# What's Decidable About Sequences?*

Carlo A. Furia

Chair of Software Engineering, ETH Zurich
caf@inf.ethz.ch http://se.inf.ethz.ch/people/furia/

**Abstract.** We present a first-order theory of (finite) sequences with integer elements, Presburger arithmetic, and regularity constraints, which can model significant properties of data structures such as lists and queues. We give a decision procedure for the quantifier-free fragment, based on an encoding into the first-order theory of concatenation; the procedure has PSPACE complexity. The quantifier-free fragment of the theory of sequences can express properties such as sortedness and injectivity, as well as Boolean combinations of periodic and arithmetic facts relating the elements of the sequence and their positions (e.g., "for all even $i$'s, the element at position $i$ has value $i + 3$ or $2i$"). The resulting expressive power is orthogonal to that of the most expressive decidable logics for arrays. Some examples demonstrate that the fragment is also suitable to reason about sequence-manipulating programs within the standard framework of axiomatic semantics.

## 1 Introduction

Verification is undecidable already for simple programs, but modern programming languages support a variety of sophisticated features that make it all the more complicated. These advanced constructs — such as arrays, pointers, dynamic allocation of resources, and object-oriented abstract data types — are needed because they raise the level of abstraction thus making programmers more productive and programs less defective. In an attempt to keep the pace with the development of programming languages, verification techniques have progressed rapidly over the years.

Further steady progress requires expressive program logics and powerful decision procedures. In response to the evolution of modern programming languages, new decidable program logic fragments and combination techniques for different fragments have mushroomed especially in recent years. Many of the most successful contributions have focused on verifying relatively restricted aspects of a program's behavior, for example by decoupling pointer structure and functional properties in the formal analysis of a dynamic data structure. This narrowing choice, partly deliberate and partly required by the formidable difficulty of the various problems, is effective because different aspects are often sufficiently decoupled that each of them can be analyzed in isolation with the most appropriate, specific technique.

This paper contributes to the growing repertory of special program logics by exploring the decidability of properties about *sequences*. Sequences of elements of homogeneous type can abstract fundamental features of data structures, such as the content of a dynamically allocated list, a stack, a queue, or an array.

* Work partially supported by Hasler Stiftung, ManCom project, grant #2146.

We take a new angle on reasoning about sequences, based on the *theory of concatenation*: a first-order theory where variables are interpreted as words (or sequences) over a finite alphabet and can be composed by concatenating them. Makanin's algorithm for solving word equations [14] implies the decidability of the quantifier-free fragment of the theory of concatenation. Based on this, we introduce a first-order *theory of sequences* $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ whose elements are integers. Section 3.3 presents a decision procedure for the quantifier-free fragment of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$, which encodes the validity problem into the quantifier-free theory of concatenation. The decision procedure is in PSPACE; it is known, however, that Makanin's algorithm is reasonably efficient in practice [1].

The theory $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ allows concatenating sequences to build new ones, and it includes Presburger arithmetic over elements. The resulting quantifier-free fragment has significant expressiveness and can formalize sophisticated properties such as sortedness, injectivity, and Boolean combinations of arithmetic facts relating elements and their indices in statements such as "for all even $i$'s, the element with index $i$ has value $i+3$ or $2i$" (see more examples in Section 3.2). It is remarkable that some of these properties are inexpressible in powerful decidable array logics such as those in [4,10,12,11].

On the other hand, $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ forbids explicit indexed access to elements. This restriction, which is required to have a decidable fragment, prevents the explicit modeling of updatable memory operations such as "swap the first element with the element at index $i$", where $i$ is a scalar program variable. It also differentiates $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ from the theory of arrays and extensions thereof (see Section 5), which can formalize such operations.

In summary, the theory of sequences $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ provides a fresh angle on reasoning "natively" about sequences of integers by means of an abstraction that is orthogonal to most available approaches and can be practically useful (see examples in Section 4). To our knowledge, the approach of the present paper is distinctly new. The absence of prior work on decision procedures for theories of sequences prompted us to compare the expressiveness of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ against that of theories of arrays, which are probably the closest fragments studied. However, the two theories are not meant as direct competitors, as they pertain to partially overlapping, yet largely distinct, domains.

In order to assess the limits of our theory of sequences better, we also prove that several natural extensions of the quantifier-free fragment of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ are undecidable. Finally, we demonstrate reasoning about sequence-manipulating programs with annotations written in the quantifier-free fragment of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$: a couple of examples in Section 4 illustrate the usage of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ formulas with the standard machinery of axiomatic semantics and backward reasoning.

*Remark.* For space constraints, some details and proofs are deferred to [9].

## 2 The Theory of Concatenation

In the rest of the paper, we assume familiarity with the standard syntax and terminology of first-order theories (e.g., [3]); in particular, we assume the standard abbreviations and symbols of first-order theories with the following operator precedence: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, \forall$ and $\exists$. $FV(\phi)$ denotes the set of free variables of a formula $\phi$. With standard terminology, a formula $\phi$ is a *sentence* iff it is *closed* iff $FV(\phi) = \emptyset$. A set $\mathcal{Q}$ of

strings over $\{\exists, \forall\}$ (usually given in the form of a regular expression) denotes the $\mathcal{Q}$-fragment of a first-order theory: the set of all formulas of the theory in the form $\partial_1 v_1 \partial_2 v_2 \cdots \partial_n v_n \bullet \psi$, where $\partial_1 \partial_2 \cdots \partial_n \in \mathcal{Q}$, $v_1, v_2, \ldots, v_n \in FV(\psi)$, and $\psi$ is quantifier-free. The universal and existential fragments are synonyms for the $\forall^*$- and $\exists^*$-fragment respectively. A fragment is decidable iff the validity problem is decidable for its sentences. It is customary to define the validity and satisfiability problems for a quantifier-free formula $\psi$ as follows: $\psi$ is valid iff the universal closure of $\psi$ is valid, and $\psi$ is satisfiable iff the existential closure of $\psi$ is valid. As a consequence of this definition, the decidability of a quantifier-free fragment whose formulas are closed under negation is tantamount to the decidability of the universal or existential fragments. Correspondingly, in the paper we will allow some freedom in picking the terminology that is most appropriate to the context.

*Sequences and concatenation.* $\mathbb{Z}$ denotes the set of integer numbers and $\mathbb{N}$ denotes the set of nonnegative integers. Given a set $A = \{a, b, c, \ldots\}$ of constants, a *sequence* over $A$ is any word $v = v(1)v(2)\cdots v(n)$ for some $n \in \mathbb{N}$ where $v(i) \in A$ for all $1 \leq i \leq n$. The symbol $\epsilon$ denotes the *empty sequence*, for which $n = 0$. $|v| = n$ denotes the *length* of $v$. $A^*$ denotes the set of all finite sequences over $A$ including $\epsilon \notin A$.

It is also convenient to introduce the shorthand $v(k_1, k_2)$ with $k_1, k_2 \in \mathbb{Z}$ to describe *subsequences* of a given sequence $v$. Informally, for positive $k_1, k_2$, $v(k_1, k_2)$ denotes the subsequence starting at position $k_1$ and ending at position $k_2$, both inclusive. For negative or null $k_1, k_2$, $v(k_1, k_2)$ denotes instead the "tail" subsequence starting from the $|k_1|$-to-last element and ending at the $|k_2|$-to-last element. Finally, for positive $k_1$ and negative or null $k_2$, $v(k_1, k_2)$ denotes the subsequence starting at position $k_1$ and ending at the $|k_2|$-to-last element. Formally, we have the following definition.

$$v(k_1, k_2) \triangleq \begin{cases} v(k_1)v(k_1+1)\cdots v(k_2) & 1 \leq k_1 \leq k_2 \leq |v| \\ v(k_1, |v| + k_2) & k_1 - |v| \leq k_2 < 1 \leq k_1 \\ v(|v| + k_1, |v| + k_2) & 1 - |v| \leq k_1 \leq k_2 < 1 \\ \epsilon & \text{otherwise} \end{cases}$$

For two sequences $v_1, v_2 \in A^*$, $v_1 \star v_2$ denotes their *concatenation*: the sequence $v_1(1)\cdots v_1(|v_1|)v_2(1)\cdots v_2(|v_2|)$. We will drop the concatenation symbol whenever unambiguous. The structure $\langle A^*, \star, \epsilon\rangle$ is also referred to as the *free monoid* with generators in $A$ and neutral element $\epsilon$. The size $|A|$ is the *rank* of the free monoid and it can be finite or infinite.

*Decidability in the theory of concatenation.* The theory of concatenation is the first-order theory $\mathcal{T}_{\mathsf{cat}}$ with signature

$$\Sigma_{\mathsf{cat}} \quad \triangleq \quad \{\doteq, \circ, \mathcal{R}\}$$

where $\doteq$ is the equality predicate,[1] $\circ$ is the binary concatenation function and $\mathcal{R} \triangleq \{\mathrm{R}_1, \mathrm{R}_2, \ldots\}$ is a set of unary (monadic) predicate symbols called *regularity constraints*. We sometimes write $R_i(x)$ as $x \in R_i$ and $\alpha \not\doteq \beta$ abbreviates $\neg(\alpha \doteq \beta)$.

[1] We use the symbol $\doteq$ to distinguish it from the standard arithmetic equality symbol $=$ used later in the paper.

An interpretation of a formula in the theory of concatenation is a structure $\langle A^*, \star, \epsilon, \underline{\mathcal{R}}, ev\rangle$ where $\langle A^*, \star, \epsilon\rangle$ is a free monoid, $\underline{\mathcal{R}} = \{\underline{\mathrm{R}}_1, \underline{\mathrm{R}}_2, \ldots\}$ is a collection of regular subsets of $A^*$, and $ev$ is a mapping from variables to values in $A^*$. The satisfaction relation $\langle A^*, \star, \epsilon, \underline{\mathcal{R}}, ev\rangle \models \phi$ for formulas in $\mathcal{T}_{\mathsf{cat}}$ is defined in a standard fashion with the following assumptions: (1) any variable $x$ takes the value $ev(x) \in A^*$; (2) the concatenation $x \circ y$ of two variables $x, y$ takes the value $ev(x) \star ev(y)$; (3) for each $\mathrm{R}_i \in \mathcal{R}$, the corresponding $\underline{\mathrm{R}}_i \in \underline{\mathcal{R}}$ defines the set of sequences $x \in \underline{\mathrm{R}}_i$ for which $\mathrm{R}_i(x)$ holds (this also subsumes the usage of constants).

The following proposition summarizes some decidability results about fragments of the theory of concatenation; they all are known results, or corollaries of them [9].

**Proposition 1.** *1. [14,7,17] The universal and existential fragments of the theory of concatenation over free monoids with finite rank are* decidable *in PSPACE.*
2. *The following fragments of the theory of concatenation are* undecidable*.*
   (a) *[8] The $\forall^*\exists^*$ and $\exists^*\forall^*$ fragments.*
   (b) *[5] The extensions of the existential and universal fragments over the free monoid $\{a, b\}^*$ with: (1) two length functions $|x|_a, |x|_b$ where $|x|_p \triangleq \{y \in p^* \mid y$ has the same number of $p$'s as $x\}$; or (2) the function $Sp(x) \triangleq |x|_a \star |x|_b$.*
3. *[5] The following are* not definable *in the existential or universal fragments.*
   (a) *The set $S^= \triangleq \{a^n b^n \mid n \in \mathbb{N}\}$.*
   (b) *The equal length predicate $Elg(x, y) \triangleq |x| = |y|$.*

It is currently unknown whether the extension of the existential or universal fragment of concatenation with $Elg$ is decidable.

## 3 A Theory of Sequences

This section introduces a first-order theory of sequences (Section 3.1) with arithmetic, demonstrates it on a few examples (Section 3.2), gives a decision procedure for its universal fragment (Sections 3.3– 3.4), and shows that "natural" larger fragments are undecidable (Section 3.5).

### 3.1 A Theory of Integer Sequences

We present an arithmetic theory of sequences whose elements are integers. It would be possible to make the theory parametric with respect to the element type. Focusing on integers, however, makes the presentation clearer and more concrete, with minimal loss of generality as one can encode any theory definable in the integer arithmetic fragment.

*Syntax.* Properties of integers are expressed in Presburger arithmetic with signature:

$$\Sigma_{\mathbb{Z}} \quad \triangleq \quad \{0, 1, +, -, =, <\}$$

Then, our theory $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ of sequences with integer values has signature:

$$\Sigma_{\mathsf{seq}(\mathbb{Z})} \quad \triangleq \quad \Sigma_{\mathsf{cat}} \cup \Sigma_{\mathbb{Z}}$$

Operator precedence is: $\circ$; $+$ and $-$; $\doteq, =$ and $<$, followed by logic connectives and quantifiers with the previously defined precedence.

We will generally consider formulas in prenex normal form $\mathcal{Q} \bullet \psi$ , where $\mathcal{Q}$ is a quantifier prefix and $\psi$ is quantifier-free written in the grammar:

$$\begin{array}{rcl} seq & ::= & var \mid 0 \mid 1 \mid seq \circ seq \mid seq + seq \mid seq - seq \\ fmla & ::= & seq \doteq seq \mid R(seq) \mid seq = seq \mid seq < seq \\ & & \mid \neg fmla \mid fmla \vee fmla \mid fmla \wedge fmla \mid fmla \Rightarrow fmla \end{array}$$

with $var$ ranging over variable names.

*Semantics.* An interpretation of a sentence of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ is a structure $\langle \mathbb{Z}^*, \star, \epsilon, \underline{\mathcal{R}}, ev \rangle$ with the same meaning as in the theory of concatenation plus the following additional assumptions about arithmetic.[2]

- The interpretation of a sequence $v_1 v_2 \cdots \in \mathbb{Z}^*$ of integers in an integer sub-expression (e.g., in a sum, in an integer equality or inequality) is the first integer in the sequence $v_1$, with the convention that the interpretation of the empty sequence is 0.[3]
- Conversely, the interpretation of an integer $v \in \mathbb{Z}$ in a sequence sub-expression (e.g., in a concatenation) is the singleton sequence $v$.
- Addition, subtraction, equality, and less than are interpreted accordingly.

For example, the expression $(1 \circ 0 \circ 0 < 1 + 0 + 1) \wedge (1 \circ 0 = 1 \circ 1)$ evaluates to true because the sequences $1 \circ 0 \circ 0$, $1 \circ 0$, and $1 \circ 1$ are all interpreted as the integer 1.

*Shorthands.* We introduce several simplifying shorthands.

- A symbol for every constant $k \in \mathbb{Z}$, defined as obvious.
- $\alpha \neq \beta$, $\alpha \leq \beta$, $\alpha \geq \beta$, and $\alpha > \beta$ defined respectively as $\neg(\alpha = \beta)$, $\alpha < \beta \vee \alpha = \beta$, $\neg(\alpha < \beta)$, and $\alpha \geq \beta \wedge \alpha \neq \beta$.
- Shorthands such as $\alpha \leq \beta < \gamma$ or $\beta \in [\alpha, \gamma)$ for $\alpha \leq \beta \wedge \beta < \gamma$.
- Bounded length predicates such as $|x| < k$ for a variable $x$ and a constant $k \in \mathbb{Z}$. $|x| < k$ abbreviates $\mathrm{R}^{<k}(x)$, where $\mathrm{R}^{<k}$ is a regularity constraint that stands for $\{\epsilon\} \cup \bigcup_{0<i<k} \mathbb{Z}^i$.
  The definition of derived expressions such as $k_1 \leq |x| < k_2$ is straightforward.
- Subsequence functions such as $x(k_1, k_2)$ for a variable $x$ and two constants $k_1, k_2 \in \mathbb{Z}$ with the intended semantics (see Section 2). We define these functions in the theory $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ by the following rewriting rules, defined on formulas in prenex normal form with quantifier prefix $\mathcal{Q}$:

$$\frac{\mathcal{Q} \bullet \psi[x(k_1, k_2)]}{\mathcal{Q} \forall u, v, w \bullet \begin{pmatrix} \kappa_1 \wedge x \doteq uvw \wedge |u| = k_1 - 1 \wedge |v| = k_2 - k_1 + 1 \\ \vee \quad \kappa_2 \wedge x \doteq uvw \wedge |u| = k_1 - 1 \wedge |w| = -k_2 \\ \vee \quad \kappa_3 \wedge x \doteq uvw \wedge |v| = -k_1 + k_2 + 1 \wedge |w| = -k_2 \\ \vee \quad \neg(\kappa_1 \vee \kappa_2 \vee \kappa_3) \wedge u \doteq v \doteq w \doteq \epsilon \end{pmatrix} \Rightarrow \psi[v]}$$

[2] The presentation of the semantics of the theory is informal and implicit for brevity.

[3] The results of Section 3.5 suggest that interpretations aggregating the values of multiple sequence elements are likely to be undecidable.

where $\kappa_1 \triangleq 1 \leq k_1 \leq k_2 \leq |x|$, $\kappa_2 \triangleq k_1 - |x| \leq k_2 < 1 \leq k_1$, $\kappa_3 \triangleq 1 - |x| \leq k_1 \leq k_2 < 1$.
- $\mathsf{fst}(x)$ and $\mathsf{lst}(x)$ for the first $x(1,1)$ and last element $x(0,0)$ of $x$, respectively.

### 3.2 Examples

A few examples demonstrate the expressiveness of the universal fragment of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$.

1. **Equality**: sequences $\mathsf{u}$ and $\mathsf{v}$ are equal.
$$\mathsf{u} \doteq \mathsf{v} \tag{1}$$
2. **Bounded equality**: sequences $\mathsf{u}$ and $\mathsf{v}$ are equal in the *constant* interval $[l, u]$ for $l, u \in \mathbb{Z}$.
$$\mathsf{u}(l,u) \doteq \mathsf{v}(l,u) \tag{2}$$
3. **Boundedness**: no element in sequence $\mathsf{u}$ is greater than value $\mathsf{v}$.
$$\forall h,t \bullet \mathsf{u} \doteq ht \wedge |t| > 0 \;\Rightarrow\; t \leq \mathsf{v} \tag{3}$$
4. **Sortedness**: sequence $\mathsf{u}$ is sorted (strictly increasing).
$$\forall h,m,t \bullet \mathsf{u} \doteq hmt \wedge |m| = 1 \wedge |t| > 0 \;\Rightarrow\; m < t \tag{4}$$
5. **Injectivity**: $\mathsf{u}$ has no repeated elements.
$$\forall h, v_1, m, v_2, t \bullet \mathsf{u} \doteq hv_1mv_2t \wedge |v_1| = 1 \wedge |v_2| = 1 \Rightarrow v_1 \neq v_2 \tag{5}$$
6. **Partitioning**: sequence $\mathsf{u}$ is partitioned at *constant* position $k > 0$.
$$\forall h_1, t_1, h_2, t_2 \bullet \begin{pmatrix} \mathsf{u}(1,k) \doteq h_1t_1 \\ \wedge\, \mathsf{u}(k+1,0) \doteq h_2t_2 \\ \wedge\, |t_1| > 0 \wedge |t_2| > 0 \end{pmatrix} \Rightarrow t_1 < t_2 \tag{6}$$
7. **Membership**: *constant* element $k \in \mathbb{Z}$ occurs in sequence $\mathsf{u}$.
$$\mathsf{u} \in (\mathbb{Z}^* k \mathbb{Z}^*) \tag{7}$$
8. **Non-membership**: no element in sequence $\mathsf{u}$ has value $\mathsf{v}$.
$$\forall h,t \bullet \mathsf{u} \doteq ht \wedge |t| > 0 \;\Rightarrow\; t \neq \mathsf{v} \tag{8}$$
9. **Periodicity**: in non-empty sequence $\mathsf{u}$, elements on even positions have value 0 and elements on odd positions have value 1 (notice that $\mathsf{lst}(h) = 0$ if $h$ is empty).
$$\forall h,t \bullet \mathsf{u} \doteq ht \wedge |t| > 0 \Rightarrow \begin{pmatrix} \mathsf{lst}(h) = 1 \\ \Rightarrow t = 0 \end{pmatrix} \wedge \begin{pmatrix} \mathsf{lst}(h) = 0 \\ \Rightarrow t = 1 \end{pmatrix} \tag{9}$$
10. **Comparison** between indices and values: for every index $i$, element at position $i$ in the non-empty sequence $\mathsf{u}$ has value $i + 3$.
$$\mathsf{fst}(\mathsf{u}) = 1{+}3 \wedge \forall h,t,v \bullet \mathsf{u} \doteq ht \wedge |h| > 0 \wedge |t| > 0 \wedge \mathsf{lst}(h) = v \Rightarrow \mathsf{fst}(t) = v{+}1 \tag{10}$$
11. **Disjunction of value constraints**: for every pair of positions $i < j$ in the sequence $\mathsf{u}$, either $\mathsf{u}(i,i) \leq \mathsf{u}(j,j)$ or $\mathsf{u}(i,i) \geq 2\mathsf{u}(j,j)$.
$$\forall h, v_1, m, v_2, t \bullet \mathsf{u} \doteq hv_1mv_2t \wedge |v_1| > 0 \wedge |v_2| > 0 \Rightarrow v_1 \leq v_2 \vee v_1 \geq v_2 + v_2 \tag{11}$$

*Comparison with theories of arrays.* Properties such as strict sortedness (4), periodicity (9), and comparisons between indices and values (10) are inexpressible in the array logic of Bradley et al. [4]. The latter is inexpressible also in the logic of Ghilardi et al. [10] because Presburger arithmetic is restricted to indices. Properties such as (11) are inexpressible both in the SIL array logic of [11] — because quantification on multiple array indices is disallowed — and in the related LIA logic of [12] — because disjunctions of comparisons of array elements are disallowed. Extensions of each of these logics to accommodate the required features would be undecidable.

Conversely, properties such as *permutation*, bounded equality *for an interval specified by indices*, length constraints *for a variable value*, membership *for a variable value*, and the *subsequence* relation, are inexpressible in the universal fragment of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$. Membership and the subsequence relation are expressible in the dual existential fragment of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$, while the other properties seem to entail undecidability of the corresponding $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ fragment (see Section 3.5). Bounded equality, length constraints, and membership, on the other hand, are expressible in all the logics of [4,10,11,12], and [10] outlines a decidable extension which supports the subsequence relation (see Section 5).

### 3.3 Deciding Properties of Integer Sequences

This section presents a decision procedure $\mathcal{D}_{\mathsf{seq}(\mathbb{Z})}$ for the universal fragment of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$. The procedure transforms any universal $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ formula into an equi-satisfiable universal formula in the theory of concatenation over the free monoid $\{a, b, c, d\}^*$. The basic idea is to encode integers as sequences over the four symbols $\{a, b, c, d\}$: the sequence $acb^{k_1}a$ encodes a nonnegative integer $k_1$, while the sequence $adb^{-k_2}a$ encodes a negative integer $k_2$. Suitable rewrite rules encode all quantifier-free Presburger arithmetic in accordance with this convention. Subsection 3.4 illustrates the correctness and complexity of $\mathcal{D}_{\mathsf{seq}(\mathbb{Z})}$.

Consider a universal formula of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ in prenex normal form:

$$\forall x_1, \ldots, x_v \bullet \psi \tag{12}$$

where $\psi$ is quantifier-free. Modify (12) by application of the following steps.

1. Introduce fresh variables to normalize formulas into the following form:

$$\begin{aligned} fmla \quad ::= \quad & var \doteq var \mid var \doteq var \circ var \mid R(var) \mid var = 0 \mid var = 1 \\ & \mid var = var \mid var = var + var \mid var = var - var \mid var < var \\ & \mid \neg fmla \mid fmla \vee fmla \mid fmla \wedge fmla \mid fmla \Rightarrow fmla \end{aligned}$$

   Clearly, we can achieve this by applying exhaustively rewrite rules that operate on $\psi$ such as:

$$\frac{\psi[x \circ y]}{e \doteq x \circ y \Rightarrow \psi[e]} \qquad \frac{\psi[x + y]}{f = x + y \Rightarrow \psi[f]}$$

   for fresh variables $e, f$.

2. For each variable $x_i \in FV(\psi) = \{x_1, \ldots, x_v\}$, introduce the fresh variables $h_i, t_i, s_i, m_i$ (for head, tail, sign, modulus) and rewrite $\psi$ as:

$$\bigwedge_{1 \leq i \leq v} \left( \begin{pmatrix} x_i \doteq h_i t_i \\ \wedge\ h_i \doteq a s_i m_i a \\ \wedge\ \begin{pmatrix} (s_i \in \{c\} \wedge m_i \in b^*) \\ \vee\ (s_i \in \{d\} \wedge m_i \in b^+) \end{pmatrix} \\ \wedge\ t_i \in (acb^*a \cup adb^+a)^* \end{pmatrix} \vee \begin{pmatrix} x_i \doteq \epsilon \\ \wedge\ h_i \doteq a s_i m_i a \\ \wedge\ s_i \doteq c \\ \wedge\ m_i \doteq \epsilon \\ \wedge\ t_i \doteq \epsilon \end{pmatrix} \right) \Rightarrow \psi$$

3. Apply the following rules exhaustively to remove arithmetic equalities:

$$\frac{\psi[x_i = x_j]}{\psi[h_i \doteq h_j]} \qquad \frac{\psi[x_i = 0]}{\psi[h_i \in 0]} \qquad \frac{\psi[x_i = 1]}{\psi[h_i \in 1]}$$

4. Apply the following rule exhaustively to remove differences:

$$\frac{\psi[x_k = x_i - x_j]}{\psi[x_i = x_k + x_j]}$$

5. Apply the following rule exhaustively to remove comparisons:

$$\frac{\psi[x_i < x_j]}{\begin{pmatrix} m_i \doteq m_j \\ \vee\ m_i \doteq m_j p \\ \vee\ m_j \doteq m_i p \end{pmatrix} \Rightarrow \psi \begin{bmatrix} s_i \doteq d \wedge s_j \doteq c \\ \vee \\ s_i \doteq s_j \doteq c \wedge m_j \doteq m_i p \\ \vee \\ s_i \doteq s_j \doteq d \wedge m_i \doteq m_j p \end{bmatrix}}$$

for fresh $p \in b^+$.

6. Apply the following rule exhaustively to remove sums:

$$\frac{\psi[x_k = x_i + x_j]}{\begin{pmatrix} m_i \doteq m_j \\ \vee\ m_i \doteq m_j p \\ \vee\ m_j \doteq m_i p \end{pmatrix} \Rightarrow \psi \begin{bmatrix} s_i \doteq s_j \wedge x_k \doteq a s_i m_i m_j a \\ \vee \\ s_i \neq s_j \wedge m_i \doteq m_j \wedge x_k \doteq aca \\ \vee \\ s_i \neq s_j \wedge m_i \doteq m_j p \wedge x_k \doteq a s_i p a \\ \vee \\ s_i \neq s_j \wedge m_j \doteq m_i p \wedge x_k \doteq a s_j p a \end{bmatrix}}$$

for fresh $p \in b^+$.

7. Modify the meaning of regularity constraints as follows: let $\underline{\mathrm{R}}_i$ be defined by a regular expression with constants in $\mathbb{Z}$. Substitute every occurrence of a nonnegative constant $k \in \mathbb{Z}$ by $acb^k a$; every occurrence of a negative constant $k \in \mathbb{Z}$ by $adb^{-k}a$; every occurrence of set $\mathbb{Z}$ by $acb^*a \cup adb^+a$.

The resulting formula is again in form (12) where $\psi$ is now a quantifier-free formula in the theory of concatenation over $\{a, b, c, d\}^*$; its validity is decidable by Proposition 1.

### 3.4 Correctness and Complexity

Let us sketch the correctness argument for the decision procedure $\mathcal{D}_{\mathsf{seq}(\mathbb{Z})}$, which shows that the transformed formula is equi-satisfiable with the original one.

The justification for step 1 is straightforward. The following steps introduce a series of substitutions to eliminate arithmetic by reducing it to equations over the theory of concatenation with the unary encoding of integers defined above.

Step 2 requires that any variable $x_i$ is a sequence of the form $(acb^*a \cup adb^+a)^*$ and introduces fresh variables to denote significant parts of the sequence: $h_i$ aliases the first element of the sequence which is further split into its sign $s_i$ ($c$ for nonnegative and $d$ for negative) and its absolute value $m_i$ encoded as a unary string in $b^*$. The second term of the disjunction deals with the case of $x_i$ being $\epsilon$, which has the same encoding as $0$.

The following steps replace elements of the signature of Presburger arithmetic by rewriting them as equations over sequences with the given encoding. Step 3 reduces the arithmetic equality of two sequences of integers to equivalence of the sequences encoding their first elements. Step 4 rewrites equations involving differences with equations involving sums.

Step 5 reduces arithmetic comparisons of two sequences of integers to a case discussion over the sequences $h_i, h_j$ encoding their first elements. Let $p$ be a sequence in $b^+$ encoding the difference between the absolute values corresponding to $h_i$ and $h_j$; obviously such a $p$ always exists unless the absolute values are equal. Then, $h_i$ encodes an integer strictly less than $h_j$ iff one of the following holds: (1) $h_i$ is a negative value and $h_j$ is a nonnegative one; (2) both $h_i$ and $h_j$ are nonnegative values and the sequence of $b$'s in $h_j$ is longer than the sequence of $b$'s in $h_i$; or (3) both $h_i$ and $h_j$ are negative values and the sequence of $b$'s in $h_i$ is longer than the sequence of $b$'s in $h_j$.

Step 6 reduces the comparison between the value of a sum of two variables and a third variable to an analysis of the three sequences $h_i, h_j, h_k$ encoding the first elements of the three variables. As in step 6, the unary sequence $p$ encodes the difference between the absolute values corresponding to $h_i$ and $h_j$. Then, $h_k$ encodes the sum of the values encoded by $h_i$ and $h_j$ iff one of the following holds: (1) $h_i$ and $h_j$ have the same sign and $h_k$ contains a sequence of $b$'s which adds up the sequences of $b$'s of $h_i$ and $h_j$, still with the same sign; (2) $h_i$ and $h_j$ have opposite sign but same absolute value, so $h_k$ must encode $0$; (3) $h_i$ and $h_j$ have opposite sign and the absolute value of $h_i$ is greater than the absolute value of $h_j$, so $h_k$ has the same sign as $h_i$ and the difference of absolute values as its absolute value; or (4) $h_i$ and $h_j$ have opposite sign and the absolute value of $h_j$ is greater than the absolute value of $h_i$, so $h_k$ has the same sign as $h_j$ and the difference of absolute values as its absolute value.

Finally, step 7 details how to translate the interpretation of the regularity constraints over $\mathbb{Z}$ into the corresponding regularity constraints over $\{a, b, c, d\}$ with the given integer encoding.

It is not difficult to see that all rewriting steps in the decision procedure $\mathcal{D}_{\mathsf{seq}(\mathbb{Z})}$ increase the size of $\psi$ at most quadratically (this accounts for fresh variables as well). Hence, the PSPACE complexity of the universal fragment of the theory of concatenation (Proposition 1) carries over to $\mathcal{D}_{\mathsf{seq}(\mathbb{Z})}$.

**Theorem 1.** *The universal fragment of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ is decidable in PSPACE with the decision procedure $\mathcal{D}_{\mathsf{seq}(\mathbb{Z})}$.*

### 3.5 Undecidable Extensions

**Theorem 2.** *The following extensions of the $\forall^*$-fragment of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ are undecidable.*

1. *The $\forall^*\exists^*$ and $\exists^*\forall^*$ fragments.*
2. *For any pair of integer constants $k_1, k_2$, the extension with the two length functions $|x|_{k_1}, |x|_{k_2}$ counting the number of occurrences of $k_1$ and $k_2$ in $x$.*
3. *The extension with an equal length predicate $Elg(x, y) \triangleq |x| = |y|$.*
4. *The extension with a sum function $\sigma(x) \triangleq \sum_{i=1}^{|x|} x(i, i)$.*

*Proof.* 1. Sentences with one quantifier alternation are undecidable already for the theory of concatenation without arithmetic and over a monoid of finite rank (Proposition 1). Notice that the set of sentences that are expressible both in the $\forall^*\exists^*$ and in the $\exists^*\forall^*$ fragment is decidable [18, Th. 4.4]; however, this set lacks a simple syntactic characterization.

2. Corollary of Proposition 1.

3. We encode the universal theory of $\mathit{\Pi} = \langle \mathbb{N}, 0, 1, +, \pi \rangle$ — where $\pi(x, y) \triangleq x2^y$ — in the universal fragment of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ extended by the $Elg$ predicate; undecidability follows from the undecidability of the existential and universal theories of $\mathit{\Pi}$ [5, Corollary 5]. All we have to do is showing that $\pi(x, y) = p$ is universally definable in $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ with $Elg$. To this end, define $l_y$ as a sequence that begins with value $y$, ends with value 1, and where every element is the successor of the following.

   $$\forall h, t \bullet \mathsf{fst}(l_y) = y \wedge \mathsf{lst}(l_y) = 1 \wedge l_y \doteq ht \wedge |h| > 0 \wedge |t| > 0 \Rightarrow \mathsf{lst}(h) = t + 1$$

   As a result $l_y$ is in the form $y, y-1, \ldots, 1$ and hence has length $y$.[4] Then, $\pi(x, y)$ is universally definable as the sequence $p$ with the same length as $l_y$, whose last element is $x$, and where every element is obtained by doubling the value of the element that follows:

   $$\forall g, u \bullet Elg(p, l_y) \wedge \mathsf{lst}(p) = x \wedge p \doteq gu \wedge |g| > 0 \wedge |u| > 0 \Rightarrow \mathsf{lst}(g) = u + u$$

   Hence $p$ has the form $2^y x, 2^{y-1}x, \ldots, 2^2 x, 2x, x$ which encodes the desired value $x2^y$ in $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$. (Notice that the two universal definitions of $l_y$ and $p$ can be combined into a single universal definition by conjoining the definition of $p$ to the consequent in the definition of $l_y$).

4. For any sequence $x$ over $\{0, 1\}$ define $Sp(x) = y$ as $y \in 0^*1^* \wedge \sigma(y) = \sigma(x)$. Then, Proposition 1 implies undecidability because this extension of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ can define universal sentences over the free monoid $\{a, b\}^*$ with the function $Sp$. □

The decidability of the following is instead currently unknown: the extension of the universal fragment with a function $x \oplus 1$ defined as the sequence $x(1) + 1, x(2) + 1, \ldots, x(|x|) + 1$. The fragment allows the definition of the set $S^{=}\{0^n1^n \mid n \in \mathbb{N}\}$ as the sequences $x$ such that $x \in 0^*1^* \wedge \forall u, v \bullet x \doteq uv \wedge u \in 0^* \wedge v \in 1^* \Rightarrow u \oplus 1 \doteq v$. This is inexpressible in the universal fragment of the theory of concatenation, but the decidability of the resulting fragment is currently unknown (see Proposition 1).

[4] This technique would allow the definition of the length function $|x|$ and full index arithmetic.

```
1  merge_sort (a: ARRAY): ARRAY
2  local l, r: ARRAY
3  do
4     if |a| ≤ 1 then
5        { sorted(a) }
6        Result := a
7     else
8        l , r := a[1:|a|/2] , a[|a|/2+1: |a|]
9        { l ∗ r = a }
10       l , r := merge_sort (l) , merge_sort (r)
11       { sorted(l) ∧ sorted(r) }
12       from Result := ϵ
13       { invariant sorted(Result) ∧ sorted(l) ∧ sorted(r) ∧
14                   lst(Result) ≤ fst(l) ∧ lst(Result) ≤ fst(r) }
15       until |l| = 0 ∨ |r| = 0
16       loop
17          if l.first > r.first then
18             Result := Result ∗ r.first ; r := r.rest
19          else
20             Result := Result ∗ l.first ; l := l.rest
21          end
22       end
23       if |l| > 0 then
24          { |r| = 0 } Result := Result ∗ l
25       else
26          { |l| = 0 } Result := Result ∗ r
27       end
28 { ensure sorted(Result) }
```

```
1  reverse (a: LIST): LIST
2  local v: INTEGER ; s: STACK
3  do
4     from s := ϵ
5     { invariant s ∘ a = old a }
6     until a = ϵ
7     loop
8        s.push (a.first)
9        a := a.rest
10    end
11    from Result := ϵ
12    { invariant
13         s ∘ Result^R = old a }
14    until s = ϵ
15    loop
16       v := s.top
17       s.pop ; Result.extend (v)
18    end
19 { ensure Result^R = old a}
```

**Table 1.** Annotated Mergesort (left) and Array Reversal (right).

## 4 Verifying Sequence-Manipulating Programs

This section outlines a couple of examples that use formulas in the theory $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ to reason about sequence-manipulating programs. An implementation of the decision procedure $\mathcal{D}_{\mathsf{seq}(\mathbb{Z})}$ is needed to tackle more extensive examples; it belongs to future work. The examples are in Eiffel-like pseudo-code [15]; it is not difficult to detail an axiomatic semantics and a backward substitution calculus, using the universal fragment of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$, for the portions of this language employed in the examples.

*Mergesort.* Consider a straightforward recursive implementation of the Mergesort algorithm; Table 1 (left) shows an annotated version, where $*$ denotes the concatenation operator in the programming language (whose semantics is captured by the corresponding logic operator $\circ$). The annotations specify that the routine produces a sorted array, where predicate $\mathrm{sorted}(\mathsf{u})$ is defined as (cmp. (4)):

$$\mathrm{sorted}(\mathsf{u}) \quad \triangleq \quad \forall h, m, t \bullet \mathsf{u} \doteq hmt \wedge |m| > 0 \wedge |t| > 0 \Rightarrow m \leq t$$

It is impossible to express in $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ another component of the full functional specification: the output is a permutation of the input. This condition is inexpressible in most of the expressive decidable extensions of the theory of arrays that are currently known, such as [4,11] (see also Section 5). Complementary automated verification techniques — using different abstractions such as the multiset [16] — can, however, verify this orthogonal aspect.

We must also abstract away the precise splitting of array $a$ into two halves in line 8. The way in which $a$ is partitioned into $l$ and $r$ is however irrelevant as far as correctness is concerned (it only influences the complexity of the algorithm), hence we can simply over-approximate the instruction on line 8 by a nondeterministic splitting in two continuous non-empty parts.

From the annotated program, we can generate verification conditions by standard backward reasoning. Universal sentences of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ can express the verification conditions, hence the verification process can be automated. Let us see an example on the non-trivial part of the process, namely checking that the formula on lines 13–14 is indeed an inductive invariant. Consider the "then" branch on line 18. Backward substitution of the invariant yields:

$$\begin{array}{c} \mathrm{sorted}(\mathbf{Result} * \mathsf{fst}(r)) \ \wedge \ \mathrm{sorted}(l) \ \wedge \ \mathrm{sorted}(r(2,0)) \ \wedge \\ \mathsf{lst}(\mathbf{Result} * \mathsf{fst}(r)) \leq \mathsf{fst}(l) \ \wedge \ \mathsf{lst}(\mathbf{Result} * \mathsf{fst}(r)) \leq \mathsf{fst}(r(2,0)) \end{array} \tag{13}$$

This condition must be discharged by the corresponding loop invariant hypothesis:

$$\begin{array}{c} \mathsf{fst}(l) > \mathsf{fst}(r) \ \wedge \ \mathrm{sorted}(\mathbf{Result}) \ \wedge \ \mathrm{sorted}(l) \ \wedge \ \mathrm{sorted}(r) \ \wedge \\ \mathsf{lst}(\mathbf{Result}) \leq \mathsf{fst}(l) \ \wedge \ \mathsf{lst}(\mathbf{Result}) \leq \mathsf{fst}(r) \ \wedge \ |l| \neq 0 \ \wedge \ |r| \neq 0 \end{array} \tag{14}$$

Checking that (14) entails (13) discharges the corresponding verification condition. Elements of this condition can be encoded in the universal fragment of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ and proven using the decision procedure of Section 3.3; for instance, the fact that $\mathsf{lst}(\mathbf{Result}) \leq \mathsf{fst}(l)$, $|l| \neq 0$, $|r| \neq 0$, and $\mathsf{fst}(l) > \mathsf{fst}(r)$ imply $\mathsf{lst}(\mathbf{Result} * \mathsf{fst}(r)) \leq \mathsf{fst}(l)$ corresponds to the validity of (all free variables are implicitly universally quantified):

$$\begin{pmatrix} r \doteq h_r m_r t_r \ \wedge \ |h_r| = 1 \ \wedge \ |r| \neq 0 \\ \wedge \ l \doteq h_l m_l t_l \ \wedge \ |h_l| = 1 \ \wedge \ |l| \neq 0 \\ \wedge \ \mathbf{Result} \circ h_r \doteq hmt \wedge |t| = 1 \\ \wedge \ h_l > h_r \end{pmatrix} \Rightarrow t \leq h_l$$

*Reversal.* In Table 1 (right), a program reverses a sequence of integers, given as a list $a$, using a stack $s$. The queries "first" and "rest" respectively return the first element in a list and a copy of the list without its first element, and the command "extend" adds an element to the right of a list; the query "top" and the commands "pop" and "push" for a stack have the usual semantics. In the annotations, $s$ is modeled by a sequence whose first element is the bottom of the stack; the expression $\mathbf{old}\ a$ denotes the value of $a$ upon entering the routine.

The superscript $^{\mathsf{R}}$ denotes the reversal of a sequence. We do not know if the extension of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ by a reversal function is decidable. However, the following two simple update axioms are sufficient to handle any program which builds the reverse $\mathsf{u}^{\mathsf{R}}$ of a sequence $\mathsf{u}$ starting from an empty sequence and adding one element at a time:

$$\mathsf{u}^{\mathsf{R}} = \epsilon \ \Leftrightarrow \ \mathsf{u} = \epsilon \qquad\qquad |x| = 1 \ \Rightarrow \ (\mathsf{u}x)^{\mathsf{R}} = x\mathsf{u}^{\mathsf{R}}$$

Consider, for instance, the verification condition that checks if the invariant of the second loop (lines 11–18) is indeed inductive:

$$s \circ \mathbf{Result}^{\mathsf{R}} = \mathbf{old}\ a \wedge s \neq \epsilon \quad \Rightarrow \quad s(1,-1) \circ (\mathbf{Result} \circ s(0,0))^{\mathsf{R}} = \mathbf{old}\ a$$

After rewriting $(\mathbf{Result} \circ s(0,0))^{\mathsf{R}}$ into $s(0,0) \circ \mathbf{Result}^{\mathsf{R}}$ the implication is straightforward to check for validity.

## 5 Related Work

A review of the staggering amount of work on decision procedures for theories of complex data types (and integrations thereof) is beyond the scope of this paper; for a partial account see [9,19,13]. In this section, we focus on a few approaches that are most similar to ours and in particular which yield decidable logics that can be compared directly to our theory of sequences (see Section 3.2). The absence of previous work on "direct" approaches to theories of sequences makes the many work on decidable extensions of the theory of arrays the closest to ours in expressive power. As discussed in Section 1, however, our theory of sequences is not meant as a replacement to the theories of arrays, but rather as a complement to them in different domains.

Bradley et al. [4] develop the *array property fragment*, a decidable subset of the $\exists^*\forall^*$ fragment of the theory of arrays. An *array property* is a formula of the form $\exists^*\forall^* \bullet \iota \Rightarrow \nu$, where the universal quantification is restricted to index variables, $\iota$ is a guard on index variables with arithmetic (restricted to existentially quantified variables), and $\nu$ is a constraint on array values without arithmetic or nested reads, and where no universally quantified index variable is used to select an element that is written to. The array property fragment is decidable with a decision procedure that eliminates universal quantification on index variables by reducing it to conjunctions on a suitable finite set of index values. Extensions of the array property fragment that relax any of the restrictions on the form of array properties are undecidable. Bradley et al. also show how to adapt their theory of arrays to reason about maps.

Ghilardi et al. [10] develop "semantic" techniques to integrate decision procedures into a decidable extension of the theory of arrays. Their $\mathcal{ADP}$ theory merges the quantifier-free extensional theory of arrays with dimension and Presburger arithmetic over indices into a decidable logic. Two extensions of the $\mathcal{ADP}$ theory are still decidable: one with a unary predicate that determines if an array is *injective* (i.e., it has no repeated elements); and one with a function that returns the *domain* of an array (i.e., the set of indices that correspond to definite values). Ghilardi et al. suggest that these extensions might be the basis for automated reasoning on Separation Logic models. The framework of [10] also supports other decidable extensions, such as the *prefix*, and *sorting* predicates, as well as the *map* combinator also discussed in [6].

De Moura and Bjørner [6] introduce *combinatory array logic*, a decidable extension of the quantifier-free extensional theory of arrays with the *map* and *constant-value* combinators (i.e., array functors). The *constant-value* combinator defines an array with all values equal to a constant; the *map* combinator applies a $k$-ary function to the elements at position $i$ in $k$ arrays $a_1, \ldots, a_k$. De Moura and Bjørner define a decision procedure for their combinatory array logic, which is implemented in the Z3 SMT solver.

Habermehl et al. introduce powerful logics to reason about arrays with integer values [12,11,2]; unlike most related work, the decidability of their logic relies on automata-theoretic techniques for a special class of counter automata. More precisely, [12] defines the *Logic of Integer Arrays* **LIA**, whose formulas are in the $\exists^*\forall^*$ frag-

ment and allow Presburger arithmetic on existentially quantified variables, difference and modulo constraints on index variables, and difference constraints on array values. Forbidding disjunctions of difference constraints on array values is necessary to ensure decidability. The resulting fragment is quite expressive, and in particular it includes practically useful formulas that are inexpressible in other decidable expressive fragments such as [4]. The companion work [11] introduces the *Single Index Logic* **SIL**, consisting of existentially quantified Boolean combinations of formulas of the form $\forall^* \bullet \iota \Rightarrow \nu$, where the universal quantification is restricted to index variables, $\iota$ is a positive Boolean combination of bound and modulo constraints on index variables, and $\nu$ is a conjunction of difference constraints on array values. Again, the restrictions on quantifier alternations and Boolean combinations are tight in that relaxing one of them leads to undecidability. The expressiveness of **SIL** is very close to that of **LIA**, and the two logics can be considered two variants of the same basic kernel. The other work [2] shows how to use **SIL** to annotate and reason automatically about array-manipulating programs; the tight correspondence between **SIL** and a class of counter automata allows the automatic generation of loop invariants and hence the automation of the full verification process.

## 6 Future Work

Future work will investigate the decidability of the universal fragment of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ extended with "weak" predicates or functions that slightly increase its expressiveness (such as that outlined at the end of Section 3.5). We will study to what extent the decision procedure for the universal fragment of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$ can be integrated with other decidable logic fragments (and possibly with the dual existential fragment). We will investigate how to automate the generation of inductive invariants for sequence-manipulating programs by leveraging the decidability of the universal fragment of $\mathcal{T}_{\mathsf{seq}(\mathbb{Z})}$. Finally, we will implement the decision procedure, integrate it within a verification environment, and assess its empirical effectiveness on real programs.

*Acknowledgements.* Thanks to Nadia Polikarpova and Stephan van Staden for comments on a draft of this paper, and to the anonymous reviewers for their remarks.

## References

1. Abdulrab, H., Pécuchet, J.P.: Solving word equations. Journal of Symbolic Computation 8(5), 499–521 (1989)
2. Bozga, M., Habermehl, P., Iosif, R., Konecný, F., Vojnar, T.: Automatic verification of integer array programs. In: Proceedings of the 21st International Conference on Computer Aided Verification (CAV'09). Lecture Notes in Computer Science, vol. 5643, pp. 157–172. Springer (2009)
3. Bradley, A.R., Manna, Z.: The Calculus of Computation. Springer (2007)
4. Bradley, A.R., Manna, Z., Sipma, H.B.: What's decidable about arrays? In: Proceedings of the 7th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'06). Lecture Notes in Computer Science, vol. 3855, pp. 427–442. Springer (2006)

5. Büchi, J.R., Senger, S.: Definability in the existential theory of concatenation and undecidable extensions of this theory. Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik 34, 337–342 (1988)
6. de Moura, L., Bjørner, N.: Generalized, efficient array decision procedures. In: Proceedings of 9th Conference on Formal Methods in Computer Aided Design (FMCAD'09). pp. 45–52 (2009)
7. Diekert, V.: Makanin's algorithm. In: Lothaire, M. (ed.) Algebraic Combinatorics on Words. Cambridge University Press (2002)
8. Durnev, V.G.: Unsolvability of the positive $\forall\exists^3$-theory of a free semi-group. Sibirskiĭ Matematicheskiĭ Zhurnal 36(5), 1067–1080 (1995)
9. Furia, C.A.: What's decidable about sequences? http://arxiv.org/abs/1001.2100 (January 2010)
10. Ghilardi, S., Nicolini, E., Ranise, S., Zucchelli, D.: Decision procedures for extensions of the theory of arrays. Annals of Mathematics and Artificial Intelligence 50(3-4), 231–254 (2007)
11. Habermehl, P., Iosif, R., Vojnar, T.: A logic of singly indexed arrays. In: Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'08). Lecture Notes in Computer Science, vol. 5330, pp. 558–573. Springer (2008)
12. Habermehl, P., Iosif, R., Vojnar, T.: What else is decidable about integer arrays? In: Proceedings of the 11th International Conference on Foundations of Software Science and Computational Structures (FOSSACS'08). Lecture Notes in Computer Science, vol. 4962, pp. 474–489. Springer (2008)
13. Kuncak, V., Piskac, R., Suter, P., Wies, T.: Building a calculus of data structures. In: Proceedings of the 11th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'10). Lecture Notes in Computer Science, Springer (2010)
14. Makanin, G.S.: The problem of solvability of equations in a free semigroup. Rossiĭskaya Akademiya Nauk. Matematicheskiĭ Sbornik (Translated in Sbornik Mathematics) 103(2), 147–236 (1977)
15. Meyer, B.: Object-oriented software construction. Prentice Hall, 2nd edn. (1997)
16. Piskac, R., Kuncak, V.: Decision procedures for multisets with cardinality constraints. In: Proceedings of the 9th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'08). Lecture Notes in Computer Science, vol. 4905, pp. 218–232. Springer (2008)
17. Plandowski, W.: Satisfiability of word equations with constants is in PSPACE. Journal of the ACM 51(3), 483–496 (2004)
18. Seibert, S.: Quantifier hierarchies over word relations. In: Proceedings of the 5th Workshop on Computer Science Logic (CSL'91). Lecture Notes in Computer Science, vol. 626, pp. 329–352. Springer (1992)
19. Zee, K., Kuncak, V., Rinard, M.C.: Full functional verification of linked data structures. In: Proceedings of the ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation (PLDI'08). pp. 349–361. ACM (2008)