The CDSAT Method for Satisfiability Modulo Theories and Assignment: an Exposition

Maria Paola Bonacina[1][0000−0001−9104−2692]

Dipartimento di Informatica, Università degli Studi di Verona, Verona, Italy
mariapaola.bonacina@univr.it
https://mariapaola.github.io/

Abstract. Most SMT instances involve symbols from more than one theory. The equality sharing method for combining theory satisfiability procedures has been a standard for over forty years. For the Boolean part, the equality sharing method is interfaced with the CDCL procedure for SAT. CDCL guides the search by learning lemmas from conflicts between clauses and candidate model. In the standard integration of CDCL and equality sharing, the conflict-driven reasoning remains propositional, even if conflict-driven theory procedures exist. The MCSAT method integrates CDCL with a single conflict-driven theory procedure. The CDSAT method generalizes CDCL, MCSAT, and equality sharing, by allowing the integration of multiple (conflict-driven or not) theory procedures.

Keywords: Conflict-driven satisfiability procedures · Satisfiability modulo theory and assignment · Combination of theories.

1 Introduction

Automated reasoning (AR) is a foundation of computer science, since the quest for a formulation and a solution of the *Entscheidungsproblem* led Alan Turing to invent Turing machines in 1936. AR is also a foundation of artificial intelligence (AI), since machines that prove theorems have been a defining objective of AI since the Dartmouth Conference in 1956. AR is also an essential methodology for the analysis, verification, synthesis, and optimization of software. Logic proved to be the calculus of computation, as predicted by John McCarthy in 1963 [9]. AR is applied to proving verification/synthesis conditions, refining abstractions for model checking, generating tests for testing and examples for synthesis. Current goals in this field include correct-by-construction software, provable privacy, and verification of distributed protocols, distributed systems, and randomized algorithms. While generative AI as in chatbots can be used by anyone, AR is currently used mostly by computer scientists and mathematicians. The integration of AR and generative AI towards a better AI is a fascinating challenge (e.g., [3]).

Automated reasoning is concerned with the design and implementation of computer programs that solve problems formulated as validity or satisfiability queries in a logic or a theory. Validity queries are handled refutationally: a conjecture is shown valid by showing that its negation is unsatisfiable. AR procedures

involve *inference* and *search*. In this paper we consider contexts where satisfiability is *decidable*, the problem is *quantifier-free*, and written as a *set of clauses*. We study *conflict-driven reasoning procedures* [4] that are at the heart of solvers for propositional satisfiability (SAT) and satisfiability modulo theories (SMT).

Conflict-driven reasoning procedures search for a model (to establish satisfiability) and perform inferences (to prove unsatisfiability), in such a way that search and inferences guide each other. The procedure searches for a model by proposing *assignments* to terms and *propagating* their consequences. Inferences are performed (almost) only when the procedure encounters a *conflict* or contradiction (e.g., one of the clauses to be satisfied is false in the current assignment). The inferences *explain* the conflict, resulting in the generation of a *lemma*. These inferences are steps towards a possible refutation. The lemma excludes the assignments that caused the conflict. Learning the lemma ensures that the model search will not incur in the same conflict, and the lemma is used to drive the search elsewhere. In a nutshell, the model search focuses the inferences on solving conflicts, and the inferences allow the search to escape dead-end's.

This paper gives an exposition of the CDSAT (*Conflict-Driven SATisfiability*) method [6, 8, 7]: to the best of our knowledge, CDSAT is currently the *most general conflict-driven reasoning procedure*. Its novel features include the following. CDSAT decides the satisfiability of a set of clauses modulo a *union of theories* and a possibly empty *initial assignment* (*satisfiability modulo theories and assignment*). The initial assignment may contain *both Boolean* (e.g., $L \leftarrow \mathsf{true}$) and *first-order* (e.g., $x \leftarrow 3$) assignments to terms occurring in the problem. The answer is sat, if there exists a satisfying assignment that includes the initial one, and unsat otherwise. CDSAT combines theories, not black-box theory procedures. The CDSAT transition system coordinates *theory modules* (theory inference systems), which collaborate as peers over a *trail* representing a candidate model by Boolean and first-order assignments. Since conflict-driveness is provided by CDSAT, the distinction between theory with conflict-driven procedure and theory without fades. CDSAT does not need the traditional architecture with the SAT solver at the center and the theory procedure(s) as satellite(s). Propositional logic is one of the theories, the *Boolean theory* denoted Bool.

The proceeding sections interleave outlines of conflict-driven reasoning procedures with examples, describing how CDSAT encompasses the other procedures. The CDSAT transition rules are described informally showing their behavior in the examples. A complete presentation with formal definitions, results, and proofs can be found in [6, 8]. More references can be found in [4–6, 8].

2 From CDCL to CDSAT

The CDCL (*Conflict-Driven Clause Learning*) procedure [23, 22] for SAT (the problem of deciding the satisfiability of a set S of propositional clauses) evolved from the DPLL (Davis-Putnam-Logemann-Loveland) procedure [10], which in turn evolved[1] from the DP (Davis-Putnam) procedure [11]. CDCL was called

[1] A historical account can be found in [21].

DPLL or DPLL-CDCL until it was agreed that the two procedures deserve distinct names. The DP procedure featured rules incorporating *resolution* (clauses $C \vee L$ and $D \vee \overline{L}$ resolve upon complementary literals L and $\overline{L}$ to generate resolvent $C \vee D$), its special case *unit resolution* (L and $D \vee \overline{L}$ resolve to generate D), and *subsumption* (L subsumes $C \vee L$, and D subsumes $D \vee \overline{L}$). DPLL replaced resolution with *splitting*: consider assignment $L \leftarrow \mathsf{true}$ (written L), propagate its consequences, and if a conflict arises, backtrack to $L \leftarrow \mathsf{false}$ or $\overline{L} \leftarrow \mathsf{true}$ (written $\overline{L}$). DPLL demphasized inference in favor of backtracking search over partial models, represented as a *trail* of *Boolean assignments*. Propagation was *unit propagation* viewed as unit resolution and subsumption: if L is on the trail, clause $C \vee L$ is deleted, and clause $D \vee \overline{L}$ is replaced by D. A conflict arises if S contains $\{L,\ \overline{L}\}$. A splitting is done if no propagation is possible. The trail is organized as a *stack*: backtracking consists of popping L and pushing $\overline{L}$.

CDCL replaced chronological backtracking with *backjumping* and splitting with *decision*: guess L, provided neither L nor $\overline{L}$ is on the trail. If chronological backtracking is abandoned, $\overline{L}$ is not necessarily considered right after L, and a tentative assignment is a decision, rather than a case in a case analysis. Propagation replaced unit resolution and unit subsumption with detection of *implied literals* and *conflict clauses*. If $(C \vee L) \in S$ and the trail contains $\overline{Q}$ for every literal Q in C, literal L is *implied* and added to the trail with *justification* $C \vee L$. If $C \in S$ and the trail contains $\overline{Q}$ for every literal Q in C, clause C is in *conflict*. Every decision opens a new *level* on the trail, still organized as a stack. All implied literals belong to the level of the most recent decision, except those implied literals that are unit input clauses, which belong to level 0.

CDCL brought resolution back, applying it only to *explain* a conflict, by resolving a conflict clause containing L with the justification of literal $\overline{L}$ on the trail. One of the resolvents is retained as a lemma, and used to direct the backjumping to a level of the trail where the lemma can be satisfied and the search can restart. The choice of how much resolution to do and which resolvent to learn is a matter of heuristic. The popular *1UIP heuristic* prescribes to explain the conflict by resolution until it generates an *assertion clause*, namely a conflict clause C such that only one of its literals, the *assertion literal* L, is falsified by the current level of the trail. After learning C, the procedure jumps back to the smallest level where L is undefined, all other literals in C are false, and hence the conflict is solved by placing L on the trail with C as justification.

In CDSAT clause C is viewed as an abbreviation of the assignment $C \leftarrow \mathsf{true}$. There is no need to separate input set S and trail, and all input assignments sit on the trail at level 0. Decisions are performed by a transition rule named Decide. A decision is written as ${}_{?}L$ to record that it is a guess. CDSAT requires that every decision be *acceptable*. A Boolean decision ${}_{?}L$ is acceptable if neither L nor $\overline{L}$ is on the trail. The notion of implied literal is generalized to that of *justified assignment*: if there is a clause $C \vee L$ and the trail contains $\overline{Q}$ for every literal Q in C, the justified assignment ${}_{J\vdash}L$ is added to the trail, with $J = \{C \vee L, \neg C\}$ as *justification*. Propagations are performed by a transition rule named Deduce. A justified assignment ${}_{J\vdash}L$ belongs to the level of its justification J, which is the

maximum among the levels of the elements of J. Input assignments are justified assignments with empty justification: input clause C is written as ${}_{\emptyset\vdash}C$. The trail is not necessarily a stack, as ${}_{J\vdash}L$ can be added after assignments of level higher than that of J. In this case, ${}_{J\vdash}L$ is called a *late propagation*.

In CDSAT the notion of conflict clause is replaced by that of *conflicting assignment*, or *conflict* tout court, meaning an unsatisfiable set E of assignments. Resolve transitions *explain* E by replacing an element ${}_{J\vdash}L$ of E with its justification J. Explanation continues until E contains only one literal whose level m is maximum in E. This is a generalization of 1UIP, because level m is not necessarily the current one. A LearnBackjump transition *learns* a clause ${}_{J\vdash}C$, where $C = L_1 \vee \ldots \vee L_k$ is the negation of a Boolean subset $H = \{\overline{L}_1, \ldots, \overline{L}_k\}$ of conflict E, and J is the rest of E, that is, $E = J \uplus H$. Indeed, if $S \cup (J \uplus H) \models \bot$, then $S \cup J \models C$. A conjunction, or set, of literals is called a *cube*. Then LearnBackjump jumps back to a level that is strictly smaller than that of H, in order to quit the conflict, and greater than or equal to that of J, so that ${}_{J\vdash}C$ can be placed on the trail. When LearnBackjump follows 1UIP, clause C is the first assertion clause and the destination level is the one prescribed by 1UIP. LearnBackjump can apply also other heuristics. Example 1 illustrates CDCL with 1UIP as done by CDSAT. The only active module is that of theory Bool.

Example 1. Let $S = \{\overline{A} \vee B,\ \overline{A} \vee C \vee E,\ \overline{B} \vee D,\ \overline{C} \vee \overline{D},\ A \vee \overline{B} \vee E,\ B \vee \overline{C},\ F \vee \overline{E}\}$ be a subset of the input, where A, B, C, D, E, and F are propositional variables [18]. The trail contains ${}_{\emptyset\vdash}\overline{A} \vee B$ and all other input clauses at level 0. Suppose that in order to satisfy other clauses a Decide transition adds ${}_{?}\overline{F}$ to the trail. Say the trail already had $n-1$ levels prior to this decision, so that ${}_{?}\overline{F}$ opens level n. The Bool-module infers by unit propagation $\{F \vee \overline{E}, {}_{?}\overline{F}\} \vdash_{\mathsf{Bool}} \overline{E}$ and a Deduce transition adds ${}_{J\vdash}\overline{E}$ to the trail with $J = \{F \vee \overline{E}, {}_{?}\overline{F}\}$. The level of ${}_{J\vdash}\overline{E}$ is n, because the level of J is n, since the level of ${}_{?}\overline{F}$ is n and that of $F \vee \overline{E}$ is 0. Suppose that two other Decide transitions create levels $n+1$ and $n+2$, and a third one adds ${}_{?}A$ opening level $n+3$. More unit propagations and Deduce transitions add to the trail ${}_{H\vdash}B$ with $H = \{\overline{A} \vee B, {}_{?}A\}$, ${}_{I\vdash}C$ with $I = \{\overline{A} \vee C \vee E, {}_{J\vdash}\overline{E}, {}_{?}A\}$, and ${}_{K\vdash}D$ with $K = \{\overline{B} \vee D, {}_{H\vdash}B\}$. All these justified assignments have level $n+3$.

The unit propagation $\{\overline{C} \vee \overline{D},\ {}_{I\vdash}C\} \vdash_{\mathsf{Bool}} \overline{D}$ reveals a conflict, as ${}_{K\vdash}D$ is on the trail. The conflict is $E_0 = \{\overline{C} \vee \overline{D},\ {}_{I\vdash}C,\ {}_{K\vdash}D\}$. Clause $\overline{C} \vee \overline{D}$ is the conflict clause, but not an assertion clause. E_0 contains two literals of maximum level, namely ${}_{I\vdash}C$ and ${}_{K\vdash}D$ (both have level $n+3$). Resolve explains E_0 by replacing one of these two literals, say D, with its justification. The resulting conflict is $E_1 = \{\overline{C} \vee \overline{D},\ {}_{I\vdash}C,\ \overline{B} \vee D,\ {}_{H\vdash}B\}$. This unfolding of the conflict corresponds to the resolution step that resolves $\overline{C} \vee \overline{D}$ and $\overline{B} \vee D$ to yield $\overline{B} \vee \overline{C}$. The literal resolved upon D is the one explained away when going from E_0 to E_1. The new conflict clause $\overline{B} \vee \overline{C}$ is the negation of the cube $\{{}_{H\vdash}B,\ {}_{I\vdash}C\}$ in E_1.

Since E_1 does not have a unique literal of highest level (both ${}_{I\vdash}C$ and ${}_{H\vdash}B$ belong to level $n+3$), Resolve explains E_1 by replacing ${}_{I\vdash}C$ with I, yielding $E_2 = \{\overline{C} \vee \overline{D},\ \overline{A} \vee C \vee E,\ {}_{J\vdash}\overline{E},\ {}_{?}A,\ \overline{B} \vee D,\ {}_{H\vdash}B\}$. This corresponds to resolving $\overline{B} \vee \overline{C}$ and $\overline{A} \vee C \vee E$ upon C to yield conflict clause $\overline{B} \vee \overline{A} \vee E$, negation of cube $\{{}_{H\vdash}B,\ {}_{?}A,\ {}_{J\vdash}\overline{E}\}$ in E_2. Since E_2 does not have a unique literal of highest

level (both $_?A$ and $_H\vdash B$ belong to level $n+3$), Resolve explains E_2 by replacing $_H\vdash B$ with H, yielding $E_3 = \{\overline{C} \vee \overline{D},\ \overline{A} \vee C \vee E,\ _J\vdash \overline{E},\ _?A,\ \overline{B} \vee D,\ \overline{A} \vee B\}$. This corresponds to resolving $\overline{B} \vee \overline{A} \vee E$ and $\overline{A} \vee B$ upon B to yield conflict clause $\overline{A} \vee E$, negation of cube $\{_?A,\ _J\vdash \overline{E}\}$ in E_3.

As $_?A$ is the only literal of maximum level in E_3 (and $\overline{A} \vee E$ is the first assertion clause), 1UIP prescribes to learn $\overline{A} \vee E$ (we learn that no model of S can contain A and $\overline{E}$), and backjump to the smallest level where $\overline{A}$ is undefined and E is still false. This level is n. LearnBackjump takes us back to level n, clearing levels $n+1$, $n+2$ and $n+3$, and adding lemma $_G\vdash(\overline{A} \vee E)$ to the trail, where $G = \{\overline{C} \vee \overline{D},\ \overline{A} \vee C \vee E,\ \overline{B} \vee D,\ \overline{A} \vee B\}$. Level n contains $_?\overline{F},\ _J\vdash \overline{E}$, and $_G\vdash(\overline{A} \vee E)$. Deduce performs unit propagation adding to the trail $_M\vdash \overline{A}$ with $M = \{_G\vdash(\overline{A} \vee E), _J\vdash \overline{E}\}$, $_N\vdash \overline{B}$ with $N = \{A \vee \overline{B} \vee E,\ _M\vdash \overline{A},\ _J\vdash \overline{E}\}$, and $_P\vdash \overline{C}$ with $P = \{B \vee \overline{C}, _N\vdash \overline{B}\}$. The trail contains a model of S, namely $\{\overline{E},\ \overline{A},\ \overline{B},\ \overline{C}\}$, and the search can continue to take care of the other input clauses.

Several authors approached the SMT problem (deciding the satisfiability of a set of clauses in a first-order theory $\mathcal{T}$) by integrating in CDCL a decision procedure for the $\mathcal{T}$-satisfiability of a set of $\mathcal{T}$-literals. This integration was systematized in the DPLL($\mathcal{T}$) framework [26], later renamed CDCL($\mathcal{T}$). In this framework, CDCL works on a propositional abstraction of the first-order problem, where $\mathcal{T}$-atoms are replaced with propositional variables. The $\mathcal{T}$-satisfiability procedure detects *theory conflicts* and contributes *theory propagations*. Suppose that literals $L_1, \ldots, L_n$ are on the trail. If the $\mathcal{T}$-procedure detects that $\{L_1, \ldots, L_n\}$ is $\mathcal{T}$-unsatisfiable, it signals a $\mathcal{T}$-conflict with conflict clause $C = \overline{L_1} \vee \ldots \vee \overline{L_n}$. If the $\mathcal{T}$-procedure detects that $\{L_1, \ldots, L_n\}$ entails L in $\mathcal{T}$, where L appears in the input problem, L is added to the trail with $C \vee L$ as justification. The model search, the reasoning about disjunction, and the explanation of conflicts remain propositional and are left to the CDCL procedure.

3 From the Equality Sharing Method to CDSAT

In many problems the theory $\mathcal{T}$ is the union of a few theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$. The *equality sharing* or *Nelson-Oppen* scheme combines procedures for the satisfiability of sets of literals in the member theories to yield a procedure for the $\mathcal{T}$-satisfiability of a set S of $\mathcal{T}$-literals [25] (see [9, Ch. 10] and [5, Sect. 3] for a modern presentation). The theories are assumed to be *disjoint*, meaning that equality is the only shared symbol, *stably infinite*, meaning that every $\mathcal{T}_i$ admits models of infinite cardinality, and endowed with $\mathcal{T}_i$-satisfiability procedures $\mathcal{P}_i$ for $i = 1, \ldots, n$. Since the theories are disjoint, they only need to agree on equalities between *shared terms* and on the cardinalities of shared sorts. The latter point is solved by the stable infiniteness assumption.

Since the input literals mix symbols from the signatures of the theories, each theory $\mathcal{T}_i$ treats as a variable every term whose top symbol belongs to another theory. The equality sharing method features a *separation* phase where new variable symbols are introduced, and the set S of literals is separated into sets $S_1, \ldots, S_n$ sharing only equality and variables.

Example 2. Consider literal $f(2,y) \simeq f(x,y)$, where f is a function in the theory of Equality with Uninterpreted Functions (EUF or UF) [9, Ch. 9], constant symbol 2 comes from a fragment of arithmetic, say LIA for linear integer arithmetic, and x and y are variables. For EUF, term 2 is a variable, because 2 is a foreign symbol for EUF. For LIA, terms $f(2,y)$ and $f(x,y)$ are variables, because f is a foreign symbol for LIA. The set of *shared terms* is $\mathcal{V}_{\mathsf{sh}}(S) = \{f(2,y),\ 2,\ f(x,y)\}$. Separation yields $S_{\mathsf{EUF}} = \{w_1 \simeq f(w_2,y),\ w_3 \simeq f(x,y),\ w_1 \simeq w_3\}$, $S_{\mathsf{LIA}} = \{w_2 \simeq 2,\ w_1 \simeq w_3\}$, and a set of shared terms containing only variables: $\mathcal{V}_{\mathsf{sh}}(S) = \{w_1,\ w_2,\ w_3\}$.

The equality sharing method reduces a $\mathcal{T}$-satisfiability problem to $\mathcal{T}_i$-satisfiability problems. The reduction rests on whether the theories can agree on a *partition* of $\mathcal{V}_{\mathsf{sh}}(S)$, where two terms are equal iff they are in the same equivalence class. A partition is expressed as a set of equalities and disequalities called an *arrangement*: arrangement α contains $u \simeq v$, if u and v are in the same equivalence class, and $u \not\simeq v$ otherwise. Set S is $\mathcal{T}$-satisfiable iff there exists an arrangement α of $\mathcal{V}_{\mathsf{sh}}(S)$ such that $S_i \wedge \alpha$ is $\mathcal{T}_i$-satisfiable for $i = 1, \ldots, n$.

In order to determine whether an arrangement exists, each $\mathcal{P}_i$ deduces from S_i equalities between shared variables. Such equalities are shared by all $\mathcal{P}_i$'s and each $\mathcal{P}_i$ may use them to deduce more equalities. If a procedure $\mathcal{P}_i$ derives a contradiction in this process, no successful arrangement exists, and the $\mathcal{T}$-procedure returns unsat. Otherwise, let $\mathcal{E}$ be the final set of deduced and shared equalities, meaning that no more can be deduced by any of the $\mathcal{P}_i$'s. Set $\mathcal{E}$ represents the arrangement $\alpha_{\mathcal{E}}$ that contains $u \simeq v$, if $(u \simeq v) \in \mathcal{E}$, and $u \not\simeq v$ otherwise. Since no more equalities can be deduced, $S_i \wedge \alpha_{\mathcal{E}}$ is $\mathcal{T}_i$-satisfiable for $i = 1, \ldots, n$, and the $\mathcal{T}$-procedure returns sat.

Having procedure $\mathcal{P}_i$ deduce equalities between shared variables suffices if theory $\mathcal{T}_i$ is *convex*. A theory is *convex* if whenever a conjunctive formula (in our case a set of literals) implies a disjunction of equalities, one of the equalities is also implied. Otherwise, procedure $\mathcal{P}_i$ must be capable of deducing disjunctions of equalities between shared variables. Whenever a disjunction $\bigvee_{j=1}^{m} u_j \simeq v_j$ is deduced, the $\mathcal{T}$-procedure should split it and call itself recursively on each subproblem obtained by adding one of the disjuncts to the current set of shared equalities. In practice, the reasoning about disjunction is entrusted to the CDCL procedure (e.g., [1]). In the extension of the CDCL($\mathcal{T}$) framework to the case where the $\mathcal{T}$-procedure is an equality-sharing combination [1, 19], the $\mathcal{T}$-procedure is allowed to send to CDCL (the propositional abstraction of) any disjunction $\bigvee_{j=1}^{m} u_j \simeq v_j$ that a procedure $\mathcal{P}_i$ may derive. The CDCL procedure treats such a disjunction as a clause to be satisfied. The $\mathcal{T}$-procedure considers the case where $u_j \simeq v_j$ is shared, when CDCL puts (the propositional literal standing for) $u_j \simeq v_j$ on the trail. Therefore, the only new (i.e., non-input) literals that can be $\mathcal{T}$-propagated are propositional abstractions of equalities between shared variables.

The equality sharing method is *not conflict-driven*. Rather, it resembles a *saturation* process, as it expects each theory procedure to be capable of deducing *all* (disjunctions of) equalities between shared variables that follow in the theory from its subproblem. Indeed, also *superposition* can emulate the equality sharing method, under suitable conditions and if superposition is a decision procedure

for each component theory (see [5, Sect. 7] for an overview of superposition-based results and references). In equality sharing the theory procedures are combined as *black-boxes*, which communicate only by broadcasting (disjunctions of) equalities between shared variables. In the integration of equality sharing in the CDCL($\mathcal{T}$) framework, the broadcasting remains hidden within the $\mathcal{T}$-procedure.

CDSAT does not apply separation, so that the shared terms are not necessarily variables. CDSAT lets each theory module place on the trail its decisions and the results of its inferences. If CDSAT emulates equality sharing, the arrangement is built on the shared trail. The theory module inferences are not used only to deduce equalities between shared terms, but also to detect conflicts as shown in the next example. Regardless of whether Bool is a member of the union, every theory has the sort prop of the Boolean values.

Example 3. Consider the set of literals $\{x \leq y,\ y \leq (x + g(x)),\ P(h(x) - h(y)),\ \neg P(0),\ g(x) \simeq 0\}$, involving theories LIA and EUF [5]. It is customary to remove free predicate symbols by introducing new free function symbols and a new free constant symbol with an arbitrary name, say $\bullet$. The resulting set $S = \{x \leq y,\ y \leq (x+g(x)),\ f(h(x)-h(y)) \simeq \bullet,\ f(0) \not\simeq \bullet,\ g(x) \simeq 0\}$ is equisatisfiable.[2] The set of shared terms is $\mathcal{V}_{\mathsf{sh}}(S) = \{x, y, g(x), h(x), h(y), h(x) - h(y), 0\}$.

Suppose a Decide transition puts ${}_?(x \not\simeq y)$ on the trail opening level 1. The LIA-module infers $\{y \leq x + g(x),\ g(x) \simeq 0\} \vdash_{\mathsf{LIA}} y \leq x$ and a Deduce transition posts ${}_{J\vdash}(y \leq x)$ on the trail with $J = \{y \leq x + g(x),\ g(x) \simeq 0\}$. This is a *late propagation* as ${}_{J\vdash}(y \leq x)$ belongs to level 0 and the trail contains level 1. The LIA-module infers by antisymmetry $\{x \leq y,\ {}_{J\vdash}(y \leq x)\} \vdash_{\mathsf{LIA}} x \simeq y$, revealing conflict $E_0 = \{{}_?(x \not\simeq y),\ x \leq y,\ {}_{J\vdash}(y \leq x)\}$. Since ${}_?(x \not\simeq y)$ is the only assignment of highest level in E, a LearnBackjump transition takes us back to level 0 adding to the trail lemma ${}_{H\vdash}(x \simeq y)$ with $H = \{x \leq y,\ {}_{J\vdash}(y \leq x)\}$.

The EUF-module infers by congruence ${}_{H\vdash}(x \simeq y) \vdash_{\mathsf{EUF}} h(x) \simeq h(y)$, and a Deduce transition posts ${}_{I\vdash}(h(x) \simeq h(y))$ on the trail with $I = \{{}_{H\vdash}(x \simeq y)\}$. The LIA-module infers ${}_{I\vdash}(h(x) \simeq h(y)) \vdash_{\mathsf{LIA}} h(x) - h(y) \simeq 0$, and a Deduce transition posts ${}_{K\vdash}(h(x) - h(y) \simeq 0)$ on the trail with $K = \{{}_{I\vdash}(h(x) \simeq h(y))\}$. Then the EUF-module infers $\{f(h(x) - h(y)) \simeq \bullet,\ {}_{K\vdash}(h(x) - h(y) \simeq 0)\} \vdash_{\mathsf{EUF}} f(0) \simeq \bullet$, which is the complement of $f(0) \not\simeq \bullet$ on the trail. Since conflict $E_1 = \{f(h(x) - h(y)) \simeq \bullet,\ {}_{K\vdash}(h(x) - h(y) \simeq 0),\ f(0) \not\simeq \bullet\}$ is on level 0 (there is nowhere to backjump to), a Fail transition returns unsat.

If the procedure $\mathcal{P}_i$ for a component theory $\mathcal{T}_i$ builds a candidate model $\mathcal{M}_i$, the equality sharing method can be implemented in CDCL($\mathcal{T}$) by *model-based theory combination* (MBTC) [12]. This variant of equality sharing allows procedure $\mathcal{P}_i$ to share equalities that are true in $\mathcal{M}_i$, even if they are not $\mathcal{T}_i$-entailed by S_i and the already shared equalities. Such an equality is placed on the CDCL($\mathcal{T}$) trail as a decision. If it turns out to cause a conflict, it gets retracted by the CDCL backjumping. CDSAT generalizes the concept of MBTC, because in CDSAT every theory $\mathcal{T}_i$ can place a decision on the shared trail by a Decide transition. In CDCL($\mathcal{T}$) with MBTC a $\mathcal{T}_i$-model $\mathcal{M}_i$ remains private

[2] Assuming trivial models of cardinality 1 are excluded.

to procedure $\mathcal{P}_i$. In CDSAT the $\mathcal{T}_i$-module share $\mathcal{M}_i$ on the trail, as all theory modules share assignments publicly on the trail.

Another variant of the equality sharing method is *polite theory combination*, where a non-stably-infinite theory can be combined with a polite theory. A polite theory satisfies stronger requirements about the cardinalities of models (e.g., [5, Sect. 4-6]). CDSAT has a different approach to the issue of agreement on the cardinalities of shared sorts. Theories are not required to be stably infinite. It suffices that there exists a *leading theory* that has all the sorts in the union of theories, and all the information about their cardinalities (e.g., at-most-m constraints) [8, Sect. 4.5]. This is because in the proof of completeness of CDSAT an agreement among all the $\mathcal{T}_i$'s is reached by having each $\mathcal{T}_i$ agree with the leading theory. It is not even necessary that the leading theory is one of the $\mathcal{T}_i$'s. If all the theories in the union are stably infinite, the leading theory is a fictional theory $\mathcal{T}_{\mathbb{N}}$, whose models interpret all sorts except prop as having the cardinality of the set $\mathbb{N}$ of the natural numbers [6, Sect. 8].

4 From GCDCL and MCSAT to CDSAT

The success of CDCL for SAT led to several conflict-driven reasoning procedures (e.g., for linear rational arithmetic [24], linear integer arithmetic [16], non-linear arithmetic [17]). In these procedures, the candidate model is a $\mathcal{T}$-model (for the respective fragment $\mathcal{T}$), assignments include *first-order assignments* (e.g., $x{\leftarrow}2$), propagation consists of inexpensive theory deductions, such as term evaluation (e.g., if the trail contains assignment $x{\leftarrow}2$ for a free variable x, then the term $x{+}1$ evaluates to 3), and the inference rules for conflict explanation are relatively more expensive theory inferences.

Let $\mathcal{T}$ be linear rational arithmetic (LRA). The input is a set S of LRA-clauses, whose satisfiability is to be determined. An LRA-clause is a disjunction of literals of the form $t_1 \lessdot t_2$, where t_1 and t_2 are LRA-terms, and $\lessdot \in \{<, \leq\}$. An LRA-term is either a rational constant c or a sum $c_1 \cdot x_1 + \ldots + c_n \cdot x_n$, where c_i is a rational constant and x_i is a rational variable for $i = 1, \ldots n$. Since the ordering on the rational numbers is total, $\overline{(t_1 < t_2)}$ and $\overline{(t_1 \leq t_2)}$ can be replaced by $t_2 \leq t_1$ and $t_2 < t_1$, respectively. An equality $t_1 \simeq t_2$ can be rewritten as $t_1 \leq t_2$ and $t_2 \leq t_1$. If variable x appears in a literal with positive (negative) coefficient, the literal can be rearranged into an upper bound $x \lessdot t$ (lower bound $t \lessdot x$), where x does not occur in t.

Consider for example the *Generalized DPLL* (GDPLL) procedure [24], here renamed GCDCL for consistency. This procedure explains conflicts by the *shadow rule*, which is a generalization of *Fourier-Motzkin (FM) resolution* [20, 18] from literals to clauses. The shadow rule collapses to FM-resolution if both premises are unit clauses. Given premises $t_1 \lessdot_1 x$ and $x \lessdot_2 t_2$, FM-resolution generates $t_1 \lessdot_3 t_2$, where $\lessdot_1, \lessdot_2, \lessdot_3 \in \{<, \leq\}$, and $\lessdot_3$ is $<$ if either $\lessdot_1$ or $\lessdot_2$ is $<$ and $\leq$ otherwise. For example, given clauses $(b < d) \vee (c < d)$ and $d < a$, the shadow rule generates clause $(b < a) \vee (c < a)$. If FM-resolution is applied systematically to eliminate one variable at a time as in the original Fourier-Motzkin

algorithm [20], termination is guaranteed, but the algorithm is very inefficient (e.g., [18] and [8, Sect. 4.4]). The GCDCL procedure applies the shadow rule only to explain conflicts, and it ensures termination by additional restrictions. A fixed total ordering $\prec_{\mathsf{LRA}}$ on rational variables is assumed, and the shadow rule is applied only if the variable resolved upon is $\prec_{\mathsf{LRA}}$-maximal in both premises.

Abstract requirements for a conflict-driven procedure for the $\mathcal{T}$-satisfiability of $\mathcal{T}$-clauses for an arbitrary theory $\mathcal{T}$ were sketched in [24]: (1) embed reasoning about disjunction into theory reasoning, by generalizing to clauses a theory reasoning rule for literals; (2) apply the generalized rule only to explain conflicts; (3) devise additional restrictions to ensure termination.

The problem of how to generalize CDCL from propositional clauses to $\mathcal{T}$-clauses was reformulated as the problem of how to integrate in CDCL a conflict-driven $\mathcal{T}$-procedure for sets of $\mathcal{T}$-literals [13]. In this reformulation the reasoning about disjunction remains entrusted to CDCL, so that inference rules for $\mathcal{T}$-clauses are not necessary, and a $\mathcal{T}$-procedure for sets of $\mathcal{T}$-literals suffices. The reformulated problem was solved by the MCSAT method, where MCSAT stands for *Model Constructing SATisfiability* [13, 15]. Unlike in CDCL($\mathcal{T}$), in MCSAT the $\mathcal{T}$-procedure also is conflict-driven, has access to the trail, proposes assignments to first-order terms, computes propagations, and explains $\mathcal{T}$-conflicts by $\mathcal{T}$-inferences that can generate *new* (i.e., non-input) literals. For termination, it suffices to ensure that the new literals come from a *finite basis* [13]. As observed in [24], in CDCL($\mathcal{T}$) the $\mathcal{T}$-procedure only derives clauses that are $\mathcal{T}$-valid. A conflict-driven $\mathcal{T}$-procedure (and MCSAT) can derive clauses that are logical consequences of the Boolean assignment on the trail (including the input), but are false under the first-order assignment on the trail and hence exclude it.

The discussion in [24] posed the open problem of *how to generalize conflict-driven reasoning to a generic combination of theories*. MCSAT is not a combination calculus, and the open problem was solved by CDSAT, generalizing MCSAT. In CDSAT, multiple theory modules have access to the trail, propose assignments, compute propagations, and explain conflicts by inferences that can generate *new* (i.e., non-input) literals. The modules have *finite local bases*, from which a *finite global basis* for termination is constructed [6, 8].

For CDSAT a set of $\mathcal{T}$-clauses is a problem in a union of theories, namely $\mathcal{T}$ and Bool. In the next example, the LRA-module in CDSAT applies FM-resolution to explain LRA-conflicts. The LRA-module restricts FM-resolution by the restriction proposed in [24] for the shadow rule: assume a *fixed total ordering* $\prec_{\mathsf{LRA}}$ *on rational variables*, and allow FM-resolution only if the *variable resolved upon is* $\prec_{\mathsf{LRA}}$*-maximal in both premises*. This restriction is used to obtain a *finite basis* for the LRA-module [8, Sect. 4.4]. The next example portrays also acceptability of first-order assignments. A first-order decision $_?(t{\leftarrow}\mathfrak{c})$ is *acceptable* if the trail does not contain an assignment to term t, and $t{\leftarrow}\mathfrak{c}$ does not trigger an inference $J \cup \{t{\leftarrow}\mathfrak{c}\} \vdash \overline{L}$ for L a literal on the trail and J a subset of the trail. This condition excludes a first-order assignment that triggers an immediate conflict from which nothing can be learned.

Example 4. Consider the clause set $S = \{x < y,\ x < z,\ (y < w) \vee (z < w),\ w < x\}$, where x, y, z and w are rational variables [24]. Let $x \prec_{\mathsf{LRA}} y \prec_{\mathsf{LRA}} z \prec_{\mathsf{LRA}} w$ be the ordering on variables. Suppose that the CDSAT derivation starts with a series of three decisions by the LRA-module. A first Decide transition adds ${}_{?}(x \leftarrow 0)$ to the trail (level 1). A second Decide transition adds ${}_{?}(y \leftarrow 1)$ on level 2. Note that ${}_{?}(y \leftarrow 0)$ is *not acceptable*, as it would enable the LRA-evaluation inference $\{x \leftarrow 0,\ y \leftarrow 0\} \vdash_{\mathsf{LRA}} \overline{(x < y)}$ when $x < y$ is on the trail. A third Decide transition adds ${}_{?}(z \leftarrow 1)$ to the trail (level 3), whereas ${}_{?}(z \leftarrow 0)$ is not acceptable with $x < z$ and ${}_{?}(x \leftarrow 0)$ on the trail. At this point there is no acceptable value for w, because any rational value would create an immediate conflict with either $\{x \leftarrow 0,\ w < x\}$ or $\{y \leftarrow 1,\ z \leftarrow 1,\ (y < w) \vee (z < w)\}$.

The system needs to perform inferences to explain this LRA-conflict. Since clause $(y < w) \vee (z < w)$ is involved, a case analysis is necessary. Suppose that the Bool-module places ${}_{?}(y < w)$ on the trail (level 4) with another Decide transition. At this point the LRA-module can explain the conflict by FM-resolution. The FM-resolution inference $\{{}_{?}(y < w),\ {}_{\emptyset\vdash}(w < x)\} \vdash_{\mathsf{LRA}} y < x$ is allowed as w is the $\prec_{\mathsf{LRA}}$-maximal variable in both $y < w$ and $w < x$. A Deduce transition adds ${}_{J\vdash}(y < x)$ to level 4 of the trail with $J = \{{}_{?}(y < w),\ {}_{\emptyset\vdash}(w < x)\}$. The FM-resolution inference $\{{}_{\emptyset\vdash}(x < y),\ {}_{J\vdash}(y < x)\} \vdash_{\mathsf{LRA}} x < x$ is also allowed as y is the $\prec_{\mathsf{LRA}}$-maximal variable in both $x < y$ and $y < x$. A Deduce transition adds ${}_{I\vdash}(x < x)$ to level 4 of the trail with $I = \{{}_{\emptyset\vdash}(x < y),\ {}_{J\vdash}(y < x)\}$. Now $E_0 = \{{}_{I\vdash}(x < x)\}$ is an LRA-conflict. Two Resolve transitions unfold E_0 into $E_1 = \{{}_{\emptyset\vdash}(x < y),\ {}_{J\vdash}(y < x)\}$ and E_1 into $E_2 = \{{}_{\emptyset\vdash}(x < y),\ {}_{?}(y < w),\ {}_{\emptyset\vdash}(w < x)\}$. At this point a LearnBackjump transition allows the system to learn $\overline{y < w}$ as ${}_{H\vdash}(\overline{y < w})$, with $H = \{{}_{\emptyset\vdash}(x < y),\ {}_{\emptyset\vdash}(w < x)\}$, and jump back to level 0, which is the smallest level where $\overline{y < w}$ is undefined.

The unit propagation $\{{}_{H\vdash}(\overline{y < w}),\ {}_{\emptyset\vdash}((y < w) \vee (z < w))\} \vdash_{\mathsf{Bool}} z < w$ supports a Deduce transition that adds ${}_{G\vdash}(z < w)$ to level 0 with justification $G = \{{}_{H\vdash}(\overline{y < w}),\ {}_{\emptyset\vdash}((y < w) \vee (z < w))\}$. The shadow rule to work with clause $(y < w) \vee (z < w)$ is unnecessary: it suffices to let the Bool-module break apart the non-unit clause by decision and unit propagation, so that the LRA-module can reason about literals with FM-resolution. The FM-resolution step $\{{}_{G\vdash}(z < w),\ {}_{\emptyset\vdash}(w < x)\} \vdash_{\mathsf{LRA}} z < x$ is allowed as w is the $\prec_{\mathsf{LRA}}$-maximal variable in both $z < w$ and $w < x$. A Deduce transition adds ${}_{K\vdash}(z < x)$ to the trail (at level 0) with $K = \{{}_{G\vdash}(z < w),\ {}_{\emptyset\vdash}(w < x)\}$. The FM-resolution step $\{{}_{\emptyset\vdash}(x < z),\ {}_{K\vdash}(z < x)\} \vdash_{\mathsf{LRA}} x < x$ is allowed as z is the $\prec_{\mathsf{LRA}}$-maximal variable in both $x < z$ and $z < x$. The Deduce transition adding ${}_{M\vdash}(x < x)$ (at level 0), with $M = \{{}_{\emptyset\vdash}(x < z),\ {}_{K\vdash}(z < x)\}$, exposes LRA-conflict $E_3 = \{{}_{M\vdash}(x < x)\}$. Since the level of E_3 is 0, a Fail transition returns unsat.

The Deduce transition in CDSAT covers both propagation and conflict explanation. This allows CDSAT to apply theory inferences (e.g., congruence in Example 3 and FM-resolution in Example 4) more liberally than MCSAT. This flexibility spares the procedure from having to make decisions and create artificial conflicts only to enable inferences needed to discover unsatisfiability.

5 More on CDSAT

In a first-order assignment $t \leftarrow \mathfrak{c}$ (e.g., $x \leftarrow 1/2$, $f(y) \leftarrow 1$), the left side is a first-order (i.e., non-Boolean) term, and the right side is a *value* of the same sort. Values are constant symbols that a *conservative theory extension* $\mathcal{T}_i^+$ adds to the signature of theory $\mathcal{T}_i$ in order to name the elements of the domains of a model of $\mathcal{T}_i$ (one domain per sort). For example, if a theory has the sort of the integers, the signature of the theory can be extended with infinitely many constant symbols to name the integers. Since all the theories in a union are assumed to have the sort prop, the Boolean values true and false are added to every theory's signature. An extension that adds only true and false is called *trivial*. CDSAT keeps terms and values separate, because values are not terms. A term appears only on the left of an assignment, whereas a value appears only on the right. An assignment $t \leftarrow \mathfrak{c}$ cannot be replaced by the equality $t \simeq \mathfrak{c}$, because such an equality cannot be written, as $\mathfrak{c}$ is not a constant symbol in the original signature.

The notion of *theory view* defines what each theory $\mathcal{T}_i$ sees of the shared trail. Suppose that the theory of arrays Arr and LIA share the sort of the integers (e.g., Arr interprets indices as non-negative integers). If the LIA-module decides $A_1 = {}_?(x \leftarrow 3)$ and the Arr-module decides $A_2 = {}_?(y \leftarrow 3)$, the 3 in A_1 is a LIA-value and the 3 in A_2 is a Arr-value. If the Arr-module also decides $A_3 = {}_?(z \leftarrow 4)$, the LIA-*view* of the trail contains $\{A_1,\ x \simeq y,\ x \not\simeq z,\ y \not\simeq z\}$, and the Arr-*view* of the trail contains $\{A_2,\ A_3,\ x \simeq y,\ x \not\simeq z,\ y \not\simeq z\}$. In general, the $\mathcal{T}_i$-view contains the assignments of $\mathcal{T}_i$-values and all equalities and disequalities that can be gleaned from first-order assignments of a $\mathcal{T}_i$-sort, even if made by another theory. A Boolean assignment belongs to every theory view.

In addition to Decide, Deduce, Resolve, LearnBackjump, and Fail, the CDSAT transition system comprises the rules UndoClear and UndoDecide, which solve conflicts due to first-order assignments. Rule UndoClear intervenes when the assignment of highest level in the conflict is a first-order decision. Rule LearnBackjump cannot apply, because a first-order assignment does not have a complement that we can learn. Rule UndoClear applies instead, as shown in the next example, which also portrays the independent concept of *forced decision*.

Example 5. Let the input contains $\{2x + y \simeq 1,\ 2x + 2y \simeq 1\}$. Suppose that the LRA-module ventures decision ${}_?(x \leftarrow 0)$, which is placed on level 1 by a Decide transition. The inference $\{2x + y \simeq 1,\ 2x + 2y \simeq 1\} \vdash_{\mathsf{LRA}} y \simeq 0$ supports a Deduce transition that adds ${}_{J\vdash}(y \simeq 0)$ to the trail with $J = \{2x + y \simeq 1,\ 2x + 2y \simeq 1\}$. The latter justified assignment belongs to level 0, and hence it is a late propagation. The LRA-inference $\{{}_?(x \leftarrow 0),\ {}_{J\vdash}(y \simeq 0)\} \vdash_{\mathsf{LRA}} 2x + y \not\simeq 1$ allows the LRA-module to detect the conflict $E = \{{}_?(x \leftarrow 0),\ {}_{J\vdash}(y \simeq 0),\ 2x + y \simeq 1\}$. Assignment ${}_?(x \leftarrow 0)$ has the highest level in the conflict. An UndoClear transition undoes ${}_?(x \leftarrow 0)$ and clears level 1, so that the trail only contains level 0. At this point ${}_?(x \leftarrow 1/2)$ is a *forced decision*, because $1/2$ is the only acceptable value for x.

UndoClear incorporates backtracking from the level of the bad decision to the previous level. The state has changed due to a *late propagation*. It is certain that

UndoClear fires after a late propagation, because the bad decision was acceptable prior to the late propagation and it causes a conflict afterwards.

If a term t of a sort s occurs on the trail (including as a subterm of another term), and theory $\mathcal{T}_i$ has values of sort s, then term t is said to be *relevant* to $\mathcal{T}_i$, because $\mathcal{T}_i$ can assign a value to t. Since all theories have the Boolean values, Boolean terms are relevant to every theory. Suppose that $\mathcal{T}_i$ does not have values of a sort s. If terms t_1 and t_2 of sort s occur on the trail, the equality $t_1 \simeq t_2$ is *relevant* to $\mathcal{T}_i$, even if $t_1 \simeq t_2$ itself does not appear on the trail. Indeed, $\mathcal{T}_i$ can assign a Boolean value to $t_1 \simeq t_2$. For a decision to be acceptable, the term being assigned must be *relevant* to the theory whose module is making the decision.

First-order assignments have impact on the *equality inferences*. In addition to the inferences based on reflexivity, symmetry, and transitivity, the equality inferences include inferences of the form $\{t_1 \leftarrow \mathfrak{c}, t_2 \leftarrow \mathfrak{c}\} \vdash t_1 \simeq t_2$, where t_1 and t_2 are terms of sort s and $\mathfrak{c}$ is a value of sort s, and $\{t_1 \leftarrow \mathfrak{c}_1, t_2 \leftarrow \mathfrak{c}_2\} \vdash t_1 \not\simeq t_2$, where $\mathfrak{c}_1$ and $\mathfrak{c}_2$ are distinct values of sort s. While the EUF-module (see Example 3) is responsible for inferences based on instances of the congruence axiom scheme for equality, all theory modules can perform equality inferences. When CDSAT emulates equality sharing, equalities derived from first-order assignments can contribute to build an arrangement.

The next example covers relevance, equality inferences, and the UndoDecide transition rule. The latter applies when the assignment of highest level in the conflict is a Boolean justified assignment L whose justification contains a first-order decision from the same level. UndoDecide undoes the first-order decision, backtracks, and puts $\overline{L}$ on the trail. A first-order assignment does not have a complement, but its Boolean consequence does. In this case Resolve is forbidden, because unfolding the conflict and undoing the first-order decision would lead us back to a previous state without learning anything.

Example 6. Assume that the input set S contains the following clauses:
C_1: $(i \not\simeq j) \vee (\mathsf{select}(\mathsf{store}(a, i, v), j) < \mathsf{select}(a, j))$,
C_2: $(\mathsf{select}(a, j) - \mathsf{select}(a, k)) \simeq 0$, and
C_3: $(\mathsf{select}(\mathsf{store}(a, i, v), j) \not< \mathsf{select}(a, j)) \vee (\mathsf{select}(a, j) + \mathsf{select}(a, k) \simeq v)$.
The theories are Bool, LRA for 0, subtraction, and the ordering, and the theory Arr of arrays, for the function symbols select and store. The select function takes an array and an index and returns the value of the array at that index. Thus, $\mathsf{select}(a, j)$ is the term that corresponds to the $a[j]$ expression in programming languages. The store function takes an array, an index, and a value, and returns the array resulting from modifying the given array by writing the given value at the given index. Suppose that array indices are interpreted as integers, which means theory Arr has the sort of the integers. Suppose also that Arr has integer values. Say that the CDSAT derivation starts with two Decide transitions: the first one places ${}_?(i \leftarrow 0)$ on level 1; the second one places ${}_?(j \leftarrow 0)$ on level 2. Both are acceptable, because i and j are relevant to Arr.

A Deduce transition supported by equality inference $\{i \leftarrow 0, j \leftarrow 0\} \vdash_{\mathsf{Arr}} i \simeq j$ adds to level 2 assignment $A_1 \colon {}_J\vdash (i \simeq j)$ with $J = \{{}_?(i \leftarrow 0), {}_?(j \leftarrow 0)\}$. Now unit propagation infers $\{A_1,\ C_1\} \vdash_{\mathsf{Bool}} \mathsf{select}(\mathsf{store}(a, i, v), j) < \mathsf{select}(a, j)$. A Deduce

transition puts assignment $A_2{:}\,_{I\vdash}(\mathsf{select}(\mathsf{store}(a,i,v),j) < \mathsf{select}(a,j))$ on level 2 with $I = \{A_1,\ C_1\}$. Next, unit propagation infers $\{A_2,\ C_3\} \vdash_{\mathsf{Bool}} \mathsf{select}(a,j) + \mathsf{select}(a,k) \simeq v$, so that assignment $A_3{:}\,_{H\vdash}(\mathsf{select}(a,j) + \mathsf{select}(a,k) \simeq v)$ with $H = \{A_2,\ C_3\}$ is added to level 2 by a Deduce transition.

An axiom of the theory of arrays says that $i \simeq j \to \mathsf{select}(\mathsf{store}(a,i,v),j) \simeq v$ for all indices i and j, arrays a, and values v. Thus, the Arr-module can infer $\{A_1,\ A_2\} \vdash_{\mathsf{Arr}} v < \mathsf{select}(a,j)$. A Deduce transition adds $A_4{:}\,_{G\vdash}(v < \mathsf{select}(a,j))$ to level 2 with $G = \{A_1,\ A_2\}$. Suppose that the LRA-module infers $\{A_3,\ C_2\} \vdash_{\mathsf{LRA}} \mathsf{select}(a,j) \simeq v/2$. A Deduce transition adds $A_5{:}\,_{M\vdash}(\mathsf{select}(a,j) \simeq v/2)$ to level 2 with $M = \{A_3,\ C_2\}$. At this point, there is an LRA-conflict $E_0 = \{A_4,\ A_5\}$. A series of four Resolve transition unfolds the conflict as follows:
$E_1 = \{A_4,\ A_3,\ C_2\}$, $E_2 = \{A_1,\ A_2,\ A_3,\ C_2\}$,
$E_3 = \{A_1,\ A_2,\ C_3,\ C_2\}$, $E_4 = \{A_1,\ C_1,\ C_3,\ C_2\}$.
The justified assignment of highest level in E_4 is A_1, which has level 2, so that 2 is also the level of conflict E_4. The justification of A_1 is J, which contains the first-order decision $_?(j{\leftarrow}0)$, whose level is also 2. Replacing A_1 with $J = \{_?(i{\leftarrow}0),\ _?(j{\leftarrow}0)\}$ and then undoing $_?(j{\leftarrow}0)$ would not help, because the system would not learn that i and j cannot be equal. The UndoDecide transition rule fires in this kind of situation: it undoes $_?(j{\leftarrow}0)$ clearing level 2 and going back to level 1, and then it opens a fresh level 2 with the decision $_?(i \not\simeq j)$. Now clause C_1 is satisfied. Three decisions by the LRA-module satisfy also clauses C_2 and C_3: for example, $_?(\mathsf{select}(a,j){\leftarrow}1)$ (level 3), $_?(\mathsf{select}(a,k){\leftarrow}1)$ (level 4), and $_?(v{\leftarrow}2)$ (level 5). Note that $_?(\mathsf{select}(a,k){\leftarrow}1)$ is a *forced decision*, because no other value is acceptable for $\mathsf{select}(a,k)$ given clause C_2 and $_?(\mathsf{select}(a,j){\leftarrow}1)$.

Suppose that theory Arr does not have values for array indices. Then terms i and j are not relevant to Arr, and the derivation cannot start with decisions that are first-order assignments to i and j. However, since i and j occur on the trail, the equality $i \simeq j$ is relevant to Arr, and the derivation can start with a Boolean decision $_?(i \simeq j)$ on level 1. The same inferences as before lead to a conflict $\{_?(i \simeq j),\ C_1,\ C_3,\ C_2\}$. A LearnBackjump transition solves this conflict jumping back to level 0 and learning $_{N\vdash}(i \not\simeq j)$ with $N = \{C_1,\ C_3,\ C_2\}$. The satisfiability of the clauses can be detected as before.

6 Current and Future Work on CDSAT

CDSAT as in [6, 8] is for a union of disjoint theories. We extended CDSAT to unions of theories that share predicate symbols other than equality [7]. We applied this extension to the theory of *arrays with abstract domain*.[3] Arrays are interpreted as updatable functions. Arrays with abstract domain are interpreted as updatable partial functions, that are defined only on those array indices that are *admissible*. The domain is abstract, because the array indices are not necessarily integers. We are currently working on theories of *maps and vectors with abstract domain*. Since admissibility is defined by a shared predicate symbol, these theories require the extension of CDSAT to the predicate-sharing case.

[3] Named theory of *arrays with abstract length* in [7].

A main direction for future work is the implementation of CDSAT. In turn this requires working on the design of the search plans and the architecture for a CDSAT-based solver. The purpose of a search plan is to order the operations of the theory modules. CDSAT leaves much flexibility in terms of both search plan and architecture. The CDCL($\mathcal{T}$) paradigm abstracted existing architectures with the SAT solver at the center, and the theory solver as a satellite, also in the case where $\mathcal{T}$ is a union of theories. CDSAT neither requires nor favors an architecture with the SAT solver at the center, because it treats all theories as peers. The concept of an architecture without SAT solver at the center was very preliminarly explored in the context of MCSAT [2]. A prototype implementation of CDSAT in the Rust programming language is ongoing [14].

References

1. Barrett, C.W., Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Splitting on demand in SAT modulo theories. In: Hermann, M., Voronkov, A. (eds.) Proc. LPAR-13. LNAI, vol. 4246, pp. 512–526. Springer, Heidelberg (2006). https://doi.org/10.1007/11916277_35
2. Bobot, F., Graham-Lengrand, S., Marre, B., Bury, G.: Centralizing equality reasoning in MCSAT. In: D'Silva, V., Dimitrova, R. (eds.) Proc. SMT-16 (2018)
3. Bonacina, M.P.: Automated reasoning for explainable artificial intelligence. In: Reger, G., Treytel, D. (eds.) Proc. ARCADE-1. EPiC Series in Computing, vol. 51, pp. 24–28. EasyChair (2017). https://doi.org/10.29007/4b7h
4. Bonacina, M.P.: On conflict-driven reasoning. In: Shankar, N., Dutertre, B. (eds.) Proc. 6th Workshop on Automated Formal Methods (AFM). Kalpa Publications, vol. 5, pp. 31–49. EasyChair (2018). https://doi.org/10.29007/spwm
5. Bonacina, M.P., Fontaine, P., Ringeissen, C., Tinelli, C.: Theory combination: beyond equality sharing. In: Lutz, C., et al. (eds.) Description Logic, Theory Combination, and All That, LNCS, vol. 11560, pp. 57–89. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-22102-7_3
6. Bonacina, M.P., Graham-Lengrand, S., Shankar, N.: Conflict-driven satisfiability for theory combination: transition system and completeness. J. Autom. Reason. **64**(3), 579–609 (2020). https://doi.org/10.1007/s10817-018-09510-y, Conference version at CADE 2017.
7. Bonacina, M.P., Graham-Lengrand, S., Shankar, N.: CDSAT for nondisjoint theories with shared predicates: arrays with abstract length. In: Hyvärinen, A., Déharbe, D. (eds.) Proc. SMT-20. CEUR Proceedings, vol. 3185, pp. 18–37. CEUR WS-org, Aachen (2022), https://ceur-ws.org/Vol-3185/
8. Bonacina, M.P., Graham-Lengrand, S., Shankar, N.: Conflict-driven satisfiability for theory combination: lemmas, modules, and proofs. J. Autom. Reason. **66**(1), 43–91 (2022). https://doi.org/10.1007/s10817-021-09606-y, Conference version at CPP 2018.
9. Bradley, A.R., Manna, Z.: The Calculus of Computation - Decision Procedures with Applications to Verification. Springer, Berlin (2007)
10. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. Commun. ACM **5**(7), 394–397 (1962). https://doi.org/10.1145/368273.368557
11. Davis, M., Putnam, H.: A computing procedure for quantification theory. J. ACM **7**, 201–215 (1960). https://doi.org/10.1145/321033.321034

12. de Moura, L., Bjørner, N.: Model-based theory combination. In: Krstić, S., Oliveras, A. (eds.) Proc. SMT-5 (2007). ENTCS, vol. 198(2), pp. 37–49. Elsevier, Amsterdam (2008). https://doi.org/10.1016/j.entcs.2008.04.079
13. de Moura, L., Jovanović, D.: A model-constructing satisfiability calculus. In: Giacobazzi, R., Berdine, J., Mastroeni, I. (eds.) Proc. VMCAI-14. LNCS, vol. 7737, pp. 1–12. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35873-9_1
14. Denis, X.: A baby CDSAT-based verified solver written in Rust (2025), https://github.com/xldenis/cdsat, Accessed 13 January 2025
15. Jovanović, D., Barrett, C., de Moura, L.: The design and implementation of the model-constructing satisfiability calculus. In: Jobstman, B., Ray, S. (eds.) Proc. FMCAD-13. ACM and IEEE (2013). https://doi.org/10.1109/FMCAD.2013.7027033
16. Jovanović, D., de Moura, L.: Cutting to the chase: solving linear integer arithmetic. J. Autom. Reason. **51**, 79–108 (2013). https://doi.org/10.1007/s10817-013-9281-x, Conference version at CADE 2011.
17. Jovanović, D., de Moura, L.: Solving non-linear arithmetic. In: Gramlich, B., Miller, D., Sattler, U. (eds.) Proc. IJCAR-6. LNAI, vol. 7364, pp. 339–354. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31365-3_27
18. Kroening, D., Strichman, O.: Decision Procedures - An Algorithmic Point of View. Texts in Theoretical Computer Science, Springer, Berlin (2008)
19. Krstić, S., Goel, A.: Architecting solvers for SAT modulo theories: Nelson-Oppen with DPLL. In: Konev, B., Wolter, F. (eds.) Proc. FroCoS-6. LNAI, vol. 4720, pp. 1–27. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74621-8_1
20. Lassez, J.L., Maher, M.J.: On Fourier's algorithm for linear arithmetic constraints. J. Autom. Reason. **9**, 373–379 (1992). https://doi.org/10.1007/BF00245296
21. Loveland, D., Sabharwal, A., Selman, B.: DPLL: The core of modern satisfiability solvers. In: Omodeo, E.G., Policriti, A. (eds.) Martin Davis on Computability, Computational Logic, and Mathematical Foundations, OCL, vol. 10, pp. 315–335. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41842-1_12
22. Marques Silva, J.P., Lynce, I., Malik, S.: Conflict-driven clause learning SAT solvers. In: Biere, A., Heule, M., Van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability (2nd Edition), Frontiers in Artificial Intelligence and Applications, vol. 336, pp. 133–182. IOS Press, Amsterdam (2021). https://doi.org/10.3233/FAIA200987
23. Marques Silva, J.P., Sakallah, K.A.: GRASP: A search algorithm for propositional satisfiability. IEEE Trans. Comput. **48**(5), 506–521 (1999). https://doi.org/10.1109/12.769433
24. McMillan, K.L., Kuehlmann, A., Sagiv, M.: Generalizing DPLL to richer logics. In: Bouajjani, A., Maler, O. (eds.) Proc. CAV-21. LNCS, vol. 5643, pp. 462–476. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02658-4_35
25. Nelson, G., Oppen, D.C.: Simplification by cooperating decision procedures. ACM Trans. Prog. Lang. Syst. **1**(2), 245–257 (1979). https://doi.org/10.1145/357073.357079
26. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: from an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T). J. ACM **53**(6), 937–977 (2006). https://doi.org/10.1145/1217856.1217859