Efficient Automated Reasoning about Sets and Multisets with Cardinality Constraints *

Ruzica Piskac[0000−0002−3267−0776]

Yale University
ruzica.piskac@yale.edu

Abstract. When reasoning about container data structures that can hold duplicate elements, multisets are the obvious choice for representing the data structure abstractly. However, the decidability and complexity of constraints on multisets has been much less studied and understood than for constraints on sets. In this presentation, we outline an efficient decision procedure for reasoning about multisets with cardinality constraints. We describe how to translate, in linear time, multisets constraints to constraints in an extension of quantifier-free linear integer arithmetic, which we call LIA*. LIA* extends linear integer arithmetic with unbounded sums over values satisfying a given linear arithmetic formula. We show how to reduce a LIA* formula to an equisatisfiable linear integer arithmetic formula. However, this approach requires an explicit computation of semilinear sets and in practice it scales poorly even on simple benchmarks. We then describe a recent more efficient approach for checking satisfiability of LIA*. The approach is based on the use of under- and over-approximations of LIA* formulas. This way we avoid the space overhead and explicitly computing semilinear sets. Finally, we report on our prototype tool which can efficiently reason about sets and multisets formulas with cardinality constraints.

Keywords: Multisets · Cardinality constraints · Linear interger arithmetic.

1 Introduction

In the verification of container data structures one often needs to reason about sets of objects – for example, abstracting the content of a container data structure as a set. The need for cardinality constraints naturally arises in order to reason about the number of the elements in the data structure. We have all witnessed to the success of the BAPA logic [4,5] that was, among others, used for verification of distributed algorithms [1].

Similarly, when reasoning about container data structures that can hold duplicate elements, multisets are the obvious choice of an abstraction. Multisets are collections of objects where an element can occur several times. They can be

* This work is partially supported by the National Science Foundation under Grant No. CCF-1553168 and No. CCF-1715387.

seen as "sets with counting". Although multisets are interesting mathematical objects that can be widely used in verification, there was no efficient reasoner for multisets and sets with cardinality constraints until recently [6]. Moreover, for a long time it was not known if the logic of multisets with cardinality constraints is even decidable [7]. Nevertheless, researchers have recognized the importance of this logic and they have been studied multisets in combination with other theories.

Zarba [13] investigated decision procedures for quantifier-free multisets but without the cardinality operator. He showed how to reduce a multiset formula to a quantifier-free defining each multiset operation pointwise on the elements of the set. Adding the cardinality operator makes such a reduction impossible.

Lugiez studied multiset constraints in the context of a more general result on multitree automata [7] and proved the decidability of multiset formulas with a weaker form of cardinality operator that counts only distinct elements in a multiset.

1.1 Multisets with Cardinality Constraints

In this paper we revive the first decision procedure for multisets with cardinality constraints [9, 10]. We represent multisets (*bags*) with their characteristic functions. A multiset m is a function $\mathbb{E} \to \mathbb{N}$, where $\mathbb{E}$ is the universe used for populating multisets and $\mathbb{N}$ is the set of non-negative integers. The value $m(e)$ is the multiplicity (the number of occurrences) of an element e in a multiset m. We assume that the domain $\mathbb{E}$ is fixed and finite but of an unknown size. We consider the logic of multisets constraints with the cardinality operator (MAPA), given in Fig. 1. An atom in MAPA is either a multiset comparison, or it is a standard quantifier-free linear integer arithmetic atom, or it is a quantified formula $(\forall e.\mathsf{F}^{\mathsf{in}})$, or it is a collecting sum formula. We allow only universal quantification over all elements of $\mathbb{E}$. This way we can express, for example, that for a multiset k it holds $\forall e.k(e) = 0 \vee k(e) = 1$ – in other words, k is a set. A collecting sum atom is used to group several formulas involving sums into a single atom. This is needed for the next step of the decision procedure. The sums are used in the definition of the cardinality operator:

$$|m| = \sum_{e \in \mathbb{E}} m(e)$$

Piskac and Kuncak [9] showed that every MAPA formula can be translated to an equisatisfiable LIA* formula. The translation is linear and described in [9]. This way reasoning about MAPA formulas reduces to reasoning about LIA* formulas.

1.2 Reasoning about LIA* Formulas

The LIA* logic [10] is a standard linear integer arithmetic extended with a new operator: the star operator, which is defined over a set of integer vectors as

top-level formulas:
$F ::= A \mid F \wedge F \mid \neg F$
$A ::= M{=}M \mid M \subseteq M \mid \forall e.\mathsf{F}^{\mathsf{in}} \mid \mathsf{A}^{\mathsf{out}}$
outer linear arithmetic formulas:
$\mathsf{F}^{\mathsf{out}} ::= \mathsf{A}^{\mathsf{out}} \mid \mathsf{F}^{\mathsf{out}} \wedge \mathsf{F}^{\mathsf{out}} \mid \neg \mathsf{F}^{\mathsf{out}}$
$\mathsf{A}^{\mathsf{out}} ::= \mathsf{t}^{\mathsf{out}} \leq \mathsf{t}^{\mathsf{out}} \mid \mathsf{t}^{\mathsf{out}}{=}\mathsf{t}^{\mathsf{out}} \mid (\mathsf{t}^{\mathsf{out}}, \ldots, \mathsf{t}^{\mathsf{out}}){=}\sum_{\mathsf{F}^{\mathsf{in}}}(\mathsf{t}^{\mathsf{in}}, \ldots, \mathsf{t}^{\mathsf{in}})$
$\mathsf{t}^{\mathsf{out}} ::= k \mid \ \mid M \mid \ \mid C \mid \mathsf{t}^{\mathsf{out}} + \mathsf{t}^{\mathsf{out}} \mid C \cdot \mathsf{t}^{\mathsf{out}} \mid \mathsf{ite}(\mathsf{F}^{\mathsf{out}}, \mathsf{t}^{\mathsf{out}}, \mathsf{t}^{\mathsf{out}})$
inner linear arithmetic formulas:
$\mathsf{F}^{\mathsf{in}} ::= \mathsf{A}^{\mathsf{in}} \mid \mathsf{F}^{\mathsf{in}} \wedge \mathsf{F}^{\mathsf{in}} \mid \neg \mathsf{F}^{\mathsf{in}}$
$\mathsf{A}^{\mathsf{in}} ::= \mathsf{t}^{\mathsf{in}} \leq \mathsf{t}^{\mathsf{in}} \mid \mathsf{t}^{\mathsf{in}}{=}\mathsf{t}^{\mathsf{in}}$
$\mathsf{t}^{\mathsf{in}} ::= m(e) \mid P \mid \mathsf{t}^{\mathsf{in}} + \mathsf{t}^{\mathsf{in}} \mid P \cdot \mathsf{t}^{\mathsf{in}} \mid \mathsf{ite}(\mathsf{F}^{\mathsf{in}}, \mathsf{t}^{\mathsf{in}}, \mathsf{t}^{\mathsf{in}})$
multiset expressions:
$M ::= m \mid \emptyset \mid M \cap M \mid M \cup M \mid M \uplus M \mid M \setminus M \mid M \setminus\!\setminus M \mid \mathsf{set}(M)$
terminals:
m - multiset variables; e - index variable (fixed)
k - integer variable; C - integer constant; P - non-negative integer constant

Fig. 1: The logic of multiset constraints with Presburger Arithmetic (MAPA)

follows:

$$S^\star \quad \triangleq \quad \left\{ \sum_{i=1}^{n} \boldsymbol{s}_i \quad \mid \ \forall i. 1 \leq i \leq n.\ \boldsymbol{s}_i \in S \right\} \tag{1}$$

The result of the star operator applied to set S is a set if all linear additive combinations of vectors from S. Its syntax is given in Fig. 2.

LIA* formulas: $\varphi ::= F_1 \wedge \boldsymbol{x_1} \in \{\boldsymbol{x_2} \mid F_2\}^\star$
such that $dim(\boldsymbol{x_1}) = dim(\boldsymbol{x_2})$ and $free\text{-}vars(F_2) \subseteq \boldsymbol{x_2}$
LIA formulas:
$F ::= A \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \neg F_1 \mid \exists x.\ F \mid \forall x.\ F$
$A ::= T_1 \leq T_2 \mid T_1 = T_2$
$T ::= x \mid C \mid T_1 + T_2 \mid C \cdot T_1 \mid \mathsf{ite}(F, T_1, T_2)$
terminals: x - integer variable; C - integer constant

Fig. 2: Linear integer arithmetic (LIA) and an extension with the Star Operator.

To check a satisfiability of a LIA* formula, we use the semilinear set characterization of solutions of integer linear arithmetic formulas.

Definition 1 (Semilinear sets). *A linear set $LS(\boldsymbol{a}, B)$ is defined by an integer vector $\boldsymbol{a}$ and a finite set of integer vectors $B = \{\boldsymbol{b_1}, \ldots, \boldsymbol{b_n}\}$, all of the same*

dimension, as follows:

$$LS(\boldsymbol{a}, B) \quad \triangleq \quad \left\{ \boldsymbol{a} + \sum_{i=1}^{n} \lambda_i \boldsymbol{b_i} \quad | \quad \bigwedge_{i=1}^{n} \lambda_i \geq 0 \right\} \tag{2}$$

Sometimes, as a shorthand, we use $\boldsymbol{\lambda} B = \sum_{i=1}^{n} \lambda_i \boldsymbol{b_i}$.

A semilinear set $SLS(ls_1, \ldots, ls_n)$ *is a finite union of linear sets* $ls_1, \ldots, ls_n$, *i.e.,* $SLS(ls_1, \ldots, ls_n) = \bigcup_{i=1}^{n} ls_i$.

Ginsburg and Spanier showed (Theorem 1.3 in [3]) that a solution set for every Presburger arithmetic formula is a semilinear set, and we use that result to eliminate the star operator.

Theorem 1 (Lemmas 2 and 3 in [10]). *Given a* LIA* *atom* $\boldsymbol{x_1} \in \{\boldsymbol{x_2} \mid F_2\}^\star$, *let* $SLS(LS(\boldsymbol{a_1}, B_1), \ldots, LS(\boldsymbol{a_k}, B_k))$ *be a semilinear set describing the set of the solutions of formula* F_2. *The atom* $\boldsymbol{x_1} \in \{\boldsymbol{x_2} \mid F_2\}^\star$ *is equisatisfiable to the following* LIA *formula:*

$$\exists \mu_1 \geq 0, \ldots, \mu_k \geq 0, \boldsymbol{\lambda}_1 \geq 0, \ldots \boldsymbol{\lambda}_k \geq 0 \ . \\ \boldsymbol{x_1} = \sum_{i=1}^{k} \mu_i \boldsymbol{a}_i + \boldsymbol{\lambda}_i B_i \wedge \bigwedge_{i=1}^{k} (\mu_i = 0 \rightarrow \boldsymbol{\lambda}_i = 0)$$

By applying Theorem 1, checking satisfiability of a LIA* formula reduces to reasoning about linear integer arithmetic. Note, however, that this approach results in automatically constructing a formula might be really large, depending on the size of a semilinear set. In addition, this approach relies on computing semilinear sets explicitly, both of which make it scale poorly even on simple benchmarks.

2 Illustrating Example

We illustrate now how is a decision procedure for MAPA working on the following simple multiset formula: for two multisets X and Y, the size of their disjoint union is the sum of their respective sizes. In other words, we need to prove the validity of the following formula

$$|X \uplus Y| = |X| + |Y|$$

As usual, we prove the unsatisfiability of the formula $|X \uplus Y| \neq |X| + |Y|$. The first step is to reduce this formula into an equisatisfiable LIA* formula. To do that, we perform a sequence of steps that resemble the purification step in the Nelson-Oppen combination procedure [8]. In a nutshell, we introduce a new variable for every non-terminal expression.

We first introduce a multiset variable M defining multiset expression $X \uplus Y$ and then we introduce integer variables k_1, k_2, k_3 for each of the cardinality expressions. This way the formula becomes:

$$k_1 \neq k_2 + k_3 \wedge k_1 = |M| \wedge k_2 = |X| \wedge k_3 = |Y| \wedge M = X \uplus Y$$

We next apply the point-wise definitions of the cardinality and $\uplus$ operators and we obtain the following formula:

$$k_1 \neq k_2 + k_3 \wedge k_1 = \sum_{e \in \mathbb{E}} M(e) \wedge k_2 = \sum_{e \in \mathbb{E}} X(e) \wedge k_3 = \sum_{e \in \mathbb{E}} Y(e) \\ \wedge \forall e. M(e) = X(e) + Y(e)$$

Grouping all the sum expressions together results in the formula:

$$k_1 \neq k_2 + k_3 \wedge (k_1, k_2, k_3) = \sum_{e \in \mathbb{E}} (M(e), X(e), Y(e)) \wedge \forall e. M(e) = X(e) + Y(e)$$

Piskac and Kuncak have shown in [9] that every multiset formula can be reduced to this form. They call it the *sum normal form*. It consists of three conjuncts. One is a pure LIA formula, the other is the summation and the third part is a universally quantified formula. By applying Theorem 2 from [9], the above MAPA formula is translated into an equisatisfiable LIA* formula, where m, x and y are non-negative integer variables:

$$k_1 \neq k_2 + k_3 \wedge (k_1, k_2, k_3) \in \{(m, x, y) | m = x + y\}^\star$$

To check the satisfiability of this formula, we first need to eliminate the star operator, which is done by computing a semilinear set describing the set of solutions of $m = x + y$. In this particular case, the semilinear set is actually a linear set, consisting of the zero vector and two vectors defining linear combinations:

$$\{(m, x, y) | m = x + y\} = LS((0, 0, 0), \{(1, 1, 0), (1, 0, 1)\})$$

Having the semilinear set representation, we can apply Theorem 1. In particular, only one linear set and the zero vector can significantly simplify the corresponding equisatisfiable formula. As the result of applying Theorem 1, we obtain that formula $(k_1, k_2, k_3) \in \{(m, x, y) | m = x + y\}^\star$ is equisatisfiable to the formula $(k_1, k_2, k_3) = \boldsymbol{\lambda}\{(1, 1, 0), (1, 0, 1)\} \Leftrightarrow (k_1, k_2, k_3) = \lambda_1(1, 1, 0) + \lambda_2(1, 0, 1)$.

This way we have eliminated the star operator from the given LIA* formula. It is now reduced to an equisatisfiable linear integer arithmetic formula:

$$k_1 \neq k_2 + k_3 \wedge k_1 = \lambda_1 + \lambda_2 \wedge k_2 = \lambda_1 \wedge k_3 = \lambda_2$$

The resulting LIA formula is unsatisfiable.

3 Efficient Reasoning about LIA* formulas

The described decision procedure is sound and complete. However, its crucial component is a computation of semilinear sets. While it is possible to compute Hilbert basis using the z3 [2] SMT solver, to the best of our knowledge there are no efficient tools for computing semilinear sets. Moreover, Pottier [12] showed that a semilinear set might contain an exponential number of vectors.

To overcome the explicit computation of semilinear sets, Piskac and Kuncak [10] developed a new decision procedure for LIA* which eliminates the star operator from the atom $\boldsymbol{x_1} \in \{\boldsymbol{x_2} \mid F\}^\star$ by showing that $\boldsymbol{x_1}$ is a linear combination of $\mathcal{O}(n^2 \log n)$ solution vectors of F, where n is the size of the input formula. Although this new decision procedure avoids computing semilinear sets, it instantly produces a very large formula that could not be solved in practice by existing tools, not even for the most simple benchmarks.

Levatich et al. [6] used those insights to develop a new efficient and scalable approach for solving LIA* formulas. The approach is based on the use of under- and over-approximations of LIA* formulas. This way one avoids the space overhead and explicitly computing semilinear sets.

The key insight of their approach is to construct a solution or a proof of unsatisfiability "on demand". Given a LIA* formula $F_1(\boldsymbol{x_1}) \wedge \boldsymbol{x_1} \in \{\boldsymbol{x_2} \mid F_2(\boldsymbol{x_2})\}^\star$, we first find any solution vector for formula F_2, let us name it $\boldsymbol{u_1}$. We next check if formula $F_1(\boldsymbol{x_1}) \wedge \boldsymbol{x_1} = \lambda_1 * \boldsymbol{u_1}$ is satisfiable. If this is the case, the given LIA* formula is satisfiable as well. However, if this is not the case, we cannot conclude anything about the satisifiability of the given LIA* formula, so we find a new different solution of formula F_2, denoted by $\boldsymbol{u_2}$: $F_2(\boldsymbol{u_2}) \wedge \boldsymbol{u_1} \neq \boldsymbol{u_2}$. Next, we check if the vector $\boldsymbol{x_1}$ is a linear combination of those two solution vectors: $F_1(\boldsymbol{x_1}) \wedge \boldsymbol{x_1} = \lambda_1 * \boldsymbol{u_1} + \lambda_2 * \boldsymbol{u_2}$. If this newly constructed formula is satisfiable, so is the original LIA* formula, otherwise we repeat the process. This way, by finding and checking solution vectors of F_2, we construct underapproximations of the set $\{\boldsymbol{x_2} \mid F_2(\boldsymbol{x_2})\}^\star$. Moreover, we know that this process will terminate once we check sufficiently many solution vectors, as shown in [10].

However, if the given LIA* formula is unsatisfiable, this approach will result in an equally large formula as in [10], and again it does not scale. Therefore, in parallel to finding an under-approximation of the set $\{\boldsymbol{x_2} \mid F_2(\boldsymbol{x_2})\}^\star$, we are also constructing a sequence of its over-approximation. The properties, that such an overapproximation should have, are encoded as a set of Constraint Horn Clauses and we use existing solvers to compute them. Such an overapproximation, if exists, is an interpolant that separates two conjuncts in the given LIA* formula, proving this way that the formula is unsatisfiable.

Finally, we have implemented the presented decision procedure and the tool is publicly available at `https://github.com/mlevatich/sls-reachability`. Because there were no MAPA benchmarks available, we had to create our own benchmarks. In addition, we also treated 240 BAPA benchmarks about sets, available in [1], as MAPA benchmarks. While the full report on the empirical results is available in [6], our general assessment is that the presented algorithm is effective on both SAT and UNSAT benchmarks. Our tool solved 83% of benchmarks in less than 50 seconds, and over 75% of those in under 3 seconds. We believe that this tool is the first efficient reasoner for multisets and sets with cardinality constraints.

4 Conclusions

The presented work describes a sequence of decision procedures that has lead towards an efficient reasoner for multisets and sets with cardinality constraints. We noticed that some constraints arising in formal verification of protocols and data structures could have been expressed more succinctly and naturally, were they using multisets as the underlying abstract datatype in the specification. Nevertheless, due to the lack of tool support they use sets, resulting in more complex constraints. While there was an older tool for reasoning about multisets with cardinality constraints [11], that tool served mainly as a proof of concept and was evaluated only on a handful of manually written formulas. We here presented a recent tool for reasoning about sets and multisets and we showed empirically that this tool scales well and can easily reason about complex multiset formulas. We hope that this work will lead to a renewed research interest in multisets and encourage their use in software analysis and verification.

Acknowledgments

This presentation is based on the previously published results on reasoning about multisets with cardinality constraints [6, 9–11]. We sincerely thank the collaborators on these projects: Nikolaj Bjørner, Maxwell Levatich, Viktor Kunčak and Sharon Shoham, without whom this work would not be possible.

References

1. Idan Berkovits, Marijana Lazic, Giuliano Losa, Oded Padon, and Sharon Shoham. Verification of threshold-based distributed algorithms by decomposition to decidable logics. In *CAV (2)*, volume 11562 of *Lecture Notes in Computer Science*, pages 245–266. Springer, 2019.
2. Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *TACAS 2008*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.
3. Seymour Ginsburg and Edwin H. Spanier. Semigroups, presburger formulas, and languages. *Pacific J. Math.*, 16(2):285–296, 1966.
4. Viktor Kuncak, Huu Hai Nguyen, and Martin C. Rinard. An algorithm for deciding BAPA: boolean algebra with presburger arithmetic. In *CADE*, volume 3632 of *Lecture Notes in Computer Science*, pages 260–277. Springer, 2005.
5. Viktor Kuncak, Huu Hai Nguyen, and Martin C. Rinard. Deciding boolean algebra with presburger arithmetic. *J. Autom. Reasoning*, 36(3):213–239, 2006.
6. Maxwell Levatich, Nikolaj Bjørner, Ruzica Piskac, and Sharon Shoham. Solving $LIA^{\star}$ using approximations. In *VMCAI*, volume 11990 of *Lecture Notes in Computer Science*, pages 360–378. Springer, 2020.
7. D. Lugiez. Multitree automata that count. *Theor. Comput. Sci.*, 333(1-2):225–263, 2005.
8. Greg Nelson and Derek C. Oppen. Fast decision procedures based on congruence closure. *J. ACM*, 27(2):356–364, 1980.

9. Ruzica Piskac and Viktor Kuncak. Decision procedures for multisets with cardinality constraints. In *VMCAI*, volume 4905 of *Lecture Notes in Computer Science*, pages 218–232. Springer, 2008.
10. Ruzica Piskac and Viktor Kuncak. Linear arithmetic with stars. In *CAV*, volume 5123 of *Lecture Notes in Computer Science*, pages 268–280. Springer, 2008.
11. Ruzica Piskac and Viktor Kuncak. MUNCH - automated reasoner for sets and multisets. In *IJCAR*, volume 6173 of *Lecture Notes in Computer Science*, pages 149–155. Springer, 2010.
12. Loic Pottier. Minimal solutions of linear diophantine systems: Bounds and algorithms. In *RTA*, volume 488 of *Lecture Notes in Computer Science*, pages 162–173. Springer, 1991.
13. Calogero G. Zarba. Combining multisets with integers. In *CADE-18*, 2002.