The CDSAT Paradigm for SMT: Extension to Nondisjoint Theories[1]

Maria Paola Bonacina

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

Talk given at the CS Dept., The University of Manchester
Manchester, England, UK, 21 March 2023

(Subsuming: "CDSAT for Nondisjoint Theories with Shared Predicates," CS Dept., Yale University, New Haven, CT, USA, 22 August 2022; and "CDSAT for Nondisjoint Theories with Shared Predicates: Arrays with Abstract Length," SRI International, Menlo Park, CA, USA, 28 July 2022)

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

The CDSAT paradigm

- CDSAT: Conflict-Driven SATisfiability in a union of theories
- Orchestrates theory modules in a conflict-driven search
- 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
- Generalizes MCSAT, CDCL(T), and Nelson-Oppen

From disjoint to nondisjoint theories

- Satisfiability of quantifier-free formulas
- In a union of theories
- Standard hypothesis: disjoint theories
- 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

An abstract approach that minimizes sharing

- New: theory of arrays with abstract length (ArrL)
- Abstraction:
  - Length is an integer $\rightsquigarrow$ can be but does not have to
  - Index within bounds $\rightsquigarrow$ admissible index
- Predicate $\mathsf{Adm}(i, l)$: index i is admissible wrt length l
- Adm is shared:
  - Adm uninterpreted in ArrL
  - Adm interpreted in another theory (e.g., LIA)
- Minimum sharing: Adm and the sorts of its arguments indices and lengths

Example: integers still covered

- Theories: ArrL and LIA
- LIA interprets both lengths and indices as integers
- LIA defines admissibility as

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

- 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:
  - Does not have to be a set of numbers
  - Does not have to be a linearly ordered set
  - Does not have to be an ordered set
- $\mathcal{T}$ interprets the sort of lengths as the powerset $\mathcal{P}(S)$
- $\mathcal{T}$ defines admissibility as

$$\mathrm{Adm}(i, n) \leftrightarrow i \in n$$

- $n \in \mathcal{P}(S)$ is a set of admissible indices

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 $\mathrm{Adm}(i,(addr,n)) \leftrightarrow 0 \le i < n$
- 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

- *Prop*: sort of Booleans
- *Ind*: sort of indices
- *Val*: sort of values
- *Len*: sort of lengths
- *A*: sort of arrays with indices of sort *Ind*, elements of sort *Val*, and lengths of sort *Len*
- No loss of generality: e.g. a theory of matrices as a disjoint union

The theory ArrL of arrays with abstract length: symbols

- select : $A \times \mathit{Ind} \to \mathit{Val}$
- store : $A \times \mathit{Ind} \times \mathit{Val} \to A$
- len : $A \to \mathit{Len}$
- Adm : $\mathit{Ind} \times \mathit{Len} \to \mathit{Prop}$

The theory ArrL of arrays with abstract length: axioms

- Congruence axioms for select, store, len, and Adm
- $\forall a, v, i, j.\ i \not\simeq j \rightarrow \mathrm{select}(\mathrm{store}(a, i, v), j) \simeq \mathrm{select}(a, j)$
- A store at an inadmissible index has no effect:
  - From: $\forall a, v, i.\ \mathrm{select}(\mathrm{store}(a, i, v), i) \simeq v$
    to: $\forall a, v, i.\ \mathrm{Adm}(i, \mathrm{len}(a)) \rightarrow \mathrm{select}(\mathrm{store}(a, i, v), i) \simeq v$
  - $\forall a, i, v.\ \mathrm{len}(\mathrm{store}(a, i, v)) \simeq \mathrm{len}(a)$
- 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:
  - A is the sort of maps with keys of sort Ind, values of sort Val, and length of sort Len
  - Hashmaps: as values are not allocated at consecutive addresses in memory, abstracting away from intervals of indices is essential
- Vectors or dynamic arrays:
  - A is the sort of vectors with indices of sort Ind, values of sort Val, and length of sort Len

A theory of maps

- Congruence axioms for select, store, len, and 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 changes length by adding 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 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, Adm, and $<$
- $\forall a, v, i, j.\ i \not\simeq j \rightarrow \mathsf{select}(\mathsf{store}(a, i, v), j) \simeq \mathsf{select}(a, j)$
- $\forall a, v, i.\ \mathsf{select}(\mathsf{store}(a, i, v), i) \simeq v$
- 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 $<$ on indices) admissible: $\forall a, j, i, v.\ \mathsf{Adm}(j, \mathsf{len}(\mathsf{store}(a, i, v))) \leftrightarrow (\mathsf{Adm}(j, \mathsf{len}(a)) \vee j \leq i)$
- Extensionality unchanged: $\forall a, b.\ [\mathsf{len}(a) \simeq \mathsf{len}(b) \wedge (\forall i.\ \mathsf{Adm}(i, \mathsf{len}(a)) \rightarrow \mathsf{select}(a, i) \simeq \mathsf{select}(b, i))] \rightarrow a \simeq b$

Satisfiability modulo theories and assignments

- Given a formula F and an initial assignment to some of its terms (Boolean or first-order)
- Find a theory model that extends the assignment and satisfies the formula F
- Or report that none exists

F can be written as $F \leftarrow \text{true}$: everything is an assignment

Assignments

- $\mathcal{T}$-assignment: $u \leftarrow \mathfrak{c}$
- u: term in the signature of the union of the theories
- $\mathfrak{c}$: $\mathcal{T}$-value (constant provided by theory extension $\mathcal{T}^+$ and used to name an element in an intended model's domain as needed)
- Boolean: $(i \simeq j) \leftarrow \mathsf{true}$ or simply $i \simeq j$
- First-order: $i \leftarrow 3$ (not the same as $(i \simeq 3) \leftarrow \mathsf{true}$)
- In general: $\{u_1 \leftarrow \mathfrak{c}_1, \ldots, u_m \leftarrow \mathfrak{c}_m\}$ mixing values, e.g.:
- $\{i \leftarrow 3,\ i \simeq j,\ \mathsf{len}(a) \simeq n,\ n \leftarrow 5,\ \mathsf{select}(\mathsf{store}(a, i, v), j) \not\simeq v\}$
- Plausible: does not contain both $u \leftarrow \mathsf{true}$ and $u \leftarrow \mathsf{false}$

Every theory has its view of a mixed assignment

- $\mathcal{T}_\infty$: union of theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$
- $\mathcal{T}$: theory with set of sorts S
- H: $\mathcal{T}_\infty$-assignment
- The $\mathcal{T}$-view $H_\mathcal{T}$ of H is the union of
  - $\{\ u{\leftarrow}\mathfrak{c} \mid\ u{\leftarrow}\mathfrak{c}$ is a $\mathcal{T}$-assignment in $H\}$
  - $\{\ u_1 \simeq u_2 \mid u_1{\leftarrow}\mathfrak{c}, u_2{\leftarrow}\mathfrak{c}$ in H of sort $s \in S \setminus \{Prop\}\}$
  - $\{\ u_1 \not\simeq u_2 \mid u_1{\leftarrow}\mathfrak{c}_1, u_2{\leftarrow}\mathfrak{c}_2$ in H of sort $s \in S \setminus \{Prop\},\ \mathfrak{c}_1{\neq}\mathfrak{c}_2\}$
- Global view: the $\mathcal{T}_\infty$-view (contains everything)

Examples of theory views

- $H =$
  $\{i \leftarrow 3,\ i \simeq j,\ \mathsf{len}(a) \simeq n,\ n \leftarrow 5,\ \mathsf{select}(\mathsf{store}(a, i, v), j) \not\simeq v\}$
- LIA-view: $H \cup \{i \not\simeq n\}$
- ArrL-view: the Boolean assignments in H and $\{i \not\simeq n\}$
- Global view: same as the LIA-view

Assignments and models

- $\mathcal{T}^+$-model $\mathcal{M}$ and $\mathcal{T}$-assignment J
- $\mathcal{M} \models J$: $\mathcal{M}$ satisfies $u \simeq \mathfrak{c}$ for all $(u \leftarrow \mathfrak{c}) \in J$
- $\{u \leftarrow \mathfrak{c}, t \leftarrow \mathfrak{c}\} \subseteq J$: $\mathcal{M}$ also satisfies $u \simeq t$
- $\mathcal{M} \models J_{\mathcal{T}}$: $\mathcal{M}$ also satisfies the disequalities $u \not\simeq t$ in $J_{\mathcal{T}}$
- J is satisfiable if there exists an $\mathcal{M}$ such that $\mathcal{M} \models J_{\mathcal{T}}$
- For $\mathcal{T}_\infty$: globally satisfiable

- L: singleton Boolean assignment
- $J \models L$: $\mathcal{M} \models L$ for all $\mathcal{M}$ such that $\mathcal{M} \models J_{\mathcal{T}}$

Theory modules

- A theory module $\mathcal{I}_k$ for every component theory $\mathcal{T}_k$
- Theory module: abstraction of a reasoning procedure
- Inference rules: $J \vdash_{\mathcal{I}} L$
  J: $\mathcal{T}$-assignment, L: singleton Boolean assignment
- Soundness: if $J \vdash L$ then $J \models L$
- Inferences can generate new (non-input) terms
- For termination:
  - Given finite set X of input terms
  - Local basis $\mathrm{basis}(X)$: finite superset of X
  - New terms must be in $\mathrm{basis}(X)$
- Global finite basis $\mathcal{B}$ built from the local bases

Equality inference rules

Every $\mathcal{T}$-module contains the 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

Reducing
$\forall a, b.\ [\mathsf{len}(a) \simeq \mathsf{len}(b) \wedge (\forall i.\ \mathsf{Adm}(i, \mathsf{len}(a)) \rightarrow$
$\mathsf{select}(a, i) \simeq \mathsf{select}(b, i))] \rightarrow a \simeq b$
to clausal form yields two clauses with Skolem function symbol diff that maps two arrays to an admissible 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 diff:

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

CDSAT works on a trail containing the current assignment

- Trail Γ: sequence of distinct singleton assignments
  - Decision: ${}_{?}A$
  - Justified assignment: ${}_{H\vdash}A$
    Justification H: assignments that appear before A in Γ
- Input assignments are justified assignments with empty H
- Justified assignments are Boolean
  except for input first-order assignments
- Level of an assignment

The CDSAT trail rule Decide

- Decide: $\Gamma \longrightarrow \Gamma, {}_{?}A$

  if A is a $\mathcal{T}$-assignment $u \leftarrow \mathfrak{c}$ that is acceptable for $\mathcal{T}$-module $\mathcal{I}$ in the $\mathcal{T}$-view $\Gamma_{\mathcal{T}}$ of the trail:

  1. $\Gamma_{\mathcal{T}}$ does not already assign a $\mathcal{T}$-value to u
  2. If $u \leftarrow \mathfrak{c}$ is first-order: for no inference $J' \cup \{u \leftarrow \mathfrak{c}\} \vdash_{\mathcal{I}} L$ with $J' \subseteq \Gamma_{\mathcal{T}}$ we have $\overline{L} \in \Gamma_{\mathcal{T}}$
  3. Term u is relevant to theory $\mathcal{T}$ in $\Gamma_{\mathcal{T}}$

Predicate-sharing relevance

- $\mathcal{T}$: theory
- J: $\mathcal{T}$-assignment
- Term u is relevant to $\mathcal{T}$ in J if:
    1. u occurs in J and $\mathcal{T}$ has values for its sort
    2. u is an equality whose sides u_1, u_2 occur in J but $\mathcal{T}$ does not have values for their sort
    3. u is a Boolean term $p(u_1, \ldots, u_m)$ such that p is a shared predicate symbol and the u_i's occur in J

Example

- $H =$ $\{i \leftarrow 3,\ i \simeq j,\ \mathrm{len}(a) \simeq n,\ n \leftarrow 5,\ \mathrm{select}(\mathrm{store}(a, i, v), j) \not\simeq v\}$
- LIA-view: $H \cup \{i \not\simeq n\}$
- ArrL-view: the Boolean assignments in H and $\{i \not\simeq n\}$
- $\mathrm{Adm}(i, n)$ does not occur in either view, but its arguments do
- $\mathrm{Adm}(i, n)$ is relevant to both LIA and ArrL
- Having the definition of Adm, LIA can decide $\mathrm{Adm}(i, n) \leftarrow \mathrm{true}$
- If ArrL decides $\mathrm{Adm}(i, n) \leftarrow \mathrm{false}$, LIA detects a conflict

The other CDSAT trail rules in words

- Deduce expands Γ with a justified assignment ${}_{J\vdash}A$ supported by a theory inference $J \vdash A$
- Deduce covers
  - Propagation: adds consequences of decisions
  - Conflict detection: detects a theory conflict
  - Conflict explanation: transforms it into a Boolean conflict: L can be derived and $\overline{L}$ is on the trail
- Boolean conflict at level 0: Fail reports unsatisfiability
- Boolean conflict at level > 0: ConflictSolve puts the system in conflict state

Example: Deduce as propagation

1. Decide: $u_2 \leftarrow \mathrm{yellow}$ (level 1)
2. Decide: $f(u_1) \leftarrow \mathrm{red}$ (level 2)
3. Decide: $u_1 \leftarrow \mathrm{yellow}$ (level 3)
4. Decide: $f(u_2) \leftarrow \mathrm{blue}$ (level 4)
5. Deduce: $u_1 \simeq u_2$ (level 3) /* equality inference */
6. Deduce: $f(u_1) \simeq f(u_2)$ (level 3) /* EUF-inference */

The Deduce steps are late propagations

Example: a conflict emerges

1. Decide: $u_2 \leftarrow \text{yellow}$ (level 1)
2. Decide: $f(u_1) \leftarrow \text{red}$ (level 2)
3. Decide: $u_1 \leftarrow \text{yellow}$ (level 3)
4. Decide: $f(u_2) \leftarrow \text{blue}$ (level 4)
5. Deduce: $u_1 \simeq u_2$ (level 3) /* late propagation */
6. Deduce: $f(u_1) \simeq f(u_2)$ (level 3) /* late propagation */
7. $\{f(u_1) \leftarrow \text{red},\ f(u_2) \leftarrow \text{blue}\} \vdash f(u_1) \not\simeq f(u_2)$: conflict
   by any theory module since it is an equality inference
8. ConflictSolve . . . .

The CDSAT conflict state rules in words

- UndoClear: solves the conflict by undoing a 1st-order assignment and clearing the trail of all its consequences
- Resolve: explains the conflict by replacing $_{H\vdash}A$ in the conflict with H
- LearnBackjump: solves the conflict by flipping a Boolean assignment (not necessarily unit: flips a cube into a clause and learns it) and backjumping
- UndoDecide: preempts Resolve to avoid a Resolve, UndoClear, Decide, Resolve loop; solves the conflict by undoing a 1st-order assignment and all its consequences, and flipping a Boolean one (a 1st-order assignment cannot be flipped)

Example: UndoClear

1. Decide: $u_2 \leftarrow \text{yellow}$ (level 1)
2. Decide: $f(u_1) \leftarrow \text{red}$ (level 2)
3. Decide: $u_1 \leftarrow \text{yellow}$ (level 3)
4. Decide: $f(u_2) \leftarrow \text{blue}$ (level 4)
5. Deduce: $u_1 \simeq u_2$ (level 3) /* late propagation */
6. Deduce: $f(u_1) \simeq f(u_2)$ (level 3) /* late propagation */
7. Conflict: $\{f(u_1) \simeq f(u_2),\ f(u_1) \leftarrow \text{red},\ f(u_2) \leftarrow \text{blue}\}$
8. UndoClear: undoes $f(u_2) \leftarrow \text{blue}$ /* max level in the conflict */
9. Decide: $f(u_2) \leftarrow \text{red}$ (level 4) /* only acceptable value */

Example: UndoDecide

Γ includes: $x > 1 \vee y < 0$, $x < -1 \vee y > 0$ (level 0)

1. Decide: $x \leftarrow 0$ (level 1)
2. Deduce[4]: $(x > 1) \leftarrow \mathsf{false}$ with justification $x \leftarrow 0$ (level 1)
   $(x < -1) \leftarrow \mathsf{false}$ with justification $x \leftarrow 0$ (level 1)
   $y < 0$ with justification $\{x > 1 \vee y < 0,\ \overline{x > 1}\}$ (level 1)
   $y > 0$ with justification $\{x < -1 \vee y > 0,\ \overline{x < -1}\}$ (level 1)
3. Conflict: $\{y{<}0,\ y{>}0\}$
4. Resolve[2]: $\{x > 1 \vee y < 0,\ x < -1 \vee y > 0,\ \overline{x > 1},\ \overline{x < -1}\}$
5. UndoDecide: $x > 1$ (level 1)

Example: Resolve + LearnBackjump

Γ includes: $(\neg L_4 \vee L_5)$, $(\neg L_2 \vee \neg L_4 \vee \neg L_5)$ (level 0)

1. Decide: A_1 (level 1)
2. Decide: L_2 (level 2)
3. Decide: A_3 (level 3)
4. Decide: L_4 (level 4)
5. Deduce: L_5 with justification $\{\neg L_4 \vee L_5,\ L_4\}$ (level 4)
6. Conflict: $\{\neg L_2 \vee \neg L_4 \vee \neg L_5,\ L_2,\ L_4,\ L_5\}$
   $\neg L_2 \vee \neg L_4 \vee \neg L_5$ is the CDCL conflict clause
7. Resolve: $\{\neg L_2 \vee \neg L_4 \vee \neg L_5,\ L_2,\ L_4,\ \neg L_4 \vee L_5\}$
   $\neg L_2 \vee \neg L_4$ is the next CDCL conflict clause (resolvent of previous one and CDCL justification $\neg L_4 \vee L_5$) and first assertion clause

Example: Resolve + LearnBackjump

Conflict: $\{\neg L_2 \vee \neg L_4 \vee \neg L_5,\ L_2,\ L_4,\ \neg L_4 \vee L_5\}$

- LearnBackjump flips cube $H = \{L_2, L_4\}$ into clause $\neg L_2 \vee \neg L_4$, learns it as a justified assignment with justification $E = \{\neg L_2 \vee \neg L_4 \vee \neg L_5,\ \neg L_4 \vee L_5\}$ (level 0)
- And backjumps to any level m ($\mathrm{level}_\Gamma(E) \leq m < \mathrm{level}_\Gamma(H)$):
  - Destination level $m = 2$ (1stUIP):
    - $\ldots(\neg L_4 \vee L_5),\ (\neg L_2 \vee \neg L_4 \vee \neg L_5),\ A_1,\ L_2,\ (\neg L_2 \vee \neg L_4)$
    - Deduce: $\neg L_4$ with justification $\{\neg L_2 \vee \neg L_4,\ L_2\}$
  - Destination level $m = 0$: restart from $\ldots(\neg L_4 \vee L_5),\ (\neg L_2 \vee \neg L_4 \vee \neg L_5),\ (\neg L_2 \vee \neg L_4)$

Summary: the CDSAT trail rules

- Decide: $\Gamma \longrightarrow \Gamma, {}_{?}A$
  if A is a $\mathcal{T}$-assignment $u \leftarrow \mathfrak{c}$ that is acceptable for $\mathcal{I}$ in $\Gamma_{\mathcal{T}}$
- Assume $J \subseteq \Gamma$, $J \vdash L$, and $L \notin \Gamma$:
  - Deduce: $\Gamma \longrightarrow \Gamma, {}_{J\vdash}L$
    if $\overline{L} \notin \Gamma$ and L is in $\mathcal{B}$ /* $\mathcal{B}$ is the finite global basis */
  - Fail: $\Gamma \longrightarrow$ unsat
    if $\overline{L} \in \Gamma$ and $\mathrm{level}_{\Gamma}(J \cup \{\overline{L}\}) = 0$
  - ConflictSolve: $\Gamma \longrightarrow \Gamma'$
    if $\overline{L} \in \Gamma$, $\mathrm{level}_{\Gamma}(J \cup \{\overline{L}\}) > 0$, and $\langle \Gamma; J \cup \{\overline{L}\} \rangle \Longrightarrow^{*} \Gamma'$
    conflict state: $\langle \Gamma; E \rangle$
    E: conflict (unsatisfiable assignment)

Summary: the CDSAT conflict state rules

- UndoClear: $\langle \Gamma; E \uplus \{A\}\rangle \implies \Gamma^{\leq m-1}$
  if A is a first-order decision of level $m > \mathrm{level}_\Gamma(E)$
- UndoDecide: $\langle \Gamma; E \uplus \{{}_{H\vdash} L\}\rangle \implies \Gamma^{\leq m-1}, {}_{?}\overline{L}$
  if for a first-order decision $A' \in H$,
  $m = \mathrm{level}_\Gamma(E) = \mathrm{level}_\Gamma(L) = \mathrm{level}_\Gamma(A')$
- Resolve: $\langle \Gamma; E \uplus \{{}_{H\vdash} A\}\rangle \implies \langle \Gamma; E \cup H\rangle$
  if for no first-order decision $A' \in H$, $\mathrm{level}_\Gamma(A') = \mathrm{level}_\Gamma(E \uplus \{A\})$
- LearnBackjump: $\langle \Gamma; E \uplus H\rangle \implies \Gamma^{\leq m}, {}_{E\vdash} L$
  if L is a clausal form of H, $L \in \mathcal{B}$, $L \notin \Gamma$, $\overline{L} \notin \Gamma$, and
  $\mathrm{level}_\Gamma(E) \leq m < \mathrm{level}_\Gamma(H)$

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 $\mathcal{B}$ containing all input terms (only the construction of $\mathcal{B}$ changes 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 $\mathcal{I}_1, \ldots, \mathcal{I}_n$:
  - $\mathcal{I}_1$ is complete for $\mathcal{T}_1$: if it cannot expand (with a trail rule) $\Gamma_{\mathcal{T}_1}$, there exists a $\mathcal{T}_1^+$-model $\mathcal{M}_1$ of $\Gamma_{\mathcal{T}_1}$
  - For all k, $2 \leq k \leq n$, $\mathcal{I}_k$ 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: a function
- Updatable function set: every function obtained by a finite number of updates to a member is a member
- Array sort A: updatable function set

With abstract length:

- Array: a partial function
  Domain of definition: the set of admissible indices
- Array sort A: a collection of updatable function sets $(X_n)_n$, one for every length n (value in the interpretation of Len)

How ArrL fits in predicate-sharing completeness

- **Thm.:** Module $\mathcal{I}_{\mathrm{ArrL}}$ 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 $\mathcal{M}_1$ there exists a collection of updatable function sets $(X_n)_{n \in Len^{\mathcal{M}_1}}$ such that

$$|A^{\mathcal{M}_1}| = |\biguplus_{n \in Len^{\mathcal{M}_1}} X_n|$$

    X_n is an updatable function set from $I_n = \{i \mid i \in Ind^{\mathcal{M}_1} \wedge \mathrm{Adm}^{\mathcal{M}_1}(i, n)\}$ to $Val^{\mathcal{M}_1}$ that interprets the arrays of length n

Example with ArrL and LIA revisited

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

Example with ArrL and bitvectors

- BV interprets *Ind* as BV[1], *Len* as BV[2]
  Adm as true everywhere except $(0, 00)$, $(1, 00)$, and $(1, 01)$
- Suppose that ArrL and BV share also *Val*
  and BV interprets it as BV[1]
- $\mathcal{T}_1$ interpreting *Len*, *Ind*, Adm, and *Val* 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 A: 11
- A theory interpreting A as countably infinite: not ArrL-suitable

Current and future work

- Develop this abstract approach to nondisjointness due to bridging functions for
  - Maps
  - Vectors aka dynamic arrays
  - Arrays (ArrL) enriched with concatenation
  - Lists with length (generalizable to recursive data structures)
- Implementation of CDSAT in Rust
  (by Xavier Denis)
- Extend CDSAT with quantifier reasoning
  (with Christophe Vauthier)

References

- Satisfiability modulo theories and assignments. At CADE 2017
- Proofs in conflict-driven theory combination. At CPP 2018
- Conflict-driven satisfiability for theory combination: transition system and completeness. JAR 64(3):579–609, 2020
- Conflict-driven satisfiability for theory combination: modules, lemmas, and proofs. JAR 66(1):43–91, 2022
- CDSAT for nondisjoint theories with shared predicates: arrays with abstract length. At SMT 2022
- CDSAT for nondisjoint theories with shared predicates. Journal version in preparation

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