Nondisjoint CDSAT: arrays, maps, and vectors with abstract domain[1]

Maria Paola Bonacina

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

Workshop on Automated Reasoning and Proof Logging
Final Symposium of COST Action CA20111
"European Research Network on Formal Proofs"
(EuroProofNet)
Paris, France, EU

12 September 2025

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

Arrays

- Data structure with direct access to values via indices
- Basic operations: read/write or select/store
- Theory of arrays:
  - Sorts: indices, values, arrays
  - Select-over-store axioms [McCarthy 1963]:
    $\forall a,\ v,\ i.\ \mathrm{select}(\mathrm{store}(a, i, v), i) \simeq v$
    $\forall a,\ v,\ i,\ j.\ i \not\simeq j \rightarrow \mathrm{select}(\mathrm{store}(a, i, v), j) \simeq \mathrm{select}(a, j)$
  - 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

Arrays: finite of infinite?

Arrays in programming languages:

- Integer-indexed
- Finite: indices in the interval $[0, n-1]$, length n
  Ada arrays: indices in the interval $[n, m]$, length $m - n + 1$

Computer memory: finite

Arrays in the theory of arrays:

- Finite or infinite depending on the cardinality of the set used to interpret the sort of indices
- If integer-indexed: infinite arrays

Array property fragment (APF) of the theory of arrays

- Limited usage of $\forall$ over index variables
- Integer-indexed arrays are infinite, but it is possible to define:
  - Bounded array equality: $beq(a, b, l, u)$ iff
    $\forall i.\ l \leq i \leq u \ \rightarrow\ \mathsf{select}(a, i) \ \simeq\ \mathsf{select}(b, i)$
  - Sortedness: $sorted(a, l, u)$ iff
    $\forall i,\ j.\ l \leq i \leq j \leq u \ \rightarrow\ \mathsf{select}(a, i) \ \leq\ \mathsf{select}(a, j)$
    assuming values are integers or rationals
- Decidable: finitely many instances of $\forall$ + decision procedure for the disjoint union of arrays, integers (LIA), theory of values
- Efficient handling of $\forall$ still a challenge in SMT

[Bradley, Manna, Sipma 2006] [Bradley, Manna 2007]

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 theories:
  they share the symbol for the integer ordering
- Similar phenomenon for lists:
  - $\mathrm{len}(\mathrm{nil}) = 0$
  - $\mathrm{len}(\mathrm{cons}(x,y)) = 1 + \mathrm{len}(y)$
  - 0, 1 and $+$ become shared symbols

Length is a bridging function

- Bridging functions: length of arrays, length of lists, size of trees, height of trees
- Bridging axioms:
  - RDS/AFDS (e.g., lists): define the bridging function over the constructors
  - Arrays: extensionality axiom
- Symbols other than equality become shared: non-disjoint theories
- Most methods for reasoning in theory unions require disjoint theories (equality is the only shared symbol)

[Ganzinger, Rueß, Shankar 2004] [Sofronie-Stokkermans 2009]

[Chocron, Fontaine, Ringeissen 2020]

Other theories: strings and sequences

- Strings: sequences of elements from a finite alphabet
  (e.g., [Liang et al. 2014] [Berzish, Ganesh, Zheng 2017])
- Sequences: generalization with generic and possibly infinite element sort
  - Empty sequence, binary associative concatenation: a monoid
  - Unary constructor wrapping single element into sequence
  - Extract function: returns the subsequence btw two positions
  - Access function: returns the element at a given position
  - Length function $|x|$: returns the number of elements in sequence x

  (e.g., [Bjørner et al. 2012] [Jeż et al. 2023])

Theories of finite sequences to model finite arrays

- Theory Seq [Sheng et al. 2023] with integer indices $[0, |x| - 1]$ and countably infinite element sort:
  - Add update function: access/update for select/store
  - Extensionality axiom as in arrays with length
  - Nondisjointness: conservative extension of the theory of integers into Seq
- Theory N-Seq [Ait-El-Hara, Bobot, Bury 2024] [Ait-El-Hara 2025]:
  - Integer indices $[n, m]$ (Ada arrays)
  - Add functions: first and last, constant (sub)sequence, relocate, subsequence update
  - Extensionality axiom using first and last in place of length
- Decidability of quantifier-free fragment: unknown (soundness results)

Summary of the issues and proposed solution

In order to model finite arrays:
deal with either $\forall$ or non-disjointness or possibly undecidability

Solution: a new theory ArrAD of arrays with abstract domain:

- No need for quantifier reasoning
- Deal with the resulting non-disjoint theory unions by CDSAT:
  - Theory combination method that requires neither stably infinite nor disjoint
  - Predicate-sharing theories: either disjoint or sharing predicates other than equality
- The quantifier-free fragment of ArrAD is decidable: follows from fitting ArrAD in CDSAT + CDSAT completeness

The theory of arrays with abstract domain: signature

- Sorts: indices I, values V, arrays A, lengths L, Booleans $Prop$
- $\mathsf{select}\colon A \times I \to V$ $\quad$ $\mathsf{store}\colon A \times I \times V \to A$ $\quad$ $\mathsf{len}\colon A \to L$
- 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 an interpretation of the sorts
- Adm is shared by ArrAD, where it is free
  and another theory $\mathcal{T}$ that provides its definition

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 with length and admissibility:
  $\forall a,\ b.\ [\ \mathrm{len}(a) \simeq \mathrm{len}(b) \wedge$
  $(\forall i.\ \mathrm{Adm}(i, \mathrm{len}(a)) \rightarrow \mathrm{select}(a, i) \simeq \mathrm{select}(b, i))\ ]$
  $\rightarrow a \simeq b$
- Congruence axioms for select, store, len, and Adm

Example: the most common interpretation

- Let LIA be the theory defining Adm
- Interpreting indices and lengths as integers and defining admissibility by the axiom

$$\forall i,\ n.\ \mathrm{Adm}(i,n)\ \leftrightarrow\ 0 \leq i < n$$

- The set of admissible indices is the interval $[0, n)$
- Under this interpretation extensionality in ArrAD covers
  - Extensionality for arrays with length given above
  - Extensionality in the theory Seq of sequences

Example: capturing bounded equality as in APF

- Let LIA be the theory defining Adm
- Interpreting indices as integers, lengths as pairs of integers, and defining admissibility by the axiom

$$\forall i,\ l,\ u.\ \mathrm{Adm}(i,(l,u)) \ \leftrightarrow\ l \leq i \leq u$$

- The set of admissible indices is the interval $[l, u]$
- Under this interpretation extensionality in ArrAD covers
  - Bounded equality in APF
  - Extensionality in the theory N-Seq of sequences

Example: length with starting address

- The theory $\mathcal{T}$ defining Adm interprets indices as integers, lengths as pairs $(addr, n)$ where
  - $addr$ is a binary number – the starting address of the array in memory
  - n is an integer – the number of admissible indices

  and defines Adm by the axiom

  $$\forall i,\ addr,\ n.\ \mathrm{Adm}(i, (addr, n)) \leftrightarrow 0 \leq i < n$$

  Starting address does not affect the admissibility of an index
- 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

- The theory $\mathcal{T}$ defining Adm interprets indices as elements of a set S and lengths as subsets of S
- $\mathcal{T}$ defines admissibility by the axiom

$$\forall i,\ N.\ \mathrm{Adm}(i, N) \ \leftrightarrow\ i \in N$$

- The set of admissible indices is the subset $N \subseteq S$

The set S does not have to be a set of numbers, neither it is required to be (linearly) ordered

Variant: a theory of maps with abstract domain

Same signature as arrays with abstract domain

- Store at inadmissible index i makes only 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: $\forall a,\ b.\ [\ \mathsf{len}(a) \simeq \mathsf{len}(b) \wedge$
  $(\forall i.\ \mathsf{Adm}(i, \mathsf{len}(a)) \rightarrow \mathsf{select}(a,i) \simeq \mathsf{select}(b,i))\ ] \rightarrow a \simeq b$
- Congruence axioms for all symbols

Another variant: a theory of vectors with abstract domain

Vectors are dynamic arrays: how to capture the change?

- Store at an inadmissible index makes that index and those in between (requires $<$ on indices) 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 is as in the theory of maps with abstract domain, except that the signature for vectors adds an ordering $<$ on indices (does not have to be linear)

Theories Seq and N-Seq do not capture the dynamic nature of vectors

Reasoning about arrays, maps, and vectors with abstract domain?
CDSAT

What is CDSAT

- CDSAT: Conflict-Driven SATisfiability in a union of theories
- Orchestrates theory modules in a conflict-driven 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

A theory module $\mathcal{I}_{\mathsf{ArrAD}}$ for ArrAD

From axioms to inference rules, e.g.:

- $n \simeq m,\ i \simeq j,\ \mathsf{Adm}(i, n),\ \neg\, \mathsf{Adm}(j, m)\ \vdash\ \bot$
- $a \simeq b\ \vdash\ \mathsf{len}(a) \simeq \mathsf{len}(b)$
- $\mathsf{len}(\mathsf{store}(a, i, v)) \not\simeq \mathsf{len}(a)\ \vdash\ \bot$
- Some rules generate $\bot$ (conflict detection) others don't: balancing finite basis design and completeness
- From $\forall a, v, i.\ \mathsf{Adm}(i, \mathsf{len}(a)) \rightarrow \mathsf{select}(\mathsf{store}(a, i, v), i) \simeq v$ to $i \simeq j,\ \mathsf{len}(a) \simeq n,\ \mathsf{Adm}(i, n),\ b \simeq \mathsf{store}(a, i, v),\ \mathsf{select}(b, j) \not\simeq v\ \vdash\ \bot$
- It suffices to have $b \simeq \mathsf{store}(a, i, v)$ and $\mathsf{select}(b, j) \not\simeq v$ not necessarily $\mathsf{select}(\mathsf{store}(a, i, v), j) \not\simeq v$

How ArrAD fits in predicate-sharing completeness

The interpretation of arrays:

- Array sort A: updatable function set:
  a set of functions such that every function obtained by a finite number of updates to a member is a member

With abstract domain:

- Partial functions with domain of definition the set of admissible indices
- Array sort A: a collection of updatable function sets $(X_n)_n$ for all values n in the interpretation of the sort L of lengths

How ArrAD fits in predicate-sharing completeness

- **Theorem:** the module for ArrAD is leading-theory-complete for all suitable leading theories $\mathcal{T}_1$
- A leading theory $\mathcal{T}_1$ is 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 $\mathcal{M}_1$
    - $f \in X_n$ is a function from admissible indices to values in the $\mathcal{M}_1$-interpretation of indices, admissibility, and values
    - the sum of the cardinalities of the X_n determines the cardinality of the sort A of arrays in $\mathcal{M}_1$
- Suitability does not restrict combinability

Proofs in CDSAT

- Proof objects in memory (checkable by proof checker)
  - The theory modules produce proofs
  - Proof-carrying CDSAT transition system
  - Proof reconstruction: from proof terms to proofs (e.g., resolution proofs)
- LCF style as in interactive theorem proving (correct by construction)
  - Trusted kernel of primitives

Current and future work

Current work:

- Theory modules for maps and vectors with abstract domain
- Leading theory completeness theorems for them

Longer term:

- Arrays with abstract domain enriched with concatenation (may subsume sequences): QF decidability to be determined
- `Sprout`: a baby CDSAT-based verified solver written in Rust by Xavier Denis
- CDSAT and QSMA (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.
  In preparation (Short version at SMT 2022)

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