A Quantifier Elimination Algorithm for a Fragment of Set Theory Involving the Cardinality Operator

Calogero G. Zarba

LORIA and INRIA-Lorraine
615, rue du Jardin Botanique, BP 101
54602 Villers les Nancy Cedex, France
E-mail: calogero.zarba@loria.fr

Abstract. We present a decision procedure based on quantifier elimination for a fragment of set theory involving elements, integers, and finite sets of elements in the presence of the cardinality operator. The language allows quantification on element variables and integer variables, but not on set variables.
We also show that if we identify the sort of elements with the sort of integers, thus considering the case of finite sets of integers, then the resulting language becomes undecidable.

1 Introduction

The cardinality of a set is the number of its elements. Thus, in order to reason about set-theoretic formulae involving the cardinality operator, a set-theoretic theorem prover needs the ability to reason about numbers. Such ability can be basically provided in two ways: either by representing numbers directly as sets by means of the set-theoretic identities $0 = \emptyset$ and $n+1 = n \cup \{n\}$, or by employing specialized decision procedures for arithmetic. We argue that, although the former solution is very elegant, the latter is more practical and more amenable to modularization.

In this paper we introduce the set-theoretic language **Q2LSC** (*Quantified Two-Level Syllogistic with Cardinality*) for expressing constraints over elements, integers, and finite sets of elements in the presence of the cardinality operator. **Q2LSC** is a many-sorted language with sorts elem, int, and set, in which quantification is allowed only on elem-variables and int-variables, and it is forbidden on set-variables.

We prove that the language **Q2LSC** is decidable by providing:

1. a quantifier-elimination algorithm that converts any **Q2LSC**-formula φ into an equivalent quantifier-free **Q2LSC**-formula ψ;
2. a decision procedure for quantifier-free **Q2LSC**-formulae.

Furthermore, we show that if we identify the sort elem with the sort int, thus considering the case of finite sets of integers, then the resulting language becomes undecidable.

Organization of the paper. In Section 2 we define the syntax and semantics of the language **Q2LSC**. In Section 3 we describe and prove correct our decision procedure for **Q2LSC**. In Section 4 we provide one example of our decision procedure in action. In Section 5 we show that if we identify the sorts elem and int then we obtain an undecidable language. In Section 6 we draw final conclusions.

2 The language Q2LSC

2.1 Syntax

The language **Q2LSC** (*Quantified Two-Level Syllogistic with Cardinality*) is a many-sorted language with three sorts elem, int, and set, plus the following symbols:

- arbitrarily many variables of sort τ, for each $\tau \in \{\mathsf{elem}, \mathsf{int}, \mathsf{set}\}$.
- the constants:
  - 0 and 1, both of sort int (*zero* and *one*);
  - $\emptyset$, of sort set (*empty set*);
- the function symbols:
  - $+$ and $-$, both of sort $\mathsf{int} \times \mathsf{int} \to \mathsf{int}$ (*addition* and *subtraction*);
  - $\{\cdot\}$, of sort $\mathsf{elem} \to \mathsf{set}$ (*singleton set*);
  - $\cup, \cap, \setminus$, of sort $\mathsf{set} \times \mathsf{set} \to \mathsf{set}$ (*union*, *intersection*, and *set difference*);
  - $|\cdot|$, of sort $\mathsf{set} \to \mathsf{int}$ (*cardinality*);
- the predicate symbols:
  - $<$, of sort $\mathsf{int} \times \mathsf{int}$ (*less-than*);
  - $\in$, of sort $\mathsf{elem} \times \mathsf{set}$ (*membership*);
  - $\subseteq$, of sort $\mathsf{elem} \times \mathsf{elem}$ (*set inclusion*);
- an equality symbol $=_\tau$, for each $\tau \in \{\mathsf{elem}, \mathsf{int}, \mathsf{set}\}$;
- the propositional connectives $\neg$, $\wedge$, $\vee$, and $\to$;
- the quantifiers $\forall_\tau$ and $\exists_\tau$, for each $\tau \in \{\mathsf{elem}, \mathsf{int}\}$.

Note that the language **Q2LSC** allows quantification over elem-variables and int-variables, but it forbids it on set-variables.

Q2LSC-*terms* (respectively, **Q2LSC**-*formulae*) are well-sorted terms (respectively, well-sorted formulae) constructed using the symbols in the language **Q2LSC**.

To simplify the notation, in the rest of the paper we will often write $=$, $\forall$, and $\exists$ in place of $=_\tau$, $\forall_\tau$, and $\exists_\tau$ when τ is clear from the context.

2.2 Semantics

Definition 1. An INTERPRETATION $\mathcal{A}$ is a mapping of the sorts, variables and symbols in the language **Q2LSC** satisfying the following conditions:

- each sort $\tau \in \{\mathsf{elem}, \mathsf{int}, \mathsf{set}\}$ is mapped to a set A_τ such that:
  - A_{elem} is infinite;

  - $A_{\mathsf{int}} = \mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$;
  - $A_{\mathsf{set}} = 2^{A_{\mathsf{elem}}}$;
- for each sort τ, each variable x of sort τ is mapped to an element $x^{\mathcal{A}}$ in A_τ;
- $x^{\mathcal{A}}$ is a finite set, for each variable x of sort set;
- the symbols 0, 1, $\emptyset$, $+$, $-$, $\{\cdot\}$, $\cap$, $\cup$, $\setminus$, $|\cdot|$, $<$, $\in$, $\subseteq$ are interpreted according to their standard interpretation;
- $=_\tau$ is interpreted as the identity in A_τ, for $\tau \in \{\mathsf{elem}, \mathsf{int}, \mathsf{set}\}$. $\square$

For each interpretation $\mathcal{A}$, we denote with $t^{\mathcal{A}}$ the evaluation of the term t under $\mathcal{A}$. Similarly, we denote with $\varphi^{\mathcal{A}}$ the truth value of φ under the interpretation $\mathcal{A}$. Note that for each set-term t, the expression $[|t|]^{\mathcal{A}}$ is a well-defined integer. In fact, since $\mathcal{A}$ maps set-variables into finite sets, it follows that $t^{\mathcal{A}}$ is finite.

A **Q2LSC**-formula φ is:

- *valid*, if $\varphi^{\mathcal{A}} = \mathtt{true}$, for each interpretation $\mathcal{A}$;
- *satisfiable*, if $\varphi^{\mathcal{A}} = \mathtt{true}$, for some interpretation $\mathcal{A}$;
- *unsatisfiable*, if it is not satisfiable.

3 A decision procedure for Q2LSC

In this section we describe a decision procedure for checking the satisfiability of **Q2LSC**-formulae.

Thus, let φ be a **Q2LSC**-formula whose satisfiability we want to check. **The first step** of our decision procedure consists in running a quantifier elimination algorithm over φ. The output of our quantifier elimination algorithm will be a formula ψ satisfying the following properties:

(i) ψ is equivalent to φ;
(ii) ψ is quantifier-free;
(iii) ψ may contain occurrences of the symbol $\mathbb{1}$ (*universal set*).

The symbol $\mathbb{1}$ in property (iii) is used by the quantifier elimination algorithm in order to denote the universal set containing all elements. Thus, for any interpretation $\mathcal{A}$, we require that $\mathbb{1}^{\mathcal{A}} = A_{\mathsf{elem}}$. Note that since A_{elem} is required to be infinite, the term $|\mathbb{1}|$ has no well-defined semantics. This is not a problem since our quantifier elimination algorithm manipulates formulae in which the symbol $\mathbb{1}$ never occurs inside terms of the form $|t|$.

The second step of our decision procedure consists in eliminating all occurrences of the symbol $\mathbb{1}$ from ψ, obtaining a quantifier-free **Q2LSC**-formula χ that is equivalent to ψ, and in which the symbol $\mathbb{1}$ does not occur.

Finally, **the third step** of our decision procedure consists in checking the satisfiability of χ.

We describe our quantifier elimination algorithm in Subsections 3.1–3.3. We describe how to remove the occurrences of the symbol $\mathbb{1}$ in Subsection 3.4. Finally, we describe how to check the satisfiability of quantifier-free **Q2LSC**-formulae in Subsection 3.5.

3.1 The quantifier elimination algorithm

Without loss of generality, we restrict ourselves to the quantifier elimination of formulae of the form

$$(\exists u)\Gamma$$

such that:

(i) u is either an elem-variable or an int-variable;
(ii) Γ is a conjunction of literals;
(iii) u occurs in every literal of Γ.

We describe how to eliminate int-variables in Section 3.2, and how to eliminate elem-variables in Section 3.3.

3.2 How to eliminate integer variables

Suppose that the variable u in $(\exists u)\Gamma$ is of sort int. Since u occurs in every literal of Γ, and since elem-literals and set-literals do not contain int-variables, it follows that Γ contains only int-literals.

Let $|t_1|, \ldots, |t_n|$ be the set of all terms of the form $|t|$ occurring in Γ, and let $v_1, \ldots, v_n$ be fresh int-variables not occurring in Γ.

We can eliminate the quantified int-variable u from

$$(\exists u)\Gamma(|t_1|, \ldots, |t_n|)$$

by using the following process:

Step 1: Abstract the terms $|t_i|$ with the variables v_i, obtaining the formula

$$(\exists u)\Gamma(v_1, \ldots, v_n)\,.$$

Step 2: Apply a quantifier eliminating algorithm for the integers, obtaining an equivalent quantifier-free formula

$$\Delta(v_1, \ldots, v_n)\,.$$

Step 3: Undo the abstraction, returning the quantifier-free formula

$$\Delta(|t_1|, \ldots, |t_n|)\,.$$

The following proposition shows that our method for elimination quantified int-variables is correct.

Proposition 2. *$(\exists u)\Gamma(|t_1|, \ldots, |t_n|)$ and $\Delta(|t_1|, \ldots, |t_n|)$ are equivalent.* □

$s =_{\mathsf{elem}} t$	$\longrightarrow$	$\{s\} = \{t\}$
$s \in t$	$\longrightarrow$	$\{s\} \subseteq t$
$s \subseteq t$	$\longrightarrow$	$s \setminus t = \emptyset$
$s =_{\mathsf{set}} t$	$\longrightarrow$	$(s \setminus t) \cup (t \setminus s) = \emptyset$

Note: In the last rule, t is not $\emptyset$.

Figure 1: Rewrite rules for normalizing the input.

PROOF. Let $\mathcal{A}$ be an interpretation. Define a new interpretation $\mathcal{B}$ which is the same as $\mathcal{A}$, except (maybe) on the variables v_i, where we force $v_i^{\mathcal{B}} = [|t_i|]^{\mathcal{A}}$. Then clearly

$$[(\exists u)\Gamma(|t_1|, \ldots, |t_n|)]^{\mathcal{A}} = [(\exists u)\Gamma(v_1, \ldots, v_n)]^{\mathcal{B}} ,$$

and

$$[\Delta(|t_1|, \ldots, |t_n|)]^{\mathcal{A}} = [\Delta(v_1, \ldots, v_n)]^{\mathcal{B}} .$$

Because Δ is the result of a quantifier elimination algorithm over the integers, we further have

$$[(\exists u)\Gamma(v_1, \ldots, v_n)]^{\mathcal{B}} = [\Delta(v_1, \ldots, v_n)]^{\mathcal{B}} .$$

Thus,

$$[(\exists u)\Gamma(|t_1|, \ldots, |t_n|)]^{\mathcal{A}} = [\Delta(|t_1|, \ldots, |t_n|)]^{\mathcal{A}} . \qquad \blacksquare$$

3.3 How to eliminate element variables

Suppose that the variable u in $(\exists u)\Gamma$ is of sort elem. By applying the rewrite rules in Figure 1, we can assume without loss of generality that all literals in Γ are of the form

$$t = \emptyset , \qquad t \neq \emptyset , \qquad \ell ,$$

where t is a set-term, and ℓ is an int-literal. Note that u can occur in Γ only as the argument of occurrences of $\{u\}$.

Let $t^{(1)}, \ldots, t^{(n)}$ be all topmost set-terms in Γ containing an occurrence of $\{u\}$. Moreover, let $t_0^{(i)}$ and $t_1^{(i)}$, for $i = 1, \ldots, n$, be the set-terms defined as follows:

- $t_0^{(i)}$ is obtained from $t^{(i)}$ by replacing all occurrences of $\{u\}$ with $\emptyset$;
- $t_1^{(i)}$ is obtained from $t^{(i)}$ by replacing all occurrences of $\{u\}$ with $\mathbb{1}$.

$$
\begin{array}{lcl}
u \in t_0 \cap t_1 \ \wedge \ \Gamma(t) & \longrightarrow & u \in t_0 \cap t_1 \ \wedge \ \Gamma(t_0) \\
u \in t_0 \setminus t_1 \ \wedge \ t = \emptyset \ \wedge \ \Delta & \longrightarrow & u \in t_0 \setminus t_1 \ \wedge \ |t_0| = 1 \ \wedge \ \Delta \\
u \in t_0 \setminus t_1 \ \wedge \ t \neq \emptyset \ \wedge \ \Delta & \longrightarrow & u \in t_0 \setminus t_1 \ \wedge \ |t_0| > 1 \ \wedge \ \Delta \\
u \in t_0 \setminus t_1 \ \wedge \ \Gamma(|t|) & \longrightarrow & u \in t_0 \setminus t_1 \ \wedge \ \Gamma(|t_0| - 1) \\
u \in t_1 \setminus t_0 \ \wedge \ t = \emptyset \ \wedge \ \Delta & \longrightarrow & \mathit{false} \\
u \in t_1 \setminus t_0 \ \wedge \ t \neq \emptyset \ \wedge \ \Delta & \longrightarrow & u \in t_1 \setminus t_0 \ \wedge \ \Delta \\
u \in t_1 \setminus t_0 \ \wedge \ \Gamma(|t|) & \longrightarrow & u \in t_1 \setminus t_0 \ \wedge \ \Gamma(|t_0| + 1) \\
u \in \mathbb{1} \setminus (t_0 \cup t_1) \ \wedge \ \Gamma(t) & \longrightarrow & u \in \mathbb{1} \setminus (t_0 \cup t_1) \ \wedge \ \Gamma(t_0)
\end{array}
$$

Note: In the above rules, t is any one of the terms $t^{(i)}$. Moreover, t_0 is the term obtained from t by replacing all occurrences of $\{u\}$ with $\emptyset$, whereas t_1 is the term obtained from t by replacing all occurrences of $\{u\}$ with $\mathbb{1}$.

Figure 2: Rewrite rules for eliminating all occurrences of $\{u\}$.

Finally, let S_i be the following set of set-terms, for $i = 1, \ldots, n$:

$$
S_i = \left\{ \begin{array}{l} t_0^{(i)} \cap t_1^{(i)} \,, \\ t_0^{(i)} \setminus t_1^{(i)} \,, \\ t_1^{(i)} \setminus t_0^{(i)} \,, \\ \mathbb{1} \setminus \left(t_0^{(i)} \cup t_1^{(i)} \right) \end{array} \right\} .
$$

In order to eliminate the quantified variable u, we rewrite $(\exists u)\Gamma$ into the (very complex!) equivalent formula

$$
(\exists u) \left[\bigvee_{s_1 \in S_1} \cdots \bigvee_{s_n \in S_n} (u \in s_1 \ \wedge \ \cdots \ \wedge \ u \in s_n \ \wedge \ \Gamma) \right] .
$$

After distributing the existential quantifier, the above formula is equivalent to

$$
\bigvee_{s_1 \in S_1} \cdots \bigvee_{s_n \in S_n} (\exists u) \, (u \in s_1 \ \wedge \ \cdots \ \wedge \ u \in s_n \ \wedge \ \Gamma) \ .
$$

Next, we eliminate all occurrences of $\{u\}$ by applying the rewrite rules in Figure 2. After all possible rewrite rules have been applied, each disjunct of the above formula has been transformed into a formula of the form

$$
(\exists u) \, (u \in s_1 \ \wedge \ \cdots \ \wedge \ u \in s_n \ \wedge \ \Gamma') \ ,
$$

where Γ' does not contain any occurrence of u. Thus, the transformed disjunct is equivalent to the quantifier-free formula

$$\left(\bigcap_{i=1}^{n} s_i \neq \emptyset\right) \ \wedge \ \Gamma' \, .$$

We now prove that the process just described effectively eliminates the quantifier elem-variable u while preserving equivalence. To do this, denote with R the rewrite system in Figure 2. Then we can conclude that our quantifier elimination method is correct, provided that we are able to show that:

- R is terminating;
- R effectively removes all occurrences of $\{u\}$;
- each application of one of the rewrite rules of R preserves equivalence.

Clearly, each application of a rewrite rule of R removes at least one occurrence of $\{u\}$. Therefore, R is a terminating rewrite system that effectively removes all occurrences of $\{u\}$.

The proof that R preserves equivalence is more complex, and relies on the following proposition due to Boole [1].

Proposition 3. *Let u be an elem-variable, and let t be a set-term. Also, let t_0 be the term obtained from t by replacing all occurrences of $\{u\}$ with $\emptyset$, and let t_1 be the term obtained from t by replacing all occurrences of $\{u\}$ with $\mathbb{1}$.*

Then, for each interpretation $\mathcal{A}$:

$$t^{\mathcal{A}} = [(t_0 \setminus \{u\}) \cup (t_1 \cap \{u\})]^{\mathcal{A}} \, . \qquad \square$$

Proposition 3 is useful because it allows us to simplify the term t, provided that we know where u lies in the Venn diagram of the sets t_0 and t_1. More in detail, we have that the following formulae are valid:

$$u \in t_0 \cap t_1 \ \rightarrow \ t = t_0 \, , \tag{1}$$

$$u \in t_0 \setminus t_1 \ \rightarrow \ t = t_0 \setminus \{u\} \, , \tag{2}$$

$$u \in t_1 \setminus t_0 \ \rightarrow \ t = t_0 \cup \{u\} \, , \tag{3}$$

$$u \in \mathbb{1} \setminus (t_0 \cup t_1) \ \rightarrow \ t = t_0 \, . \tag{4}$$

Proposition 4. *Let φ be a formula, and let ψ be the result of applying to φ one of the rules in Figure 2. Then φ and ψ are equivalent.* $\square$

Proof. The first rule in Figure 2 preserves equivalence because of the validity of (1). The second, third, and fourth rule preserve equivalence because of the validity of (2). The fifth, sixth, and eighth rule preserve equivalence because of the validity of (3). Finally, the ninth rule preserves equivalence because of the validity of (4). ■

$$
\begin{array}{rclrcl}
t \cap \mathbb{1} & \longrightarrow & t & \mathbb{1} \cap t & \longrightarrow & t \\
t \cup \mathbb{1} & \longrightarrow & \mathbb{1} & \mathbb{1} \cup t & \longrightarrow & \mathbb{1} \\
t \setminus \mathbb{1} & \longrightarrow & \emptyset & & & \\
(\mathbb{1} \setminus s) \cup t & \longrightarrow & \mathbb{1} \setminus (s \setminus t) & t \cup (\mathbb{1} \setminus s) & \longrightarrow & \mathbb{1} \setminus (s \setminus t) \\
(\mathbb{1} \setminus s) \cap t & \longrightarrow & t \setminus s & t \cap (\mathbb{1} \setminus s) & \longrightarrow & t \setminus s \\
(\mathbb{1} \setminus s) \setminus t & \longrightarrow & \mathbb{1} \setminus (s \cup t) & & & \\
s \setminus (\mathbb{1} \setminus t) & \longrightarrow & s \cap t & & & \\
\mathbb{1} = \emptyset & \longrightarrow & \mathit{false} & \mathbb{1} = \mathbb{1} & \longrightarrow & \mathit{true} \\
\mathbb{1} \setminus t = \emptyset & \longrightarrow & t = \mathbb{1} & \mathbb{1} \setminus t = \mathbb{1} & \longrightarrow & t = \emptyset \\
 & & & t = \mathbb{1} & \longrightarrow & \mathit{false}
\end{array}
$$

Note: In the last rule, $\mathbb{1}$ does not occur in t.

Figure 3: Rewrite rules for eliminating all occurrences of $\mathbb{1}$.

3.4 How to eliminate the symbol $\mathbb{1}$

Let φ be a quantifier-free formula containing occurrences of the symbol $\mathbb{1}$. By applying the rewrite rules in Figure 1, we can assume without loss of generality that all literals in φ are of the form

$$t = \emptyset\,, \qquad\qquad t \neq \emptyset\,, \qquad\qquad \ell\,,$$

where t is a set-term, and ℓ is an int-literal. Then we can eliminate all occurrences of $\mathbb{1}$ from φ by applying the rewrite rules in Figure 3.

In order to show that the process just described effectively eliminates all occurrences of $\mathbb{1}$ while preserving equivalence, denote with R the rewriting system in Figure 3.

Clearly, R preserves equivalence. That is, if φ is a formula and ψ is the result of applying to φ one of the rewrite rules in R, then $\varphi^{\mathcal{A}} = \psi^{\mathcal{A}}$, for each interpretation $\mathcal{A}$.

In addition, if $\mathbb{1}$ occurs in φ, then at least one of the rewrite rules of R can be applied.

Finally, R is terminating. To show termination, it is enough to note that any application of a rewrite rule of R to a formula φ always decreases—according to the lexicographic order on pairs of natural numbers—the value of the pair

$$(f(\varphi), g(\varphi))\,,$$

where:

- $f(\varphi)$ is the number of occurrences of $\mathbb{1}$ in φ;
- $g(\varphi)$ is the sum of all the depths of the occurrences of $\mathbb{1}$ in φ.[1]

3.5 How to check the satisfiability of quantifier-free formulae

Let φ be a quantifier-free **Q2LSC**-formula. In order to check whether φ is satisfiable, we use a decision procedure for the quantifier-free language **F2LSC** studied in [4, 5].

The language **F2LSC** is both less general and more general than the language **Q2LSC**. Precisely, the differences between **Q2LSC** and **F2LSC** are the following:

- **F2LSC** is a quantifier-free language, whereas **Q2LSC** allows quantification over elem-variables and int-variables.
- **F2LSC** allows for an arbitrary background theory T of the elements over a given signature Σ_{elem} containing symbols over the sort elem. In contrast, **Q2LSC** can be thought as having a fixed background theory T_0 of the elements. This theory has an empty signature $\Sigma_{\mathsf{elem}} = \emptyset$, and has as axioms formulae that force the domain of the elements to be infinite. In other words, T_0 contains, for each integer $n > 0$, the axiom

$$(\forall_{\mathsf{elem}} x_1) \ldots (\forall_{\mathsf{elem}} x_n) \left[\left(\bigwedge_{1 \leq i < j \leq n} x_i \neq x_j \right) \to (\exists_{\mathsf{elem}} y) \left(\bigwedge_{i=1}^{n} y \neq x_i \right) \right] .$$

In [4, 5], we described how to obtain a decision procedure for **F2LSC** given a decision procedure P for the theory T of the elements. More precisely, it is required that P can decide the satisfiability in the theory T of quantifier-free elem-formulae over the signature Σ_{elem}.

Thus, in order to decide the satisfiability of quantifier-free **Q2LSC**-formulae, we only need to have a decision procedure for T_0. The following proposition shows that this decision procedure exists. In particular, any decision procedure for the theory of equality is also a decision procedure for T_0.

Proposition 5. *Let φ be a quantifier-free* elem*-formula. Then φ is satisfiable in T_0 if and only if φ is satisfiable in the theory of equality.* □

Proof. Clearly, if φ is satisfiable in T_0, then φ is satisfiable in the theory of equality.

Vice versa, assume that φ is satisfiable in the theory of equality. Since the theory of equality is stably infinite, there exists an interpretation $\mathcal{A}$ satisfying both the theory of equality and φ, and such that A_{elem} is infinite. But then, $\mathcal{A}$ also satisfies T_0, which implies that φ is satisfiable in T_0. ■

[1] The depth of an occurrence *occ* within a formula φ is the distance between the node representing *occ* and the root node in the tree representing φ.

4 An example

Consider the **Q2LSC**-formula

$$(\forall u)(u \in x)\,,$$

where u is an elem-variable and x is a set-variable. This formula is unsatisfiable because, for every interpretation $\mathcal{A}$, we have that $x^{\mathcal{A}}$ is finite and A_{elem} is infinite. Thus $x^{\mathcal{A}} \neq A_{\mathsf{elem}}$.

Let us prove the unsatisfiability using our quantifier elimination method. We first replace the universal quantifier with an existential quantifier, obtaining

$$\neg(\exists u)(u \notin x)\,.$$

After applying the rewrite rules in Figure 1, we obtain

$$\neg(\exists u)(\{u\} \setminus x \neq \emptyset)\,.$$

Next, we let t, t_0, and t_1 be the following terms:

$$\begin{aligned} t &\equiv \{u\} \setminus x\,, \\ t_0 &\equiv \emptyset \setminus x\,, \\ t_1 &\equiv \mathbb{1} \setminus x\,. \end{aligned}$$

According to our method, we obtain

$$\neg(\exists u)\begin{bmatrix} (u \in t_0 \cap t_1 \ \wedge\ t \neq \emptyset) \ \vee \\ (u \in t_0 \setminus t_1 \ \wedge\ t \neq \emptyset) \ \vee \\ (u \in t_1 \setminus t_0 \ \wedge\ t \neq \emptyset) \ \vee \\ (u \in \mathbb{1} \setminus (t_0 \cup t_1) \ \wedge\ t \neq \emptyset) \end{bmatrix}.$$

After distributing the existential quantifier and applying the rewrite rules in Figure 2, we obtain

$$\neg\begin{bmatrix} (\exists u)(u \in t_0 \cap t_1 \ \wedge\ t_0 \neq \emptyset) \ \vee \\ (\exists u)(u \in t_0 \setminus t_1 \ \wedge\ |t_0| > 1) \ \vee \\ (\exists u)(u \in t_1 \setminus t_0) \ \vee \\ (\exists u)(u \in \mathbb{1} \setminus (t_0 \cup t_1) \ \wedge\ t_0 \neq \emptyset) \end{bmatrix},$$

which is equivalent to

$$\neg\begin{bmatrix} (t_0 \cap t_1 \neq \emptyset \ \wedge\ t_0 \neq \emptyset) \ \vee \\ (t_0 \setminus t_1 \neq \emptyset \ \wedge\ |t_0| > 1) \ \vee \\ (t_1 \setminus t_0 \neq \emptyset) \ \vee \\ (\mathbb{1} \setminus (t_0 \cup t_1) \neq \emptyset \ \wedge\ t_0 \neq \emptyset) \end{bmatrix}.$$

We then eliminate the symbol $\mathbb{1}$ by applying the rewrite rules in Figure 3, obtaining

$$\neg \begin{bmatrix} (\mathit{false} \ \wedge \ t_0 \neq \emptyset) \ \vee \\ (\mathit{false} \ \wedge \ |t_0| > 1) \ \vee \\ \mathit{true} \ \vee \\ (x \neq \emptyset \ \wedge \ t_0 \neq \emptyset) \end{bmatrix},$$

which is equivalent to *false*. Since *false* is unsatisfiable, we conclude that also the original formula $(\forall u)(u \in x)$ is unsatisfiable.

5 Sets of integers

In this section we define and prove undecidable the language **SI** (*Sets and Integers*) that combines integers with finite sets of integers.

Formally, the language **SI** has two sorts, int for integers, and set for finite sets of integers. It also has all the function and predicate symbols of the language **Q2LSC**, but opportunely resorted and interpreted over the domain combining integers with finite sets of integers. To obtain undecidability, it is enough for **SI** to allow quantification over int-variables only.

The language **SI** is very similar to the language $\mathcal{PS}_\forall$ studied in [2], the only significant difference being that in $\mathcal{PS}_\forall$ sets are not required to be finite. In [2] it is proved that $\mathcal{PS}_\forall$ is undecidable. Unfortunately, that proof uses infinite sets, and therefore it cannot be directly used to show undecidability of **SI**.

Nevertheless, using a similar proof, we prove in this section that **SI** is undecidable. This will be done by showing that multiplication of positive integers is expressible in **SI** (cf. [3]).

Given two integers $x, y > 0$, we can mutually define in **SI** the sets A and B such that:

- A is the set of all multiples of x that are less than or equal to $\mathit{lcm}(x, y)$;[2]
- B is the set of all multiples of y that are less than or equal to $\mathit{lcm}(x, y)$.

The following formula defines A and B as desired.

$$\begin{aligned} & 0 \in A \ \wedge \ x \in A \ \wedge \ 0 \in B \ \wedge \ y \in B \ \wedge \\ & (\forall z)(0 < z < x \ \rightarrow \ z \notin A) \ \wedge \ (\forall z)(0 < z < y \ \rightarrow \ z \notin B) \ \wedge \\ & (\forall z)(z \notin A \ \rightarrow \ z + x \notin A) \ \wedge \ (\forall z)(z \notin B \ \rightarrow \ z + y \notin B) \ \wedge \\ & (\forall z)\left((z \in A \ \wedge \ (\exists u)(u \in B \ \wedge \ z < u)) \ \rightarrow \ z + x \in A\right) \ \wedge \\ & (\forall z)\left((z \in B \ \wedge \ (\exists u)(u \in A \ \wedge \ z < u)) \ \rightarrow \ z + y \in B\right) . \end{aligned}$$

[2] $\mathit{lcm}(x, y)$ denotes the least common multiple of x and y.

Note that $A \cap B = \{0, lcm(x, y)\}$. Therefore, for $x, y > 0$, the relationship $z = lcm(x, y)$ can be expressed in **SI** by the formula

$$z \neq 0 \ \wedge \ z \in A \cap B\,.$$

Finally, for $x, y > 0$, the relationship $z = xy$ can be expressed in **SI** by the formula

$$z + z = lcm(x + y, x + y - 1) - lcm(x, x - 1) - lcm(y, y - 1)\,.$$

In fact, for $u \geq 2$, we have that u and $u - 1$ are relative prime, and therefore $lcm(u, u - 1) = u(u - 1)$. Thus, $2z = (x + y)(x + y - 1) - x(x - 1) - y(y - 1)$, which implies $2z = 2xy$.

6 Conclusion

We introduced the set-theoretic language **Q2LSC**, in which we can express constraints over elements, integers, and finite sets of elements in the presence of the cardinality operator. The language **Q2LSC** allows quantification over elem-variables and int-variables, but not on set-variables.

We presented a decision procedure for **Q2LSC**, based on a quantifier elimination method, and a decision procedure for quantifier-free **Q2LSC**-formulae.

Note that quantifier elimination is possible mainly thanks to the fact that the sort elem is disjoint with the sort int. In fact, we showed that if ones identifies elements with integers, thus considering the case of finite sets of integers, then the resulting language is undecidable.

Another reason that makes quantifier elimination possible is the absence of uninterpreted function and predicate symbols over the sort elem. This implies that the only elem-atoms that can be written in **Q2LSC** are of the form $u = v$, where u, v are elem-variables. This syntactic restriction is essential in order to obtain a decidability result. In fact, if one allows uninterpreted function and predicate symbols over the sort elem, then the restriction of **Q2LSC** to the sort elem becomes a full-fledged first-order logic language, and therefore it is undecidable.

The language **Q2LSC** does not allow quantification on set-variables. To date, we do not know yet whether this restriction is necessary for having a quantifier elimination result. We conjecture that if quantification over set-variables is allowed then the resulting language is undecidable, but we have not been able to prove it so far.

Acknowledgements

This work was in part supported by the project GECCOO in the context of the French national research program *ACI Sécurité Informatique*.

References

1. George Boole. *An Investigation of the Laws of Thought*. Walton, London, 1847.
2. Domenico Cantone, Vincenzo Cutello, and Jacob T. Schwartz. Decision problems for Tarski and Presburger arithmetics extended with sets. In Egon Börger, Hans Kleine Büning, Michael M. Richter, and Wolfgang Schönfeld, editors, *Computer Science Logic*, volume 533 of *Lecture Notes in Computer Science*, pages 95–109. Springer, 1991.
3. Yuri V. Matiyasevich. Enumerable sets are diophantine. *Soviet Mathematics Doklady*, 11(2):354–357, 1970.
4. Calogero G. Zarba. Combining sets with integers. In Alessandro Armando, editor, *Frontiers of Combining Systems*, volume 2309 of *Lecture Notes in Computer Science*, pages 103–116. Springer, 2002.
5. Calogero G. Zarba. *The Combination Problem in Automated Reasoning*. PhD thesis, Stanford University, 2004.