

What's Decidable About Arrays - An Abstract

Philip Busch

14. Juli 2008

Reference:

Aaron R. Bradley, Zohar Manna and Henny B. Sipma,
What's Decidable About Arrays?
In "Verification, Model Checking, and Abstract Interpretation,
7th International Conference", VMCAI 2006,
eds: E. Allen Emerson and Kedar S. Namjoshi,
Lecture Notes in Computer Science 3855,
pages 427–442,
Springer Verlag, 2006.

Motivation

For the full theory of arrays, satisfiability is undecidable (FOL), in contrast to satisfiability of a quantifier-free fragment. Motivated by applications to program verification, the authors study a decision procedure for satisfiability in an expressive fragment of a theory of arrays, which is parameterized by the theories of the array elements. The decision procedure reduces satisfiability of a formula of the fragment to satisfiability of an equisatisfiable quantifier-free formula in the combined theory of equality with uninterpreted functions (EUF), Presburger arithmetic, and the element theories. This fragment allows a constrained use of universal quantification, so that one quantifier alternation is allowed, with some syntactic restrictions. A decision procedure for a theory of arrays is of interest for applications in formal verification, program analysis and automated theorem proving.

Definitions

Definition 1 (Theories) *The theory of arrays uses Presburger arithmetic for array indices and the parameter element theories $T_{elem}^1, \dots, T_{elem}^m$, for $m > 0$, for its elements. Assume each T_{elem}^k has signature Σ_{elem}^k . T_A then has signature*

$$\Sigma_A = \Sigma_{\mathbb{Z}} \cup \bigcup_k \Sigma_{elem}^k \cup \{read, write\}$$

Definition 2 (Array Property) *An array property is a formula of the form*

$$(\forall \bar{i}) (\phi_I(\bar{i}) \rightarrow \phi_V(\bar{i}))$$

where \bar{i} is a vector of index variables, and $\phi_I(\bar{i})$ and $\phi_V(\bar{i})$ are the index guard and the value constraint, respectively.

The form of an index guard $\phi_I(\bar{i})$ is constrained according to the following grammar

$$\begin{aligned}
\text{iguard} &\rightarrow \text{iguard} \wedge \text{iguard} \mid \text{iguard} \vee \text{iguard} \mid \text{atom} \\
\text{atom} &\rightarrow \text{expr} \leq \text{expr} \mid \text{expr} = \text{expr} \\
\text{expr} &\rightarrow \text{uvar} \mid \text{pexpr} \\
\text{pexpr} &\rightarrow \mathbb{Z} \mid \mathbb{Z} \cdot \text{evar} \mid \text{pexpr} + \text{pexpr}
\end{aligned}$$

where *uvar* is any universally quantified variable, and *evar* is any existentially quantified integer variable.

The form of a value constraint $\phi_V(\bar{i})$ is also constrained. Any occurrence of a quantified index variable $i \in \bar{i}$ in $\phi_V(\bar{i})$ must be as a read into an array, $a[i]$, for array term a . Array reads may not be nested; e.g. $a_1[a_2[i]]$ is not allowed.

The following table lists some examples of array properties.

<i>Equality</i>	$a = b \Leftrightarrow (\forall i) (a[i] = b[i])$
<i>Bounded Equality</i>	$\text{beq}(l, u, a, b) \Leftrightarrow (\forall i) (l \leq i \leq u \rightarrow a[i] = b[i])$
<i>Sorted</i>	$\text{sorted}(l, u, a) \Leftrightarrow (\forall i, j) (l \leq i \leq j \leq u \rightarrow a[i] \leq a[j])$

Definition 3 (Array Property Fragment) *The array property fragment of T_A consists of all existentially-closed Boolean combinations of array property formulae and quantifier-free T_A -formulae.*

Definition 4 (Read Set) *The read set for a formula ϕ is the set*

$$\mathcal{R} := \{t \mid a[t] \in \phi \wedge t \text{ not univ. quantified} \}$$

for t representing index terms that are not universally quantified index variables.

Definition 5 (Bounds Set) *The bounds set \mathcal{B} for a formula ϕ is the set of Presburger arithmetic terms that arise as root pexpr during the parsing of all index guards in ϕ .*

Definition 6 (Index Set) *The index set is the union of the reads set and the bounds set, or $\{0\}$ if both are empty.*

$$\mathcal{I}_\phi = \begin{cases} \{0\} & \text{if } \mathcal{R} = \mathcal{B} = \emptyset \\ \mathcal{R} \cup \mathcal{B} & \text{otherwise} \end{cases}$$

The read set \mathcal{R} is the set of index terms at which some array is read, while the bounds set \mathcal{B} is the set of index terms that define boundaries on some array for an array property.

Decision Procedure SAT_A

1. Insert predicate definitions and convert to negation normal form.
2. Apply the following rule exhaustively to remove writes:

$$\frac{\psi[a \{i \leftarrow e\}]}{\psi[b] \wedge b[i] = e \wedge (\forall j)(j \neq i \rightarrow a[j] = b[j])} \text{ for fresh } b \text{ (write)}$$

To meet the syntactic requirements on an index guard, we rewrite the third conjunct as

$$(\forall j)(j \leq i - 1 \vee i + 1 \leq j \rightarrow a[j] = b[j])$$

3. Apply the following rule exhaustively:

$$\frac{\psi[(\exists \bar{i})(\phi_I(\bar{i}) \wedge \neg \phi_V(\bar{i}))]}{\psi[\phi_I(\bar{j}) \wedge \neg \phi_V(\bar{j})]} \text{ for fresh } \bar{j} \text{ (exists)}$$

4. Apply the following rule exhaustively, where $\mathcal{I}_{\psi_3}^n$ is determined by the formula constructed in Step 3.

$$\frac{\psi[(\forall \bar{i})(\phi_I(\bar{i}) \rightarrow \phi_V(\bar{i}))]}{\psi \left[\bigwedge_{\bar{i} \in \mathcal{I}_{\psi_3}^n} (\phi_I(\bar{i}) \rightarrow \phi_V(\bar{i})) \right]} \text{ (forall)}$$

5. Associate with each n -dimensional array variable a a fresh n -ary uninterpreted function f_a , and replace each array read $a[i, \dots, j]$ by $f_a(i, \dots, j)$, or shorter, replace array reads by uninterpreted function symbols. Decide this formula's satisfiability using a procedure for quantifier-free formulae of $T_{EUF} \cup T_{\mathbb{Z}} \cup \bigcup_k T_{elem}^k$.

Let's have a closer look at the decision procedure. The first step is trivial, replacing occurrences of predicates with their definitions and converting the formula to negation normal form, i.e. negation occurs only immediately above elementary propositions, and $\{\neg, \vee, \wedge\}$ are the only allowed Boolean connectives.

The second step removes array writes $a \{i \leftarrow e\}$ by replacing each array write with a new array constant b with exactly the same values as a , except that $b[i]$ shall equal e . In layman's terms, the second step of SAT_A replaces an array write with a new array where the write has already happened.

The third step removes existential quantifiers by Skolemization which may have resulted from converting to negation normal form in the first step.

The fourth step is the interesting part; SAT_A rewrites universally quantified subterms to finite conjunctions by instantiating the quantified variables over the set of index terms. Note that by definition of index guards which only allow any of $\{=, \leq\}$ as comparison operators, the index set $\mathcal{I}_{\psi_3}^n$ not only contains indices where writes occur, but also the indices to surrounding array elements.

The fifth step then merely replaces array reads by uninterpreted function symbols.

Example

Consider the array property formula

$$w < x < y < z \wedge 0 < k < l < n \wedge l - k > 1 \\ \wedge \text{sorted}(0, n - 1, a \{k \leftarrow w\} \{l \leftarrow x\}) \wedge \text{sorted}(0, n - 1, a \{k \leftarrow y\} \{l \leftarrow z\})$$

(which is existentially closed). The first step of SAT_A replaces the *sorted* literals with definitions; the second applies *write* to remove array writes. For readability, index guards resulting from *write* use disequalities:

$$w < x < y < z \wedge 0 < k < l < n \wedge l - k > 1 \\ \wedge (\forall i, j)(0 \leq i \leq j \leq n - 1 \rightarrow c[i] \leq c[j]) \\ \wedge (\forall i, j)(0 \leq i \leq j \leq n - 1 \rightarrow e[i] \leq e[j]) \\ \wedge (\forall i)(i \neq l \rightarrow b[i] = c[i]) \wedge c[l] = x \\ \wedge (\forall i)(i \neq k \rightarrow a[i] = b[i]) \wedge b[k] = w \\ \wedge (\forall i)(i \neq l \rightarrow d[i] = e[i]) \wedge e[l] = z \\ \wedge (\forall i)(i \neq k \rightarrow a[i] = d[i]) \wedge d[k] = y$$

Then $\mathcal{R} = \{k, l\}$, $\mathcal{B} = \{0, n - 1, l - 1, l + 1, k - 1, k + 1\}$, and $\mathcal{I}_\phi = \{0, n - 1, k - 1, k, k + 1, l - 1, l, l + 1\}$. Note that \mathcal{R} and \mathcal{B} do not include i or j , which

are universally quantified, while \mathcal{B} contains the terms produced by converting disequalities to disjunctions of inequalities. Applying *forall* to each array property subformula converts universal quantification to finite conjunction over \mathcal{I}_ϕ . We have in particular that

$$c[k + 1] \leq c[l] = x < y = d[k] \leq d[k + 1],$$

yet $c[k + 1] = b[k + 1] = a[k + 1] = d[k + 1]$ as $k + 1$ doesn't occur as an index in array writes: the array a is modified only at indices k and l with $l - k > 1$. Thus we derived a contradiction, hence the original formula is unsatisfiable. Note that the index term $k + 1$ is essential for this proof, although it doesn't occur in the original formula.

Related Topics

Several extensions to the array property fragment result in a fragment of T_A^Z for which satisfiability is undecidable, such as nested reads (e.g., $a_1[a_2[i]]$, where i is universally quantified), array reads by an universally quantified variable in the index guard, or general Presburger arithmetic expressions over universally quantified index variables in the index guard or in the value constraint.

Furthermore, the authors also present a decision procedure for an array theory in which indices are uninterpreted. With reference to the corresponding data structure in modern programming languages, these arrays are called *maps*. The corresponding decision procedure is called SAT_M and, aside from minor adjustments, is completely analogous to SAT_A .

On a sidenote, there exists an implementation of SAT_A for the π VC verifying compiler, which verifies programs written in the pi (for Prove It) programming language.