*Proof.* (Idea) Firstly, all unit clauses can be eliminated by recursively resolving away the respective literals, following the algorithm of Proposition 2.14.7. For a clause set $N$ containing only clauses of length two a directed graph is constructed. The nodes are the propositional literals from $N$. For each clause $L \vee K \in N$, the graph contains the two directed edges $(\text{comp}(L), K)$ and $(\text{comp}(K), L)$. Then $N$ is unsatisfiable iff there is a cycle in the graph containing two nodes $L$, $\text{comp}(L)$. This can be decided in time at most quadratic in $N$. □

$$\begin{tikzcd}
A \arrow[d, shift right] \arrow[dr, shift left] & \neg A \arrow[d, shift right] \arrow[dl, shift right] \\
B \arrow[u, shift right] \arrow[ur, shift right] & \neg B \arrow[u, shift right] \arrow[ul, shift left]
\end{tikzcd}$$

Interestingly, 2-SAT constitutes the border to NP-completeness, because 3-SAT is already NP-complete. This can be seen by reducing any clause set to a satisfiability equivalent 3-SAT clause set via the following transformation. For any clause

$$L_1 \vee \ldots \vee L_n$$

consisting of more than three literals ($n > 3$) replace the clause by the clauses

$$\begin{gathered} L_1 \vee \ldots \vee L_{\lfloor n/2 \rfloor} \vee P \\ L_{\lfloor n/2 \rfloor + 1} \vee \ldots \vee L_n \vee \neg P \end{gathered}$$

where $P$ is a fresh propositional variable. Obviously, $L_1 \vee \ldots \vee L_n$ is satisfiable iff $L_1 \vee \ldots \vee L_{\lfloor n/2 \rfloor} \vee P$, $L_{\lfloor n/2 \rfloor + 1} \vee \ldots \vee L_n \vee \neg P$ are.

**Proposition 2.14.9.** 3-SAT is NP-complete.

## 2.15 CDCL Extensions

In this section I extend the basic CDCL calculus from Section 2.9 in three different directions. Firstly, by adding two new calculus rules CDCL turns into the calculus OCDCL for finding models of minimal cost. Secondly, the OCDCL calculus can be turned into a calculus for Max-SAT. The Max-SAT problem consists of finding an assignment that falsifies a minimal number of clauses. Thirdly, CDCL is extended for finding a minimal covering set of models such that each variable is true in at least one model. All problems are optimization problems that have a number of applications in practice. Of course, they are just my personal choice but they nicely show how the CDCL calculus as well as a SAT problem can be modified to solve problems beyond pure satisfiability.

For an application example, if a computer program is represented on some level as a SAT problem, then a minimal model can represent a shortest path to a bug by adding some failure condition to the SAT problem. The respective Max-SAT problem can point to a minimal change in order to fix the bug. A covering set can represent all different inputs needed to explore all branches of the program.

All CDCL extensions utilize the Branch-And-Bound principle. First, a not necessarily optimal solution is derived and then this solution is used as a basis for improvement and for cutting off states that cannot lead to a better solution anymore as early as possible. In addition, the minimal covering set variant of CDCL also relies on a preprocessing of the input SAT problem, where for each propositional variable a separate clause set is created.

Cutting off states of the CDCL calculus means extra conflicts. These extra conflicts are not of a logical nature but result from an additional Conflict rule justified by the properties of the cost function. Therefore, the two new variants of this section are of a different nature than the original CDCL calculus, because a learned clause is no longer always a logical consequence of the input clause set.

## 2.15.1 Computing Cost Optimal Models

A OCDCL problem state becomes a six-tuple $(M; N; U; k; C; O)$ where the first five components and the respective notation and notions are inherited from the standard CDCL calculus, Section 2.9. The sixth component $O$ represents the best model found so far. In addition, I assume a positive cost function cost on literals with $\text{cost}(L) \geq 0$ for all literals $L$. The function is naturally extended to sequences and sets of literals by computing the sum of the elements.

Similar to the CDCL calculus, duplicate occurrences of literals in clauses are always silently removed. This applies to the input clause set and the below rule Resolve. The following states can be distinguished:

| | |
|---|---|
| $(\epsilon; N; \emptyset; 0; \top; \epsilon)$ | is the start state for some clause set $N$ |
| $(M; N; U; k; \bot; O)$ | is the final state, where $N$ has no model if $O = \epsilon$, or otherwise $O$ is a cost optimal model |
| $(M; N; U; k; \top; O)$ | is an intermediate model search state |
| $(M; N; U; k; D; O)$ | is a backtracking state if $D \notin \{\top, \bot\}$ |

The OCDCL rules are

**Propagate** $(M; N; U; k; \top; O) \Rightarrow_{\text{OCDCL}} (ML^{C\vee L}; N; U; k; \top; O)$
provided $C \vee L \in (N \cup U)$, $M \models \neg C$, $L$ is undefined in $M$

**Decide** $(M; N; U; k; \top; O) \Rightarrow_{\text{OCDCL}} (ML^{k+1}; N; U; k+1; \top; O)$
provided $L$ is undefined in $M$, contained in $N$

**ConflSat** $(M;N;U;k;\top;O) \Rightarrow_{\text{OCDCL}} (M;N;U;k;D;O)$
provided $D \in (N \cup U)$ and $M \models \neg D$

**ConflOpt** $(M;N;U;k;\top;O) \Rightarrow_{\text{OCDCL}} (M;N;U;k;\neg M;O)$
provided $O \neq \epsilon$ and $\text{cost}(M) \geq \text{cost}(O)$

**Skip** $(ML^{C\vee L};N;U;k;D;O) \Rightarrow_{\text{OCDCL}} (M;N;U;k;D;O)$
provided $D \notin \{\top, \bot\}$ and $\text{comp}(L)$ does not occur in $D$

**Resolve** $(ML^{C\vee L};N;U;k;D\vee\text{comp}(L);O) \Rightarrow_{\text{OCDCL}} (M;N;U;k;D\vee C;O)$
provided $D$ is of level $k$

**Backtrack** $(M_1K^{i+1}M_2;N;U;k;D \vee L;O) \Rightarrow_{\text{OCDCL}} (M_1L^{D\vee L};N;U \cup \{D \vee L\};i;\top;O)$
provided $L$ is of level $k$ and $D$ is of level $i$

**Improve** $(M;N;U;k;\top;O) \Rightarrow_{\text{OCDCL}} (M;N;U;k;\top;M)$
provided $M \models N$, $M$ is total, i.e., contains all atoms in $N$, and $O = \epsilon$ or $\text{cost}(M) < \text{cost}(O)$

The trail $M$ represents a conjunction, so $\neg M$ denotes the disjunction of its literals. Recall that $\bot$ denotes the empty clause. The level of the empty clause $\bot$ is 0. For simplicity, I omit the rules Restart and Forget. For OCDCL they serve the same purpose as for CDCL. For Improve it is sufficient to consider satisfiability with respect to $N$, because if $M \models N$ but $M \not\models U$ then $\text{cost}(M) \geq \text{cost}(O)$, see Proposition 2.15.3.6. Furthermore, in this case rule ConflSat is applicable.

**Definition 2.15.1** (Reasonable OCDCL Strategy)**.** An OCDCL strategy is *reasonable* if ConflSat is preferred over ConflOpt is preferred over Improve is preferred over Propagate which is preferred over the remaining rules.

Below I will show that by executing a reasonable OCDCL strategy, if $\bot$ is derived this either indicates unsatisfiability of $N$ if $O = \epsilon$, or the successful derivation of a cost minimal model if $O \neq \epsilon$. In the latter case the generated resolution refutation by the OCDCL calculus is a certificate for the optimality of $O$. If OCDCL stops in a state $(M;N;U;k;\bot;O)$ with $O \neq \epsilon$, then for every total model $M'$ of $N$ there is a clause $C \in U$ with $M' \models \neg C$ and $\text{cost}(M') \geq \text{cost}(O)$.

**Example 2.15.2** (Optimal Model)**.** Consider the clause set $N = \{P \vee Q\}$ with $\text{cost}(P) = 1$, $\text{cost}(L) = 0$ for all other literals over $\{P, Q\}$. Then a OCDCL derivation is

$$\begin{array}{ll}
\multicolumn{2}{l}{(\epsilon; N; \emptyset; 0; \top, \epsilon)} \\
\Rightarrow^{\text{Decide}}_{\text{OCDCL}} & (P^1; N; \emptyset; 1; \top; \epsilon) \\
\Rightarrow^{\text{Decide}}_{\text{OCDCL}} & (P^1 \neg Q^2; N; \emptyset; 2; \top; \epsilon) \\
\Rightarrow^{\text{Improve}}_{\text{OCDCL}} & (P^1 \neg Q^2; N; \emptyset; 2; \top; P \neg Q) \\
\Rightarrow^{\text{ConflOpt}}_{\text{OCDCL}} & (P^1 \neg Q^2; N; \emptyset; 2; \neg P \lor Q; P \neg Q) \\
\Rightarrow^{\text{Backtrack}}_{\text{OCDCL}} & (P^1 Q^{\neg P \lor Q}; N; \{\neg P \lor Q\}; 1; \top; P \neg Q) \\
\Rightarrow^{\text{ConflOpt}}_{\text{OCDCL}} & (P^1 Q^{\neg P \lor Q}; N; \{\neg P \lor Q\}; 1; \neg P \lor \neg Q; P \neg Q) \\
\Rightarrow^{\text{Resolve}}_{\text{OCDCL}} & (P^1; N; \{\neg P \lor Q\}; 1; \neg P; P \neg Q) \\
\Rightarrow^{\text{Backtrack}}_{\text{OCDCL}} & (\neg P^{\neg P}; N; \{\neg P \lor Q, \neg P\}; 0; \top; P \neg Q) \\
\Rightarrow^{\text{Propagate}}_{\text{OCDCL}} & (\neg P^{\neg P} Q^{P \lor Q}; N; \{\neg P \lor Q, \neg P\}; 0; \top; P \neg Q) \\
\Rightarrow^{\text{Improve}}_{\text{OCDCL}} & (\neg P^{\neg P} Q^{P \lor Q}; N; \{\neg P \lor Q, \neg P\}; 0; \top; \neg P Q) \\
\Rightarrow^{\text{ConflOpt}}_{\text{OCDCL}} & (\neg P^{\neg P} Q^{P \lor Q}; N; \{\neg P \lor Q, \neg P\}; 0; P \lor \neg Q; \neg P Q) \\
\Rightarrow^{\text{Resolve}}_{\text{OCDCL}} & (\neg P^{\neg P}; N; \{\neg P \lor Q, \neg P\}; 0; P; \neg P Q) \\
\Rightarrow^{\text{Resolve}}_{\text{OCDCL}} & (\epsilon; N; \{\neg P \lor Q, \neg P\}; 0; \bot; \neg P Q)
\end{array}$$

The rule Improve requires the model represented by $M$ to be total. It seems that one could also weaken this condition to partial models. However, this immediately breaks the optimality result for OCDCL. Consider the clause set $N = \{P \lor Q\}$ where $\text{cost}(P) = \text{cost}(\neg P) = 3$ and $\text{cost}(Q) = 1$. Obviously, the optimal partial model is $[Q]$ at cost 1 whereas the optimal total model has cost 4. If Improve considers partial models, then after deciding $P$, applying the partial model variant of Improve and afterwards ConflOpt and Backtrack, a run results in the state $(\neg P^{\neg P}; \{P \lor Q\}; \{\neg P\}; 0; \top; P)$. After application of ConflOpt, Resolve, the final state $(\epsilon; \{P \lor Q\}; \{\neg P\}; 0; \bot; P)$ is reached representing a solution at cost 3.

Similarly, the rule ConfOpt relies on the monotonicity of cost: $\text{cost}(M) \leq \text{cost}(ML)$ for any $L$. This can be relaxed to non-monotone cost functions, e.g., functions that also assign negative weights, see Exercise **??**.

The OCDCL calculus shares many properties with the CDCL calculus. For both OCDCL and CDCL, all literals on the trail at level $k = 0$ are propagated literals. For CDCL they are then logical consequences out of the initial clause set, for OCDCL they are consequences out of the inital clause set and the best found model so far. Where CDCL may either stop by finding a model or with a proof of the empty clause, OCDCL always derives the empty clause.

**Proposition 2.15.3** (OCDCL Basic Properties)**.** Consider an OCDCL state $(M; N; U; k; D'; O)$ derived by a reasonable strategy from start state $(\epsilon, N, \emptyset, 0, \top, \epsilon)$. Then the following properties hold:

1. $M$ is consistent.

2. If $O \neq \epsilon$ then $O$ is consistent and $O \models N$.

3. If $D' \notin \{\top, \bot\}$ then $M \models \neg D'$.

4. If $D' \notin \{\top, \bot\}$ then (i) $D'$ is entailed by $N \cup U$, or (ii) for any model $M' \models \{\neg D'\} \cup N \cup U$: $\text{cost}(M') \geq \text{cost}(O)$.

5. If $D' = \top$ and $M$ contains only propagated literals then for each valuation $\mathcal{A}$ with $\mathcal{A} \models (N \cup U)$ it holds $\mathcal{A} \models M$.

6. For all models $M$ with $M \models N$: if $O = \epsilon$ or $\text{cost}(M) < \text{cost}(O)$ then $M \models (N \cup U)$.

7. If $D' = \bot$ then OCDCL terminates and there is no model $M'$ with $M' \models N$ and $\text{cost}(M') < \text{cost}(O)$.

8. Each infinite derivation

$$(\epsilon; N; \emptyset; 0; \top; \epsilon) \Rightarrow_{\text{OCDCL}} (M_1; N; U_1; k_1; D_1; O_1) \Rightarrow_{\text{OCDCL}} \cdots$$

contains an infinite number of Backtrack applications.

9. OCDCL never learns the same clause twice.

*Proof.* 1. By Proposition 2.9.6.1, because Decide and Propagate are identical to CDCL.

2. By 1. above and the definition of Improve.

3. Both rules ConflSat and ConflOpt produce a clause $D'$ with $M \models \neg D'$. The rest follows by an inductive argument over the application of Resolve, in analogy to the proof of property 2.9.6.3.

4. By induction both on the overall derivation and the derivation of $D'$. For the latter, a state with a clause $D' \notin \{\top, \bot\}$ can only be produced by the rule ConflSat or ConflOpt. In case of ConflSat property (i) holds by Proposition 2.9.6.2. In case it is produced by ConflOpt then $O \neq \epsilon$ and $\text{cost}(\neg D') \geq \text{cost}(O)$. In particular, for any model $M'$ with $M' \models \neg D'$ it holds $\text{cost}(M') \geq \text{cost}(O)$ because cost is monotone and rule Improve is correct in that $O$ is in fact a model for $N$. Now by induction on the number of Resolve (Skip) applications assume for any model $M' \models \{\neg(D \vee \text{comp}(L))\} \cup N \cup U$: $\text{cost}(M') \geq \text{cost}(O)$, where $D' = D \vee \text{comp}(L)$. It needs to be shown for any model $M'' \models \{\neg(D \vee C)\} \cup N \cup U$: $\text{cost}(M'') \geq \text{cost}(O)$ after resolving $D \vee \text{comp}(L)$ with $C \vee L$. Now $M'' \models \neg D$, $M'' \models \neg C$ and hence $M'' \models L$ because $C \vee L \in (N \cup U)$. Therefore, $M'' \models \{\neg(D \vee \text{comp}(L))\} \cup N \cup U$ and thus $\text{cost}(M'') \geq \text{cost}(O)$.

5. Analogous to the proof of Proposition 2.9.6.4.

6. If $O = \epsilon$, then $N \models U$, because ConflOpt was not applied so far and by Proposition 2.9.6.2. If $O \neq \epsilon$ and $\text{cost}(M) < \text{cost}(O)$ by contradiction. Let $M \not\models (N \cup U)$ and let $C \in U$ be the first clause learned in the derivation with $M \models \neg C$ from state $(M'; N; U'; k; C; O)$, $U' \subseteq U$, by a Backtrack application. By assumption, $M' \models (N \cup U')$, thus $(N \cup U') \not\models C$ so by Proposition 2.15.3.4.(ii) it holds $\text{cost}(M) \geq \text{cost}(O)$.

7. If $D' = \bot$ no OCDCL rule is applicable. The calculus terminates. If $O = \epsilon$ then all clauses in $U$ are consequences of $N$, Proposition 2.15.3.4.(i), so $N \models \bot$ and $N$ has no model. If $O \neq \epsilon$, $N$ has a model, by contradiction. Assume there is a model $M'$ with $M' \models N$ and $\text{cost}(M') < \text{cost}(O)$. The final sequence of OCDCL generating $\bot$ starts with an application of ConflSat or ConflOpt generating a state $(M''; N; U; 0; D; O)$ with $D \notin \{\top, \bot\}$ and then $(M''; N; U; 0; D; O) \Rightarrow^*_{\text{OCDCL}} (M'''; N; U; 0; \bot; O)$ by rules Skip and Resolve. Obviously, $M' \not\models U$, because $N \cup U$ does not have a model anymore. Therefore, with respect to the length of the derivation, there is a minimal clause $C \in U$ that was generated out of a ConflictOpt application resulting in state $(M''''; N; U'; k; D; O)$, $D \notin \{\top, \bot\}$ and $M' \not\models C$. Now by Proposition 2.15.3.4.(ii) it holds $\text{cost}(M') \geq \text{cost}(O)$.

8. Proof by contradiction. Assume Backtrack is applied only finitely often in the infinite trace. Then there exists an $i \in \mathbb{N}^+$ such that Backtrack is not applied for all $j > i$. Propagate and Decide can only be applied as long as there are undefined literals in $M$. Since there is only a finite number of propositional variables they can only be applied finitely often.

By definition the application of the rules Skip, Resolve and Backtrack is preceded by an application of the rule ConflOpt or ConflSat since the fifth component of initial state is $\top$ and the two Conflict rules are the only rules that replace the fifth component by a clause. For the rules ConflSat and ConflOpt to be applied infinitely often the last component has to change to $\top$. By definition that can only be performed by the rules Resolve and Backtrack (a contradiction to the assumption). For Resolve assume the following rule application $(ML^{C \vee L}; N; U; k; D \vee \neg L; O) \Rightarrow_{\text{OCDCL}} (M; N; U; k; D \vee C; O)$. For $D \vee C = \top$ there must be a literal $K$ with $\{K, \text{comp}(K)\} \subseteq (D \vee C)$. With Proposition 2.15.3.3 $M \models \neg(D \vee C)$ holds which is equivalent to $M \models \bot$, a contradiction because of Proposition 2.15.3.1. Therefore ConflSat and ConflOpt are applied finitely often.

Skip and Resolve are also applied finitely often since the Conflict rules are applied finitely often and they cannot be applied infinitely often interchangeably. Otherwise the first component $M$ has to be of infinite length, a contradiction.

9. By Lemma 2.9.7. The proof carries over except that after an application of the ConflOpt rule the literal $K_1^k$ may also be the complement of $L$. □

If partial models are considered, property 2.15.3.6 breaks, because for partial models it might happen that a partial model satisfies $N$ but there are clauses in $U$ solely consisting of undefined literals with respect to the partial model. The above proof of property 2.15.3.6 assumes total models, hence from the fact that $M \not\models (N \cup U)$ we can conclude a false clause $C$ in $U$.

The rule ConflOpt starts with the negation of the overall trail as a conflict clause. This can be changed to the negation of all decision literals on the trail, see Exercise **??**.

**Lemma 2.15.4.** The OCDCL calculus with a reasonable strategy has only 2 normal forms: $(M; N; U; 0; \bot; O)$ where $O \neq \epsilon$, $O \models N$ and $\text{cost}(O)$ is optimal, and $(M; N; U; 0; \bot; \epsilon)$ where $N$ is unsatisfiable.

*Proof.* By Proposition 2.15.3.7 any state $(M; N; U; k; \bot; O)$ is a normal form. By the same proposition: if $O \neq \epsilon$ then $\text{cost}(O)$ is optimal and by Proposition 2.15.3.2 it holds $O \models N$. In case $O = \epsilon$ then during the run neither Improve nor ConflOpt have been applied, so $N$ is unsatisfiable.

A reasonable strategy cannot generate a state $(M; N; U; k; \bot; O)$ with $k > 0$, Exercise **??**. To any state $(M; N; U; k; C; O)$ with $C \notin \{\top, \bot\}$, either Skip, Resolve or Backtrack is applicable. To any state $(M; N; U; k; \top; O)$ either Propagate, or Decide, or Improve or one of the conflict rules is applicable. □

**Lemma 2.15.5** (OCDCL Termination)**.** OCDCL with a reasonable strategy terminates in a state $(M; N; U; 0; \bot; O)$.

*Proof.* Proof by contradiction. Assume there is an infinite trace that starts in a state $(\epsilon; N; \emptyset; 0; \top; \epsilon)$. With Proposition 2.15.3.9 and 2.15.3.9 there can only be a finite number of clauses that are learned during the infinite run. By definition of the rules only the rule Backtrack causes that a clause is learned so that the rule Backtrack can only be applied finitely often. But with Proposition 2.15.3.8 the rule Backtrack must be applied infinitely often, a contradiction. Therefore there does not exist an infinite trace, i.e., OCDCL always terminates. □

**Theorem 2.15.6** (OCDCL Correctness)**.** OCDCL with a reasonable strategy starting from a state $(\epsilon; N; \emptyset; 0; \top; \epsilon)$ terminates in a state $(M; N; U; 0; \bot; O)$. If $O = \epsilon$ then $N$ is unsatisfiable. If $O \neq \epsilon$ then $O \models N$ and for any other model $M'$ with $M' \models N$ it holds $\text{cost}(M') \geq \text{cost}(O)$.

*Proof.* By Lemma 2.15.5 and Lemma 2.15.4. □

If the OCDCL calculus is extended with the rules Restart and Forget it does not terminate, in general. If they are applied only finitely often or they are controlled by a fair strategy, see Section 2.10, then OCDCL terminates with Restart and Forget as CDCL does.

As an alternative for the proof of Lemma 2.15.5 the termination can be shown by assigning a well-founded measure $\mu$ and proving that it decreases with each rule application except for the rules Restart and Forget. Let $n$ be the number of propositional variables in $N$. The domain for the measure $\mu$ is $\mathbb{N} \times \{0, 1\} \times \mathbb{N} \times \mathbb{N}$. Let $\text{cost}^\epsilon$ be defined as cost but $\text{cost}^\epsilon(\epsilon) = k + 1$ where $k$ is the maximum of cost on the propositional literals occurring in $N$.

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

The well-founded ordering is the lexicographic extension of $<$ to quadruples. What remains to be shown is that each rule application except Restart and Forget decreases $\mu$. This is done via a case analysis over the rules:

Propagate:

$$\begin{aligned}\mu((M;N;U;k;\top;O)) &= (3^n-1-|U|,1,n-|M|,\operatorname{cost}^\epsilon(O))\\ &> (3^n-1-|U|,1,n-|ML^{C\vee L}|,\operatorname{cost}^\epsilon(O))\\ &= \mu((ML^{C\vee L};N;U;k;\top;O))\end{aligned}$$

Decide:

$$\begin{aligned}\mu((M;N;U;k;\top;O)) &= (3^n-1-|U|,1,n-|M|,\operatorname{cost}^\epsilon(O))\\ &> (3^n-1-|U|,1,n-|ML^{k+1}|,\operatorname{cost}^\epsilon(O))\\ &= \mu((ML^{k+1};N;U;k;\top;O))\end{aligned}$$

ConflSat:

$$\begin{aligned}\mu((M;N;U;k;\top;O)) &= (3^n-1-|U|,1,n-|M|,\operatorname{cost}^\epsilon(O))\\ &> (3^n-1-|U|,0,|M|,\operatorname{cost}^\epsilon(O))\\ &= \mu((M;N;U;k;D;O))\end{aligned}$$

ConflOpt:

$$\begin{aligned}\mu((M;N;U;k;\top;O)) &= (3^n-1-|U|,1,n-|M|,\operatorname{cost}^\epsilon(O))\\ &> (3^n-1-|U|,0,|M|,\operatorname{cost}^\epsilon(O))\\ &= \mu((M;N;U;k;\neg M;O))\end{aligned}$$

Skip:

$$\begin{aligned}\mu((ML^{C\vee L};N;U;k;D;O)) &= (3^n-1-|U|,0,|ML^{C\vee L}|,\operatorname{cost}^\epsilon(O))\\ &> (3^n-1-|U|,0,|M|,\operatorname{cost}^\epsilon(O))\\ &= \mu((M;N;U;k;D;O))\end{aligned}$$

Resolve:

$$\begin{aligned}\mu((ML^{C\vee L};N;U;k;D\vee\neg L;O)) &= (3^n-1-|U|,0,|ML^{C\vee L}|,\operatorname{cost}^\epsilon(O))\\ &> (3^n-1-|U|,0,|M|,\operatorname{cost}^\epsilon(O))\\ &= \mu((M;N;U;k;D\vee C;O))\end{aligned}$$

Backtrack: with Proposition 2.15.3-9 it holds that $D \vee L \notin U$ so that the first component decreases.

$$\begin{aligned}\mu((M_1K^{i+1}M_2;N;U;k;D\vee L;O)) &= (3^n-1-|U|,0,|M_1K^{i+1}M_2|,\operatorname{cost}^\epsilon(O))\\ &> (3^n-1-|U\cup\{D\vee L\}|,1,n-|M_1L^{D\vee L}|,\operatorname{cost}^\epsilon(O))\\ &= \mu((M_1L^{D\vee L};N;U\cup\{D\vee L\};i;\top;O))\end{aligned}$$

Improve:

$$\begin{aligned}\mu((M;N;U;k;\top;O)) &= (3^n-1-|U|,1,n-|M|,\operatorname{cost}^\epsilon(O))\\ &> (3^n-1-|U|,1,n-|M|,\operatorname{cost}^\epsilon(M))\\ &= \mu((M;N;U;k;\top;M))\end{aligned}$$

The calculus can be further improved by a pruning rule.

**Prune** $(M;N;U;k;\top;O) \Rightarrow_{\text{OCDCL}} (M;N;U;k;\neg M;O)$
provided for all total trail extensions $MM'$ of $M$ it holds $\operatorname{cost}(MM') \geq \operatorname{cost}(O)$

### 2.15.2 Max-SAT

Let $N$ be a clause set separated into hard and soft clauses: $N = N_H \uplus N_S$. The Max-SAT problems consists of finding a valuation $\mathcal{A}$ with $\mathcal{A} \models N_H$ and $\sum_{\mathcal{A}\models\neg C}^{C\in N_S} \omega(C)$ is minimal, where $\omega$ assigns a positive cost to each clause from $N_S$. The clauses in $N_H$ are called *hard clauses*, because they have to be satisfied, whereas the clauses in $N_S$ are called *soft clauses*.

The difference between hard and soft clauses occurrs naturally in practice. For example, consider the task of finding a maximal consistent subset of a clause set $N$ encoding some piece of common knowledge. In this context it is typically the case that there are facts (clauses) that are known to be true, i.e., they are *hard*, whereas for some facts this is not the case, they are *soft*. Or consider fault detection in some technical system. The eventual fault and the causal dependencies in the system constitute *hard* facts (clauses), whereas all components that may be broken represent *soft* facts.

Max-SAT can be nicely reduced to the computation of cost optimal models by introducing a separate fresh extra variable $S_i$ to each clause in $N_S = \{C_1, \ldots, C_n\}$:

$$N'_S = \{S_i \lor C_i \mid C_i \in N_S\}$$

Now instead of solving Max-SAT for $N$ a cost optimal model for $N' = N_H \uplus N'_S$ can be computed. The cost function is

$$\mathrm{cost}(L) = \begin{cases} \omega(C_i) & \text{if } L = S_i \\ 0 & \text{otherwise} \end{cases}$$

Now any valuation $\mathcal{A}$ that satisfies $N'$ satisfies all clauses in $N_H$ by construction. For the clauses $C_i \in N_S$ they are either satisfied by $\mathcal{A}$ or the respective variable $S_i$ is satisfied. All $S_i$ occur only positively in $N'$, so any model for $N_H$ can be extended to a model for $N'$. In case some $S_i$ is satisfied by $\mathcal{A}$, the OCDCL calculus accounts for the cost of the respective $C_i$.

**Theorem 2.15.7.** $\mathcal{A}$ is a Max-SAT solution for $N = N_H \uplus N_S$ with minimal value $k = \sum_{\mathcal{A}\models\neg C}^{C\in N_S} \omega(C)$ iff $(\epsilon; N'; \emptyset; 0; \top; \epsilon) \Rightarrow^*_{\mathrm{OCDCL}} (M; N'; U; k; \bot; O)$ with a reasonable strategy where $N' = N_H \uplus N'_S$, and $\mathrm{cost}(O) = k$.

*Proof.* (Sketch) Firstly, note that $O$ and $\mathcal{A}$ might actually disagree but result in the same optimum $k$.

$\Rightarrow$: If $\mathcal{A}$ is a Max-SAT solution with optimal value $k$, then $\mathcal{A}'$ defined by $\mathcal{A}'(L) = \mathcal{A}(L)$ and $\mathcal{A}'(S_i) = 1$ iff $\mathcal{A}(C_i) = 0$ is a valuation satisfying $N'$. By construction $\mathrm{cost}(\mathcal{A}') = k$, because $\mathcal{A} \not\models C_i$ iff $\mathcal{A}'(S_i) = 1$. Furthermore, $O \models N_H$ by Theorem 2.15.6. Now assume $\mathrm{cost}(O) < k$. Then we can construct a valuation $\mathcal{A}''(L) = O(L)$ for all literals $L \in N$. Obviously, $\mathcal{A}'' \models N_H$ and $\mathrm{cost}(O) = \sum_{\mathcal{A}''\models\neg C}^{C\in N_S} \omega(C)$, contradicting that $k$ is optimal. Note that $O(S_i) = 1$ and $O(C_i) = 1$ contradicts Theorem 2.15.6.

$\Leftarrow$: the arguments of the previous case can be applied in a symmetric way. $\square$

Actually, $N'$ can be further extended by all clauses generated from $C_i \rightarrow \neg S_i$ preventing that any valuation is considered where both $S_i$ and $C_i$ are satisfied.

**Example 2.15.8.** Let $N_H = \{P\}$ and $N_S = \{\neg P \lor Q, \neg P \lor \neg Q\}$ where $\omega(\neg P \lor Q) = 2$ and $\omega(\neg P \lor \neg Q) = 1$ be a Max-SAT problem. Then $N = \{P, \neg P \lor Q \lor S_1, \neg P \lor \neg Q \lor S_2\}$ is the input set for the OCDCL run with $\operatorname{cost}(S_1) = 2$, $\operatorname{cost}(S_2) = 1$ and $\operatorname{cost}(L) = 0$ for all other literals $L$ over $\{P, Q, S_1, S_2\}$. Then the following OCDCL run leads to the cost optimal model for the Max-SAT problem:

$$\begin{array}{ll}
 & (\epsilon; N; \emptyset; 0; \top; \epsilon) \\
\Rightarrow^{\text{Propagate}}_{\text{OCDCL}} & (P^P; N; \emptyset; 0; \top; \epsilon) \\
\Rightarrow^{\text{Decide}}_{\text{OCDCL}} & (P^P \neg S_1^1; N; \emptyset; 1; \top; \epsilon) \\
\Rightarrow^{\text{Propagate}}_{\text{OCDCL}} & (P^P \neg S_1^1 Q^{\neg P \lor Q \lor S_1}; N; \emptyset; 1; \top; \epsilon) \\
\Rightarrow^{\text{Propagate}}_{\text{OCDCL}} & (P^P \neg S_1^1 Q^{\neg P \lor Q \lor S_1} S_2^{\neg P \lor \neg Q \lor S_2}; N; \emptyset; 1; \top; \epsilon) \\
\Rightarrow^{\text{Improve}}_{\text{OCDCL}} & (P^P \neg S_1^1 Q^{\neg P \lor Q \lor S_1} S_2^{\neg P \lor \neg Q \lor S_2}; N; \emptyset; 1; \top; O) \\
\Rightarrow^{\text{ConflOpt}}_{\text{OCDCL}} & (P^P \neg S_1^1 Q^{\neg P \lor Q \lor S_1} S_2^{\neg P \lor \neg Q \lor S_2}; N; \emptyset; 1; \neg P \lor S_1 \lor \neg Q \lor \neg S_2; O) \\
\Rightarrow^{\text{Resolve}}_{\text{OCDCL}} & (P^P \neg S_1^1 Q^{\neg P \lor Q \lor S_1}; N; \emptyset; 1; \neg P \lor S_1 \lor \neg Q; O) \\
\Rightarrow^{\text{Resolve}}_{\text{OCDCL}} & (P^P \neg S_1^1; N; \emptyset; 1; \neg P \lor S_1; O) \\
\Rightarrow^{\text{Backtrack}}_{\text{OCDCL}} & (P^P S_1^{\neg P \lor S_1}; N; U; 0; \top; O) \\
\Rightarrow^{\text{ConflOpt}}_{\text{OCDCL}} & (P^P S_1^{\neg P \lor S_1}; N; U; 0; \neg P \lor \neg S_1; O) \\
\Rightarrow^{\text{Resolve}}_{\text{OCDCL}} & (P^P; N; U; 0; \neg P; O) \\
\Rightarrow^{\text{Resolve}}_{\text{OCDCL}} & (\epsilon; N; U; 0; \bot; O)
\end{array}$$

with $O = P \neg S_1 Q S_2$ and $U = \{\neg P \lor S_1\}$

### 2.15.3 Minimal Covering Models

Given the set of all models $\mathcal{M}$ for a set of clauses $N$, find a subset $\mathcal{M}' \subseteq \mathcal{M}$ such that $|\mathcal{M}'|$ is minimal and for each propositional variable $P$ there is a model $M \in \mathcal{M}'$ with $M(P) = 1$.

Let $n$ be the number of propositional variables $P_1, \ldots, P_n$ in $N$. If there is a solution to the problem, $n$ different models are an upper bound. The basic idea of the first solution is to consider these models in parallel by creating $n$ duplicates of $N$ with fresh variables. So each $P_i$ is replaced by $n$ fresh copies $P_i^j$, $1 \leq j \leq n$. Furthermore, the $n$ fresh variables $Q_1, \ldots, Q_n$ denote whether the $j$-th model for all variables $P_i^j$ is needed. Then the duplicated clause sets are:

$$N_j := \{C\{P_i \mapsto P_i^j \mid 1 \leq i \leq n\} \lor \neg Q_j \mid C \in N\}$$

meaning that if $Q_j$ is true in the solution and therefore the $j$-th model is needed, then the $j$-th copy of $N$ must be fulfilled. Next, every $P_i$ must have at least one true copy in the model of the overall clause set:

$$N_+ := \{P_i^1 \lor \ldots \lor P_i^n \mid 1 \leq i \leq n\}$$

and finally, if some $P_i^j$ is true in the overall model, the overall $j$-th copy of $N$ needs to be satisfied by the model:

$$N_Q := \{\neg P_i^j \vee Q_j \mid 1 \leq i, j \leq n\}.$$

Then the problem can be solved by finding a minimal cost model (Section 2.15.1) to the clause set $(\cup_{j=1}^n N_j) \cup N_+ \cup N_Q$ with cost function $\text{cost}(M) = \sum_{j=1}^n M(Q_j)$.

This encoding requires $O(n^2)$ additional variables and $O(n \cdot \max(m, n))$ additional clauses where $n$ is the number of variables and $m$ the number of clauses in $N$. The encoding depends on the upper bound $n$ of models needed to satisfy all variables. It can be significantly reduced if the upper bound for the number of models can be reduced. The idea is to start with a better upper bound for the number of necessary models than just the number of variables. The upper bound can be obtained by greedily adding models satisfying variables which were false in all previously considered models (see Algorithm 10). Then the above construction can be executed with respect to the improved upper bound.

**Algorithm 10:** GreedyFewModels($N$)

```
Input  : A clause set $N$ with variables $P_1 \ldots P_n$.
Output: A set of models for $N$ such that each variable is true in at least
        one model if such a set exists.
1  $\mathcal{P} = \{P_1, ...., P_n\}$;
2  $\mathcal{M} = \emptyset$;
3  while $\mathcal{P} \neq \emptyset$ do
4      $P$ = select a variable from $\mathcal{P}$;
5      $(P^P; N; \{P\}; 0; \top) \Rightarrow^{\downarrow}_{\text{CDCL}} (M; N; U; k; \top)$ ;
6      if ($M \cap \mathcal{P} = \emptyset$) then
7          return $\emptyset$;
8      $\mathcal{M} = \mathcal{M} \cup M$;
9      $\mathcal{P} = \mathcal{P} \setminus M$;
10 end
11 return $\mathcal{M}$;
```

For the execution of CDCL inside GreedyFewModels($N$), Algorithm 10, a decision heuristic preferring atoms from $\mathcal{P}$ is beneficial. Note that $M \cap \mathcal{P} = \emptyset$ implies that for $P \in \mathcal{P}$ no model can be found.

Another possibility is to first compute all independent models $\mathcal{M}$ such that they dominate all models with respect to the set of satisfied variables: for all $M$, $M \models N$, $M \notin \mathcal{M}$ there is an $M' \in \mathcal{M}$ such that $\{P \mid M(P) = 1\} \subseteq \{P \mid M'(P) = 1\}$ This set can be computed by creating another CDCL variant, where the set $\mathcal{M}$ is explicitely built in the sixth component of a state. The following states can occur:

| | |
|---|---|
| $(\epsilon; N; \emptyset; 0; \top; \emptyset)$ | is the start state for some clause set $N$ |
| $(M; N; U; k; \bot; \mathcal{M})$ | is a final state, where $N$ has no model if $\mathcal{M} = \emptyset$ or $\mathcal{M}$ is a covering set of models |
| $(M; N; U; k; \top; \mathcal{M})$ | is an intermediate model search state if $M \not\models N$ |
| $(M; N; U; k; D; \mathcal{M})$ | is a backtracking state if $D \notin \{\top, \bot\}$ |

**Propagate** $(M; N; U; k; \top; \mathcal{M}) \Rightarrow_{\text{CDCLcm}} (ML^{C \vee L}; N; U; k; \top; \mathcal{M})$
provided $C \vee L \in (N \cup U)$, $M \models \neg C$ and $L$ is undefined in $M$

**Decide** $(M; N; U; k; \top; \mathcal{M}) \Rightarrow_{\text{CDCLcm}} (ML^{k+1}; N; U; k+1; \top; \mathcal{M})$
provided $L$ is undefined in $M$

**ConflSat** $(M; N; U; k; \top; \mathcal{M}) \Rightarrow_{\text{CDCLcm}} (M; N; U; k; D; \mathcal{M})$
provided $D \in (N \cup U)$ and $M \models \neg D$

**ConflCM** $(M; N; U; k; \top; \mathcal{M}) \Rightarrow_{\text{CDCLcm}} (M; N; U; k; \neg M; \mathcal{M})$
provided for all total extensions $MM'$ with $MM' \models N$, there is an $I \in \mathcal{M}$ which dominates $MM'$

**Skip** $(ML^{C \vee L}; N; U; k; D; \mathcal{M}) \Rightarrow_{\text{CDCLcm}} (M; N; U; k; D; \mathcal{M})$
provided $D \notin \{\top, \bot\}$ and comp$(L)$ does not occur in $D$

**Resolve** $(ML^{C \vee L}; N; U; k; D \vee \text{comp}(L); \mathcal{M}) \Rightarrow_{\text{CDCLcm}} (M; N; U; k; D \vee C; \mathcal{M})$
provided $D$ contains a literal of level $k$

**Backtrack** $(M_1 K^{i+1} M_2; N; U; k; D \vee L; \mathcal{M}) \Rightarrow_{\text{CDCLcm}} (M_1 L^{D \vee L}; N; U \cup \{D \vee L\}; i; \top; \mathcal{M})$
provided $L$ is of level $k$ and $D$ is of level $i$

**Add** $(M; N; U; k; \top; \mathcal{M}) \Rightarrow_{\text{CDCLcm}} (M; N; U; k; \top; \mathcal{M} \cup \{M\})$
provided $M \models N$, all literals from $N$ are defined in $M$ and $M$ is not dominated by a model in $\mathcal{M}$

Once the set $\mathcal{M}$ is computed, the remaining task is to find a minimal subset of $\mathcal{M}$, covering the $P_1, \ldots, P_n$. This is the classical NP-complete Set Cover Problem [56].

Analogous to a reasonable OCDCL strategy, a CDCLcm strategy is *reasonable* if ConflSat is preferred over ConflCM is preferred over Add is preferred over Propagate which is preferred over the remaining rules. Proving the respective properties for CDCLcm is again analogous to the proofs for OCDCL.

**Theorem 2.15.9** (CDCLcm Correctness)**.** For a CDCLcm run starting from $(\epsilon, N, \emptyset, 0, \top, \emptyset)$ and ending in a state $(M; N; U; k; \bot; \mathcal{M})$, and for every variable $P$ occurring in $N$, there is a model $M$ of $N$, $M \in \mathcal{M}$, where $M \models P$ , or there is no model satisfying both $P$ and $N$.

*Proof.* See Exercise **??**. □

## 2.16 Applications

For the application of propositional logic on an arbitrary problem it needs to be encoded into a propositional formula $\phi$. The satisfiability of $\phi$ can then be checked via one of the calculi developed in this chapter, typically, CDCL. In case $\phi$ is satisfiable the corresponding calculus derives a model which has to be interpreted as a solution to the original problem. The unsatisfiability of $\phi$ must be interpreted correspondingly.

### 2.16.1 Combinatorial Finite Domain Problems

Whenever a combinatorial finite domain problem can be encoded as a SAT problem in a reasonable way, solving the problem this way is often a good choice. I start with the example of a Sudoku puzzle and discuss afterwards the overall picture.

As a suitable application of propositional logic serves the Sudoku puzzle. In chapter 1.1 a specific $4 \times 4$ Sudoku puzzle was solved using a specific calculus. In this section a general $n^2 \times n^2$ Sudoku puzzle is encoded into propositional logic and exemplarily the Resolution calculus from this chapter is applied to a $4 \times 4$ Sudoku puzzle.

For the encoding propositional variables $P^d_{i,j}$ are defined where $P^d_{i,j}$ is *true* iff the value of square $(i, j)$ is $d$. Square boxes are denoted by $Q_{i,j}$ where $Q_{i,j}$ includes the squares $(i, j), \ldots, (i+n-1, j+n-1)$. The corresponding propositional clause constraints are constructed as follows:

1. For every initially assigned square $(i, j)$ with value $d$ generate $P^d_{i,j}$
2. For every square $(i, j)$ generate $P^1_{i,j} \lor \ldots \lor P^{n^2}_{i,j}$
3. For every square $(i, j)$ and pair of values $d < d'$ generate $\neg P^d_{i,j} \lor \neg P^{d'}_{i,j}$
4. For every value $d$ and column $i$ generate $P^d_{i,1} \lor \ldots \lor P^d_{i,n^2}$ (analogously for rows)
5. For every value $d$ and square box $Q_{i,j}$ generate $P^d_{i,j} \lor \ldots \lor P^d_{i+n-1,j+n-1}$
6. For every value $d$, column $i$ and pair of rows $j < j'$ generate $\neg P^d_{i,j} \lor \neg P^d_{i,j'}$ (analogously for rows)