Array Logic

Deepak D'Souza

Department of Computer Science and Automation
Indian Institute of Science, Bangalore.

17, 19 May 2021

Outline

Array Logic (BM Ch 11, KS Ch 7)

Consider a (quantified or unquantified) program logic in which we allow

- A finite number of array variables a, b etc
- Terms corresponding to array reads: $a[i]$ (value stored at position i in a)
- Update operation on arrays: $a\langle i \triangleleft e\rangle$ (new array a' which coincides with a except at position i where it has value e).

Example formula

$$[(x < m) \wedge (0 \leq i) \wedge \forall k(((0 \leq k) \wedge (k < i)) \implies (a[k] \leq m)) \wedge a' = a\langle i \triangleleft x\rangle] \implies \forall k(((0 \leq k) \wedge (k \leq i)) \implies (a'[k] \leq m)).$$

Application: Verifying Array Programs

Floyd-Hoare style verification of array programs:

```
for (i = 0; i < N; i++)
  a[i] := 0;
// assert for each k: (0 <= k < N) => a[k] = 0
```

What is an adequate loop invariant for this program?

Application: Verifying Array Programs

Floyd-Hoare style verification of array programs:

```
for (i = 0; i < N; i++)
  a[i] := 0;
// assert for each k: (0 <= k < N) => a[k] = 0
```

What is an adequate loop invariant for this program?

$$\forall k((0 \leq k < i) \implies (a[k] = 0))$$

Application: Verifying Array Programs

Floyd-Hoare style verification of array programs:

```
for (i = 0; i < N; i++)
  a[i] := 0;
// assert for each k: (0 <= k < N) => a[k] = 0
```

What is an adequate loop invariant for this program?

$$\forall k((0 \leq k < i) \implies (a[k] = 0))$$

One of the verification conditions: Is the formula:

$$[\forall k((0 \leq k < i) \implies (a[k] = 0)) \wedge (i < N) \wedge (i' = i + 1) \wedge a' = a\{i \leftarrow 0\}] \implies \forall k((0 \leq k < i')) \implies (a'[k] = 0)).$$

valid?

General array logic is undecidable

- Problem of deciding whether a 2-counter machine halts is undecidable (no algorithm/decision-procedure can exist)
- Reduce to checking validity of array logic

Quantifier-Free Array Logic [BM Sec 9.5]

Consider FO signature

$$\Sigma_A = (\cdot[\cdot], \cdot\langle\cdot \lhd \cdot\rangle)$$

Consider quantifier-free formulas over Σ_A (equality only between Value-Terms allowed).

- Array-Term (a): $b \mid a\langle t \lhd t\rangle$
- Value-Term (t): $x \mid a[t]$
- Atomic-Formula: Value-Term $=$ Value-Term
- Formula: Boolean combination of Atomic-Formulas

Example formula

$$i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge a\langle i_1 \lhd v_1\rangle\langle i_2 \lhd v_2\rangle[j] \neq a[j]$$

Quantifier-Free Array Logic [BM Sec 9.5]

Reduce to EUF by using the "read-over-write" rule: Repeatedly replace $F(\cdots a\langle i \lhd v\rangle[j]\cdots)$ by

$$(i = j) \wedge F(\cdots v \cdots) \vee$$
$$(i \neq j) \wedge F(\cdots a[j] \cdots).$$

If no array writes then replace array variables a by functions f_a and array-reads $a[i]$ by $f_a(i)$ to get an EUF formula. Use decision procedure for EUF.

Array Property Formulas [Bradley, Manna, Sipma VMCAI 2006]

$T_{\mathsf{A}}^{\mathbb{Z}}$. An **array property** is again a $\Sigma_{\mathsf{A}}^{\mathbb{Z}}$-formula of the form

$$\forall \bar{i}.\ F[\bar{i}] \ \rightarrow \ G[\bar{i}]\ ,$$

where $\bar{i}$ is a list of integer variables, and $F[\bar{i}]$ and $G[\bar{i}]$ are the **index guard** and the **value constraint**, respectively. The form of an index guard is constrained according to the following grammar:

$$\begin{array}{rcl} \mathsf{iguard} & \rightarrow & \mathsf{iguard} \wedge \mathsf{iguard} \mid \mathsf{iguard} \vee \mathsf{iguard} \mid \mathsf{atom} \\ \mathsf{atom} & \rightarrow & \mathsf{expr} \leq \mathsf{expr} \mid \mathsf{expr} = \mathsf{expr} \\ \mathsf{expr} & \rightarrow & \mathit{uvar} \mid \mathsf{pexpr} \\ \mathsf{pexpr} & \rightarrow & \mathsf{pexpr}' \\ \mathsf{pexpr}' & \rightarrow & \mathbb{Z} \mid \mathbb{Z} \cdot \mathit{evar} \mid \mathsf{pexpr}' + \mathsf{pexpr}' \end{array}$$

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

Array Property Fragment [Bradley, Manna, Sipma VMCAI 2006]

Consider the fragment of FO logic of the combined signatures $\Sigma_A = (\cdot[\cdot], \cdot\langle\cdot \lhd \cdot\rangle)$ and $\Sigma_{LA} = (+, -, <, 0, 1)$ consisting of: Boolean combinations of quantifier-free formulas over $\Sigma_A \cup \Sigma_{LA}$ and Array Property formulas.

Example APF formula

$$
\begin{aligned}
& l \leq k \leq u + 1 \wedge \\
& a' = a\langle k \lhd 0\rangle \wedge \\
& a'[k] \neq b'[k] \wedge \\
& a'[u+1] = b[u+1] \wedge \\
& \forall i((l \leq i \leq u) \implies a[i] = b[i])
\end{aligned}
$$

Properties we can say in APF

- $\forall i(a[i] = b[j])$ (array equality)
- $\forall i((l \leq i \leq u) \implies a[i] = b[i])$ (bounded array equality)
- $\forall i((l \leq i \leq u) \implies 0 \leq a[i])$ (bounded universal property)
- $\forall i \forall j(i \leq j \implies a[i] \leq a[j])$ (increasing)

What we cannot say:

- $\forall i \forall j(i \neq j \implies a[i] \neq a[i])$ (distinct elements)
- $\forall i \forall j(i < j \implies a[i] < a[j])$ (strictly increasing)
- $\forall i(b[a[i]] = c[i])$

Reduction Algorithm

Algorithm 7.3.1: ARRAY-REDUCTION

Input: An array property formula ϕ_A in NNF
Output: A formula ϕ_{UF} in the index and element theories with uninterpreted functions

1. Apply the write rule to remove all array updates from ϕ_A.
2. Replace all existential quantifications of the form $\exists i \in T_I.\ P(i)$ by $P(j)$, where j is a fresh variable.
3. Replace all universal quantifications of the form $\forall i \in T_I.\ P(i)$ by

$$\bigwedge_{i \in \mathcal{I}(\phi)} P(i)\ .$$

4. Replace the array read operators by uninterpreted functions and obtain ϕ_{UF};
5. **return** ϕ_{UF};

Reduces APF to EUF+LA, which can be shown to be decidable using Nelson-Oppen

Example Reduction

Example

$$\forall i((l \leq i \leq u) \implies a[i] = b[i]) \wedge$$
$$\neg(\forall i((l \leq i \leq u+1) \implies (a\langle(u+1) \triangleleft b[u+1]\rangle[i] = b[i])$$

Proof of Correctness [BM]

Let $K = J[\vec{i} \mapsto \vec{v}]$.