\}_{x \dot\in X}]\!]$$
$$= \begin{cases} \forall y.\ y \dot\in X_2 \leftrightarrow (y \doteq 10 \vee y \doteq 20 \vee y \doteq 30) & - \ F_1 : X_2 = \{10,20,30\} \\ \forall x.\ (x * 10 \dot\in X_3) \leftrightarrow (x \dot\in X \wedge x < 4) & - \ F_2 : X_3 = \{x * 10 \mid x < 4\}_{x \dot\in X} \\ \forall z.\ z \dot\in X_2 \leftrightarrow z \dot\in X_3 & - \ F_3 : X_2 = X_3 \end{cases}$$

- Finally, F_3 states that X_2 and X_3 are extensionally equal

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

- $[\![S]\!] = F$ is the encoding in $U{+}LIA$ of $SC(LIA)$ formula S
- An example:

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

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

Set Comprehension Encoding (Special Case)

- This was a special case

$$\text{Encode } \{t_x \mid T_x\}_{x \dot{\in} s} \text{ as } \forall x.\ (t_x \dot{\in} X_3) \leftrightarrow (x \dot{\in} s \wedge T_x)$$

- Here's why:

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

Set Comprehension Encoding (Special Case)

- This was a special case

$$\text{Encode } \{t_x \mid T_x\}_{x \dot{\in} s} \text{ as } \forall x.\ (t_x \dot{\in} X_3) \leftrightarrow (x \dot{\in} s \wedge T_x)$$

- Here's why:

$$\llbracket \{0,2\} \doteq \{x\%3 \mid \top\}_{x \dot{\in} \{3,6,8\}} \rrbracket$$

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

- We expect $\{0,2\} \doteq \{x\%3 \mid \top\}_{x \dot{\in} \{3,6,8\}}$ to be satisfiable ...
- but $F_1 \wedge F_2 \wedge F_3 \wedge F_4$ is not!

Set Comprehension Encoding (Special Case)

- This was a special case

$$\text{Encode } \{t_x \mid T_x\}_{x \dot{\in} s} \text{ as } \forall x.\ (t_x \dot{\in} X_3) \leftrightarrow (x \dot{\in} s \wedge T_x)$$

- Here's why:

$$\llbracket \{0,2\} \doteq \{x\%3 \mid \top\}_{x \dot{\in} \{3,6,8\}} \rrbracket$$
$$= \begin{cases} \forall y.\ y \dot{\in} X_2 \leftrightarrow (y \doteq 0 \vee y \doteq 2) & - F_1 : X_2 = \{0,2\} \\ \forall x.\ (x\%3 \dot{\in} X_3) \leftrightarrow (x \dot{\in} X_4) & - F_2 : X_3 = \{x\%3 \mid \top\}_{x \dot{\in} X_4} \\ \forall z.\ z \dot{\in} X_4 \leftrightarrow (z \doteq 3 \vee z \doteq 6 \vee z \doteq 8) & - F_3 : X_4 = \{3,6,8\} \\ \forall w.\ w \dot{\in} X_2 \leftrightarrow w \dot{\in} X_3 & - F_3 : X_2 = X_3 \end{cases}$$

- The problem: F_2 is "malfunctioning" on the $\rightarrow$ case
- A counterexample $9\%3 = 0$, but $0 \dot{\in} X_3 \not\rightarrow 9 \dot{\in} X_4$

Set Comprehension Encoding (In General)

- Encode comprehensions with $\forall x.\ (t_x \mathrel{\dot\in} X_3) \leftrightarrow (x \mathrel{\dot\in} s \wedge T_x)$ on work if t_x is *injective*.
- In general, comprehension patterns are encoding with *two* $U{+}LIA$ formulas

$$
[\![\{t_x \mid T_x\}_{x \dot\in X}]\!] = \begin{cases} X' \\ \text{such that} \\ \quad \forall x.\ (x \mathrel{\dot\in} X \wedge T_x) \rightarrow t_x \mathrel{\dot\in} X' & - F_{max} \\ \quad \forall z.\ z \mathrel{\dot\in} X' \rightarrow \exists x.\ (z \doteq t_x \wedge x \mathrel{\dot\in} X \wedge T_x) & - F_{rg} \end{cases}
$$

- 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

Encoding $SC(LIA)$ into U+LIA

- Given a $SC(LIA)$ formula S, is $\mathcal{M} \models [\![S]\!]$ 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

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)

$$
\begin{aligned}
M &= X_1 \doteq \wr x * 10 \mid x \leq 3 \int_{x \dot\in X_2} \\
[\![M]\!] &= \begin{cases} \forall x, m.\ (X_2[x] = m \wedge m > 0 \wedge x \leq 3) \rightarrow X_1[x * 10] = m \\ \forall z, m.\ (X_1[z] = m \wedge m > 0) \rightarrow \exists x.(z = x * 10 \wedge x \leq 3 \wedge X_2[x] = m) \end{cases}
\end{aligned}
$$

- 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`

Bibliography

De Moura, L. and Bjørner, N. (2008).
Z3: An Efficient SMT Solver.
In *Proc. of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'08/ETAPS'08, pages 337–340, Berlin, Heidelberg. Springer-Verlag.

Halpern, J. Y. (1991).
Presburger Arithmetic with Unary Predicates is Π^1_1 Complete.
Journal of Symbolic Logic, 56:56–2.

Lam, E. S. L. and Cervesato, I. (2014).
Optimized Compilation of Multiset Rewriting with Comprehensions (Full-Version).
Technical Report CMU-CS-14-119, Carnegie Mellon University.

Leino, K. R. M. and Monahan, R. (2009).
Reasoning about Comprehensions with First-Order SMT Solvers.
In *In Proc. of the 2009 ACM symposium on Applied Computing*, pages 615–622. ACM.

Piskac, R. and Kuncak, V. (2010).
MUNCH — Automated Reasoner for Sets and Multisets.
In *IJCAR'10*, volume 6173 of *Lecture Notes in Computer Science*, pages 149–155. Springer.

Suter, P., Steiger, R., and Kuncak, V. (2011).
Sets with Cardinality Constraints in Satisfiability Modulo Theories.
In *Verification, Model Checking, and Abstract Interpretation*, pages 403–418. Springer.