Arrays, Maps, and Vectors With Abstract Domain[1]

Maria Paola Bonacina

Dipartimento di Informatica
Università degli Studi di Verona
Verona, Italy, EU

Dagstuhl Seminar # 26091: "Revisiting the Foundations of Deduction in a New World"
Schloß Dagstuhl, near Wadern, Germany, EU, 24 February 2026

(Subsumes talks given at the Workshops
"Theory and Practice of SAT and Combinatorial Solving"
Banff International Research Station, Banff, Alberta, Canada, 15 January 2026,
and "Automated Reasoning and Proof Logging"
Final Symposium of COST Action CA20111 "European Research Network on Formal Proofs"
Paris, France, EU, 12 September 2025)

[1] Joint work with S. Graham-Lengrand and N. Shankar

Arrays

- Data structure with direct access to values via indices
- Theory of arrays:
  - Sorts: indices, values, arrays
  - Basic operations: read/write or select/store
  - Select-over-store axioms:
    $\forall a,\ v,\ i.\ \mathrm{select}(\mathrm{store}(a, i, v), i) \simeq v$
    $\forall a,\ v,\ i,\ j.\ i \not\simeq j \rightarrow \mathrm{select}(\mathrm{store}(a, i, v), j) \simeq \mathrm{select}(a, j)$
  - Extensionality axiom:
    $\forall a,\ b.\ (\forall i.\ \mathrm{select}(a, i) \simeq \mathrm{select}(b, i)) \rightarrow a \simeq b$
- Not decidable, but the quantifier-free fragment is
- Considered useful to reason about computer memory (e.g., the heap)

Arrays: finite of infinite?

In programming languages:

- Integer-indexed arrays
- Finite: indices in the interval $[0, n-1]$, length n
- A store within bounds works, error/exception otherwise
- The computer memory is finite

In the theory of arrays:

- All arrays have the same length:
  the cardinality of the sort of indices
- If integer-indexed: infinite arrays
- No distinction btw in-bounds and out-of-bounds store

How about adding a length function len?

- Maps every array to its length: $\mathrm{len}(a) \simeq n$
- Revised axiom of extensionality for integer-index arrays:
  $\forall a,\ b.\ [\ \mathrm{len}(a)\ \simeq\ \mathrm{len}(b) \wedge$
  $(\forall i.\ 0 \leq i < \mathrm{len}(a) \rightarrow \mathrm{select}(a, i)\ \simeq\ \mathrm{select}(b, i))\ ]\ \rightarrow\ a\ \simeq\ b$
- Arrays and integers no longer disjoint
- Length is a bridging function and extensionality becomes a bridging axiom
- Most methods for reasoning in theory unions require disjoint theories (equality is the only shared symbol)

How about allowing quantifiers?

- Array property fragment
- Allows limited usage of $\forall$ over index variables, preserving decidability
- Bounded array equality: $beq(a, b, l, u)$ iff $\forall i.\ l \leq i \leq u \ \rightarrow\ \mathsf{select}(a, i) \ \simeq\ \mathsf{select}(b, i)$
- The theory of arrays and its limitations remain unchanged
- Efficient quantifier reasoning may be challenging

Recurring to other theories? Sequences

- Using finite sequences with integer indices $[0, |x|)$ to model finite arrays
- access/update for select/store
- Extensionality axiom as in arrays with length
- Update axiom: update does not change length
  only an update in $[0, |x|)$ modifies the element
- Conservative extension of the theory of integers
- Decidability of quantifier-free fragment: unknown
  (sound and complete inference system, termination not guaranteed)

How about quantifiers and sequences together?

- The array property fragment with concatenation
- Arrays interpreted as finite integer-indexed sequences
- More expressive than the array property fragment: allows index shifting (e.g., $a[i]$ and $a[i+n]$)
  concatenation can be defined
- Undecidable: halting problem of a two-register machine
- Decision procedure for tangle-free formulas
  same as stratified arrays

Solution: a theory of arrays with abstract domain

- Neither quantifiers nor sequences:
  enrich the theory of arrays with length and admissibility
- Abstract domain of definition of an array
  indices not necessarily integers, nor linearly ordered
- Admissibility defined by another theory:
  flexibility + minimum sharing
- Also maps, and vectors meaning dynamic arrays
- Theory combination method CDSAT extended to
  predicate-sharing theories: sound, terminating, complete
- Decidable quantifier-free fragment:
  from fitting the 3 theories in CDSAT

The theory of arrays with abstract domain: signature

- ArrAD: theory of arrays with abstract domain
- Sorts: indices I, values V, arrays A, lengths L, and $Prop$
- $\mathsf{select}\colon A \times I \to V \qquad \mathsf{store}\colon A \times I \times V \to A \qquad \mathsf{len}\colon A \to L$
- Free admissibility predicate: $\mathsf{Adm}\colon I \times L \to Prop$
  $\mathsf{Adm}(i, l)$: index i is admissible wrt length l
- Abstract domain: definition of Adm
- Concrete domain: set of admissible indices given Adm's definition and the interpretation of I
- Adm is shared with another theory $\mathcal{T}$ that defines it

The theory of arrays with abstract domain: axioms

- Select-over-store axioms:
  - $\forall\ a,\ v,\ i.\ \mathrm{select}(\mathrm{store}(a,i,v),i) \simeq v$ is replaced by
    $\forall\ a,\ v,\ i.\ \mathrm{Adm}(i,\mathrm{len}(a)) \rightarrow \mathrm{select}(\mathrm{store}(a,i,v),i) \simeq v$
    a store at an inadmissible index has no effect
  - $\forall\ a,\ v,\ i,\ j.\ i \not\simeq j \rightarrow \mathrm{select}(\mathrm{store}(a,i,v),j) \simeq \mathrm{select}(a,j)$
- Store does not change length:
  $\forall\ a,\ i,\ v.\ \mathrm{len}(\mathrm{store}(a,i,v)) \simeq \mathrm{len}(a)$
- Extensionality:
  $\forall\ a,\ b.\ [\ \mathrm{len}(a) \simeq \mathrm{len}(b) \wedge$
  $(\forall i.\ \mathrm{Adm}(i,\mathrm{len}(a)) \rightarrow \mathrm{select}(a,i) \simeq \mathrm{select}(b,i))\ ]$
  $\rightarrow a \simeq b$

Example: the most common interpretation

- Let LIA be the theory defining Adm
- Indices and lengths are integers
- $\forall i,\ n.\ \mathsf{Adm}(i, n)\ \leftrightarrow\ 0 \le i < n$
- The set of admissible indices is the interval $[0, n)$
- Under this interpretation extensionality in ArrAD covers extensionality as in
  - Integer-indexed arrays with length
  - The theory of sequences
  - The array property fragment with concatenation

Example: capturing bounded equality

- Let LIA be the theory defining Adm
- Indices are integers, lengths are pairs of integers
- $\forall i,\ l,\ u.\ \mathrm{Adm}(i, (l, u)) \ \leftrightarrow\ l \leq i \leq u$
- The set of admissible indices is the interval $[l, u]$
- Under this interpretation extensionality in ArrAD covers bounded equality as in the array property fragment

Example: length with starting address

- Let $\mathcal{T}$ be the theory defining Adm
- Indices are integers, length is a pair $(addr, n)$:
  - $addr$ (binary number): starting address of the array in memory
  - n (integer): the number of admissible indices
- $\forall i,\ addr,\ n.\ \mathsf{Adm}(i, (addr, n)) \ \leftrightarrow\ 0 \leq i < n$
- The starting address does not affect admissibility
  but it affects array equality
- Extensionality: arrays a and b with same set of admissible indices, same values at all admissible indices, but different starting addresses are different
  (as it is in programming languages)

Example: admissibility as membership

- Let $\mathcal{T}$ be the theory defining Adm
- Indices are elements of a set S, length is a subsets of S
- $\forall i,\ N.\ \mathrm{Adm}(i, N) \ \leftrightarrow\ i \in N$
- The set of admissible indices is the subset $N \subseteq S$
- S does not have to be a set of numbers, nor linearly ordered, nor ordered

A theory of maps with abstract domain

- MapAD: theory of maps with abstract domain
- Store at inadmissible index i makes i admissible:
  $\forall\ a,\ j,\ i,\ v.\ \mathsf{Adm}(j, \mathsf{len}(\mathsf{store}(a, i, v))) \ \leftrightarrow\ (\mathsf{Adm}(j, \mathsf{len}(a)) \vee j \simeq i)$
- Store does not change length if the index is admissible:
  $\forall\ a,\ i,\ v.\ \mathsf{Adm}(i, \mathsf{len}(a)) \ \rightarrow\ \mathsf{len}(\mathsf{store}(a, i, v)) \ \simeq\ \mathsf{len}(a)$
- Select-over-store axioms:
  - Restored: $\forall\ a,\ v,\ i.\ \mathsf{select}(\mathsf{store}(a, i, v), i) \ \simeq\ v$
  - $\forall\ a,\ v,\ i,\ j.\ i \not\simeq j \ \rightarrow\ \mathsf{select}(\mathsf{store}(a, i, v), j) \ \simeq\ \mathsf{select}(a, j)$
- Extensionality unchanged: $\forall a,\ b.\ [\ \mathsf{len}(a) \ \simeq\ \mathsf{len}(b) \ \wedge$
  $(\forall i.\ \mathsf{Adm}(i, \mathsf{len}(a)) \ \rightarrow\ \mathsf{select}(a, i) \ \simeq\ \mathsf{select}(b, i))\ ] \ \rightarrow\ a \ \simeq\ b$

A theory of vectors (dynamic arrays) with abstract domain

- VecAD: theory of vectors with abstract domain
- Store at an inadmissible index i makes i and the indices smaller than i admissible:
  $\forall\ a,\ j,\ i,\ v.\ \mathsf{Adm}(j, \mathsf{len}(\mathsf{store}(a, i, v))) \ \leftrightarrow\ (\mathsf{Adm}(j, \mathsf{len}(a)) \vee j \leq i)$
- Everything else as in MapAD
  except for adding to the signature an ordering $<$ on indices
  (does not have to be linear)
- Dynamic data structures modeled for the first time:
  The theories of sequences do not do it

CDSAT: Conflict-Driven SATisfiability in n theories

- Orchestrates theory modules in a conflict-driven model search
- Generalizes MCSAT to theory combination:
  - Assignments of values to terms: both Boolean and first-order
  - Theory conflict explanation by theory inferences that can generate new terms
- Propositional logic is one of the theories: no hierarchy btw Boolean reasoning and theory reasoning
- Input first-order assignments: Satisfiability Modulo Assignment
- Sound, terminating, and complete for predicate-sharing theories without requiring stable infiniteness

How to fit a component theory in CDSAT?

- A theory module $\mathcal{I}_k$ for theory $\mathcal{T}_k$: an inference system (abstraction of a decision procedure)
- Requirements on a theory module:
  - Soundness (for the soundness of CDSAT)
  - Finite local basis: $\text{basis}_k(X)$ – all the terms that $\mathcal{I}_k$ can generate from set X of input terms
    Used to construct the finite global basis for the theory union (for the termination of CDSAT)
  - Completeness (for the completeness of CDSAT):
    - Leading theory $\mathcal{T}_1$: has all sorts and all shared predicates
    - Leading theory $\mathcal{T}_1$: $\mathcal{I}_1$ is complete
    - All other theories $\mathcal{T}_k$: $\mathcal{I}_k$ is leading-theory complete

Theory modules for ArrAD, MapAD, VecAD

- From axioms to inference rules, e.g.:
  - $a \simeq b \ \vdash \ \mathrm{len}(a) \simeq \mathrm{len}(b)$
  - For ArrAD:
    $b \simeq \mathrm{store}(a, i, v),\ \mathrm{len}(b) \not\simeq \mathrm{len}(a) \ \vdash \ \bot$
  - For MapAD and VecAD:
    $\mathrm{len}(a) \simeq n,\ \mathrm{Adm}(i, n),\ b \simeq \mathrm{store}(a, i, v),\ \mathrm{len}(b) \not\simeq \mathrm{len}(a) \ \vdash \ \bot$
- Some rules generate $\bot$ (conflict detection) others do not: balancing finite local basis design and completeness
- A finite local basis for ArrAD, MapAD, VecAD

Interpretation of arrays with abstract domain

Interpretation of arrays:

- An array: a function from indices to values
- Sort of arrays: an updatable function set X:
  $f \in X$ and g differs from f at finitely many indices: $g \in X$

Interpretation of arrays with abstract domain:

- An array of length n: a function from the set I_n of admissible indices (for n) to values
- Sort of arrays: a collection of updatable function sets $(X_n)_n$
  one for each n in the interpretation of the sort L of lengths

Interpretation of maps with abstract domain

Sort of maps:
the collection $(X_n)_n$ must be incrementally updatable:

- if function f is in X_n and
- g is the function that maps i to v and is identical to f otherwise, then
- $\exists m$ such that $g \in X_m$ and
  - Either $m = n$ (store at an admissible i)
  - Or $I_m = I_n \cup \{i\}$
    (store at an inadmissible index i which is admissible in the resulting map)

Interpretation of vectors with abstract domain

Sort of vectors:
the collection $(X_n)_n$ must be extensibly updatable:

- if function f is in X_n and
- g is the function that maps i to v and is identical to f otherwise, then
- $\exists m$ such that $g \in X_m$ and
  - Either $m = n$ (store at an admissible i)
  - Or $I_m = I_n \cup \{j \mid j \leq i\}$
    (store at an inadmissible index i which is admissible in the resulting vector together with the smaller indices)

Suitability of a leading theory

- A leading theory $\mathcal{T}_1$ is ArrAD-suitable if:
  - $\mathcal{T}_1$ has all the sorts of ArrAD
  - $\mathcal{T}_1$ shares with ArrAD equality and Adm
  - For all $\mathcal{T}_1$-models $\mathcal{M}_1$ there exists a collection of updatable function sets $(X_n)_n$ such that
    - n ranges over all possible values for lengths according to $\mathcal{M}_1$
    - Every $f \in X_n$ is a function from admissible indices to values in the $\mathcal{M}_1$-interpretation of indices, admissibility, and values
    - The cardinality of the sort A of arrays in $\mathcal{M}_1$ is equal to the sum of the cardinalities of the X_n
- MapAD-suitable: use an incrementally updatable collection
- VecAD-suitable: share also $<$ and use an extensibly updatable collection

Leading-theory-completeness

Theorem:
For $\mathcal{T} \in \{\text{ArrAD}, \text{ MapAD}, \text{ VecAD}\}$,
the $\mathcal{T}$-module $\mathcal{I}_{\mathcal{T}}$ is leading-theory-complete
for all $\mathcal{T}$-suitable leading theories $\mathcal{T}_1$

Remark:
Suitability does not restrict combinability

Future work

- Add concatenation?
- See whether decidability of the quantifer-free fragment can be preserved
- Other theories and bridging functions?
  Appropriate shared predicates and CDSAT modules
- QSMA(CDSAT) (for quantified satisfiability)

References

- Conflict-driven satisfiability for theory combination: transition system and completeness.
  JAR 64(3):579–609, 2020 (Conference version at CADE 2017)
- Conflict-driven satisfiability for theory combination: modules, lemmas, and proofs.
  JAR 66(1):43–91, 2022 (Conference version at CPP 2018)
- The CDSAT method for satisfiability modulo theories and assignments: an exposition.
  Proc. CiE-21, LNAI 15764, 1–16, Springer, July 2025.
- CDSAT for predicate-sharing theories: arrays, maps, and vectors with abstract domain.
  Submitted to a journal, 45 pages (Short version at SMT 2022)

Authors: MPB, S. Graham-Lengrand, and N. Shankar

Thank you!