A Verified SAT Solver Framework including Optimization and Partial Valuations

Mathias Fleury[1,2] and Christoph Weidenbach[2]

[1] Johannes Kepler University Linz, Austria mathias.fleury@jku.at
[2] Max-Planck-Institut für Informatik, Saarland Informatics Campus, Saarland
weidenb@mpi-inf.mpg.de

Abstract

Based on our formal framework for CDCL (conflict-driven clause learning) using the proof assistant Isabelle/HOL, we verify an extension of CDCL computing cost-minimal models called OCDCL. It is based on branch and bound and computes models of minimal cost with respect to total valuations. The verification starts by developing a framework for CDCL with branch and bound, called $CDCL_{BnB}$, which is then instantiated to get OCDCL. We then apply our formalization to three different applications. Firstly, through the dual rail encoding, we reduce the search for cost-optimal models with respect to partial valuations to searching for total cost-optimal models, as derived by OCDCL. Secondly, we instantiate OCDCL to solve MAX-SAT, and, thirdly, $CDCL_{BnB}$ to compute a set of covering models. A large part of the original CDCL verification framework was reused without changes to reduce the complexity of the new formalization. To the best of our knowledge, this is the first rigorous formalization of CDCL with branch and bound and its application to an optimizing CDCL calculus, and the first solution that computes cost-optimal models with respect to partial valuations.

1 Introduction

Researchers in automated reasoning spend a significant portion of their work time specifying calculi and proving metatheorems about them. These proofs are mostly carried out with pen and paper, which is error-prone and can be tedious. Especially when working on variants of previously devised calculi, it is easy to miss subtle but important differences. We are part of an effort, called IsaFoL (Isabelle Formalization of Logic) [1] that aims at developing libraries and methodologies to formalize these calculi using the Isabelle/HOL proof assistant [20]. This does not only include developing formal companions to paper proofs but also includes simplifying the verification of variants of earlier devised calculi.

We extend our previous formalization of conflict-driven clause learning (CDCL) [4]. An important extension to CDCL is optimizing propositional satisfiability (OPT-SAT): Given a cost function on literals, OPT-SAT aims at finding a cost-optimal model for a set of clauses. In Weidenbach's upcoming textbook, tentatively called *Automated Reasoning—Some Basics*, he developed an optimizing CDCL for OPT-SAT called OCDCL. It is based on his CDCL calculus [26] and the correctness is proved by copy-pasting and adapting the CDCL proofs.

As a case study, we formalized it. When doing so, we realized that the calculus did not find optimal models if partial valuations (i.e., valuations not assigning values to all variables) are considered: CDCL's learned-clause mechanism in the context of searching for (optimal) models is not correct with respect to partial valuations. Therefore, the developed OCDCL calculus searches for optimal models based on *total* valuations. It has similarities to Larrosa et al.'s DPLL$_{BB}$ calculus [13] but is far more concrete. Their calculus is only correct for models with respect to total valuations, too. We took this as a further motivation for formalizing OCDCL on total valuations, as well as for finding an encoding such that OCDCL is also correct with respect to partial valuations.

In contrast to most of the existing literature on finding cost-optimal models [23], we define our cost function on literals, not on atoms. For example, this more general definition arises in applications where a propositional variable encodes membership in one class and its negation membership in a different class. This is a typical encoding in a product configuration scenario. The truth of a propositional variable may encode the containment of a component in a product generating some cost. However, the absence of the component typically also generates some (often lower) cost.

Our contributions are firstly a rigorous proof of the correctness of OCDCL with respect to total valuations. Formalizations and verification in proof assistants are often justified by the idea that they can be reused and extended. This is exactly what we are doing here. CDCL with branch and bound amounts to around 1600 lines, its instantiation to weights to 1200 lines, and its restriction to OCDCL to 600 lines of correctness proof. This is small compared to the 2600 lines of shared libraries (e.g., on levels and valuations), plus 6000 lines for the original formalization of CDCL [4], and 4500 lines for Nieuwenhuis et al.'s account for CDCL [18]. Thus, secondly, we show that one of the standard arguments supporting formal verification, the reuse of already proved results, works out in the case of CDCL and OCDCL. The overall formalization took around 1.5 person−months of work and was straightforward to do. Thirdly, we further demonstrate the potential of reuse, by applying our framework to three additional problems. First, we prove that OCDCL is also correct with respect to partial valuations using a dual rail encoding [6, 21], eventually solving the problem of finding optimal models with respect to both partial valuations and total valuations using the very same framework. The extension to partial valuations amounts to 1300 lines. Second, we reduce MAX-SAT to OCDCL. Third, we apply our approach to covering models. The overall effort for the extensions was 1100, 800 and, 400 lines, respectively.

After some preliminaries on Isabelle and our previous formalization of CDCL (Section 2), the optimizing OCDCL calculus is introduced (Section 3). It is described as an abstract non-deterministic transition system and has the conflict analysis for the first unique implication point [3] built into its CDCL part. In Isabelle, instead of working directly on an optimizing calculus, we first verify an abstract calculus, CDCL$_{BnB}$. This calculus implements an abstract CDCL with branch and bound and it can be instantiated with weights to get OCDCL (Section 4). We then apply the OCDCL calculus to partial models using the dual rail encoding to solve MAX-SAT and we apply the CDCL$_{BnB}$ framework to find minimal sets of covering models (Section 5). The paper ends with a discussion of the obtained results and related work (Section 6).

All theorems that have been proved in Isabelle show in the theorem name the Isabelle theorem name from the formalization with the notation "Isa:fact ☑" with a link to the documentation of our formalization. Throughout this paper, we distinguish "Lemmas" that follow the presentation on paper from "Isabelle Lemmas" that are specific to our approach in the Isabelle formalization. We link the lemmas to their formalized counterpart even if the formalization path is different. This presentation is based on Fleury's Ph.D. thesis [8].

2   Formalization of CDCL in Isabelle

Isabelle.   Isabelle [19, 20] is a generic proof assistant and a framework that supports various logics. We use the instantiation with a fragment of higher-order logic (HOL) [7]. Base types are type variables $'a, 'b, \ldots$ and can be combined to n-ary type constructors written in postfix notation (e.g., $'a\ multiset$). We usually use curried notation for functions (e.g., f $x\ y$). The notation $t :: \tau$ is a judgment and indicates that term t has type τ.

Following the LCF system [10], inferences are derived by a small trusted kernel. Types and functions are defined instead of being axiomatized to avoid inconsistencies. For example, we can define inductive predicates and the system generates the internal low-level definitions that are used to derive the provided specification.

Isabelle developments are composed of theory files, containing definitions, properties, and proofs written in the proof language Isar. Isabelle offers two proof styles. The first one is backward where tactics manipulate the proof obligations directly. The second is forward and it is a more readable and more natural style. We mostly use the latter.

CDCL.   We have previously developed a framework [4] to formalize the conflict-driven-clause-learning procedure. Clauses are defined as (finite) multisets. For readability, we will write $\bot$ and $A \vee B$ instead of their Isabelle counterparts $\{\#\}$ and $A + B$. Given a literal L and a set of literals $\mathcal{A}$, called a *valuation*, we define entailment by $\mathcal{A} \vDash L$ iff $L \in \mathcal{A}$. We can lift it to clauses (multiset of literals) by $\mathcal{A} \vDash C$ iff there is a literal $L \in C$ such that $\mathcal{A} \vDash L$. We can also lift it to clause sets $\mathcal{A} \vDash N$ iff $\forall C \in N.\ \mathcal{A} \vDash C$. If $\mathcal{A}$ is also consistent (i.e., does not contain a literal L and its negation $\neg L$), $\mathcal{A}$ is called a model of N. A model $\mathcal{A}$ is total over the set of clauses N when all atoms of N are defined in $\mathcal{A}$; otherwise, $\mathcal{A}$ is *partial*. An atom L is defined in a valuation $\mathcal{A}$, if either the literal L or $\neg L$ belong to $\mathcal{A}$. We identify $\neg\neg L$ and L.

Conflict-driven clause learning is a procedure that builds a candidate model, called the *trail* or M. It contains literals that have either been decided ($L^\dagger$) or propagated (L^C where the clause C justifies the propagation). We identify trails and valuations by ignoring the annotations and the order of literals. As in the CDCL formalization, trails grow on the left (following Isabelle conventions on lists) and not on the right (as usually done in the CDCL literature). Each time a clause is falsified by the trail, CDCL analyzes the clauses to adapt the trail and learns a new clause to avoid running into the same dead end again. CDCL is presented as a non-deterministic transition system. It operates on tuples $(M; N; U; D)$ where M is the current candidate model, N are the initial clauses, U are learned clauses, and D is either a conflicting clause that is currently analyzed or $\top$ if there is no conflict. For example, the Decide rule extends M by an arbitrary choice L, whereas Propagate extends the trail by propagating a clause:

Decide $(M; N; U; \top) \implies (L^\dagger M; N; U; \top)$

provided L is undefined in M and atom$(L) \in N$.

Propagate $(M; N; U; \top) \implies (L^{C \vee L} M; N; U; \top)$

if L is undefined in M, $C \vee L \in N \cup U$, and $M \vDash \neg C$.

Instead of a tuple, the formalization uses abstract states of type $'st$ associated with selectors to access the different components of the states. Isabelle's *locale* mechanism [11] makes it possible to abstract over the concrete state representation and simply specify the behavior of the selectors. For example, the locale defining CDCL starts with:

```
locale CDCL_locale =
  fixes prepend_trail :: 'v ann_lit ⇒ 'st ⇒ 'st and
```

$$\text{trail} :: {'st} \Rightarrow {'v} \; ann\_lit \; list \; \texttt{and} \; \ldots$$
$$\texttt{assumes} \; \forall L \, S. \; \text{trail}(\text{prepend\_trail} \, L \, S) = L \cdot \text{trail} \, S \; \texttt{and} \; \ldots$$

Definitions in a locale can depend on the types, constants, and axioms of the locales. Locales can be combined (including sharing of parameters) and inherit from other locales. Seen from the outside, theorems in the locale are universally quantified over the constants and have as additional hypotheses the assumptions of the locales. A locale can be instantiated by the command

> **interpretation** CDCL_locale **where**
> prepend_trail = prepend_trail$'$ **and**
> trail = trail$'$ **and** ...

yielding the assumptions as the proof obligation, here $\forall L \, S. \; \text{trail}'(\text{prepend\_trail}' \, L \, S) = L \cdot \text{trail}' \, S$. Then all definitions and lemmas of the locale become available. Locales can be instantiated several times yielding different version of the constants. This mechanism turned out to be very useful in our formalization by providing different ways to instantiate the type $'st$.

In the locale of our CDCL formalization, the Decide and Propagate rules are actually defined in the following way:

inductive decide :: $'st \Rightarrow {'st} \Rightarrow bool$
where
  undefined_lit (trail S) $L \Longrightarrow$
  $L \in$ atom (clauses S) $\Longrightarrow$
  $S' \sim$ prepend_trail $(L^{\dagger})$ $S \Longrightarrow$
  decide S S'

inductive propagate :: $'st \Rightarrow {'st} \Rightarrow bool$
where
  undefined_lit (trail S) $L \Longrightarrow$
  $C \vee L \in$ clauses $S \Longrightarrow$
  trail $S \vDash \neg C \Longrightarrow$
  $S' \sim$ prepend_trail $(L^{C \vee L})$ $S \Longrightarrow$
  propagate S S'

where the functions trail S and clauses S select M and $N \cup U$, and prepend_trail $L \, S$ prepends the annotated literal L to the trail without changing the other components. Since states are equivalent if the selectors return the same components, we don't use the usual equality on $'st$, and instead use $S \sim T$ that holds if all components are equal.

To simplify the notation we will use tuples $(M; N; U; D)$ instead of referring to each component with the selectors.

3   Optimizing Conflict-Driven Clause Learning

We want to find an optimal total model w.r.t. a set of clauses and the total cost function cost on the set of all literals $Lit(\Sigma)$ over Σ into the positive reals, cost $: Lit(\Sigma) \rightarrow \mathbb{R}^+$. In Isabelle, we take all values of an arbitrary type instead of the set $Lit(\Sigma)$ and the codomain simply needs to be equipped with a linear order compatible with $+$, as we do not rely on other properties of $\mathbb{R}^+$ in our proofs. The cost function is extended to (partial) valuations by $\text{cost}(\mathcal{A}) = \sum_{L \in M_{\mathcal{A}}} \text{cost}(L)$.

The optimizing conflict-driven clause learning calculus (OCDCL) solves the OPT-SAT problem on total valuations. Compared with a normal CDCL state, a fifth component O is added. It either stores the best model so far or $\top$. We extend the cost function to $\top$ by defining cost $(\top) = \infty$ (i.e., $\top$ is the worst possible outcome). OCDCL is a strict extension of the CDCL rules with additional rules to take the cost of models into account. The additional component O is simply ignored by the original CDCL rules.

The intuition behind the extension is to see CDCL as a special case of a branch-and-bound algorithm: CDCL finds a model that is stored (the branching part), while the stored model is

4

used to limit the depth of the search (the bounding part) by adding conflict clauses that are subsequently analyzed by the usual CDCL rules. Therefore, CDCL is extended in two ways: First, models are identified as such and stored in a new component of the state. Second, CDCL runs are pruned by adding conflict clauses on-the-fly, based on the best model found so far.

The initial state for some clause set N is $(\epsilon; N; \emptyset; \top; \top)$. The calculus searches for models in the usual CDCL-style. If a partial model M already exceeds the current cost bound, a conflict clause can be generated (rule ConflOpt, defined below). Once a better model is found (rule Improve, defined below), it can be ruled out by generating a conflict (rule ConflictOpt).

The level of a literal is the number of decisions right of its atom in the trail M, including the atom. We lift the definition to clauses, by defining the level of a clause as the maximum of the levels of its literals or 0 if it is empty.

First, there are three rules involving the last component O that implement a branch-and-bound approach on the models:

Improve $\quad (M; N; U; \top; O) \implies_{\text{OCDCL}} (M; N; U; \top; M)$

provided $M \models N$, M is total over N, and $\text{cost}(M) < \text{cost}(O)$.

ConflOpt $\quad (M; N; U; \top; O) \implies_{\text{OCDCL}} (M; N; U; \neg M; O)$

provided $O \neq \top$ and $\text{cost}(M) \geq \text{cost}(O)$.

Prune $\quad (M; N; U; \top; O) \implies_{\text{OCDCL}} (M; N; U; \neg M; O)$

provided for all total trail extensions $M'M$ of M, it holds that $\text{cost}(M'M) \geq \text{cost}(O)$.

The Prune rule is not necessary for the correctness and completeness of the calculus. In practice, Prune would be an integral part of any optimizing solver where a lower-bound on the cost of all extensions of M is maintained for efficiency.

The other rules are unchanged imports from the CDCL calculus. They simply ignore the additional component O. The rules Propagate and Decide extend the trail searching for a model. The rule ConflSat detects a conflict. All three rules implement the classical CDCL-style model search until conflict or success.

Propagate $\quad (M; N; U; \top; O) \implies_{\text{OCDCL}} (L^{C \vee L} M; N; U; \top; O)$

provided $C \vee L \in N \cup U$, $M \models \neg C$, L is undefined in M.

Decide $\quad (M; N; U; \top; O) \implies_{\text{OCDCL}} (L^{\dagger} M; N; U; \top; O)$

provided L is undefined in M, contained in N.

ConflSat $\quad (M; N; U; \top; O) \implies_{\text{OCDCL}} (M; N; U; D; O)$

provided $D \in N \cup U$ and $M \models \neg D$.

Once a conflict has been found, it is analyzed to derive a new clause that is then a first unique implication point [3]. After that, the trail is adapted.

Skip $\quad (L^{C \vee L} M; N; U; D; O) \implies_{\text{OCDCL}} (M; N; U; D; O)$

provided $D \notin \{\top, \bot\}$ and $\neg L$ does not occur in D.

Resolve $\quad (L^{C \vee L} M; N; U; D \vee -L; O) \implies_{\text{OCDCL}} (M; N; U; D \vee C; O)$

provided D is of level k, where k is the number of decisions in M.

Backtrack $\quad (M_2 K^{\dagger} M_1; N; U; D \vee L; O) \implies_{\text{OCDCL}} (L^{D \vee L} M_1; N; U \cup \{D \vee L\}; \top; O)$

provided L is of level k and D and K are of level $i < k$, where k is the number of decisions in M.

Without any restriction on rule applications CDCL can get stuck in states where neither SAT nor UNSAT can be concluded. To avoid these we follow the *reasonable* strategy. Under this strategy the OCDCL calculus always terminates in deriving the empty clause $\perp$. If in this case $O = \top$, then N was unsatisfiable. Otherwise, O contains a cost-optimal total model for N.

Definition 1 (Reasonable OCDCL Strategy). *An OCDCL strategy is* reasonable *if ConflSat is preferred over ConflOpt, which is preferred over Improve, which is preferred over Propagate, which is preferred over the remaining rules.*

Example 2. *Consider the clause set $N = \{P \vee Q\}$ with $\mathrm{cost}(P) = 3$ and $\mathrm{cost}(\neg P) = \mathrm{cost}(\neg Q) = \mathrm{cost}(Q) = 1$. Here is a possible sequence of transitions:*

$$(\epsilon, N, \emptyset, \top, \top) \Longrightarrow_{Decide} (P^{\dagger}, N, \emptyset, \top, \top) \tag{1}$$
$$\Longrightarrow_{Decide} (Q^{\dagger} P^{\dagger}, N, \emptyset, \top, \top)$$
$$\Longrightarrow_{Improve} (Q^{\dagger} P^{\dagger}, N, \emptyset, \top, PQ)$$

The total model PQ of weight 4 was found, we can now exclude it:

$$\Longrightarrow_{ConflOpt} (Q^{\dagger} P^{\dagger}, N, \emptyset, \neg P \vee \neg Q, PQ)$$
$$\Longrightarrow_{Backtrack} ((\neg Q)^{\neg P \vee \neg Q} P^{\dagger}, N, \{\neg P \vee \neg Q\}, \top, PQ)$$

The trail is a total model with weight 4. We can exclude the current model:

$$\Longrightarrow_{ConflOpt} (\neg Q^{\dagger} P^{\dagger}, N, \emptyset, P \vee \neg Q, PQ)$$
$$\Longrightarrow_{Backtrack} (\neg P^{\neg P}, N, \{\neg P \vee \neg Q, \neg P\}, \top, PQ)$$
$$\Longrightarrow_{Propagate} (Q^{P \vee Q} \neg P^{\neg P}, N, \{\neg P \vee \neg Q, \neg P\}, \top, \neg PQ) \tag{2}$$

The model can be improved again:

$$\Longrightarrow_{Improve} (Q^{P \vee Q} \neg P^{\neg P}, N, \{\neg P \vee \neg Q, \neg P\}, \top, \neg PQ)$$
$$\Longrightarrow^{\star}_{ConflOpt+Resolve} (\epsilon, N, \{\neg P \vee \neg Q, \neg P\}, \perp, \neg PQ)$$

Since the conflict clause was reduced to $\perp$, we know that the clause set $N \cup \{\neg P \vee \neg Q, \neg P\}$ is unsatisfiable and the best model is $\neg PQ$ of weight 1.

Lemma 3 (Termination, Isa:wf_ocdcl$_W$ ✔). *OCDCL with a reasonable strategy terminates.*

Proof Sketch. *If the derivation started from $(\epsilon, N, \emptyset, \top, \top)$, the following function is a measure for OCDCL:*

$$\mu((M; N; U; D; O)) = \begin{cases} (3^n - 1 - |U|, 1, n - |M|, \mathrm{cost}(O)) & \text{if } D = \top \\ (3^n - 1 - |U|, 0, |M|, \mathrm{cost}(O)) & \text{otherwise} \end{cases}$$

It is decreasing for the lexicographic order. The hardest part of the proof is the decrease when the rule Backjump is applied: $3^n - 1 - |U \cup \{D \vee L\}| < 3^n - 1 - |U|$ is decreasing since no clause is relearned (i.e., $D \vee L \notin U$). The proof is similar to the one we made for CDCL [4]. □

Theorem 4 (Correctness, Isa:full_ocdcl$_w$_stgy_no_conflicting_clause_from_init_state ✔). *An OCDCL run with a reasonable strategy starting from state $(\epsilon; N; \emptyset; \top; \top)$ terminates in a state $(M; N; U; \perp; O)$. If $O = \top$ then N is unsatisfiable. If $O \neq \top$ then $O \models N$ and for any other total model M' with $M' \models N$ it holds that $\mathrm{cost}(M') \geq \mathrm{cost}(O)$.*

The rule Improve can actually be generalized to situations where M is not total, but all literals with weights have been already set.

Improve$^+$ $(M; N; U; \top; O) \implies_{\text{OCDCL}} (M; N; U; \top; MM')$

provided $M \models N$, the model MM' is a total extension, $\text{cost}\,(M) < \text{cost}\,(O)$, and for any total extension MM'' of the trail, $\text{cost}\,(M) = \text{cost}\,(MM'')$ holds.

Lemma 5 (OCDCL with Improve$^+$, Isa:full_ocdcl$_W$_p_stgy_no_conflicting_clause_from_init_-state ☑ and Isa:wf_ocdcl$_W$_p ☑). *The rule Improve can be replaced by Improve$^+$: All previously established OCDCL properties are preserved.*

The rule ConflOpt may produce very long conflict clauses. Moreover, without conflict minimization, the subsequent backtrack is only *chronological*; i.e., only the last decision literal is removed from the trail and flipped by the rule Backtrack. Even with conflict minimization, they will contain the negation of all decision literals from the trail. It can be advantageous to generate the conflict composed of only the literals with a non-zero weight, i.e., $\neg\{L \in M \mid \text{cost}\,(L) > 0\}$ instead of $\neg M$. In this case a more general Skip rule is required, such that the eventual conflict before application of Backtrack contains one literal of highest level. As said, this is not always beneficial, e.g., the rule used in Lingeling, based on an idea by Donald Knuth (private communication between Biere and Knuth [2]), switches between the two options by taking the shortest clause.

The rules Restart and Forget can also be added to OCDCL with the same well-known implications from CDCL. For example, completeness is only preserved if Restart is applied after longer and longer intervals.

CDCL's learned-clause mechanism in the context of searching for (optimal) models does not apply with respect to partial valuations. We consider again Example 2 but drop the totality condition in Improve for the clause set $N = \{P \vee Q\}$ and cost function $\text{cost}\,(P) = 3$, $\text{cost}\,(\neg P) = \text{cost}\,(Q) = \text{cost}\,(\neg Q) = 1$. As derived in the example, an optimal-cost model based on total valuations is $[\neg P, Q]$ at overall cost 2, whereas an optimal-cost model based on partial valuations is just $[Q]$ at cost 1. The cost of undefined variables is always considered to be 0. Now the run of an optimizing branch-and-bound CDCL framework may start by deciding $[P^\dagger]$ (Step 1), detect that this is already a model for N, and learn $\neg P$ and establishes 3 as the best current bound on an optimal-cost model. After backtracking and reaching a state similar to Step 2 (without the learned clause $\neg P \vee \neg Q$), it can propagate Q with trail $[Q^{P \vee Q}, \neg P^{\neg P}]$ resulting in a model of cost 2, learning the clause $P \vee \neg Q$. The resulting clause set $\{P \vee Q, \neg P, P \vee \neg Q\}$ is unsatisfiable and hence 2 is considered to be the cost-optimal result. The issue is that although this CDCL run already stopped as soon as a partial valuation (trail) is a model for the clause set, it does not compute the optimal result with respect to partial valuations. From the existence of the model $[P]$, we cannot conclude the clause $\neg P$ to eliminate all further models containing P, because P could be undefined.

4   Formalization of OCDCL

The formalization takes a different path than described in Section 3 to reuse as much as possible from our CDCL formalization, including definitions and invariants, to avoid redefining concepts and reproving lemmas, like the consistency of the trail. Additionally, we formalize a more general approach, CDCL with branch and bound, that can be instantiated to OCDCL.

A first observation is that an OCDCL run can be seen as a series of CDCL runs, separated by applications of the rules Improve or ConflOpt. This is conceptually similar to the incremental

version of the calculus developed in our CDCL framework [4], except that CDCL runs are not complete between addition of clauses and conflicts. We can go further with the analogy: ConflOpt is similar to CDCL's ConflSat rule, except that the conflict does not appear in either N or U, but only in a richer set of clauses, like $\{\neg C. \operatorname{cost}(C) \geq \operatorname{cost}(O)\}$. Therefore, we can see OCDCL as a CDCL run extended with $N \cup \{\neg C. \operatorname{cost}(C) \geq \operatorname{cost}(O)\}$ instead of N. With this idea, the Prune rule is now another special case of ConflSat allowing us to reuse proofs. Only Improve is different and has no counterpart in CDCL: It adds new clauses because more models are now excluded.

To generalize over the optimizing CDCL, we abstract over the additional set of clauses by taking $N \cup \mathcal{T}_N(O)$ and use a predicate is_improving $M\ O'\ O$ to indicate that Improve can be applied. That way we obtain an abstract branch-and-bound calculus CDCL_{BnB} (Section 4.1). Here O is an arbitrary extra component.

As described above for OCDCL, CDCL_{BnB} can be seen as a generalized CDCL extended with the clause set $\mathcal{T}_N(O)$, interleaved with adding new clauses (Section 4.2), called CDCL_e+Improve. However, since it might not always be possible to calculate explicitly $\mathcal{T}_N(O)$, the calculus does not eagerly find conflict clauses for clauses in $\mathcal{T}_N(O)$.

The advantage of this approach is twofold: First, we formalize a more general calculus that can be instantiated with a different set of clauses to obtain a different branch-and-bound CDCL. Second, and more importantly, the resulting calculus can reuse many proofs already done on CDCL, because it is a CDCL run interleaved with adding new clauses. In particular, if applying Improve is terminating, then CDCL_{BnB} is also terminating, because CDCL is terminating.

Finally, we can instantiate CDCL_{BnB} to get a generalized version OCDCL_g: The set of clauses $\mathcal{T}_N(O)$ is instantiated by the set of clauses $\{D \mid \{\neg C \mid \operatorname{cost}(C) \geq \operatorname{cost}(O)\} \vDash D\}$ (Section 4.3), instead of $\{\neg C \mid \operatorname{cost}(C) \geq \operatorname{cost}(O)\}$, because the latter is not strong enough to express Prune. At last, we specialize OCDCL_g to get exactly the OCDCL calculus from Section 3 (Section 4.4). Fig. 1 summarizes the links between the formalized calculi.

4.1   Defining the Branch-and-Bound Calculus, CDCL_BnB

We use a similar approach to the CDCL formalization with an abstract state and selectors. Compared to CDCL, we only add an additional component representing information for the optimizing branch-and-bound part of the calculus and we do not yet specify the type of this additional component nor its content. For OCDCL, the additional component is the best model found so far by the calculus.

We parameterize the calculus by a set of clauses $\mathcal{T}_N$ that contains the conflicting clauses that can be used by the rules ConflOpt and Prune, and a predicate is_improving $M\ O'\ O$ to indicate that Improve can be applied (without any intended semantics). For OCDCL, the predicate is_improving $M\ O'\ O$ means that the current trail M is a model, O' is the information that will be stored, and O is the currently stored information. We separate M and O' because we might want to store a different information than the current trail (e.g., to remove some literals that are not required to satisfy the formulas in order to produce shorter conflicts later). $\mathcal{T}$ represents all the clauses that are entailed. We require several preconditions on the set of clauses. These preconditions are implemented in Isabelle as assumptions on the locale used to define CDCL_{BnB} that have to be discharged whenever instantiating it.

Precondition 6 (Preconditions on the clauses $\mathcal{T}_N(O)$ and on is_improving $M\ O'\ O$)**.** *We require that:*

-   *the atoms of $\mathcal{T}_N(O)$ are included in the atoms of N; i.e., no new variables are introduced.*
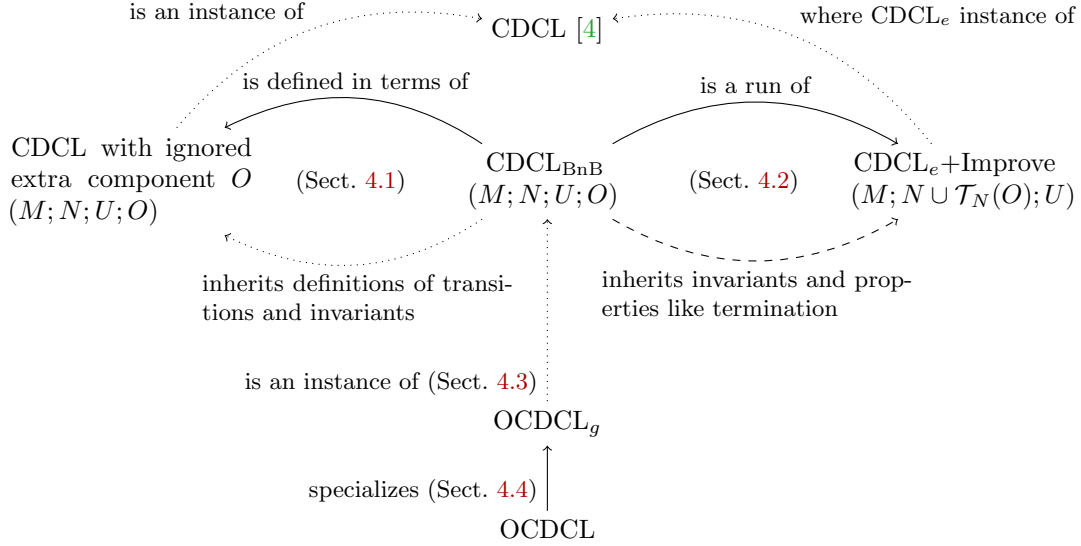
is an instance of ....→ CDCL [4] ←............ where CDCL_e instance of

is defined in terms of   is a run of

CDCL with ignored extra component O (Sect. 4.1) CDCL_{BnB} (Sect. 4.2) CDCL_e+Improve
$(M;N;U;O)$ $(M;N;U;O)$ $(M;N\cup\mathcal{T}_N(O);U)$

inherits definitions of transitions and invariants   inherits invariants and properties like termination

is an instance of (Sect. 4.3)
OCDCL_g

specializes (Sect. 4.4)
OCDCL

Figure 1: The calculi CDCL_{BnB}, OCDCL, and CDCL_e. Dotted links are locale instantiations and are for free, while dashed links indicate that some parts are for free. The arrows $X \xrightarrow{\text{does}} Y$ and $Y \xleftarrow{\text{does}} X$ both denote that "X does Y."

- the clauses of $\mathcal{T}_N(O)$ do not contain duplicate literals, because duplicates are incompatible with conflict analysis.

- if is_improving $M\ O'\ O$, then $\mathcal{T}_N(O) \subseteq \mathcal{T}_N(O')$; i.e., extending the set of clauses is monotone.

- if is_improving $M\ O'\ O$, then $\neg M \in \mathcal{T}_N(O')$; i.e., the clause $\neg M$ is entailed by $\mathcal{T}_N(O')$ and can be used as conflict clause.

The rules $\text{ConflOpt}_{\text{BnB}}$, $\text{Improve}_{\text{BnB}}$, and $\text{Backtrack}_{\text{BnB}}$ (slightly generalized compared to Backtrack from Sect. 3, as we did for CDCL) are defined as follows:

ConflOpt$_{\text{BnB}}$ $(M;N;U;\top;O) \implies_{\text{CDCL}_{\text{BnB}}} (M;N;U;\neg M;O)$

provided $\neg M \in \mathcal{T}_N(O)$.

Improve$_{\text{BnB}}{}^+$ $(M;N;U;\top;O) \implies_{\text{CDCL}_{\text{BnB}}} (M;N;U;\neg M;O')$.

provided is_improving $M\ O'\ O$ holds.

Backtrack$_{\text{BnB}}$ $(M_2K^\dagger M_1;N;U;D\vee L;O) \implies_{\text{CDCL}_{\text{BnB}}} (L^{D'\vee L}M_1;N;U\cup\{D'\vee L\};\top;O)$

provided L is of the maximum level, $D' \subseteq D$, $N\cup U\cup\mathcal{T}_N(O) \vDash D'\vee L$, D' and K are of the same level i, and i is strictly less than the maximum level.

We can simply instantiate in the CDCL formalization the states with weights and reuse the previous definitions, properties, and invariants by interpreting OCDCL states $(M;N;U;D;O)$ to CDCL states $(M;N;U;D)$ (i.e., the selectors simply ignore the additional component). For example, we can reuse the Decide rule and the proofs on it.

Compared with the rule from Section 3, we make it possible to express conflict-clause minimization: Instead of $D\vee L$, a clause $D'\vee L$ is learned such that $D' \subseteq D$ and $N\cup U\cup\mathcal{T}_N(O) \vDash$

$D' \lor L$. While all other CDCL rules are reused, the Backtrack rule is not reused for OCDCL: If we had reused Backtrack from CDCL, only the weaker entailment $N \cup U \vDash D' \lor L$ would be used. The weaker version is sufficient to express conflict minimization as implemented in most SAT solvers [24], but the former is stronger and makes it possible for example to remove decision literals without cost from D.

We use the Improve$^+$ rule instead of the Improve rule, because the latter is a special case of the former. The strategy favors ConflictSat and Propagate over all other rules. We do not need to favor ConflictOpt over the other rules for correctness, although doing so helps in an implementation, by providing shorter conflicts.

4.2   Embedding into CDCL

In order to reuse the proofs that were done previously about CDCL, CDCL$_{\mathrm{BnB}}$ is seen as a special instance of CDCL by mapping the states $(M; N; U; D; O)$ to the CDCL state $(M; N \cup \mathcal{T}_N (O); U; D)$ (and not $(M; N; U; D)$ as in the previous section). To distinguish between both, we will refer to the CDCL with the *enriched* set of clauses as CDCL$_e$.

In Isabelle, the most direct solution would be to instantiate the CDCL calculus with a selector returning $N \cup \mathcal{T}_N (O)$ instead of just N. This, however, would lead to a loop in the locale hierarchy which is not allowed. Instead, we add an explicit conversion via the function state$_e$_of from $(M; N; U; D; O)$ to $(M; N \cup \mathcal{T}_N (O); U; D)$ and consider normal CDCL on tuples of the latter; i.e., we map $S \Longrightarrow_{\mathrm{CDCL_{BnB}}} T$ to state$_e$_of$(S) \Longrightarrow_{\mathrm{CDCL}_e}$ state$_e$_of(T).

Except for the Improve rule, every OCDCL rule can be mapped to a CDCL$_e$ rule: The ConflictOpt$_{\mathrm{BnB}}$ rule corresponds to the ConflictSat rule (because it can pick a clause from $\mathcal{T}_N (O)$) and Backtrack$_{\mathrm{BnB}}$ is mapped to CDCL$_e$'s Backtrack. The Improve rule has no counterpart and requires new proofs, but adding clauses is compatible with the CDCL$_e$ invariants.

In our CDCL formalization, we distinguish structural properties that are invariants of the state (e.g., consistency of the trail) from the strategy-specific properties that make sure CDCL$_{\mathrm{BnB}}$ does not get stuck. The latter makes sure that the conflict clause contains a literal of highest level, which does not necessarily hold for the unrestricted CDCL$_e$: The clause $\bot$ might be in $\mathcal{T}_N (O)$ and can be picked by ConflictSat leading to a state where CDCL$_e$ is stuck. However, we can easily prove that the strategy-specific invariants hold for CDCL$_{\mathrm{BnB}}$ and we can reuse the CDCL proof we have already done for most transitions. To reuse some proofs on CDCL's Backtrack, we slightly generalized some proofs by dropping the assumption $N \cup U \vDash D' \lor L'$ when not required. This is the only change made to the formalization of CDCL.

Not all transitions of CDCL$_e$ can be taken by OCDCL: Propagating clauses in $\mathcal{T}_N (O)$ is not possible and would violate invariants if propagation can be done out-of-order (CDCL can be extended to support that as described by Möhle and Biere [17] but that would require some changes in our calculus). The structural properties are sufficient to prove that OCDCL is terminating as long as Improve$^+$ can be applied only finitely often, because CDCL$_e$ is terminating. At this level, Improve$^+$ is too abstract to prove that CDCL$_{\mathrm{BnB}}$ terminates. With the additional assumptions that Improve can always be applied when the trail is a total model satisfying the clauses, we show that the final set of clauses is unsatisfiable. An example where Improve cannot always be applied is the search for a model that fulfills some properties: If such a model is found, we stop and there is no need to search for further models.

Theorem 7 (CDCL$_{\mathrm{BnB}}$ Termination, Isa:wf_cdcl_bnb ☑). *If Improve is well founded, then CDCL$_{BnB}$ is also well founded.*

Theorem 8 (CDCL$_{\mathrm{BnB}}$ Termination, Isa:full_cdcl_bnb_stgy_no_conflicting_clss_unsat ☑ and

Isa:no_step_cdcl_bnb_stgy_empty_conflict ☑). *If Improve can be applied when all literals are set in the trail M and $M \vDash N \cup U$, then a $CDCL_{BnB}$ run terminates in a state $(M'; N; U'; \bot; O)$ and the resulting clause set $N \cup U'$ is unsatisfiable.*

4.3   Instantiating with Weights, OCDCL$_g$

Now having developed an abstract CDCL$_{BnB}$ framework, we can instantiate it with weights and save the best current found model in O. We assume the existence of a cost function that is monotone with respect to inclusion:

> **locale** cost =
> **fixes** cost :: *'v literal multiset* $\Rightarrow$ *'c* :: *linorder*
> **assumes** $\forall C$. consistent_interp $B \wedge$ distinct_mset $B \wedge A \subseteq B \longrightarrow \mathrm{cost}\,(A) \leq \mathrm{cost}\,(B)$

We assume that the function cost is monotone with respect to inclusion for consistent duplicate-free models. This is natural for trails, which by construction do not contain duplicates. The monotonicity is less restrictive than the condition from Sect. 3, which mandates that the cost is a sum over the literals. We instantiate the set of clauses and the predicate indicating if the current model is improving with

$$
\begin{aligned}
\mathcal{T}_N(O) = \quad & \{C \mid \mathrm{atom}(C) \subseteq \mathrm{atom}(N) \\
& \wedge\, C \text{ neither is a tautology nor contains duplicates} \\
& \wedge\, \{\neg D \mid \mathrm{cost}\,(D) \geq \mathrm{cost}\,(O)\} \vDash C\} \\
\text{is\_improving } M\, O'\, O \leftrightarrow \quad & O' \text{ is a total extension of } M,\ M \vDash N, \\
& \text{any total extension of } M \text{ has the same cost, and} \\
& \mathrm{cost}\,(M) < \mathrm{cost}\,(O)
\end{aligned}
$$

and then discharge the assumptions given in Precondition 6 to be able to instantiate the locale. We call OCDCL$_g$ the version of CDCL$_{BnB}$ with this instantiation.

As the locale was instantiated, OCDCL$_g$'s invariants are exactly the ones from CDCL$_{BnB}$. For termination, we only have to prove that Improve$^+$ terminates, which makes it possible to use Theorem 7 to prove termination. The key property for the correctness of OCDCL$_g$ is the following:

Lemma 9 (Cheaper Models stay Models, Isa:entails_too_heavy_clauses_too_heavy_clauses ☑). *If I is a total consistent model of N, then either $\mathrm{cost}\,(I) \geq \mathrm{cost}\,(O)$ or I is a total model of $N \cup \mathcal{T}_N(O)$.*

Isabelle Proof. *Assume $\mathrm{cost}\,(I) < \mathrm{cost}\,(O)$. First, we have show that $I \vDash \{\neg C \mid \mathrm{cost}\,(C) \geq \mathrm{cost}\,(O)\}$. Let D be a clause of $\{\neg C \mid \mathrm{cost}\,(C) \geq \mathrm{cost}\,(O)\}$. D is not a subset of I (by monotonicity of cost, $\mathrm{cost}\,(I) \geq \mathrm{cost}\,(D)$). Therefore, there is at least a literal L in D such that $\neg L$ in I and $I \vDash D$.*

By transitivity, since I is total, I is also a model of $\mathcal{T}_N(O)$ and therefore of $N \cup \mathcal{T}_N(O)$. □

This is the proof that breaks if partial models are allowed: Literals of the clause D might not be defined in I. The fact that the proof does not work for partial models does not imply that the theorem is wrong, but it gave us the idea of the counterexample described in Section 3.

Some additional proofs are required to specify the content of the component O. For example, the sequence of literals O is always a total consistent model of N. This property cannot be inherited from the correctness of CDCL, because it does not express any property of the component O.

4.4 OCDCL

Finally, we can restrict the calculus to precisely the rules expressed in Section 3. We define two calculi: one with only the rule Improve, and the other with both Improve$^+$ and Prune. In both cases, the rule ConflictOpt is only applied when $\mathrm{cost}\,(M) > \mathrm{cost}\,(O)$ and is therefore a special case of ConflictOpt$_{\mathrm{BnB}}$. The Prune rule is also seen as a special case of ConflictOpt$_{\mathrm{BnB}}$. Therefore, every transition of OCDCL is also a transition of OCDCL$_g$. Moreover, since final states of both calculi are the same, a complete run of OCDCL is also a complete run of OCDCL$_g$. Ultimately, the correctness theorem is inherited.

Overall, the full formalization was straightforward to do, once we got the idea how to see OCDCL as a special case of a sequence of CDCL runs without strategy. Formalizing a changing target is different than a fixed version of a calculus, as we did previously for CDCL [4]. We had to change our formalization several times to take into account additional rules. The set of clauses $\{\neg C \mid \mathrm{cost}\,(C) \geq \mathrm{cost}\,(O)\}$ as $\mathcal{T}_N(O)$ is sufficient for Improve$^+$. However, when we added the Prune rule, we had to change $\mathcal{T}_N(O)$ to $\{D \mid \{\neg C \mid \mathrm{cost}\,(C) \geq \mathrm{cost}\,(O)\} \vDash D\}$. This forced us to change the formalization a little, but our approach is robust enough that except for a few changes in intermediate proof steps, nothing else had to be changed.

5 Solving Further Optimization Problems

In this section, we show three applications of our framework. First, we verify an encoding that reduces the search for partial models to the search for total models (Section 5.1). Second, CDCL can be used to solve MAX-SAT (Section 5.2) by adding new variables to the problem. Instead of considering the weight of literals, in MAX-SAT the weight of clauses is considered. Finally, we apply our CDCL$_{\mathrm{BnB}}$ framework to another problem, covering models (Section 5.3). All extensions are verified using Isabelle.

5.1 Optimal Partial Valuations

To reduce the search from optimal partial valuations to optimal total valuations, we use the dual rail encoding [6, 21]. For every proposition variable P, it creates two variables P^1 and P^0 indicating that P is defined positively or negatively. If both are negative, then P is unset. Adding the clause $\neg P^1 \vee \neg P^0$ ensures that P is not defined both positively and negatively at the same time. The resulting set is called penc$\,(N)$.

More precisely, the encoding penc is defined on literals by $\mathrm{penc}(P) := (P^1)$, $\mathrm{penc}(\neg P) := (P^0)$, and lifted to clauses and clause sets by $\mathrm{penc}(L_1 \vee \cdots \vee L_n) := \mathrm{penc}(L_1) \vee \cdots \vee \mathrm{penc}(L_n)$, and, $\mathrm{penc}(C_1 \wedge \cdots \wedge C_m) := \mathrm{penc}(C_1) \wedge \cdots \wedge \mathrm{penc}(C_m)$. We call Σ' the set of all new atoms.

The important property of this encoding is that $\neg P^1$ does not entail P^0: If P is not positive, it does not have to be negative either.

Given the encoding penc(N) of N the cost function is extended to a valuation $\mathcal{A}'$ on $\Sigma \cup \Sigma'$ by $\mathrm{cost}'(\mathcal{A}') = \mathrm{cost}\left(\left\{L \mid L^1 \in \mathcal{A}'\right\} \cup \left\{-L \mid L^0 \in \mathcal{A}'\right\}\right).$

Let $\mathrm{pdec}\,(\mathcal{A}) : P \mapsto \begin{cases} 1 & \text{if } \mathcal{A}(P^1) = 1 \\ 0 & \text{if } \mathcal{A}(P^0) = 1, \\ \text{unset} & \text{otherwise} \end{cases}$ be a function that transforms a total model of penc$\,(N)$ into a (possibly partial) model of N and $\mathrm{pdec}^-\,(\mathcal{A})$ does the opposite transformation, with $\mathrm{pdec}^-\,(\mathcal{A})(P^1) = 1$ if $\mathcal{A}(P) = 1$, $\mathrm{pdec}^-\,(\mathcal{A})(P^1) = 0$ if $\mathcal{A}(P) = 0$, $\mathrm{pdec}^-\,(\mathcal{A})(P^0) = 1$ if $\mathcal{A}(P) = 0$, $\mathrm{pdec}^-\,(\mathcal{A})(P^0) = 0$ if $\mathcal{A}(P) = 1$, unset otherwise.

Lemma 10 (Partial and Total Valuations Coincide up to penc, Isa:penc_ent_postp ☑ and Isa:penc_ent_upostp ☑). *Let N be a clause set.*

  1. *If $\mathcal{A} \models N$ for a partial model $\mathcal{A}$ then $\mathrm{pdec}^-(\mathcal{A}) \models \mathrm{penc}(N)$;*

  2. *If $\mathcal{A}' \models \mathrm{penc}(N)$ for a total model $\mathcal{A}'$ on N, then $\mathrm{pdec}(\mathcal{A}') \models N$.*

Lemma 11 (penc Preserves Cost Optimal Models, Isa:full_encoding_OCDCL_correctness ☑). *Let N be a clause set and cost a cost function over literals from N. If $\mathcal{A}'$ is a cost-optimal total model for $\mathrm{penc}(N)$ over cost', resulting in $\mathrm{cost}'(\mathcal{A}') = m$, then the partial model $\mathrm{pdec}(\mathcal{A}')$ is cost-optimal for N and $\mathrm{cost}(\mathrm{pdec}(\mathcal{A}')) = m$.*

Proof. *Assume there is a partial model $\mathcal{A}$ for N such that $\mathrm{cost}(\mathcal{A}) = k$. The model $\mathrm{pdec}^-(\mathcal{A})$ is another model of N. As $\mathcal{A}'$ is cost optimal, $\mathrm{cost}'(\mathrm{pdec}^-(\mathcal{A})) \geq \mathrm{cost}'(\mathcal{A}')$. Moreover, $\mathrm{cost}'(\mathrm{pdec}(\mathcal{A}')) = \mathrm{cost}(\mathcal{A}')$ and $\mathrm{cost}'(\mathrm{pdec}(\mathcal{A})) = \mathrm{cost}(\mathcal{A})$. Ultimately $\mathcal{A}$ is not better than $\mathcal{A}'$ and $\mathcal{A}'$ has cost m.* □

Lemma 12 (OCDCL on the Encoding Isa:full_encoding_OCDCL_complexity ☑). *Consider an OCDCL run on $N' = \mathrm{penc}(N)$. At most $4^{\mathrm{atom}(N)}$ conflicts will be found.*

Proof. *The OCDCL run starts in a state $(\epsilon; N'; \emptyset; D; \top)$ and terminates in a state $(M; N'; U; D; O)$. The run learns the same clauses as a CDCL run (without strategy) going from $(\epsilon; N' \cup \mathcal{T}_N(O); \emptyset; D)$ to $(M; N \cup \mathcal{T}_N(O); U; D)$.*
CDCL can learn at most $2^{\mathrm{atom}(N')}$ [8, Sect. 3.2.3], hence the bound is $4^{\mathrm{atom}(N)}$. □

Formalization of the Partial Encoding. In Isabelle, total valuations are defined by Herbrand interpretations, i.e., a set of all true atoms (all others being implicitly false) [22], but we work on partial models for CDCL, because they are similar to a trail. To reason on total models, we have defined a predicate to indicate whether a model is total or not. We distinguish between literals that can have a nonzero weight $\Delta\Sigma$ from the others ($\Sigma \setminus \Delta\Sigma$) because they can be left unchanged by the encoding.

The proofs are very similar to the proofs described in Section 5.1. We instantiate the OCDCL calculus with the cost' function:

    interpretation OCDCL where cost = cost'

We only have to discharge the proof obligation from the locale cost (Sect. 4.3) that the cost function cost' is monotone. Finally, we can prove the correctness Lemma 11. The formalization is 800 lines long.

In Isabelle, for the proof of Lemma 12, we have verified first a more general theorem about the complexity of $\mathrm{CDCL_{BnB}}$, Lemma 13.

Lemma 13 ($\mathrm{CDCL_{BnB}}$ Complexity). *Consider a $\mathrm{CDCL_{BnB}}$ starting from $(\epsilon; N; \emptyset; D; O)$. At most $2^{\mathrm{atom}(N)}$ conflicts will be found.*

The proof of Lemma 13 is similar to the one described for Lemma 12. Then Lemma 12 is a simple corollary by seeing that $\mathrm{atom}(\mathrm{penc}(N)) = 2 \times \mathrm{atom}(N)$ yielding the $4^{\mathrm{atom}(N)}$ bound. We have made a similar construct to the one described here for an ODPLL calculus (with a $\mathrm{DPLL_{BnB}}$ calculus). For this calculus on the encoding we proved the stronger bound $3^{\mathrm{atom}(N)}$, but we have not managed to adapt the proof to the OCDCL case.

5.2 MAX-SAT Encoded as an OCDCL Problem

The maximum satisfiability problem (MAX-SAT) is another well-known optimization problem [14]. It consists of two clause sets N_H (hard constraints, mandatory to satisfy) and N_S (soft constraints, optional to satisfy). The set N_S comes with a cost function for clauses that are not satisfied. The aim is to find a total model with minimal cost.

Theorem 14 (Reduction of MAX-SAT to OCDCL, Isa:partial_max_sat_is_weight_sat ☑). *Let* $(N_H, N_S, \mathrm{cost})$ *be a MAX-SAT problem and let* $\mathrm{active} : N_S \to \Sigma'$ *be an injective and surjective mapping for a set* Σ' *of fresh propositional variables that assigns to each soft constraint an activation variable.*

Let I be a solution to the OPT-SAT problem $N = N_H \cup \{\mathrm{active}(C) \vee C \mid C \in N_S\}$ with the cost function $\mathrm{cost}'(L) = \mathrm{cost}(C)$ if $\mathrm{active}(C) = L$ for some $C \in N_S$ and $\mathrm{cost}'(L) = 0$ otherwise.

If there is no model I of N, the MAX-SAT problem has no solution. Otherwise, I without the additional atoms from Σ' is an optimal solution to the MAX-SAT problem.

Isabelle Proof. N_H *is satisfiable if and only if MAX-SAT has a solution. Therefore, if there is no model I of N, then N_H is unsatisfiable.*

Let $I' = \{L \mid L \in I \wedge \mathrm{atom}(L) \notin \Sigma'\}$. Let J be any other model of $(N_H \cup N_S)$ and J' its total extension to Σ': $J' = J \cup \{\mathrm{active}(C) \mid C \in N_S \wedge J \nvDash C\} \cup \{\neg\mathrm{active}(C) \mid C \in N_S \wedge J \vDash C\}$.

J' satisfies N_H and is a total consistent model of N. Hence, $\mathrm{cost}'(J') \geq \mathrm{cost}(I)$, because I is the optimal model of N. By definition, $\mathrm{cost}'(I) = \mathrm{cost}(I')$ and $\mathrm{cost}'(J) = \mathrm{cost}(J')$. Therefore I is an optimal MAX-SAT model. □

The problem of this MAX-SAT encoding is that, in practice, it reduces the number of unit propagations that can be done, in particular if there are many soft constraints.

5.3 Another Instantiation of CDCL$_{\mathrm{BnB}}$: Model Covering

Our third example demonstrates that our framework can be applied beyond OCDCL. We consider the calculation of covering models. Again this is motivated by a product configuration scenario where a propositional variable encodes the containment of a certain component in a product. For product testing, finding a bucket of products is typically required such that every component occurs at least once in the bucket. Translated into propositional logic: given a set N of clauses, we search for a set of models $\mathcal{M}$ such that for each propositional variable P occurring in N, $M \vDash P$ for at least one $M \in \mathcal{M}$ or there is no model of N such that P holds.

In order to solve the model-covering problem, we define the domination relation: A model is *dominated* if there is another model that contains the same true variables and some others. More formally, if I and J are total models for N, then I is *dominated* by J if $\{P \mid I \vDash P\} \subseteq \{Q \mid J \vDash Q\}$. In the context of Horn clauses, this is written as $I < J$ (Horn clauses have a *minimal* model unlike clauses in general). If a total model is dominated by a model already contained in $\mathcal{M}$, then it is not required in the set of covering models. The extension to CDCL are the following two additional rules:

ConflCM $(M; N; U; \top; \mathcal{M}) \implies_{\mathrm{MCCDCL}} (M; N; U; \neg M; \mathcal{M})$

   if for all total extensions $M'M$ with $M'M \models N$, there is an $I \in \mathcal{M}$ which dominates $M'M$.

Add $(M; N; U; \top; \mathcal{M}) \implies_{\mathrm{MCCDCL}} (M; N; U; \top; \mathcal{M} \cup \{M\})$

   if $M \models N$, all literals from N are defined in M, and M is not dominated by a model in $\mathcal{M}$.

The full calculus called MCCDCL (model covering CDCL) does not necessarily compute a minimal set of covering models. Minimization is a classical NP-complete problem [12] and

can then be done in a second step. The minimal model covering can be computed by creating another CDCL extension, where the set $\mathcal{M}$ is explicitly added as a component to a state and used for a branch-and-bound optimization approach, similar to OCDCL [16].

In the formalization of MCCDCL, we instantiate CDCL_{BnB} with:

$$
\begin{aligned}
\mathcal{T}_N(\mathcal{M}) = \quad &\{C.\ \text{atom}(C) \subseteq \text{atom}(N) \\
&\wedge\ C \text{ is not a tautology nor contains duplicates} \\
&\wedge\ \{\neg D.\ \text{is\_dominating } \mathcal{M}\ D,\ \text{total}\} \cup N \vDash C\} \\
\text{is\_improving } M\ M'\ \mathcal{M} \leftrightarrow \quad &M = M' \text{ and } M \vDash N \\
&\text{and } M \text{ is not dominated by } \mathcal{M} \\
&\text{and } M \text{ is consistent, total, duplicate free}
\end{aligned}
$$

Compared with OCDCL, $\mathcal{T}_N(\emptyset)$ is never empty, because it contains all clauses entailed by N, including the clauses of N. This is another reason why CDCL_e cannot run with the reasonable strategy: If N is unsatisfiable, we would have to pick the conflict clause $\bot$ immediately, which is exactly the problem we are solving with CDCL.

Theorem 15 (Correctness Isa:cdclcm_correctness ✔). *If the clauses in N do not contain duplicated literals, then an $MCCDCL$ starting from $(\epsilon, N, \emptyset, \top, \emptyset)$ ends in a state $(\epsilon, N, U, \bot, \mathcal{M})$, and for every variable P in N, there is a model M of N, $M \in \mathcal{M}$, where $M \vDash P$, or there is no model satisfying both P and N.*

The proof involves a theorem similar to Lemma 9: Every model of the initial clause set N is either dominated by a model in $\mathcal{M}$ or is still a model of $N \cup \mathcal{T}_N(\mathcal{M})$.

6 Related Work and Conclusion

There are several other formalizations of CDCL, as discussed in our article [4, Section 6], but we are not aware of any formalization of an optimizing CDCL calculus, or more generally, a formalization used as a starting point to formalize variants of CDCL.

There are several variants of optimizing SAT. Our OCDCL has similarities with the calculus by Larrosa et al. [13]. Our calculus is slightly more general on the optimizing part due to the inclusion of the rule Improve$^+$. On the CDCL part our calculus is more concrete, or, the other way round, theirs is more abstract. For example, the computation of a first unique implication point is built into our calculus where they only require necessary semantic conditions on learned clauses. Our rule Prune rule can be simulated by their Learn rule: $\neg M \vee c \geq \text{cost}(O)$ is entailed by the clauses (depending on the cost function). In summary, our calculus is closer to real-world implementations of (O)CDCL.

A related problem to finding the minimum partial model is called minimum-weight propositional satisfiability by Sebastiani et al. [23]. However, negative literals do not cost anything: This means that the opposite of L is $\neg L$ (because $\neg L$ and L undefined have the same weight).

Liberatore has developed a variant of DPLL to solve this problem [15]. Each time a variable is decided, it is first set to true, then set to false. Moreover, if the current model is larger than a given bound, then the search stops exploring the current branch. When a new better model is found, the search is restarted with the new lower bound. A version lifted to CDCL has been implemented in zChaff [9] to solve MAX-SAT. Although Liberatore's method can return partial models, it is still an Herbrand model: It is entirely given by the set of all true atoms. Therefore, the method actually builds total models.

We have presented here a variant of CDCL to find optimal models and used the dual rail encoding to reduce the search of optimal models with respect to partial valuations to the search of optimal models with respect to total valuations. Both have been formalized using the proof assistant Isabelle/HOL. This formalization fits nicely into the framework that was previously developed and the abstraction in Isabelle simplifies reuse and studying variants and extensions.

We started our encoding for cost-minimal models with respect to partial valuations by introducing three extra variables for each variable, where, compared to the dual rail encoding, the third extra variable explicitly modeled whether a variable is defined or undefined [27]. We performed the content of Section 5.1 with this encoding and only afterwards were pointed by a reviewer to the dual rail encoding. It took us half a day to redo the overall formalization. For us this is another example that the reuse of formalizations can work. This is further demonstrated by the application of the OCDCL results to MAX-SAT and the reuse of the formalization framework to verify the model covering calculus MCCDCL. Minimization of the model covering set computed by MCCDCL can also be solved by an afterwards application of a CDCL calculus with branch-and-bound [16], and would probably fit in our framework.

One idea for future work would be to refine OCDCL to executable code as we have done for CDCL [4]. However, we have started to work on a different extension, CDCL($\mathcal{T}$) [18]. On an abstract level, CDCL$_{\mathrm{BnB}}$ is close to an incremental version of CDCL($\mathcal{T}$), the calculus used in several modern SMT solvers. The main difference is that the conflicts generated are not the negation of the trail, but clauses implied by the theory. The theory of linear integer arithmetic (LA) has already been verified in Isabelle/HOL by Thiemann [5,25], so proving correctness of CDCL(LA) does not need a from-scratch new effort.

References

[1] Becker, H., Bentkamp, A., Blanchette, J.C., Fleury, M., From, A.H., Jensen, A.B., Lammich, P., Larsen, J.B., Michaelis, J., Nipkow, T., Peltier, N., Popescu, A., Robillard, S., Schlichtkrull, A., Tourret, S., Traytel, D., Villadsen, J., Petar, V.: IsaFoL: Isabelle Formalization of Logic, https://bitbucket.org/isafol/isafol/

[2] Biere, A.: Splatz, Lingeling, Plingeling, Treengeling, YalSAT Entering the SAT Competition 2016. In: Balyo, T., Heule, M.J.H., Järvisalo, M. (eds.) SAT Competition 2016. Department of Computer Science Series of Publications B, vol. B-2016-1, pp. 44–45. University of Helsinki (2016), http://fmv.jku.at/papers/Biere-SAT-Competition-2016-solvers.pdf

[3] Biere, A., Heule, M.J.H., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press (2009)

[4] Blanchette, J.C., Fleury, M., Lammich, P., Weidenbach, C.: A verified SAT solver framework with learn, forget, restart, and incrementality. J. Autom. Reasoning **61**(1-4), 333–365 (Mar 2018). doi:10.1007/s10817-018-9455-7

[5] Bottesch, R., Haslbeck, M.W., Thiemann, R.: Verifying an incremental theory solver for linear arithmetic in isabelle/HOL. In: FroCoS 2019. LNCS, vol. 11715, pp. 223–239. Springer (2019). doi:10.1007/978-3-030-29007-8_13

[6] Bryant, R.E., Beatty, D., Brace, K., Cho, K., Sheffler, T.: COSMOS: A compiled simulator for MOS circuits. In: O'Neill, A., Thomas, D. (eds.) DAC'87. pp. 9–16. ACM Press (1987). doi:10.1145/37888.37890

[7] Church, A.: A formulation of the simple theory of types. J. symb. log. **5**(02), 56–68 (Jun 1940). doi:10.2307/2266170

[8] Fleury, M.: Formalization of Logical Calculi in Isabelle/HOL. Ph.D. thesis, Universität des Saarlandes (2020)

[9] Giunchiglia, E., Maratea, M.: Solving optimization problems with DLL. In: Brewka, G., Coradeschi, S., Perini, A., Traverso, P. (eds.) ECAI 2006. Frontiers in Artificial Intelligence and Applications, vol. 141, pp. 377–381. IOS Press (2006)

[10] Gordon, M.J., Milner, A.J., Wadsworth, C.P.: Edinburgh LCF, LNCS, vol. 78. Springer Berlin Heidelberg (1979). doi:10.1007/3-540-09724-4

[11] Kammüller, F., Wenzel, M., Paulson, L.C.: Locales—A sectioning concept for Isabelle. In: Bertot, Y., Dowek, G., Hirschowitz, A., Paulin, C., Théry, L. (eds.) TPHOLs'99. LNCS, vol. 1690, pp. 149–166. Springer (1999). doi:10.1007/3-540-48256-3_11

[12] Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA. pp. 85–103. The IBM Research Symposia Series, Plenum Press, New York (1972), http://www.cs.berkeley.edu/%7Eluca/cs172/karp.pdf

[13] Larrosa, J., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E.: A framework for certified Boolean branch-and-bound optimization. J. Autom. Reasoning **46**(1), 81–102 (Apr 2010). doi:10.1007/s10817-010-9176-z

[14] Li, C., Manyà, F.: MaxSAT, hard and soft constraints. In: Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, pp. 613–631. IOS Press (2009). doi:10.3233/978-1-58603-929-5-613

[15] Liberatore, P.: Algorithms and experiments on finding minimal models. Tech. Rep. 09-99, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza" (1999), http://www.dis.uniroma1.it/%7Eliberato/papers/libe-99.ps.gz

[16] Manquinho, V.M., Marques-Silva, J.P.: Satisfiability-based algorithms for pseudo-boolean optimization using gomory cuts and search restarts. In: ICTAI'05. pp. 150–155. IEEE (2005). doi:10.1109/ictai.2005.113

[17] Möhle, S., Biere, A.: Backing backtracking. In: Janota, M., Lynce, I. (eds.) SAT 2019. LNCS, vol. 11628, pp. 250–266. Springer (2019). doi:10.1007/978-3-030-24258-9_18

[18] Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). JACM **53**(6), 937–977 (Nov 2006). doi:10.1145/1217856.1217859

[19] Nipkow, T., Klein, G.: Concrete Semantics. Springer International Publishing (2014). doi:10.1007/978-3-319-10542-0

[20] Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer Berlin Heidelberg (2002). doi:10.1007/3-540-45949-9

[21] Palopoli, L., Pirri, F., Pizzuti, C.: Algorithms for selective enumeration of prime implicants. Artif. Intell. **111**(1-2), 41–72 (Jul 1999). doi:10.1016/s0004-3702(99)00035-1

[22] Schlichtkrull, A.: Formalization of the resolution calculus for first-order logic. In: Blanchette, J.C., Merz, S. (eds.) ITP 2016. LNCS, vol. 9807, pp. 341–357. Springer (2016). doi:10.1007/978-3-319-43144-4_21

[23] Sebastiani, R., Giorgini, P., Mylopoulos, J.: Simple and minimum-cost satisfiability for goal models. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 20–35. Springer (2004). doi:10.1007/978-3-540-25975-6_4

[24] Sörensson, N., Biere, A.: Minimizing learned clauses. In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 237–243. Springer (2009). doi:10.1007/978-3-642-02777-2_23

[25] Thiemann, R.: Extending a verified simplex algorithm. In: Barthe, G., Korovin, K., Schulz, S., Suda, M., Sutcliffe, G., Veanes, M. (eds.) LPAR-22. Kalpa Publications in Computing, vol. 9, pp. 37–48. EasyChair (2018). doi:10.29007/5vlq

[26] Weidenbach, C.: Automated reasoning building blocks. In: Correct System Design—Symposium in Honor of Ernst-Rüdiger Olderog on the Occasion of His 60th Birthday, Oldenburg, Germany, September 8-9, 2015. Proceedings. LNCS, vol. 9360, pp. 172–188. Springer (2015). doi:10.1007/978-3-319-23506-6_12

[27] Zimmer, D.: Computing Partial Cost-Optimal Satisfying Assignments for Propositional Formulas. Bachelor thesis, Universität des Saarlandes (Sep 2019)