2.9 Conflict Driven Clause Learning (CDCL)

The CDCL calculus tests satisfiability of a finite set N of propositional clauses. Similar to DPLL, Section 2.8, the CDCL calculus explicitely builds a candidate model for a clause set. If such a sequence of literals $L_1, \ldots, L_n$ satisfies the clause set N, it is done. If not, there is a false clause $C \in N$ with respect to $L_1, \ldots, L_n$. Now instead of just backtracking through the literals $L_1, \ldots, L_n$ as done in DPLL, CDCL generates an additional clause, called *learned clause*, that actually guarantees that the subsequence of $L_1, \ldots, L_n$ that caused C to be false will not be generated anymore. This causes CDCL to be exponentially more powerful in proof length than its predecessor DPLL, Section 2.8, or Tableau, Section 2.4, see Theorem 2.14.2. The learned clause is always a resolvent from clauses in N, so CDCL can be viewed as a combination of DPLL (Tableau) and Resolution. More precisely, it can be understood as a resolution variant where a partial model assumption triggers which resolvents are actually generated. In this regard it is similar to propositional Superposition, Section 2.7, where a model assumption generated out of an a priori total ordering on the propositional variables triggers the relevant resolution steps, see the proof of propositional superposition completeness, Theorem 2.7.11. I investigate the connection between model assumptions, proof length, completeness and orderings in Section 2.11.

For any clause set N, I assume that $\bot \notin N$ and that the clauses in N do not contain duplicate literal occurrences. Furthermore, duplicate literal occurrences are always silently removed during rule applications of the calculus. A CDCL problem state is a five-tuple $(M; N; U; k; D)$ where M a sequence of annotated literals, called a *trail*, N and U are sets of clauses, $k \in \mathbb{N}$, and D is a non-empty clause or $\top$ or $\bot$, called the *mode* of the state. The set N is initialized by the problem clauses where the set U contains all newly learned clauses that are consequences of clauses from N derived by resolution. In particular, the following states can be distinguished:

$(\epsilon; N; \emptyset; 0; \top)$	is the start state for some clause set N
$(M; N; U; k; \top)$	is a final state, if $M \models N$ and all literals from N are defined in M
$(M; N; U; k; \bot)$	is a final state, where N has no model
$(M; N; U; k; \top)$	is an intermediate model search state if $M \not\models N$
$(M; N; U; k; D)$	is a backtracking state if $D \notin \{\top, \bot\}$

Literals in $L \in M$ are either annotated with a number, a level, i.e., they have the form L^k meaning that L is the $k-th$ guessed decision literal, or they are annotated with a clause that forced the literal to become true. A literal L is of *level* k with respect to a problem state $(M; N; U; j; C)$ if L or $\text{comp}(L)$ occurs in M and the first decision literal left from L ($\text{comp}(L)$) in M is annotated with k. If there is no such decision literal then $k = 0$. A clause D is of *level* k with respect to a problem state $(M; N; U; j; C)$ if k is the maximal level of a literal in D. Recall C is a non-empty clause or $\top$ or $\bot$. The rules are

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

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

Conflict $(M; N; U; k; \top) \Rightarrow_{\text{CDCL}} (M; N; U; k; D)$
provided $D \in (N \cup U)$ and $M \models \neg D$

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

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

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

Restart $(M; N; U; k; \top) \Rightarrow_{\text{CDCL}} (\epsilon; N; U; 0; \top)$
provided $M \not\models N$

Forget $(M; N; U \uplus \{C\}; k; \top) \Rightarrow_{\text{CDCL}} (M; N; U; k; \top)$
provided $M \not\models N$

Compared to expositions of this calculus in the literature, e.g. [38], the above rule set is more concrete. It does not need a Fail rule anymore and 1-UIP backjumping [9] is build in. The clause $D \vee L$ immediately propagates after Backtracking.

Recall that $\bot$ denotes the empty clause, hence failure of searching for a model. The level of the empty clause $\bot$ is 0. The clause $D \vee L$ added in rule Backtrack to U is called a *learned clause*. When applying Resolve I silently assumed that duplicate literal occurrences are merged, i.e., the clause $D \vee \text{comp}(L)$ is always condensed (see Section 2.7). Compared to superposition, condensation is always applied eagerly without mentioning. The CDCL algorithm stops with a model M if neither Propagate nor Decide nor Conflict are applicable to a state $(M; N; U; k; \top)$, hence $M \models N$ and all literals of N are defined in M. The only possibility to generate a state $(M; N; U; k; \bot)$ is by the rule Resolve. So in case of detecting unsatisfiability the CDCL algorithm actually generates a resolution proof as a certificate. I will discuss this aspect in more detail in Section 2.11. In the special case of a unit clause L, the rule Propagate actually annotates the literal L with itself. So the propagated literals on the trail are annotated with the respective propagating clause and the decision literals with the respective level.

Obviously, the CDCL rule set does not terminate in general for a number of reasons. For example, starting with $(\epsilon; N; \emptyset; 0; \top)$ any combination of the rules Propagate, Decide and eventually Restart yields the start state again. Even after a successful application of Backtrack, exhaustive application of Forget followed by Restart again may produce the start state. So why these rules Forget and Restart? Actually, any modern SAT solver makes use of the two rules. The rule Forget is needed to get rid of "redundant" clauses. For otherwise, the number of clauses in $N \cup U$ may get too large to be processed anymore in an efficient way. The rule Restart makes sense with respect to a suitable heuristic (see Section 2.10.2) for selecting the decision literals. If applied properly, it helps the calculus to focus on a part of N where it currently can make progress [9]. I will further motivate the rules later on.

The original SAT literature [48, 31, 37, 9] does not contain a theoretical foundation for a redundancy concept for CDCL. Redundancy has been experimentally developed based on heuristics where completeness is guaranteed in many implementations by increasing the number of overall kept clauses over a run and by following the DPLL style systematic exploration of potential models. I will develop a theoretical foundation for CDCL redundancy in Section 2.11. There, I will also discuss clause minimization, Section 2.13.2, a technique to strenghten learned clauses implemented in almost all CDCL solvers.

The following examples show that if the CDCL rules are applied in an arbitrary way, then many unwanted phenomena can happen. Proofs can become longer than needed, the rules can produce stuck states and clauses are learned that are already contained in the set $N \cup U$. In order to overcome all these situations, a strategy prioritizing certain rule applications is eventually added.

Example 2.9.1 (CDCL Lengthy Proof)**.** Consider the clause set $N = \{P \vee Q, \neg P \vee Q, \neg Q\}$ which is unsatisfiable. The below is a CDCL derivation proving this fact. The chosen strategy for CDCL rule selection priorizing the rule Decide produces a lengthy proof.

$$
\begin{array}{ll}
(\epsilon; N; \emptyset; 0; \top) & \\
\Rightarrow^{\text{Decide}}_{\text{CDCL}} & (P^1; N; \emptyset; 1; \top) \\
\Rightarrow^{\text{Decide}}_{\text{CDCL}} & (P^1 \neg Q^2; N; \emptyset; 2; \top) \\
\Rightarrow^{\text{Conflict}}_{\text{CDCL}} & (P^1 \neg Q^2; N; \emptyset; 2; \neg P \vee Q) \\
\Rightarrow^{\text{Backtrack}}_{\text{CDCL}} & (P^1 Q^{\neg P \vee Q}; N; \{\neg P \vee Q\}; 1; \top) \\
\Rightarrow^{\text{Conflict}}_{\text{CDCL}} & (P^1 Q^{\neg P \vee Q}; N; \{\neg P \vee Q\}; 1; \neg Q) \\
\Rightarrow^{\text{Backtrack}}_{\text{CDCL}} & (\neg Q^{\neg Q}; N; \{\neg P \vee Q, \neg Q\}; 0; \top) \\
\Rightarrow^{\text{Decide}}_{\text{CDCL}} & (\neg Q^{\neg Q} P^1; N; \{\neg P \vee Q, \neg Q\}; 1; \top) \\
\Rightarrow^{\text{Conflict}}_{\text{CDCL}} & (\neg Q^{\neg Q} P^1; N; \{\neg P \vee Q, \neg Q\}; 1; \neg P \vee Q) \\
\Rightarrow^{\text{Backtrack}}_{\text{CDCL}} & (\neg Q^{\neg Q} \neg P^{\neg P \vee Q}; N; \{\neg P \vee Q, \neg Q\}; 0; \top) \\
\Rightarrow^{\text{Conflict}}_{\text{CDCL}} & (\neg Q^{\neg Q} \neg P^{\neg P \vee Q}; N; \{\neg P \vee Q, \neg Q\}; 0; P \vee Q) \\
\Rightarrow^{\text{Resolve}}_{\text{CDCL}} & (\neg Q^{\neg Q}; N; \{\neg P \vee Q, \neg Q\}; 0; Q) \\
\Rightarrow^{\text{Resolve}}_{\text{CDCL}} & (\epsilon; N; \{\neg P \vee Q, \neg Q\}; 0; \bot)
\end{array}
$$

Example 2.9.2 (CDCL Short Proof)**.** Consider again the clause set $N = \{P \vee Q, \neg P \vee Q, \neg Q\}$ from Example 2.9.1. For the following CDCL derivation the rules Propagate and Conflict are preferred over the other rules.

$$
\begin{array}{ll}
(\epsilon; N; \emptyset; 0; \top) & \\
\Rightarrow^{\text{Propagate}}_{\text{CDCL}} & (\neg Q^{\neg Q}; N; \emptyset; 0; \top) \\
\Rightarrow^{\text{Propagate}}_{\text{CDCL}} & (\neg Q^{\neg Q} P^{Q \vee P}; N; \emptyset; 0; \top) \\
\Rightarrow^{\text{Conflict}}_{\text{CDCL}} & (\neg Q^{\neg Q} P^{Q \vee P}; N; \emptyset; 0; \neg P \vee Q) \\
\Rightarrow^{\text{Resolve}}_{\text{CDCL}} & (\neg Q^{\neg Q}; N; \emptyset; 0; Q) \\
\Rightarrow^{\text{Resolve}}_{\text{CDCL}} & (\epsilon; N; \emptyset; 0; \bot)
\end{array}
$$

Example 2.9.3 (CDCL Stuck)**.** The CDCL calculus can even get stuck, i.e., a sequence of rule applications leads to a state where no rule is applicable anymore, but the state does neither indicate satisfiability, nor unsatisfiability. Consider a clause set $N = \{Q \vee P, \neg P \vee \neg R, \ldots\}$ and the derivation

$$
\begin{array}{ll}
(\epsilon; N; \emptyset; 0; \top) & \\
\Rightarrow^{\text{Decide}}_{\text{CDCL}} & (P^1; N; \emptyset; 1; \top) \\
\Rightarrow^{\text{Decide}}_{\text{CDCL}} & (P^1 R^2; N; \emptyset; 2; \top) \\
\Rightarrow^{\text{Decide}}_{\text{CDCL}} & (P^1 R^2 Q^3; N; \emptyset; 3; \top) \\
\Rightarrow^{\text{Conflict}}_{\text{CDCL}} & (P^1 R^2 Q^3; N; \emptyset; 3; \neg P \vee \neg R).
\end{array}
$$

Obviously, neither Skip nor Resolve are applicable to the final state. Backtracking is not applicable as well because $\neg P \vee \neg R$ is of level 2 and the actual level of the final state is 3.

C Stuck states could be prevented in various ways. Either by following a particular strategy in case several rules are applicable. The reasonable strategy below, Definition 2.9.5, has this property. Or by changing the rules themselves. For example, if Skip is modified to remove decision literals from the trail, this can also prevent stuck states, see Exercise **??**.

Example 2.9.4 (CDCL Redundancy)**.** The CDCL calculus can also produce redundant clauses, in particular learn a clause that is already contained in $N \cup U$. Consider again a clause set $N = \{Q \vee P, \neg P \vee \neg R, \ldots\}$ and the derivation

$$\begin{aligned}
&(\epsilon; N; \emptyset; 0; \top)\\
\Rightarrow^{\text{Decide}}_{\text{CDCL}} \quad &(P^1; N; \emptyset; 1; \top)\\
\Rightarrow^{\text{Decide}}_{\text{CDCL}} \quad &(P^1 R^2; N; \emptyset; 2; \top)\\
\Rightarrow^{\text{Conflict}}_{\text{CDCL}} \quad &(P^1 R^2; N; \emptyset; 2; \neg P \vee \neg R).\\
\Rightarrow^{\text{Backtrack}}_{\text{CDCL}} \quad &(P^1 \neg R^{\neg P \vee \neg R}; N; \{\neg P \vee \neg R\}; 1; \top)
\end{aligned}$$

where the clause $\neg P \vee \neg R$ is learned although it is already contained in N.

I In an implementation the rule Conflict is preferred over the rule Propagate and both over all other rules. Exactly this strategy has been used in Example 2.9.2 and is called *reasonable* below. A further ingredient of a state-of-the-art implementation is a dynamic heuristic which literal is actually used by the rule Decide. This heuristic typically depends on the literals resolved on by the rule Resolve or contained in eventually learned clause. All these literals "get a bonus", see Section 2.10.2 for details.

Definition 2.9.5 (Reasonable CDCL Strategy)**.** A CDCL strategy is *reasonable* if the rules Conflict and Propagate are always preferred over all other rules.

Proposition 2.9.6 (CDCL Basic Properties)**.** Consider a CDCL state $(M; N; U; k; C)$ derived from a start state $(\epsilon, N, \emptyset, 0, \top)$ by any strategy but without using the rules Restart and Forget. Then the following properties hold:

1. M is consistent.
2. All learned clauses are entailed by N.
3. If $C \notin \{\top, \bot\}$ then $M \models \neg C$.
4. If $C = \top$ and M contains only propagated literals then for each valuation $\mathcal{A}$ with $\mathcal{A} \models N$ it holds that $\mathcal{A} \models M$.
5. If $C = \top$, M contains only propagated literals and $M \models \neg D$ for some $D \in (N \cup U)$ then N is unsatisfiable.
6. If $C = \bot$ then CDCL terminates and N is unsatisfiable.
7. k is the maximal level of a literal in M.

8. Each infinite derivation

$$(\epsilon; N; \emptyset; 0; \top) \Rightarrow_{\text{CDCL}} (M_1; N; U_1; k_1; D_1) \Rightarrow_{\text{CDCL}} \dots$$

contains an infinite number of Backtrack applications.

Proof. 1. M is consistent if it does does not contain L and $\text{comp}(L)$ at the same time. The only rules that add literals are Propagate, Decide, and Backtrack. The rules Propagate and Decide only add undefined literals to M. By an inductive argument this holds also for Backtrack as it just removes literals from M and flips one literal already contained in M.

2. A learned clause is always a resolvent of clauses from $N \cup U$ and eventually added to U where U is initially empty. By soundness of resolution (Theorem 2.6.1) and an inductive argument it is entailed by N.

3. A clause $C \notin \{\top, \bot\}$ can only occur after Conflict where $M \models \neg C$. The rule Skip does not change C and only deletes propagated literals from M that are not contained in C. By an inductive argument, if the rule Resolve is applied to a state $(M' \, \text{comp}(L)^{D' \vee \text{comp}(L)}; N; U; k; D \vee L)$ where $C = D \vee L$ resulting in $(M'; N; U; k; D \vee D')$ then $M' \models \neg D$ because $M' \models \neg C$ and $M' \models \neg D'$ because $\text{comp}(L)$ was propagated with respect to M' and $D' \vee \text{comp}(L)$.

4. Proof by induction on the number n of propagated literals in M. Let $M = L_1, \dots, L_n, L_{n+1}$. There are two rules that could have added L_{n+1}. (i) rule Propagate: in this case there is a clause $C = D \vee L_{n+1}$ where L_{n+1} was undefined in M and $M \models \neg D$. By induction hypothesis for each valuation $\mathcal{A}$ with $\mathcal{A} \models N$ it holds that $\mathcal{A}(L_i) = 1$ for all $i \in \{1, \dots, n\}$. Since all literals in D appear negated in M with the induction hypothesis it holds that all those literals must have the truth value 1 in any valuation $\mathcal{A}$. Therefore, for the clause C to be true L_{n+1} must be true as well in any valuation. It follows that for each valuation $\mathcal{A}$ it holds that $\mathcal{A}(L_i) = 1$ for all $i \in \{1, \dots, n+1\}$. (ii) rule Backtrack: the state $(M_1 K^{i+1} M_2; N; U; k; D \vee L_{n+1})$ where $M \models \neg(D \vee L_{n+1})$ (by Proposition 2.9.6.3) and $M_1 = L_1 \dots L_n$ with only propagated literals results in $(M_1 L_{n+1}^{D \vee L_{n+1}}; N; U; i; \top)$. With the induction hypothesis for each valuation $\mathcal{A}$ with $\mathcal{A} \models N$ it holds that $\mathcal{A}(L_i) = 1$ for all $1 \leq i \leq n$, i.e., in particular for each literal L in D it holds $\mathcal{A}(L) = 0$ since each literal in D appears negated in M_1. Thus, for each each valuation $\mathcal{A}$ with $\mathcal{A} \models N$ it holds $\mathcal{A}(L_{n+1}) = 1$.

5. Let $D = K_1 \vee \dots \vee K_m$. Since $M \models \neg D$ it holds that $\text{comp}(K_i) \in M$ for all $1 \leq i \leq m$. With Proposition 2.9.6.4 for each valuation $\mathcal{A}$ with $\mathcal{A} \models N$ it holds that $\mathcal{A}(L) = 1$ for all $L \in M$. Thus in particular it holds that $\mathcal{A}(\text{comp}(K_i)) = 1$ for all $1 \leq i \leq m$. Therefore D is always false under any valuation $\mathcal{A}$ and N is unsatisfiable.

6. By the definition of the rules the state $(M; N; U; k; \bot)$ can only be reached if the rule Conflict sets the mode of a state $(M'; N; U; k; \top)$ to a conflict clause D. Then Resolve is eventually used to derive $\bot$. By Proposition 2.9.6-2 it follows that N is unsatisfiable.

7. By induction on the number of rule applications. Actually, I prove something stronger, for any state $(M; N; U; k; D)$ reachable from a start state, k is the maximal level of literal in M and M includes exactly one literal L^i for $1 \leq i \leq k$ in increasing order from left to right. For the initial state $(\epsilon; N; \emptyset; 0; \top)$ the trail M is empty so $k = 0$ is fine. The only rules that manipulate decision literals in M or k are Decide, Backtrack, and Restart. Restart is fine as well. For Decide the level k is increased to $k + 1$ and a literal L^{k+1} is added right to M, so Decide is also fine. For Backtrack all decision literals including K^{i+1} are deleted from right to left from the trail, so by induction hypothesis it includes a decision literal of level i. But this is exactly the level that Backtrack sets for the new state.

8. Proof by contradiction. Assume Backtrack is applied only finitely often in the infinite trace. Then there exists an i and a state $(M_i; N; U_i; k_i; D_i)$ such that there is no Backtrack application beyond this state. Propagate and Decide can only be applied as long as there are undefined literals in M. Since N is finite there can only be finitely many rule applications of Propagate and Decide.

By definition the application of the rules Skip, Resolve and Backtrack is preceded by an application of the rule Conflict. The rules Skip and Resolve can only be applied finitely often until a decision literal is the rightmost literal of M, or M becomes empty. For the rule Conflict to be applied infinitely often mode has to change back to $\top$. But the only rule that changes the mode to $\top$ is Backtrack which is not applied anymore. A contradiction. □

Lemma 2.9.7 (CDCL Redundancy)**.** Consider a CDCL derivation by a reasonable strategy. Then CDCL never learns a clause contained in $N \cup U$.

Proof. By contradiction. Assume CDCL learns the same clause twice, i.e., it reaches a state $(M; N; U; k; D \vee L)$ where Backtracking is applicable and $D \vee L \in (N \cup U)$. More precisely, the state has the form $(M_1 K^{i+1} M_2' K_1^k K_2 \dots K_n; N; U; k; D \vee L)$ where the K_i, $i > 1$ are propagated literals that do not occur complemented in D, as for otherwise D cannot be of level i. Furthermore, one of the K_i is the complement of L. But now, because $D \vee L$ is false in $M_1 K^{i+1} M_2' K_1^k K_2 \dots K_n$ and $D \vee L \in (N \cup U)$ instead of deciding K_1^k the literal L should be propagated by a reasonable strategy. A contradiction. Note that none of the K_i can be annotated with $D \vee L$. □

A reasonable strategy is mandatory for the above result. Example 2.9.4 shows that prioritizing Decide over Propagate can result in learning a clause contained in $N \cup U$. The below example shows that even if CDCL starts Conflict with a redundant clause, one that is already subsumed, the eventually learned clause is not redundant.

Example 2.9.8 (CDCL Subsumed Conflict)**.** Consider the clause set $N = \{\neg P \vee Q, \neg P \vee R, \neg P \vee \neg R \vee \neg Q, \neg P \vee \neg R \vee Q, \dots\}$ and a derivation

$$
\begin{array}{ll}
(\epsilon; N; \emptyset; 0; \top) & \\
\Rightarrow^{\text{Decide}}_{\text{CDCL}} & (P^1; N; \emptyset; 1; \top) \\
\Rightarrow^{\text{Propagate}}_{\text{CDCL}} & (P^1 R^{\neg P \vee R}; N; \emptyset; 1; \top) \\
\Rightarrow^{\text{Propagate}}_{\text{CDCL}} & (P^1 R^{\neg P \vee R} \neg Q^{\neg P \vee \neg R \vee \neg Q}; N; \emptyset; 1; \top) \\
\Rightarrow^{\text{Conflict}}_{\text{CDCL}} & (P^1 R^{\neg P \vee R} \neg Q^{\neg P \vee \neg R \vee \neg Q}; N; \emptyset; 1; \neg P \vee \neg R \vee Q) \\
\Rightarrow^{\text{Resolve}}_{\text{CDCL}} & (P^1 R^{\neg P \vee R}; N; \emptyset; 1; \neg P \vee \neg R) \\
\Rightarrow^{\text{Resolve}}_{\text{CDCL}} & (P^1; N; \emptyset; 1; \neg P) \\
\Rightarrow^{\text{Backtrack}}_{\text{CDCL}} & (\neg P^{\neg P}; N; \{\neg P\}; 0; \top)
\end{array}
$$

Note that although the conflict clause $\neg P \vee \neg R \vee Q$ is subsumed by $\neg P \vee Q$, the eventually learned clause $\neg P$ is not redundant.

Lemma 2.9.9 (CDCL Soundness)**.** In a reasonable CDCL derivation, CDCL can only terminate in two different types of final states: $(M; N; U; k; \top)$ where $M \models N$ and $(M; N; U; k; \bot)$ where N is unsatisfiable.

Proof. If CDCL terminates with $(M; N; U; k; \top)$ then all literals of N are defined in M and Conflict is not applicable, i.e., for all clauses $C \in N$ it holds $M \models C$, so $M \models N$. In addition if CDCL terminates with $(M; N; U; k; \bot)$ then by Proposition 2.9.6.2 the clause set N is unsatisfiable.

What remains is to show that with a reasonable strategy CDCL cannot get stuck, see Example 2.9.3. I prove that no stuck state can reached by contradiction. Assume that CDCL terminates in a terminating state $(M_1 K^{i+1} M_2' K_1^k K_2 \dots K_n; N; U; k; D \vee L)$, where the K_i, $i > 1$, are propagated literals. If $\text{comp}(K_n) \neq L$ and $n > 1$ then Skip is applicable. If $\text{comp}(K_n) = L$ then either Resolve or Backtrack is applicable. Since neither Skip, Resolve, or Backtrack are applicable, it holds $n = 1$ and the complement of K_1^k does not occur in $D \vee L$. But then $M_1 K^{i+1} M_2' \models \neg(D \vee L)$ so the decision on K_1^k contradicts a reasonable strategy. □

Proposition 2.9.10 (CDCL Soundness)**.** The rules of the CDCL algorithm are sound: (i) if CDCL terminates from $(\epsilon; N; \emptyset; 0; \top)$ in the state $(M; N; U; k; \top)$, then N is satisfiable, (ii) if CDCL terminates from $(\epsilon; N; \emptyset; 0; \top)$ in the state $(M; N; U; k; \bot)$, then N is unsatisfiable.

Proof. (i) by Proposition 2.9.6.1 the sequence M is always consistent. Since $(M; N; U; k; \top)$ is a final state neither Decide nor Propagate can be applied and hence all atoms of literals in N occur in M. Furthermore, the state is not a result of an application of the Conflict rule nor is the Conflict rule applicable. Therefore, there is no $D \in (N \cup U)$ with $M \models \neg D$ and since all literals in N are defined in M this means $M \models N$, so N is satisfiable.

(ii) the only rule that can produce $(M; N; U; k; \bot)$ is Conflict, where $\bot \in (N \cup U)$. By Proposition 2.9.6.2 all learned clauses are entailed by N, hence N is unsatisfiable. □

Proposition 2.9.11 (CDCL Strong Completeness)**.** The CDCL rule set is complete: for any valuation M with $M \models N$ there is a reasonable sequence of rule