Satisfiability Modulo Theories and Assignments

Maria Paola Bonacina[1], Stéphane Graham-Lengrand[2,3], and Natarajan Shankar[2]

[1] Università degli Studi di Verona, Verona, Italy
[2] SRI International, Menlo Park, USA
[3] CNRS - INRIA - École Polytechnique, Palaiseau, France

Abstract. The CDCL procedure for SAT is the archetype of *conflict-driven* procedures for satisfiability of quantifier-free problems in a single theory. In this paper we lift CDCL to CDSAT (*Conflict-Driven Satisfiability*), a system for conflict-driven reasoning in combinations of disjoint theories. CDSAT combines *theory modules* that interact through a global *trail* representing a candidate model by Boolean and first-order assignments. CDSAT generalizes to generic theory combinations the *model-constructing satisfiability calculus* (MCSAT) introduced by de Moura and Jovanović. Furthermore, CDSAT generalizes the *equality sharing* (Nelson-Oppen) approach to theory combination, by allowing theories to share equality information both explicitly through equalities and disequalities, and implicitly through assignments. We identify sufficient conditions for the *soundness*, *completeness*, and *termination* of CDSAT.

Keywords: Theory combination· Conflict-driven decision procedures · Model building· Satisfiability modulo assignment

1 Introduction

A growing trend in automated deduction is the generalization of *conflict-driven reasoning* from propositional to first-order logic (cf. [2] for a brief coeval survey). For propositional satisfiability (SAT), the conflict-driven clause learning (CDCL) procedure works by guessing assignments to variables, propagating their consequences through clauses, and learning new clauses, or lemmas, when assignments lead to conflicts [14]. The conflict-driven paradigm has been extended to decide the $\mathcal{T}$-satisfiability of sets of literals when $\mathcal{T}$ is one of several fragments of arithmetic [4,7,10,11,12,15,17,18]. Key features of such *conflict-driven $\mathcal{T}$-satisfiability procedures* are the use of assignments to first-order variables and the explanation of conflicts with lemmas, which may contain atoms that are not in the input. We illustrate these features by an example. Consider the following set of literals, which is unsatisfiable in Linear Rational Arithmetic (LRA):

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

A conflict-driven LRA-satisfiability procedure attempts to build a model by guessing a value for one of the variables, say $y{\leftarrow}0$. This lets l_0 yield the lower bound $x > 0$. Given the upper bound l_2, the space of possible values for x is empty, revealing that the guess and the constraints are in conflict. The procedure explains the conflict by the *new atom* $l_3 : (-y < -2)$, the linear combination of l_0 and l_2 that eliminates x. This excludes not only $y \leftarrow 0$, but also all assignments $y \leftarrow \mathfrak{c}$ where $\mathfrak{c} \leq 2$. Suppose the procedure retracts the assignment $y \leftarrow 0$ and tries $y \leftarrow 4$. This lets l_1 yield the upper bound $x < -4$, and l_0 the lower bound $x > -2$. The procedure explains this conflict by the new atom $l_4 : (y < 0)$, the linear combination of l_0 and l_1 that eliminates x. As l_4 is violated by the assignment $y \leftarrow 4$, the procedure retracts $y \leftarrow 4$. Then no assignment to y can satisfy both l_3 and l_4. This third conflict is explained by the linear combination of l_3 and l_4 that eliminates y, namely $0 < -2$, which is also a new atom. Since this consequence of the original problem is a contradiction, the procedure returns unsatisfiable.

Applications typically require deciding the satisfiability of arbitrary quantifier-free formulae, or, equivalently, sets of ground clauses, in a combination $\mathcal{T}$ of theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$. The DPLL($\mathcal{T}$) approach [1, 13] combines a CDCL-based SAT-solver with a $\mathcal{T}$-satisfiability procedure, obtained from $\mathcal{T}_i$-satisfiability procedures by the *equality sharing* method [16], assuming that the theories $\mathcal{T}_i$ are *disjoint* and *stably infinite*: they do not share symbols other than equality and admit countably infinite models. The $\mathcal{T}_i$-satisfiability procedures are combined as black-boxes that only propagate equalities between shared variables. DPLL($\mathcal{T}$) uses their combination as a black-box that only detects $\mathcal{T}$-conflicts and propagates $\mathcal{T}$-lemmas, while the SAT-solver tries assignments and builds a candidate model. If conflict-driven $\mathcal{T}_i$-satisfiability procedures were integrated in this manner, the combination would not be conflict-driven. To make it conflict-driven, the $\mathcal{T}_i$-satisfiability procedures need to cooperate to build a model, sharing assignments and exporting lemmas to explain conflicts, possibly using new atoms.

MCSAT, for *Model-Constructing Satisfiability*, is a paradigm for integrating a CDCL-based SAT-solver with a conflict-driven $\mathcal{T}$-satisfiability procedure [5]. MCSAT uses first-order assignments on a par with Boolean ones, coordinates the conflict explanation mechanisms at the Boolean and theory levels in a unified manner, and incorporates the capability of creating new atoms. MCSAT lifted CDCL to SMT in the sense of *satisfiability modulo a single theory* $\mathcal{T}$, which was instantiated to the theory of bit-vectors [19] and to non-linear integer arithmetic [8]. A version of MCSAT was also given for the specific combination of LRA and Equality with Uninterpreted Function symbols (EUF) [9], but the conflict-driven combination of a generic range of theories remained an open problem.

In this paper we generalize *conflict-driven reasoning* to generic combinations of disjoint theories, solving the problem of combining multiple conflict-driven $\mathcal{T}_i$-satisfiability procedures into a conflict-driven $\mathcal{T}$-satisfiability procedure. We introduce a new method for theory combination, called CDSAT for *Conflict-Driven Satisfiability*. For example, it decides the satisfiability of problems in the combination of LRA, EUF and the theory of arrays, such as

$$P = \{f(\mathsf{select}(\mathsf{store}(a, i, v), j)) \simeq w,\ f(u) \simeq w{-}2,\ i \simeq j,\ u \simeq v\}.$$

CDSAT treats propositional and theory reasoning uniformly: formulae are terms of sort prop (for proposition); propositional logic is one of the theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$;

and CDCL is one of the $\mathcal{T}_i$-satisfiability procedures to be combined. With formulae reduced to terms, *assignments* become the data manipulated by inferences. CDSAT combines $\mathcal{T}_i$-inference systems, called *theory modules* [6], rather than $\mathcal{T}_i$-satisfiability procedures. Ideally, a $\mathcal{T}_i$-satisfiability procedure, like any reasoning procedure, is defined by an inference system and a search plan. Since all conflict-driven procedures have the same conflict-driven search plan, what needs to be combined are the inference systems, while the common conflict-driven control is factored out and handled centrally by CDSAT. We prove that CDSAT is *sound*, *complete*, and *terminating*, identifying the sufficient conditions that theories and theory modules need to fulfill for these properties to hold.

We believe that the abstraction of viewing combination of theories as combination of inference systems, rather than procedures, allows us to bring simplicity and elegance to theory combination. A $\mathcal{T}_i$-satisfiability procedure that is not conflict-driven can still be integrated in CDSAT by treating it as a theory module whose only inference rule invokes the $\mathcal{T}_i$-satisfiability procedure to detect the $\mathcal{T}_i$-unsatisfiability of a set of assignments. Therefore CDSAT subsumes both MCSAT and equality sharing. CDSAT reduces to MCSAT, if there are only propositional logic with CDCL and another theory $\mathcal{T}$ with a conflict-driven $\mathcal{T}$-satisfiability procedure. CDSAT reduces to equality sharing, if none of the theories has a conflict-driven $\mathcal{T}$-satisfiability procedure.

2 Preliminaries

We assume the basic definitions of multi-sorted first-order logic. A *signature* $\Sigma = (S, F)$ consists of a set S of *sorts*, including prop, and a set F of *symbols*, including a collection $\simeq_S$ of equality symbols $\simeq_s : (s \times s) \rightarrow \mathsf{prop}$ for every $s \in S$. Sorts can be omitted when clear from context. The other symbols in F may be constant, function, and predicate symbols, as well as logical connectives such as $\wedge$, $\vee$, and $\neg$. Given a class $\mathcal{V} = (\mathcal{V}^s)_{s \in S}$ of sets of sorted variables, $\Sigma[\mathcal{V}]$*-terms* are defined as usual, and formulae are terms of sort prop. We use l for formulae and t and u for terms of any sort. Formulae in the standard sense are obtained as the closure of our formulae under quantifiers and logical connectives; Σ*-sentences* are those with no free variables.

A $\Sigma[\mathcal{V}]$*-interpretation* $\mathcal{M}$ interprets each sort s in S as a non-empty set $s^{\mathcal{M}}$ with $\mathsf{prop}^{\mathcal{M}} = \{\mathsf{true}, \mathsf{false}\}$; each symbol $f : (s_1 \times \cdots \times s_m) \rightarrow s$ in F as a function $f^{\mathcal{M}} : s_1^{\mathcal{M}} \times \cdots \times s_m^{\mathcal{M}} \rightarrow s^{\mathcal{M}}$ with $\simeq_s^{\mathcal{M}}$ returning true if and only if its arguments are identical; and each variable $v \in \mathcal{V}^s$ as an element $v^{\mathcal{M}} \in s^{\mathcal{M}}$. The interpretation $\mathcal{M}(t)$ of a $\Sigma[\mathcal{V}]$-term t is defined as usual. $\Sigma[\emptyset]$-interpretations, known as Σ*-structures*, suffice for Σ-sentences.

A *theory* $\mathcal{T}$ on signature Σ is defined axiomatically as a set of Σ-sentences, called its *axioms*, or model-theoretically as the class of Σ-structures, called $\mathcal{T}$*-models*, that satisfy the axioms of $\mathcal{T}$. A $\mathcal{T}[\mathcal{V}]$*-model* is any $\Sigma[\mathcal{V}]$-interpretation whose underlying Σ-structure is a $\mathcal{T}$-model.

Let $\mathcal{T}_1, \ldots, \mathcal{T}_n$ be *disjoint* theories with signatures $\Sigma_1 = (S_1, F_1), \ldots, \Sigma_n = (S_n, F_n)$: they do not share symbols other than equality, but can share sorts, meaning that $F_i \cap F_j = (\simeq_{S_i \cap S_j})$ for $i \neq j$. Let $\mathcal{T}_\infty$ be their union, with signature $\Sigma_\infty = (S_\infty, F_\infty)$ for $S_\infty = \bigcup_{k=1}^n S_k$ and $F_\infty = \bigcup_{k=1}^n F_k$, and axiomatization given by the union of those of $\mathcal{T}_1, \ldots, \mathcal{T}_n$. We fix a global collection of variables $\mathcal{V}_\infty = (\mathcal{V}_\infty^s)_{s \in S_\infty}$, use *variables* for variables in $\mathcal{V}_\infty$, and *terms* for $\Sigma_\infty[\mathcal{V}_\infty]$-terms.

Example 1. Problem P from Sect. 1 is written in the signatures

$$\begin{aligned}
\Sigma_{\mathsf{LRA}} &= (\ \{\mathsf{prop}, \mathsf{Q}\},\ \simeq_{\{\mathsf{prop},\mathsf{Q}\}} \cup \{0, 1 : \mathsf{Q},\ + : \mathsf{Q} \times \mathsf{Q} \rightarrow \mathsf{Q}\} \cup \{q \cdot : \mathsf{Q} \rightarrow \mathsf{Q} \mid q \in \mathbb{Q}\}\ ) \\
\Sigma_{\mathsf{EUF}} &= (\ \{\mathsf{prop}, \mathsf{Q}, V\},\ \simeq_{\{\mathsf{prop},\mathsf{Q},V\}} \cup \{f : V \rightarrow \mathsf{Q}\}\ ) \\
\Sigma_{\mathsf{Arr}} &= (\ \{\mathsf{prop}, V, I, A\},\ \simeq_{\{\mathsf{prop},V,I,A\}} \cup \{\mathsf{select} : A \times I \rightarrow V,\ \mathsf{store} : A \times I \times V \rightarrow A\}\ )
\end{aligned}$$

where Q and $\mathbb{Q}$ are the sort and the set of the rationals, $q \cdot$ is scalar multiplication, and A, I, and V are the sorts of arrays, indices, and values.

3 Assignments and Theory Modules

CDSAT solves $\mathcal{T}_\infty$-satisfiability problems presented as *assignments* of values to terms. For example, a set of formulae $\{l_1, \ldots, l_m\}$ is represented as the assignment $\{l_1 \leftarrow \mathsf{true}, \ldots, l_m \leftarrow \mathsf{true}\}$. The input assignment may contain terms of any sort. Assignments are *public*, meaning visible to all $\mathcal{T}_k$-inference systems.

3.1 Assignments

We need to identify the language of values that the system can assign to terms, besides true and false. Assignable values are not necessarily in the theories' signatures (e.g., consider $x \leftarrow \sqrt{2}$ for the sort of the reals). Therefore, we introduce for each theory $\mathcal{T}_k$, $1 \leq k \leq n$, a *conservative extension* $\mathcal{T}_k^+$ with signature $\Sigma_k^+ = (S_k, F_k^+)$, where F_k^+ is the extension of F_k with a possibly empty set of *new* constant symbols called *$\mathcal{T}_k^+$-values*. An extension is *conservative* if any $\mathcal{T}_k^+$-unsatisfiable set of $\Sigma_k[\mathcal{V}]$-formulae is also $\mathcal{T}_k$-unsatisfiable. This ensures that reasoning in the extension does not change the problem: if CDSAT discovers $\mathcal{T}_k^+$-unsatisfiability, the problem is $\mathcal{T}_k$-unsatisfiable; if the problem is $\mathcal{T}_k$-satisfiable, there is a $\mathcal{T}_k^+$-model that CDSAT can build. A sort $s \in S_k$ with at least one $\mathcal{T}_k^+$-value is called *$\mathcal{T}_k$-public*, as a term of sort s may be assigned a $\mathcal{T}_k^+$-value. A sort s may be $\mathcal{T}_i$-public and $\mathcal{T}_j$-public for $i \neq j$. We stipulate that sort prop is $\mathcal{T}_k$-public for all k, with $\mathcal{T}_k^+$-values true and false, which are valid and unsatisfiable, respectively, in $\mathcal{T}_k^+$. We use $\mathfrak{b}$ for true or false. We assume that the extended theories are still disjoint except for the two Boolean values true and false.

Example 2. Let RA be the theory of real arithmetic with sorts $\{\mathsf{R}, \mathsf{prop}\}$ and symbols $\simeq_{\{\mathsf{R},\mathsf{prop}\}} \cup \{0, 1 : \mathsf{R};\ +, -, \cdot : \mathsf{R} \times \mathsf{R} \rightarrow \mathsf{R}\}$. RA^+ adds a new constant for every real number, so R is RA-public. The axioms of RA^+ are the formulae that hold in the standard model of the reals interpreting every RA^+-value as itself.

Extending the signature with names that denote all individuals in the domain of a $\mathcal{T}_k$-model is a standard move in automated reasoning, where models need to be built out of syntax, and especially when $\mathcal{T}_k$ has an "intended model" as in arithmetic. In such cases a $\mathcal{T}_k^+$-value is both the domain element and the constant symbol that names it.

$$\begin{array}{rl}
t_1{\leftarrow}\mathfrak{c}_1, t_2{\leftarrow}\mathfrak{c}_2 & \vdash\ t_1 \simeq_s t_2 \text{ if } \mathfrak{c}_1 \text{ and } \mathfrak{c}_2 \text{ are the same } \mathcal{T}^+\text{-value of sort } s \\
t_1{\leftarrow}\mathfrak{c}_1, t_2{\leftarrow}\mathfrak{c}_2 & \vdash\ t_1 \not\simeq_s t_2 \text{ if } \mathfrak{c}_1 \text{ and } \mathfrak{c}_2 \text{ are distinct } \mathcal{T}^+\text{-values of sort } s \\
 & \vdash\ t_1 \simeq_s t_1 \\
t_1 \simeq_s t_2 & \vdash\ t_2 \simeq_s t_1 \\
t_1 \simeq_s t_2, t_2 \simeq_s t_3 & \vdash\ t_1 \simeq_s t_3
\end{array}$$

Fig. 1. Equality inference rules: t_1, t_2, and t_3 are terms of sort s

Definition 1 (Assignment). *Given a theory $\mathcal{T}$ with extension $\mathcal{T}^+$, a $\mathcal{T}$-*assignment *is a set of pairs $t{\leftarrow}\mathfrak{c}$, where t is a term and $\mathfrak{c}$ is a $\mathcal{T}^+$-value of the same sort. Term t and all its subterms are said to* occur *in the assignment. An assignment is* plausible *if it does not contain both $l{\leftarrow}\mathsf{true}$ and $l{\leftarrow}\mathsf{false}$ for any formula l.*

For example, $\{x{\leftarrow}\sqrt{2},\ x+y{\leftarrow}\sqrt{3}\}$ and $\{f(x){\leftarrow}\sqrt{2},\ (1{\cdot}x \simeq x){\leftarrow}\mathsf{true}\}$ are RA-assignments. If for all pairs $t{\leftarrow}\mathfrak{c}$ the sort of t and $\mathfrak{c}$ is s, the $\mathcal{T}$-assignment is *of sort s*, and if $s = \mathsf{prop}$ it is *Boolean*. A *first-order* $\mathcal{T}$-assignment is a $\mathcal{T}$-assignment that is not Boolean. We use J for generic $\mathcal{T}$-assignments, A for singleton ones, and L for Boolean singletons. We abbreviate $l{\leftarrow}\mathsf{true}$ as l, $l{\leftarrow}\mathsf{false}$ as $\overline{l}$, and $t \simeq_s u\ {\leftarrow}\mathsf{false}$ as $t \not\simeq_s u$. The *flip* $\overline{L}$ of L assigns to the same formula the opposite Boolean value. The union of $\mathcal{T}_1^+, \ldots, \mathcal{T}_n^+$ is an extension $\mathcal{T}_\infty^+$ of $\mathcal{T}_\infty$, with signature $\Sigma_\infty^+ = (S_\infty, F_\infty^+)$ for $F_\infty^+ = \bigcup_{k=1}^n F_k^+$. We use H for $\mathcal{T}_\infty$-assignments, called *assignments* for short. Plausibility does not forbid an assignment $\{t \leftarrow 3.1, u \leftarrow 5.4, t \leftarrow \mathsf{red}, u \leftarrow \mathsf{blue}\}$, where the first two pairs are $\mathcal{T}_1$-assignments and the last two are $\mathcal{T}_2$-assignments; the sort of t and u is both $\mathcal{T}_1$-public and $\mathcal{T}_2$-public. When building a model from this assignment, 3.1 will be identified with red and 5.4 with blue.

Definition 2 (Theory view). *Let $\mathcal{T}$ and S be either $\mathcal{T}_\infty$ and S_∞, or $\mathcal{T}_k$ and S_k for $1 \le k \le n$. The $\mathcal{T}$-*view *of an assignment H is the $\mathcal{T}$-assignment $H_\mathcal{T} =$*

$$\{\ t{\leftarrow}\mathfrak{c}\quad |\ t{\leftarrow}\mathfrak{c} \textit{ is a } \mathcal{T}\textit{-assignment in } H\ \} \cup$$
$$\bigcup_{k=1}^n(\{\ t_1 \simeq_s t_2\ |\ \ t_1{\leftarrow}\mathfrak{c}, t_2{\leftarrow}\mathfrak{c} \textit{ are } \mathcal{T}_k\textit{-assignments in } H \textit{ of sort } s{\in}S\backslash\{\mathsf{prop}\}\ \} \cup$$
$$\{\, t_1 \not\simeq_s t_2\ |\ t_1{\leftarrow}\mathfrak{c}_1, t_2{\leftarrow}\mathfrak{c}_2 \textit{ are } \mathcal{T}_k\textit{-assignments in } H \textit{ of sort } s{\in}S\backslash\{\mathsf{prop}\},\ \mathfrak{c}_1{\neq}\mathfrak{c}_2\,\}\,).$$

$H_\mathcal{T}$ contains the $\mathcal{T}$-assignments of H, plus all equalities and disequalities induced by the $\mathcal{T}_k$-assignments in H, $1 \le k \le n$. We introduce next *theory modules*, the abstract counterpart of theory solvers or theory plugins [9].

3.2 Theory Modules

A *theory module* $\mathcal{I}$ for theory $\mathcal{T}$ is an inference system whose inferences, called *$\mathcal{I}$-inferences* and written $J \vdash_\mathcal{I} L$, derive a singleton Boolean assignment L from a $\mathcal{T}$-assignment J. Since all theories include equality, all theory modules include the *equality inference rules* of Fig. 1. The following inferences are $\mathcal{I}_\mathsf{RA}$-inferences:

$$\begin{array}{l}
(x{\leftarrow}\sqrt{2}), (y{\leftarrow}\sqrt{2}) \vdash_{\mathcal{I}_\mathsf{RA}} (x{\cdot}y \simeq 1{+}1) \\
(y{\leftarrow}\sqrt{2}), (x{\leftarrow}\sqrt{2}) \vdash_{\mathcal{I}_\mathsf{RA}} (y \simeq x) \\
(y{\leftarrow}\sqrt{2}), (x{\leftarrow}\sqrt{3}) \vdash_{\mathcal{I}_\mathsf{RA}} (y \not\simeq x)
\end{array}$$

$\mathcal{I}$-inferences only derive Boolean assignments because CDSAT does not justify first-order assignments by inferences, not even when a first-order assignment is forced by others (e.g., $y{\leftarrow}2$ by $x{\leftarrow}1$ and $(x{+}y){\leftarrow}3$). We assume we have theory modules $\mathcal{I}_1, \dots, \mathcal{I}_n$ for $\mathcal{T}_1, \dots, \mathcal{T}_n$. We now define acceptability and relevance.

Definition 3 (Acceptability). *Given $\mathcal{T}_k$-assignments $t{\leftarrow}\mathfrak{c}$ and J, $t{\leftarrow}\mathfrak{c}$ is* acceptable *for J and $\mathcal{I}_k$, if (i) J does not assign a value to t and (ii) either $t{\leftarrow}\mathfrak{c}$ is Boolean or there are no $\mathcal{I}_k$-inferences $J', (t{\leftarrow}\mathfrak{c}) \vdash_{\mathcal{I}_k} L$ with $J', \overline{L} \subseteq J$.*

When adding $t \leftarrow \mathfrak{c}$ to J, acceptability prevents repetitions (cf. Condition (i)) and contradictions: if $t \leftarrow \mathfrak{c}$ is Boolean, its flip should not be in J, preserving plausibility (cf. Condition (i)); if $t \leftarrow \mathfrak{c}$ is first-order, and therefore has no flip, so that plausibility does not apply, acceptability ensures that none of the consequences one inference step away has its flip in J (cf. Condition (ii)).

Definition 4 (Relevance). *A term is* $\mathcal{T}_k$-relevant *for an assignment H, if either (i) it occurs in H and has a $\mathcal{T}_k$-public sort, or (ii) it is an equality $t_1 \simeq_s t_2$ whose terms t_1 and t_2 occur in H and whose sort $s \in S_k$ is not $\mathcal{T}_k$-public.*

Relevance organizes the division of labor among modules. For instance in the assignment $\{x{\leftarrow}\sqrt{5}, f(x){\leftarrow}\sqrt{2}, f(y){\leftarrow}\sqrt{3}\}$, x and y of sort R are RA-relevant, not EUF-relevant, assuming R is not EUF-public, while $x \simeq_{\mathsf{R}} y$ is EUF-relevant, not RA-relevant. Each theory has a mechanism to fix and communicate equalities between terms of a known sort, such as x and y: EUF does it by assigning a truth-value to $x \simeq_{\mathsf{R}} y$; RA does it by assigning values to x and y.

4 Examples of Theory Modules

In this section we give theory modules for several theories. We may use $\bot$ to stand for the assignment $(x \simeq_{\mathsf{prop}} x){\leftarrow}\mathsf{false}$ for an arbitrary variable x. For brevity, we omit equality symbols from signatures and equality inference from modules.

4.1 A Module for Propositional Logic

Σ_{Bool} has only the sort prop and the symbols $\neg : \mathsf{prop} \to \mathsf{prop}$, and $\vee, \wedge : (\mathsf{prop}{\times}\mathsf{prop}){\to}\mathsf{prop}$. Let Bool^+ be the trivial extension with only $\{\mathsf{true}, \mathsf{false}\}$ as Bool^+-values. $\mathcal{I}_{\mathsf{Bool}}$ features an *evaluation* inference rule that derives the truth value $\mathfrak{b}$ of formula l, given truth values $\mathfrak{b}_1, \dots, \mathfrak{b}_m$ of subformulae $l_1, \dots, l_m$, and then, from left to right, two rules for negation, two rules for conjunction elimination, and two rules for unit propagation:

$$l_1{\leftarrow}\mathfrak{b}_1, \dots, l_m{\leftarrow}\mathfrak{b}_m \vdash_{\mathsf{Bool}} l{\leftarrow}\mathfrak{b}$$

$$\begin{array}{lll} \neg l \vdash_{\mathsf{Bool}} \overline{l} & \overline{l_1 \vee \cdots \vee l_m} \vdash_{\mathsf{Bool}} \overline{l_i} & l_1 \vee \cdots \vee l_m, \{\overline{l_j} \mid j \neq i\} \vdash_{\mathsf{Bool}} l_i \\ \overline{\neg l} \vdash_{\mathsf{Bool}} l & l_1 \wedge \cdots \wedge l_m \vdash_{\mathsf{Bool}} l_i & \overline{l_1 \wedge \cdots \wedge l_m}, \{l_j \mid j \neq i\} \vdash_{\mathsf{Bool}} \overline{l_i} \end{array}$$

where $1 \leq j, i \leq m$, and, for the first rule, l must be in the closure of $l_1, \dots, l_m$ with respect to the Σ_{Bool}-connectives. Although the evaluation rule alone is sufficient for completeness (cf. Sect. 6.3), the other six rules, including in particular unit propagation as in CDCL, are obviously desirable.

4.2 A Theory Module for LRA

Let Σ_{LRA} be as in Example 1 and LRA^+ be the extension that adds a constant $\tilde{q}$ and the axiom $\tilde{q} \simeq_{\mathsf{Q}} q \cdot 1$ for each rational number $q \in \mathbb{Q}$. Here too, the first rule of $\mathcal{I}_{\mathsf{LRA}}$ is an *evaluation* inference rule that derives the value $\mathfrak{b}$ of formula l, given values $\tilde{q_1}, \dots, \tilde{q_m}$ of subterms $t_1, \dots, t_m$ of sort Q:

Evaluation	$t_1 \leftarrow \tilde{q_1}, \dots, t_m \leftarrow \tilde{q_m} \vdash_{\mathsf{LRA}} l \leftarrow \mathfrak{b}$
Positivization	$\overline{t_1 < t_2} \vdash_{\mathsf{LRA}} t_2 \le t_1$
	$\overline{t_1 \le t_2} \vdash_{\mathsf{LRA}} t_2 < t_1$
Equality elimination	$t_1 \simeq_{\mathsf{Q}} t_2 \vdash_{\mathsf{LRA}} t_i \le t_j \quad \text{with } \{i, j\} = \{1, 2\}$
Disequality elimination	$(t_1 \le x), (x \le t_2), (t_1 \simeq_{\mathsf{Q}} t_0), (t_2 \simeq_{\mathsf{Q}} t_0), (x \not\simeq_{\mathsf{Q}} t_0) \vdash_{\mathsf{LRA}} \bot$
Fourier-Motzkin resolution	$(t_1 \lessdot_1 x), (x \lessdot_2 t_2) \vdash_{\mathsf{LRA}} (t_1 \lessdot_3 t_2)$

where t_0, t_1, t_2, and x are terms of sort Q; x is a Σ_{LRA}*-variable* that is not free in t_0, t_1, t_2 (cf. Sect. 6); $\lessdot_1, \lessdot_2, \lessdot_3 \in \{<, \le\}$ and $\lessdot_3$ is $<$ if and only if either $\lessdot_1$ or $\lessdot_2$ is $<$. For the first rule, l must be a formula whose normal form is in the closure of $t_1, \dots, t_m$ with respect to the symbols of F_{LRA}. For example, $w - 2 \simeq_{\mathsf{Q}} w$ can be normalized to $-2 \simeq_{\mathsf{Q}} 0$ and evaluates to false. In the last two rules, each formula l appearing on the left stands for any formula that can be normalized to l. For instance, Fourier-Motzkin (FM) resolution applies to $y - x < 2 \cdot y$ and $2 \cdot x < 3$ yielding $-y < \frac{3}{2}$. The three linear combinations of constraints in the solution of problem R in Sect. 1 are instances of FM resolution, as a linear combination $e_1 + z < c_1,\ e_2 - z < c_2 \vdash e_1 + e_2 < c_1 + c_2$ is expressed as an FM resolution $e_2 - c_2 < z,\ z < c_1 - e_1 \vdash_{\mathsf{LRA}} e_2 - c_2 < c_1 - e_1$.

4.3 A Theory Module for EUF

For a signature $\Sigma_{\mathsf{EUF}} = (S, \simeq_S \cup F)$, $\mathcal{I}_{\mathsf{EUF}}$ may include

$$(t_i \simeq u_i)_{i=1\dots m}, f(t_1, \dots, t_m) \not\simeq f(u_1, \dots, u_m) \vdash_{\mathsf{EUF}} \bot$$
$$(t_i \simeq u_i)_{i=1\dots m} \vdash_{\mathsf{EUF}} f(t_1, \dots, t_m) \simeq f(u_1, \dots, u_m)$$
$$(t_i \simeq u_i)_{i=1\dots m, i \ne j}, f(t_1, \dots, t_m) \not\simeq f(u_1, \dots, u_m) \vdash_{\mathsf{EUF}} t_j \not\simeq u_j$$

for all symbols $f \in F$. The first rule alone is sufficient for completeness: it captures a lazy approach that does not propagate anything before equalities between existing terms are found to be in contradiction with a congruence axiom [9]. The other two rules can be used directly for eager congruence propagation. Since $\mathcal{I}_{\mathsf{EUF}}$ does not use first-order assignments, no sort needs to be EUF-public, and the only assignments assign truth values to equalities. Alternatively, one may make the sorts in S EUF-public, with a countably infinite collection of EUF^+-values in each sort and no axioms about them. Equality inferences can employ assignments of EUF^+-values to determine whether terms are equal, using EUF^+-values as identifiers for equivalence classes of terms. For example, assume that $\mathfrak{c}_1$, $\mathfrak{c}_2$, and $\mathfrak{c}_3$ are distinct EUF^+-values. The assignment $\{x \leftarrow \mathfrak{c}_1, y \leftarrow \mathfrak{c}_1, f(x) \leftarrow \mathfrak{c}_2\}$ places x and y in the equivalence class $\mathfrak{c}_1$, and $f(x)$ in class $\mathfrak{c}_2$. If $f(y) \leftarrow \mathfrak{c}_3$ is added to the assignment, two equality inferences and an application of the first rule of $\mathcal{I}_{\mathsf{EUF}}$ expose a conflict in the above-mentioned lazy style.

4.4 A Theory Module for Arrays

The array sort constructor builds from an *index sort* I and a *value sort* V the sort $I \Rightarrow V$ of arrays with indices in I and values in V. Consider a signature $\Sigma_{\mathsf{Arr}} = (S, F)$, where S is the free closure of a set of basic sorts with respect to the array sort constructor, and F is

$$\begin{array}{rll} \{\mathsf{select}_{I\Rightarrow V} & : (I\Rightarrow V)\times I \to V & \mid\ (I\Rightarrow V) \in S\} \\ \cup\ \{\mathsf{store}_{I\Rightarrow V} & : (I\Rightarrow V)\times I\times V \to (I\Rightarrow V) & \mid\ (I\Rightarrow V) \in S\} \\ \cup\ \{\mathsf{diff}_{I\Rightarrow V} & : (I\Rightarrow V)\times (I\Rightarrow V) \to I & \mid\ (I\Rightarrow V) \in S\}, \end{array}$$

where $\mathsf{diff}_{I\Rightarrow V}$ is the Skolem function symbol that arises from clausifying the extensionality axiom for array sort $I \Rightarrow V$. For brevity, subscripts are omitted, $\mathsf{select}(a, i)$ is written as $a[i]$, and $\mathsf{store}(a, i, v)$ as $a[i]{:=}v$. Module $\mathcal{I}_{\mathsf{Arr}}$ features the following rules, where a, b, c, d are variables of any $I\Rightarrow V$ sort, u, v are variables of sort V, and i, j, k of sort I:

$$\begin{array}{rl} a \simeq b,\ i \simeq j,\ a[i] \not\simeq b[j] & \vdash_{\mathsf{Arr}} \bot \\ a \simeq b,\ i \simeq j,\ u \simeq v,\ (a[i]{:=}u) \not\simeq (b[j]{:=}v) & \vdash_{\mathsf{Arr}} \bot \\ b \simeq (a[i]{:=}u),\ i \simeq j,\ b[j] \not\simeq u & \vdash_{\mathsf{Arr}} \bot \\ b \simeq (a[i]{:=}u),\ i \not\simeq j,\ j \simeq k,\ a[j] \not\simeq b[k] & \vdash_{\mathsf{Arr}} \bot \\ a \not\simeq b & \vdash_{\mathsf{Arr}} a[\mathsf{diff}(a,b)] \not\simeq b[\mathsf{diff}(a,b)] \\ a \simeq c,\ b \simeq d,\ \mathsf{diff}(a,b) \not\simeq \mathsf{diff}(c,d) & \vdash_{\mathsf{Arr}} \bot \end{array}$$

The first two inference rules capture the congruence axioms for select and store. The third and fourth rules correspond to the read-over-write axioms. The fifth rule corresponds to the clausal form of the extensionality axiom; it is the only rule that can produce new terms. The last rule states the congruence axiom for diff. These rules are triggered by the truth-values of equalities. As with $\mathcal{I}_{\mathsf{EUF}}$, in order to determine whether equalities hold, one has the option of declaring all sorts to be Arr-public, with infinitely many Arr^+-values used as identifiers of equivalence classes. One can also add rules for eager propagations of equalities.

4.5 Generic Theory Modules for Equality Sharing

Assume $\mathcal{T}$ is a stably infinite theory with signature $\Sigma = (S, F)$ and equipped with a $\mathcal{T}$-satisfiability procedure. Its inference module $\mathcal{I}_{\mathcal{T}}$ comprises the rule

$$l_1 \leftarrow \mathfrak{b}_1, \ldots, l_m \leftarrow \mathfrak{b}_m \vdash_{\mathcal{T}} \bot$$

that fires when the conjunction of the literals corresponding to the Boolean assignments on the left is $\mathcal{T}$-unsatisfiable. Unlike the previous ones, this module is coarse-grained, in the sense that a single application of its inference rule requires the execution of a $\mathcal{T}$-satisfiability procedure. As with EUF and Arr, one has the option of declaring non-Boolean sorts $\mathcal{T}$-public to determine equalities. If the $\mathcal{T}$-satisfiability procedure can produce *unsatisfiable cores*, we can restrict the above rule so that the assignment on the left is an unsatisfiable core. This provides a more precise conflict resolution mechanism, which leads us to Sect. 5.

5 The CDSAT Inference System

In this section we present CDSAT and exemplify its features by applying it to problems R and P in the introduction. A CDSAT derivation transforms a state consisting of a *trail* Γ. A *trail* is a sequence of distinct singleton assignments that are either *justified assignments*, denoted ${}_{H\vdash}A$, or *decisions*, denoted ${}_{?}A$. The justification H in ${}_{H\vdash}A$ is a set of singleton assignments that appear before A in the trail. For instance, a theory inference $J \vdash_{\mathcal{I}_k} L$ for some k, $1 \leq k \leq n$, can justify adding ${}_{J\vdash}L$ to the trail. A decision is written ${}_{?}A$ because it is generally a guess. A trail can be used as an assignment by ignoring order and justifications.

Phase 1			
id	trail items	just	lev
0	$-2\cdot x - y < 0$	$\{\}$	0
1	$x + y < 0$	$\{\}$	0
2	$x < -1$	$\{\}$	0
3	$y \leftarrow 0$		1
4	$-y < -2$	$\{0,2\}$	0
	conflict E^1: $\{3,4\}$		1

Phase 2			
id	trail items	just	lev
0	$-2\cdot x - y < 0$	$\{\}$	0
1	$x + y < 0$	$\{\}$	0
2	$x < -1$	$\{\}$	0
3	$-y < -2$	$\{0,2\}$	0
4	$y \leftarrow 4$		1
5	$y < 0$	$\{0,1\}$	0
	conflict E^2: $\{4,5\}$		1

Phase 3			
id	trail items	just	lev
0	$-2\cdot x - y < 0$	$\{\}$	0
1	$x + y < 0$	$\{\}$	0
2	$x < -1$	$\{\}$	0
3	$-y < -2$	$\{0,2\}$	0
4	$y < 0$	$\{0,1\}$	0
5	$0 < -2$	$\{3,4\}$	0
	conflict E^3: $\{5\}$		0

Fig. 2. CDSAT derivation in one theory (LRA)

The evolution of the trail for problem R is described in three successive phases in Fig. 2. The input is shown above the horizontal line of Phase 1. In each proceeding phase, the assignments above the horizontal line are those inherited from the previous phase. Assignments are numbered in chronological order, and their numbers, shown in the first column, are used as identifiers. For every justified assignment, the justification is shown as a set of identifiers in the third column. In the sequel, A_n^m is the assignment with identifier n in phase m. For example, the justification of A_4^1 is $\{0, 2\}$, because A_4^1 is derived by the FM resolution rule of $\mathcal{I}_{\mathsf{LRA}}$ from A_0^1 and A_2^1. The last column shows the *level* of A as defined next.

Definition 5 (Level). *Given a trail Γ with assignments $A_0, \ldots, A_m$,*

$$\mathsf{level}_\Gamma(A_i) = \begin{cases} 1 + max\{\mathsf{level}_\Gamma(A_j) \mid j < i\} & \textit{if } A_i \textit{ is a decision,} \\ \mathsf{level}_\Gamma(H) & \textit{if } A_i \textit{ has justification } H; \end{cases}$$

Given a $\mathcal{T}_\infty$-assignment $H \subseteq \Gamma$,

$$\mathsf{level}_\Gamma(H) = \begin{cases} 0 & \textit{if } H = \emptyset, \\ max\{\mathsf{level}_\Gamma(A) \mid A \in H\} & \textit{otherwise.} \end{cases}$$

The restriction of a trail Γ to its elements of level at most m is written $\Gamma^{\leq m}$.

The rules of the CDSAT inference system, given in Fig. 3, comprise *search rules*, whose application is denoted by $\longrightarrow$, and *conflict resolution rules*, whose application is denoted by $\Longrightarrow$, with transitive closure $\Longrightarrow^*$. The system is parameterized by a set $\mathcal{B}$ of terms, called *global basis*, used to limit the range of terms that CDSAT may generate, in order to ensure termination (cf. Sect. 6.2).

Search rules		
Decide	$\Gamma \longrightarrow \Gamma, {}_{?}A$	if A is a $\mathcal{T}_k$-assignment for a $\mathcal{T}_k$-relevant term of Γ that is acceptable for $\Gamma_{\mathcal{T}_k}$ and $\mathcal{I}_k$, with $1 \leq k \leq n$
The next three rules share the conditions: $J \subseteq \Gamma$, $(J \vdash_{\mathcal{I}_k} L)$, and $L \notin \Gamma$, for some k, $1 \leq k \leq n$.		
Deduce	$\Gamma \longrightarrow \Gamma, {}_{J\vdash}L$	if $\overline{L} \notin \Gamma$, and L is of the form $l \leftarrow \mathfrak{b}$ for some $l \in \mathcal{B}$
Fail	$\Gamma \longrightarrow$ unsat	if $\overline{L} \in \Gamma$ and $\mathsf{level}_{\Gamma}(J, \overline{L}) = 0$
ConflictSolve	$\Gamma \longrightarrow \Gamma'$	if $\overline{L} \in \Gamma$, $\mathsf{level}_{\Gamma}(J, \overline{L}) > 0$, and $\langle \Gamma; J, \overline{L} \rangle \Longrightarrow^{*} \Gamma'$
Conflict resolution rules		
Undo		
$\langle \Gamma; E, A \rangle$	$\Longrightarrow \Gamma^{\leq m-1}$	if A is a first-order decision of level $m > \mathsf{level}_{\Gamma}(E)$
Backjump		
$\langle \Gamma; E, L \rangle$	$\Longrightarrow \Gamma^{\leq m}, {}_{E\vdash}\overline{L}$	if $\mathsf{level}_{\Gamma}(L) > m$, where $m = \mathsf{level}_{\Gamma}(E)$
Resolve		
$\langle \Gamma; E, A \rangle$	$\Longrightarrow \langle \Gamma; E \cup H \rangle$	if ${}_{H\vdash}A$ is in Γ and H does not contain a first-order decision whose level is $\mathsf{level}_{\Gamma}(E, A)$
UndoDecide		
$\langle \Gamma; E, L, L' \rangle$	$\Longrightarrow \Gamma^{\leq m-1}, {}_{?}\overline{L}$	if ${}_{H\vdash}L$ and ${}_{H'\vdash}L'$ are in Γ and $H \cap H'$ contains a first-order decision of level $m = \mathsf{level}_{\Gamma}(E, L, L')$

Fig. 3. The CDSAT inference system

The global basis is fixed throughout a CDSAT derivation but depends on the input problem. We describe next the CDSAT rules, beginning with the search rules. The Decide rule extends a trail Γ with a theory assignment A without justifying it by a theory inference: it is a decision. A assigns a value to a relevant term, and is acceptable for the theory view of the trail and the theory module (cf. Definitions 2, 3 and 4). In Fig. 2, y is the relevant term in A_3^1 and A_4^2.

The Deduce rule extends a trail Γ with an assignment L justified by a theory inference $J \vdash_{\mathcal{I}_k} L$. In Fig. 2, Deduce infers A_4^1 from $\{A_0^1, A_2^1\}$, A_5^2 from $\{A_0^2, A_1^2\}$, and A_5^3 from $\{A_3^3, A_4^3\}$, by using the FM resolution rule of $\mathcal{I}_{\mathsf{LRA}}$.

Rules Fail and ConflictSolve apply to a trail Γ that is conflicting because a theory inference $J \vdash_{\mathcal{I}_k} L$ contradicts an assignment $\overline{L}$ already present in Γ. The set $J \cup \{\overline{L}\}$ is the *conflict*. Its level is denoted $\mathsf{level}_{\Gamma}(J, \overline{L})$. If it is 0, rule Fail returns unsat (e.g., E^3 in Fig. 2). If it is greater than 0 (e.g., E^1 and E^2 in Fig. 2), rule ConflictSolve triggers a series of conflict resolution steps transforming Γ into a trail Γ' where the conflict is solved. In all three conflicts in Fig. 2, the inferences that expose the conflict are applications of the evaluation rule of $\mathcal{I}_{\mathsf{LRA}}$: $y \leftarrow 0 \vdash \overline{-y < -2}$ for E^1, $y \leftarrow 4 \vdash \overline{y < 0}$ for E^2, and $\emptyset \vdash \overline{0 < -2}$ for E^3.

We now describe the conflict resolution rules, referring to Fig. 2 and 4 for their application to problems R and P, respectively. Conflict resolution rules operate on pairs $\langle \Gamma; E \rangle$, where Γ is a trail and E is a set of assignments in Γ termed *conflict*. If the conflict contains a first-order decision A, whose level n is greater than that of the rest of the conflict, rule Undo removes A and all assignments of level greater than or equal to n. In Fig. 2, rule Undo solves conflicts E^1 and E^2.

Phase 1

id	trail items	just	lev
0	$f((a[i]{:=}v)[j]) \simeq w$	$\{\}$	0
1	$w{-}2 \simeq f(u)$	$\{\}$	0
2	$i \simeq j$	$\{\}$	0
3	$u \simeq v$	$\{\}$	0
4	$u{\leftarrow}\mathfrak{c}$		1
5	$v{\leftarrow}\mathfrak{c}$		2
6	$(a[i]{:=}v)[j]{\leftarrow}\mathfrak{c}$		3
7	$w{\leftarrow}0$		4
8	$f((a[i]{:=}v)[j]){\leftarrow}0$		5
9	$f(u){\leftarrow}{-2}$		6
10	$u \simeq (a[i]{:=}v)[j]$	$\{4,6\}$	3
11	$f(u) \not\simeq f((a[i]{:=}v)[j])$	$\{8,9\}$	6
	conflict E^1: $\{10,11\}$		6

Phase 2

id	trail items	just	lev
0	$f((a[i]{:=}v)[j]) \simeq w$	$\{\}$	0
1	$w{-}2 \simeq f(u)$	$\{\}$	0
2	$i \simeq j$	$\{\}$	0
3	$u \simeq v$	$\{\}$	0
4	$u{\leftarrow}\mathfrak{c}$		1
5	$v{\leftarrow}\mathfrak{c}$		2
6	$(a[i]{:=}v)[j]{\leftarrow}\mathfrak{c}$		3
7	$u \simeq (a[i]{:=}v)[j]$	$\{4,6\}$	3
8	$f(u) \simeq f((a[i]{:=}v)[j])$	$\{7\}$	3
9	$f(u) \simeq w$	$\{0,8\}$	3
10	$w{-}2 \simeq w$	$\{1,9\}$	3
	conflict E^2_1: $\{10\}$		3
	conflict E^2_2: $\{1,9\}$		3
	conflict E^2_3: $\{0,1,8\}$		3
	conflict E^2_4: $\{0,1,7\}$		3

Phase 3

id	trail items	just	lev
0	$f((a[i]{:=}v)[j]) \simeq w$	$\{\}$	0
1	$w{-}2 \simeq f(u)$	$\{\}$	0
2	$i \simeq j$	$\{\}$	0
3	$u \simeq v$	$\{\}$	0
4	$u \not\simeq (a[i]{:=}v)[j]$	$\{0,1\}$	0
5	$u{\leftarrow}\mathfrak{c}$		1
6	$v{\leftarrow}\mathfrak{c}$		2
7	$(a[i]{:=}v)[j]{\leftarrow}\mathfrak{d}$		3
8	$v \not\simeq (a[i]{:=}v)[j]$	$\{6,7\}$	3
	conflict E^3: $\{2,8\}$		3

Phase 4

id	trail items	just	lev
0	$f((a[i]{:=}v)[j]) \simeq w$	$\{\}$	0
1	$w{-}2 \simeq f(u)$	$\{\}$	0
2	$i \simeq j$	$\{\}$	0
3	$u \simeq v$	$\{\}$	0
4	$u \not\simeq (a[i]{:=}v)[j]$	$\{0,1\}$	0
5	$v \simeq (a[i]{:=}v)[j]$	$\{2\}$	0
	conflict E^4: $\{3,4,5\}$		0

Fig. 4. CDSAT derivation in three theories (LRA, EUF, and Arr)

The Backjump rule is similar in that the conflict contains an assignment whose level is greater than that of all others. Backjump applies if this assignment is a Boolean assignment L; its flip $\overline{L}$ is justified by the rest of the conflict E. Therefore we backjump to the level of E, and add ${}_{E\vdash}\overline{L}$ to the trail. Assignment $\overline{L}$ is a Unique Implication Point [14]. We see an application of this rule in Fig. 4. Phase 1 starts with a series of decisions, from A^1_4 through A^1_9, where $\mathfrak{c}$ is an Arr^+-value of sort V, and A^1_5 is the only acceptable choice given A^1_3 and A^1_4. Then Deduce generates A^1_{10} and A^1_{11} by equality inferences (cf. Fig. 1), and ConflictSolve applies, as $E^1 \vdash_{\mathcal{I}_{\mathsf{EUF}}} \bot$ by the first inference rule of $\mathcal{I}_{\mathsf{EUF}}$. Rule Undo does not apply to E^1, because E^1 does not contain first-order decisions, but rule Backjump does apply, with A^1_{11} playing the role of L. CDSAT jumps back to level 3, the level of A^1_{10}, and places $\overline{A^1_{11}}$ on the trail with justification A^1_{10}, named A^2_7 in Phase 2. Deduce places A^2_9 and A^2_{10} on the trail by transitivity of equality (cf. Fig. 1). ConflictSolve applies as $\emptyset \vdash_{\mathsf{LRA}} w{-}2 \not\simeq w$.

The Resolve rule unfolds a conflict by replacing an assignment A in the conflict with its justification H, provided H does not introduce a first-order decision of the same level as that of the conflict. Starting from E_1^2 in Fig. 4, three Resolve steps yield conflict E_4^2. Resolve does not apply to A_7^2 because its justification contains A_6^2. Backjump solves E_4^2 by jumping back to level 0 and flipping A_7^2 into A_4^3. Then CDSAT guesses A_5^3 through A_7^3, where A_6^3 is forced by A_3^3 and A_5^3. For A_7^3, another Arr^+-value $\mathfrak{d}$ of sort V is used, since A_4^3 and A_5^3 prevent assigning $\mathfrak{c}$ to $(a[i]{:=}v)[j]$. Deduce generates A_8^3 by an equality inference, and conflict E^3 arises as $(i \simeq j), (v \not\simeq (a[i]{:=}v)[j]) \vdash_{\mathsf{Arr}} \bot$. Backjump solves E^3 by jumping back to level 0 and flipping A_8^3 into A_5^4. The final conflict E^4 violates transitivity of equality, and because E^4 is at level 0, rule Fail closes the derivation.

The UndoDecide rule corresponds to *T-backjump-decide* [5] and *semantic split* [9]. It applies when the conflict contains two assignments L and L' whose justifications include a first-order decision of maximal level in the conflict. Rule Resolve is barred from replacing L or L' by their justification, so the only way to solve the conflict is to trade the first-order decision for a Boolean decision on the flip of L or L'. In Fig. 5, conflict E_4 is solved by UndoDecide, as both A_3^1 and A_4^1 are justified by the first-order decision A_2^1. UndoDecide arbitrarily chooses to flip A_3^1, and then values can be found for variables without raising a conflict: the problem is satisfiable.

Phase 1			
id	trail items	just	lev
0	$(x{>}1) \vee (y{<}0)$	$\{\}$	0
1	$(x{<}{-1}) \vee (y{>}0)$	$\{\}$	0
2	$x{\leftarrow}0$		1
3	$\overline{x{>}1}$	$\{2\}$	1
4	$\overline{x{<}{-1}}$	$\{2\}$	1
5	$y{<}0$	$\{0,3\}$	1
6	$y{>}0$	$\{1,4\}$	1
7	$0{<}0$	$\{5,6\}$	1
	conflict E_1: $\{7\}$		1
	conflict E_2: $\{5,6\}$		1
	conflict E_3: $\{0,3,6\}$		1
	conflict E_4: $\{0,1,3,4\}$		1

Phase 2			
id	trail items	just	lev
0	$(x{>}1) \vee (y{<}0)$	$\{\}$	0
1	$(x{<}{-1}) \vee (y{>}0)$	$\{\}$	0
2	$x{>}1$		1
3	$x{\leftarrow}2$		2
4	$\overline{x{<}{-1}}$	$\{3\}$	2
5	$y{>}0$	$\{1,4\}$	2
6	$y{\leftarrow}1$		3
7	$\overline{y{<}0}$	$\{6\}$	3

Fig. 5. CDSAT derivation in two theories (Bool and LRA)

6 Soundness, Termination, and Completeness of CDSAT

In this section we establish *soundness*, *termination*, and *completeness* of CDSAT. The proofs of these theorems can be found in the technical report [3]. The key point is to reduce such global properties to theory-local requirements for the theory modules involved in the combination. In other words, we need to discover sufficient conditions whose fulfillment by all theory modules $\mathcal{I}_1, \ldots, \mathcal{I}_n$ ensures soundness, termination, and completeness of the combined system.

This reduction raises the issue of how to handle the fact that assignments contain symbols unknown to a theory. For the combination of theory modules to be truly *modular*, $\mathcal{I}_k$ treats as a variable any subterm whose root is a symbol foreign to $\mathcal{T}_k$. Formally, if $\Sigma = (S, F)$ is a signature included in Σ_∞, the *free Σ-variables* $\mathsf{fv}_\Sigma(t)$ of a term t are the *maximal* subterms of t, in the *subterm ordering* $\lhd$, whose root is not in F. For a set X of terms, $\mathsf{fv}_\Sigma(X) = \{u \mid u \in \mathsf{fv}_\Sigma(t), t \in X\}$, and for an assignment H, $\mathsf{fv}_{\Sigma}(H) = \{u \mid u \in \mathsf{fv}_{\Sigma}(t), t{\leftarrow}\mathfrak{c} \in H\}$. For problem P in Sect. 1, signatures $\Sigma_{\mathsf{LRA}}, \Sigma_{\mathsf{EUF}}, \Sigma_{\mathsf{Arr}}$ of Example 1 define for instance:

$$\begin{array}{rl}
\mathsf{fv}_{\Sigma_{\mathsf{LRA}}}(P) = & \{\ f(\mathsf{select}(\mathsf{store}(a,i,v),j)),\ w,\ f(u),\ i \simeq j,\ u \simeq v\ \} \\
\mathsf{fv}_{\Sigma_{\mathsf{EUF}}}(P) = & \{\ \mathsf{select}(\mathsf{store}(a,i,v),j),\ w,\ u,\ w{-}2,\ i \simeq j,\ v\ \} \\
\mathsf{fv}_{\Sigma_{\mathsf{Arr}}}(P) = & \{\ f(\mathsf{select}(\mathsf{store}(a,i,v))j) \simeq w,\ f(u) \simeq w{-}2,\ i,\ j,\ u,\ v\ \}
\end{array}$$

In the next two definitions, $\mathcal{T}$ and Σ stand for either $\mathcal{T}_\infty$ and Σ_∞ or $\mathcal{T}_k$ and Σ_k, $1 \le k \le n$. The identification of sufficient conditions for soundness and completeness and their proofs demand that we relate the assignments manipulated by CDSAT to models. This is the purpose of the notion of *endorsement*:

Definition 6 (Endorsement). *A $\mathcal{T}^+[\mathcal{V}]$-model $\mathcal{M}$* endorses *a $\mathcal{T}$-assignment J, such that $\mathsf{fv}_{\Sigma}(J) \subseteq \mathcal{V}$, if for all $t{\leftarrow}\mathfrak{c}$ in J, $\mathcal{M}(t) = \mathfrak{c}^{\mathcal{M}}$.*

For Boolean assignments, it means that formulae are interpreted with the correct truth values. Definition 6 uses $\mathcal{T}^+$-models, because assignments contain $\mathcal{T}^+$-values (e.g., $\sqrt{2}$), and therefore we need models that interpret $\mathcal{T}^+$-values, and interpret them consistently with the axioms (e.g., $\sqrt{2}{\cdot}\sqrt{2} = 2$).

Definition 7 (View endorsement). *A $\mathcal{T}^+[\mathcal{V}]$-model $\mathcal{M}$* view-endorses *a $\mathcal{T}_\infty$-assignment H with $\mathsf{fv}_{\Sigma}(H) \subseteq \mathcal{V}$, if it endorses its $\mathcal{T}$-view $H_{\mathcal{T}}$.*

This definition combines endorsement and view (cf. Definition 2) because CDSAT works with $\mathcal{T}_\infty$-assignments, which mix $\mathcal{T}_k$-assignments for any k, $1 \le k \le n$. If H is Boolean, view endorsement collapses to endorsement.

6.1 Soundness

The sufficient condition for soundness is that for every theory module $\mathcal{I}_k$, for all $\mathcal{I}_k$-inferences $J \vdash_{\mathcal{I}_k} L$, and all $\mathcal{V}$ such that $\mathsf{fv}_{\Sigma_k}(J \cup \{L\}) \subseteq \mathcal{V}$, every $\mathcal{T}_k^+[\mathcal{V}]$-model that view-endorses J endorses L. Under this assumption, we prove that CDSAT is sound, by showing that each transition rule produces a trail whose restriction to level 0 is equisatisfiable to the input assignment.

Theorem 1 (Soundness). *For all input assignments H, if a CDSAT derivation from H reaches state unsat, no $\mathcal{T}_\infty^+[\mathcal{V}]$-model with $\mathsf{fv}_{\Sigma_\infty}(H) \subseteq \mathcal{V}$ view-endorses H; if H is Boolean, no $\mathcal{T}_\infty[\mathcal{V}]$-model with $\mathsf{fv}_{\Sigma_\infty}(H) \subseteq \mathcal{V}$ endorses H.*

All theory modules in Sect. 4 satisfy the soundness requirement.

6.2 Termination

As CDSAT allows the introduction of terms that are not in the input problem (cf. Deduce), termination is imperiled. For instance, applying the FM resolution rule of $\mathcal{I}_{\mathsf{LRA}}$ to problem R from Sect. 1, one can infer the formulae of Fig. 6. Such divergence is prevented by imposing finiteness of the global basis $\mathcal{B}$, that is the source of new terms in a CDSAT derivation.

$$\begin{array}{lll}
l_0: & -2\cdot x - y < 0 & \\
l_1: & x + y < 0 & \\
l_2: & x < -1 & \\
l_3: & -y < -2 & (l_0 + 2l_2) \\
l_4: & x < -2 & (l_1 + l_3) \\
l_5: & -y < -4 & (l_0 + 2l_4) \\
l_6: & x < -4 & (l_1 + l_5) \\
l_7: & -y < -8 & (l_0 + 2l_6) \\
\ldots & &
\end{array}$$

Fig. 6. Divergence

Theorem 2 (Termination). *If the global basis $\mathcal{B}$ is finite, every CDSAT derivation is guaranteed to terminate.*

Then the issue is to give sufficient conditions for the existence of a global basis $\mathcal{B}$, that is finite, and yet sufficiently rich for CDSAT to be *complete*. To address this question at the combination level we begin by imposing a similar requirement at the single theory level. We require that each theory module comes with a function basis_k, called *local basis*, that maps any finite set X of terms to a *finite* set of terms $\mathsf{basis}_k(X)$, and has the following properties: it is (i) *extensive* ($X \subseteq \mathsf{basis}_k(X)$), (ii) *monotone* ($X \subseteq Y$ implies $\mathsf{basis}_k(X) \subseteq \mathsf{basis}_k(Y)$), (iii) *idempotent* ($\mathsf{basis}_k(\mathsf{basis}_k(X)) = \mathsf{basis}_k(X)$), (iv) *downward-closed with respect to the subterm ordering* (if $t \lhd u$ and $u \in \mathsf{basis}_k(X)$ then $t \in \mathsf{basis}_k(X)$), (v) *closed with respect to equality* (if $t, u \in \mathsf{basis}_k(X)$, of a sort s different from prop, then $(t \simeq_s u) \in \mathsf{basis}_k(X)$), and (vi) does not introduce foreign symbols ($\mathsf{fv}_{\Sigma_k}(\mathsf{basis}_k(X)) \subseteq \mathsf{fv}_{\Sigma_k}(X) \cup \mathcal{V}_\infty$).

Intuitively, $\mathsf{basis}_k(X)$ is the supply of terms that $\mathcal{I}_k$ is allowed to introduce during a derivation from an input problem whose terms are in X. However, $\mathsf{basis}_k(X)$ is not pre-computed. Furthermore, basis_k should provide enough terms to make $\mathcal{I}_k$ *complete*, according to a notion of completeness of a theory module defined in the sequel in terms of both $\mathcal{I}_k$-inferences and basis_k.

The divergence in Fig. 6 involves only $\mathcal{I}_{\mathsf{LRA}}$. It can be avoided by assuming a fixed arbitrary order $\prec$ on Σ_{LRA}-variables [9], and defining $\mathsf{basis}_{\mathsf{LRA}}$ as the function that saturates its argument with the terms introduced by all positivization inferences and by the FM resolution inferences $(t_1 \lessdot_1 x), (x \lessdot_2 t_2) \vdash_{\mathsf{LRA}} (t_1 \lessdot_3 t_2)$ where x is the $\prec$-greatest Σ_{LRA}-variable in both $t_1 \lessdot_1 x$ and $x \lessdot_2 t_2$. For Fig. 6, assume that $y \prec x$. Then l_3, generated by $(-y < 2 \cdot x), (2 \cdot x < -2) \vdash_{\mathsf{LRA}} (-y < -2)$, is in the local basis, whereas l_4, generated by $(x < -y), (-y < -2) \vdash_{\mathsf{LRA}} (x < -2)$, is not, so that the series of inferences halts.

6.3 Completeness

As theory modules are used to extend the trail and reveal conflicts, the aim for completeness is that whenever no theory module can extend the trail, then the trail provides enough information to build a $\mathcal{T}_\infty^+$-model of the input problem. We begin by formalizing the concept that a theory module can extend an assignment.

Definition 8 (Assignment extension). *Module $\mathcal{I}_k$ with local basis* basis_k can extend a $\mathcal{T}_k$-assignment J *if*

- *Either there exists a $\mathcal{T}_k$-assignment $t{\leftarrow}\mathfrak{c}$, for a $\mathcal{T}_k$-relevant term t of J, that is acceptable for J and $\mathcal{I}_k$;*
- *Or there exist a $\mathcal{T}_k$-assignment $J' \subseteq J$, a formula $l \in \mathsf{basis}_k(J)$, and an $\mathcal{I}_k$-inference $J' \vdash_{\mathcal{I}_k} (l \leftarrow \mathfrak{b})$ such that $(l \leftarrow \mathfrak{b}) \notin J$.*

The first case is used for a Decide step and the second one for a Deduce, Fail, or $\mathsf{ConflictSolve}$ step. A module $\mathcal{I}_k$ is said to be *complete*, if for all plausible $\mathcal{T}_k$-assignments J that $\mathcal{I}_k$ cannot extend, there exists a $\mathcal{T}_k^+[\mathsf{fv}_{\Sigma_k}(J)]$-model $\mathcal{M}$ that view-endorses J. However, when no theory module can extend its view of a trail Γ, the existence of a theory-specific model for Γ for each theory does not imply the existence of a model for the combination of the theories. As in the equality-sharing method [16], these models need to agree on equalities between shared variables and on cardinalities of shared sorts. If all theories are *stably infinite*, the common cardinality is countably infinite. Nonetheless, there are interesting combinations that involve finite cardinalities, such as combining a theory with finite sorts and the theory of arrays with extensionality. CDSAT can handle such cases, if one of the combined theories, say $\mathcal{T}_1$, knows all the sorts (i.e., $S_1 = S_\infty$) and offers information about their cardinalities. A combination of stably infinite theories is the instance of this scheme where $\mathcal{T}_1$ is a theory $\mathcal{T}_\mathbb{N}$ whose models interpret every sort in $S_\infty \setminus \{\mathsf{prop}\}$ as a countably infinite set.

The *theory-specific requirements for completeness* of CDSAT are that $\mathcal{I}_1$ is complete, and all other modules are complete *relative* to $\mathcal{T}_1$. The latter notion, that we call $\mathcal{T}_1$-*completeness*, in turn relies on $\mathcal{T}_1$-*compatibility*, defined below.

Definition 9 ($\mathcal{T}_1$-compatibility). *A $\mathcal{T}_k$-assignment J is* $\mathcal{T}_1$-compatible with $\mathcal{T}_k^+$, sharing a set of terms G, *if for any $\mathcal{T}_1^+[\mathsf{fv}_{\Sigma_1}(J \cup G)]$-model $\mathcal{M}_1$ that view-endorses J, there exists a $\mathcal{T}_k^+[\mathsf{fv}_{\Sigma_k}(J \cup G)]$-model $\mathcal{M}$ that view-endorses J, such that for all sorts $s \in S_k$, $\left|s^{\mathcal{M}}\right| = \left|s^{\mathcal{M}_1}\right|$, and for all terms t and t' in G of sort s, $\mathcal{M}(t) = \mathcal{M}(t')$ if and only if $\mathcal{M}_1(t) = \mathcal{M}_1(t')$.*

A module $\mathcal{I}_k$ is $\mathcal{T}_1$-*complete*, if for all plausible $\mathcal{T}_k$-assignments J that $\mathcal{I}_k$ cannot extend, J is $\mathcal{T}_1$-compatible with $\mathcal{T}_k^+$, sharing all terms that occur in J. Then, the global basis $\mathcal{B}$ is *stable*, if $\mathsf{basis}_k(\mathcal{B}) \subseteq \mathcal{B}$ holds for all k, $1 \leq k \leq n$.

Theorem 3 (Completeness). *For all input assignments H, if the global basis $\mathcal{B}$ is stable and contains all terms that occur in H, whenever a CDSAT derivation from H reaches a state Γ other than unsat such that no CDSAT inference applies, there exists a $\mathcal{T}_\infty^+[\mathsf{fv}_{\Sigma_\infty}(\Gamma)]$-model that view-endorses Γ and H contained in Γ.*

The proof of this theorem [3] relies on the following:

Lemma 1 (Model glueing). *Let H be an assignment and G be the collection of* shared terms *inductively defined by:*

$$\frac{(t \leftarrow \mathfrak{c}) \in H}{t \in G} \qquad \frac{u, u' \in G \quad t \in \mathsf{fv}_{\Sigma_i}(u) \cap \mathsf{fv}_{\Sigma_j}(u') \quad i \neq j}{t \in G} \qquad \frac{u \in G \quad t \in \mathsf{fv}_{\Sigma_k}(u) \setminus \mathcal{V}_\infty}{t \in G}$$

If there exists a $\mathcal{T}_1^+[\mathsf{fv}_{\Sigma_1}(H)]$-model that view-endorses H, and such that for all k, $2 \leq k \leq n$, the $\mathcal{T}_k$-view $H_{\mathcal{T}_k}$ is $\mathcal{T}_1$-compatible with $\mathcal{T}_k^+$ sharing G, then there exists a $\mathcal{T}_\infty^+[\mathsf{fv}(H)]$-model that view-endorses H.

A derivation can reach a state satisfying the hypotheses of Lemma 1 long before it reaches a state that no module can extend. An implementation of a module could notify the main algorithm when the trail becomes $\mathcal{T}_1$-compatible with its theory. In this sense, Theorem 3 covers the worst-case scenario. A *stable* global basis can be obtained by taking $\mathcal{B} = \mathsf{basis}_{\pi(k)}(\ldots \mathsf{basis}_{\pi(1)}(X))$, where X is the set of terms occurring in the input assignment and π is a permutation of $\{1, \ldots, k\}$ that satisfies the following property: if $i < j$ then $\mathsf{basis}_{\pi(i)}(\mathsf{basis}_{\pi(j)}(X)) \subseteq \mathsf{basis}_{\pi(j)}(\mathsf{basis}_{\pi(i)}(X))$. A syntactic criterion on the local bases implies this permutability property [3].

For all theory modules of Sect. 4, except $\mathcal{I}_{\mathsf{LRA}}$, we can define a local basis that makes them $\mathcal{T}_{\mathbb{N}}$-complete (cf. [3] for stronger completeness properties). For $\mathcal{I}_{\mathsf{LRA}}$, the local basis $\mathsf{basis}_{\mathsf{LRA}}$ given above makes the module $\mathcal{T}_{\mathbb{N}}$-complete only under the strategy that assigns Σ_{LRA}-variables in $\prec$-increasing order. Otherwise, considering again problem R and the ordering $y \prec x$, the LRA-assignment $l_0, l_1, l_2, l_3, (x \leftarrow 0)$ cannot be extended by $\mathcal{I}_{\mathsf{LRA}}$ even though it is LRA-unsatisfiable. Indeed, the obvious FM resolution combining l_1 and l_3 would eliminate y, which is not maximal in l_1, as required by $\mathsf{basis}_{\mathsf{LRA}}$. An additional inference rule can be added to $\mathcal{I}_{\mathsf{LRA}}$ to make it complete regardless of strategy [3].

7 Discussion

In this paper we introduced CDSAT, a *conflict-driven* system for deciding the satisfiability of quantifier-free problems in the union of disjoint theories. CDSAT combines theory inference systems, termed *theory modules*. We presented several theory modules, including one for arrays which is the first integration of this theory in a conflict-driven combination. CDSAT lifts CDCL to SMT in the sense of *satisfiability modulo multiple theories*. Since it accepts input problems containing Boolean and first-order assignments, CDSAT solves a class of problems that extends SMT and that we call SMA for *Satisfiability Modulo Assignments*. For such problems, the input format presupposes the theory extensions (cf. Sect. 3).

CDSAT generalizes MCSAT [5,8,9,19] to theory combinations. Furthermore, CDSAT solves the hitherto open problem of integrating conflict-driven procedures and the black-box solvers used in the *equality sharing* method [16]. CDSAT generalizes equality sharing itself, which corresponds to the case where all theories are stably infinite, all theory modules are black-boxes (cf. Sect. 4.5), and CDSAT decisions are limited to equalities between shared variables.

Clause learning, including theory lemmas, can be easily added to the version of CDSAT presented here [3]. Directions for future work include: the generation of proofs, by composition of theory inferences; efficient techniques to detect the applicability of theory inference rules and determine whether an assignment is acceptable (e.g., watched variables [9]); and heuristic strategies to make decisions and prioritize theory inferences.

Acknowledgments The authors thank Dejan Jovanović for fruitful discussions. Part of this research was conducted while the first author was an international fellow at the Computer Science Laboratory of SRI International, whose support is greatly appreciated. This research was funded in part by NSF grants 1528153 and CNS-0917375, by DARPA under agreement number FA8750-16-C-0043, and by grant "Ricerca di base 2015" of the Università degli Studi di Verona. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, DARPA, or the U.S. Government.

References

1. Barrett, C., Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Splitting on demand in SAT modulo theories. In: Hermann, M., Voronkov, A. (eds.) Proceedings of the Thirteenth International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR). Lecture Notes in Artificial Intelligence, vol. 4246, pp. 512–526. Springer (2006) 2
2. Bonacina, M.P.: On conflict-driven reasoning. In: Dutertre, B., Shankar, N. (eds.) Proceedings of the Sixth Workshop on Automated Formal Methods (AFM), at the Ninth NASA Formal Methods Symposium (NFM). pp. 1–9. To appear (2017), http://fm.csl.sri.com/AFM17/ 1
3. Bonacina, M.P., Graham-Lengrand, S., Shankar, N.: A model-constructing framework for theory combination. Tech. Rep. 99/2016, Dipartimento di Informatica, Università degli Studi di Verona, Verona, Italy, EU (November 2016), https://hal.archives-ouvertes.fr/hal-01425305, also Technical Report of SRI International and INRIA - CNRS - École Polytechnique; revised April 2017 12, 15, 16
4. Cotton, S.: Natural domain SMT: A preliminary assessment. In: Chatterjee, K., Henzinger, T.A. (eds.) Proceedings of the Eighth International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS). Lecture Notes in Computer Science, vol. 6246, pp. 77–91. Springer (2010) 1
5. de Moura, L., Jovanović, D.: A model-constructing satisfiability calculus. In: Giacobazzi, R., Berdine, J., Mastroeni, I. (eds.) Proceedings of the Fourteenth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI). Lecture Notes in Computer Science, vol. 7737, pp. 1–12. Springer (2013) 2, 12, 16
6. Ganzinger, H., Rueß, H., Shankar, N.: Modularity and refinement in inference systems. Tech. Rep. CSL-SRI-04-02, Computer Science Laboratory, SRI International, Menlo Park, CA, USA (2004) 3
7. Haller, L., Griggio, A., Brain, M., Kroening, D.: Deciding floating-point logic with systematic abstraction. In: Cabodi, G., Singh, S. (eds.) Proceedings of the Twelfth International Conference on Formal Methods in Computer Aided Design (FMCAD). ACM and IEEE (2012) 1
8. Jovanović, D.: Solving nonlinear integer arithmetic with MCSAT. In: Bouajjani, A., Monniaux, D. (eds.) Proceedings of the Eighteenth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI). Lecture Notes in Computer Science, vol. 10145, pp. 330–346. Springer (2017) 2, 16
9. Jovanović, D., Barrett, C., de Moura, L.: The design and implementation of the model-constructing satisfiability calculus. In: Jobstman, B., Ray, S. (eds.) Proceedings of the Thirteenth Conference on Formal Methods in Computer Aided Design (FMCAD). ACM and IEEE (2013) 2, 5, 7, 12, 14, 16
10. Jovanović, D., de Moura, L.: Cutting to the chase: solving linear integer arithmetic. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) Proceedings of the Twenty-Third International Conference on Automated Deduction (CADE). Lecture Notes in Artificial Intelligence, vol. 6803, pp. 338–353. Springer (2011) 1
11. Jovanović, D., de Moura, L.: Solving non-linear arithmetic. In: Gramlich, B., Miller, D., Sattler, U. (eds.) Proceedings of the Sixth International Joint Conference on Automated Reasoning (IJCAR). Lecture Notes in Artificial Intelligence, vol. 7364, pp. 339–354. Springer (2012) 1

12. Korovin, K., Tsiskaridze, N., Voronkov, A.: Conflict resolution. In: Gent, I.P. (ed.) Proceedings of the Fifteenth International Conference on Principles and Practice of Constraint Programming (CP). Lecture Notes in Computer Science, vol. 5732, pp. 509–523. Springer (2009) 1
13. Krstić, S., Goel, A.: Architecting solvers for SAT modulo theories: Nelson-Oppen with DPLL. In: Wolter, F. (ed.) Proceedings of the Sixth International Symposium on Frontiers of Combining Systems (FroCoS). Lecture Notes in Artificial Intelligence, vol. 4720, pp. 1–27. Springer (2007) 2
14. Marques Silva, J., Lynce, I., Malik, S.: Conflict-driven clause learning SAT solvers. In: Biere, A., Heule, M., Van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, pp. 131–153. IOS Press (2009) 1, 11
15. McMillan, K.L., Kuehlmann, A., Sagiv, M.: Generalizing DPLL to richer logics. In: Bouajjani, A., Maler, O. (eds.) Proceedings of the Twenty-First International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 5643, pp. 462–476. Springer (2009) 1
16. Nelson, G., Oppen, D.C.: Simplification by cooperating decision procedures. ACM Transactions on Programming Languages and Systems 1(2), 245–257 (1979) 2, 15, 16
17. Wang, C., Ivančić, F., Ganai, M., Gupta, A.: Deciding separation logic formulae by SAT and incremental negative cycle elimination. In: Sutcliffe, G., Voronkov, A. (eds.) Proceedings of the Twelfth International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR). Lecture Notes in Artificial Intelligence, vol. 3835, pp. 322–336. Springer (2005) 1
18. Wolfman, S.A., Weld, D.S.: The LPSAT engine and its application to resource planning. In: Dean, T. (ed.) Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI). vol. 1, pp. 310–316. Morgan Kaufmann Publishers (1999) 1
19. Zeljić, A., Wintersteiger, C.M., Rümmer, P.: Deciding bit-vector formulas with `mcSAT`. In: Creignou, N., Le Berre, D. (eds.) Proceedings of the Nineteenth International Conference on Theory and Applications of Satisfiability Testing (SAT). Lecture Notes in Computer Science, vol. 9710, pp. 249–266. Springer (2016) 2, 16