Automated Reasoning 2018

Lecture 17: Theory of arrays

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2018-10-01

Where are we and where are we going?

We have seen solvers for

- QF_EUF,QF_LRA,QF_IDL,QF_RDL
- Octagon logic

We will see today

- Theory of arrays(QF_AX)

Topic 17.1

Theory of arrays

Theory of arrays

The presence of arrays in programs is ubiquitous.

A solving engine needs to be able to reason over arrays.

Here we present an axiomatization of arrays, which has the following properties.

- arrays are accessible by function symbols $\_[\_]$ and *store*
- $\_[\_]$ and *store* can access an index at a time
- arrays have unbounded length

Commentary: This is a simplified view of arrays. It does not model length of arrays.

Many sorted FOL

FOL as we have defined has no sorts.

We need many sorted FOL to model arrays, indexes, and values.

In many sorted FOL, a model has a domain that is partitioned for values of each sort.

A term takes value only from its own sort.

Understanding *store* and $\_[\_]$

- $\_[\_]$ returns a value stored in an array at an index

$$\_[\_] : Array \times Index \to Value$$

- *store* places a value at an index in an array and returns a modified array

$$store : Array \times Index \times Value \to Array$$

Array, *Index*, and *Elem* are disjoint parts of the domain of a model.

Commentary: Please leave your programmer's intuition behind. Here arrays are not mutable. A write on array does not modifies the array, but produces a modified copy.

Axiom of theory of arrays (with extensionality)

Let $\mathbf{S}_A \triangleq (\{\_[\_]/2, store/3\}, \emptyset)$. Assuming $=$ is part of FOL syntax.

Definition 17.1

Theory of arrays[†] $\mathcal{T}_A$ *is defined by the following three axioms.*

1. $\forall a \forall i \forall v.\ store(a,i,v)[i] = v$
2. $\forall a \forall i \forall j \forall v.\ i \neq j \Rightarrow store(a,i,v)[j] = a[j]$
3. $\forall a, b.\ \exists i.\ (a \neq b \Rightarrow a[i] \neq b[i])$ *(extensionality axiom)*

The theories that replace the 3rd axiom with some other axiom(s) are called non-extensional theory of arrays

[†] McCarthy, J.: Towards a mathematical science of computation. In: IFIP Congress. (1962) 21-28

Commentary: The axiomatization is simple and powerful. Various solvers use the axioms. The extensionality axiom is considered to be the key source of difficulty, since it introduces a fresh symbol during instantiation.

Models for theory of arrays

A model m contains a set of indexes $Index_m$, a set of values $Value_m$, and a set of arrays $Array_m$. Constants take values from their respective sorts.

Exercise 17.1

Prove $|Array_m| = |Index_m \rightarrow Value_m|$ *for model* m.

Example 17.1

Consider: $a[i] = a[j] \wedge i \neq j$

Consider the following satisfying model of the above formula:
Let $Index_m = \{1, 2\}$ *and* $Value_m = \{3, 8\}$ *and* $Array_m = \{a_1, a_2, a_3, a_4\}$

- $a_1[1]_m = 3, a_1[2]_m = 3,$
- $a_2[1]_m = 8, a_2[2]_m = 8$
- $a_3[1]_m = 3,\ a_3[2]_m = 8,$
- $a_4[1]_m = 8,\ a_4[2]_m = 3$

$store_m(a_1, 1, 3) = a_1$, $store_m(a_1, 1, 8) = a_4$, $store_m(a_4, 2, 8) =$ ___, ...

$i_m = 1,\ j_m = 2,\ a = a_1$

Decidability

Theory of arrays is undecidable.

However, quantifier-free (QF) fragment is decidable and its complexity is NP.

Example : checking sat in theory of arrays

Example 17.2

Consider the following QF_AX formula:
$store(a, i, b[i]) = store(b, i, a[i]) \wedge a \neq b$

Apply axiom 3,
$store(a, i, b[i]) = store(b, i, a[i]) \wedge a[j] \neq b[j]$

Due to congruence,
$store(a, i, b[i])[j] = store(b, i, a[i])[j] \wedge a[j] \neq b[j]$

case $i = j$: Due to the axiom 1,
$b[i] = a[i] \wedge a[j] \neq b[j] \leftarrow$ *Contradiction.*

case $i \neq j$: Due to the axiom 2,
$b[j] = a[j] \wedge a[j] \neq b[j] \leftarrow$ *Contradiction.*

Therefore, the formula is unsat.

Exercise

Exercise 17.2

Show if the following formulas are sat or unsat

1. $a = b \wedge a[i] \neq b[i]$
2. $a = b \wedge a[i] \neq b[j]$
3. $store(store(a, j, y), i, x) \neq store(store(a, i, x), j, y) \wedge i \neq j$

Topic 17.2

A theory solver for $\mathcal{T}_A$

Theory solver for arrays

The key issues of checking sat of conjunction of $\mathcal{T}_A$ literals are

- finding the set of the indices of interest
- finding the witness of disequality

Array solvers lazily/eagerly add instantiations of the axioms for relevant indices

Commentary: An eager solver instantiates axioms all possible relevant ways at pre-solving phase and solves using some EUF solver. A lazy solver instantiates on demand. And, there can be a combination of the two approaches.

A policy of axiom instantiations

Here we present the policy used in Z3 to add the instantiations.

- flattening of clauses
- solve flattened clauses using CDCL($\mathcal{T}_{\text{EUF}}$)
- time to time introduce new clauses due to instantiations.

L. Moura, N. Bjorner, Generalized, Efficient Array Decision Procedures. FMCAD09 (section 2-4)

Commentary: Here is another policy used in another solver:
M. Bofill and R. Nieuwenhuis, A Write-Based Solver for SAT Modulo the Theory of Arrays, FMCAD2008

Flattening

The solver maintains a set of definitions and a set of clauses.

$\_[\_]/store$ terms are replaced by a fresh symbol and the definitions record the replacement.

Example 17.3

Consider clauses: $store(store(a, j, y), i, x) \neq store(store(a, i, x), j, y) \wedge i \neq j$

Flattened clauses: $u \neq v \wedge i \neq j$

Definition store:
$u \triangleq store(u', i, x), u' \triangleq store(a, j, y),\ v \triangleq store(v', j, y), v' \triangleq store(a, i, y)$

Exercise 17.3

Translate the following in flattened clauses:
$store(a, i, b[i]) = store(b, i, a[i]) \wedge a \neq b$

Commentary: The example is not chosen well. It has only unit clauses.

CDCL($\mathcal{T}_{\mathsf{EUF}}$) on flattened clauses

CDCL($\mathcal{T}_{\mathsf{EUF}}$) is iteratively applied on the flattened clauses as follows.

1. Run CDCL($\mathcal{T}_{\mathsf{EUF}}$) on the flattened clauses
2. If no assignment found then return unsat. Otherwise, $\mathcal{T}_{\mathsf{EUF}}$ has found equivalences that are compatible with the current clauses
3. Add relevant instantiations of array axioms due to the discovery of new equivalent classes
4. If no new instantiations added then return sat. Otherwise, goto 1

Commentary: CDCL assigns truth value to atoms and EUF solver translates the truth values into equivalence classes.

Relevant axiom instantiation

The following rules add new instantiations of the axioms in the clause set.
The instantiated clauses are flattened and added in CDCL($\mathcal{T}_{\text{EUF}}$).

$$\frac{a \triangleq store(b, i, v)}{a[i] = v}$$

$\sim$ denotes the discovered equivalences in $\mathcal{T}_{\text{EUF}}$

$$\frac{a \triangleq store(b, i, \_) \quad \_ \triangleq a'[j] \quad a \sim a'}{i = j \vee a[j] = b[j]}$$

$$\frac{a \triangleq store(b, i, \_) \quad \_ \triangleq b'[j] \quad b \sim b'}{i = j \vee a[j] = b[j]}$$

$$\frac{a : Array \quad b : Array}{a = b \vee a[k_{a,b}] \neq b[k_{a,b}]}$$

Reading the above rules:
In the 2nd rule, if a is defined as above, a and a' are equivalent under current assignment, and a' is accessed at j then we instantiate the 2nd axiom involving indexes i and j, and arrays a and b

Soundness and completeness

The solver is sound because it only introduces the instantiations of axioms.

Theorem 17.1

The solver is complete

Proof sketch.

We need to show that only finite and all relevant instantiations are added.
If no conflict is discovered after saturation then we can construct a model. $\square$

Exercise 17.4

Fill the details in the above proof

- *Only finite instantiations are added*
- *Construct a model at saturation*

Optimizations

We may reduce the number of instantiations that are needed to be complete.

Here, we discuss three such optimizations.

- Instantiations for equivalent symbols are redundant
- Instantiate extensionality only if a disequality is discovered in EUF
- Instantiate 2nd axiom only if the concerning index is involved in the final model construction

Redundant Instantiations

If EUF solver has proven $i \sim i'$, $j \sim j'$, $a \sim a'$, and $b \sim b'$ then

$$i = j \vee a[j] = b[j]$$
$$\text{and}$$
$$i' = j' \vee a'[j'] = b'[j']$$

are mutually redundant instantiations.

We need to instantiate only one of the two.

Similarly, if EUF solver has proven $a \sim a'$, and $b \sim b'$ then

$$a = b \vee a[k_{a,b}] \neq b[k_{a,b}]$$
$$\text{and}$$
$$a' = b' \vee a'[k_{a',b'}] \neq b'[k_{a',b'}]$$

are mutually redundant instantiations.

Extensionality axiom only for disequalities

We only need to produce evidence that two arrays are disequal only if EUF finds such disequality

$$\frac{a : Array \quad b : Array \quad a \not\sim b}{a = b \vee a[k_{a,b}] \neq b[k_{a,b}]}$$

Restricted instantiation of the 2nd axiom

Definition 17.2

$b \in$ nonlinear if

1. $b \triangleq store(\_, \_, \_)$ *and there is another* b' *such that* $b \sim b'$ *and* $b' \triangleq store(\_, \_, \_)$
2. $a \triangleq store(b, \_, \_)$ *and* $a \in nonlinear$
3. $a \sim b$ *and* $a \in nonlinear$

We restrict the third instantiation rule as follows.

$$\frac{a \triangleq store(b, i, v) \quad w \triangleq b'[j] \quad b \sim b' \quad b \in nonlinear}{i = j \vee a[j] = b[j]}$$

Theorem 17.2

If $b \notin$ nonlinear, value of index j has no effect in the model construction of a.

Topic 17.3

A decidable fragment of quantified arrays

Decidable fragments

Definition 17.3

An undecidable class often has non-obvious sub-classes that are decidable, which are called decidable fragments.

For example, QF_AX is a decidable fragment of AX.

Finding decidable fragments of various logics is an active area of research.

Now we will present a decidable fragment of AX called "array properties", which allows some restricted form of quantifiers.

For ease of introducing core ideas, the fragment presented here is smaller than the original proposal in

Aaron R. Bradley, Zohar Manna, Henny B. Sipma: What's Decidable About Arrays? VMCAI 2006

Some notation

For formulas/terms F and G, we say

- $G \in F$ if G occurs in F and
- G is QF in F if $G \in F$ and no variable in $FV(G)$ is universally quantified in F

Array properties

Array properties fragment puts the following restrictions.

- *Index* $= \mathbb{Z}$.
- *Value* sort is part of some decidable theory $\mathcal{T}_v$.
- the formulas in the fragment are conjunctions of array properties that are defined in the next slide.

Array property

Definition 17.4

An *array property* *is a formula that has the following shape.*

$$\forall \vec{i}.\ (F_I(\vec{i}) \Rightarrow F_V(\vec{i}))$$

- *there are other array, index, and value variables that are free*
- $F_I(\vec{i}) \in guard$

$$guard ::= guard \vee guard \mid guard \wedge guard \mid exp \leq exp \mid exp = exp$$

$$exp ::= i \mid pexp \qquad i \in \vec{i}$$

$$pxpr ::= \mathbb{Z} \mid \mathbb{Z}j \mid pexp + pexp \qquad j \notin \vec{i}$$

- $F_V(\vec{i})$ *is a QF formula from* $\mathcal{T}_v$. *If* $i \in \vec{i}$ *and* $i \in F_V$ *then* i *only occurs as parameter of some array read and nested accesses are disallowed.*

Example: array properties

Example 17.4

Are the following formulas array properties?

- $\forall i.\ a[i] = b[i]$ ✓
- $\forall i.\ a[i] = b[i+1]$ ✗
- $\forall i.\ a[i] = b[j+1]$ ✓
- $\forall i, j.\ i \leq j \Rightarrow a[i] \leq a[j]$ ✓
- $\forall i, j.\ i \leq j \Rightarrow a[a[i]] \leq a[j]$ ✗
- $\forall i, j.\ i \leq k+1 \Rightarrow a[i] \leq a[j]$ ✓
- $\forall i, j.\ \neg(i \leq k+1) \Rightarrow a[i] \leq a[j]$ ✗
- $\forall i, j.\ i \leq j+1 \Rightarrow a[i] \leq a[j]$ ✗

Decision procedure: notation

Definition 17.5

For an array property F, read Set R_F is the set

$$R_F \triangleq \{t | \_[t] \in F \wedge t \text{ is QF in } F\}$$

Definition 17.6

For an array property F, bound Set B_F is the set

$$B_F \triangleq \{t | (\forall \vec{i}.\ F_I(\vec{i}) \Rightarrow F_V(\vec{i})) \in F \wedge t \bowtie i \in F_I \wedge t \text{ is QF in } F\}$$

where $\bowtie \in \{\leq, =, \geq\}$.

Definition 17.7

For an array property F, index set $I_F = B_F \cup R_F$

Decision procedure for array properties

1. Replace writes by 1st and 2nd axioms of arrays

$$F[store(a,t,v)] \quad \rightsquigarrow \quad F[b] \wedge b[t] = v \wedge \forall i.\ (i \neq t \Rightarrow a[i] = b[i])$$

Remains in array property fragment

We will call the transformed formula F'.

2. Replace universal quantifiers by index sets

$$F'[(\forall \vec{i}.\ F_I(\vec{i}) \Rightarrow F_V(\vec{i}))] \quad \rightsquigarrow \quad F'[\bigwedge_{\vec{t} \in I_{F'}^{len(\vec{i})}} (F_I(\vec{t}) \Rightarrow F_V(\vec{t}))]$$

no universal quantifiers

We will call the transformed formula F''.

3. F'' is in QF fragment of $\mathcal{T}_A + \mathcal{T}_{\mathbb{Z}} + \mathcal{T}_v$. We solve it using a decision procedure for the theory combination. We have not covered theory combination yet!!

Exercise 17.5

Extend this procedure for the boolean combinations of array properties.

Example: solving array properties

Example 17.5

Consider:

$x < y \wedge k+1 < \ell \wedge b = store(a, \ell, x) \wedge c = store(a, k, y) \wedge$
$\forall i, j. \, (k \leq i \leq j \leq \ell \Rightarrow b[i] \leq b[j]) \wedge \forall i, j. \, (k \leq i \leq j \leq \ell \Rightarrow c[i] \leq c[j])$

After removing stores:

$x < y \wedge k+1 < \ell \wedge$
$b[\ell] = x \wedge \forall i. \, (\ell + 1 \leq i \vee i \leq \ell - 1) \Rightarrow b[i] = a[i] \wedge$
$c[k] = y \wedge \forall i. \, (k + 1 \leq i \vee i \leq k - 1) \Rightarrow c[i] = a[i] \wedge$
$\forall i, j. \, (k \leq i \leq j \leq \ell \Rightarrow b[i] \leq b[j]) \wedge$
$\forall i, j. \, (k \leq i \leq j \leq \ell \Rightarrow c[i] \leq c[j])$

Exercise 17.6

The index set for the above formula includes expression $k-1$. Instantiate the last quantified formula for term $k-1$

Commentary: Removing stores may introduce new arrays. The above example is simple enough and we need not introduce new arrays.

Example: solving array properties(contd.)

Index set $I = \{k-1, k, k+1, \ell-1, \ell, \ell+1\}$
We instantiate each universal quantifier 6 times.
Therefore, 84 quantifier-free clauses are added.

Let us consider only the following instantiations of the quantifiers:
$x < y \wedge k+1 < \ell \wedge b[\ell] = x \wedge c[k] = y \wedge$
$(\ell + 1 \leq k+1 \vee k+1 \leq \ell - 1) \Rightarrow b[k+1] = a[k+1] \wedge$
$(k+1 \leq k+1 \vee k+1 \leq k-1) \Rightarrow c[k+1] = a[k+1] \wedge$
$k \leq k \leq k+1 \leq \ell \Rightarrow c[k] \leq c[k+1] \wedge$
$k \leq k+1 \leq \ell \leq \ell \Rightarrow b[k+1] \leq b[\ell] \wedge$ ....(many more)

Since all the above mentioned guards are true,
$x < y = c[k] \leq c[k+1] = a[k+1] = b[k+1] \leq b[\ell] = x$
Contradiction.

Why are finite instantiations sufficient for checking sat of $\forall$ quantifiers?

Correctness

Theorem 17.3

If F is sat iff F' is sat

Proof.

This step only explicates theory axioms. Trivially holds. □

Theorem 17.4

If F' is sat iff F'' is sat

Proof.

Since F'' is finite instantiations of F', if F'' is unsat then F' is unsat.

Now we show that if $m'' \models F''$ then we can construct a model m' for F'.

Let $I_{F'} = \{t^1, ..., t^\ell\}$. Wlog, we assume $t^1_{m''} \leq ... \leq t^\ell_{m''}$. ...

Correctness (contd.)

Proof(contd.)

Observation:

m'' assigns values to all non-array variables of F'.

In arrays, m'' assigns values only at indexes $I_{F'}$.(why?)

Constructing m':

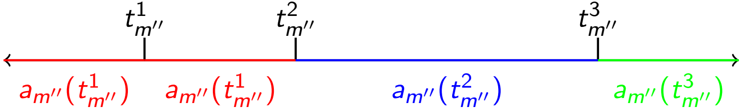
We copy assignment of non-array variables from m'' to m'.

Let a be an array appearing in F'. We construct $a_{m'}$ as follows.

For each $j \in \mathbb{Z}$,

$$a_{m'}(j) \triangleq a_{m''}(t^k_{m''})$$

where $k = \max\{1\} \cup \{j | t^j_{m''} \leq j\}$.



...

Correctness (contd.)

Proof(contd.)

claim: $m' \models F'$

Consider $\forall \vec{i}.\ F_I(\vec{i}) \Rightarrow F_V(\vec{i}) \in F'$.

$\vec{v}$ is inside a hyper-cube defined by corners $\vec{u}$ and $\vec{w}$

Let $\vec{v} \in \mathbb{Z}^n$, where $n = len(i)$.

Choose $\vec{u} \triangleq (t^{j_1}_{m''}, .., t^{j_n}_{m''})$ and $\vec{w} \triangleq (t^{j_1+1}_{m''}, .., t^{j_n+1}_{m''})$ such that $\vec{u} \leq \vec{v} < \vec{w}$.

Since $m'' \models F''$, $m''[\vec{i} \to \vec{u}] \models F_I(\vec{i}) \Rightarrow F_V(\vec{i})$.

Case $m''[\vec{i} \to \vec{u}] \models F_I(\vec{i})$:

Therefore, $m''[\vec{i} \to \vec{u}] \models F_V(\vec{i})$.

Therefore, $m''[\vec{i} \to \vec{v}] \models F_V(\vec{i})$.(why?)

Therefore, $m''[\vec{i} \to \vec{v}] \models F_I(\vec{i}) \Rightarrow F_V(\vec{i})$.

...

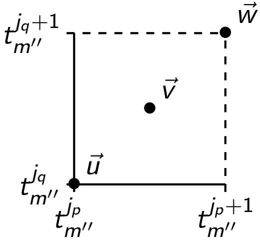
Commentary: This proof divides quantified space into finite parts and chooses a representative such that if it satisfies the formula then all in the partition satisfy the formula.

Correctness (contd.)

Case $m''[\vec{i} \to \vec{u}] \not\models F_I(\vec{i})$:

For $i_p, i_q \in \vec{i}$, there are three kinds of atoms in F_I.

- $i_p \leq t^r$
- $t^r \leq i_p$
- $i_p \leq i_q$



If an atom is false in $m''[\vec{i} \to \vec{u}]$ then it is false in $m''[\vec{i} \to \vec{v}]$.(why?)
Since F_I is positive boolean combination of the atoms, $m''[\vec{i} \to \vec{v}] \not\models F_I(\vec{i})$.
Therefore, $m''[\vec{i} \to \vec{v}] \models F_I(\vec{i}) \Rightarrow F_V(\vec{i})$.

Exercise 17.7

Some ranges of $\vec{i}$ are missing in the above argument. Complete the proof.

Commentary: Note that if any atom is true at $\vec{u}$ then it can become false at $\vec{v}$.

Topic 17.4

Problems

Prove sorting

Exercise 17.8

Give a model that satisfies the following formula:

$$\forall i, j.\ (i < j \Rightarrow a[i] = b[i]) \Rightarrow \forall i.\ a[i] = b[i+1]$$

Can we also prove?

$$\forall i.\ a[i] = b[i+1] \Rightarrow \forall i, j.\ (i < j \Rightarrow a[i] = b[i])$$

Model generation

Exercise 17.9

Give a model that satisfies the following formula:

$$store(store(b, i_0, b[i_1]), i_1, b[i_0]) = store(store(b, i_1, b[i_1]), i_1, b[i_1])$$

Run Z3

Exercise 17.10

Run Z3 in proof producing mode on the following example:

$$store(store(a, j, y), i, x) \neq store(store(a, i, x), j, y) \wedge i \neq j$$

explain the proof of unsatisfiability produced by Z3.

Note that: In smt-lib format select denotes $\_[\_]$.

End of Lecture 17