CDSAT for Nondisjoint Theories with Shared Predicates: Arrays With Abstract Length[1]

Maria Paola Bonacina

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

20th Int. Workshop on Satisfiability Modulo Theories (SMT)
satellite of the 11th IJCAR held at the 8th FLoC, Haifa, Israel

12 August 2022

[1] Joint work with Stéphane Graham-Lengrand and Natarajan Shankar

From disjoint to nondisjoint theories

- Satisfiability of quantifier-free formulas
- In a union of theories
- Standard hypothesis: the theories are disjoint
- Not true in general, e.g.: length of arrays
  - Two arrays are equal if they have the same length n and the same elements at all indices between 0 and $n - 1$
  - It forces the indices to be integers
  - It forces arrays and integer arithmetic to share symbols
- Length is a bridging function
- Bridging functions make theories nondisjoint

The CDSAT paradigm

- CDSAT: Conflict-Driven SATisfiability in a union of theories
- It orchestrates theory modules in a conflict-driven search
- Theory modules are inference systems, one per theory
- Propositional logic is one of the theories: no hierarchy btw Boolean reasoning and theory reasoning
- Assignments of values to terms: both Boolean and first-order
- Input first-order assignments: satisfiability modulo assignment
- Sound, terminating, and complete for disjoint theories
- How about nondisjoint theories?

An abstract approach that minimizes sharing

- ArrL: theory of arrays with abstract length
- Length is an integer $\rightsquigarrow$ can be but does not have to
- Index within bounds $\rightsquigarrow$ admissible index
- Shared predicate Adm with index and length as arguments
- Adm uninterpreted in ArrL
- Adm interpreted in another theory (e.g., LIA)
- Minimum sharing: Adm, sort of indices, sort of lengths

Example: integers still covered

- Theories: ArrL and LIA
- LIA interprets both lengths and indices as integers
- LIA defines Adm by $\mathrm{Adm}(i, n) \leftrightarrow 0 \leq i < n$
- The set of admissible indices is the interval $[0, n)$

More general example: admissibility as membership

- Theories: ArrL and $\mathcal{T}$
- $\mathcal{T}$ interprets the sort of indices as a set S
- $\mathcal{T}$ interprets the sort of lengths as the powerset $\mathcal{P}(S)$
- $\mathcal{T}$ defines Adm by $\mathrm{Adm}(i, n) \leftrightarrow i \in n$
- $n \in \mathcal{P}(S)$ is a set of admissible indices
- n does not have to be an interval nor even an ordered set
- Indices are not necessarily numbers

More concrete example: length with start address

- Theories: ArrL and $\mathcal{T}$
- $\mathcal{T}$ interprets indices as integers and lengths as pairs $(addr, n)$
- *addr*: binary number representing the start address in memory
- *n*: integer representing the number of admissible indices
- $\mathcal{T}$ defines Adm by $\mathrm{Adm}(i, (addr, n)) \leftrightarrow 0 \leq i < n$
- Arrays *a* and *b* with the same set of admissible indices but different start addresses are different

The theory ArrL of arrays with abstract length: sorts

- Basic sorts including the sort prop of Booleans
- Sorts I of indices, V of elements, L of lengths
- Array sort constructor $\Rightarrow$
- $I \stackrel{L}{\Rightarrow} V$: sort of arrays with
  indices of sort I
  elements of sort V
  lengths of sort L

The theory ArrL of arrays with abstract length: symbols

- $\mathsf{select}\colon (I \stackrel{L}{\Rightarrow} V) \times I \to V$
- $\mathsf{store}\colon (I \stackrel{L}{\Rightarrow} V) \times I \times V \to (I \stackrel{L}{\Rightarrow} V)$
- $\mathsf{len}\colon (I \stackrel{L}{\Rightarrow} V) \to L$
- $\mathsf{Adm}\colon I \times L \to \mathsf{prop}$

The theory ArrL of arrays with abstract length: axioms

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

Alternative choices yield other theories

- What if a store at an inadmissible index i makes it admissible?
- We get other theories:
  - Maps
  - Vectors or dynamic arrays

A theory of maps

- Congruence axioms for select, store, len, and Adm
- Select-over-store axioms do not use Adm:
  - $\forall a, v, i, j.\ i \not\simeq j \rightarrow \mathrm{select}(\mathrm{store}(a, i, v), j) \simeq \mathrm{select}(a, j)$
  - $\forall a, v, i.\ \mathrm{select}(\mathrm{store}(a, i, v), i) \simeq v$
- Store does not change length if the index is admissible:
  $\forall a, i, v.\ \ \mathrm{Adm}(i, \mathrm{len}(a)) \rightarrow \mathrm{len}(\mathrm{store}(a, i, v)) \simeq \mathrm{len}(a)$
- Store at an inadmissible index adds only that index to the admissible set:
  $\forall a, j, i, v.\ \ \mathrm{Adm}(j, \mathrm{len}(\mathrm{store}(a, i, v))) \leftrightarrow (\mathrm{Adm}(j, \mathrm{len}(a)) \vee j \simeq i)$
- Extensionality remains unchanged:
  $\forall a, b.\ [\mathrm{len}(a) \simeq \mathrm{len}(b) \wedge (\forall i.\ \ \mathrm{Adm}(i, \mathrm{len}(a)) \rightarrow$
  $\mathrm{select}(a, i) \simeq \mathrm{select}(b, i))] \rightarrow a \simeq b$

A theory of vectors or dynamic arrays

- Congruence axioms for select, store, len, and Adm
- Select-over-store axioms:
  - $\forall a, v, i, j.\ i \not\simeq j \rightarrow \mathsf{select}(\mathsf{store}(a, i, v), j) \simeq \mathsf{select}(a, j)$
  - $\forall a, v, i.\ \mathsf{select}(\mathsf{store}(a, i, v), i) \simeq v$
- Store at an admissible index does not change length:
  $\forall a, i, v.\ \mathsf{Adm}(i, \mathsf{len}(a)) \rightarrow \mathsf{len}(\mathsf{store}(a, i, v)) \simeq \mathsf{len}(a)$
- Store at an inadmissible index makes that index and those in between (requires an ordering) admissible:
  $\forall a, j, i, v.\ \mathsf{Adm}(j, \mathsf{len}(\mathsf{store}(a, i, v))) \leftrightarrow (\mathsf{Adm}(j, \mathsf{len}(a)) \vee j \leq i)$
- Extensionality:
  $\forall a, b.\ [\mathsf{len}(a) \simeq \mathsf{len}(b) \wedge (\forall i.\ \mathsf{Adm}(i, \mathsf{len}(a)) \rightarrow \mathsf{select}(a, i) \simeq \mathsf{select}(b, i))] \rightarrow a \simeq b$

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

Every CDSAT $\mathcal{T}$-module has equality inference rules:

- $\vdash t_1 \simeq t_1$ (reflexivity)
- $t_1 \simeq t_2 \vdash t_2 \simeq t_1$ (symmetry)
- $t_1 \simeq t_2, t_2 \simeq t_3 \vdash t_1 \simeq t_3$ (transitivity)
- $t_1 \leftarrow \mathfrak{c}, t_2 \leftarrow \mathfrak{c} \vdash t_1 \simeq t_2$ ($\mathfrak{c}$ is a $\mathcal{T}$-value)
- $t_1 \leftarrow \mathfrak{c}_1, t_2 \leftarrow \mathfrak{c}_2 \vdash t_1 \not\simeq t_2$ ($\mathfrak{c}_1$ and $\mathfrak{c}_2$ are $\mathcal{T}$-values, $\mathfrak{c}_1 \neq \mathfrak{c}_2$)

and then adds its own theory-specific rules

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

Rules corresponding to congruence axioms:

- $a \simeq b,\ i \simeq j,\ \mathsf{select}(a, i) \not\simeq \mathsf{select}(b, j)\ \vdash_{\mathsf{ArrL}}\ \bot$
- $a \simeq b,\ i \simeq j,\ u \simeq v,\ \mathsf{store}(a, i, u) \not\simeq \mathsf{store}(b, j, v)\ \vdash_{\mathsf{ArrL}}\ \bot$
- $a \simeq b\ \vdash_{\mathsf{ArrL}}\ \mathsf{len}(a) \simeq \mathsf{len}(b)$
- $n \simeq m,\ i \simeq j,\ \mathsf{Adm}(i, n),\ \neg \mathsf{Adm}(j, m)\ \vdash_{\mathsf{ArrL}}\ \bot$

Some rules generate $\bot$ (conflict detection) and others do not: balancing finite basis design and completeness

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

For the select-over-store axioms

- $\forall a, v, i, j.\ i \not\simeq j \rightarrow \mathsf{select}(\mathsf{store}(a, i, v), j) \simeq \mathsf{select}(a, j)$
- $\forall a, v, i.\ \mathsf{Adm}(i, \mathsf{len}(a)) \rightarrow \mathsf{select}(\mathsf{store}(a, i, v), i) \simeq v$

the rules are:

$$i \not\simeq j,\ k \simeq j,\ b \simeq \mathsf{store}(a, i, v),\ a \simeq c,\ \mathsf{select}(b, k) \not\simeq \mathsf{select}(c, j)\ \vdash_{\mathsf{ArrL}}\ \bot$$
$$i \simeq j,\ \mathsf{len}(a) \simeq n,\ \mathsf{Adm}(i, n),\ b \simeq \mathsf{store}(a, i, v),\ \mathsf{select}(b, j) \not\simeq v\ \vdash_{\mathsf{ArrL}}\ \bot$$

where the premises are flattened:
it suffices to have $b \simeq \mathsf{store}(a, i, v)$ and $\mathsf{select}(b, j) \not\simeq v$
not necessarily $\mathsf{select}(\mathsf{store}(a, i, v), j) \not\simeq v$
(that the equality rules do not infer: no replacement rule for basis finiteness)

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

For the axiom saying that store does not change length:

$\forall a, i, v.\ \mathsf{len}(\mathsf{store}(a, i, v)) \simeq \mathsf{len}(a)$

the rule is

$\mathsf{len}(\mathsf{store}(a, i, v)) \not\simeq \mathsf{len}(a)\ \vdash_{\mathsf{ArrL}}\ \bot$

A theory module $\mathcal{I}_{\mathsf{ArrL}}$ for ArrL: extensionality

Reduce to clausal form

$\forall a, b.\ [\mathsf{len}(a) \simeq \mathsf{len}(b) \wedge (\forall i.\ \mathsf{Adm}(i, \mathsf{len}(a)) \rightarrow$
$\mathsf{select}(a, i) \simeq \mathsf{select}(b, i))] \rightarrow a \simeq b$

Two clauses with Skolem function symbol diff that maps two arrays to an index where they differ:

$a \not\simeq b,\ \mathsf{len}(a) \simeq \mathsf{len}(b)\ \vdash_{\mathsf{ArrL}}\ \mathsf{select}(a, \mathsf{diff}(a, b)) \not\simeq \mathsf{select}(b, \mathsf{diff}(a, b))$

$a \not\simeq b,\ \mathsf{len}(a) \simeq \mathsf{len}(b)\ \vdash_{\mathsf{ArrL}}\ \mathsf{Adm}(\mathsf{diff}(a, b), \mathsf{len}(a))$

A congruence rule also for the Skolem symbol diff:

$a \simeq c,\ b \simeq d,\ \mathsf{diff}(a, b) \not\simeq \mathsf{diff}(c, d)\ \vdash_{\mathsf{ArrL}}\ \bot$

Soundness, termination, and completeness of CDSAT

- Soundness: whenever a derivation reaches unsat, the input is unsatisfiable
  It suffices that the theory modules are sound (unchanged wrt the disjoint case)
- Termination: every derivation is guaranteed to halt
  It suffices that there exists a finite global basis containing all input terms (unchanged wrt the disjoint case)
- Completeness: whenever a derivation halts in a state other than unsat, there exists a $\mathcal{T}_\infty^+$-model of the trail (and hence of the input) (re-proved for the predicate-sharing case)

Sufficient conditions for completeness

- Predicate-sharing union $\mathcal{T}_\infty$ of theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$:
  - Disjoint or sharing predicate symbols
  - Leading theory $\mathcal{T}_1$ that has all sorts and all shared symbols
- Complete collection of theory modules $\mathcal{I}_1, \ldots, \mathcal{I}_n$:
  - Module $\mathcal{I}_1$ is complete for $\mathcal{T}_1$: if it cannot expand its view $\Gamma_{\mathcal{T}_1}$ of trail Γ, there exists a $\mathcal{T}_1^+$-model $\mathcal{M}_1$ of $\Gamma_{\mathcal{T}_1}$
  - For all k, $2 \leq k \leq n$, module $\mathcal{I}_k$ is leading-theory-complete: if it cannot expand $\Gamma_{\mathcal{T}_k}$, there exists a $\mathcal{T}_k^+$-model $\mathcal{M}_k$ of $\Gamma_{\mathcal{T}_k}$ that agrees with $\mathcal{M}_1$ on the interpretation of shared predicates and on the cardinalities of shared sorts

How ArrL fits in predicate-sharing completeness

The interpretation of arrays:

- Array: function
- Updatable function set: every function obtained by a finite number of updates to a member is a member
- Array sort $I \Rightarrow V$: updatable function set

With abstract length:

- Array: partial function
  Domain of definition: the set of admissible indices
- Array sort $I \stackrel{L}{\Rightarrow} V$: a collection of updatable function sets, one for every value in the interpretation of L

How ArrL fits in predicate-sharing completeness

- Module $\mathcal{I}_{\mathsf{ArrL}}$ is leading-theory-complete for all ArrL-suitable leading theories
- A leading theory $\mathcal{T}_1$ is ArrL-suitable if
  - $\mathcal{T}_1$ has all the sorts of ArrL
  - $\mathcal{T}_1$ shares with ArrL only the symbol Adm (and equality)
  - For all $\mathcal{T}_1$-models $\mathcal{M}_1$ and sorts $I \overset{L}{\Rightarrow} V$ there exists a collection of updatable function sets $(X_n)_{n \in L^{\mathcal{M}_1}}$ such that

    $$|(I \overset{L}{\Rightarrow} V)^{\mathcal{M}_1}| = |\biguplus_{n \in L^{\mathcal{M}_1}} X_n|$$

    for all $n \in L^{\mathcal{M}_1}$: X_n is the set of partial updatable functions with domain $I_n = \{i \mid i \in I^{\mathcal{M}_1} \wedge \mathrm{Adm}^{\mathcal{M}_1}(i, n)\}$ and codomain $V^{\mathcal{M}_1}$ used to interpret the arrays of length n

Example with ArrL and LIA revisited

- LIA interprets L and I as $\mathbb{Z}$
- LIA defines Adm by $\text{Adm}(i, n) \leftrightarrow 0 \leq i < n$
- Suppose ArrL interprets also V as $\mathbb{Z}$
- $\mathcal{T}_1$ interpreting L, I, and Adm like LIA, and V like ArrL is ArrL-suitable:
  for all $n \in \mathbb{Z}$, $I_n = \{i \mid i \in \mathbb{Z} \wedge 0 \leq i < n\}$
  for all n, $n > 0$, X_n is countably infinite
  Cardinality of the interpretation of $I \overset{L}{\Rightarrow} V$: countably infinite
- A theory interpreting $I \overset{L}{\Rightarrow} V$ as being finite: not ArrL-suitable

Example with ArrL and bitvectors

- BV interprets I as BV[1], L as BV[2]
  Adm as true everywhere except $(0, 00)$, $(1, 00)$, and $(1, 01)$
- Suppose that ArrL and BV share also V
  and BV interprets it as BV[1]
- $\mathcal{T}_1$ interpreting L, I, Adm, and V like BV is ArrL-suitable:
  $I_{00} = \emptyset$, $I_{01} = \{0\}$, and $I_{10} = I_{11} = \{0, 1\}$
  $|X_{00}| = 2^0 = 1$, $|X_{01}| = 2^1 = 2$, and $|X_{10}| = |X_{11}| = 2^2 = 4$
  Cardinality of the interpretation of $I \stackrel{L}{\Rightarrow} V$: 11
- A theory interpreting $I \stackrel{L}{\Rightarrow} V$ as countably infinite: not ArrL-suitable

Current and future work

- Develop this abstract approach to nondisjointness due to bridging functions for
  - A version of theory ArrL enriched with concatenation
  - The theory of finite maps
  - The theory of vectors or dynamic arrays
  - Lists with length (generalized to recursive data structures)
- Implementation of CDSAT in Rust
  (by Xavier Denis)
- Extend CDSAT with quantifier reasoning
  (with Christophe Vauthier)