

Decision Procedures

Jochen Hoenicke



Software Engineering
Albert-Ludwigs-University Freiburg

Summer 2012

Theory of Arrays

$$\Sigma_A : \{ \cdot[\cdot], \cdot\langle \cdot \triangleleft \cdot \rangle, = \} ,$$

where

- $a[i]$ is a binary function representing read of array a at index i ;
- $a\langle i \triangleleft v \rangle$ is a ternary function representing write of value v to index i of array a ;
- $=$ is a binary predicate. It is not used on arrays.

Axioms of T_A :

- 1 axioms of (reflexivity), (symmetry), and (transitivity) of T_E
- 2 $\forall a, i, j. i = j \rightarrow a[i] = a[j]$ (array congruence)
- 3 $\forall a, v, i, j. i = j \rightarrow a\langle i \triangleleft v \rangle[j] = v$ (read-over-write 1)
- 4 $\forall a, v, i, j. i \neq j \rightarrow a\langle i \triangleleft v \rangle[j] = a[j]$ (read-over-write 2)

Given quantifier-free conjunctive Σ_A -formula F .
To decide the T_A -satisfiability of F :

Step 1

For every read-over-write term $a\langle i \triangleleft v \rangle[j]$ in F , replace F with the formula

$$(i = j \wedge F\{a\langle i \triangleleft v \rangle[j] \mapsto v\}) \vee \\ (i \neq j \wedge F\{a\langle i \triangleleft v \rangle[j] \mapsto a[j]\})$$

Repeat until there are no more read-over-write terms.

Step 2

Associate array variables a with fresh function symbol f_a .
Replace read terms $a[i]$ with $f_a(i)$.

Step 2

Associate array variables a with fresh function symbol f_a .
Replace read terms $a[i]$ with $f_a(i)$.

Step 3

Now F is a T_E -Formula. Decide T_E -satisfiability using the congruence-closure algorithm for each of the disjuncts produced in Step 1.

Example: Consider Σ_A -formula

$$F : i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge a\langle i_1 \triangleleft v_1 \rangle \langle i_2 \triangleleft v_2 \rangle [j] \neq a[j] .$$

Example: Consider Σ_A -formula

$$F : i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge a\langle i_1 \triangleleft v_1 \rangle \langle i_2 \triangleleft v_2 \rangle [j] \neq a[j] .$$

F contains a read-over-write term,

$$a\langle i_1 \triangleleft v_1 \rangle \langle i_2 \triangleleft v_2 \rangle [j] \neq a[j] .$$

Example: Consider Σ_A -formula

$$F : i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge a\langle i_1 \triangleleft v_1 \rangle \langle i_2 \triangleleft v_2 \rangle [j] \neq a[j] .$$

F contains a read-over-write term,

$$a\langle i_1 \triangleleft v_1 \rangle \langle i_2 \triangleleft v_2 \rangle [j] \neq a[j] .$$

Rewrite it to $F_1 \vee F_2$ with:

$$F_1 : i_2 = j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge v_2 \neq a[j] ,$$

$$F_2 : i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge a\langle i_1 \triangleleft v_1 \rangle [j] \neq a[j] .$$

Example: Consider Σ_A -formula

$$F : i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge a\langle i_1 \triangleleft v_1 \rangle \langle i_2 \triangleleft v_2 \rangle [j] \neq a[j] .$$

F contains a read-over-write term,

$$a\langle i_1 \triangleleft v_1 \rangle \langle i_2 \triangleleft v_2 \rangle [j] \neq a[j] .$$

Rewrite it to $F_1 \vee F_2$ with:

$$F_1 : i_2 = j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge v_2 \neq a[j] ,$$

$$F_2 : i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge a\langle i_1 \triangleleft v_1 \rangle [j] \neq a[j] .$$

F_1 does not contain any write terms, so rewrite it to

$$F'_1 : i_2 = j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge f_a(j) = v_1 \wedge v_2 \neq f_a(j) .$$

The first two literals imply that $i_1 = i_2$, contradicting the third literal, so F'_1 is T_E -unsatisfiable.

Now, we try the second case (F_2):

F_2 contains the read-over-write term $a\langle i_1 \triangleleft v_1 \rangle[j]$. Rewrite it to $F_3 \vee F_4$ with

$$F_3 : i_1 = j \wedge i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge v_1 \neq a[j] ,$$

$$F_4 : i_1 \neq j \wedge i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge a[j] \neq a[j] .$$

Now, we try the second case (F_2):

F_2 contains the read-over-write term $a\langle i_1 \triangleleft v_1 \rangle[j]$. Rewrite it to $F_3 \vee F_4$ with

$$F_3 : i_1 = j \wedge i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge v_1 \neq a[j] ,$$

$$F_4 : i_1 \neq j \wedge i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge a[j] \neq a[j] .$$

Rewrite the array reads to

$$F_3' : i_1 = j \wedge i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge f_a(j) = v_1 \wedge v_1 \neq f_a(j) ,$$

$$F_4' : i_1 \neq j \wedge i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge f_a(j) = v_1 \wedge f_a(j) \neq f_a(j) .$$

Now, we try the second case (F_2):

F_2 contains the read-over-write term $a\langle i_1 \triangleleft v_1 \rangle[j]$. Rewrite it to $F_3 \vee F_4$ with

$$F_3 : i_1 = j \wedge i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge v_1 \neq a[j] ,$$

$$F_4 : i_1 \neq j \wedge i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge a[j] \neq a[j] .$$

Rewrite the array reads to

$$F'_3 : i_1 = j \wedge i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge f_a(j) = v_1 \wedge v_1 \neq f_a(j) ,$$

$$F'_4 : i_1 \neq j \wedge i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge f_a(j) = v_1 \wedge f_a(j) \neq f_a(j) .$$

In F'_3 there is a contradiction because of the final two terms. In F'_4 , there are two contradictions: the first and third literals contradict each other, and the final literal is contradictory. Since F is equisatisfiable to $F'_1 \vee F'_3 \vee F'_4$, F is T_A -unsatisfiable.

Now, we try the second case (F_2):

F_2 contains the read-over-write term $a\langle i_1 \triangleleft v_1 \rangle[j]$. Rewrite it to $F_3 \vee F_4$ with

$$F_3 : i_1 = j \wedge i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge v_1 \neq a[j] ,$$

$$F_4 : i_1 \neq j \wedge i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge a[j] \neq a[j] .$$

Rewrite the array reads to

$$F_3' : i_1 = j \wedge i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge f_a(j) = v_1 \wedge v_1 \neq f_a(j) ,$$

$$F_4' : i_1 \neq j \wedge i_2 \neq j \wedge i_1 = j \wedge i_1 \neq i_2 \wedge f_a(j) = v_1 \wedge f_a(j) \neq f_a(j) .$$

In F_3' there is a contradiction because of the final two terms. In F_4' , there are two contradictions: the first and third literals contradict each other, and the final literal is contradictory. Since F is equisatisfiable to $F_1' \vee F_3' \vee F_4'$, F is T_A -unsatisfiable.

Suppose instead that F does not contain the literal $i_1 \neq i_2$. Is this new formula T_A -satisfiable?

Our algorithm has a big disadvantage. Step 1 doubles the size of the formula:

$$(i = j \wedge F\{a\langle i \triangleleft v \rangle[j] \mapsto v\}) \vee \\ (i \neq j \wedge F\{a\langle i \triangleleft v \rangle[j] \mapsto a[j]\})$$

Our algorithm has a big disadvantage. Step 1 doubles the size of the formula:

$$(i = j \wedge F\{a\langle i \triangleleft v \rangle[j] \mapsto v\}) \vee \\ (i \neq j \wedge F\{a\langle i \triangleleft v \rangle[j] \mapsto a[j]\})$$

This can be avoided by introducing fresh variables x_{ajv} :

$$F\{a\langle i \triangleleft v \rangle[j] \mapsto x_{ajv}\} \wedge \\ ((i = j \wedge x_{ajv} = v) \vee (i \neq j \wedge x_{ajv} = a[j]))$$

However, this is not in the **conjunctive** fragment of T_E .

Our algorithm has a big disadvantage. Step 1 doubles the size of the formula:

$$(i = j \wedge F\{a\langle i \triangleleft v \rangle[j] \mapsto v\}) \vee \\ (i \neq j \wedge F\{a\langle i \triangleleft v \rangle[j] \mapsto a[j]\})$$

This can be avoided by introducing fresh variables x_{ajv} :

$$F\{a\langle i \triangleleft v \rangle[j] \mapsto x_{ajv}\} \wedge \\ ((i = j \wedge x_{ajv} = v) \vee (i \neq j \wedge x_{ajv} = a[j]))$$

However, this is not in the **conjunctive** fragment of T_E .

There is no way around:

The conjunctive fragment of T_A is NP-complete.

In programming languages, one often needs to express the following concepts:

- **Containment** $contains(a, \ell, u, e)$: the array a contains element e at some index between ℓ and u .
- **Sortedness** $sorted(a, \ell, u)$: the array a is sorted between index ℓ and index u .
- **Partitioning** $partition(a, \ell_1, u_1, \ell_2, u_2)$: The array elements between ℓ_1 and u_1 are smaller than all elements between ℓ_2 and u_2 .

In programming languages, one often needs to express the following concepts:

- **Containment** $\text{contains}(a, \ell, u, e)$: the array a contains element e at some index between ℓ and u .

$$\exists i. \ell \leq i \leq u \wedge a[i] = e$$

- **Sortedness** $\text{sorted}(a, \ell, u)$: the array a is sorted between index ℓ and index u .

$$\forall i, j. \ell \leq i \leq j \leq u \implies a[i] \leq a[j]$$

- **Partitioning** $\text{partition}(a, \ell_1, u_1, \ell_2, u_2)$: The array elements between ℓ_1 and u_1 are smaller than all elements between ℓ_2 and u_2 .

$$\forall i, j. \ell_1 \leq i \leq u_1 \wedge \ell_2 \leq j \leq u_2 \implies a[i] \leq a[j]$$

These concepts can only be expressed as first-order formulae **with** quantifiers.

These concepts can only be expressed as first-order formulae **with quantifiers**.

However: the general theory of arrays T_A with quantifier is **not decidable**.

These concepts can only be expressed as first-order formulae **with quantifiers**.

However: the general theory of arrays T_A with quantifier is **not decidable**.

Is there a **decidable** fragment of T_A that contains the above formulae?

We want to prove validity for a formula, such as:

$$\neg \text{contains}(a, l, u, e) \wedge e \neq f \rightarrow \neg \text{contains}(a\langle j \triangleleft f \rangle, l, u, e)$$

$$\begin{aligned} &\neg(\exists i. l \leq i \leq u \wedge a[i] = e) \wedge e \neq f \\ &\rightarrow \neg(\exists i. l \leq i \leq u \wedge a\langle j \triangleleft f \rangle[i] \neq e). \end{aligned}$$

We want to prove validity for a formula, such as:

$$\neg \text{contains}(a, l, u, e) \wedge e \neq f \rightarrow \neg \text{contains}(a\langle j \triangleleft f \rangle, l, u, e)$$

$$\begin{aligned} &\neg(\exists i. l \leq i \leq u \wedge a[i] = e) \wedge e \neq f \\ &\rightarrow \neg(\exists i. l \leq i \leq u \wedge a\langle j \triangleleft f \rangle[i] \neq e). \end{aligned}$$

Check satisfiability of negated formula:

$$\neg(\exists i. l \leq i \leq u \wedge a[i] = e) \wedge e \neq f \wedge (\exists i. l \leq i \leq u \wedge a\langle j \triangleleft f \rangle[i] \neq e).$$

We want to prove validity for a formula, such as:

$$\neg \text{contains}(a, \ell, u, e) \wedge e \neq f \rightarrow \neg \text{contains}(a \langle j \triangleleft f \rangle, \ell, u, e)$$

$$\begin{aligned} & \neg(\exists i. \ell \leq i \leq u \wedge a[i] = e) \wedge e \neq f \\ & \rightarrow \neg(\exists i. \ell \leq i \leq u \wedge a \langle j \triangleleft f \rangle [i] \neq e). \end{aligned}$$

Check satisfiability of negated formula:

$$\neg(\exists i. \ell \leq i \leq u \wedge a[i] = e) \wedge e \neq f \wedge (\exists i. \ell \leq i \leq u \wedge a \langle j \triangleleft f \rangle [i] \neq e).$$

Negation Normal Form:

$$(\forall i. \ell > i \vee i > u \vee a[i] \neq e) \wedge e \neq f \wedge (\exists i. \ell \leq i \wedge i \leq u \wedge a \langle j \triangleleft f \rangle [i] = e).$$

We want to prove validity for a formula, such as:

$$\neg \text{contains}(a, l, u, e) \wedge e \neq f \rightarrow \neg \text{contains}(a \langle j \triangleleft f \rangle, l, u, e)$$

$$\begin{aligned} &\neg(\exists i. l \leq i \leq u \wedge a[i] = e) \wedge e \neq f \\ &\rightarrow \neg(\exists i. l \leq i \leq u \wedge a \langle j \triangleleft f \rangle [i] \neq e). \end{aligned}$$

Check satisfiability of negated formula:

$$\neg(\exists i. l \leq i \leq u \wedge a[i] = e) \wedge e \neq f \wedge (\exists i. l \leq i \leq u \wedge a \langle j \triangleleft f \rangle [i] \neq e).$$

Negation Normal Form:

$$(\forall i. l > i \vee i > u \vee a[i] \neq e) \wedge e \neq f \wedge (\exists i. l \leq i \wedge i \leq u \wedge a \langle j \triangleleft f \rangle [i] = e).$$

or the equisatisfiable formula

$$\forall i. l > i \vee i > u \vee a[i] \neq e \wedge e \neq f \wedge l \leq i_2 \wedge i_2 \leq u \wedge a \langle j \triangleleft f \rangle [i_2] = e.$$

We want to prove validity for a formula, such as:

$$\neg \text{contains}(a, l, u, e) \wedge e \neq f \rightarrow \neg \text{contains}(a \langle j \triangleleft f \rangle, l, u, e)$$

$$\begin{aligned} &\neg(\exists i. l \leq i \leq u \wedge a[i] = e) \wedge e \neq f \\ &\rightarrow \neg(\exists i. l \leq i \leq u \wedge a \langle j \triangleleft f \rangle [i] \neq e). \end{aligned}$$

Check satisfiability of negated formula:

$$\neg(\exists i. l \leq i \leq u \wedge a[i] = e) \wedge e \neq f \wedge (\exists i. l \leq i \leq u \wedge a \langle j \triangleleft f \rangle [i] \neq e).$$

Negation Normal Form:

$$(\forall i. l > i \vee i > u \vee a[i] \neq e) \wedge e \neq f \wedge (\exists i. l \leq i \wedge i \leq u \wedge a \langle j \triangleleft f \rangle [i] = e).$$

or the equisatisfiable formula

$$\forall i. l > i \vee i > u \vee a[i] \neq e \wedge e \neq f \wedge l \leq i_2 \wedge i_2 \leq u \wedge a \langle j \triangleleft f \rangle [i_2] = e.$$

We need to handle satisfiability for **universal quantifiers**.

Array Property Fragment of T_A

Decidable fragment of T_A that includes \forall quantifiers

Array Property Fragment of T_A

Decidable fragment of T_A that includes \forall quantifiers

Array property

Σ_A -formula of form

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

where \vec{i} is a list of variables.

Decidable fragment of T_A that includes \forall quantifiers

Array property

Σ_A -formula of form

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

where \vec{i} is a list of variables.

- index guard $F[\vec{i}]$:

$$\begin{aligned} \text{iguard} &\rightarrow \text{iguard} \wedge \text{iguard} \mid \text{iguard} \vee \text{iguard} \mid \text{atom} \\ \text{atom} &\rightarrow \text{var} = \text{var} \mid \text{evar} \neq \text{var} \mid \text{var} \neq \text{evar} \mid \top \\ \text{var} &\rightarrow \text{evar} \mid \text{uvar} \end{aligned}$$

where *uvar* is any universally quantified index variable,
and *evar* is any constant or unquantified variable.

Array Property Fragment of T_A

Decidable fragment of T_A that includes \forall quantifiers

Array property

Σ_A -formula of form

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

where \vec{i} is a list of variables.

- **index guard** $F[\vec{i}]$:

$$\begin{aligned} \text{iguard} &\rightarrow \text{iguard} \wedge \text{iguard} \mid \text{iguard} \vee \text{iguard} \mid \text{atom} \\ \text{atom} &\rightarrow \text{var} = \text{var} \mid \text{evar} \neq \text{var} \mid \text{var} \neq \text{evar} \mid \top \\ \text{var} &\rightarrow \text{evar} \mid \text{uvar} \end{aligned}$$

where *uvar* is any universally quantified index variable,
and *evar* is any constant or unquantified variable.

- **value constraint** $G[\vec{i}]$: a universally quantified index can occur in a value constraint $G[\vec{i}]$ only in a read $a[i]$, where a is an array term. The read cannot be nested; for example, $a[b[i]]$ is not allowed.

Array property Fragment: Boolean combinations of quantifier-free T_A -formulae and array properties

Example: Array Property Fragment

Is this formula in the array property fragment?

$$F : \forall i. i \neq a[k] \rightarrow a[i] = a[k]$$

Is this formula in the array property fragment?

$$F : \forall i. i \neq a[k] \rightarrow a[i] = a[k]$$

The antecedent is not a legal index guard since $a[k]$ is not a variable (neither a *uvar* nor an *evar*); however, by simple manipulation

$$F' : v = a[k] \wedge \forall i. i \neq v \rightarrow a[i] = a[k]$$

Here, $i \neq v$ is a legal index guard, and $a[i] = a[k]$ is a legal value constraint. F and F' are equisatisfiable.

Is this formula in the array property fragment?

$$F : \forall i. i \neq a[k] \rightarrow a[i] = a[k]$$

The antecedent is not a legal index guard since $a[k]$ is not a variable (neither a *uvar* nor an *evar*); however, by simple manipulation

$$F' : v = a[k] \wedge \forall i. i \neq v \rightarrow a[i] = a[k]$$

Here, $i \neq v$ is a legal index guard, and $a[i] = a[k]$ is a legal value constraint. F and F' are equisatisfiable.

This trick works for every term that does not contain a *uvar*.

Example: Array Property Fragment

Is this formula in the array property fragment?

$$F : \forall i. i \neq a[k] \rightarrow a[i] = a[k]$$

The antecedent is not a legal index guard since $a[k]$ is not a variable (neither a *uvar* nor an *evar*); however, by simple manipulation

$$F' : v = a[k] \wedge \forall i. i \neq v \rightarrow a[i] = a[k]$$

Here, $i \neq v$ is a legal index guard, and $a[i] = a[k]$ is a legal value constraint. F and F' are equisatisfiable.

This trick works for every term that does not contain a *uvar*.

However, no manipulation works for:

$$G : \forall i. i \neq a[i] \rightarrow a[i] = a[k] .$$

Thus, G is not in the array property fragment.

Is this formula in the array property fragment?

$$F' : \forall ij. i \neq j \rightarrow a[i] \neq a[j]$$

Is this formula in the array property fragment?

$$F' : \forall ij. i \neq j \rightarrow a[i] \neq a[j]$$

No, the term $\text{uvar} \neq \text{uvar}$ is not allowed in the index guard. There is no workaround.

Remark: Array property fragment allows expressing equality between arrays (**extensionality**): two arrays are equal precisely when their corresponding elements are equal.

For given formula

$$F : \dots \wedge a = b \wedge \dots$$

with array terms a and b , rewrite F as

$$F' : \dots \wedge (\forall i. \top \rightarrow a[i] = b[i]) \wedge \dots .$$

F and F' are equisatisfiable.

F' is in array property fragment of T_A .

Basic Idea: Similar to quantifier elimination.

Basic Idea: Similar to quantifier elimination.

Replace universal quantification

$$\forall i. F[i]$$

by finite conjunction

$$F[t_1] \wedge \dots \wedge F[t_n].$$

We call t_1, \dots, t_n the **index terms** and they depend on the formula.

Consider

$$F : a \langle i \triangleleft v \rangle = a \wedge a[i] \neq v ,$$

Consider

$$F : a\langle i \triangleleft v \rangle = a \wedge a[i] \neq v ,$$

which expands to

$$F' : \forall j. a\langle i \triangleleft v \rangle[j] = a[j] \wedge a[i] \neq v .$$

Consider

$$F : a\langle i \triangleleft v \rangle = a \wedge a[i] \neq v ,$$

which expands to

$$F' : \forall j. a\langle i \triangleleft v \rangle[j] = a[j] \wedge a[i] \neq v .$$

Intuitively, only the index i is important:

$$F'' : \left(\bigwedge_{j \in \{i\}} a\langle i \triangleleft v \rangle[j] = a[j] \right) \wedge a[i] \neq v ,$$

Consider

$$F : a\langle i \triangleleft v \rangle = a \wedge a[i] \neq v ,$$

which expands to

$$F' : \forall j. a\langle i \triangleleft v \rangle[j] = a[j] \wedge a[i] \neq v .$$

Intuitively, only the index i is important:

$$F'' : \left(\bigwedge_{j \in \{i\}} a\langle i \triangleleft v \rangle[j] = a[j] \right) \wedge a[i] \neq v ,$$

or simply

$$a\langle i \triangleleft v \rangle[i] = a[i] \wedge a[i] \neq v .$$

Consider

$$F : a \langle i \triangleleft v \rangle = a \wedge a[i] \neq v ,$$

which expands to

$$F' : \forall j. a \langle i \triangleleft v \rangle [j] = a[j] \wedge a[i] \neq v .$$

Intuitively, only the index i is important:

$$F'' : \left(\bigwedge_{j \in \{i\}} a \langle i \triangleleft v \rangle [j] = a[j] \right) \wedge a[i] \neq v ,$$

or simply

$$a \langle i \triangleleft v \rangle [i] = a[i] \wedge a[i] \neq v .$$

Simplifying,

$$v = a[i] \wedge a[i] \neq v ,$$

it is clear that this formula, and thus F , is T_A -unsatisfiable.

Decision Procedure for Array Property Fragment

Given array property formula F , decide its T_A -satisfiability by the following steps:

Given array property formula F , decide its T_A -satisfiability by the following steps:

Step 1

Put F in NNF, but do not rewrite inside a quantifier.

Given array property formula F , decide its T_A -satisfiability by the following steps:

Step 1

Put F in NNF, but do not rewrite inside a quantifier.

Step 2

Apply the following rule exhaustively to remove writes:

$$\frac{F[a\langle i \triangleleft v \rangle]}{F[a'] \wedge a'[i] = v \wedge (\forall j. j \neq i \rightarrow a[j] = a'[j])} \text{ for fresh } a' \quad (\text{write})$$

After an application of the rule, the resulting formula contains at least one fewer write terms than the given formula.

Decision Procedure for Array Property Fragment

Given array property formula F , decide its T_A -satisfiability by the following steps:

Step 1

Put F in NNF, but do not rewrite inside a quantifier.

Step 2

Apply the following rule exhaustively to remove writes:

$$\frac{F[a\langle i \triangleleft v \rangle]}{F[a'] \wedge a'[i] = v \wedge (\forall j. j \neq i \rightarrow a[j] = a'[j])} \text{ for fresh } a' \quad (\text{write})$$

After an application of the rule, the resulting formula contains at least one fewer write terms than the given formula.

Step 3

Apply the following rule exhaustively to remove existential quantification:

$$\frac{F[\exists \bar{i}. G[\bar{i}]]}{F[G[\bar{j}]]} \text{ for fresh } \bar{j} \quad (\text{exists})$$

Existential quantification can arise during Step 1 if the given formula has a negated array property.

Steps 4-6 accomplish the reduction of universal quantification to finite conjunction.

Main idea: select a set of symbolic index terms on which to instantiate all universal quantifiers. The set is sufficient for correctness.

Steps 4-6 accomplish the reduction of universal quantification to finite conjunction.

Main idea: select a set of symbolic index terms on which to instantiate all universal quantifiers. The set is sufficient for correctness.

Step 4

From the output F_3 of Step 3, construct the **index set** \mathcal{I} :

$$\mathcal{I} = \{\lambda\} \cup \{t : \cdot[t] \in F_3 \text{ such that } t \text{ is not a universally quantified variable}\} \cup \{t : t \text{ occurs as an } evar \text{ in the parsing of index guards}\}$$

This index set is the finite set of indices that need to be examined. It includes

- all terms t that occur in some read $a[t]$ anywhere in F (unless it is a universally quantified variable)
- all terms t (constant or unquantified variable) that are compared to a universally quantified variable in some index guard.
- λ is a fresh constant that represents all other index positions that are not explicitly in \mathcal{I} .

Step 5 (Key step)

Apply the following rule exhaustively to remove universal quantification:

$$\frac{H[\forall \vec{i}. F[\vec{i}] \rightarrow G[\vec{i}]]}{H \left[\bigwedge_{\vec{i} \in \mathcal{I}^n} (F[\vec{i}] \rightarrow G[\vec{i}]) \right]} \quad (\text{forall})$$

where n is the number of quantified variables \vec{i} .

Step 5 (Key step)

Apply the following rule exhaustively to remove universal quantification:

$$\frac{H[\forall \bar{i}. F[\bar{i}] \rightarrow G[\bar{i}]]}{H \left[\bigwedge_{\bar{i} \in \mathcal{I}^n} (F[\bar{i}] \rightarrow G[\bar{i}]) \right]} \quad (\text{forall})$$

where n is the number of quantified variables \bar{i} .

Step 6

From the output F_5 of Step 5, construct

$$F_6 : F_5 \wedge \bigwedge_{i \in \mathcal{I} \setminus \{\lambda\}} \lambda \neq i .$$

The new conjuncts assert that the variable λ introduced in Step 4 is indeed unique.

Step 5 (Key step)

Apply the following rule exhaustively to remove universal quantification:

$$\frac{H[\forall \vec{i}. F[\vec{i}] \rightarrow G[\vec{i}]]}{H \left[\bigwedge_{\vec{i} \in \mathcal{I}^n} (F[\vec{i}] \rightarrow G[\vec{i}]) \right]} \quad (\text{forall})$$

where n is the number of quantified variables \vec{i} .

Step 6

From the output F_5 of Step 5, construct

$$F_6 : F_5 \wedge \bigwedge_{i \in \mathcal{I} \setminus \{\lambda\}} \lambda \neq i .$$

The new conjuncts assert that the variable λ introduced in Step 4 is indeed unique.

Step 7

Decide the T_A -satisfiability of F_6 using the decision procedure for the quantifier-free fragment.

Is this $T_A^=$ -formula valid?

$$F : (\forall i. i \neq k \rightarrow a[i] = b[i]) \wedge b[k] = v \rightarrow a\langle k \triangleleft v \rangle = b$$

Is this $T_A^=$ -formula valid?

$$F : (\forall i. i \neq k \rightarrow a[i] = b[i]) \wedge b[k] = v \rightarrow a\langle k \triangleleft v \rangle = b$$

Check satisfiability of:

$$\neg((\forall i. i \neq k \rightarrow a[i] = b[i]) \wedge b[k] = v \rightarrow (\forall i. a\langle k \triangleleft v \rangle[i] = b[i]))$$

Is this $T_A^=$ -formula valid?

$$F : (\forall i. i \neq k \rightarrow a[i] = b[i]) \wedge b[k] = v \rightarrow a\langle k \triangleleft v \rangle = b$$

Check satisfiability of:

$$\neg((\forall i. i \neq k \rightarrow a[i] = b[i]) \wedge b[k] = v \rightarrow (\forall i. a\langle k \triangleleft v \rangle[i] = b[i]))$$

Step 1: NNF

$$F_1 : (\forall i. i \neq k \rightarrow a[i] = b[i]) \wedge b[k] = v \wedge (\exists i. a\langle k \triangleleft v \rangle[i] \neq b[i])$$

Step 2: Remove array writes

$$F_2 : (\forall i. i \neq k \rightarrow a[i] = b[i]) \wedge b[k] = v \wedge (\exists i. a'[i] \neq b[i]) \\ \wedge a'[k] = v \wedge (\forall i. i \neq k \rightarrow a'[i] = a[i])$$

Step 3: Remove existential quantifier

$$F_3 : (\forall i. i \neq k \rightarrow a[i] = b[i]) \wedge b[k] = v \wedge a'[j] \neq b[j] \\ \wedge a'[k] = v \wedge (\forall i. i \neq k \rightarrow a'[i] = a[i])$$

Step 4: Compute index set

Step 4: Compute index set $\mathcal{I} = \{\lambda, k, j\}$

Step 5+6: Replace universal quantifier:

Step 4: Compute index set $\mathcal{I} = \{\lambda, k, j\}$

Step 5+6: Replace universal quantifier:

$$\begin{aligned} F_6 : & (\lambda \neq k \rightarrow a[\lambda] = b[\lambda]) \\ & \wedge (k \neq k \rightarrow a[k] = b[k]) \\ & \wedge (j \neq k \rightarrow a[j] = b[j]) \\ & \wedge b[k] = v \wedge a'[j] \neq b[j] \wedge a'[k] = v \\ & \wedge (\lambda \neq k \rightarrow a'[\lambda] = a[\lambda]) \\ & \wedge (k \neq k \rightarrow a'[k] = a[k]) \\ & \wedge (j \neq k \rightarrow a'[j] = a[j]) \\ & \wedge \lambda \neq k \wedge \lambda \neq j \end{aligned}$$

Step 4: Compute index set $\mathcal{I} = \{\lambda, k, j\}$

Step 5+6: Replace universal quantifier:

$$\begin{aligned}
 F_6 : & (\lambda \neq k \rightarrow a[\lambda] = b[\lambda]) \\
 & \wedge (k \neq k \rightarrow a[k] = b[k]) \\
 & \wedge (j \neq k \rightarrow a[j] = b[j]) \\
 & \wedge b[k] = v \wedge a'[j] \neq b[j] \wedge a'[k] = v \\
 & \wedge (\lambda \neq k \rightarrow a'[\lambda] = a[\lambda]) \\
 & \wedge (k \neq k \rightarrow a'[k] = a[k]) \\
 & \wedge (j \neq k \rightarrow a'[j] = a[j]) \\
 & \wedge \lambda \neq k \wedge \lambda \neq j
 \end{aligned}$$

Case distinction on $j = k$ proves unsatisfiability of F_6 .

Therefore F is valid

Is this formula satisfiable?

$$F : (\forall i. i \neq j \rightarrow a[i] = b[i]) \wedge (\forall i. i \neq k \rightarrow a[i] \neq b[i])$$

Is this formula satisfiable?

$$F : (\forall i. i \neq j \rightarrow a[i] = b[i]) \wedge (\forall i. i \neq k \rightarrow a[i] \neq b[i])$$

The algorithm produces:

$$\begin{aligned} F_6 : & \lambda \neq j \rightarrow a[\lambda] = b[\lambda] \\ & \wedge j \neq j \rightarrow a[j] = b[j] \\ & \wedge k \neq j \rightarrow a[k] = b[k] \\ & \wedge \lambda \neq k \rightarrow a[\lambda] \neq b[\lambda] \\ & \wedge j \neq k \rightarrow a[j] \neq b[j] \\ & \wedge k \neq k \rightarrow a[k] \neq b[k] \\ & \wedge \lambda \neq j \wedge \lambda \neq k \end{aligned}$$

The first, fourth and last line give a contradiction!

Without λ we had the formula:

$$\begin{aligned} F'_6 : & j \neq j \rightarrow a[j] = b[j] \\ & \wedge k \neq j \rightarrow a[k] = b[k] \\ & \wedge j \neq k \rightarrow a[j] \neq b[j] \\ & \wedge k \neq k \rightarrow a[k] \neq b[k] \end{aligned}$$

which simplifies to:

$$j \neq k \rightarrow a[k] = b[k] \wedge a[j] \neq b[j].$$

This formula is satisfiable!

Theorem

Consider a Σ_A -formula F from the array property fragment of T_A . The output F_6 of Step 6 of the algorithm is T_A -equisatisfiable to F .

Theorem

Consider a Σ_A -formula F from the array property fragment of T_A . The output F_6 of Step 6 of the algorithm is T_A -equisatisfiable to F .

This also works when extending the Logic with an arbitrary theory T with signature Σ for the elements:

Theorem

Consider a $\Sigma_A \cup \Sigma$ -formula F from the array property fragment of $T_A \cup T$. The output F_6 of Step 6 of the algorithm is $T_A \cup T$ -equisatisfiable to F .

Proof: It is easy to see that steps 1–3 do not change the satisfiability of formula.

Proof: It is easy to see that steps 1–3 do not change the satisfiability of formula.

For step 4–6 we need to show:

- (1) $H[\forall \bar{i}. (F[\bar{i}] \rightarrow G[\bar{i}])] is satisfiable$
iff.
- (2) $H[\bigwedge_{\bar{i} \in \mathcal{I}^n} (F[\bar{i}] \rightarrow G[\bar{i}])] \wedge \bigwedge_{i \in \mathcal{I} \setminus \{\lambda\}} \lambda \neq i$ is satisfiable.

Proof: It is easy to see that steps 1–3 do not change the satisfiability of formula.

For step 4–6 we need to show:

(1) $H[\forall \bar{i}. (F[\bar{i}] \rightarrow G[\bar{i}])] is satisfiable$
iff.

(2) $H[\bigwedge_{\bar{i} \in \mathcal{I}^n} (F[\bar{i}] \rightarrow G[\bar{i}])] \wedge \bigwedge_{i \in \mathcal{I} \setminus \{\lambda\}} \lambda \neq i$ is satisfiable.

If the formula (1) is satisfied some Interpretation, then (2) holds in the same interpretation.

If the formula (2) holds in some interpretation I , we construct an interpretation J as follows:

$$\text{proj}_{\mathcal{I}}(j) = \begin{cases} i & \text{if } i \in \mathcal{I} \wedge \alpha_I[j] = \alpha_I[i] \\ \lambda & \text{otherwise} \end{cases}$$

$$\alpha_J[a[j]] = \alpha_I[a[\text{proj}_{\mathcal{I}}(j)]]$$

$$\alpha_J[x] = \alpha_I[x] \text{ for every non-array variable and constant}$$

J interprets the symbols occurring in formula (2) in the same way as I . Therefore, (2) holds in J .

If the formula (2) holds in some interpretation I , we construct an interpretation J as follows:

$$\text{proj}_{\mathcal{I}}(j) = \begin{cases} i & \text{if } i \in \mathcal{I} \wedge \alpha_I[j] = \alpha_I[i] \\ \lambda & \text{otherwise} \end{cases}$$

$$\alpha_J[a[j]] = \alpha_I[a[\text{proj}_{\mathcal{I}}(j)]]$$

$$\alpha_J[x] = \alpha_I[x] \text{ for every non-array variable and constant}$$

J interprets the symbols occurring in formula (2) in the same way as I . Therefore, (2) holds in J .

To prove that formula (1) holds in J , it suffices to show:

$$J \models \bigwedge_{\vec{i} \in \mathcal{I}^n} (F[\vec{i}] \rightarrow G[\vec{i}]) \text{ implies } J \models \forall \vec{i}. (F[\vec{i}] \rightarrow G[\vec{i}])$$

Proof of Theorem (cont)

Assume $J \models \bigwedge_{\bar{i} \in \mathcal{I}^n} (F[\bar{i}] \rightarrow G[\bar{i}])$. Show:

$$F[\bar{i}] \rightarrow F[\text{proj}_{\mathcal{I}}(\bar{i})] \rightarrow G[\text{proj}_{\mathcal{I}}(\bar{i})] \rightarrow G[\bar{i}]$$

Assume $J \models \bigwedge_{\bar{i} \in \mathcal{I}^n} (F[\bar{i}] \rightarrow G[\bar{i}])$. Show:

$$F[\bar{i}] \rightarrow F[\text{proj}_{\mathcal{I}}(\bar{i})] \rightarrow G[\text{proj}_{\mathcal{I}}(\bar{i})] \rightarrow G[\bar{i}]$$

The first implication $F[\bar{i}] \rightarrow F[\text{proj}_{\mathcal{I}}(\bar{i})]$ can be shown by structural induction over F . Base cases:

- $\text{var}_1 = \text{var}_2 \rightarrow \text{proj}_{\mathcal{I}}(\text{var}_1) = \text{proj}_{\mathcal{I}}(\text{var}_2)$: trivial.

- $\text{evar}_1 \neq \text{var}_2 \rightarrow \text{proj}_{\mathcal{I}}(\text{evar}_1) \neq \text{proj}_{\mathcal{I}}(\text{var}_2)$:

By definition of \mathcal{I} : $\text{evar}_1 \in \mathcal{I} \setminus \{\lambda\}$.

If $\text{evar}_1 = \text{proj}_{\mathcal{I}}(\text{evar}_1) = \text{proj}_{\mathcal{I}}(\text{var}_2)$, then $\text{var}_2 \in \mathcal{I} \setminus \{\lambda\}$, hence $\text{evar}_1 = \text{proj}_{\mathcal{I}}(\text{var}_2) = \text{var}_2$

- $\text{var}_1 \neq \text{evar}_2$ analogously.

The induction step is trivial.

Assume $J \models \bigwedge_{\bar{i} \in \mathcal{I}^n} (F[\bar{i}] \rightarrow G[\bar{i}])$. Show:

$$F[\bar{i}] \rightarrow F[\text{proj}_{\mathcal{I}}(\bar{i})] \rightarrow G[\text{proj}_{\mathcal{I}}(\bar{i})] \rightarrow G[\bar{i}]$$

The first implication $F[\bar{i}] \rightarrow F[\text{proj}_{\mathcal{I}}(\bar{i})]$ can be shown by structural induction over F . Base cases:

- $\text{var}_1 = \text{var}_2 \rightarrow \text{proj}_{\mathcal{I}}(\text{var}_1) = \text{proj}_{\mathcal{I}}(\text{var}_2)$: trivial.

- $\text{evar}_1 \neq \text{var}_2 \rightarrow \text{proj}_{\mathcal{I}}(\text{evar}_1) \neq \text{proj}_{\mathcal{I}}(\text{var}_2)$:

By definition of \mathcal{I} : $\text{evar}_1 \in \mathcal{I} \setminus \{\lambda\}$.

If $\text{evar}_1 = \text{proj}_{\mathcal{I}}(\text{evar}_1) = \text{proj}_{\mathcal{I}}(\text{var}_2)$, then $\text{var}_2 \in \mathcal{I} \setminus \{\lambda\}$, hence $\text{evar}_1 = \text{proj}_{\mathcal{I}}(\text{var}_2) = \text{var}_2$

- $\text{var}_1 \neq \text{evar}_2$ analogously.

The induction step is trivial.

The second implication $F[\text{proj}_{\mathcal{I}}(\bar{i})] \rightarrow G[\text{proj}_{\mathcal{I}}(\bar{i})]$ holds by assumption.

Proof of Theorem (cont)

Assume $J \models \bigwedge_{\bar{i} \in \mathcal{I}^n} (F[\bar{i}] \rightarrow G[\bar{i}])$. Show:

$$F[\bar{i}] \rightarrow F[\text{proj}_{\mathcal{I}}(\bar{i})] \rightarrow G[\text{proj}_{\mathcal{I}}(\bar{i})] \rightarrow G[\bar{i}]$$

The first implication $F[\bar{i}] \rightarrow F[\text{proj}_{\mathcal{I}}(\bar{i})]$ can be shown by structural induction over F . Base cases:

- $\text{var}_1 = \text{var}_2 \rightarrow \text{proj}_{\mathcal{I}}(\text{var}_1) = \text{proj}_{\mathcal{I}}(\text{var}_2)$: trivial.

- $\text{evar}_1 \neq \text{var}_2 \rightarrow \text{proj}_{\mathcal{I}}(\text{evar}_1) \neq \text{proj}_{\mathcal{I}}(\text{var}_2)$:

By definition of \mathcal{I} : $\text{evar}_1 \in \mathcal{I} \setminus \{\lambda\}$.

If $\text{evar}_1 = \text{proj}_{\mathcal{I}}(\text{evar}_1) = \text{proj}_{\mathcal{I}}(\text{var}_2)$, then $\text{var}_2 \in \mathcal{I} \setminus \{\lambda\}$, hence $\text{evar}_1 = \text{proj}_{\mathcal{I}}(\text{var}_2) = \text{var}_2$

- $\text{var}_1 \neq \text{evar}_2$ analogously.

The induction step is trivial.

The second implication $F[\text{proj}_{\mathcal{I}}(\bar{i})] \rightarrow G[\text{proj}_{\mathcal{I}}(\bar{i})]$ holds by assumption.

The third implication $G[\text{proj}_{\mathcal{I}}(\bar{i})] \implies G[\bar{i}]$ holds because G contains variables i only in array reads $a[i]$. By definition of J :

$$\alpha_J[a[i]] = \alpha_J[a[\text{proj}_{\mathcal{I}}(i)]]$$

Theory of Integer-Indexed Arrays

\leq enables reasoning about subarrays and properties such as subarray is sorted or partitioned.

signature of $T_A^{\mathbb{Z}}$: $\Sigma_A^{\mathbb{Z}} = \Sigma_A \cup \Sigma_{\mathbb{Z}}$

axioms of $T_A^{\mathbb{Z}}$: both axioms of T_A and $T_{\mathbb{Z}}$

Array Property Fragment of $T_A^{\mathbb{Z}}$

Array property: $\Sigma_A^{\mathbb{Z}}$ -formula of the form

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

where \bar{i} is a list of integer variables.

- $F[\bar{i}]$ **index guard:**

$$\text{iguard} \rightarrow \text{iguard} \wedge \text{iguard} \mid \text{iguard} \vee \text{iguard} \mid \text{atom}$$

$$\text{atom} \rightarrow \text{expr} \leq \text{expr} \mid \text{expr} = \text{expr}$$

$$\text{expr} \rightarrow \text{uvar} \mid \text{pexpr}$$

$$\text{pexpr} \rightarrow \text{pexpr}'$$

$$\text{pexpr}' \rightarrow \mathbb{Z} \mid \mathbb{Z} \cdot \text{evar} \mid \text{pexpr}' + \text{pexpr}'$$

where $uvar$ is any universally quantified integer variable,

and $evar$ is any existentially quantified or free integer variable.

- $G[\bar{i}]$ **value constraint:**

Any occurrence of a quantified index variable i must be as a read into an array, $a[i]$, for array term a . Array reads may not be nested; e.g., $a[b[i]]$ is not allowed.

Array property fragment of $T_A^{\mathbb{Z}}$ consists of formulae that are Boolean combinations of quantifier-free $\Sigma_A^{\mathbb{Z}}$ -formulae and array properties.

- Array equality $a = b$ in T_A :

- Array equality $a = b$ in T_A :

$$\forall i. a[i] = b[i]$$

- Bounded array equality $\text{beq}(a, b, \ell, u)$ in $T_A^{\mathbb{Z}}$:

- Array equality $a = b$ in T_A :

$$\forall i. a[i] = b[i]$$

- Bounded array equality $\text{beq}(a, b, \ell, u)$ in $T_A^{\mathbb{Z}}$:

$$\forall i. \ell \leq i \leq u \rightarrow a[i] = b[i]$$

- Universal properties $F[x]$ in T_A :

- Array equality $a = b$ in T_A :

$$\forall i. a[i] = b[i]$$

- Bounded array equality $\text{beq}(a, b, \ell, u)$ in $T_A^{\mathbb{Z}}$:

$$\forall i. \ell \leq i \leq u \rightarrow a[i] = b[i]$$

- Universal properties $F[x]$ in T_A :

$$\forall i. F[a[i]]$$

- Bounded universal properties $F[x]$ in $T_A^{\mathbb{Z}}$:

- Array equality $a = b$ in T_A :

$$\forall i. a[i] = b[i]$$

- Bounded array equality $\text{beq}(a, b, \ell, u)$ in $T_A^{\mathbb{Z}}$:

$$\forall i. \ell \leq i \leq u \rightarrow a[i] = b[i]$$

- Universal properties $F[x]$ in T_A :

$$\forall i. F[a[i]]$$

- Bounded universal properties $F[x]$ in $T_A^{\mathbb{Z}}$:

$$\forall i. \ell \leq i \leq u \rightarrow F[a[i]]$$

- Bounded and unbounded sorted arrays $\text{sorted}(a, \ell, u)$ in $T_A^{\mathbb{Z}} \cup T_{\mathbb{Z}}$ or $T_A^{\mathbb{Z}} \cup T_{\mathbb{Q}}$:

- Array equality $a = b$ in T_A :

$$\forall i. a[i] = b[i]$$

- Bounded array equality $\text{beq}(a, b, \ell, u)$ in $T_A^{\mathbb{Z}}$:

$$\forall i. \ell \leq i \leq u \rightarrow a[i] = b[i]$$

- Universal properties $F[x]$ in T_A :

$$\forall i. F[a[i]]$$

- Bounded universal properties $F[x]$ in $T_A^{\mathbb{Z}}$:

$$\forall i. \ell \leq i \leq u \rightarrow F[a[i]]$$

- Bounded and unbounded sorted arrays $\text{sorted}(a, \ell, u)$ in $T_A^{\mathbb{Z}} \cup T_{\mathbb{Z}}$ or $T_A^{\mathbb{Z}} \cup T_{\mathbb{Q}}$:

$$\forall i, j. \ell \leq i \leq j \leq u \rightarrow a[i] \leq a[j]$$

- Partitioned arrays $\text{partitioned}(a, \ell_1, u_1, \ell_2, u_2)$ in $T_A^{\mathbb{Z}} \cup T_{\mathbb{Z}}$ or $T_A^{\mathbb{Z}} \cup T_{\mathbb{Q}}$:

- Array equality $a = b$ in T_A :

$$\forall i. a[i] = b[i]$$

- Bounded array equality $\text{beq}(a, b, \ell, u)$ in $T_A^{\mathbb{Z}}$:

$$\forall i. \ell \leq i \leq u \rightarrow a[i] = b[i]$$

- Universal properties $F[x]$ in T_A :

$$\forall i. F[a[i]]$$

- Bounded universal properties $F[x]$ in $T_A^{\mathbb{Z}}$:

$$\forall i. \ell \leq i \leq u \rightarrow F[a[i]]$$

- Bounded and unbounded sorted arrays $\text{sorted}(a, \ell, u)$ in $T_A^{\mathbb{Z}} \cup T_{\mathbb{Z}}$ or $T_A^{\mathbb{Z}} \cup T_{\mathbb{Q}}$:

$$\forall i, j. \ell \leq i \leq j \leq u \rightarrow a[i] \leq a[j]$$

- Partitioned arrays $\text{partitioned}(a, \ell_1, u_1, \ell_2, u_2)$ in $T_A^{\mathbb{Z}} \cup T_{\mathbb{Z}}$ or $T_A^{\mathbb{Z}} \cup T_{\mathbb{Q}}$:

The Decision Procedure (Step 1–2)

The idea again is to reduce universal quantification to finite conjunction.
Given F from the array property fragment of $T_A^{\mathbb{Z}}$, decide its $T_A^{\mathbb{Z}}$ -satisfiability as follows:

The idea again is to reduce universal quantification to finite conjunction.
Given F from the array property fragment of $T_A^{\mathbb{Z}}$, decide its $T_A^{\mathbb{Z}}$ -satisfiability as follows:

Step 1

Put F in NNF.

The idea again is to reduce universal quantification to finite conjunction. Given F from the array property fragment of $T_A^{\mathbb{Z}}$, decide its $T_A^{\mathbb{Z}}$ -satisfiability as follows:

Step 1

Put F in NNF.

Step 2

Apply the following rule exhaustively to remove writes:

$$\frac{F[a\langle i \triangleleft e \rangle]}{F[a'] \wedge a'[i] = e \wedge (\forall j. j \neq i \rightarrow a[j] = a'[j])} \text{ for fresh } a' \quad (\text{write})$$

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

$$\forall j. j \leq i - 1 \vee i + 1 \leq j \rightarrow a[j] = a'[j].$$

Step 3

Apply the following rule exhaustively to remove existential quantification:

$$\frac{F[\exists \bar{i}. G[\bar{i}]]}{F[G[\bar{j}]]} \text{ for fresh } \bar{j} \quad (\text{exists})$$

Existential quantification can arise during Step 1 if the given formula has a negated array property.

Step 4

From the output of Step 3, F_3 , construct the index set \mathcal{I} :

$$\mathcal{I} = \cup \left\{ \begin{array}{l} \{t : \cdot[t] \in F_3 \text{ such that } t \text{ is not a universally quantified variable}\} \\ \{t : t \text{ occurs as a pexpr in the parsing of index guards}\} \end{array} \right\}$$

If $\mathcal{I} = \emptyset$, then let $\mathcal{I} = \{0\}$. The index set contains all relevant symbolic indices that occur in F_3 .

Step 5

Apply the following rule exhaustively to remove universal quantification:

$$\frac{H[\forall \vec{i}. F[\vec{i}] \rightarrow G[\vec{i}]]}{H \left[\bigwedge_{\vec{i} \in \mathcal{I}^n} (F[\vec{i}] \rightarrow G[\vec{i}]) \right]} \quad (\text{forall})$$

n is the size of the block of universal quantifiers over \vec{i} .

Step 6

F_5 is quantifier-free in the combination theory $T_A \cup T_{\mathbb{Z}}$. Decide the $(T_A \cup T_{\mathbb{Z}})$ -satisfiability of the resulting formula.

$\Sigma_{\mathbb{A}}^{\mathbb{Z}}$ -formula:

$$F : \begin{aligned} & (\forall i. \ell \leq i \leq u \rightarrow a[i] = b[i]) \\ & \wedge \neg(\forall i. \ell \leq i \leq u + 1 \rightarrow a\langle u + 1 \triangleleft b[u + 1]\rangle[i] = b[i]) \end{aligned}$$

$\Sigma_{\mathbb{A}}^{\mathbb{Z}}$ -formula:

$$F : \begin{aligned} & (\forall i. \ell \leq i \leq u \rightarrow a[i] = b[i]) \\ & \wedge \neg(\forall i. \ell \leq i \leq u + 1 \rightarrow a\langle u + 1 \triangleleft b[u + 1]\rangle[i] = b[i]) \end{aligned}$$

In NNF, we have

$$F_1 : \begin{aligned} & (\forall i. \ell \leq i \leq u \rightarrow a[i] = b[i]) \\ & \wedge (\exists i. \ell \leq i \leq u + 1 \wedge a\langle u + 1 \triangleleft b[u + 1]\rangle[i] \neq b[i]) \end{aligned}$$

Step 2 produces

$$F_2 : \begin{aligned} & (\forall i. \ell \leq i \leq u \rightarrow a[i] = b[i]) \\ & \wedge (\exists i. \ell \leq i \leq u + 1 \wedge a'[i] \neq b[i]) \\ & \wedge a'[u + 1] = b[u + 1] \\ & \wedge (\forall j. j \leq u + 1 - 1 \vee u + 1 + 1 \leq j \rightarrow a[j] = a'[j]) \end{aligned}$$

Step 3 removes the existential quantifier by introducing a fresh constant k :

$$F_3 : \begin{aligned} & (\forall i. \ell \leq i \leq u \rightarrow a[i] = b[i]) \\ & \wedge \ell \leq k \leq u + 1 \wedge a'[k] \neq b[k] \\ & \wedge a'[u + 1] = b[u + 1] \\ & \wedge (\forall j. j \leq u + 1 - 1 \vee u + 1 + 1 \leq j \rightarrow a[j] = a'[j]) \end{aligned}$$

Simplifying,

$$F'_3 : \begin{aligned} & (\forall i. \ell \leq i \leq u \rightarrow a[i] = b[i]) \\ & \wedge \ell \leq k \leq u + 1 \wedge a'[k] \neq b[k] \\ & \wedge a'[u + 1] = b[u + 1] \\ & \wedge (\forall j. j \leq u \vee u + 2 \leq j \rightarrow a[j] = a'[j]) \end{aligned}$$

The index set is

$$\mathcal{I} = \{k, u + 1\} \cup \{\ell, u, u + 2\},$$

which includes the read terms k and $u + 1$ and the terms ℓ , u , and $u + 2$ that occur as pexprs in the index guards.

Step 5 rewrites universal quantification to finite conjunction over this set:

$$F_5 : \begin{aligned} & \bigwedge_{i \in \mathcal{I}} (\ell \leq i \leq u \rightarrow a[i] = b[i]) \\ & \wedge \ell \leq k \leq u + 1 \wedge a'[k] \neq b[k] \\ & \wedge a'[u + 1] = b[u + 1] \\ & \wedge \bigwedge_{j \in \mathcal{I}} (j \leq u \vee u + 2 \leq j \rightarrow a[j] = a'[j]) \end{aligned}$$

Expanding the conjunctions according to the index set \mathcal{I} and simplifying according to trivially true or false antecedents (e.g., $\ell \leq u + 1 \leq u$ simplifies to \perp , while $u \leq u \vee u + 2 \leq u$ simplifies to \top) produces:

$$(\ell \leq k \leq u \rightarrow a[k] = b[k]) \quad (1)$$

$$\wedge (\ell \leq u \rightarrow a[\ell] = b[\ell] \wedge a[u] = b[u]) \quad (2)$$

$$\wedge \ell \leq k \leq u + 1 \quad (3)$$

$$F'_5 : \wedge a'[k] \neq b[k] \quad (4)$$

$$\wedge a'[u + 1] = b[u + 1] \quad (5)$$

$$\wedge (k \leq u \vee u + 2 \leq k \rightarrow a[k] = a'[k]) \quad (6)$$

$$\wedge (\ell \leq u \vee u + 2 \leq \ell \rightarrow a[\ell] = a'[\ell]) \quad (7)$$

$$\wedge a[u] = a'[u] \wedge a[u + 2] = a'[u + 2] \quad (8)$$

$(T_A \cup T_{\mathbb{Z}})$ -unsatisfiability of this quantifier-free $(\Sigma_A \cup \Sigma_{\mathbb{Z}})$ -formula can be decided using the techniques of Combination of Theories.

Informally, $\ell \leq k \leq u + 1$ (3)

- If $k \in [\ell, u]$ then $a[k] = b[k]$ (1). Since $k \leq u$ then $a[k] = a'[k]$ (6), contradicting $a'[k] \neq b[k]$ (4).
- if $k = u + 1$, $a'[k] \neq b[k] = b[u + 1] = a'[u + 1] = a'[k]$ by (4) and (5), a contradiction.

Hence, F is $T_A^{\mathbb{Z}}$ -unsatisfiable.

Theorem

Consider a $\Sigma_A^{\mathbb{Z}} \cup \Sigma$ -formula F from the array property fragment of $T_A^{\mathbb{Z}} \cup T$. The output F_5 of Step 5 of the algorithm is $T_A^{\mathbb{Z}} \cup T$ -equisatisfiable to F .

Proof: The proof proceeds using the same strategy as for T_A .
It is easy to see that steps 1–3 do not change the satisfiability of formula.

Proof: The proof proceeds using the same strategy as for T_A .
It is easy to see that steps 1–3 do not change the satisfiability of formula.
For step 4–5 we need to show:

- (1) $H[\forall \bar{i}. (F[\bar{i}] \rightarrow G[\bar{i}])] is satisfiable$
iff.
- (2) $H[\bigwedge_{\bar{i} \in \mathcal{I}^n} (F[\bar{i}] \rightarrow G[\bar{i}])] is satisfiable.$

Proof: The proof proceeds using the same strategy as for T_A .
It is easy to see that steps 1–3 do not change the satisfiability of formula.
For step 4–5 we need to show:

- (1) $H[\forall \bar{i}. (F[\bar{i}] \rightarrow G[\bar{i}])] is satisfiable$
iff.
- (2) $H[\bigwedge_{\bar{i} \in \mathcal{I}^n} (F[\bar{i}] \rightarrow G[\bar{i}])] is satisfiable.$

\Rightarrow : Obviously formula (1) implies formula (2).

If the formula (2) holds in some interpretation $I = (D_I, \alpha_I)$, we construct an interpretation $J = (D_J, \alpha_J)$

If the formula (2) holds in some interpretation $I = (D_I, \alpha_I)$, we construct an interpretation $J = (D_J, \alpha_J)$ with $D_J := D_I$ and

$$\text{proj}_{\mathcal{I}}(j) = \begin{cases} \max\{\alpha_I[i] \mid i \in \mathcal{I} \wedge \alpha_I[i] \leq \alpha_I[j]\} & \text{if for some } i \in \mathcal{I}: \\ & \alpha_I[i] \leq \alpha_I[j] \\ \min\{\alpha_I[i] \mid i \in \mathcal{I} \wedge \alpha_I[i] \geq \alpha_I[j]\} & \text{otherwise} \end{cases}$$

$$\alpha_J[a[j]] = \alpha_I[a[\text{proj}_{\mathcal{I}}(j)]]$$

$$\alpha_J[x] = \alpha_I[x] \text{ for every non-array variable and constant}$$

J interprets the symbols occurring in formula (2) in the same way as I . Therefore, (2) holds in J .

If the formula (2) holds in some interpretation $I = (D_I, \alpha_I)$, we construct an interpretation $J = (D_J, \alpha_J)$ with $D_J := D_I$ and

$$\text{proj}_{\mathcal{I}}(j) = \begin{cases} \max\{\alpha_I[i] \mid i \in \mathcal{I} \wedge \alpha_I[i] \leq \alpha_I[j]\} & \text{if for some } i \in \mathcal{I}: \\ & \alpha_I[i] \leq \alpha_I[j] \\ \min\{\alpha_I[i] \mid i \in \mathcal{I} \wedge \alpha_I[i] \geq \alpha_I[j]\} & \text{otherwise} \end{cases}$$

$$\alpha_J[a[j]] = \alpha_I[a[\text{proj}_{\mathcal{I}}(j)]]$$

$$\alpha_J[x] = \alpha_I[x] \text{ for every non-array variable and constant}$$

J interprets the symbols occurring in formula (2) in the same way as I . Therefore, (2) holds in J .

To prove that formula (1) holds in J , it suffices to show:

$$J \models \bigwedge_{\vec{i} \in \mathcal{I}^n} (F[\vec{i}] \rightarrow G[\vec{i}]) \text{ implies } J \models \forall \vec{i}. (F[\vec{i}] \rightarrow G[\vec{i}])$$

Assume $J \models \bigwedge_{\vec{i} \in \mathcal{I}^n} (F[\vec{i}] \rightarrow G[\vec{i}])$. Show:

$$F[\vec{i}] \rightarrow F[\text{proj}_{\mathcal{I}}(\vec{i})] \rightarrow G[\text{proj}_{\mathcal{I}}(\vec{i})] \rightarrow G[\vec{i}]$$

Assume $J \models \bigwedge_{\vec{i} \in \mathcal{I}^n} (F[\vec{i}] \rightarrow G[\vec{i}])$. Show:

$$F[\vec{i}] \rightarrow F[\text{proj}_{\mathcal{I}}(\vec{i})] \rightarrow G[\text{proj}_{\mathcal{I}}(\vec{i})] \rightarrow G[\vec{i}]$$

The first implication $F[\vec{i}] \rightarrow F[\text{proj}_{\mathcal{I}}(\vec{i})]$ can be shown by structural induction over F . Base cases:

- $\text{expr}_1 \leq \text{expr}_2$: see exercise.
- $\text{expr}_1 = \text{expr}_2$: follows from first case since it is equivalent to

$$\text{expr}_1 \leq \text{expr}_2 \wedge \text{expr}_2 \leq \text{expr}_1 .$$

The induction step is trivial.

Assume $J \models \bigwedge_{\bar{i} \in \mathcal{I}^n} (F[\bar{i}] \rightarrow G[\bar{i}])$. Show:

$$F[\bar{i}] \rightarrow F[\text{proj}_{\mathcal{I}}(\bar{i})] \rightarrow G[\text{proj}_{\mathcal{I}}(\bar{i})] \rightarrow G[\bar{i}]$$

The first implication $F[\bar{i}] \rightarrow F[\text{proj}_{\mathcal{I}}(\bar{i})]$ can be shown by structural induction over F . Base cases:

- $\text{expr}_1 \leq \text{expr}_2$: see exercise.
- $\text{expr}_1 = \text{expr}_2$: follows from first case since it is equivalent to

$$\text{expr}_1 \leq \text{expr}_2 \wedge \text{expr}_2 \leq \text{expr}_1 .$$

The induction step is trivial.

The second implication $F[\text{proj}_{\mathcal{I}}(\bar{i})] \rightarrow G[\text{proj}_{\mathcal{I}}(\bar{i})]$ holds by assumption.

Assume $J \models \bigwedge_{\vec{i} \in \mathcal{I}^n} (F[\vec{i}] \rightarrow G[\vec{i}])$. Show:

$$F[\vec{i}] \rightarrow F[\text{proj}_{\mathcal{I}}(\vec{i})] \rightarrow G[\text{proj}_{\mathcal{I}}(\vec{i})] \rightarrow G[\vec{i}]$$

The first implication $F[\vec{i}] \rightarrow F[\text{proj}_{\mathcal{I}}(\vec{i})]$ can be shown by structural induction over F . Base cases:

- $\text{expr}_1 \leq \text{expr}_2$: see exercise.
- $\text{expr}_1 = \text{expr}_2$: follows from first case since it is equivalent to

$$\text{expr}_1 \leq \text{expr}_2 \wedge \text{expr}_2 \leq \text{expr}_1.$$

The induction step is trivial.

The second implication $F[\text{proj}_{\mathcal{I}}(\vec{i})] \rightarrow G[\text{proj}_{\mathcal{I}}(\vec{i})]$ holds by assumption.

The third implication $G[\text{proj}_{\mathcal{I}}(\vec{i})] \implies G[\vec{i}]$ holds because G contains variables i only in array reads $a[i]$. By definition of J :

$$\alpha_J[a[i]] = \alpha_J[a[\text{proj}_{\mathcal{I}}(i)]].$$