On Array Theory of Bounded Elements*

Min Zhou[1,2,3], Fei He[1,2,3], Bow-Yaw Wang[4], and Ming Gu[1,2,3]

[1] School of Software, Tsinghua University, Beijing 100084, China
[2] Tsinghua National Laboratory for Information Science and Technology
[3] Key Laboratory for Information System Security, MOE, China
[4] INRIA, France, Tsinghua University, China, and Academia Sinica, Taiwan

Abstract. We investigate a first-order array theory of bounded elements. By reducing to weak second-order logic with one successor (WS1S), we show that the proposed array theory is decidable. Finally, two natural extensions to the new theory are shown to be undecidable.

1 Introduction

Consider the following pseudo code for simplified bucket sort:

Input a : `array of` $[0..255]$
`for` $i \leftarrow 0$ `to` 255 `do` $b[i] \leftarrow 0$;
`for` $i \leftarrow 0$ `to` $|a| - 1$ `do` $b[a[i]] \leftarrow b[a[i]] + 1$;
$k \leftarrow 0$;
`for` $i \leftarrow 0$ `to` 255 `do`
  `for` $j \leftarrow 1$ `to` $b[i]$ `do`
    $a[k] \leftarrow i$; $k \leftarrow k + 1$;

In the pseudo code, a is an array of bytes and $|a|$ denotes the size of the array. The array b consists of *buckets*. The v-th bucket records the number of elements in a with value v. Essentially, the pseudo code sorts the input by counting. Since each element of the array a is accessed a constant number of times, the complexity of bucket sort algorithm is linear in the size of the input array.

Suppose we would like to specify properties about the pseudo code. After examining the algorithm, we see that the v-th bucket is positive if and only if there is an index i such that $a[i] = v$. Or, more formally,

$$\forall v \exists i.(b[v] > 0 \leftrightarrow a[i] = v)$$

must hold at the end of the program. This gives a formula in the array logic.

* This work was supported in part by the Chinese National 973 Plan under grant No. 2010CB328003, the NSF of China under grants No. 60635020, 60903030 and 90718039, the National Science Council of Taiwan projects No. NSC97-2221-E-001-003-MY3, NSC97-2221-E-001-006-MY3, the FORMES Project within LIAMA Consortium, and the French ANR project SIVES ANR-08-BLAN-0326-01.

T. Touili, B. Cook, and P. Jackson (Eds.): CAV 2010, LNCS 6174, pp. 570–584, 2010.

Array theory in its most general form is undecidable. Several authors have investigated various decidable fragments of the theory. In [12,2,13], arrays are modeled as uninterpreted functions. Existing decision procedures for uninterpreted functions can decide the satisfiability of formulae in array theory. In [4,1], counter automata are used. Satisfiability of array formulae is reduced to the reachability problem of counter automata. In both approaches, their decidable fragments are very restrictive. Arbitrary quantification induces undecidability, and few of the theory fragments above allow nested reads. Particularly, the sample formula given above does not belong to the decidable fragments of these theories.

We consider the theory of arrays with bounded elements in this paper. In our theory, an array can have an arbitrary but finite size. Its elements, however, must be bounded. Quantification is allowed in our first-order theory. Indices, for instance, can be quantified arbitrarily. We show that the first-order theory of arrays with bounded elements is decidable. With this theory, programmers can specify many program properties which cannot be verified by existing fragments. Particularly, the sample formula given above can be solved.

In our theory, bounded elements reflect data storage format in physical world. Arrays with bounded elements can express most program properties naturally. Due to memory limitation, all data types are actually stored in the physical world with a bounded size. For example, variables of the `int` type are actually stored in most of systems as 32-bit integers. Our array theory of bounded elements is surely a realistic abstraction of the array data type.

Adopting bounded elements moreover relieve us from the imposed restriction of decidable fragments in previous theories. In the array theory of bounded elements, arbitrarily quantified first-order formulae are decidable. Nested reads do not induce undecidability either. Both can be freely used in array formulae without incurring computability issues.

In addition to its generality, another significant advantage of adopting the array theory of bounded elements is its simplicity. It is almost standard to establish the decidability result by reducing it to WS1S. Since decision procedures for WS1S are publicly available (for example, MONA [8]), our reduction immediately gives a decision procedure for the array theory of bounded elements.

We also discuss two natural extensions to the array theory of bounded elements. Either unbounded elements or linear arithmetic on indices would introduce undecidability to our theory. Arrays with unbounded elements subsume previous array theories and hence are undecidable. For arrays of bounded elements, the undecidability result of linear arithmetic on array indices is somewhat surprising. We show that multiplication is expressible with linear arithmetic and arrays of bounded elements. Hence linear arithmetic would make the array theory of bounded elements undecidable.

Related Works. In [13], a quantifier-free extensional multi-dimensional array theory is considered. The author employs a variant of congruence closure algorithm to check the satisfiability. Other quantifier-free array theory fragments are considered in [12] and [14].

In [2], a more expressive logic is presented. The author defined a $\exists^*\forall^*$ fragment of array theory. The index theory is alike to Presburger arithmetic, but addition is only allowed on the existential quantified index variables. The authors present a mechanism to instantiate universal quantifiers by replacing the quantified index variables with each term in the index set. Then arrays are treated as uninterpreted functions. It is shown that allowing nested reads or general Presburger arithmetic over universally quantified index variables (even just $i + 1$) induces undecidability.

An automata based approach is presented in [4]. The authors define an specialized array theory SIL which allows formulas like $\forall i.\phi(i) \rightarrow \gamma(i)$ where i is the only index variable in $\gamma(i)$, and disjunction is forbidden in $\gamma(i)$. The same as in [2], quantifiers can be used only in the form of $\exists^*\forall^*$. An SIL formula is translated into a flatten counter automaton, then the satisfiability is checked by testing if the language of automaton is empty. Furthermore, in [1], they encode pieces of program into counter automata. Although SIL is expressive, some restrictions on the structure of formulas are unnatural. Nested reads are not allowed, neither.

WS1S is a simple but expressive logic. An example where integer arrays are broken to bits and expressed as finite sets in WS1S is known in [7]. We employ and further develop this idea in this paper. Furthermore, solving decidability problem by reducing to WS1S is not uncommon. In [11], a decision procedure for bit-vectors by mapping a bit-vector to a set in WS1S is presented.

This paper is organized as follows: First, we give the preliminaries in Sect. 2. Then in Sect. 3, the array theory UABE is presented. In Sect. 4, we show UABE is decidable by a reduction to WS1S. In Sect. 5 we show that extending UABE with unbounded integer element theory or with addition on index variables incurs undecidability. Examples are given in Sect. 6, followed by conclusion.

2 Preliminaries

2.1 WS1S

WS1S [3] is short for Weak Second-order logic with One Successor. It is a second order logic with the signature $\{=, \in, \mathcal{S}\}$. There are both first- and second-order variables in WS1S. Use $\mathcal{V}_1, \mathcal{V}_2$ to represent them respectively. The syntax is defined as:

$$\phi ::= (p = \mathcal{S}(q)) \mid p \in X \mid \neg\phi \mid \phi \vee \phi \mid \exists p.\phi \mid \exists X.\phi, \qquad p, q \in \mathcal{V}_1, X \in \mathcal{V}_2$$

An interpretation is an assignment on variables. First-order variables are interpreted over $\mathbb{N}$ while second-order variables are interpreted over *finite sets* of $\mathbb{N}$. $\mathcal{S}$ is the successor function on $\mathbb{N}$. $\in$ is the membership relation on $\mathbb{N} \times \mathscr{P}(\mathbb{N})$. Given an interpretation σ, the semantics of WS1S is defined as:

$$\begin{array}{ll}
\sigma \models (p = \mathcal{S}(q)) & \iff \sigma(p) = \sigma(q) + 1 \\
\sigma \models p \in X & \iff \sigma(p) \in \sigma(X) \\
\sigma \models \neg\phi & \iff \sigma \not\models \phi \\
\sigma \models \phi_1 \vee \phi_2 & \iff \sigma \models \phi_1 \text{ or } \sigma \models \phi_2 \\
\sigma \models \exists p.\phi & \iff \sigma[p \leftarrow i] \models \phi, \text{ for some } i \in \mathbb{N} \\
\sigma \models \exists X.\phi & \iff \sigma[X \leftarrow M] \models \phi, \text{ for some } \textit{finite set } M \subseteq \mathbb{N}
\end{array}$$

Based on the syntax declared above, other predicates such as "$<$", "$\subseteq$" can be defined as: $(p < q) \iff \forall X.(q \in X \wedge \forall r.(\mathcal{S}(r) \in X \rightarrow r \in X) \rightarrow p \in X)$ and $(X \subseteq Y) \iff \forall p.(p \in X \rightarrow p \in Y)$.

There is a correspondence between WS1S formulas and regular languages. Based on this fact, some verification tools such as MONA[8,6] are developed. Given a WS1S formula, MONA is able to give a `valid`, `satisfiable` or `unsatisfiable` answer as well as a satisfying example or a counter-example if any.

We mention another decidable logic named S1S. The difference between S1S and WS1S is: in S1S, second order variables are interpreted as *infinite* sets while in WS1S are interpreted as *finite* sets. S1S corresponds to Büchi Automata. Büchi proved in 1960 and 1961 that both WS1S and S1S are decidable[3].

2.2 Terminology

In this paper, the domains of arrays, array indices and array elements are denoted by $\mathbb{A}$, $\mathbb{N}$, and $\mathbb{Z}_n$ respectively, where $\mathbb{A} = \mathbb{Z}_n^*$, $\mathbb{N} = \{0, 1, 2, \ldots\}$ and $\mathbb{Z}_n = \{0, 1, \ldots, 2^n - 1\}$. To simplify the discussion, we assume $\mathbb{Z}_n$ contains only non-negative values. This is not a limitation of our theory, we will show it is intuitive to extend $\mathbb{Z}_n$ to a more general domain.

We use $Var(\lambda)$ to denote the set of variables whose type is λ, where λ can be $\mathbb{N}$, $\mathbb{Z}_n$ or $\mathbb{A}$. We use Var to denote the set of all variables, i.e. $Var = Var(\mathbb{N}) \cup Var(\mathbb{Z}_n) \cup Var(\mathbb{A})$. Throughout the paper, $i, j, k \in Var(\mathbb{N})$; $x, y, z \in Var(\mathbb{Z}_n)$; $a, b \in Var(\mathbb{A})$; c a constant in $\mathbb{N}$; and l a constant in $\mathbb{Z}_n$. Boolean values are expressed as 0 and 1.

Given an array variable a, we use $a[i]$ to denote the *array read* which returns the i-th element of a, and $a\{i \leftarrow x\}$ the *array write* which returns a new array that is obtained by replacing the i-th element in a with x.

3 The Theory of UABE

In our array theory, the index type is not bounded while the element type is bounded, so we call it Unbounded Array with Bounded Element (UABE). It consists of:

- the index theory $T_{\mathbb{N}}$: with domain of $\mathbb{N}$, and signature of $\{\mathcal{S}, <, =\}$;
- the element theory $T_{\mathbb{Z}_n}$: with domain of $\mathbb{Z}_n$, and signature of $\{+_n, <_n, =_n\}$;
- and one-dimensional extensional array theory with the "*read-over-write*" axiom [10]:

$$\begin{array}{rl}
\forall a, i, j, x. & (i = j) \rightarrow a\{i \leftarrow x\}[j] = x \\
& \wedge\ (i \neq j) \rightarrow a\{i \leftarrow x\}[j] = a[j]
\end{array}$$

Table 1. Syntax of UABE

Identifier	Definition	Remarks
$\sim$	$\in \{=, <\}$	comparisons on $\mathbb{N}$
$\sim_n$	$\in \{=_n, <_n\}$	comparisons on $\mathbb{Z}_n$
P	$:= x \sim_n y \mid x \sim_n l \mid x =_n y +_n z \mid x =_n a[i]$	atomic formulas
	$i \sim j \mid i \sim c \mid i = j + c \mid i = \|a\|$	
	$i \simeq x \mid a = b \mid a = b\{i \leftarrow x\}$	
F	$:= P \mid \neg F \mid F_1 \wedge F_2 \mid \exists v.F[v]$	$v \in Var$

Our array theory is *extensional*, that is, for $a, b \in \mathbb{A}$, we have $(a = b) \Leftrightarrow (\forall i.a[i] = b[i])$. Hence the notion of equality is extended to arrays.

3.1 Syntax

The syntax of UABE formulas is shown in Table 1. Note the comparison predicates on $\mathbb{Z}_n$ are different from those on $\mathbb{N}$ by the subscript n. Predicate $\simeq$ is defined on $\mathbb{N} \times \mathbb{Z}_n$, which establishes the equality between $\mathbb{N}$ and $\mathbb{Z}_n$.

In the index theory $T_{\mathbb{N}}$, addition of a variable and a constant, such as $i + 3$, is allowed, while addition of two variables, such as $i + j$, is forbidden. We will show later that addition of index variables leads to undecidability. In the element theory $T_{\mathbb{Z}_n}$, arbitrary arithmetic operators are allowed. Quantifiers can be freely used.

3.2 Semantics

A valuation[1] for UABE is a triple $V = (\mu, \nu, \tau)$, where

- $\mu : Var(\mathbb{Z}_n) \mapsto \mathbb{Z}_n$ assigns each element variable an integer in $\mathbb{Z}_n$,
- $\nu : Var(\mathbb{N}) \mapsto \mathbb{N}$ assigns each index variable a non-negative integer,
- $\tau : Var(\mathbb{A}) \mapsto \mathbb{Z}_n^*$ assigns each array variable a a *finite* sequence of $\mathbb{Z}_n$, i.e. $\tau(a) = a_0, a_1, \ldots, a_{|a|-1}$.

The *size* of array a is denoted by $|a|$. Array index should not exceed the array size. This condition is checked in all array reads. Although the size of an array is considered and checked here, it need not be given a concrete value. In other words, the size of an array is finite but can be arbitrarily large (we say it *unbounded*). For example, the proposition "for all integer m, there is an array a whose length is no less than m and all elements in a are identically 0" can be expressed in UABE as: $\forall m.\exists a.(|a| \geq m \wedge (\forall i < |a|.a[i] = 0))$.

Given a variable v, the value of v under valuation V is denoted as $V(v)$. Given a finite sequence $\tau(a)$, we denote $|\tau(a)|$ the length of it, $\tau(a)_p$ the p-th element of it, and $\tau(a)\{p \leftarrow v\}$ a new sequence by replacing the p-th element of $\tau(a)$ with v.

[1] For UABE we say *valuation* to distinguish from *interpretation* of WS1S.

Table 2. Semantics of UABE

(1)	$V \models (x \sim_n y)$	$\iff \mu(x) \sim \mu(y)$
(2)	$V \models (x \sim_n l)$	$\iff \mu(x) \sim l$
(3)	$V \models (x =_n y +_n z)$	$\iff \mu(x) = \mu(y) + \mu(z)$
(4)	$V \models (x =_n a[i])$	$\iff (\nu(i) < \lvert\tau(a)\rvert) \wedge (\mu(x) = \tau(a)_{\nu(i)})$
(5)	$V \models (i \sim j)$	$\iff \nu(i) \sim \nu(j)$
(6)	$V \models (i \sim c)$	$\iff \nu(i) \sim c$
(7)	$V \models (i = j + c)$	$\iff \nu(i) = \nu(j) + c$
(8)	$V \models (i = \lvert a\rvert)$	$\iff \nu(i) = \lvert\tau(a)\rvert$
(9)	$V \models (i \simeq x)$	$\iff \nu(i) = \mu(x)$
(10)	$V \models (a = b)$	$\iff \tau(a) = \tau(b)$
(11)	$V \models (a = b\{i \leftarrow x\})$	$\iff \tau(a) = \tau(b)\{\nu(i) \leftarrow \mu(x)\}$
(12)	$V \models \exists v.F[v]$	$\iff V \models F[v \leftarrow p], p$ is type consistent with v
(13)	$V \models \neg F$	$\iff V \not\models F$
(14)	$V \models F_1 \wedge F_2$	$\iff (V \models F_1) \wedge (V \models F_2)$

The semantics of UABE is listed in Table 2. Note that $\mathbb{Z}_n \subseteq \mathbb{N}$, hence the semantics of comparisons in (1) and (2) are comparisons between integer values. The semantics of addition of two variables in $\mathbb{Z}_n$ is defined in (3). Note that x, y, z are all n-bit integers, as a sanity condition we need to check the sum of $y +_n z$ does not exceed the range of n-bit integer. Since $\mu(x) < 2^n$ is already satisfied by the definition of μ, so is $\mu(y) + \mu(z) < 2^n$ (provided the equality holds). With rule (9), a variable in $\mathbb{N}$ and a variable in $\mathbb{Z}_n$ can be compared. Hence, the index theory and the element theory are related. As a result, the value of an element can be used as an index. In rule (10), $a = b$ means the sequences of $\tau(a)$ and $\tau(b)$ are identical, which implies $|a| = |b|$. The semantics of existential quantifier is defined in (12). We say p, v are type consistent, if $p \in \mathbb{N}$ and $v \in Var(\mathbb{N})$, or $p \in \mathbb{Z}_n$ and $v \in Var(\mathbb{Z}_n)$, or $p \in \mathbb{A}$ and $v \in Var(\mathbb{A})$.

3.3 Expressive Power

With the syntax in Table 1, many array formulas can be expressed. For example, any Boolean combinations of atomic formulas are expressible since $\{\neg, \wedge\}$ is sufficient to express $\{\vee, \rightarrow, \leftrightarrow\}$; universal quantifier is expressible since $\forall v.F[v] \iff \neg(\exists v.\neg F[v])$; minus function $-$ is expressible since it can be converted to $+$ by transposing the negative terms; comparison operations $\{\leq, >, \geq\}$ on index theory are expressible by using $\{\neg, <, +\}$; comparison operations $\{\leq_n, >_n, \geq_n\}$ on element theory are expressible by using $\{\neg, <_n, +_n\}$. Moreover, consecutive additions on elements, such as $x_1 +_n x_2 +_n x_3$, is also expressible, since one can replace $x_2 +_n x_3$ with a single variable y_2.

Although index and element terms are of different types, nested reads are allowed in UABE. When one want to use an $\mathbb{Z}_n$ term t as an index, one need to declare an existential-quantified variable t' in $Var(\mathbb{N})$, and then assert t and t' are equal. For example, the formula $z =_n a[a[i]]$ can be written in UABE as:

$\exists j.(z =_n a[j] \wedge j \simeq a[i])$. The property $\forall v \exists i.(b[v] > 0 \leftrightarrow a[i] = v)$ specified in the bucket sort algorithm can be expressed in UABE as

$$\forall v \exists i.\,((\exists v'.v' \simeq v \wedge 0 <_n b[v']) \leftrightarrow (v =_n a[i]))\,,$$

where $v' \in Var(\mathbb{N})$ is a fresh variable.

4 Decidability

In this section, we show the satisfiability and validity of formulas in UABE are decidable by giving a reduction to WS1S.

4.1 Representation of $\mathbb{Z}_n$ and Arrays in WS1S

Note a value in $\mathbb{Z}_n$ can be viewed as an n-bit integer. Given a term t of type $\mathbb{Z}_n$, t can be either a constant, a variable or an array read. We represent it as a bit vector $t = \overline{t^{[n-1]}t^{[n-2]}\cdots t^{[0]}}$, where $t^{[d]}$ is the d-th bit of t, $d = 0, 1, \ldots, n-1$, and $t^{[n-1]}$ is the most significant bit. If t is a constant or a variable, we encode the value of t as a single set $S_t = \{i | t^{[i]} = 1\}$ in WS1S. S_t is a subset of $\{0, 1, 2, \ldots, n-1\}$, and $d \in S_t$ iff $t^{[d]} = 1$. If t is an array read $a[i]$, since an array is actually a sequence of $\mathbb{Z}_n$ elements, we encode the array a as n finite sets $a^{(0)}, a^{(1)}, \ldots, a^{(n-1)}$, where $a^{(d)} = \{i | a[i]^{[d]} = 1\}$. Note here we do not encode each element of array as a single set, but the bit values of all elements in the same position as a single set. i.e $a[i]^{[d]} = 1$ iff $i \in a^{(d)}$.

For example, suppose $n = 4$, and all $\mathbb{Z}_n$ integers range over $\{0, 1, \ldots, 15\}$. Then the value $x = 5$ (in binary, $\overline{0101}$) can be encoded as $S_x = \{0, 2\}$, and the array a which is initialized with $a[0] = 1$ and $a[1] = 5$ can be encoded as $a^{(0)} = \{0, 1\}$, $a^{(1)} = \emptyset$, $a^{(2)} = \{1\}$ and $a^{(3)} = \emptyset$.

From above definitions, we can conclude that given any term t of type $\mathbb{Z}_n$, $t^{[d]}$ is a well-formed WS1S formula. Furthermore, the comparison and addition of $\mathbb{Z}_n$ values can be encoded in WS1S based on the bit-wise representations.

Lemma 1. *Equality relation $P_=(s,t)$ on $\mathbb{Z}_n$ can be encoded in WS1S.*

Proof. $P_=(s,t) \equiv \bigwedge_{d=0}^{n-1} \left(s^{[d]} \leftrightarrow t^{[d]}\right)$ □

Lemma 2. *Order relation $P_<(s,t)$ on $\mathbb{Z}_n$ can be encoded in WS1S.*

Proof. $P_<(s,t) \equiv \bigvee_{d=0}^{n-1} \left((\neg s^{[d]} \wedge t^{[d]}) \wedge \left(\bigwedge_{d'=d+1}^{n-1} (s^{[d']} \leftrightarrow t^{[d']})\right)\right)$ □

Lemma 3. *Addition on $\mathbb{Z}_n$ can be encoded in WS1S.*

Proof. Addition can be encoded as:

$$\begin{aligned} P_+(s,t,u) \equiv \exists C.(\ & C \subseteq \{1, \ldots, n-1\} \\ & \textstyle\bigwedge_{d=0}^{n-1}((d+1) \in C \leftrightarrow (d \in C \wedge t^{[d]} \vee d \in C \wedge u^{[d]} \vee t^{[d]} \wedge u^{[d]})) \\ & \textstyle\bigwedge_{d=0}^{n-1}(s^{[d]} \leftrightarrow \neg((d \in C) \leftrightarrow \neg(t^{[d]} \leftrightarrow u^{[d]})))) \end{aligned}$$

In the definition of $P_+(s,t,u)$, C is the add-carry. The first line restricts the sum not to exceed 2^n-1. The second line constraints $C^{[d+1]}=1$ whenever at least two of $t^{[d]}, u^{[d]}, C^{[d]}$ is 1. The third line constraints $s^{[d]} = (t^{[d]} \text{ xor } u^{[d]} \text{ xor } C^{[d]})$. It is guaranteed that $P_+(s,t,u) \iff s =_n t +_n u$. □

The domain of element theory can be generalized. If negative numbers are allowed in $\mathbb{Z}_n$, then a sign bit is needed. $P_=$, $P_<$ and P_+ can also be defined, although in a more complex form. Actually, a much richer set of operators can be supported. We can even replace the element domain $\mathbb{Z}_n$ with fixed size bit vectors. Bit-vector operations such as multiplication, division, modulo, etc, can be supported since they can also be encoded in WS1S.

4.2 Translation

A translation rule is denoted by a horizontal line which separates the original and translated formula. Above the line is the original formula in UABE, while the translated formula in WS1S is shown below the line.

In the translation, each index variable i in UABE is translated to a variable i in WS1S; each element variable x is translated to a set S_x and each array variable a corresponds to a variable $|a|$ and n sets: $a^{(0)}, \ldots, a^{(n-1)}$.

A special array $A_\simeq$ is used in the translation to establish the connection between $\mathbb{N}$ and $\mathbb{Z}_n$. We require $A_\simeq$ to be $(0,1,\ldots,2^n-1)$. Notice that $\mathbb{Z}_n \subseteq \mathbb{N}$. $A_\simeq$ is defined such that for indices $i < 2^n$, i and $A_\simeq[i]$ are equal in value. It is not a good idea to encode $A_\simeq$ one by one in WS1S. By observation, $A_\simeq^{(d)}$ is a set which does not contain $\{0,\ldots,2^d-1\}$ but contains $\{2^d,\ldots,2^{d+1}-1\}$. Furthermore, for numbers $i \geq 2^{d+1}$, $i \in A_\simeq^{(d)} \iff (i-2^{d+1}) \in A_\simeq^{(d)}$. For example, $A_\simeq^{(0)}$ contains all the 0-th (lowest) bit of array $(0,1,2,\ldots,2^n-1)$, i.e. $A_\simeq^{(0)} = \{1,3,5,7,\ldots\}$. Thus $A^{(d)}$ is defined directly in WS1S:

$$\begin{aligned} &\forall i.\left((i<2^d \rightarrow i \notin A_\simeq^{(d)}) \wedge (2^d \leq i \wedge i < 2^{d+1} \rightarrow i \in A_\simeq^{(d)})\right) \\ \bigwedge\ &\forall i.\left(i+2^{(d+1)} < 2^n \rightarrow ((i \in A_\simeq^{(d)}) \leftrightarrow (i+2^{(d+1)} \in A_\simeq^{(d)}))\right) \\ \bigwedge\ &\forall i.\left(i \geq 2^n \rightarrow i \notin A_\simeq^{(d)}\right) \end{aligned}$$

Use Θ to denote the conjunction of $|A_\simeq| = 2^n$ and all the definition of $A^{(d)}$ $(d = 0,1,\ldots,n-1)$. Thus, Θ is a WS1S formula which encodes $A_\simeq$. $A_\simeq$ is shared and reused during the translation.

Lemma 4. *Formula Θ is satisfiable.*

Proof. $A_\simeq = (0,1,\ldots,2^n-1)$, an interpretation which satisfies Θ can be obtained by encoding $A_\simeq$ into WS1S. □

Given a UABE formula, one can exhaustively apply the following translation rules until it is rewritten into WS1S. We will focus on the translation of atomic formulas, since both of UABE and WS1S do not have any restriction on quantifiers.

Table 3. Translation Rules for Atomic Formulas

$$(1.1)\frac{x =_n y}{P_=(x,y)} \quad (1.2)\frac{x <_n y}{P_<(x,y)} \quad (2.1)\frac{x =_n l}{P_=(x,l)} \quad (2.2)\frac{x <_n l}{P_<(x,l)} \quad (3)\frac{x =_n y +_n z}{P_+(x,y,z)}$$

$$(4)\frac{x =_n a[i]}{(i < |a|) \wedge P_=(x, a[i])} \quad (5)\frac{i \sim j}{i \sim j} \quad (6)\frac{i \sim c}{i \sim c} \quad (7)\frac{i = j + c}{i = j + c} \quad (8)\frac{i = |a|}{i = |a|}$$

$$(9)\frac{i \simeq x}{i < 2^n \wedge P_=(A_\simeq[i], x)} \quad (10)\frac{a = b}{(|a| = |b|) \wedge (\forall i.(i < |a|) \rightarrow P_=(a[i], b[i]))}$$

$$(11)\frac{a = b\{i \leftarrow x\}}{(i < |a|) \wedge P_=(a[i], x) \wedge |a| = |b| \wedge (\forall j.(j \neq i \wedge j < |a|) \rightarrow P_=(a[j], b[j]))}$$

Translation rules for atomic formulas are listed in Table 3. Each rule is tagged with a number. Rule $(x.y)$ in this table corresponds to item (x) in Table 2. Among these rules, (1.1), (1.2), (2.1), (2.2), (3), (4) are comparisons or addition on $\mathbb{Z}_n$, defined with the help of predefined shortcuts $P_=$, $P_<$ and P_+. In (4), we added the sanity condition for array read, $i < |a|$. (5), (6), (7), (8) are translated without any modification since they are already well-formed in WS1S. For (9), to translate atomic formula $i \simeq x$, the difficulty is type inconsistency between i and x: $i \in Var(\mathbb{N})$ but $x \in Var(\mathbb{Z}_n)$. With the help of $A_\simeq$, our method is replacing the original formula with $x =_n A_\simeq[i]$. In (10), $a = b$ implies the length of a, b are identical. In (11), array a and b are identical except at index i.

To handle quantifiers and Boolean combination of formulas, we add more rules, shown in Table 4. Denote $\Delta(\cdot)$ the translation procedure on atomic formulas which is defined in Table 3. In Table 4, (12.1), (12.2) and (12.3) handles quantifiers and (13), (14) handles Boolean combination of formulas. Boundary constraints Γ_1, Γ_2 are used. $\Gamma_1(x) \equiv S_x \subseteq \{0, 1, \ldots, n-1\}$, in (12.1) it is required because x should has at most n-bits. $\Gamma_2(a) \equiv \bigwedge_{d=0}^{n-1}(a^{(d)} \subseteq \{0, 1, \ldots, |a| - 1\})$, it is used in (12.3) to restrict the array size to $|a|$.

Table 4. Translation Rule for Complex Formulas

$$(12.1)\frac{\exists x.F}{\exists S_x.\Gamma_1(x) \wedge \Delta(F)} \quad (12.2)\frac{\exists i.F}{\exists i.\Delta(F)} \quad (12.3)\frac{\exists a.F}{\exists\,(|a|, a^{(0)}, \ldots, a^{(n-1)})\,.\Gamma_2(a) \wedge \Delta(F)}$$

$$(13)\frac{\neg F}{\neg\Delta(F)} \qquad (14)\frac{F_1 \wedge F_2}{\Delta(F_1) \wedge \Delta(F_2)}$$

Following the procedure above, these properties hold.

Theorem 1. *Given any UABE formula f and any valuation V, there is a corresponding interpretation σ_V for $\Delta(f)$, such that $\sigma_V \models \Theta$ and $V \models f \iff \sigma_V \models \Delta(f)$.*

Theorem 2. *Given any UABE formula f and any interpretation σ_V for $\Delta(f)$, if $\sigma_V \models \Theta$, then there is a valuation V for f such that $V \models f \iff \sigma_V \models \Delta(f)$.*

Theorem 1 and 2 actually show the fact that for each UABE formula f, there is a bijection between all valuations for f and all interpretations for $\Delta(f)$ that satisfies Θ. We give the idea how the connection between V and σ_V is established:

- $\forall i \in Var(\mathbb{N}).\ \sigma_V(i) = V(i)$,
- $\forall x \in Var(\mathbb{Z}_n).\forall d.(d \in \sigma_V(S_x) \iff V(x)^{[d]} = 1)$,
- $\forall a \in Var(\mathbb{A}).\ \sigma_V(|a|) = |V(a)|$,
- $\forall 0 \leq d < n.\forall a \in Var(\mathbb{A}).\forall i.(i \in \sigma_V(a^{(d)}) \iff V(a)^{[d]}_{V(i)} = 1$.

If $\sigma_V \models \Theta$, the above formulas already informally defined the bijection.

Theorem 3. *Given any UABE formula f, f is satisfiable if and only if $\Theta \wedge \Delta(f)$ is, f is valid if and only if $\Theta \rightarrow \Delta(f)$ is.*

Theorem 4. *The satisfiability for UABE is decidable.*

Proof. Our theory of UABE is decidable because WS1S is. □

Compared with other fragments of array theory, UABE has following characteristics: quantifiers can be freely used; nested reads of array are allowed; and there is no restriction on the structure of formula. Many array properties can be expressed in UABE, such as:

- Sorted: $\forall i, j.\ ((i < j < |a|) \rightarrow (a[i] \leq_n a[j]))$;
- Periodical: $\forall i.\ (i + T < |a| \rightarrow a[i] =_n a[i + T])$;
- Partitioned: elements with indices less than p are no larger than those with indices greater or equal to p:
  $\forall i, j.\ (i < p \leq j < |a| \rightarrow a[i] \leq_n a[j])$;
- Fibonacci array:
  $(a[0] =_n 1) \wedge (a[1] =_n 1) \wedge \forall i.\ (i + 2 < |a| \rightarrow a[i + 2] =_n a[i] +_n a[i + 1])$;
- an array monotonically increasing by 1 each step:
  $\forall i.(i < |a| \rightarrow a[i + 1] =_n a[i] +_n 1)$;

4.3 Infinite Arrays with Bounded Elements

In Sect. 3.2, the semantics of an array is referred as a *finite* sequence. We note that the semantics of an array can be extended to *infinite* sequence by a similar reduction to S1S without sacrificing decidability.

In the new theory, the semantics of arrays is changed to *infinite* sequence over $\mathbb{Z}_n$. $|a|$ is no longer needed because the size of an array can be infinite. In the translation, $a = b$ need not imply $|a| = |b|$ and the boundary constraint Γ_2 is no longer required. Then everything can work through. Nevertheless, in practice, if the target model is S1S, both finite and infinite interpretation of array can be mixed.

5 Extensions

In this section, we show that either extending the element theory to unbounded domain or allow addition of variables in the index theory such as $k = i + j$ will cause undecidability. In both cases, we show that the Hilbert's tenth problem [9] is reducible to the extended version of UABE. What we need to do is to define a formula $\varphi_+(k, i, j)$ which is satisfiable if and only if $k = i + j$, and a formula $\varphi_\times(k, i, j)$ which is satisfiable if and only if $k = i \times j$.

5.1 Extend Array Element to $\mathbb{N}$

We call this extended theory UAUE (Unbounded Array with Unbounded Elements). In this case, such $\varphi_+(k, i, j)$ and $\varphi_\times(k, i, j)$ can be defined in the same way as [4]. In particular, the $\forall^*\exists^*\forall^*\exists^*$-**SIL** logic fragment is then contained in UAUE. Undecidability of UAUE follows from [4].

5.2 Allow Addition on $\mathbb{N}$ Variables

We call this extended theory UABE$^+$. If we extend the theory of index to allow additions, $\varphi_+(k, i, j)$ can be defined as $k = i + j$. It remains to define $\varphi_\times(k, i, j)$. At first, assume i, j are both no less than 2.

The proof in [4] can not be applied because it relies on the fact that elements of an array can be arbitrarily large while the element theory is bounded in UABE$^+$. However, in UABE$^+$, we can assume the domain of array element contains at least two values, say $\{0, 1\}$. Then an array a can be treated as a finite set of integers which consists of all the indices where the corresponding array element is 1. Use $\hat{a}$ to denote the set represented by the array a. i.e: $\hat{a} = \{i \in \mathbb{N} | a[i] = 1\}$.

In this section, we also use:

- $[x, y]$ to denote the Least Common Multiple of x and y;
- $\langle x \rangle$ to denote the set $\{\alpha \cdot x | \alpha \in \mathbb{N}\}$, i.e, all multiple of x;
- $\langle x, y \rangle$ to denote the set $\{\alpha \cdot [x, y] | \alpha \in \mathbb{N}\}$, i.e, all multiple of $[x, y]$;

One can check immediately: $\forall x, y. \langle x, y \rangle = \langle [x, y] \rangle = \langle x \rangle \cap \langle y \rangle$

A finite set P is a *prefix* of some set Q (finite or infinite) if there is an integer r such that $P = \{0, 1, 2, \ldots, r\} \cap Q$. denoted by $P \sqsubset Q$. If P is a set, c is an integer, then $P + c = \{\alpha + c | \alpha \in P\}$ is the set obtained by adding c to each element in P.

The idea beyond the construction is expressed as a theorem:

Lemma 5. *Use $\Pi(i, j)$ to denote the set $= \langle i, j \rangle \cap (\langle i - 1, j - 1 \rangle + i + j - 1)$, if i, j are two positive integers and both are no less than 2, then the smallest number in $\Pi(i, j)$ is $i \times j$.*

Proof. Firstly, $ij - i - j + 1 = (i - 1)(j - 1)$, so $ij - i - j + 1 \in \langle i - 1, j - 1 \rangle$, Thus $ij \in (\langle i - 1, j - 1 \rangle + i + j - 1)$. Moreover, $ij \in \langle i, j \rangle$. Therefore $ij \in \Pi(i, j)$.

By contradiction, suppose there is an integer $p \in \Pi(i,j)$ and $p < ij$. Then $p \in \langle i,j\rangle$ and $p-i-j+1 \in \langle i-1, j-1\rangle$ both hold, which implies $i|p$, $j|p$, $(i-1)|(p-i-j+1)$ and $(j-1)|(p-i-j+1)$.

Suppose $[i,j] = ij/b$ where b is the Greatest Common Divisor of i and j. then $p = t[i,j] = t \cdot (ij/b)$ where $1 \le t < b$. Since $(i-1)|(p-i-j+1)$, we know:

$$\begin{aligned}
&(i-1)|(p-j) \Longleftrightarrow (i-1)|\left(\tfrac{tij}{b} - j\right)\\
\Longleftrightarrow\ &(i-1)|\left(((i-1)+1)\tfrac{tj}{b} - j\right) \Longleftrightarrow (i-1)|\left((i-1)\tfrac{tj}{b} + \tfrac{tj}{b} - j\right)\\
\Longleftrightarrow\ &(i-1)|\left(\tfrac{tj}{b} - j\right) \Longleftrightarrow (i-1)|(t-b)j/b
\end{aligned}$$

Symmetrically, $(j-1)|(t-b)i/b$. Hence $(i-1)(j-1)|(b-t)^2 \times j/b \times i/b$.

We assert that $(i-1) > (b-t) \times i/b$, it suffices to show $ti > b$: since $p \in \Pi(i,j)$, it is a necessary condition that $p \ge i+j-1$. i.e: $(tij/b \ge i+j-1) \Longleftrightarrow (tij/b > j) \Longleftrightarrow (ti > b)$. Symmetrically, we know $(j-1) > (b-t) \times j/b$. Hence $(i-1)(j-1) > (b-t)^2 \times j/b \times i/b$ holds. Combined with the fact $(i-1)(j-1)|(b-t)^2 \times j/b \times i/b$, we know $(b-t)^2 \times j/b \times i/b = 0$ must hold. But this is impossible because $t < b$ and $i, j \ge 2$. A contradiction. □

Following the idea presented in the previous lemma, we are willing to construct such $\varphi_\times(k,i,j)$. But the problem is, sets such as $\langle i,j\rangle$ are infinite. However, arrays should be finite in UABE$^+$. Good news is the finite prefixes of these infinite sets are sufficient to define $\varphi_\times(k,i,j)$, Because we can use a variable γ to bound the size of arrays, then quantify γ with an existential quantifier. For sufficiently large γ (larger than $i \times j$), everything works through. Detailed steps are:

- construct a set $\hat{M}_i \sqsubset \langle i\rangle$, by:

$$(M_i[0] =_n 1) \wedge (\forall p.(0 < p < i) \rightarrow M_i[p] =_n 0) \wedge (\forall p.p < \gamma \rightarrow M_i[p] =_n M_i[p+i])$$

- construct a set $\hat{M}_j \sqsubset \langle j\rangle$ in the same way;
- construct a set $\hat{M}_{i,j} \sqsubset \hat{M}_i \cap \hat{M}_j$, i.e: $\hat{M}_{i,j} \sqsubset \langle i,j\rangle$ by:

$$\forall p < \gamma.((M_{i,j}[p] =_n 1) \leftrightarrow (M_i[p] =_n 1 \wedge M_j[p] =_n 1))$$

- construct $\hat{M}_{i-1,j-1}$ in the same way;
- shift every element in $\hat{M}_{i-1,j-1}$ by $i+j-1$, result in $\hat{M}'_{i-1,j-1}$:

$$\begin{aligned}
&\forall p.((p < i+j-1) \rightarrow M'_{i-1,j-1}[p] =_n 0)\\
\wedge\ &\forall p < \gamma.M_{i-1,j-1}[p] =_n M'_{i-1,j-1}[p+i+j-1]
\end{aligned}$$

- find the minimal number in the intersection of $M_{i,j}$ and $M'_{i-1,j-1}$, say it k;

$$(M_{i,j}[k] = 1 \wedge M'_{i-1,j-1}[k] =_n 1) \wedge (\forall p.(M_{i,j}[p] =_n 1 \wedge M'_{i-1,j-1}[p] = 1) \rightarrow k \le p)$$

Let $\psi(k,i,j)$ be the conjunction of all the formulas where γ, M_i, M_j, $M_{i,j}$, M_{i-1}, M_{j-1}, $M_{i-1,j-1}$, $M'_{i-1,j-1}$ are existentially quantified. Because when γ is sufficiently large, such k will exists in the intersection of $M_{i,j}$ and $M'_{i-1,j-1}$. Then $\psi(k,i,j)$ is satisfiable if and only if $k = i \times j$.

When one of i, j is less than 2, trivial. Hence $\varphi_\times(k, i, j)$ can be defined as:

$$\begin{aligned}&((i = 0 \vee j = 0) \rightarrow k = 0)\\ \wedge\ &(i = 1 \rightarrow k = j) \wedge (j = 1 \rightarrow k = i)\\ \wedge\ &(i \geq 2 \wedge j \geq 2 \rightarrow \psi(k, i, j))\end{aligned}$$

Theorem 5. *The satisfiability for UABE$^+$ is undecidable.*

We mention the fact that "Presburger Arithmetic with Predicate is undecidable" is already proved in [5]. However it can not be used directly because array is finite, and finite arrays can not be used to represent predicates of Presburger arithmetic.

6 Example

As an illustrating example, we assume integers are 4-bit integers. Consider the formula F:

$$\exists a.(|a| = 10 \wedge \forall i.(i < |a| \rightarrow a[i + 1] =_n a[i] +_n 1))$$

This formula states: there exists an array a which increases by 1 each step, and the size of a is 10. Following the steps in Section 3.1, F is rewritten to UABE syntax, say F':

$$\begin{aligned}&\exists a.(|a| = 10 \wedge \neg\exists i.((i < |a|)\wedge\\ &\neg(\exists x, y, z, j.j = i + 1 \wedge z =_n 1 \wedge y =_n a[i] \wedge x =_n a[j] \wedge x =_n y +_n z)))\end{aligned}$$

Then F' is translated into $\Delta(F')$:

$$\begin{aligned}&\exists |a|, a^{(0)}, a^{(1)}, a^{(2)}, a^{(3)}.\Gamma_2(a) \wedge (P_=(|a|, 10) \wedge \neg\exists i.(P_<(i < |a|)\wedge\\ &\neg(\exists S_x, S_y, S_z, j.j = i + 1 \wedge P_=(z, 1) \wedge P_=(y, a[i]) \wedge P_=(x, a[j]) \wedge P_+(x, y, z))))\end{aligned}$$

MONA is used to check $\Theta \wedge \Delta(F')$. In a second, the result shows "satisfiable", and an example is given: $a = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)$.

Many other formulas can be checked, such as:

- Fibonacci array is sorted: Because Fibonacci array increases exponentially. For a 4-bit element domain, we have to restrict $|a| < 7$ so that the value does not exceed $2^4 - 1 = 15$. The formula is given as:

$$\begin{aligned}&(|a| < 7) \wedge (a[0] =_n 1) \wedge (a[1] =_n 1)\\ &\wedge\forall i.i + 2 < |a| \rightarrow (a[i + 2] =_n a[i + 1] +_n a[i])\\ \rightarrow\ &(\forall i, j.i < j < |a| \rightarrow a[i] \leq_n a[j])\end{aligned}$$

  In a second, the result shows "valid", i.e. all Fibonacci array with length less than 7 is sorted.

– all values stored in array a are distinct:

$$\forall i, j.(i < |a| \wedge j < |a| \wedge (a[i] =_n a[j]) \rightarrow (j = i))$$

Check this formula, result is "satisfiable", and an example where all elements are distinct is given.

– other properties such as: if $a[i] < i$ for all indices then $a[a[i]] < i$ for all indices can be expressed as:

$$\forall i.(i < |a| \rightarrow a[i] < i) \rightarrow \forall i.(i < |a| \wedge a[i] < |a| \rightarrow a[a[i]] < i)$$

This formula is "valid".

As we can see, UABE has good expressive power. Many formulas that are not expressible in other logic fragments can be expressed and checked.

7 Conclusion

In this paper, we investigated a new array theory UABE. It is quite expressive but decidable. A decision procedure is given by translating UABE into WS1S. Two extensions of UABE are shown to be undecidable. Some examples of UABE formulas and their verification results are given. In the future, we want to implement the translation and do more experiments. We also want to see how it performs in program verification.

References

1. Bozga, M., Habermehl, P., Iosif, R., Konečný, F., Vojnar, T.: Automatic verification of integer array programs. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 157–172. Springer, Heidelberg (2009)
2. Bradley, A.R., Manna, Z., Sipma, H.B.: What's decidable about arrays. In: Emerson, E.A., Namjoshi, K.S. (eds.) VMCAI 2006. LNCS, vol. 3855, pp. 427–442. Springer, Heidelberg (2005)
3. Büchi, J.: Weak second-order arithmetic and finite automata (1959)
4. Habermehl, P., Iosif, R., Vojnar, T.: A logic of singly indexed arrays. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR 2008. LNCS (LNAI), vol. 5330, pp. 558–573. Springer, Heidelberg (2008)
5. Halpern, J.Y.: Presburger arithmetic with unary predicates is Π_1^1 complete. Journal of Symbolic Logic 56, 56–62 (1991)
6. Henriksen, J.G., Jensen, O.J., Jørgensen, M.E., Klarlund, N., Paige, R., Rauhe, T., Sandholm, A.B.: Mona: Monadic second-order logic in practice. In: Brinksma, E., Steffen, B., Cleaveland, W.R., Larsen, K.G., Margaria, T. (eds.) TACAS 1995. LNCS, vol. 1019, pp. 89–110. Springer, Heidelberg (1995)
7. Klarlund, N.: Mona & Fido: The logic-automaton connection in practice. In: Nielsen, M., Thomas, W. (eds.) CSL 1997. LNCS, vol. 1414, pp. 311–326. Springer, Heidelberg (1998)
8. Klarlund, N., Møller, A.: MONA Version 1.4 User Manual. BRICS, Department of Computer Science, Aarhus University, Notes Series NS-01-1 (2001), `http://www.brics.dk/mona/` (revision of BRICS NS-98-3)

9. Matiyasevich, Y.: Enumerable sets are diophantine. Journal of Sovietic Mathematics, 354–358 (1970)
10. Mccarthy, J.: Towards a mathematical science of computation. In: IFIP Congress, pp. 21–28. North-Holland, Amsterdam (1962)
11. Möller, M., Rueß, H.: Solving bit-vector equations. In: Gopalakrishnan, G.C., Windley, P. (eds.) FMCAD 1998. LNCS, vol. 1522, p. 524. Springer, Heidelberg (1998)
12. Nelson, C.G.: Techniques for program verification. PhD thesis, Stanford University, Stanford, CA, USA (1980)
13. Stump, A., Barrett, C.W., Dill, D.L., Levitt, J.: A decision procedure for an extensional theory of arrays. In: LICS '01, Washington, DC, USA, pp. 29–37. IEEE Computer Society Press, Los Alamitos (2001)
14. Suzuki, N., Jefferson, D.: Verification decidability of presburger array programs. JACM 27(1), 191–205 (1980)