Reasoning about Data Structures with CDSAT[1]

Maria Paola Bonacina

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

19th Summer School on Verification Technology, Systems and Applications (VTSA)
MPI Saarbrücken, Germany, EU
A 3.5h lecture on 24 August and a 3.5h lecture on 26 August 2026

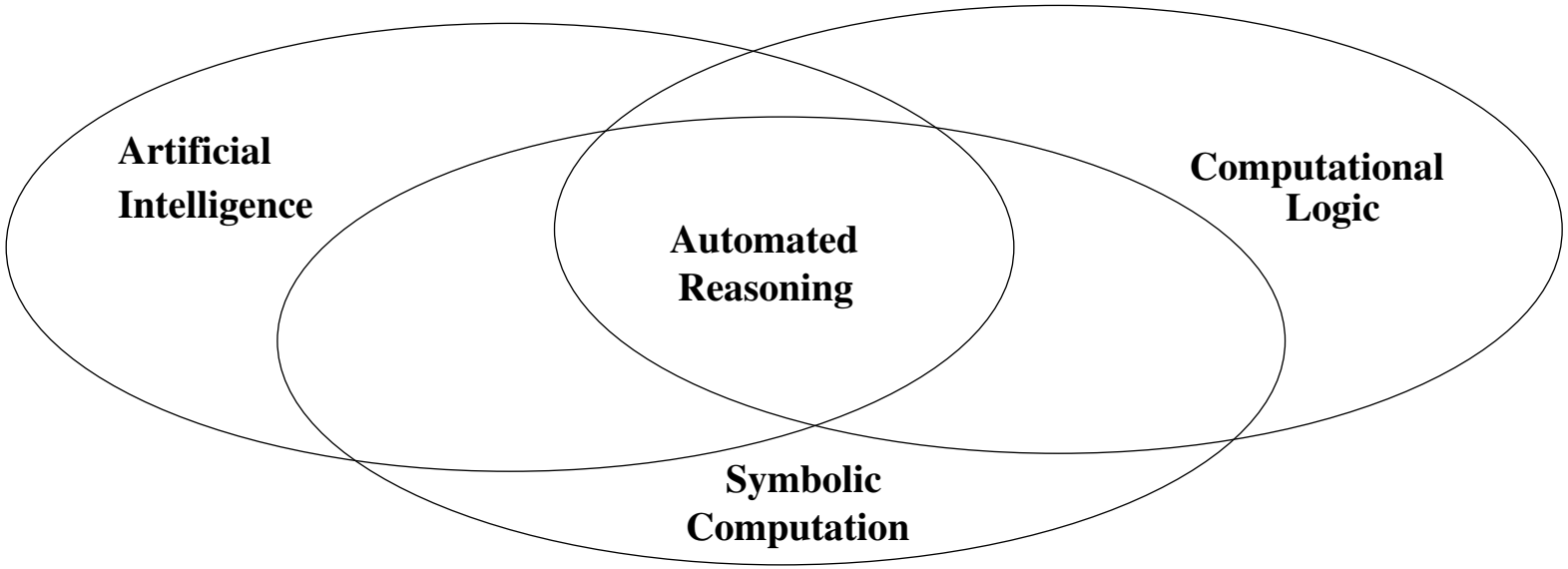
(Subsumes a 2h invited tutorial entitled "The CDSAT Paradigm for Theory Combination in SMT,"
21st Int. Conf. on Computability in Europe (CiE), Lisbon, Portugal, EU, 15 and 17 July 2025,
and a 3h invited talk of the same title, CEA, Paris, France, EU, 15 October 2025)

[1] Based on joint work with S. Graham-Lengrand and N. Shankar

First lecture

- Introduction and motivation
- Conflict-driven reasoning and CDSAT in a nutshell
- Propositional conflict-driven reasoning: CDSAT and CDCL
- Combination of theories: CDSAT and Nelson-Oppen
- Conflict-driven reasoning in one theory: LRA procedures

What is automated reasoning (AR)



- AR: enable computers to reason ... in their own way
- AR is AI: logic-based, symbolic AI
- AR is SC: logico-deductive, symbolic reasoning
- AR is CL: use computing to perform logical reasoning

What automated reasoning does

- Design and implementation of computer programs that reason
- To solve problems formulated as
- Validity or satisfiability queries in a logic or a theory
- Using inference and search

- Valid: true in all circumstances
- Satisfiable: true in at least one circumstance

Why automated reasoning at a verification school

Applications of automated reasoning:

- AR is embedded in tools for the verification, synthesis, and optimization of software
- AR gets combined with:
  - Testing: AR for test generation
  - Model checking: AR for bounded model checking and abstraction in counter-example guided abstraction refinement
  - Static analysis: AR for invariant synthesis
- AR is also applied in deductive knowledge bases, computer mathematics, mathematical libraries, education, ...
- Could the integration of AR and ML lead to a better AI?

Automated reasoning and verification

Objectives of verification/synthesis:

- Classic:
  - Partial correctness, termination
  - Correct-by-construction software
- More recent:
  - Provable privacy
  - Distributed systems/protocols
  - Randomized algorithms
  - Probabilistic programs

Automated reasoning and deductive verification

Objectives: partial correctness, termination

- Program state: assignment to program variables
- Annotation: formula where the only free (i.e., not quantified) variables are program variables
- Annotated program: code interspersed with annotations
- When the execution reaches the annotation, the program state should satisfy the annotation:
  the annotation is an invariant
- Key analogy: a program state as a model

Automated reasoning and deductive verification

- Verifying compiler: takes annotated program and generates executable + verification conditions (VCs)
- If the VC's are valid, the program satisfies the annotations
- Automated reasoner: takes each VC and generates either proof of validity or counter-model
- Proof: certificate of correctness
- Counter-model: assignment to program variables useful for debugging

The automated reasoning problem

- Given a set H of assumptions and a conjecture φ
- Problem:
  $H \models \varphi$: is φ a logical consequence of H?
  $\models (H \supset \varphi)$: is $H \supset \varphi$ valid?
  $H \cup \{\neg\varphi\} \models \bot$: is $H \cup \{\neg\varphi\}$ unsatisfiable?
- Solution:
  Either a proof of the unsatisfiability of $H \cup \{\neg\varphi\}$
  Or a model of $H \cup \{\neg\varphi\}$: counter-model of φ

Discharging verification conditions is an instance of the AR problem

The automated reasoning problem in these lectures

- Satisfiability modulo theory: defined symbols in queries
  Not many theories, but more than one: combination
  Verification conditions: e.g., arithmetic and data structures
- Fragments where satisfiability is decidable: decision procedure
  Almost a requirement for verification
- Quantifier-free and clausal input problems
  Quantifier-free: a limitation, but there is progress
  Clauses: standard machine language for logic
- Focus on conflict-driven reasoning: key ingredient
  in SAT and SMT solvers (not the only one)

Examples: clauses involving theory-defined symbols

- Propositional logic (the Boolean theory):
  $\{\overline{A} \vee B,\ \overline{A} \vee C \vee E,\ \overline{B} \vee D,\ \overline{C} \vee \overline{D}, A \vee \overline{B} \vee E,\ B \vee \overline{C},\ F \vee \overline{E}\}$
- Linear integer arithmetic (LIA) and Equality with Uninterpreted Functions (EUF or UF):
  $\{x \leq y,\ y \leq (x + g(x)),\ p(h(x) - h(y)),\ \neg p(0),\ g(x) \simeq 0\}$
- Bool and linear rational arithmetic (LRA):
  $\{x < y,\ x < z,\ (y < w) \vee (z < w),\ w < x\}$
- Bool, LRA, and Arrays (Arr):
  $(i \not\simeq j) \vee (\text{select}(\text{store}(a, i, v), j) < \text{select}(a, j))$
  $(\text{select}(a, j) - \text{select}(a, k)) \simeq 0$
  $(\text{select}(\text{store}(a, i, v), j) \not< \text{select}(a, j)) \vee$
  $(\text{select}(a, j) + \text{select}(a, k) \simeq v)$

Conflict-driven reasoning

- Decision procedure for satisfiability of a set of clauses
- Search for a model: build candidate, propose assignments, amend to solve conflict
- Perform inferences to
  - Simplify the problem
  - Propagate consequences of assignments
  - Explain conflict btw candidate model and clauses
  - Prove unsatisfiability
- Search and Inference guide each other:
  - Search focuses Inference on conflicts
    which may facilitate proving unsatisfiability
  - Inference allows search to escape dead-end's

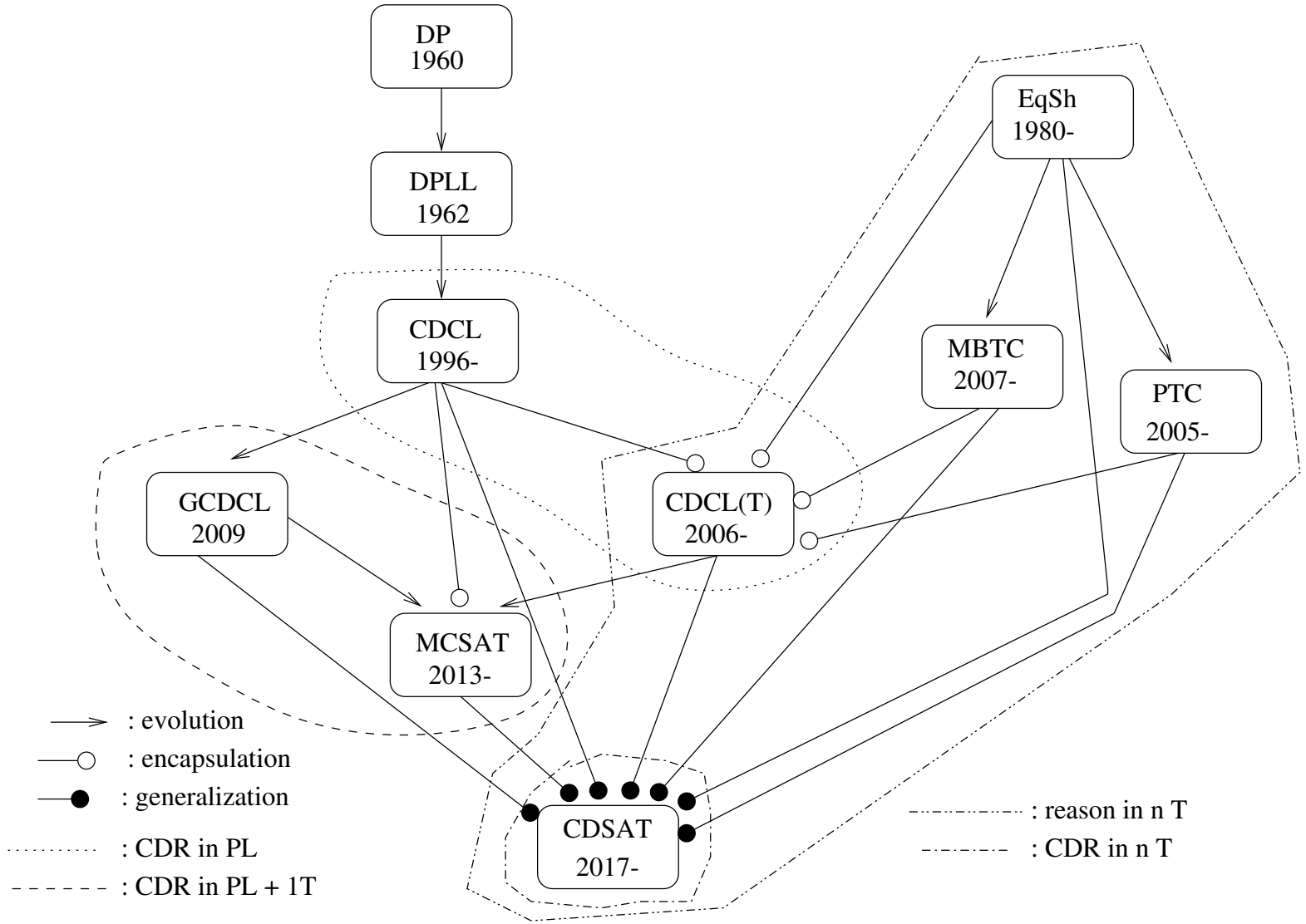
Conflict-driven reasoning

- Search for a model:
  - Decide assignments of values to terms
  - Propagate consequences of assignments ("inexpensive" inferences)
  - Conflict: contradiction
- Either reach unsatisfiability or solve conflict:
  - Explain conflict by "expensive" inferences (steps towards a possible refutation)
  - Learn generated lemma which excludes current assignment and avoids hitting same conflict
  - Solve conflict by amending assignment to satisfy lemma

Plan of these lectures

- Present the conflict-driven method CDSAT (Conflict-Driven Satisfiability) for satisfiability modulo a union of theories
  [MPB, S. Graham-Lengrand, N. Shankar: 2017-]
- Use it as a lense to revisit several SMT methods that CDSAT generalizes, introducing gradually most CDSAT features
- Illustrate an application of CDSAT to reasoning about novel theories of arrays, maps, and vectors with abstract domain that increase expressiveness capturing realistic features of these data structures

The big picture: conflict-driven reasoning procedures



CDSAT: conflict-driven reasoning in theory unions

- Decide satisfiability in a theory union $\mathcal{T} = \bigcup_{k=1}^{n} \mathcal{T}_k$
- Predicate-sharing theories
  Disjoint theories if $\simeq$ is the only shared predicate
- SMA (Satisfiability Modulo Theory and Assignment): input includes initial assignment
  - Boolean assignment: $L \leftarrow$ true (Boolean value)
  - First-order assignment: $x \leftarrow 3$ (non-Boolean value)
  - Relevant for parallelization, model interpolation, quantified satisfiability (QSMA), hybridization of local search and complete SMT methods
- Answer sat if there exists satisfying assignment including initial one, unsat otherwise

CDSAT: conflict-driven reasoning in theory unions

- Transition system: transition rules (e.g., Decide, Deduce)
- Coordinates theory modules:
  theory inference system + finite local basis
- Offers conflict-driven control accommodating also
  non-conflict-driven procedures (black-box modules)
- The modules collaborate as peers
  on a shared trail Γ containing the current assignment
- Each module offers decisions and deductions
  propagation, simplification, conflict detection, conflict explanation
- Sound, complete, terminating under suitable hypotheses
  - Finite global basis from the local ones for termination

Assignments take center stage

- Assignments of values to terms:
  $(x > 1) \leftarrow \text{false}, \quad ((x > 1) \vee (y < 0)) \leftarrow \text{true},$
  $(\text{store}(a, i, v) \simeq b) \leftarrow \text{true}, \quad y \leftarrow \sqrt{2}, \quad \text{select}(a, j) \leftarrow 3$
- For each pair term and value have the same sort
- Boolean assignments are routinely abbreviated:
  $(x > 1) \vee (y < 0)$ for $((x > 1) \vee (y < 0)) \leftarrow \text{true}$
  $\overline{x > 1}$ for $(x > 1) \leftarrow \text{false}$
- Formulas are Boolean terms (sort prop)
- Plausible assignment: does not contain $L \leftarrow \text{true}$ and $L \leftarrow \text{false}$

Assignments take center stage

- Terms and values are kept separate:
  term only on the left, value only on the right of an assignment
- select$(a, j) \leftarrow 3$ (assignment) cannot be replaced by
  $select(a, j) \simeq 3$ (literal)
  and we cannot write $select(a, 3)$:
  a value is not a term, is not in the signature
- What are values?

Theory extension

- Theory $\mathcal{T}$ with signature Σ
- Theory extension $\mathcal{T}^+$ with signature Σ^+:
  - Add new constant symbols (and possibly axioms)
  - E.g.: add a constant symbol for every number (integers, rationals, algebraic reals)
    $\sqrt{2}$ is a constant symbol interpreted as $\sqrt{2}$
  - All $\mathcal{T}$ have sort prop: all $\mathcal{T}^+$ add true and false
- Conservative extension: for Σ-formulas
  - $\mathcal{T}$-satisfiable implies $\mathcal{T}^+$-satisfiable, or, equivalently,
  - $\mathcal{T}^+$-unsatisfiable implies $\mathcal{T}$-unsatisfiable

Theory extensions provide values

- From theory $\mathcal{T}$ to theory extension $\mathcal{T}^+$
- Values in assignments are the constant symbols added by $\mathcal{T}^+$: they are called $\mathcal{T}$-values
- An assignment $\{x_1 \leftarrow \mathfrak{c}_1, \ \ldots, \ x_m \leftarrow \mathfrak{c}_m\}$ is a $\mathcal{T}$-assignment if for all i, $1 \leq i \leq m$, $\mathfrak{c}_i$ is a $\mathcal{T}$-value

Two kinds of theory extension

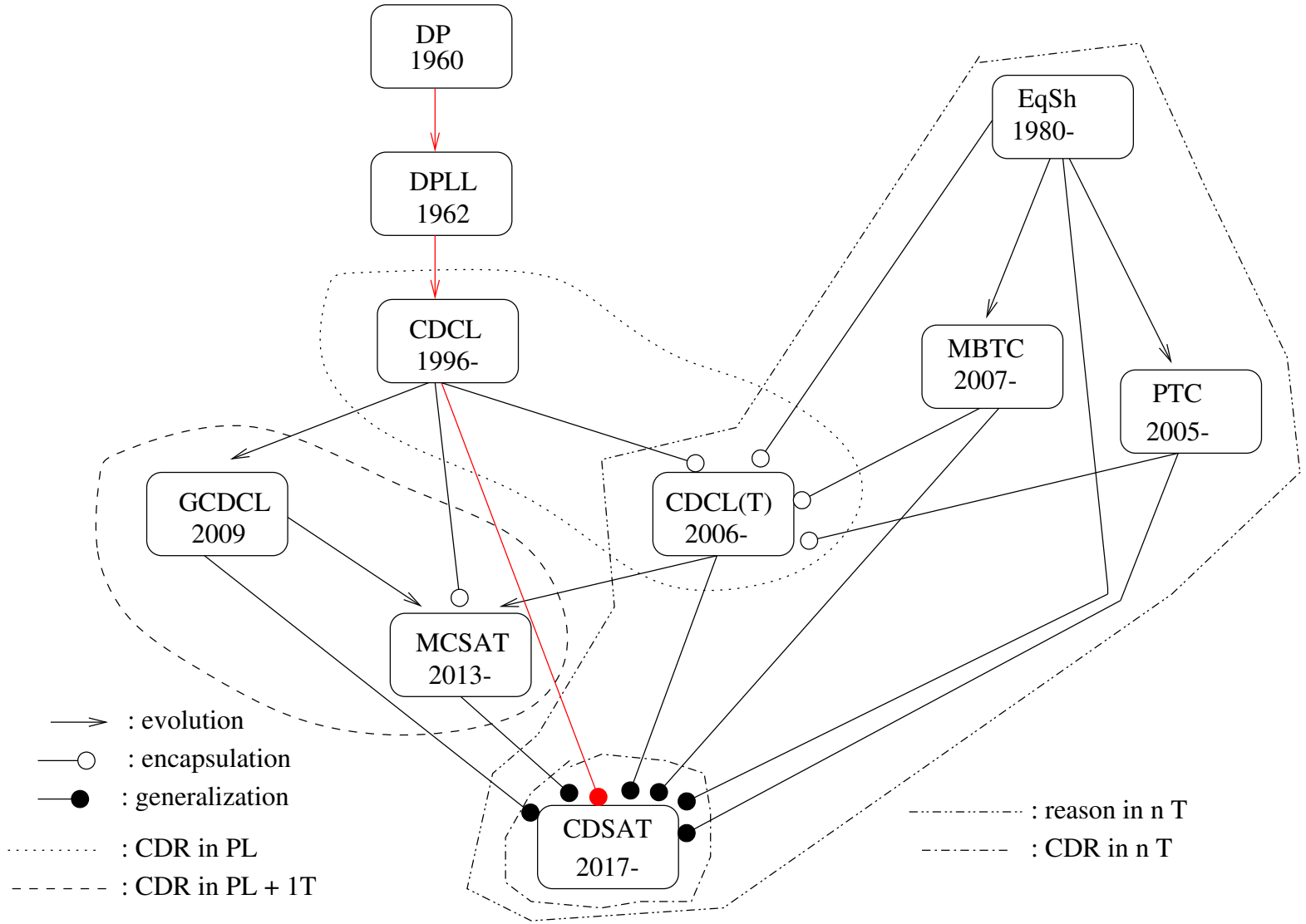
- Trivial: adds only true and false with axioms $\{\text{true}, \neg\text{false}\}$
- Countable: adds true and false with axioms $\{\text{true}, \neg\text{false}\}$ for sort prop, and a countably infinite set of $\mathcal{T}$-values for each sort s other than prop
- A countable extension may or may not add axioms, e.g.:
  - Equality (EUF^+): no axioms, the $\mathcal{T}$-values act as labels of congruence classes
  - NRA^+: the sentences true in the intended model of the reals interpreting every NRA-value as itself
    (e.g., $\sqrt{2}$ with $\sqrt{2} \cdot \sqrt{2} \simeq 2$)

CDSAT theory modules

Theory modules $\mathcal{I}_1, \ldots, \mathcal{I}_n$ for theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$

- $\mathcal{I}_k$ is a set of inference rules $J \vdash_k L$ where
  - J is a $\mathcal{T}_k$-assignment: may contain first-order assignments (a $\mathcal{T}_k$-assignment assigns $\mathcal{T}_k$-values)
  - L is a singleton Boolean assignment
- A function basis_k from sets of terms to sets of terms called local basis:
  - Given a set X of terms (e.g., the input) terms, $\text{basis}_k(X)$ is the finite set of all terms that the $\mathcal{I}_k$ inferences can generate

The big picture: propositional reasoning



Propositional satisfiability

- DP [Davis, Putnam: JACM 1960]:
  - Resolution ($C \vee L$ and $D \vee \overline{L}$ resolve to generate $C \vee D$)
  - Subsumption (L subsumes $C \vee L$, and D subsumes $D \vee \overline{L}$)
- DPLL [Davis, Putnam, Logeman, Loveland: CACM 1962]:
  - Backtracking search for a model
    partial model represented by Boolean assignments
    on a trail Γ (stack)
  - Resolution replaced by splitting on L and $\overline{L}$
  - Unit propagation: unit subsumption + unit resolution:
    if L on Γ, delete $C \vee L$, and replace $D \vee \overline{L}$ with D
  - Conflict (e.g., $\{P, \overline{P}\}$): backtrack last guess
    (e.g., from L to $\overline{L}$) or fail and return unsat if none
- CDCL [Marques Silva, Sakallah: ICCAD 1996, IEEE TOC 1999]

CDCL: Conflict-Driven Clause Learning

- Decision replaces the splitting of DPLL:
  add L to trail Γ provided $L \notin \Gamma$ and $\overline{L} \notin \Gamma$
- Every decision opens new level on trail Γ which is a stack
- Unit propagation detects
  - Implied literal L with justification $C = L_1 \vee \ldots \vee L_k \vee L$
    if $\overline{L_i} \in \Gamma$ $(1 \leq i \leq k)$ and adds L to current level
  - Conflict clause $D = Q_1 \vee \ldots \vee Q_n$ if $\overline{Q_i} \in \Gamma$ $(1 \leq i \leq n)$
- Conflict at level 0 (no decision to flip):
  fail and return unsat
- Otherwise: explain and/or solve conflict

CDCL: Conflict-Driven Clause Learning

- Explain conflict by resolving conflict clause $D = Q_1 \vee \ldots \vee Q_n$ and justification of $\overline{Q}_i$
- The generated resolvent is still a conflict clause
- Learn a resolvent C as a lemma
- A heuristic determines how many resolution steps and which lemma
- Conflict-driven backjumping replaces the backtracking of DPLL: backjump away from conflict to a state that satisfies C (backjump clause)

The first assertion clause heuristic

- Resolve until the conflict clause is an assertion clause
- Assertion clause: a conflict clause $C = L_1 \vee \ldots \vee L_k \vee L$ where only L is falsified by the current level (1UIP)
- L is the assertion literal
- Learn C
- Backjump to the smallest level such that $\overline{L}_i \in \Gamma$ $(1 \leq i \leq k)$ and L undefined
- L is an implied literal with justification C

CDCL procedure: example

$S = \{\overline{A} \vee B,\ \overline{A} \vee C \vee E,\ \overline{B} \vee D,\ \overline{C} \vee \overline{D}, A \vee \overline{B} \vee E,\ B \vee \overline{C},\ F \vee \overline{E}\}$
subset of input

1. A decision adds $\overline{F}$ to trail Γ (opening level n)
2. Unit propagation adds $\overline{E}$ with justification $F \vee \overline{E}$ (level n)
3. Two more decisions unrelated to S (levels $n+1$ and $n+2$)
4. Another decision adds A to Γ (level $n+3$)
5. Unit propagation adds B with justification $\overline{A} \vee B$
   C with justification $\overline{A} \vee C \vee E$
   D with justification $\overline{B} \vee D$ (all on level $n+3$)

CDCL procedure: example

6. Conflict: $\overline{C} \vee \overline{D}$ is conflict clause
7. Why is $\overline{D}$ false? Because D is true with justification $\overline{B} \vee D$
   Resolve $\overline{C} \vee \overline{D}$ and $\overline{B} \vee D$ to yield $\overline{B} \vee \overline{C}$
   (still a conflict clause)
8. Why is $\overline{C}$ false? Because C is true with justification $\overline{A} \vee C \vee E$
   Resolve $\overline{B} \vee \overline{C}$ and $\overline{A} \vee C \vee E$ to yield $\overline{B} \vee \overline{A} \vee E$
   (still a conflict clause)
9. Why is $\overline{B}$ false? Because B is true with justification $\overline{A} \vee B$
   Resolve $\overline{B} \vee \overline{A} \vee E$ and $\overline{A} \vee B$ to yield $\overline{A} \vee E$
   $\overline{A} \vee E$: only $\overline{A}$ is falsified by the current level $n + 3$

CDCL procedure: example

10. Learn $\overline{A} \vee E$: no model with A and $\overline{E}$
11. Backjump to the smallest level where $\overline{A}$ is undefined and E is still false: level n
    Add $\overline{A}$ to level n with justification $\overline{A} \vee E$
12. Unit propagation adds
    $\overline{B}$ with justification $A \vee \overline{B} \vee E$
    $\overline{C}$ with justification $B \vee \overline{C}$ (both on level n)
    Γ contains $\{\overline{E},\ \overline{A},\ \overline{B},\ \overline{C}\}$ which is a model of S

CDSAT emulates CDCL if Bool is the only theory in the union $\mathcal{T}$

CDSAT module for theory Bool

- $\Sigma_{\mathsf{Bool}} = \langle\ \{\mathsf{prop}\},\ \{\neg, \vee, \wedge, \simeq_{\mathsf{prop}}\}\ \rangle$
- Theory extension Bool^+ is trivial
- Evaluation: $(L_1 \leftarrow \mathfrak{b}_1, \ldots, L_m \leftarrow \mathfrak{b}_m) \vdash_{\mathsf{Bool}} L \leftarrow \mathfrak{b}$
  where each $\mathfrak{b}_i$ and $\mathfrak{b}$ is true or false
- $\neg$-elim: $\neg L \vdash_{\mathsf{Bool}} \overline{L}$ and $\overline{\neg L} \vdash_{\mathsf{Bool}} L$
- $\bigwedge$-elim: $L_1 \wedge \cdots \wedge L_m \vdash_{\mathsf{Bool}} L_i$ and $\overline{L_1 \vee \cdots \vee L_m} \vdash_{\mathsf{Bool}} \overline{L_i}$
- Unit propagation: $L_1 \vee \cdots \vee L_m, \{\overline{L_j} \mid j \neq i\} \vdash_{\mathsf{Bool}} L_i$ and $\overline{L_1 \wedge \cdots \wedge L_m}, \{L_j \mid j \neq i\} \vdash_{\mathsf{Bool}} \overline{L_i}$
- $\mathsf{basis}_{\mathsf{Bool}}(X)$: all subformulas of formulas in X
  and all their disjunctions (for clause learning)

The CDSAT trail for the Boolean case

- The trail is a sequence of assignments: $\Gamma = L_0, \ldots, L_m$
  Boolean term L: L is $L \leftarrow$ true, $\overline{L}$ is $L \leftarrow$ false
  Satisfaction of assignment: same as satisfaction of formula
- Input H_0: written as an assignment $H_0 \subseteq \Gamma$
- Two kinds of assignments:
  - Decision: ${}_{?}L$ (a guess)
  - Justified assignment: ${}_{H\vdash}L$
    justification H: a set of assignments that appear before L in Γ
    Sound: $(H_0 \cup H) \models L$
- Input assignments: ${}_{\emptyset\vdash}L$

The trail is organized in levels

$\Gamma = L_0, \ldots, L_m$

- Every decision ${}_?L$ opens a new level:
  $\text{level}_\Gamma(L_i) = 1 + max\{\text{level}_\Gamma(L_j) \mid j < i\}$ if L_i is a decision ${}_?L$
- A justified assignment ${}_{H\vdash}L$ has the level of its justification H:
  $\text{level}_\Gamma(H) = 0$ if $H = \emptyset$ (e.g., input assignment)
  $\text{level}_\Gamma(H) = max\{\text{level}_\Gamma(A) \mid A \in H\}$
  ${}_{H\vdash}L$ has level 0 also when all elements in H have level 0

CDSAT-derivation

- CDSAT-derivation: a sequence of states where
  - The initial state is a trail Γ_0 containing only the input assignment H_0, and
  - Every other state is generated from its predecessor by applying a CDSAT transition rule
- First kind of state: a trail Γ

Boolean decisions

Transition rule Decide adds to the trail a Boolean decision:

$$\text{Decide}: \Gamma \longrightarrow \Gamma, {}^{?}L$$

provided assignment L is acceptable:
$L \notin \Gamma$ and $\overline{L} \notin \Gamma$
which preserves plausibility of the assignment on the trail

Boolean deductions

- Transition rule Deduce adds to the trail a Boolean justified assignment ${}_{J}\!\vdash L$:

$$\text{Deduce}: \Gamma \longrightarrow \Gamma, {}_{J}\!\vdash L$$

if a $\mathcal{T}_k$-module has inference $J \vdash_k L$ for $J \subseteq \Gamma$ and $L \notin \Gamma$ and $\overline{L} \notin \Gamma$

- Deduce covers unit propagation when $\mathcal{T}_k$ is Bool:
$J = \{L_1 \vee \ldots \vee L_k \vee L,\ \overline{L}_1, \ldots, \overline{L}_k\}$
Note that the justification is richer than in CDCL

Conflict

- Given input assignment H_0 and trail Γ
- Conflict: an unsatisfiable subset of Γ
  $E \subseteq \Gamma$, $E \neq \emptyset$, $(H_0 \cup E) \models \bot$
- Conflict detection:
  - $J \subseteq \Gamma$
  - A $\mathcal{T}_k$-module can infer $J \vdash_k L$ but $\overline{L} \in \Gamma$
  - Then $E = J \cup \{\overline{L}\}$ is a conflict
- The notion of conflict covers that of conflict clause:
  $E = \{Q_1 \vee \ldots \vee Q_n, \ \overline{Q}_1, \ldots, \overline{Q}_n\}$
  conflict clause: negation of the cube $\bigwedge_{i=1}^{n} \overline{Q}_i$ in E
  cube: set (conjunction) of literals

Conflict and failure

- Level of a conflict E: $\text{level}_\Gamma(E) = max\{\text{level}_\Gamma(L) \mid L \in E\}$
- $\text{level}_\Gamma(E) = 0$:
  since $E \neq \emptyset$, $\text{level}_\Gamma(L) = 0$ for all $L \in E$:
  all $L \in E$ are input assignments or justified assignments derived from input assignments
  no decision to undo, nowhere to backjump to
  transition rule Fail returns unsat:

$$\text{Fail}: \Gamma \longrightarrow \text{unsat}$$

Conflict and conflict solving

- Level of a conflict E: $\text{level}_\Gamma(E) = max\{\text{level}_\Gamma(L) \mid L \in E\}$
- $\text{level}_\Gamma(E) > 0$: transition rule ConflictSolve transfers control to the conflict state rules and returns the trail they produce:

$$\text{ConflictSolve}: \Gamma \longrightarrow \Gamma'$$

if $\langle \Gamma; E \rangle \Longrightarrow^* \Gamma'$

- $\longrightarrow$ for trail rule transition
  $\Longrightarrow$ for conflict state rule transition
- Second kind of state: conflict state $\langle \Gamma; E \rangle$

Conflict state rule Resolve

- Conflict state: $\langle \Gamma; E \uplus \{L\} \rangle$ and ${}_{H}\vdash L \in \Gamma$
- Transition rule Resolve explains the conflict by replacing ${}_{H}\vdash L$ with its justification H:

$$\text{Resolve}: \langle \Gamma; E \uplus \{L\} \rangle \Longrightarrow \langle \Gamma; E \cup H \rangle$$

- Explanation: why L? Because of H
- $E \uplus H$ is still a conflict:
  $(H_0 \cup E \uplus \{L\}) \models \bot$ and $(H_0 \cup H) \models L$ imply
  $(H_0 \cup E \cup H) \models \bot$

Resolve covers propositional resolution

- Conflict $E_1 = \{Q_1 \vee \ldots \vee Q_n,\ \overline{Q}_1, \ldots, \overline{Q}_n\}$
  Conflict clause in E_1: $Q_1 \vee \ldots \vee Q_n$
- Say $\overline{Q}_1$ has justification $\{\overline{Q}_1 \vee C,\ \overline{C}\}$ where C is a disjunction of literals and $\overline{C}$ is the set of their flips
- Resolve applied to $\overline{Q}_1$ in E_1 yields
  $E_2 = \{Q_1 \vee \ldots \vee Q_n,\ \overline{Q}_1 \vee C,\ \overline{C},\ \overline{Q}_2, \ldots, \overline{Q}_n\}$
  Conflict clause in E_2: $C \vee Q_2 \vee \ldots \vee Q_n$

Outstanding assignment and restriction of a trail

- Given conflict $E \uplus \{L\}$ where $E \neq \emptyset$
  assignment L is outstanding in $E \uplus \{L\}$
  if $\text{level}_\Gamma(L) > \text{level}_\Gamma(E)$
- Given trail $\Gamma = L_0, \ldots, L_m$
  the restriction $\Gamma^{\leq m}$ to level m
  contains all L_i such that $\text{level}_\Gamma(L_i) \leq m$
  in the same order in which they appear in Γ

Conflict state rule Backjump

- Conflict state: $\langle \Gamma; E \uplus \{L\} \rangle$ where $E \neq \emptyset$, assignment L is outstanding in $E \uplus \{L\}$, and $\text{level}_\Gamma(E) = m$
- Transition rule Backjump solves the conflict by jumping back to level m and adding ${}_{E\vdash}\overline{L}$ to the trail:

$$\text{Backjump}: \langle \Gamma; E \uplus \{L\} \rangle \Longrightarrow \Gamma^{\leq m}, {}_{E\vdash}\overline{L}$$

- $(H_0 \cup E \uplus \{L\}) \models \bot$ implies $(H_0 \cup E) \models \overline{L}$: justified assignment ${}_{E\vdash}\overline{L}$ is sound
- $\overline{L}$ is a learned lemma: no model can contain L as long as $H_0 \cup E$ is on the trail

First assertion clause heuristic with Backjump

- Apply Resolve until conflict has the form $E \uplus H$ where
  - $H = \{\overline{Q}_1, \ldots, \overline{Q}_n\}$
  - $\overline{Q}_1$ is outstanding in $E \uplus H$
  - $E \uplus H$ is the first conflict with an outstanding assignment
  - $\text{level}_\Gamma(\overline{Q}_1)$ is the current one (maximum on the trail)
  - Q_1 is the assertion literal in the first assertion clause $Q_1 \vee \ldots \vee Q_n$
- Let $H' = \{\overline{Q}_2, \ldots, \overline{Q}_n\}$
- Backjump to $\text{level}_\Gamma(E \uplus H')$ adding $_{E \uplus H' \vdash} Q_1$

The CDCL example in CDSAT

$S = \{\overline{A} \vee B,\ \overline{A} \vee C \vee E,\ \overline{B} \vee D,\ \overline{C} \vee \overline{D}, A \vee \overline{B} \vee E,\ B \vee \overline{C},\ F \vee \overline{E}\}$
subset of input

1. Decide adds $?\overline{F}$ to trail Γ opening level n
2. Deduce adds $\overset{J}{\vdash}\overline{E}$ with $J = \{F \vee \overline{E},\ \overline{F}\}$ to level n
   since $\{F \vee \overline{E},\ \overline{F}\} \vdash_{\text{Bool}} \overline{E}$
3. Two more Decide create levels $n+1$ and $n+2$
4. Another Decide adds $?A$ opening level $n+3$
5. Deduce adds to level $n+3$
   $\overset{H}{\vdash} B$ with $H = \{\overline{A} \vee B,\ A\}$
   $\overset{I}{\vdash} C$ with $I = \{\overline{A} \vee C \vee E,\ \overline{E},\ A\}$
   $\overset{K}{\vdash} D$ with $K = \{\overline{B} \vee D,\ B\}$

The CDCL example in CDSAT

6. $\{\overline{C} \vee \overline{D},\ C\} \vdash_{\text{Bool}} \overline{D}$ but ${}_{K}\!\vdash D \in \Gamma$
   Conflict: $E_0 = \{\overline{C} \vee \overline{D},\ C,\ D\}$
   /* $\overline{C} \vee \overline{D}$ conflict clause, not assertion clause */
7. E_0 contains literals C and D of max level $(n+3)$
   Resolve: $E_1 = \{\overline{C} \vee \overline{D},\ C,\ \overline{B} \vee D,\ B\}$
   /* $\overline{C} \vee \overline{D}$ and $\overline{B} \vee D$ yield $\overline{B} \vee \overline{C}$ (not assertion clause) */
8. E_1 contains literals C and B of max level $(n+3)$
   Resolve: $E_2 = \{\overline{C} \vee \overline{D},\ \overline{A} \vee C \vee E,\ \overline{E},\ A,\ \overline{B} \vee D,\ B\}$
   /* $\overline{B} \vee \overline{C}$ and $\overline{A} \vee C \vee E$ yield $\overline{B} \vee \overline{A} \vee E$ (not assertion clause) */
9. E_2 contains literals A and B of max level $(n+3)$
   Resolve: $E_3 = \{\overline{C} \vee \overline{D},\ \overline{A} \vee C \vee E,\ \overline{E},\ A,\ \overline{B} \vee D,\ \overline{A} \vee B\}$
   /* $\overline{B} \vee \overline{A} \vee E$ and $\overline{A} \vee B$ yield $\overline{A} \vee E$ (assertion clause) */

The CDCL example in CDSAT with Backjump

$E_3 = \{\overline{C} \vee \overline{D},\ \overline{A} \vee C \vee E,\ \overline{E},\ A,\ \overline{B} \vee D,\ \overline{A} \vee B\}$
A has level $n+3$ and is outstanding in E_3,
$\overline{E}$ has level n, and the rest has level 0

10. Backjump to level n adding ${}_{G}\vdash(\overline{A})$ with
    $G = \{\overline{C} \vee \overline{D},\ \overline{A} \vee C \vee E,\ \overline{E},\ \overline{B} \vee D,\ \overline{A} \vee B\}$
11. Deduce adds to level n:
    ${}_{N}\vdash \overline{B}$ with $N = \{A \vee \overline{B} \vee E,\ \overline{A},\ \overline{E}\}$
    ${}_{P}\vdash \overline{C}$ with $P = \{B \vee \overline{C},\ \overline{B}\}$
    Γ contains $\{\overline{E},\ \overline{A},\ \overline{B},\ \overline{C}\}$ model of S

However, the assertion clause $\overline{A} \vee E$ is not learned

The CDSAT transition system

For the Boolean case (only Boolean assignments):

- Basic transition system:
  - Trail rules: Decide, Deduce, Fail, ConflictSolve
  - Conflict state rules: Resolve, Backjump
- Standard transition system:
  - Trail rules: Decide, Deduce, Fail, ConflictSolve
  - Conflict state rules: Resolve, LearnBackjump

LearnBackjump generalizes Backjump by learning any clause $L_1 \vee \ldots \vee L_k$ which is the flip of a cube $\{\overline{L}_1, \ldots, \overline{L}_k\}$ in the conflict

Conflict state rule LearnBackjump

- Conflict state: $\langle \Gamma; E \uplus H \rangle$ where $E \neq \emptyset$ and $H = \{\overline{L}_1, \ldots, \overline{L}_k\}$
- Let $L = L_1 \vee \ldots \vee L_k$: clausal form of H
- Transition rule LearnBackjump solves the conflict by jumping back to a level m such that $\text{level}_\Gamma(E) \leq m < \text{level}_\Gamma(H)$ and adding ${}_{E\vdash}L$, provided $L \notin \Gamma$, $\overline{L} \notin \Gamma$:

$$\text{LearnBackjump}: \langle \Gamma; E \uplus H \rangle \Longrightarrow \Gamma^{\leq m}, {}_{E\vdash}L$$

- $(H_0 \cup E \uplus H) \models \bot$ implies $(H_0 \cup E) \models L$: justified assignment ${}_{E\vdash}L$ is sound and L is a learned lemma: every model must satisfy L as long as $H_0 \cup E$ is on the trail

LearnBackjump generalizes Backjump

- Given conflict state $\langle \Gamma; E \uplus H \rangle$, if
  - H is a singleton $\{\overline{L}\}$ so that the clausal form of H is simply L
  - $\overline{L}$ is outstanding in $E \uplus \{\overline{L}\}$: $\mathsf{level}_\Gamma(\overline{L}) > \mathsf{level}_\Gamma(E)$
  - The destination level m is $\mathsf{level}_\Gamma(E)$
- Then LearnBackjump simulates Backjump

First assertion clause heuristic with LearnBackjump

Apply Resolve until conflict has the form $E \uplus H$ where

- $H = \{\overline{Q}_1, \ldots, \overline{Q}_n\}$
- The clausal form of H is the first assertion clause $C = Q_1 \vee \ldots \vee Q_n$ with assertion literal Q_1
- $\overline{Q}_1$ is outstanding in $E \uplus H$
- $E \uplus H$ is the first conflict with an outstanding literal
- $\text{level}_\Gamma(\overline{Q}_1)$ is the current one (maximum on the trail)

Let $H' = \{\overline{Q}_2, \ldots, \overline{Q}_n\}$

First assertion clause heuristic with LearnBackjump

- LearnBackjump to level $m = \text{level}_\Gamma(E \uplus H')$ adding $_{E}\vdash C$
- The condition $\text{level}_\Gamma(E) \leq m < \text{level}_\Gamma(H)$ is met:
  - $m < \text{level}_\Gamma(H)$ as $\text{level}_\Gamma(H) = \text{level}_\Gamma(\overline{Q_1})$ which is the current one (maximum on the trail)
  - $m \geq \text{level}_\Gamma(E)$ as $E \uplus H'$ is a superset of E
- Apply Deduce to add $_{\{C\}\uplus H'}\vdash Q_1$ supported by unit propagation inference $\{C\} \uplus H' \vdash_{\text{Bool}} Q_1$

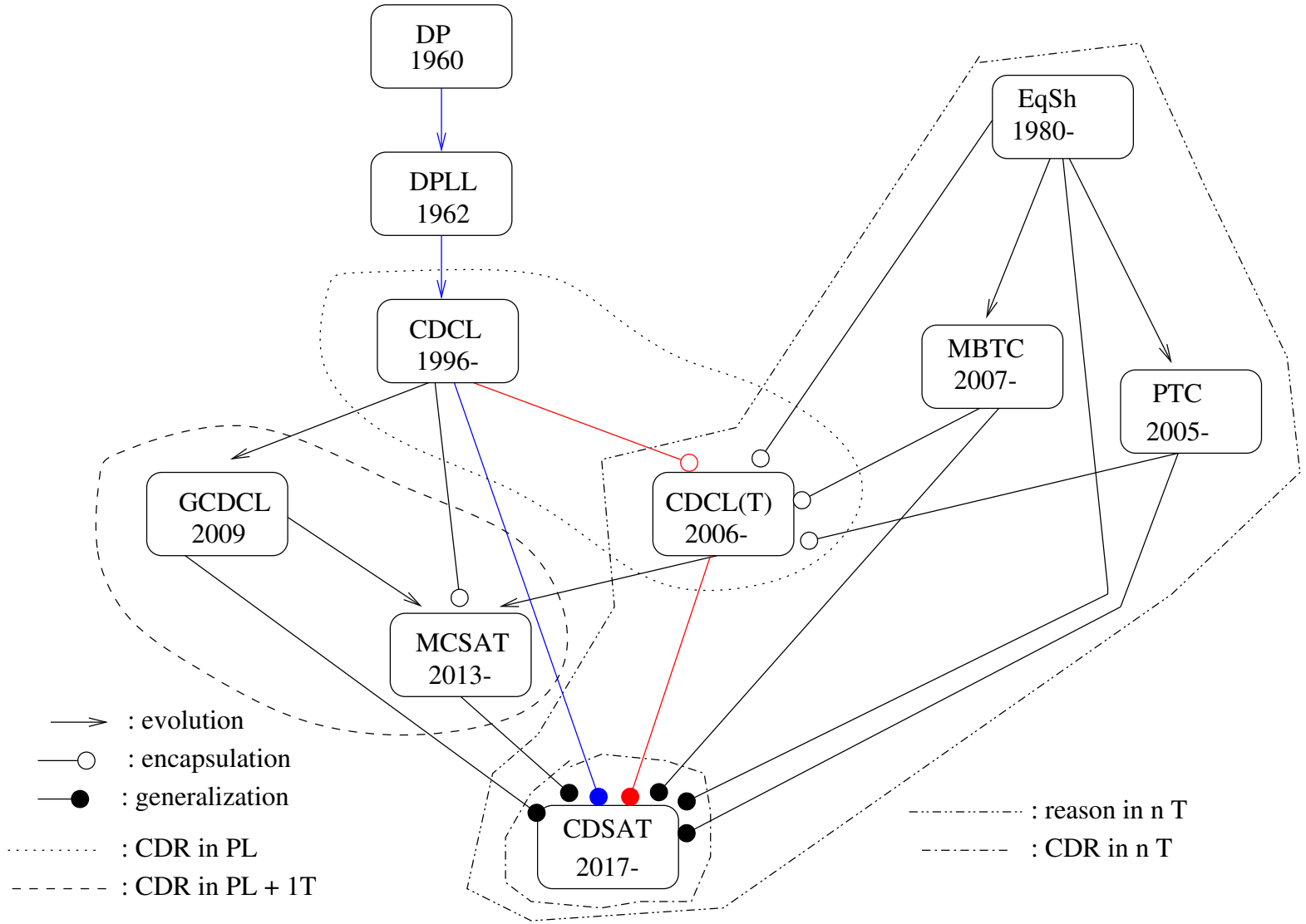
LearnBackjump allows other heuristics:
e.g., learn and restart when $m = \text{level}_\Gamma(E) = 0$

The CDCL example in CDSAT with LearnBackjump

$E_3 = \{\overline{C} \vee \overline{D},\ \overline{A} \vee C \vee E,\ \overline{E},\ A,\ \overline{B} \vee D,\ \overline{A} \vee B\}$
A has level $n+3$ and is outstanding in E_3,
$\overline{E}$ has level n, and the rest has level 0

10. LearnBackjump to level n adding ${}_{G}\vdash(\overline{A} \vee E)$ with $G = \{\overline{C} \vee \overline{D},\ \overline{A} \vee C \vee E,\ \overline{B} \vee D,\ \overline{A} \vee B\}$
11. Deduce adds ${}_{M}\vdash\overline{A}$ to level n with $M = \{\overline{A} \vee E,\ \overline{E}\}$
12. Deduce adds to level n:
    ${}_{N}\vdash\overline{B}$ with $N = \{A \vee \overline{B} \vee E,\ \overline{A},\ \overline{E}\}$
    ${}_{P}\vdash\overline{C}$ with $P = \{B \vee \overline{C},\ \overline{B}\}$
    Γ contains $\{\overline{E},\ \overline{A},\ \overline{B},\ \overline{C}\}$ model of S

The big picture: from SAT to SMT



CDCL($\mathcal{T}$): from SAT to SMT for one theory

DPLL($\mathcal{T}$) later renamed CDCL($\mathcal{T}$) for $\mathcal{T}$ a single theory
[Nieuwenhuis, Oliveras, Tinelli: JACM 2006]

- CDCL + decision procedure for the $\mathcal{T}$-satisfiability of a set of $\mathcal{T}$-literals
- CDCL works on a propositional abstraction: input $\mathcal{T}$-atoms replaced by propositional variables

CDCL($\mathcal{T}$): from SAT to SMT for one theory

Let $\{L_1, \ldots, L_n\} \subseteq \Gamma$ and $C = \overline{L}_1 \vee \ldots \vee \overline{L}_n$

The $\mathcal{T}$-sat procedure contributes only:

- $\mathcal{T}$-conflict detection:
  if $\{L_1, \ldots, L_n\} \models_{\mathcal{T}} \bot$
  C is a conflict clause
- $\mathcal{T}$-propagation:
  if $\{L_1, \ldots, L_n\} \models_{\mathcal{T}} L$ and $L \notin \Gamma$ and $\overline{L} \notin \Gamma$
  add L to Γ with justification $C \vee L$:
  the atom of L must be an input atom (i.e., not new)
  already mapped to a propositional variable

CDCL($\mathcal{T}$): from SAT to SMT for one theory

- $\mathcal{T}$-sat procedure integrated as a black-box that only raises a flag if it detects an inconsistency in the propositional model built by CDCL ignoring the theory:
  - $\mathcal{T}$-conflict: $\{L_1, \ldots, L_n\} \models_{\mathcal{T}} \bot$
    $\overline{L}_1 \vee \ldots \vee \overline{L}_n$ is $\mathcal{T}$-valid
  - $\mathcal{T}$-propagation: $\{L_1, \ldots, L_n, \overline{L}\} \models_{\mathcal{T}} \bot$
    $\overline{L}_1 \vee \ldots \vee \overline{L}_n \vee L$ is $\mathcal{T}$-valid

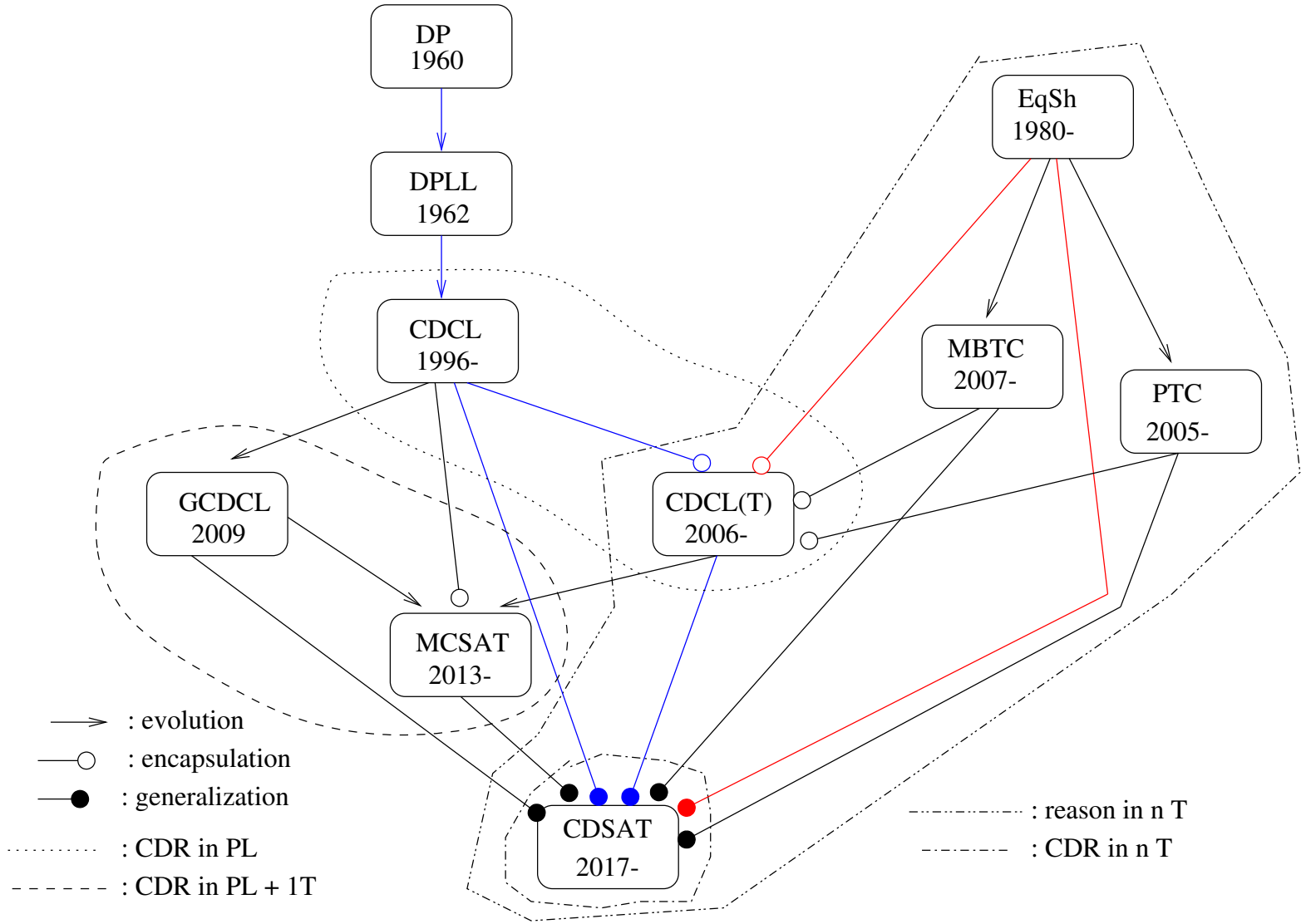
  Never deduce anything that excludes a $\mathcal{T}$-model
  but is not a $\mathcal{T}$-valid clause or a logical consequence of the input by propositional resolution
- Model search, trail, conflict explanation, conflict-driven reasoning remain propositional

CDSAT generalizes CDCL($\mathcal{T}$)

- Consider a theory union whose members are Bool and $\mathcal{T}$
- Theory modules:
  - Bool-module
  - Black-box $\mathcal{T}$-module:
    - Only one inference rule: $L_1, \ldots, L_m \vdash \bot$
    - That invokes the $\mathcal{T}$-procedure to detect that a set of literals is $\mathcal{T}$-unsatisfiable

CDSAT can use a black-box $\mathcal{T}$-module
whenever a theory $\mathcal{T}$ is not involved in conflict-driven reasoning

The big picture: theory combination



Classical approach to theory combination: equality sharing

Equality sharing mostly known as Nelson-Oppen method

[Nelson, Oppen: ACM TOPLAS 1979], [Nelson: ATP After 25 Years 1983], [Tinelli, Harandi: FroCoS 1996]

- $\mathcal{T} = \bigcup_{k=1}^{n} \mathcal{T}_k$: disjoint theories (share $\simeq$ and sorts)
- Decision procedures for $\mathcal{T}_k$-satisfiability of set of $\mathcal{T}_k$-literals
- Stably infinite: $\mathcal{T}_k$-model with infinite cardinality
- Get decision procedure for $\mathcal{T}$-satisfiability of set of $\mathcal{T}$-literals
- Combination of decision procedures as black-boxes
- By disjointness, agreement is needed only on:
  - Cardinalities of shared sorts: by stable infiniteness
  - Interpretation of equality

Equality sharing: separation

- Input set S: $\mathcal{T}$-literals mix symbols from the $\mathcal{T}_k$'s signatures
- Separate S into sets S_k of $\mathcal{T}_k$-literals $(1 \leq k \leq n)$
- The sets S_k have only $\simeq$ and variables in common

Example of separation

- $\{x \leq y,\ y \leq (x + g(x)),\ P(h(x) - h(y)),\ \neg P(0),\ g(x) \simeq 0\}$
- Theory combination: LIA and EUF sharing sorts Z and prop
- EUF-procedure: congruence closure algorithm
  [Nelson, Oppen: JACM 1980] [Downey, Sethi, Tarjan: JACM 1980]
- Standard elimination of predicate symbols other than $\simeq$:
  $S = \{x \leq y,\ y \leq (x + g(x)),$
  $f_p(h(x) - h(y)) \simeq \bullet,\ f_p(0) \not\simeq \bullet,\ g(x) \simeq 0\}$

Example of separation

$S = \{x \leq y,\ y \leq (x + g(x)),$
$f_p(h(x) - h(y)) \simeq \bullet,\ f_p(0) \not\simeq \bullet,\ g(x) \simeq 0\}$

Separation:

- $S_{\mathsf{LIA}} = \{x \leq y,\ y \leq x + w_1,\ w_3 - w_4 \simeq w_2,\ w_5 \simeq 0,\ w_1 \simeq 0\}$
- $S_{\mathsf{EUF}} =$
  $\{g(x) \simeq w_1,\ f_p(w_2) \simeq \bullet,\ h(x) \simeq w_3,\ h(y) \simeq w_4,\ f_p(w_5) \not\simeq \bullet\}$
- Set of shared variables: $\{x,\ y,\ w_1,\ w_2,\ w_3,\ w_4,\ w_5\}$

Equality sharing: the reduction

- Reduce the $\mathcal{T}$-sat problem to $\mathcal{T}_k$-sat problems
- S is $\mathcal{T}$-sat iff $\bigcup_{k=1}^{n} S_k$ is $\mathcal{T}$-sat
- Arrangement α: represents a partition of the set of shared variables
- Conjunction $\alpha = \alpha^+ \wedge \alpha^-$ where
  - $(u \simeq v) \in \alpha^+$ if u and v in the same class
  - $(u \not\simeq v) \in \alpha^-$ otherwise
- Combination theorem:
  $\bigcup_{k=1}^{n} S_k$ is $\mathcal{T}$-sat iff $\exists\ \alpha$ s.t. $S_k \wedge \alpha$ is $\mathcal{T}_k$-sat $(1 \leq k \leq n)$

Convex theories

A theory $\mathcal{T}_k$ is convex if for all conjunctions H of $\mathcal{T}_k$-literals $H \models_{\mathcal{T}_k} \bigvee_{i=1}^{n} u_i \simeq v_i$ implies $\exists j,\ 1 \leq j \leq n,\ H \models_{\mathcal{T}_k} u_j \simeq v_j$.

Meaning: $H \models_{\mathcal{T}_k} \bigvee_{i=1}^{n} u_i \simeq v_i$ means that for all $\mathcal{T}_k$-models $\mathcal{M}$ of H there exists an i, $1 \leq i \leq n$, such that $\mathcal{M} \models_{\mathcal{T}_k} u_i \simeq v_i$, e.g.:
$\mathcal{M}_1 \models_{\mathcal{T}_k} u_1 \simeq v_1$,
$\mathcal{M}_2 \models_{\mathcal{T}_k} u_2 \simeq v_2$,
$\mathcal{M}_3 \models_{\mathcal{T}_k} u_1 \simeq v_1$, and so on.

Convexity means that there exists a j, $1 \leq j \leq n$, such that for all $\mathcal{T}_k$-models $\mathcal{M}$ of H it is $\mathcal{M} \models_{\mathcal{T}_k} u_j \simeq v_j$.

Equality sharing: build arrangement (convex theories)

$\mathcal{E}$: shared set of equalities between shared variables

- $\mathcal{E}_0 = \emptyset$
- $\mathcal{E}_i = \mathcal{E}_{i-1} \cup \{u \simeq v\}$ if a $\mathcal{T}_k$-procedure deduces $u \simeq v$ from $S_k \cup \mathcal{E}_{i-1}$
  (every $\mathcal{T}_k$-procedure must be able to deduce all entailed equalities)
- If a $\mathcal{T}_k$-sat procedure deduces $\bot$ from $S_k \cup \mathcal{E}_i$ for some i:
  Return unsat (S is $\mathcal{T}$-unsat)
- If neither $\bot$ nor additional equalities arise
  (i.e., $\mathcal{E}_q = \mathcal{E}_{q-1}$ for some q): let $\alpha^+ = \mathcal{E}_q$
  let α^- contain $u \neq v$ for all (u, v) not in $\mathcal{E}_q$
  Return sat (S is $\mathcal{T}$-sat)

Equality sharing: build arrangement (non-convex theories)

- $\mathcal{T}_k$ not convex: $\mathcal{T}_k$-procedure deduces $\bigvee_{j=1}^{m} u_j \simeq v_j$
  (every $\mathcal{T}_k$-procedure must be capable of deducing all entailed disjunctions of equalities)
- $\bigvee_{j=1}^{m} u_j \simeq v_j$ is subject to splitting
- The $\mathcal{T}$-procedure (implementing the equality sharing method) calls itself recursively on each subproblem obtained by adding $u_j \simeq v_j$ to the current $\mathcal{E}_i$
- If a branch returns sat then return sat (S is $\mathcal{T}$-sat)
- If all branches return unsat then return unsat (S is $\mathcal{T}$-unsat)

Equality sharing: example

- $\{x \leq y,\ y \leq (x + g(x)),\ P(h(x) - h(y)),\ \neg P(0),\ g(x) \simeq 0\}$
- Theory combination: LIA and EUF sharing sorts Z and prop
- EUF-procedure: congruence closure algorithm
  [Nelson, Oppen: JACM 1980] [Downey, Sethi, Tarjan: JACM 1980]
- Standard elimination of predicate symbols other than $\simeq$:
  $S = \{x \leq y,\ y \leq (x + g(x)),$
  $f_p(h(x) - h(y)) \simeq \bullet,\ f_p(0) \not\simeq \bullet,\ g(x) \simeq 0\}$

Equality sharing: example

$S = \{x \leq y,\ y \leq (x + g(x)),$
$f_p(h(x) - h(y)) \simeq \bullet,\ f_p(0) \not\simeq \bullet,\ g(x) \simeq 0\}$

Separation:

- $S_{\mathsf{LIA}} = \{x \leq y,\ y \leq x + w_1,\ w_3 - w_4 \simeq w_2,\ w_5 \simeq 0,\ w_1 \simeq 0\}$
- $S_{\mathsf{EUF}} =$
  $\{g(x) \simeq w_1,\ f_p(w_2) \simeq \bullet,\ h(x) \simeq w_3,\ h(y) \simeq w_4,\ f_p(w_5) \not\simeq \bullet\}$
- Set of shared variables: $\{x,\ y,\ w_1,\ w_2,\ w_3,\ w_4,\ w_5\}$

Equality sharing: example

$S_{\mathsf{LIA}} = \{x \leq y,\ y \leq x + w_1,\ w_3 - w_4 \simeq w_2,\ w_5 \simeq 0,\ w_1 \simeq 0\}$
$S_{\mathsf{EUF}} =$
$\{g(x) \simeq w_1,\ f_p(w_2) \simeq \bullet,\ h(x) \simeq w_3,\ h(y) \simeq w_4,\ f_p(w_5) \not\simeq \bullet\}$

$\mathcal{E}_0 = \emptyset$

1. LIA-procedure: from $w_5 \simeq 0$ and $w_1 \simeq 0$ get $w_1 \simeq w_5$
   $\mathcal{E}_1 = \{w_1 \simeq w_5\}$
2. LIA-procedure: from $y \leq x + w_1$ and $w_1 \simeq 0$ get $y \leq x$
3. LIA-procedure: from $x \leq y$ and $y \leq x$ get $x \simeq y$
   $\mathcal{E}_2 = \{w_1 \simeq w_5,\ x \simeq y\}$

Equality sharing: example

$S_{\mathsf{LIA}} = \{x \leq y,\ y \leq x + w_1,\ w_3 - w_4 \simeq w_2,\ w_5 \simeq 0,\ w_1 \simeq 0\}$
$S_{\mathsf{EUF}} =$
$\{g(x) \simeq w_1,\ f_p(w_2) \simeq \bullet,\ h(x) \simeq w_3,\ h(y) \simeq w_4,\ f_p(w_5) \not\simeq \bullet\}$

$\mathcal{E}_2 = \{w_1 \simeq w_5,\ x \simeq y\}$

4. EUF-procedure: from $x \simeq y$ get $h(x) \simeq h(y)$
5. EUF-procedure: from $h(x) \simeq h(y)$, $h(x) \simeq w_3$, and $h(y) \simeq w_4$ get $w_3 \simeq w_4$
   $\mathcal{E}_3 = \{w_1 \simeq w_5,\ x \simeq y,\ w_3 \simeq w_4\}$
6. LIA-procedure: from $w_3 \simeq w_4$ get $w_3 - w_4 \simeq 0$
7. LIA-procedure: from $w_3 - w_4 \simeq 0$ and $w_3 - w_4 \simeq w_2$ get $w_2 \simeq 0$

Equality sharing: example

$S_{\mathsf{LIA}} = \{x \leq y,\ y \leq x + w_1,\ w_3 - w_4 \simeq w_2,\ w_5 \simeq 0,\ w_1 \simeq 0\}$
$S_{\mathsf{EUF}} =$
$\{g(x) \simeq w_1,\ f_p(w_2) \simeq \bullet,\ h(x) \simeq w_3,\ h(y) \simeq w_4,\ f_p(w_5) \not\simeq \bullet\}$

$\mathcal{E}_3 = \{w_1 \simeq w_5,\ x \simeq y,\ w_3 \simeq w_4\}$

8. LIA-procedure: from $w_2 \simeq 0$ and $w_1 \simeq 0$ get $w_2 \simeq w_1$
   $\mathcal{E}_4 = \{w_1 \simeq w_5,\ x \simeq y,\ w_3 \simeq w_4,\ w_2 \simeq w_1\}$
9. LIA-procedure: from $w_2 \simeq 0$ and $w_5 \simeq 0$ get $w_2 \simeq w_5$
   $\mathcal{E}_5 = \{w_1 \simeq w_5,\ x \simeq y,\ w_3 \simeq w_4,\ w_2 \simeq w_1,\ w_2 \simeq w_5\}$
10. EUF-procedure: from $w_2 \simeq w_5$ get $f_p(w_2) \simeq f_p(w_5)$

Equality sharing: example

$S_{\mathsf{LIA}} = \{x \leq y,\ y \leq x + w_1,\ w_3 - w_4 \simeq w_2,\ w_5 \simeq 0,\ w_1 \simeq 0\}$
$S_{\mathsf{EUF}} =$
$\{g(x) \simeq w_1,\ f_p(w_2) \simeq \bullet,\ h(x) \simeq w_3,\ h(y) \simeq w_4,\ f_p(w_5) \not\simeq \bullet\}$

$\mathcal{E}_5 = \{w_1 \simeq w_5,\ x \simeq y,\ w_3 \simeq w_4,\ w_2 \simeq w_1,\ w_2 \simeq w_5\}$

11. EUF-procedure: from $f_p(w_2) \simeq f_p(w_5)$ and $f_p(w_2) \simeq \bullet$ get $f_p(w_5) \simeq \bullet$
12. EUF-procedure: from $f_p(w_5) \simeq \bullet$ and $f_p(w_5) \not\simeq \bullet$ get $\bot$ and return unsat

Equality sharing is not conflict-driven

- Combining theories by combining procedures
- $\mathcal{T}_k$-procedures combined as black-boxes sharing only equalities btw shared variables
- Generation of all entailed (disjunctions of) equalities resembles saturation (equality sharing can be emulated by superposition for theories that satisfy certain hypotheses)

CDCL($\mathcal{T}$) and equality sharing

CDCL($\mathcal{T}$) where $\mathcal{T}$-procedure is equality sharing combination
[Barrett et al.: LPAR 2006], [Krstić, Goel: FroCoS 2007]

- $\mathcal{T}$-procedure sends (propositional abstraction of) $\bigvee_{j=1}^{m} u_j \simeq v_j$ to CDCL
- Reasoning about disjunction is entrusted to CDCL
- Case $u_j \simeq v_j$ is considered when CDCL puts it on the trail (instance of splitting on demand)
- Sole new (i.e., non-input) literals in CDCL($\mathcal{T}$): (propositional abstractions of) equalities btw shared variables
- Model search, trail, conflict explanation, conflict-driven reasoning remain propositional

CDSAT can emulate equality sharing

Towards how CDSAT can emulate equality sharing:

1. What CDSAT does for separation
2. More about CDSAT theory modules: equality inferences
3. The trail remains Boolean (equalities are Boolean terms) but is shared by more than one theory module

CDSAT: foreign terms as variables

$\lhd$: proper subterm ordering
$top(t)$: top symbol of term t

- A symbol that is in $\mathcal{T}_j$'s signature but not in $\mathcal{T}_k$'s signature ($k \neq j$) is foreign to $\mathcal{T}_k$ or $\mathcal{T}_k$-foreign
- A term t is $\mathcal{T}_k$-foreign if $top(t)$ is
- Each $\mathcal{T}_k$ sees as a variable a $\lhd$-maximal occurrence of a $\mathcal{T}_k$-foreign term

Why $\lhd$-maximal occurrences?

Example:

- Theory $\mathcal{T}_k$
- Symbol f is in $\mathcal{T}_k$'s signature and g is not
- Symbols g is $\mathcal{T}_k$-foreign and f is not
- Term $f(g(x), g(g(y)))$
- $\mathcal{T}_k$ sees $g(x)$ and $g(g(y))$ as variables, not $g(y)$
- $g(y)$ is not visible once $g(g(y))$ is seen as a variable

CDSAT: shared terms

- Given: an assignment H
- The set of shared terms $\mathcal{V}_{\text{sh}}(H)$ consists of:
  - Assigned terms
  - Foreign terms (seen as variables by at least one theory)
  - Shared variables including both
    - Original variables that are shared and
    - Foreign terms that are shared variables because they are foreign to more than one theory

The separation example done by CDSAT

- $H = \{x \leq y,\ y \leq (x + g(x)),\ p(h(x) - h(y)),\ \overline{p(0)},\ g(x) \simeq 0\}$
- Theory union: LIA $\cup$ EUF sharing sorts Z and prop
- EUF-module: abstraction of the congruence closure algorithm
- EUF: $\leq$, $+$, $-$, and 0 are foreign symbols
  $x \leq y$ and $y \leq (x + g(x))$ seen as variables of sort prop
  $h(x) - h(y)$ and 0 seen as variables of sort Z
- LIA: g, p, and h are foreign symbols
  $g(x)$ seen as a variable of sort Z
  $p(h(x) - h(y))$ and $p(0)$ seen as variables of sort prop

The separation example done by CDSAT

- Shared terms: assigned terms + foreign terms + shared vars
- Assigned terms:
  $\{x \leq y,\ y \leq (x + g(x)),\ p(h(x) - h(y)),\ p(0),\ g(x) \simeq 0\}$
- Foreign terms: EUF-foreign + LIA-foreign
  EUF-foreign: $\{x \leq y,\ y \leq (x + g(x)),\ h(x) - h(y),\ 0\}$
  LIA-foreign: $\{g(x),\ p(h(x) - h(y)),\ p(0)\}$
- Shared vars: $\{x\}$: LIA sees $\{x,\ y\}$; EUF sees $\{x\}$ as all occurrences of y are below EUF-foreign terms
- Shared terms: $\mathcal{V}_{\text{sh}}(H) = \{x \leq y,\ y \leq (x + g(x)),\ p(h(x) - h(y)),\ p(0),\ g(x) \simeq 0,\ 0,\ h(x) - h(y),\ g(x),\ p(0),\ p(h(x) - h(y)),\ x\}$

Separation in Equality Sharing vs. CDSAT: comparison

- Shared variables for equality sharing: $\{x, y, w_1, w_2, w_3, w_4, w_5\}$
  w_1 stands for $g(x)$, w_2 stands for $h(x) - h(y)$, w_3 stands for $h(x)$
  w_4 stands for $h(y)$, w_5 stands for 0
- Shared terms for CDSAT: $\mathcal{V}_{\text{sh}}(H) = \{x \leq y,\ y \leq (x + g(x)),\ p(h(x) - h(y)),\ p(0),\ g(x) \simeq 0,\ 0,\ h(x) - h(y),\ g(x),\ p(0),\ p(h(x) - h(y)),\ x\}$
- $h(x)$ and $h(y)$ not in $\mathcal{V}_{\text{sh}}(S)$: they are below EUF-foreign term $h(x) - h(y)$ and LIA-foreign term $p(h(x) - h(y))$
- Assigned terms are shared in CDSAT as they sit on a shared trail, not in equality sharing where each black-box procedure has its own literals

CDSAT modules: equality inference rules

All CDSAT theory modules include equality inference rules:

- Reflexivity: $\vdash t \simeq t$
- Symmetry: $t \simeq s \vdash s \simeq t$
- Transitivity: $t \simeq s,\ s \simeq u \vdash t \simeq u$
- Same value: $t \leftarrow \mathfrak{c},\ s \leftarrow \mathfrak{c} \vdash t \simeq s$
- Different values: $t \leftarrow \mathfrak{c},\ s \leftarrow \mathfrak{q} \vdash t \not\simeq s$

- All modules know that equality is an equivalence
- Only the module for the theory of equality (EUF) knows that it is a congruence

CDSAT module for equality with uninterpreted functions

- $\Sigma_{\mathsf{EUF}} = \langle S, F \rangle$
- $\mathsf{prop} \in S$
- $\simeq_s \in F$ for all sorts $s \in S$
- EUF^+ may be
  - Trivial or
  - Countable: values used as labels of congruence classes, e.g.:
    $t_1 \leftarrow \mathfrak{c},\ t_2 \leftarrow \mathfrak{c},\ t_3 \leftarrow \mathfrak{c}_3,\ t_4 \leftarrow \mathfrak{c}_4,\ t_5 \leftarrow \mathfrak{c}_5$
    shorter than
    $t_1 \simeq t_2,\ t_1 \not\simeq t_3,\ t_1 \not\simeq t_4,\ t_1 \not\simeq t_5,\ t_3 \not\simeq t_4,\ t_3 \not\simeq t_5,\ t_4 \not\simeq t_5$

CDSAT module for equality with uninterpreted functions

- Inference rules for congruence:
  - $(t_i \simeq u_i)_{i=1\ldots m}, (f(t_1, \ldots, t_m) \not\simeq f(u_1, \ldots, u_m)) \vdash_{\mathsf{EUF}} \bot$
  - $(t_i \simeq u_i)_{i=1\ldots m} \vdash_{\mathsf{EUF}} f(t_1, \ldots, t_m) \simeq f(u_1, \ldots, u_m)$
  - $(t_i \simeq u_i)_{i=1\ldots m, i\neq j}, f(t_1, \ldots, t_m) \not\simeq f(u_1, \ldots, u_m) \vdash_{\mathsf{EUF}} t_j \not\simeq u_j$
- $\mathsf{basis}_{\mathsf{EUF}}(X)$: all subterms of terms in X and all equalities between them

CDSAT can emulate equality sharing

- The Deduce transition rule of CDSAT covers $\mathcal{T}_k$-propagation
- Each $\mathcal{T}_k$ module can place an inference $J \vdash_k L$ as justified assignment ${}_J\vdash L$ on the shared trail, including:
  - Equality inferences deducing $u \simeq v$
  - EUF-module inferences deducing $u \simeq v$ by congruence
  - Other $\mathcal{T}_k$-specific inferences deducing $u \simeq v$
  - $\mathcal{T}_k$-specific inferences deducing $\bigvee_{j=1}^{m} u_j \simeq v_j$ in non-convex $\mathcal{T}_k$

CDSAT can emulate equality sharing

- The $\mathcal{T}_k$ modules cooperate to build an arrangement publicly on the shared trail
- The Bool-module can handle disjunctions, if any, by decision and unit propagation
- However, in CDSAT the arrangement
  1. Is about the set $\mathcal{V}_{\text{sh}}(H)$ of shared terms, not the set of shared variables of equality sharing
  2. Does not have to be complete: a partial arrangement of $\mathcal{V}_{\text{sh}}(H)$ suffices for completeness, if Γ is model-describing

The equality sharing example done by CDSAT

$$H = \{x \leq y,\ y \leq (x + g(x)),\ p(h(x) - h(y)),\ \overline{p(0)},\ g(x) \simeq 0\}$$

1. LIA-module: $\{y \leq x + g(x),\ g(x) \simeq 0\} \vdash_{\text{LIA}} y \leq x$
   Deduce: ${}_{J}\vdash(y \leq x)$ (level 0)
   with $J = \{y \leq x + g(x),\ g(x) \simeq 0\}$
2. LIA-module: $\{x \leq y,\ y \leq x\} \vdash_{\text{LIA}} x \simeq y$
   Deduce: ${}_{H}\vdash(x \simeq y)$ (level 0)
   with $H = \{x \leq y,\ y \leq x\}$
3. EUF-module: $x \simeq y \vdash_{\text{EUF}} h(x) \simeq h(y)$
   Deduce: ${}_{I}\vdash(h(x) \simeq h(y))$ (level 0)
   with $I = \{x \simeq y\}$

The equality sharing example done by CDSAT

4. LIA-module: $h(x) \simeq h(y) \vdash_{\mathsf{LIA}} h(x) - h(y) \simeq 0$
   Deduce: ${}_{K}\vdash (h(x) - h(y) \simeq 0)$ (level 0)
   with $K = \{h(x) \simeq h(y)\}$
5. EUF-module: $\{h(x) - h(y) \simeq 0\} \vdash_{\mathsf{EUF}} p(h(x) - h(y)) \simeq p(0)$
   Deduce: ${}_{N}\vdash (p(h(x) - h(y)) \simeq p(0))$ (level 0)
   with $N = \{h(x) - h(y) \simeq 0\}$
6. EUF-module: $\{(p(h(x) - h(y))) \leftarrow \text{true},\ p(0) \leftarrow \text{false}\} \vdash_{\mathsf{EUF}}$
   $p(h(x) - h(y)) \not\simeq p(0)$ (equality inference: also LIA-module)
   but the trail contains $p(h(x) - h(y)) \simeq p(0)$: conflict
   $\{p(h(x) - h(y)),\ \overline{p(0)},\ p(h(x) - h(y)) \simeq p(0)\}$ (level 0)
   Fail returns unsat

CDSAT can handle the problem in a different way

- Each $\mathcal{T}_k$ module can place decisions on the shared trail by Decide transitions
- A $\mathcal{T}_k$-inference $J \vdash_k L$ from $J \subseteq \Gamma$ leads to $\mathcal{T}_k$-conflict $E = J \cup \{\overline{L}\}$ if $\overline{L} \in \Gamma$
- Solved by Backjump or LearnBackjump
- Deductions may detect and explain conflicts rather than being all in saturation mode as in equality sharing
- Interplay of decisions and deductions by different theory modules: late propagation

Late propagation

- Multiple theories and multiple theory modules sharing the trail
  A justified assignment ${}_{H}\vdash A$ may be added after assignments of higher level (decisions and possibly deductions by another module): late propagation
- It follows that the CDSAT trail is not a stack
- Equality sharing does not use a shared trail, because each procedure is a black-box
- The trail is a stack in
  - CDCL
  - CDCL($\mathcal{T}$) where $\mathcal{T}$ is one theory
  - CDCL($\mathcal{T}$) where $\mathcal{T}$ is an equality sharing combination

The equality sharing example done by CDSAT: variant

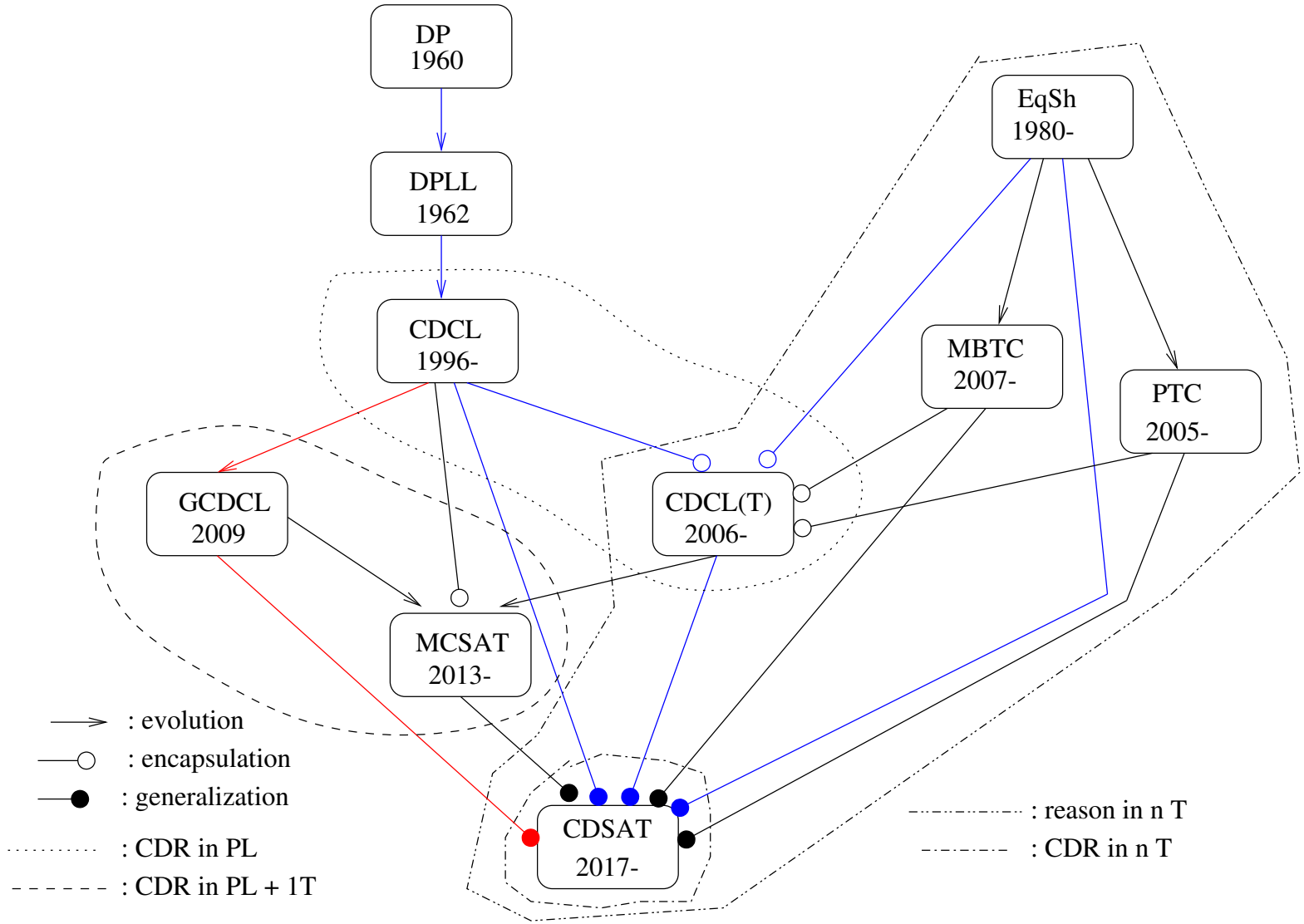
$$H = \{x \leq y,\ y \leq (x + g(x)),\ p(h(x) - h(y)),\ \overline{p(0)},\ g(x) \simeq 0\}$$

1. EUF-module: Decide adds $_?(x \not\simeq y)$ (level 1)
2. LIA-module: $\{y \leq x + g(x),\ g(x) \simeq 0\} \vdash_{\text{LIA}} y \leq x$
   Deduce: $_J\vdash(y \leq x)$ (level 0)
   with $J = \{y \leq x + g(x),\ g(x) \simeq 0\}$ /* late propagation */
3. LIA-module: $\{x \leq y,\ y \leq x\} \vdash_{\text{LIA}} x \simeq y$
   but the trail contains $_?(x \not\simeq y)$
   LIA-conflict: $E_0 = \{x \not\simeq y,\ x \leq y,\ y \leq x\}$
   The deduction of $x \simeq y$ detects a conflict

The equality sharing example done by CDSAT: variant

4. LIA-conflict: $E_0 = \{x \not\simeq y,\ x \leq y,\ y \leq x\}$
   ${}_?(x \not\simeq y)$ has level 1, the rest has level 0
5. Backjump or LearnBackjump: back to level 0 adding
   ${}_H\vdash(x \simeq y)$
   with $H = \{x \leq y,\ y \leq x\}$
   ${}_H\vdash(x \simeq y)$ added by a conflict solving transition
   rather than a Deduce transition
   the derivation may continue as before

The big picture: conflict-driven one-theory reasoning



Conflict-driven satisfiability procedures in arithmetic

- Input problem: set of literals in the theory signature
- Candidate model: theory model
- Assignments to first-order variables (e.g., $x \leftarrow 3$)
- Propagation: evaluation of arithmetic expressions (e.g., $y \leftarrow 0 \vdash_{\mathsf{LRA}} (y > 2) \leftarrow \mathsf{false}$)
- Explanation of theory-conflicts by (expensive) theory inferences: used only on demand to respond to conflicts
- Learn lemmas that may contain new (non-input) atoms and may exclude first-order assignments, hence theory models
- Devise restrictions to ensure termination

Conflict-driven satisfiability procedures in arithmetic

Examples:

- Linear rational (or real) arithmetic (LRA):
  Conflict Resolution [Korovin, Tsiskaridze, Voronkov: CP 2009]
  GDPLL then GCDCL [McMillan, Kuehlmann, Sagiv: CAV 2009]
  Natural Domain SMT [Cotton: FORMATS 2010]
- Linear integer arithmetic (LIA):
  Cutting-to-the-chase procedure [Jovanovic, de Moura: CADE 2011, JAR 2013] [Bromberger, Sturm, Weidenbach: CADE 2015]
- Nonlinear real arithmetic (NRA):
  NLSAT [Jovanovic, de Moura: IJCAR 2012]
  k-SMT [Braußе, Korovin, Korovina, Müller: FroCoS 2019]

The language of LRA-literals

- Input: set S of LRA-literals
- LRA-term:
  - rational constant c
  - sum $c_1 \cdot x_1 + \ldots + c_n \cdot x_n$
- LRA-literal: $t_1 \lessdot t_2$ where $\lessdot \in \{<, \leq\}$
- $\overline{(t_1 < t_2)}$ and $\overline{(t_1 \leq t_2)}$ replaced by $t_2 \leq t_1$ and $t_2 < t_1$
- $t_1 \simeq t_2$ rewritten as $t_1 \leq t_2$ and $t_2 \leq t_1$

The Fourier-Motzkin algorithm for LRA

- Decide the satisfiability of a set of LRA-literals
- Apply variable elimination until
  - Either a contradiction $0 \leq q$ with negative q is generated
    Return unsat
  - Or all variables are eliminated
    Returns sat
- Termination guaranteed: each round eliminates one variable
  Finitely many variables

[Lassez, Maher: JAR 1992]

Variable elimination by linear combination in LRA

1. Select a variable x
2. If x appears only with positive, or negative, coefficients, remove all literals where x appears
3. Otherwise, compute all linear combinations of literals
   $t_1 + c_1 \cdot x \leq u_1$ and $t_2 - c_2 \cdot x \leq u_2$
   generating the literal
   $c_2 \cdot t_1 + c_1 \cdot t_2 \leq c_2 \cdot u_1 + c_1 \cdot u_2$
   (if a premise is strict, the result is also)

[Lassez, Maher: JAR 1992]

Variable elimination by transitive closure in LRA

3. Otherwise (recall $\lessdot \in \{<, \leq\}$):
   - 3.1 Rearrange every literal where x has positive coefficient into upper bound $x \lessdot t$
   - 3.2 Rearrange every literal where x has negative coefficient into lower bound $t \lessdot x$
   - 3.3 Compute all transitive closures concatenating $u \leq x$ and $x \leq t$ to generate $u \leq t$
     (if a premise is strict, the result is also)

The Fourier-Motzkin algorithm generates a doubly exponential number of literals in the worst case

[Schrijver: 1998], [Kroening, Strichman: 2008]

The level-saturation strategy in propositional logic

- Decide the satisfiability of a set of propositional clauses
- Apply variable elimination until
  - Either a contradiction $\{L, \overline{L}\}$ is generated
    Return unsat
  - Or all variables are eliminated
    Returns sat
- Termination guaranteed: each round eliminates one variable
  Finitely many variables

[Chang, Lee: 1973]

Variable elimination by resolution in propositional logic

1. Select a propositional variable l
2. If l appears only with positive, or negative, sign, remove all clauses where l appears: pure literal rule
3. Otherwise:
   - Add all resolvents generated by resolving upon l
   - Remove all clauses where l appears

The level-saturation strategy generates too many resolvents

An inference rule for LRA

- Extract from the Fourier-Motzkin algorithm an inference rule for LRA and use it in a conflict-driven manner like CDCL does with resolution
- Fourier-Motzkin (FM) resolution:
  $\{t_1 \lessdot_1 x,\ x \lessdot_2 t_2\} \vdash_{\text{LRA}} t_1 \lessdot_3 t_2$
  $\lessdot_1, \lessdot_2, \lessdot_3 \in \{<, \leq\}$
  $\lessdot_3$ is $<$ if either $\lessdot_1$ or $\lessdot_2$ is $<$ and $\leq$ otherwise
  - Linear combination of constraints:
    $\{x + y < 0,\ -y + 2 < 0\} \vdash_{\text{LRA}} x + 2 < 0$
  - Transitive closure: $\{x < -y,\ -y < -2\} \vdash_{\text{LRA}} x < -2$

Conflict Resolution or GCDCL for LRA

It stands to the Fourier-Motzkin algorithm like CDCL stands to the level-saturation strategy for propositional logic

- Candidate LRA-model written as assignments to rational variables
- Propagation: evaluation of arithmetic expressions
- Explanation of LRA-conflicts by FM-resolution: used only on demand to respond to conflicts
- Learn lemmas that may contain new (non-input) atoms and may exclude LRA-models

Conflict-driven one-theory reasoning in CDSAT

In order to see how CDSAT covers conflict-driven procedures such as Conflict Resolution or GCDCL for LRA:

1. First-order decisions $?(t \leftarrow \mathfrak{c})$: more about acceptability
2. Transition rule Deduce beyond propositional inferences (e.g., unit propagation) and deduction of equalities as in equality sharing
3. The CDSAT module for LRA
4. A CDSAT transition rule named UndoClear to solve conflicts due to first-order decisions

The CDSAT trail (beyond the Boolean case)

A trail Γ is a sequence of assignments $\Gamma = A_0, \ldots, A_m$ where

- Each A_i is either a decision ${}_?A$ or a justified assignment ${}_{H\vdash}A$
- Decision: either Boolean ${}_?L$ or first-order ${}_?(t \leftarrow \mathfrak{c})$ for $\mathfrak{c}$ other than true or false
- Justified assignment ${}_{H\vdash}A$:
  - Supported by an inference $H \vdash_k A$ in a theory module $\mathcal{I}_k$
  - Input assignment ($H = \emptyset$)
  - Due to conflict-solving transitions (Backjump, LearnBackjump)
  - All Boolean except for input first-order assignments
- Levels and the notion of conflict are unchanged except that also first-order assignments may appear in conflicts

More about acceptability of a decision: one theory case

Transition rule Decide adds to the trail a decision:

$$\text{Decide}: \Gamma \longrightarrow \Gamma, ?(u \leftarrow \mathfrak{c})$$

provided $\mathcal{T}$-assignment $u \leftarrow \mathfrak{c}$ is acceptable for $\mathcal{T}$-module $\mathcal{I}$:

- Γ does not contain a $\mathcal{T}$-assignment to term u
  In the Boolean case: $(u \leftarrow \text{true}) \notin \Gamma$ and $(u \leftarrow \text{false}) \notin \Gamma$
- If $(u \leftarrow \mathfrak{c})$ is a first-order $\mathcal{T}$-assignment: for no $J \subseteq \Gamma$ there exists an $\mathcal{I}$-inference $J \cup \{u \leftarrow \mathfrak{c}\} \vdash_{\mathcal{T}} L$ and $\overline{L} \in \Gamma$
  Exclude a first-order decision that triggers an immediate conflict from which nothing can be learned

More on the CDSAT transition rule Deduce

- Propagation:
  - Boolean propagation: e.g., unit propagation
  - $\mathcal{T}_k$-propagation: e.g., propagation of equalities when emulating equality sharing
- Preliminary conflict explanation with lemma learning:
  $\mathcal{T}_k$-inferences explain a $\mathcal{T}_k$-conflict
  generating lemmas excluding $\mathcal{T}_k$-assignments until the
  $\mathcal{T}_k$-conflict can be detected as a Boolean conflict on the trail:
  $J \vdash_k L$ and $\overline{L} \in \Gamma$
  unsatisfiable assignment $E = J \cup \{\overline{L}\}$

CDSAT module for linear rational arithmetic (LRA)

- Signature Σ_{LRA}:
  - Sorts: $S = \{\mathsf{prop}, \mathsf{Q}\}$
  - Symbols: $\simeq_s$ for all $s \in S$ (s can be omitted)
    $1, +, <, \leq, q\cdot$ for all rational numbers $q \in \mathbb{Q}$
  - (e.g., $(-2{\cdot}x + (-1{\cdot}y)) < 0{\cdot}1$)
- Theory extension LRA^+ adds
  - Constants $\tilde{q}$ for all $q \in \mathbb{Q}$ ($\tilde{q}$ written q)
    /* countable */
  - Axioms $\tilde{q} \simeq q{\cdot}1$ for all $q \in \mathbb{Q}$
  - (e.g., $(-2{\cdot}x + (-1{\cdot}y)) < 0$ hence $-2x - y < 0$)

Summary of the first lecture

- Introduction and motivation
- Conflict-driven reasoning and CDSAT in a nutshell
- Propositional conflict-driven reasoning: CDSAT and CDCL
- Combination of theories: CDSAT and Nelson-Oppen
- Conflict-driven reasoning in one theory: LRA procedures

Second lecture

- Conflict-driven reasoning in one theory: LRA in CDSAT
- Conflict-driven reasoning in one theory from literals to clauses: CDSAT and MCSAT
- Conflict-driven reasoning in many theories: CDSAT
- CDSAT and arrays, maps, and vectors with abstract domain

CDSAT module for linear rational arithmetic (LRA)

Inference rules:

- Evaluation: $\{t_1 \leftarrow \tilde{q}_1, \ldots, t_m \leftarrow \tilde{q}_m\} \vdash_{\mathsf{LRA}} l \leftarrow \mathfrak{b}$
- Positivization: $\overline{t_1 < t_2} \vdash_{\mathsf{LRA}} t_2 \leq t_1$
  $\overline{t_1 \leq t_2} \vdash_{\mathsf{LRA}} t_2 < t_1$
- Equality elimination: $t_1 \simeq_{\mathsf{Q}} t_2 \vdash_{\mathsf{LRA}} \{t_1 \leq t_2,\ t_2 \leq t_1\}$

CDSAT module for linear rational arithmetic (LRA)

Inference rules:

- FM-resolution: $\{t_1 \prec_1 x,\ x \prec_2 t_2\} \vdash_{\mathsf{LRA}} t_1 \prec_3 t_2$
  $\prec_1, \prec_2, \prec_3 \in \{<, \leq\}$
  $\prec_3$ is $<$ if either $\prec_1$ or $\prec_2$ is $<$ and $\leq$ otherwise
- Disequality elimination:
  $\{t_1 \leq x,\ x \leq t_2,\ t_1 \simeq t_0,\ t_2 \simeq t_0,\ x \not\simeq t_0\} \vdash_{\mathsf{LRA}} \bot$
  detects an LRA-conflict: no value for variable x

Conflict state rule UndoClear

- Given conflict $E \uplus \{A\}$, assignment A is outstanding in $E \uplus \{A\}$ if $\text{level}_\Gamma(A) > \text{level}_\Gamma(E)$
- Conflict state: $\langle \Gamma; E \uplus \{A\} \rangle$ where A is an outstanding first-order assignment s.t $\text{level}_\Gamma(A) = m$
- Transition rule UndoClear solves the conflict by undoing A and removing all assignments of level greater than or equal to m:

$$\text{UndoClear}: \langle \Gamma; E \uplus \{A\} \rangle \Longrightarrow \Gamma^{\leq m-1}$$

- A is a decision: $m - 1 \geq 0$ implies $m \geq 1$, and first-order assignments in Γ are either input (level 0) or decisions

Conflict state rule UndoClear

- The removed assignments
  - Include ${}_{H}\vdash L$ such that $A \in H$: if A goes, ${}_{H}\vdash L$ must go
  - Are not necessarily the most recent ones, as there may have been late propagations that are more recent
- $\Gamma^{\leq m-1}$ is new (no loop) even if nothing is added:
  - A was acceptable when decided: no conflict
  - Conflict $E \uplus \{A\}$: some ${}_{H}\vdash L$ such that $\text{level}_{\Gamma}(L) < m$ was deduced afterwards (late propagation)
  - $\Gamma^{\leq m-1}$ contains ${}_{H}\vdash L$
- UndoClear fires after a late propagation

A simple example with UndoClear

$\{I_0 : 2x + y \simeq 1,\ I_1 : 2x + 2y \simeq 1\}$ subset of the input (level 0)

1. Decide: $_?(x \leftarrow 0)$ (level 1) /* acceptable */
2. Deduce: $_J\vdash(y \simeq 0)$ with $J = \{2x + y \simeq 1,\ 2x + 2y \simeq 1\}$ (level 0)
   FM-resolution: $\{2x + y \simeq 1,\ 2x + 2y \simeq 1\} \vdash_{\text{LRA}} y \simeq 0\ (I_1 - I_0)$
   /* late propagation */
3. Evaluation: $\{x \leftarrow 0,\ y \simeq 0\} \vdash_{\text{LRA}} 2x + y \not\simeq 1$ detects
   LRA-conflict $E = \{x \leftarrow 0,\ y \simeq 0,\ 2x + y \simeq 1\}$
   UndoClear: undo $_?(x \leftarrow 0)$ (outstanding in E) back to level 0
4. Decide: $_?(x \leftarrow 1/2)$ (level 1)
   /* only acceptable value for x: forced decision */

Forced decision

- If a first-order assignment $t \leftarrow \mathfrak{c}$ follows from the trail as $\mathfrak{c}$ is the only acceptable value for t for a theory module, $t \leftarrow \mathfrak{c}$ is a forced decision for that module
- This is a reason why theory module inferences only infer Boolean assignments

Forced decision: examples

- $\{u \simeq t,\ t \leftarrow \mathfrak{c}\} \subseteq \Gamma$:
  $u \leftarrow \mathfrak{c}$ is a forced decision for the EUF-module
- $\{u \leq t,\ t \leq u,\ t \leftarrow \mathfrak{c}\} \subseteq \Gamma$
  where $\leq$ is the ordering on the rationals:
  $u \leftarrow \mathfrak{c}$ is a forced decision for the LRA-module

How about termination of conflict-driven FM-resolution?

FM-resolutions alternating on distinct variables may diverge:

$$
\begin{array}{lcll}
I_0: & -2{\cdot}x - y < 0 & & \\
I_1: & x + y < 0 & & \\
I_2: & x < -1 & & \\
I_3: & -y < -2 & (I_0 + 2I_2) & \mathit{elim}\ x \\
I_4: & x < -2 & (I_1 + I_3) & \mathit{elim}\ y \\
I_5: & -y < -4 & (I_0 + 2I_4) & \mathit{elim}\ x \\
I_6: & x < -4 & (I_1 + I_5) & \mathit{elim}\ y \\
I_7: & -y < -8 & (I_0 + 2I_6) & \mathit{elim}\ x \\
\ldots & \ldots & \ldots & \ldots
\end{array}
$$

also when FM-resolution is applied only to respond to conflicts

The non-terminating LRA example in CDSAT

$l_0 : -2{\cdot}x - y < 0,\ l_1 : x + y < 0,\ l_2 : x < -1$ (level 0)

1. Decide: $_?(y \leftarrow 0)$ (level 1) /* acceptable */
   LRA-conflict: $\{-2{\cdot}x - y < 0,\ x < -1,\ y \leftarrow 0\}$
2. Explained by $l_0 + 2l_2 : \{-y < 2{\cdot}x,\ 2{\cdot}x < -2\} \vdash_{\text{LRA}} -y < -2$
   Deduce: $l_3 : -y < -2$ (level 0) /* late propagation */
3. $y \leftarrow 0 \vdash_{\text{LRA}} \overline{-y < -2}$ detects LRA-conflict $\{y \leftarrow 0,\ -y < -2\}$
   UndoClear: undo $_?(y \leftarrow 0)$ and back to level 0
4. Decide: $_?(x \leftarrow -2)$ (level 1) /* acceptable */
   LRA-conflict: $\{x + y < 0,\ -y < -2,\ x \leftarrow -2\}$
5. Explained by $l_1 + l_3 : \{x < -y,\ -y < -2\} \vdash_{\text{LRA}} x < -2$
   Deduce: $l_4 : x < -2$ (level 0) /* late propagation */

The non-terminating LRA example in CDSAT

6. $x \leftarrow -2 \vdash_{\text{LRA}} \overline{x < -2}$ detects LRA-conflict $\{x \leftarrow -2,\ x < -2\}$
   UndoClear: undo $_?(x \leftarrow -2)$ and back to level 0
7. Decide: $_?(y \leftarrow 3)$ (level 1) /* acceptable */
   LRA-conflict: $\{-2 \cdot x - y < 0,\ x < -2,\ y \leftarrow 3\}$
8. Explained by $l_0 + 2l_4 : \{-y < 2 \cdot x,\ 2 \cdot x < -4\} \vdash_{\text{LRA}} -y < -4$
   Deduce: $l_5 : -y < -4$ (level 0) /* late propagation */
9. $y \leftarrow 3 \vdash_{\text{LRA}} \overline{-y < -4}$ detects LRA-conflict $\{y \leftarrow 3,\ -y < -4\}$
   UndoClear: undo $_?(y \leftarrow 3)$ and back to level 0
10. Decide: $_?(x \leftarrow -3)$ (level 1) /* acceptable */
    LRA-conflict: $\{x + y < 0,\ -y < -4,\ x \leftarrow -3\}$

The non-terminating LRA example in CDSAT

11. Explained by $l_1 + l_5 : \{x < -y,\ -y < -4\} \vdash_{\text{LRA}} x < -4$
    Deduce: $l_6 : x < -4$ (level 0) /* late propagation */
12. $x \leftarrow -3 \vdash_{\text{LRA}} \overline{x < -4}$ detects LRA-conflict $\{x \leftarrow -3,\ x < -4\}$
    UndoClear: undo $_?(x \leftarrow -3)$ and back to level 0
13. Decide: $_?(y \leftarrow 5)$ (level 1) /* acceptable */
    LRA-conflict: $\{-2 \cdot x - y < 0,\ x < -4,\ y \leftarrow 5\}$
14. Explained by $l_0 + 2l_6 : \{-y < 2 \cdot x,\ 2 \cdot x < -8\} \vdash_{\text{LRA}} -y < -8$
    Deduce: $l_7 : -y < -8$ (level 0) /* late propagation */
15. $y \leftarrow 5 \vdash_{\text{LRA}} \overline{-y < -8}$ detects LRA-conflict $\{y \leftarrow 5,\ -y < -8\}$
    UndoClear: undo $_?(y \leftarrow 5)$ and back to level 0

    ... ...

Solution of the non-termination problem

- Assume a fixed total ordering $\prec_{\mathsf{LRA}}$ on rational variables
- Apply FM-resolution only if the variable resolved upon is $\prec_{\mathsf{LRA}}$-maximum in both premises
- Solution adopted in:
  - Conflict Resolution [Korovin, Tsiskaridze, Voronkov: CP 2009]
  - GDPLL aka GCDCL [McMillan, Kuehlmann, Sagiv: CAV 2009]
  - The LRA-module in CDSAT

Completing the CDSAT module for LRA

- FM-resolution: $\{t_1 \prec_1 x,\ x \prec_2 t_2\} \vdash_{\mathsf{LRA}} t_1 \prec_3 t_2$
  $\prec_1, \prec_2, \prec_3 \in \{<, \leq\}$
  $\prec_3$ is $<$ if either $\prec_1$ or $\prec_2$ is $<$ and $\leq$ otherwise
- $\mathsf{basis}_{\mathsf{LRA}}(X)$: subterms, equalities, disequalities generated by the inference rules
  for FM-resolution only the disequalities that FM-resolution generates by resolving on the $\prec_{\mathsf{LRA}}$-maximum variable in the premises are included

Completing the CDSAT module for LRA (I version)

- Detection of an empty solution space:
  $\{y_1 \leftarrow \tilde{q}_1, \ldots, y_m \leftarrow \tilde{q}_m\} \uplus E \vdash_{\mathsf{LRA}} \bot$
  if for all x in E there is an i $(1 \leq i \leq m)$ such that
  $x \prec_{\mathsf{LRA}} y_i$ or $x = y_i$ (1st horn for at least one x)
  and $\{y_1 \leftarrow \tilde{q}_1, \ldots, y_m \leftarrow \tilde{q}_m\} \uplus E$ is LRA-unsatisfiable
- No value for some x given the assignments to $y_1, \ldots, y_m$

Completing the CDSAT module for LRA (II version)

- Omit detection of an empty solution space
- An LRA-assignment J is sensible if $x \prec_{\mathsf{LRA}} y$ and J assigns a value to y imply that J assigns a value to x
- Adopt a sensible search plan: selects rational variables for decision in $\prec_{\mathsf{LRA}}$-increasing order
- LRA-assignments generated by a sensible search plan are sensible

The non-terminating example fixed in CDSAT

- Assume $y \prec_{\mathsf{LRA}} x$
- 2nd FM-resolution inference in the non-halting sequence:
  $\{x < -y,\ -y < -2\} \vdash_{\mathsf{LRA}} x < -2$
  is barred: it resolves on y when x occurs in the premises
- All derivations embedding that diverging series of FM-resolution inferences are barred
- Multiple CDSAT-derivations discover that
  $l_0 :\ -2{\cdot}x - y < 0,\ l_1 : x + y < 0,\ l_2 : x < -1$
  is LRA-unsatisfiable
- A simple one does it by mere LRA-propagations at level 0

The non-terminating example fixed in CDSAT

$l_0 : \ -2{\cdot}x - y < 0, \ l_1 : x + y < 0, \ l_2 : x < -1$ (level 0)

Assume $y \prec_{\mathsf{LRA}} x$

1. Deduce: $l_3 : -y < -2$ (level 0)
   $l_0 + 2l_2 : \quad \{-y < 2{\cdot}x, \ 2{\cdot}x < -2\} \vdash_{\mathsf{LRA}} -y < -2$
   /* x is $\prec_{\mathsf{LRA}}$-max variable in both premises */
2. Deduce: $l_4 : y < 0$ (level 0) /*normal form of $-y < -2{\cdot}y$ */
   $l_0 + 2l_1 : \quad \{-y < 2{\cdot}x, \ 2{\cdot}x < -2{\cdot}y\} \vdash_{\mathsf{LRA}} -y < -2{\cdot}y$
   /* x is $\prec_{\mathsf{LRA}}$-max variable in both premises */

The non-terminating example fixed in CDSAT

$I_0 : -2 \cdot x - y < 0,\ I_1 : x + y < 0,\ I_2 : x < -1$ (level 0)

Assume $y \prec_{\mathsf{LRA}} x$

3. Deduce: $I_5 : 2 < 0$ (level 0)
   $-I_3 + I_4 : \{2 < y,\ y < 0\} \vdash_{\mathsf{LRA}} 2 < 0$
   /* y is $\prec_{\mathsf{LRA}}$-max variable in both premises as there is no x*/
4. $\emptyset \vdash_{\mathsf{LRA}} \overline{2 < 0}$ reveals conflict at level 0:
   Fail returns unsat

Outline of GCDCL procedure for generic single theory $\mathcal{T}$

- How about going from $\mathcal{T}$-literals to $\mathcal{T}$-clauses?
- Embed reasoning about disjunction into theory reasoning by generalizing to $\mathcal{T}$-clauses a theory reasoning inference rule for $\mathcal{T}$-literals
- Apply the generalized rule only to explain conflicts
- Devise restrictions to ensure termination

Achieved in GCDCL: only linear rational arithmetic (LRA)

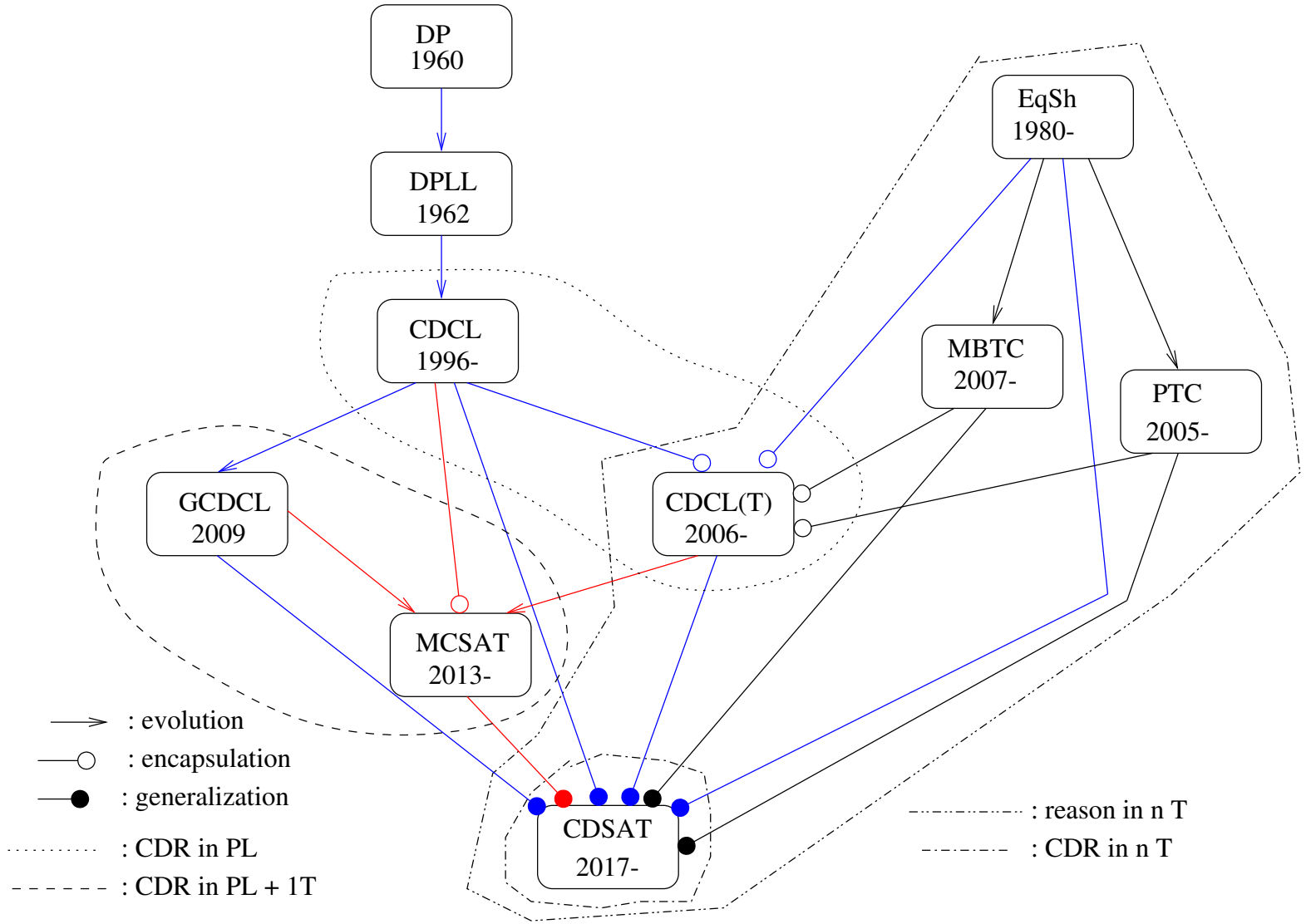
[McMillan, Kuehlmann, Sagiv: CAV 2009]

Generalized CDCL (GCDCL) for LRA

- Input: set S of LRA-clauses
- LRA-clause: disjunction of LRA-literals
- Generalize FM-resolution to LRA-clauses: shadow rule e.g.:
  $\{(b < d) \vee (c < d),\ d < a\} \vdash_{\mathsf{LRA}} (b < a) \vee (c < a)$
- Generates new (non-input) atoms
- Applied only to explain LRA-conflicts
  generating lemmas excluding LRA-assignments
- Add restrictions to ensure termination

[McMillan, Kuehlmann, Sagiv: CAV 2009]

The big picture: better conflict-driven one-theory reasoning



MCSAT (Model-Constructing SATisfiability)

- Entrust the reasoning about disjunction to CDCL
- Integrate in CDCL a conflict-driven $\mathcal{T}$-satisfiability procedure for sets of $\mathcal{T}$-literals
- CDCL($\mathcal{T}$)?
  No, it allows neither first-order assignment
  nor new atoms on the trail
  nor $\mathcal{T}$-inferences generating lemmas excluding $\mathcal{T}$-assignments

[de Moura, Jovanović: VMCAI 2013], [Jovanović, Barrett, de Moura: FMCAD 2013], [Jovanović: VMCAI 2017], [Bobot, Graham-Lengrand, Marre, Bury: SMT 2018], [Graham-Lengrand, Jovanović, Dutertre: IJCAR 2020]

MCSAT (Model-Constructing SATisfiability)

- Integrate CDCL and one model-constructing conflict-driven $\mathcal{T}$-sat procedure for sets of $\mathcal{T}$-literals ($\mathcal{T}$-plugin) that
  - Has access to the trail
  - Proposes assignments to first-order terms: $\mathcal{T}$-assignment
  - Computes $\mathcal{T}$-propagations
  - Explains $\mathcal{T}$-conflicts by $\mathcal{T}$-inferences generating lemmas excluding $\mathcal{T}$-assignments
  - Lemma may contain new (i.e., non-input) atoms coming from a finite basis for termination
- CDCL and the $\mathcal{T}$-plugin cooperate in model construction
- Both propositional and $\mathcal{T}$-reasoning are conflict-driven

CDSAT generalizes MCSAT

- CDSAT generalizes MCSAT to a generic union $\mathcal{T} = \bigcup_{k=1}^{n} \mathcal{T}_k$
- MCSAT is not a combination calculus
  hence does not cover, e.g.:
  - Interaction of multiple first-order theories on the trail
  - Conflict-drivenness for more than one first-order theory
  - Combination of conflict-driven and black-box procedures
  - Soundness, completeness, termination for theory combination
  - Construction of finite global basis from local ones
- CDSAT does not require model-constructing $\mathcal{T}_k$-sat procedures in MCSAT arith-inspired strong sense

CDSAT generalizes MCSAT

- The CDSAT and MCSAT transition systems are different
- What MCSAT does by model-evaluation mechanism and theory-explanation function is done in CDSAT by $\mathcal{T}_k$-inferences
  - MCSAT: explanation function private to $\mathcal{T}$-plugin
  - CDSAT: $\mathcal{T}_k$-inferences public on shared trail
- MCSAT: integration of CDCL and a $\mathcal{T}$-plugin
  CDSAT: reasoning in the union of Bool and $\mathcal{T}$

CDSAT emulates MCSAT if the theory union contains only Bool and one theory $\mathcal{T}$ equipped with a conflict-driven model-constructing $\mathcal{T}$-sat procedure for sets of $\mathcal{T}$-literals

Example where CDSAT emulates MCSAT

- $H = \{x < y,\ x < z,\ (y < w) \vee (z < w),\ w < x\}$
- Theory union: LRA and Bool sharing sort prop
- Bool: $<$ is a foreign symbol, the $<$-terms are foreign terms seen as propositional variables
- LRA: $\vee$ is a foreign symbol, the disjunction is a foreign term seen as a propositional variable
- $\mathcal{V}_{\text{sh}}(H) = \{x < y,\ x < z,\ (y < w) \vee (z < w),\ w < x,\ y < w,\ z < w\}$

Example where CDSAT emulates MCSAT

$x < y,\ x < z,\ (y < w) \vee (z < w),\ w < x \quad$ (level 0)

LRA-module:

- Ordering on rational variables: $x \prec_{\text{LRA}} y \prec_{\text{LRA}} z \prec_{\text{LRA}} w$
- Sensible search plan

1. Decide: $_?(x \leftarrow 0) \quad$ (level 1) /* acceptable for LRA */
2. Decide: $_?(y \leftarrow 1) \quad$ (level 2) /* acceptable for LRA */
   /* $_?(y \leftarrow 0)$ not acceptable for LRA: $\{x \leftarrow 0,\ y \leftarrow 0\} \vdash_{\text{LRA}} \overline{(x < y)}$*/
3. Decide: $_?(z \leftarrow 1) \quad$ (level 3) /* acceptable for LRA*/
   /* $_?(z \leftarrow 0)$ not acceptable for LRA: $\{x \leftarrow 0,\ z \leftarrow 0\} \vdash_{\text{LRA}} \overline{(x < z)}$*/

Example where CDSAT emulates MCSAT

4. Decide: $_?(y < w)$ (level 4) /* acceptable for Bool*/
5. Deduce: $_J\vdash(y < x)$ with $J = \{y < w,\ w < x\}$ (level 4)
   $\{y < w,\ w < x\} \vdash_{\text{LRA}} y < x$
   /* w is $\prec_{\text{LRA}}$-max variable in both $y < w$ and $w < x$ */
6. Deduce: $_I\vdash(x < x)$ with $I = \{x < y,\ y < x\}$ (level 4)
   $\{x < y,\ y < x\} \vdash_{\text{LRA}} x < x$
   /* y is $\prec_{\text{LRA}}$-max variable in both $x < y$ and $y < x$ */
   LRA-conflict: $E_0 = \{x < x\}$ (level 4)
7. Resolve: $E_1 = \{x < y,\ y < x\}$
8. Resolve: $E_2 = \{x < y,\ y < w,\ w < x\}$

Example where CDSAT emulates MCSAT

$E_2 = \{x < y,\ y < w,\ w < x\}$: outstanding in E_2
$\text{level}_\Gamma(y < w) = 4$, $\text{level}_\Gamma(x < y) = \text{level}_\Gamma(w < x) = 0$

9. Backjump: back to level 0 adding ${}_K\vdash(\overline{y < w})$
   $K = \{x < y,\ w < x\}$
10. Deduce: ${}_G\vdash(z < w)$ (level 0)
    $G = \{\overline{y < w},\ (y < w) \vee (z < w)\}$ (level 0)
    $\{\overline{y < w},\ (y < w) \vee (z < w)\} \vdash_{\text{Bool}} z < w$
    /* shadow rule unnecessary: Bool-module handles disjunction
    LRA-module reasons about LRA-literals */

Example where CDSAT emulates MCSAT

11. Deduce: ${}_{M}\vdash(z < x)$ with $M = \{z < w,\ w < x\}$ (level 0)
    $\{z < w,\ w < x\} \vdash_{\mathsf{LRA}} z < x$
    /* w is $\prec_{\mathsf{LRA}}$-max variable in both $z < w$ and $w < x$ */
12. Deduce: ${}_{N}\vdash(x < x)$ with $N = \{x < z,\ z < x\}$ (level 0)
    $\{x < z,\ z < x\} \vdash_{\mathsf{LRA}} x < x$
    /* z is $\prec_{\mathsf{LRA}}$-max variable in both $x < z$ and $z < x$ */
13. LRA-conflict: $E_3 = \{x < x\}$ (level 0)
    Fail returns unsat

CDSAT: Conflict-driven reasoning from a theory to many

- Conflict-driven behavior and black-box integration are at odds:
  each conflict-driven $\mathcal{T}_k$-sat procedure needs to access the trail, post assignments, perform inferences, explain $\mathcal{T}_k$-conflicts, post lemmas as assignments
- Key abstraction in CDSAT: open the black-boxes
  pull out the $\mathcal{T}_k$-inference systems and coordinate them in a conflict-driven way
- If $\mathcal{T}_k$ has no conflict-driven $\mathcal{T}_k$-sat procedure:
  black-box inference rule $L_1, \ldots, L_m \vdash_k \bot$
  invokes the $\mathcal{T}_k$-procedure to detect $\mathcal{T}_k$-unsat

Reasoning in a theory union

$\mathcal{T} = \bigcup_{i=1}^{n} \mathcal{T}_k$

- Each component theory $\mathcal{T}_k$ has
  - Signature $\Sigma_k = (S_k, F_k)$ where $S_k$is the set of sorts and F_k is the set of symbols
  - Conservative extension $\mathcal{T}_k^+$ providing $\mathcal{T}_k$-values
- The union theory $\mathcal{T}$ has all sorts, all symbols, all values:
  - Signature $\Sigma = (S, F)$ where $S = \bigcup_{i=1}^{n} S_k$ and $F = \bigcup_{i=1}^{n} F_k$
  - Conservative extension $\mathcal{T}^+ = \bigcup_{i=1}^{n} \mathcal{T}_k^+$

Key notions for conflict-driven reasoning in a theory union

$\mathcal{T} = \bigcup_{i=1}^{n} \mathcal{T}_k$

- Theory view: the view that a component theory $\mathcal{T}_k$ has of a $\mathcal{T}$-assignment
- Assignments and models: endorsement
- Relevance of a term to a theory
- Acceptability of an assignment for a theory module in the presence of more than one theory

Theory view of an assignment

$\mathcal{T} = \bigcup_{k=1}^{n} \mathcal{T}_k$

- The $\mathcal{T}_k$-view of a $\mathcal{T}$-assignment H, denoted H_k, is the $\mathcal{T}_k$-assignment that contains:
  - $\mathcal{T}_k$-assignments in H: those that assign $\mathcal{T}_k$-values
  - $u \simeq t$ if H contains $u \leftarrow \mathfrak{c}$ and $t \leftarrow \mathfrak{c}$ of a $\mathcal{T}_k$ sort s ($s \neq$ prop)
  - $u \not\simeq t$ if H contains $u \leftarrow \mathfrak{c}$ and $t \leftarrow \mathfrak{q}$ of a $\mathcal{T}_k$ sort s ($s \neq$ prop) with $\mathfrak{c} \neq \mathfrak{q}$
- The assignments $u \leftarrow \mathfrak{c}$ and $t \leftarrow \mathfrak{c}$ (or $t \leftarrow \mathfrak{q}$) may have been posted by another theory $\mathcal{T}_j$ ($k \neq j$) that also has sort s
- A Boolean assignment belongs to every theory view

Theory view in the CDSAT as MCSAT example

- Input: $x < y,\ x < z,\ (y < w) \vee (z < w),\ w < x$
- Theories: LRA and Bool
- All input assignments are Boolean: in both theory views
- First-order decisions by LRA such as $_?(x \leftarrow 0)$ and $_?(y \leftarrow 1)$ and their consequences such as $x \neq y$:
  only in the LRA-view
- Boolean decisions: in both theory views

The view of the union theory: global view

$\mathcal{T} = \bigcup_{k=1}^{n} \mathcal{T}_k$

- The $\mathcal{T}$-view of a $\mathcal{T}$-assignment H, denoted $H_{\mathcal{T}}$, has everything:
  - Since $\mathcal{T}$ has all the values, all assignments are $\mathcal{T}$-assignments and hence are in $H_{\mathcal{T}}$
  - Since $\mathcal{T}$ has all the sorts, all equalities and disequalities gleaned from first-order assignments are in $H_{\mathcal{T}}$
- The $\mathcal{T}$-view is called global view

In order to see another example let's introduce another theory

A theory of a data structure: arrays

- Signature $\Sigma_{\text{Arr}} = (S, F)$ where:
  - Sorts: $S = \{\text{prop}, I, V, A\}$
  - I: sort of indices
  - V: sort of (array) values
  - A: sort of arrays with indices of sort I and values of sort V
  - Symbols: $F = \{\simeq_s | \ s \in S\} \cup \{\text{select}, \ \text{store}\}$
  - select: $A \times I \to V$ (read)
  - store: $A \times I \times V \to A$ (write)
- Theory extension Arr^+: trivial or countable
  (same choice as for EUF)

Theory view: 3-theory example

$$H = \{x > 1,\ \mathsf{store}(a, i, v) \simeq b,\ \mathsf{select}(a, j) \leftarrow \mathsf{red},\ y \leftarrow -1,\ z \leftarrow 2\}$$

- Theories: Bool, Arr, and EUF where
  EUF has the sort Q of the rational numbers
  red is an Arr-value of sort V
- $H_{\mathsf{Bool}} = \{x > 1,\ \mathsf{store}(a, i, v) \simeq b\}$
- $H_{\mathsf{Arr}} = \{x > 1,\ \mathsf{store}(a, i, v) \simeq b,\ \mathsf{select}(a, j) \leftarrow \mathsf{red}\}$
- $H_{\mathsf{LRA}} = \{x > 1,\ \mathsf{store}(a, i, v) \simeq b,\ y \leftarrow -1,\ z \leftarrow 2,\ y \not\simeq z\}$
- $H_{\mathsf{EUF}} = \{x > 1,\ \mathsf{store}(a, i, v) \simeq b,\ y \not\simeq z\}$
- Global view: $H \cup \{y \not\simeq z\}$

Assignments and models: endorsement

- Let $\mathcal{M}$ be a $\mathcal{T}^+$-model
- $\mathcal{M}$ endorses $u \leftarrow \mathfrak{c}$, written $\mathcal{M} \models (u \leftarrow \mathfrak{c})$:
  $\mathcal{M}$ interprets term u and $\mathcal{T}$-value $\mathfrak{c}$ as the same element in $\mathcal{M}$'s domain for their sort
- Boolean assignment:
  - Endorsement is the same as formula satisfaction:
    $\mathcal{M} \models (l \leftarrow \text{true})$ iff $\mathcal{M} \models l$ and $\mathcal{M} \models (l \leftarrow \text{false})$ iff $\mathcal{M} \models \bar{l}$
  - No difference btw $\mathcal{T}$-model and $\mathcal{T}^+$-model
- First-order assignment: $\mathcal{T}^+$-model needed to interpret $\mathcal{T}$-values and interpret them consistently with $\mathcal{T}^+$-axioms

Endorsement and theory view in one theory

- Let $\mathcal{M}$ be a $\mathcal{T}_k^+$-model
- $\mathcal{M} \models \{u \leftarrow \mathfrak{c},\ t \leftarrow \mathfrak{c}\}$ implies $\mathcal{M} \models (u \simeq t)$
- Let $J = \{u \leftarrow \mathfrak{c},\ t \leftarrow \mathfrak{q}\}$ where $\mathfrak{c} \neq \mathfrak{q}$
  - $\mathcal{M} \models J$ does not imply $\mathcal{M} \models (u \not\simeq t)$
  - $\mathcal{M} \models J_{\mathcal{T}_k}$ implies $\mathcal{M} \models (u \not\simeq t)$
- The desired property is endorsement of the theory view

Satisfiability of an assignment in one theory

- A $\mathcal{T}_k$-assignment J is satisfiable if there exists a $\mathcal{T}_k^+$-model $\mathcal{M}$ such that $\mathcal{M} \models J_{\mathcal{T}_k}$
- $J \models L$: for all $\mathcal{T}_k^+$-models $\mathcal{M}$
  if $\mathcal{M} \models J_{\mathcal{T}_k}$ then $\mathcal{M} \models L$
- Sound $\mathcal{T}_k$-inference: if $J \vdash_k L$ then $J \models L$

Global endorsement

$\mathcal{T} = \bigcup_{k=1}^{n} \mathcal{T}_k$

- A $\mathcal{T}$-assignment H is satisfiable if there exists a $\mathcal{T}^+$-model $\mathcal{M}$ such that $\mathcal{M} \models H_{\mathcal{T}}$
- A $\mathcal{T}^+$-model may identify values from different theories:
  - $H = \{t \leftarrow 3.1,\ u \leftarrow 5.4,\ t \leftarrow \text{red},\ u \leftarrow \text{blue}\}$
  - $\mathcal{T}_i^+$-model $\mathcal{M}_i$: $\mathcal{M}_i \models \{t \leftarrow 3.1,\ u \leftarrow 5.4,\ t \not\simeq u\}$
  - $\mathcal{T}_j^+$-model $\mathcal{M}_j$: $\mathcal{M}_j \models \{t \leftarrow \text{red},\ u \leftarrow \text{blue},\ t \not\simeq u\}$
  - $\mathcal{T}^+$-model $\mathcal{M}$: $\mathcal{M} \models H \cup \{t \not\simeq u\}$
    interpreting 3.1 and red as the same $\mathcal{M}$-element e_1
    interpreting 5.4 and blue as the same $\mathcal{M}$-element e_2
    with $e_1 \neq e_2$

Towards relevance: terms occurring in an assignment

- $\trianglelefteq$ is the subterm relation
- Given assignment $J = \{u_1 \leftarrow \mathfrak{c}_1, \ldots, u_m \leftarrow \mathfrak{c}_m\}$
- The set $G(J)$ of the terms that occur in J contains the assigned terms and their subterms:
  $G(J) = \{t \mid t \trianglelefteq u_i, 1 \leq i \leq m\}$

Relevance for disjoint theories

Term u is relevant to $\mathcal{T}_k$ in a $\mathcal{T}_k$-assignment J if

- Either u occurs in J (i.e., $u \in G(J)$) and $\mathcal{T}_k$ has the sort s of u and has $\mathcal{T}_k$-values for s
- Or term u is an equality $u_1 \simeq_s u_2$ such that
  - u_1 and u_2 occur in assignment J (i.e., $u_1 \in G(J)$, $u_2 \in G(J)$)
  - $\mathcal{T}_k$ has sort s
  - $\mathcal{T}_k$ does not have $\mathcal{T}_k$-values of sort s

Either way $\mathcal{T}_k$ can propose an assignment for term u

Relevance and equality

Relevance determines how different theories make decisions about equalities:

- Suppose that theory $\mathcal{T}$ has sort s
- If $\mathcal{T}$ has $\mathcal{T}$-values for s (i.e., $\mathcal{T}^+$ adds constants of sort s), it can decide whether two terms u_1 and u_2 of sort s are equal by assigning them equal or different values:
  $\{u_1 \leftarrow \mathfrak{c},\ u_2 \leftarrow \mathfrak{c}\}$ or $\{u_1 \leftarrow \mathfrak{c},\ u_2 \leftarrow \mathfrak{q}\}$
- Otherwise, $\mathcal{T}$ can decide whether u_1 and u_2 are equal by assigning a Boolean value to $u_1 \simeq u_2$:
  $(u_1 \simeq u_2) \leftarrow$ true or $(u_1 \simeq u_2) \leftarrow$ false

Relevance: example

- $H = \{x \leftarrow 5,\ f(x) \leftarrow 2,\ f(y) \leftarrow 3\}$
- $x, y : \mathsf{Q}, \quad f : \mathsf{Q} \to \mathsf{Q}, \quad$ LRA and EUF share sort Q
- $H_{\mathsf{LRA}} = H \cup \{x \not\simeq f(x),\ x \not\simeq f(y),\ f(x) \not\simeq f(y)\}$
- $H_{\mathsf{EUF}} = \{x \not\simeq f(x),\ x \not\simeq f(y),\ f(x) \not\simeq f(y)\}$
- x and y are LRA-relevant, not EUF-relevant
- $x \simeq y$ is EUF-relevant, not LRA-relevant
- LRA makes x and y equal/different by assigning them same/different values
- EUF makes x and y equal/different by assigning a truth value to $x \simeq y$

Relevance for predicate-sharing theories

Term u is relevant to $\mathcal{T}_k$ in a $\mathcal{T}_k$-assignment J if

- Either $u \in G(J)$ and $\mathcal{T}_k$ has the sort s of u and $\mathcal{T}_k$ has $\mathcal{T}_k$-values for s
- Or term u is $u_1 \simeq_s u_2$ such that $u_1 \in G(J)$, $u_2 \in G(J)$, $\mathcal{T}_k$ has sort s, but does not have $\mathcal{T}_k$-values of sort s
- Or u is a Boolean term $p(u_1, \ldots, u_m)$ such that $p \colon (s_1 \times \cdots \times s_m) \to \text{prop}$ is a shared predicate symbol by $\mathcal{T}_k$ and at least another theory $\mathcal{T}_j$ and for all i $(1 \leq i \leq m)$ $u_i \in G(J)$ and $\mathcal{T}_k$ has sort s_i

The CDSAT transition system assumes a global basis

- Basic transition system:
  - Trail rules: Decide, Deduce, Fail, ConflictSolve
  - Conflict state rules: Resolve, Backjump, UndoClear, UndoDecide
- Standard transition system:
  - Trail rules: Decide, Deduce, Fail, ConflictSolve
  - Conflict state rules: Resolve, LearnBackjump, UndoClear, UndoDecide
- Parameter: global basis $\mathcal{B}$:
  - A set from which CDSAT can draw new terms
  - Used to prove termination of CDSAT
  - It can be contructed from the local bases

Trail rule Decide and acceptability revisited

$\Gamma_{\mathcal{T}_k}$: the $\mathcal{T}_k$-view of trail Γ

$$\text{Decide}: \quad \Gamma \longrightarrow \Gamma, ?(u \leftarrow \mathfrak{c})$$

if $u \leftarrow \mathfrak{c}$ is an acceptable $\mathcal{T}_k$-assignment for the $\mathcal{T}_k$-module $\mathcal{I}_k$ in Γ_k:

- Term u is relevant to $\mathcal{T}_k$ in $\Gamma_{\mathcal{T}_k}$
- $\Gamma_{\mathcal{T}_k}$ does not assign a $\mathcal{T}_k$-value to term u
- If $u \leftarrow \mathfrak{c}$ is a first-order assignment: for no $J \subseteq \Gamma_{\mathcal{T}_k}$ there exists an $\mathcal{I}_k$-inference $J \cup \{u \leftarrow \mathfrak{c}\} \vdash_{\mathcal{T}_k} L$ and $\overline{L} \in \Gamma_{\mathcal{T}_k}$

Trail rule Deduce revisited

- Transition rule Deduce adds to the trail a Boolean justified assignment ${}_{J\vdash}L$:

$$\text{Deduce}: \Gamma \longrightarrow \Gamma, {}_{J\vdash}L$$

  if a $\mathcal{T}_k$-module $\mathcal{I}_k$ has inference $J \vdash_k L$ for $J \subseteq \Gamma$ and $L \notin \Gamma$ and $\overline{L} \notin \Gamma$
- L is in the global basis $\mathcal{B}$
- Deduce covers both $\mathcal{T}_k$-propagation and preliminary explanation of $\mathcal{T}_k$-conflicts

Explanation of conflicts in CDSAT: recap

- Preliminary explanation of a $\mathcal{T}_k$-conflict by $\mathcal{T}_k$-inferences encapsulated as Deduce steps: not in conflict state
- Until the conflict surfaces as a Boolean conflict: $J \vdash_k L$ and $\overline{L} \in \Gamma$: $H = J \cup \{\overline{L}\}$ is a conflict
- Switch to conflict state $\langle \Gamma; H \rangle$
- Explanation of conflict H by replacing justified assignments in H with their justifications: Resolve transition rule

Conflict state rule Resolve revisited

- Conflict state: $\langle \Gamma; E \uplus \{A\} \rangle$ and ${}_{H}\vdash A \in \Gamma$
- Transition rule Resolve explains the conflict by replacing ${}_{H}\vdash A$ with its justification H:

$$\text{Resolve}: \langle \Gamma; E \uplus \{A\} \rangle \Longrightarrow \langle \Gamma; E \cup H \rangle$$

- Provided H does not contain a first-order decision A' such that $\text{level}_\Gamma(A') = \text{level}_\Gamma(E \uplus \{A\})$ /* case covered by UndoDecide */
- A can be a Boolean or a first-order assignment: if first-order, then input ($H = \emptyset$) and removed from conflict, not from Γ

Conflict state rule Backjump revisited

- Conflict state: $\langle \Gamma; E \uplus \{L\} \rangle$ where $E \neq \emptyset$, assignment L is outstanding in $E \uplus \{L\}$, and $\text{level}_\Gamma(E) = m$
- Transition rule Backjump solves the conflict by jumping back to level m and adding ${}_{E\vdash}\overline{L}$ to the trail:

$$\text{Backjump}: \langle \Gamma; E \uplus \{L\} \rangle \Longrightarrow \Gamma^{\leq m}, {}_{E\vdash}\overline{L}$$

- Bool only: $\text{level}_\Gamma(L)$ is the current level, the trail is a stack
  Theory union: L may be a late propagation, $\text{level}_\Gamma(L)$ may be smaller than the current one, the trail is not a stack

Conflict state rule LearnBackjump revisited

- Conflict state: $\langle \Gamma; E \uplus H \rangle$ where $E \neq \emptyset$ and H contains only Boolean assignments: $H = \{\overline{L}_1, \ldots, \overline{L}_k\}$
- Let $L = L_1 \vee \ldots \vee L_k$: clausal form of H
- Transition rule LearnBackjump solves the conflict by jumping back to a level m such that $\text{level}_\Gamma(E) \leq m < \text{level}_\Gamma(H)$ and adding ${}_{E\vdash}L$:

$$\text{LearnBackjump}: \langle \Gamma; E \uplus H \rangle \Longrightarrow \Gamma^{\leq m}, {}_{E\vdash}L$$

- Provided $L \notin \Gamma$, $\overline{L} \notin \Gamma$, and L is in the global basis $\mathcal{B}$

Conflict state rule UndoDecide

- Conflict state: $\langle \Gamma; E \uplus \{L\} \rangle$ where
  - L is a justified assignment ${}_{H}\vdash L$
  - Justification H contains a first-order decision A
  - $\mathsf{level}_\Gamma(A) = \mathsf{level}_\Gamma(H) = \mathsf{level}_\Gamma(L) = \mathsf{level}_\Gamma(E) = m$
    A determines the level of H (hence of L) and also that of E
- Transition rule UndoDecide solves the conflict by undoing A, removing all assignments of level greater than or equal to m, and posting $\overline{L}$ on the trail as a decision:

$$\text{UndoDecide}: \langle \Gamma; E \uplus \{L\} \rangle \Longrightarrow \Gamma^{\leq m-1}, {}_{?}\overline{L}$$

- A cannot be flipped but its Boolean consequence L can

Conflict state rule UndoDecide

- Decision $\overline{L}$ prevents repeating first-order decision A that caused the conflict, as $\overline{L}$ makes A unacceptable
- Similar to UndoClear, UndoDecide undoes a first-order decision and clears the trail of all assignments of equal or greater level
- Unlike UndoClear, and similar to Backjump or LearnBackjump, UndoDecide flips a Boolean assignment
- Unlike Backjump and LearnBackjump, the flipped Boolean assignment is a decision, not a justified assignment

Why is UndoDecide needed?

- UndoDecide applies when no other conflict state rule does
- UndoDecide applies to a conflict without outstanding assignment: neither Backjump nor UndoClear apply
- LearnBackjump generalizes Backjump by flipping a cube rather than a singleton, but it does not handle a conflict caused by a first-order decision
- Resolve must be forbidden:
  replacing ${}_{H}\vdash L$ with H in E and undoing ${}_{?}A$ by UndoClear cause a loop if Decide reiterates ${}_{?}A$
- Towards an example, let's continue with arrays

CDSAT module for the theory of arrays

The axiomatization of Arr features the select-over-store axioms:

1. $\forall a, v, i.\ \mathsf{select}(\mathsf{store}(a, i, v), i) \simeq v$
2. $\forall a, v, i, j.\ i \not\simeq j \longrightarrow \mathsf{select}(\mathsf{store}(a, i, v), j) \simeq \mathsf{select}(a, j)$

That yield inference rules in the module for Arr:

1. $\{i \simeq j,\ b \simeq \mathsf{store}(a, i, v),\ \mathsf{select}(b, j) \not\simeq v\} \vdash_{\mathsf{Arr}} \bot$
2. $\{i \not\simeq j,\ b \simeq \mathsf{store}(a, i, v),\ \mathsf{select}(b, j) \not\simeq \mathsf{select}(a, j)\} \vdash_{\mathsf{Arr}} \bot$

CDSAT module for the theory of arrays

- When going from axioms to rules, premises are
  - Flattened: e.g., replace $\text{select}(\text{store}(a, i, v), j)$ with $\text{select}(b, j)$ adding $b \simeq \text{store}(a, i, v)$, and
  - Linearized (no repeated variables): e.g., replace $\text{select}(\text{store}(a, i, v), i)$ with $\text{select}(\text{store}(a, i, v), j)$ adding $i \simeq j$
- So that they apply more generally:
  - Flattening and linearization do not require inferences
  - Replacement of equals by equals does
  - Equality rules do not have replacement: only the EUF module reasons about congruence

CDSAT module for arrays with extensionality

- Extensionality axiom:
  $\forall a, b.\ (\forall i.\ \mathsf{select}(a, i) \simeq \mathsf{select}(b, i)) \longrightarrow a \simeq b$
- Clausal form:
  $\mathsf{select}(a, \mathsf{diff}(a, b)) \not\simeq \mathsf{select}(b, \mathsf{diff}(a, b)) \vee a \simeq b$
  Skolem function $\mathsf{diff} : A \times A \to I$ captures the witness index
- Inference rule:
  $a \not\simeq b \vdash_{\mathsf{Arr}} \mathsf{select}(a, \mathsf{diff}(a, b)) \not\simeq \mathsf{select}(b, \mathsf{diff}(a, b))$
- $\mathsf{basis}_{\mathsf{Arr}}(X)$: all subterms of terms in X, equalities btw them, and witness terms $\mathsf{select}(a, \mathsf{diff}(a, b))$, $\mathsf{select}(b, \mathsf{diff}(a, b))$

Example with theory clauses and UndoDecide

- Input set S contains clauses:
  - $C_1 : (i \not\simeq j) \vee (\text{select}(\text{store}(a, i, v), j) < \text{select}(a, j))$
  - $C_2 : (\text{select}(a, j) - \text{select}(a, k)) \simeq 0$
  - $C_3 : (\text{select}(\text{store}(a, i, v), j) \not< \text{select}(a, j)) \vee (\text{select}(a, j) + \text{select}(a, k) \simeq v)$
- Theory union: Bool $\cup$ LRA $\cup$ Arr
- Suppose Arr interprets indices as integers: $I = \mathbb{Z}$ and Arr^+ adds integer constants as Arr-values

Example with theory clauses and UndoDecide

1. Arr-module: Decide $_?(i \leftarrow 0)$ (level 1)
   /* acceptable as i is relevant to Arr */
2. Arr-module: Decide $_?(j \leftarrow 0)$ (level 2)
3. Arr-module: equality inference $\{i \leftarrow 0, j \leftarrow 0\} \vdash_{\text{Arr}} i \simeq j$
   Deduce: $A_1 : {}_J\vdash (i \simeq j)$ with $J = \{i \leftarrow 0, j \leftarrow 0\}$ (level 2)
4. Bool-module: unit propagation
   $\{A_1,\ C_1\} \vdash_{\text{Bool}} \text{select}(\text{store}(a, i, v), j) < \text{select}(a, j)$
   Deduce: $A_2 : {}_I\vdash (\text{select}(\text{store}(a, i, v), j) < \text{select}(a, j))$
   with $I = \{A_1,\ C_1\}$ (level 2)

Example with theory clauses and UndoDecide

5. Bool-module: unit propagation
   $\{A_2,\ C_3\} \vdash_{\mathsf{Bool}} \mathsf{select}(a,j) + \mathsf{select}(a,k) \simeq v$
   Deduce: $A_3 : {}_{H}\vdash (\mathsf{select}(a,j) + \mathsf{select}(a,k) \simeq v)$
   with $H = \{A_2,\ C_3\}$ (level 2)
6. Arr-module: first select-over-store rule
   $\{A_1,\ A_2\} \vdash_{\mathsf{Arr}} v < \mathsf{select}(a,j)$
   Deduce: $A_4 : {}_{G}\vdash (v < \mathsf{select}(a,j))$ with $G = \{A_1,\ A_2\}$ (level 2)
7. LRA-module: FM-resolution $A_3 + C_2$
   $\{A_3,\ C_2\} \vdash_{\mathsf{LRA}} \mathsf{select}(a,j) \simeq v/2$
   Deduce: $A_5 : {}_{M}\vdash (\mathsf{select}(a,j) \simeq v/2)$ with $M = \{A_3,\ C_2\}$ (level 2)

Example with theory clauses and UndoDecide

LRA-conflict: $E_0 = \{A_4,\ A_5\}$
as $A_4 : {}_{G}\vdash (v < \text{select}(a, j))$ and $A_5 : {}_{M}\vdash (\text{select}(a, j) \simeq v/2)$

8. E_0 contains literals A_4 and A_5 of max level (2)
   Resolve: $E_1 = \{A_4,\ A_3,\ C_2\}$
9. E_1 contains literals A_3 and A_4 of max level (2)
   Resolve: $E_2 = \{A_1,\ A_2,\ A_3,\ C_2\}$
10. E_2 contains literals A_1, A_2 and A_3 of max level (2)
    Resolve: $E_3 = \{A_1,\ A_2,\ C_3,\ C_2\}$
11. E_3 contains literals A_1, and A_2 of max level (2)
    Resolve: $E_4 = \{A_1,\ C_1,\ C_3,\ C_2\}$

Example with theory clauses and UndoDecide

$E_4 = \{A_1,\ C_1,\ C_3,\ C_2\}$

E_4 contains an outstanding literal: $\mathsf{level}_\Gamma(A_1) = 2 = \mathsf{level}_\Gamma(E_4)$
Let $m = 2$

A_1 is ${}_J\vdash(i \simeq j)$ and $J = \{i \leftarrow 0,\ j \leftarrow 0\}$
where ${}_?(j \leftarrow 0)$ also has level $m = 2$

Apply Resolve to replace A_1 with J
and UndoClear to undo ${}_?(j \leftarrow 0)$?

No, the system could loop by repeating ${}_?(j \leftarrow 0)$
(still acceptable for Arr)

Example with theory clauses and UndoDecide

12. UndoDecide: undo $_?(j \leftarrow 0)$, go back to level $m - 1 = 1$, and add decision $_?(i \not\simeq j)$ (level 2)
    /* clause C_1 is satisfied */
13. LRA-module: Decide $_?(\text{select}(a, j) \leftarrow 1)$ (level 3)
14. LRA-module: Decide $_?(\text{select}(a, k) \leftarrow 1)$ (level 4)
    /* clause C_2 is satisfied */
15. LRA-module: Decide $_?(v \leftarrow 2)$ (level 5)
    /* clause C_3 is satisfied */

Example with theory clauses and UndoDecide: variant

Suppose theory Arr does not have values for array indices:
i and j not relevant to Arr, Arr-module cannot decide their values

1. Arr-module: Decide $_?(i \simeq j)$ (level 1)
   /* acceptable as $i \simeq j$ is relevant to Arr */
2. The same transitions as before lead to conflict
   $\{i \simeq j,\ C_1,\ C_3,\ C_2\}$ (level 1)
3. Backjump or LearnBackjump backtracks to level 0 and places
   $_{N\vdash}(i \not\simeq j)$ on the trail with $N = \{C_1,\ C_3,\ C_2\}$
4. The satisfiability of the clauses can be detected as before

Three main theorems

- Input assignment: H_0
- All terms occurring in H are in global basis $\mathcal{B}$
- CDSAT is
  - Sound: if all theory modules are sound,
    if CDSAT returns unsat, H_0 is unsatisfiable
  - Terminating: if $\mathcal{B}$ is finite,
    CDSAT is guaranteed to terminate
  - Complete: if the leading theory module is complete
    and the others are leading-theory-complete,
    if CDSAT terminates without returning unsat,
    there exists a $\mathcal{T}^+$-model of Γ and hence of H_0

CDSAT for data structures

- Theories of data structures:
  - CDSAT allows to increase expressivity
  - Motivation for the extension of CDSAT to predicate-sharing theories
- Arrays, maps, and vectors with abstract domain

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

Arrays: finite of infinite?

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: $\mathsf{len}(a) \simeq n$
- Revised axiom of extensionality for integer-index arrays:
  $\forall a,\ b.\ [\ \mathsf{len}(a) \simeq \mathsf{len}(b) \wedge$
  $(\forall i.\ 0 \leq i < \mathsf{len}(a) \rightarrow \mathsf{select}(a, i) \simeq \mathsf{select}(b, i))\ ] \rightarrow a \simeq b$
- 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
  $\forall i.\ l \leq i \leq u \ \rightarrow\ \text{select}(a, i) \simeq \text{select}(b, i)$
- The theory of arrays and its limitations remain unchanged
- Efficient quantifier reasoning may be challenging

[Bradley, Manna, Sipma: VMCAI 2006]

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)

[Sheng et al.: IJCAR 2022, JAR 2023], [Ait-El-Hara, Bobot, Bury: 2025]

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

[Wang, Appel: JAR 2023], [Ge, de Moura: CAV 2009]

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 unions including these theories are decidable:
  from fitting the three 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
- select: $A \times I \to V$ store: $A \times I \times V \to A$ len: $A \to L$
- Free admissibility predicate: Adm: $I \times L \to$ prop
  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:
  - $\forall\ a,\ v,\ i.\ \mathsf{select}(\mathsf{store}(a, i, v), i) \simeq v$ is replaced by
    $\forall\ a,\ v,\ i.\ \mathsf{Adm}(i, \mathsf{len}(a)) \rightarrow \mathsf{select}(\mathsf{store}(a, i, v), i) \simeq v$
    a store at an inadmissible index has no effect
  - $\forall\ a,\ v,\ i,\ j.\ i \not\simeq j \rightarrow \mathsf{select}(\mathsf{store}(a, i, v), j) \simeq \mathsf{select}(a, j)$
- Store does not change length:
  $\forall\ a,\ i,\ v.\ \mathsf{len}(\mathsf{store}(a, i, v)) \simeq \mathsf{len}(a)$
- Extensionality:
  $\forall\ a,\ b.\ [\ \mathsf{len}(a) \simeq \mathsf{len}(b) \wedge$
  $(\forall i.\ \mathsf{Adm}(i, \mathsf{len}(a)) \rightarrow \mathsf{select}(a, i) \simeq \mathsf{select}(b, i))\ ]$
  $\rightarrow\ a \simeq b$

Example: the most common interpretation

- Let LIA be the theory defining Adm
- Indices and lengths are integers
- $\forall i,\ n.\ \mathrm{Adm}(i, n) \leftrightarrow 0 \leq i < n$
- The set of admissible indices is the interval $[0, n)$
- Under this interpretation extensionality in ArrAD covers extensionality 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
- $\forall i,\ addr,\ n.\ \mathrm{Adm}(i, (addr, n)) \leftrightarrow 0 \leq i < n$
- 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 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:
  $\forall\ a,\ j,\ i,\ v.\ \mathrm{Adm}(j, \mathrm{len}(\mathrm{store}(a, i, v))) \leftrightarrow (\mathrm{Adm}(j, \mathrm{len}(a)) \vee j \simeq i)$
- Store does not change length if the index is admissible:
  $\forall\ a,\ i,\ v.\ \mathrm{Adm}(i, \mathrm{len}(a)) \rightarrow \mathrm{len}(\mathrm{store}(a, i, v)) \simeq \mathrm{len}(a)$
- Select-over-store axioms:
  - Restored: $\forall\ a,\ v,\ i.\ \mathrm{select}(\mathrm{store}(a, i, v), i) \simeq v$
  - $\forall\ a,\ v,\ i,\ j.\ i \not\simeq j \rightarrow \mathrm{select}(\mathrm{store}(a, i, v), j) \simeq \mathrm{select}(a, j)$
- Extensionality 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 (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:
  $\forall\ a,\ j,\ i,\ v.\ \mathrm{Adm}(j, \mathrm{len}(\mathrm{store}(a, i, v))) \ \leftrightarrow\ (\mathrm{Adm}(j, \mathrm{len}(a)) \vee j \leq i)$
- Everything else as in MapAD
  except for adding to the signature an ordering $<$ on indices
  (does not have to be linear)
- Dynamic data structures axiomatized for the first time:
  The theories of sequences do not do it

Theory modules for ArrAD, MapAD, VecAD

- From axioms to inference rules, e.g.:
  - $a \simeq b \ \vdash \ \mathsf{len}(a) \simeq \mathsf{len}(b)$
  - For ArrAD:
    $b \simeq \mathsf{store}(a, i, v),\ \mathsf{len}(b) \not\simeq \mathsf{len}(a) \ \vdash \ \bot$
  - For MapAD and VecAD:
    $\mathsf{len}(a) \simeq n,\ \mathsf{Adm}(i, n),\ b \simeq \mathsf{store}(a, i, v),\ \mathsf{len}(b) \not\simeq \mathsf{len}(a) \ \vdash \ \bot$
- 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:

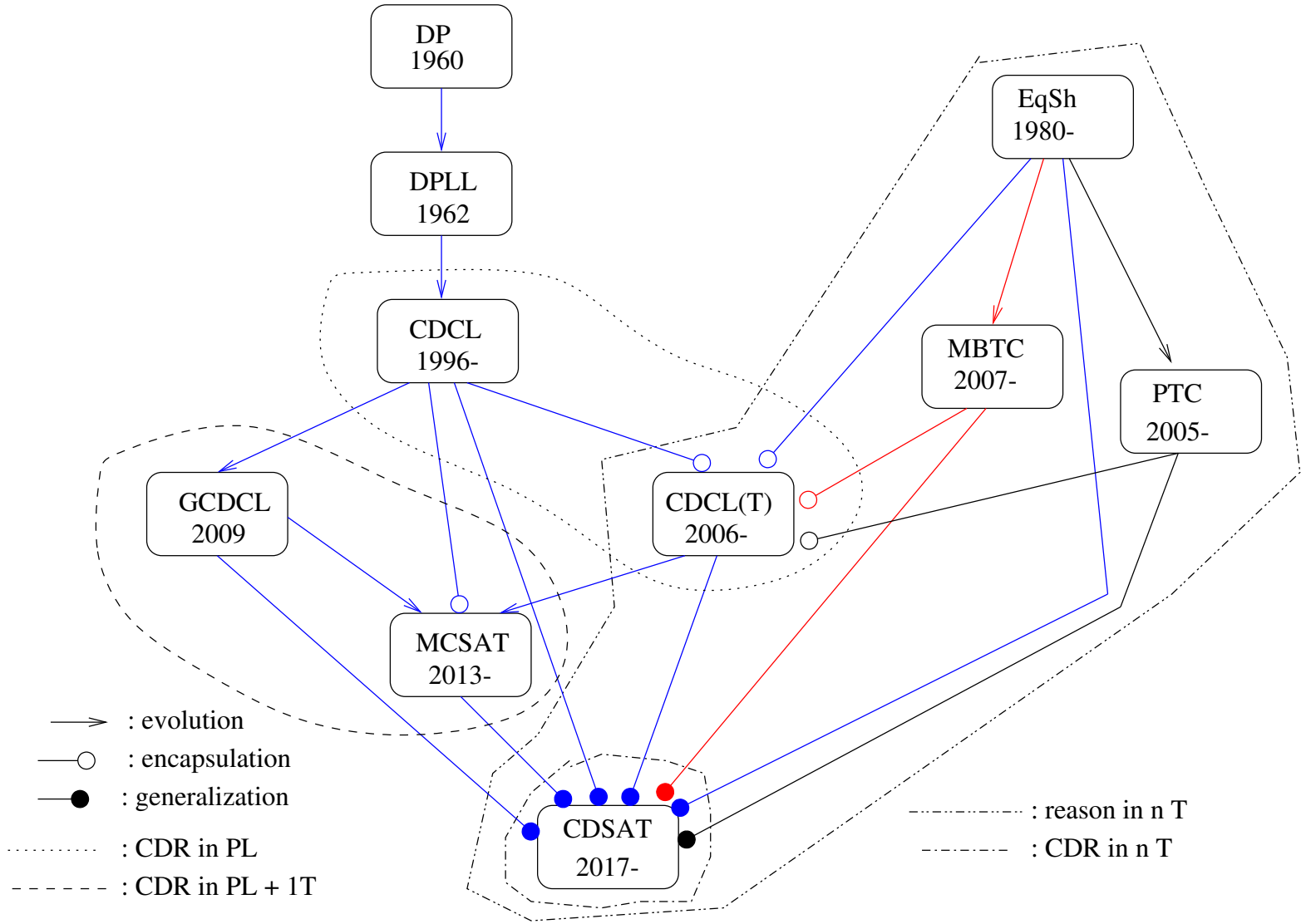
- if function f is in X_n and
- g is the function that maps i to v and is identical to f otherwise, then
- $\exists m$ such that $g \in X_m$ and
  - Either $m = n$ (store at an admissible i)
  - Or $I_m = I_n \cup \{i\}$
    (store at an inadmissible index i which is admissible in the resulting map)

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)

The big picture: more theory combination



Model-based theory combination (MBTC)

[de Moura, Bjørner: SMT 2007]

- Variant of equality sharing in CDCL($\mathcal{T}$)
- Only for $\mathcal{T}_k$ such that the $\mathcal{T}_k$-procedure builds a candidate model $\mathcal{M}_k$ (e.g., LRA and Simplex)
- Idea: share $u \simeq v$ if $\mathcal{M}_k \models_{\mathcal{T}_k} (u \simeq v)$
- Motivation: enumerating $u \simeq v$ s.t. $\mathcal{M}_k \models_{\mathcal{T}_k} (u \simeq v)$ less expensive than enumerating $u \simeq v$ true in all $\mathcal{T}_k$-models consistent with the current Boolean candidate model

Model-based theory combination (MBTC)

- Propositional abstraction of $u \simeq v$ is posted on the CDCL($\mathcal{T}$) trail by a decision
- If a conflict ensues, the decision is undone by CDCL backjumping and the $\mathcal{T}_k$-procedure amends $\mathcal{M}_k$
- Terms u and v cannot be new
- $\mathcal{M}_k$ and conflict-driven updates remain inside the black-box $\mathcal{T}_k$-procedure
- Useful to accelerate reaching sat

CDSAT generalizes MBTC

- All theory modules cooperate as peers to build a model for the union of the theories on the shared trail
- A theory module $\mathcal{I}_k$ can build a partial $\mathcal{T}_k$-model $\mathcal{M}_k$ publicly on the shared trail by first-order decisions
- $\mathcal{I}_k$ can deduce an equality $u \simeq v$ that follows from assignments in $\mathcal{M}_k$, but is not a logical consequence of either the input or the Boolean contents of the trail because CDSAT modules deduce from first-order assignments

CDSAT generalizes MBTC

- If a conflict ensues, $u \simeq v$ and the first-order decisions from which it depends will be undone by undoclear or undodecide amending $\mathcal{M}_k$
- MBTC does it with a Boolean decision, because CDCL($\mathcal{T}$) deduces only $\mathcal{T}$-valid lemmas
- CDSAT opens the black-box procedures and achieves by theory inferences and first-order decisions on the shared trail what in other methods is hidden in the theory procedure

CDCL($\mathcal{T}$) with splitting on demand

- Entrust to CDCL the reasoning about any disjunction produced by the procedure of a non-convex theory
- The atoms in the clause must come from the existing clause set (i.e., input + equalities btw shared variables) or a literal-generating function (for termination)
- CDSAT: terms inferred by theory module inferences must come from finite global basis (for termination)
- CDSAT decisions are restricted by acceptability (which prevents $t \leftarrow \mathfrak{c}_1, \ldots, t \leftarrow \mathfrak{c}_n, t \leftarrow \mathfrak{c}_{n+1}, \ldots$) not by the global basis

CDCL($\mathcal{T}$) with splitting on demand

A first-order decision $t \leftarrow \mathfrak{c}$ in CDSAT cannot be mimicked by generating $t \simeq \mathfrak{c} \vee t \not\simeq \mathfrak{c}$ and then splitting it on demand, because

- $\mathfrak{c}$ is not a term, it is a value coming from a theory extension (e.g., any rational number)
- No literal-generating function can produce $t \simeq \mathfrak{c}$ for an arbitrary $\mathfrak{c}$, because it would violate finiteness

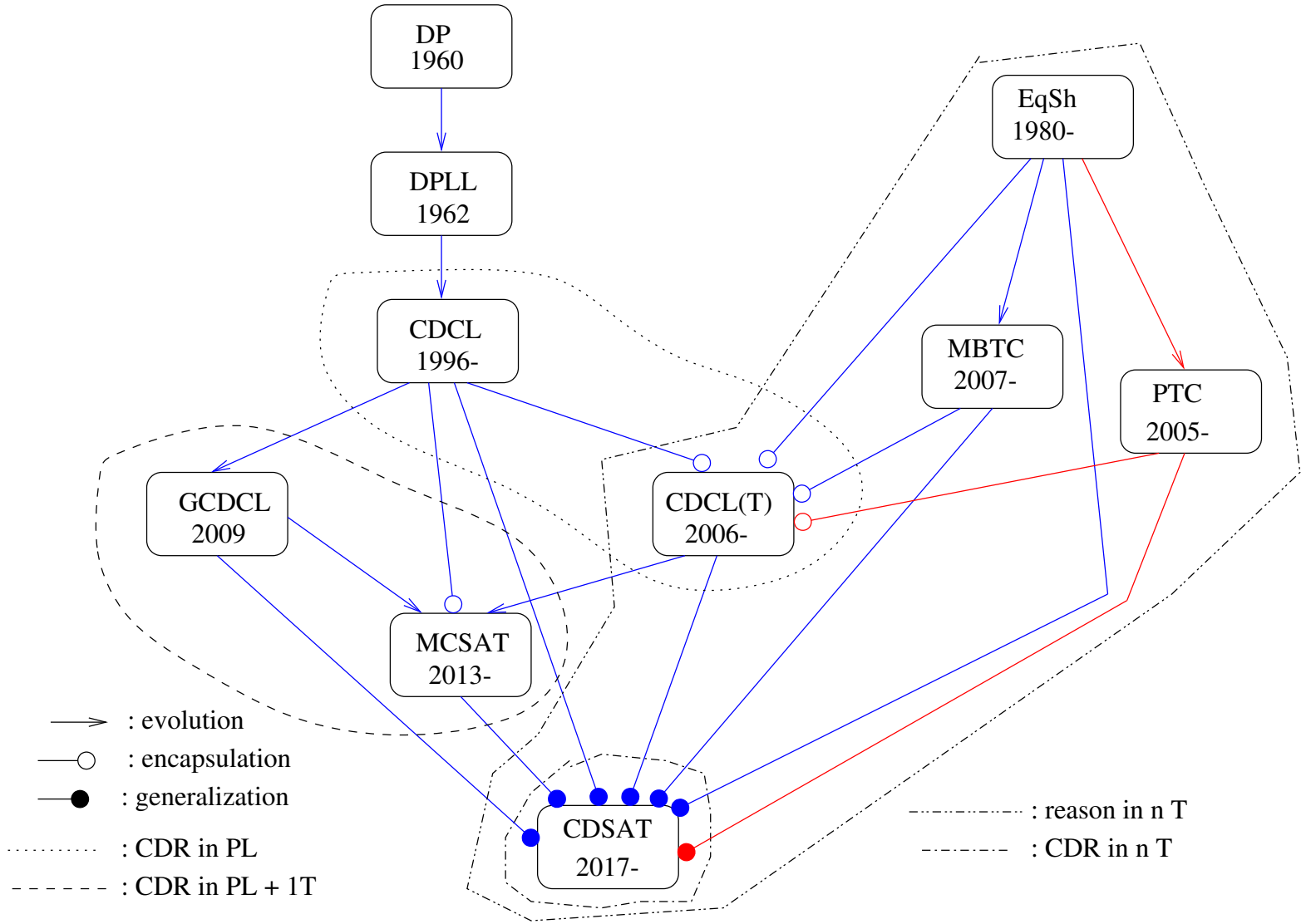
CDCL($\mathcal{T}$) with splitting on demand

If a conflict-driven $\mathcal{T}_k$-procedure were integrated in CDCL($\mathcal{T}$):

- It'd be treated as a black-box
- Its $\mathcal{T}_k$-specific conflict-driven mechanisms, namely
  - First-order $\mathcal{T}_k$-assignments, and
  - New atoms generated by $\mathcal{T}_k$-inferences

  would remain private, would not be exported to the CDCL($\mathcal{T}$) trail and could not guide the global search

The big picture: more theory combination



Polite theory combination (PTC)

[Jovanović, Barrett: LPAR 2010] [Toledo, Przybocki, Zohar: CADE 2025]

- Variant of equality sharing in CDCL($\mathcal{T}$)
- Equality sharing requires the theories to be stably infinite
- PTC allows $\mathcal{T}_1$ not stably infinite, but $\mathcal{T}_2$ satisfies stronger cardinality requirements: strongly polite
- PTC combines theories by combining procedures
- Procedures combined as black-boxes
- Completeness approach like equality sharing: hypotheses on theories + combination theorem

CDSAT requires neither stable infiniteness nor strong politeness

CDSAT and agreement on cardinalities of shared sorts

- CDSAT requires that there exists leading theory, say $\mathcal{T}_1$, that
  - Has all sorts in the theory union
  - Has all cardinality constraints aggregated and enforced by $\mathcal{T}_1$-module inferences
- Every $\mathcal{T}_k$ $(k \neq 1)$ has to agree with $\mathcal{T}_1$ on what's shared: any two $\mathcal{T}_k$ and $\mathcal{T}_j$ $(k \neq j)$ agree
- Agreement guaranteed by theory modules completeness requirements
- Different approach to completeness:
  - $\mathcal{T}_1$-module complete
  - $\mathcal{T}_k$-module $(k \neq 1)$ leading-theory-complete

Examples about cardinalities of shared sorts

1. All theories stably infinite: $\mathcal{T}_1$ is fictional $\mathcal{T}_{\mathbb{N}}$ that interprets all sorts (except prop) as having the cardinality of $\mathbb{N}$
2. At-most-m cardinality constraint on sort s:
   $\forall x_0, \ldots, \forall x_m.\ \bigvee_{0 \leq i \neq k \leq m} x_i \simeq_s x_k$
   $x_0, \ldots, x_m$: $m+1$ distinct variables of sort s

   Inference rule in the $\mathcal{T}_1$-module:
   $\bigwedge_{0 \leq i \neq k \leq m} u_i \not\simeq_s u_k \vdash_{\mathcal{T}_1} \bot$
   $u_0, \ldots u_m$: any $m+1$ distinct terms of sort s
3. Aggregation: if $\mathcal{T}_2$ says at-most-m and $\mathcal{T}_3$ says at-most-p, leading theory $\mathcal{T}_1$ says at-most-$min(m, p)$

References: CDSAT for disjoint theories

- Satisfiability modulo theories and assignments.
  Proc. CADE-26, LNAI 10395, 42–59, Springer, 2017
- Proofs in conflict-driven theory combination.
  Proc. CPP-7, ACM Press, 186–200, 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

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

References: CDSAT for predicate-sharing theories

- CDSAT for nondisjoint theories with shared predicates: arrays with abstract length.
  Proc. SMT-20, CEUR 3185, 18–37, CEUR WS-org, 2022
- CDSAT for predicate-sharing theories: arrays, maps, and vectors with abstract domain.
  Submitted to journal, Dec. 2025, 46 pages

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

References: surveys and invited, less technical

- On conflict-driven reasoning.
  Proc. AFM-7, Kalpa Publications 5, 31–49, EasyChair, 2018
- Conflict-driven reasoning in unions of theories. (Abstract)
  Proc. FroCoS-12, LNAI 11715, xi–xiii, Springer, 2019
- Proof generation in CDSAT. (Abstract)
  Proc. PxTP-7, EPTCS 366, 1–4, 2021
- The CDSAT method for satisfiability modulo theories and assignments: an exposition.
  Proc. CiE-21, LNAI 15764, 1–16, Springer, 2025

Author: MPB

A forthcoming book: stay tuned

- Conflict-Driven Procedures for Automated Reasoning.
  Research monograph, Springer Series on Theories and Applications of Computability, in preparation, and expected June 2027.

Author: MPB

Thank you!