A Verified SAT Solver Framework with Learn, Forget, Restart, and Incrementality

Jasmin C. Blanchette[1,2]

Mathias Fleury[2,3]

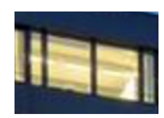
Christoph Weidenbach[2]

[1] Inria Nancy – Grand Est

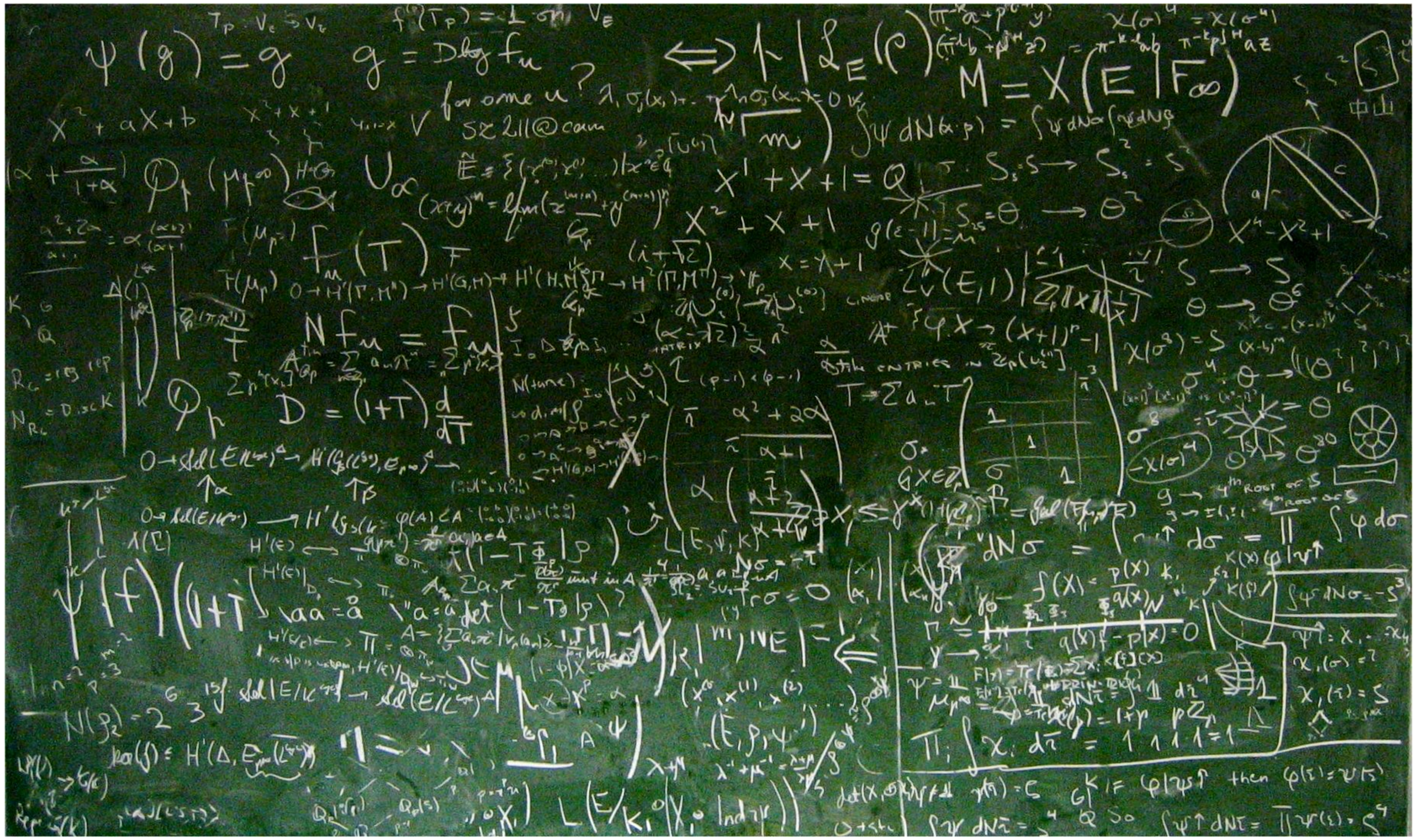
[2] Max-Planck-Institut für Informatik

[3] Graduate School of Computer Science Saarbrücken

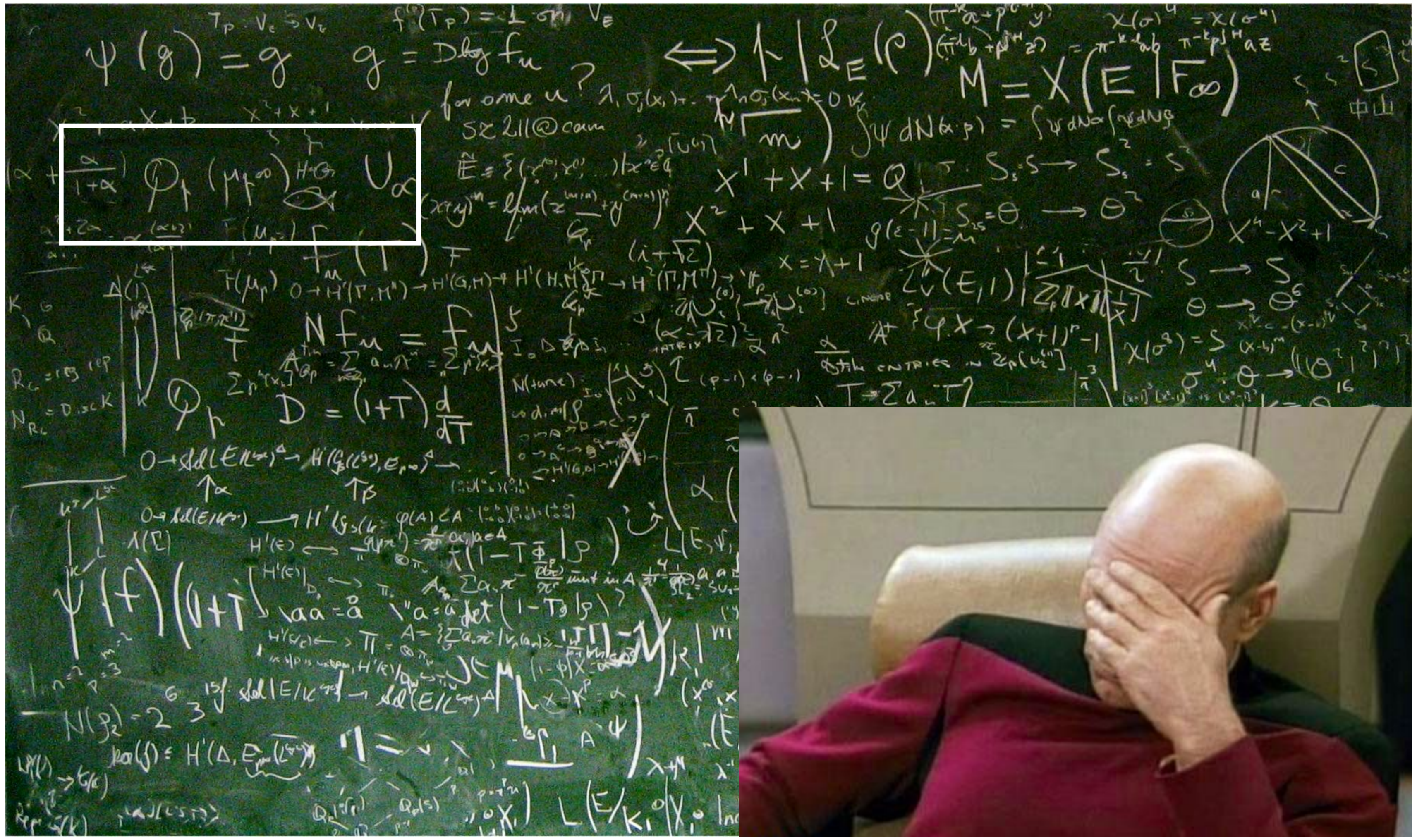
A researcher working on a new calculus



When you start a proof



When you do a proof



When you find your error on the first line

Write the paper

4 CDCL – Conflict Driven Clause Learning

The CDCL calculus tests satisfiability of a finite set N of propositional clauses. I assume that $\bot \notin N$ and that the clauses in N do not contain duplicate literal occurrences.

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 $L_1, \ldots, L_n$, CDCL generates in addition a new 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 [7] or classical Tableau [17].

A CDCL problem state is a five-tuple $(M; N; U; k; C)$ where M a sequence of annotated literals representing a partial model, called a *trail*, N and U are sets of clauses, $k \in \mathbb{N}$, and C is a non-empty clause or $\top$ or $\bot$, called the *mode* of the state. 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 $M \not\models N$ or not all literals from N are defined in M
$(M; N; U; k; D)$ 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 that the mode 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. [12], the above rule set is more concrete. It does not need a Fail rule anymore and 1UIP backtracking [6] 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 assume that duplicate literal occurrences are merged, i.e., the clause $D \vee C$ is always condensed (see Section 3). 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 5. 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 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 [6].

The original SAT literature [6,10,11,16] does not contain a redundancy notion for CDCL. A huge part of the results were found out via system design, such as the early Chaff or RelSAT, and experimental evaluation. I will develop a theoretical foundation in Section 5.

The following examples show that if the CDCL rules are applied in an arbitrary order, then unwanted phenomena can happen. The rules 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 8 (CDCL Proof). Consider the clause set $N = \{P\vee Q, \neg P\vee Q, \neg Q\vee P, \neg P\vee\neg Q\}$. For the following CDCL derivation the rules Conflict and Propagate are preferred over the other rules.

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

For the clause set $N \backslash \{\neg P \vee Q\}$ the fourth last state $(\neg Q^{\neg Q} P^{P\vee Q}; N; \{\neg Q\}; 0; \top)$ is terminal, representing the model $\neg Q\, P$.

Example 9 (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.

Example 10 (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

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

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

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 8 and is called *reasonable* below. A further ingredient of a state-of-the-art implementation is a dynamic heuristic suggesting which literal is actually used by the rule Decide. This heuristic typically depends on the literals resolved by the rule Resolve or that are contained in an eventually learned clause. All these literals "get a bonus", e.g., see [6].

Definition 11 (Reasonable CDCL Strategy). A CDCL strategy is *reasonable* if the rule Conflict is always preferred over the rule Propagate which is always preferred over all other rules.

Proposition 12 (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 C is 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 interpretation $\mathcal{I}$ with $\mathcal{I} \models N$ it holds that $\mathcal{I} \models M$, i.e., $M \subseteq \mathcal{I}$.*
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*
$$(\epsilon; N; \emptyset; 0; \top) \Rightarrow_{CDCL} (M_1; N; U_1; k_1; D_1) \Rightarrow_{CDCL} \cdots$$
contains an infinite number of Backtrack applications.

Lemma 13 (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 $(M_1 K^{i+1} M_2' K_1^k K_2 \ldots K_n; N; U; k; D \vee L)$ 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 $M_1 K^{i+1} M_2' K_1^k K_2 \ldots K_n$ and $D \vee L \in (N \cup U)$ instead of deciding K^k the literal L should have been propagated by a reasonable strategy. A contradiction. □

Lemma 14 (CDCL Soundness). *In a reasonable CDCL derivation, CDCL can only terminate in two different 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 12.2 the clause set N is unsatisfiable.

What remains is to show that with a reasonable strategy CDCL cannot get stuck, see Example 9. I prove that no stuck state can be reached by contradiction. Assume that CDCL terminates in a state $(M_1 K^{i+1} M_2' K_1^k K_2 \ldots K_n; N; U; k; D \vee L)$, 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 15 (CDCL Strong Completeness). *The CDCL rule set is strongly complete: for any interpretation M restricted to the variables occurring in N with $M \models N$, there is a reasonable sequence of rule applications generating $(M'; N; U; k; \top)$ as a final state, where M and M' only differ in the order of literals.*

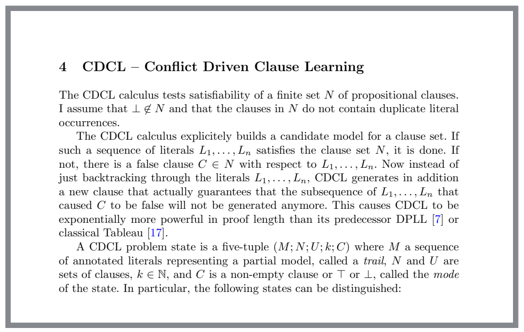
Proof. By induction on the length of M. Assume we have already reached a state $(M'; N; U; k; \top)$ where $M' \subset M$. If Propagate is applicable to $(M'; N; U; k; \top)$ extending it to $(M' L^{C\vee L}; N; U; k; \top)$ then $L \in M$. For otherwise, I pick a literal $L \in M$ that is not defined in M' and apply Decide yielding $(M' L^{k+1}; N; U; k+1; \top)$. The rule Conflict is not applicable, because $M \models N$ and $M' \subset M$. □

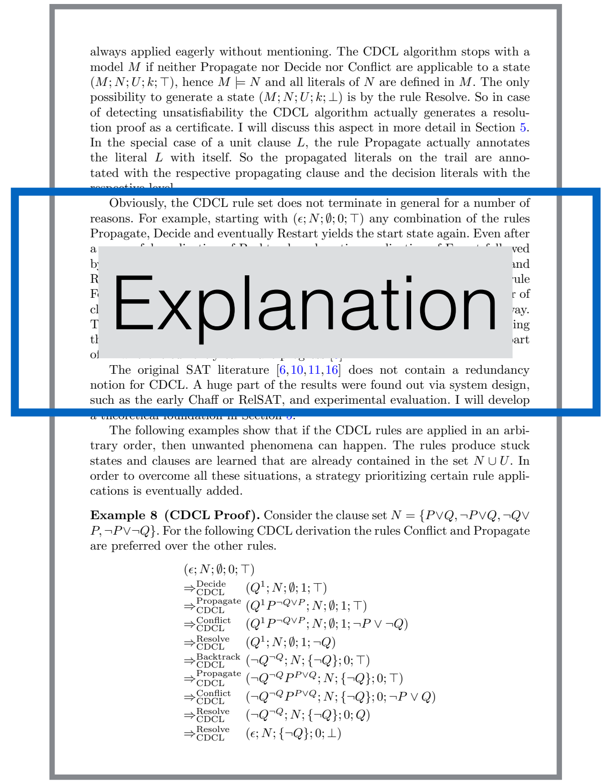
Proposition 16 (CDCL Termination). *Assume the algorithm CDCL with all rules except Restart and Forget is applied using a reasonable strategy. Then it terminates in a state $(M; N; U; k; D)$ with $D \in \{\top, \bot\}$.*

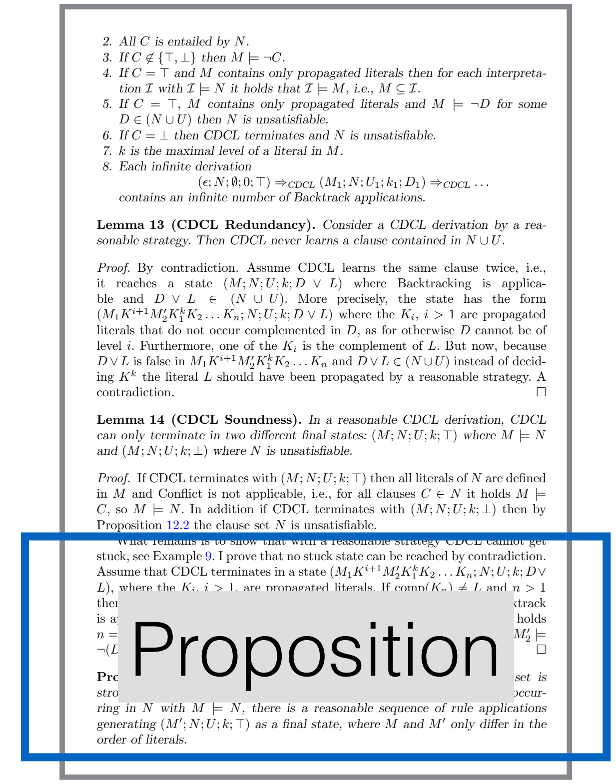
Proof. By Lemma 14 if CDCL terminates using a reasonable strategy then $D \in \{\top, \bot\}$. I show termination by contradiction. By Proposition 12.8 an infinite run includes infinitely many Backtrack applications. By Lemma 13 each learned clause does not occur in $N \cup U$. But there are only finitely many different condensed clauses with respect to the finite signature contained in N. A contradiction. □

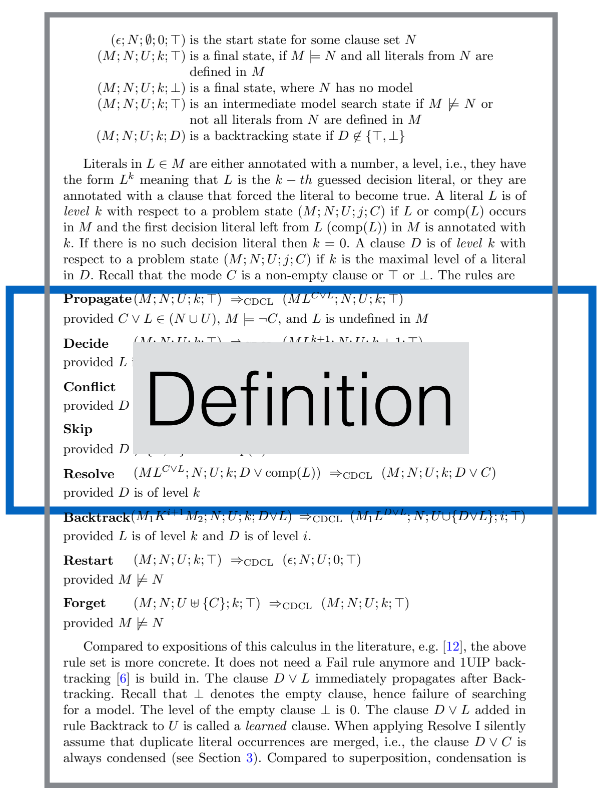
Paper accepted = proof correct

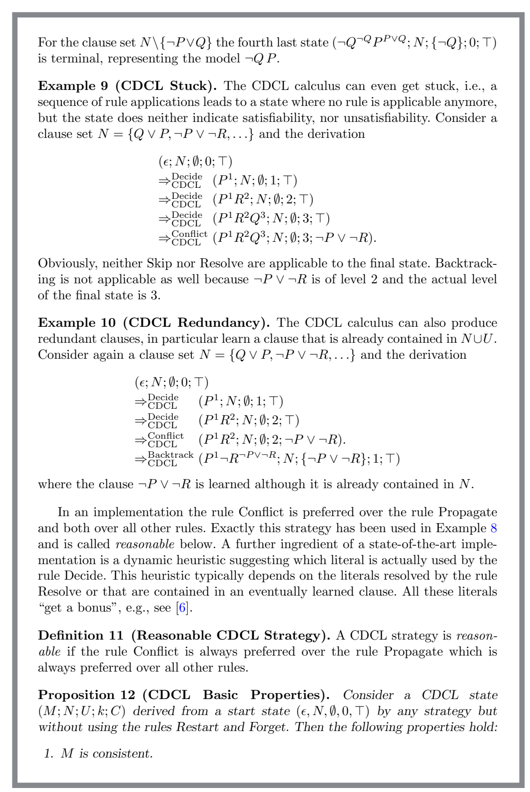
Now we want to improve the calculus

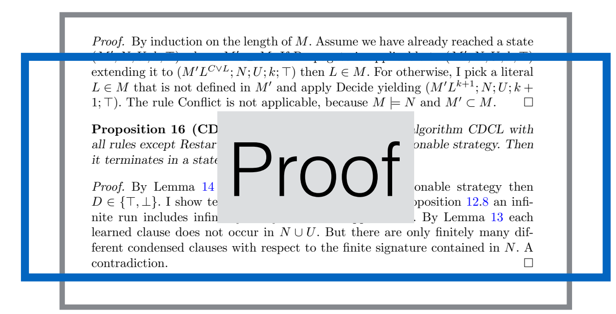






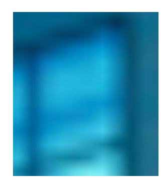




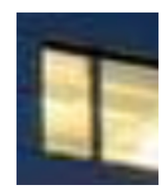



What must be updated?

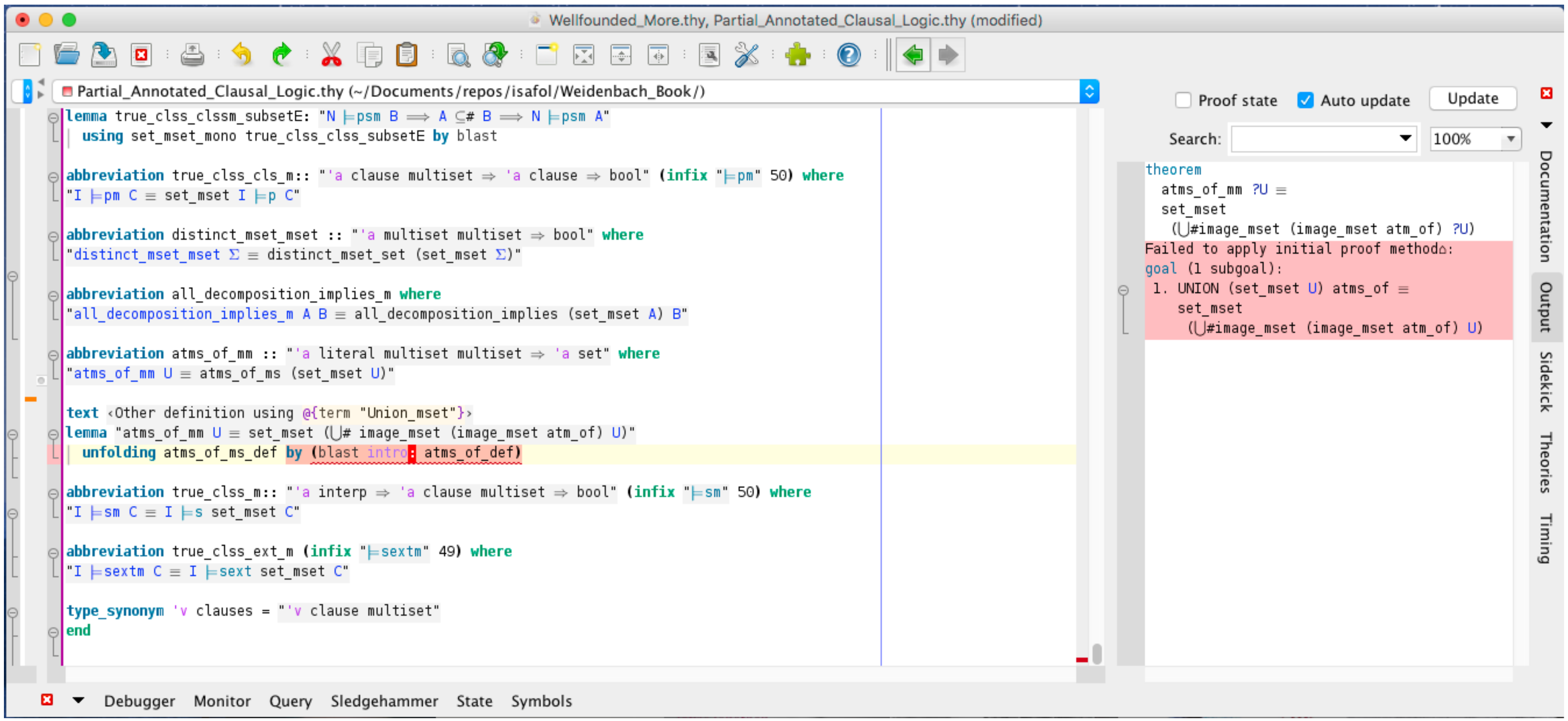
Could proof assistants help here?





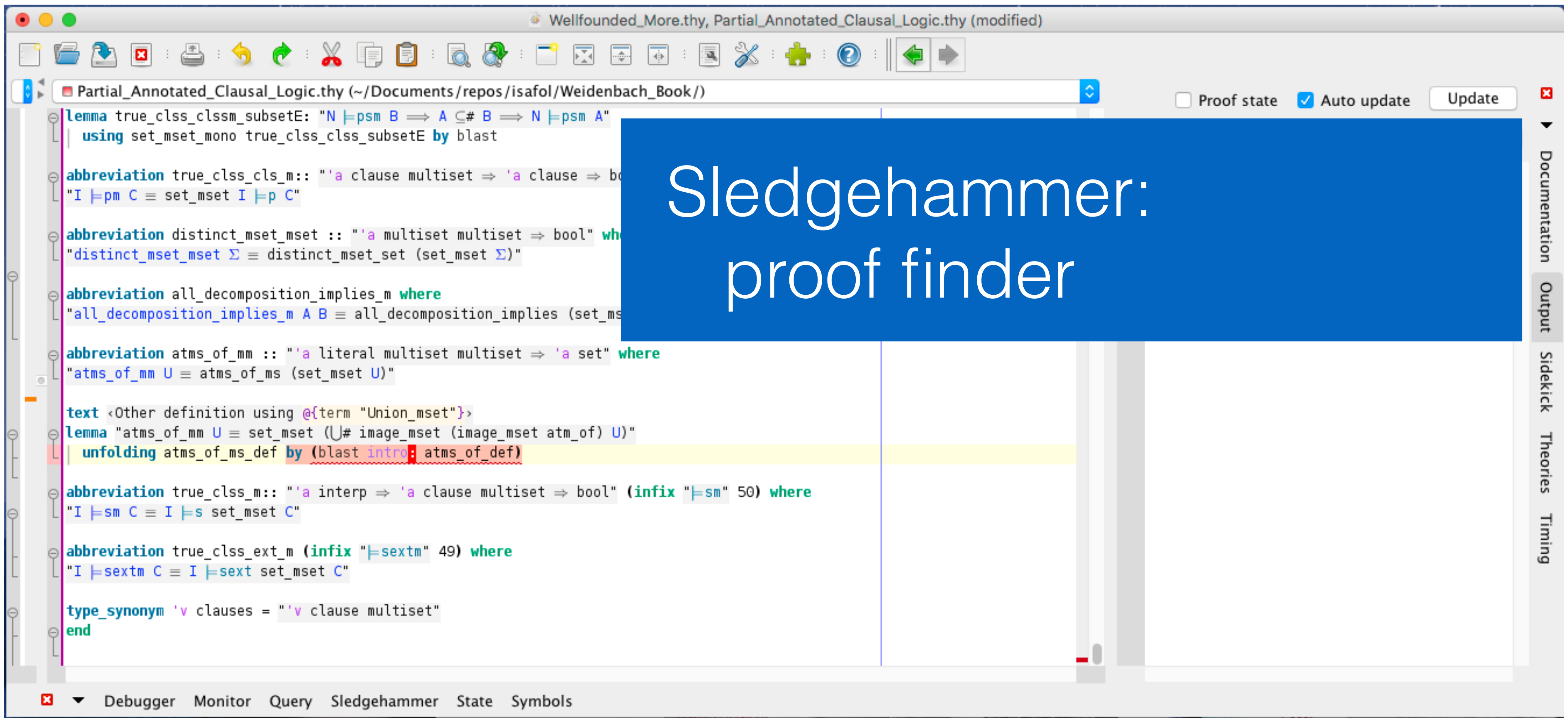
Isabelle/HOL



INVENTORS FOR THE DIGITAL WORLD

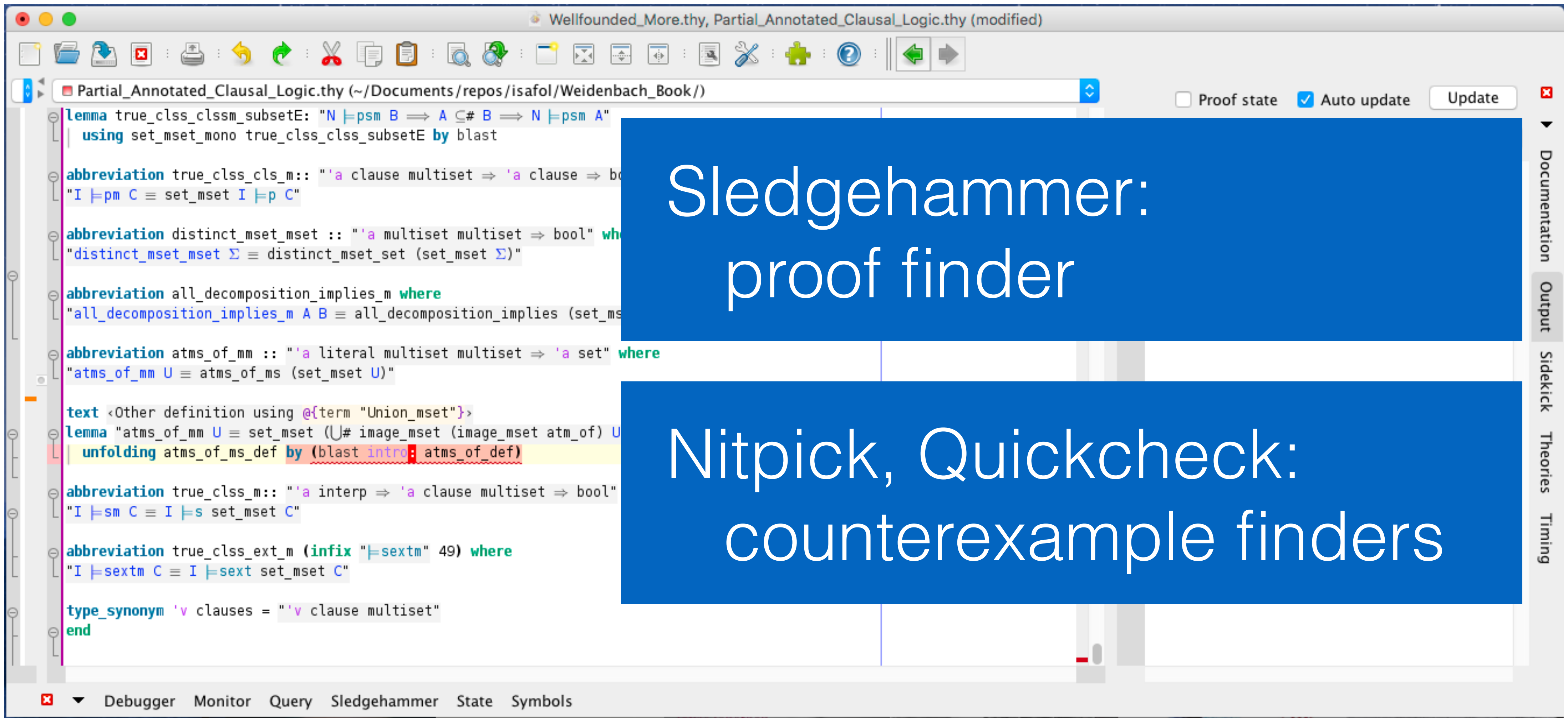
max planck institut
informatik

Isabelle/HOL



Sledgehammer:
proof finder

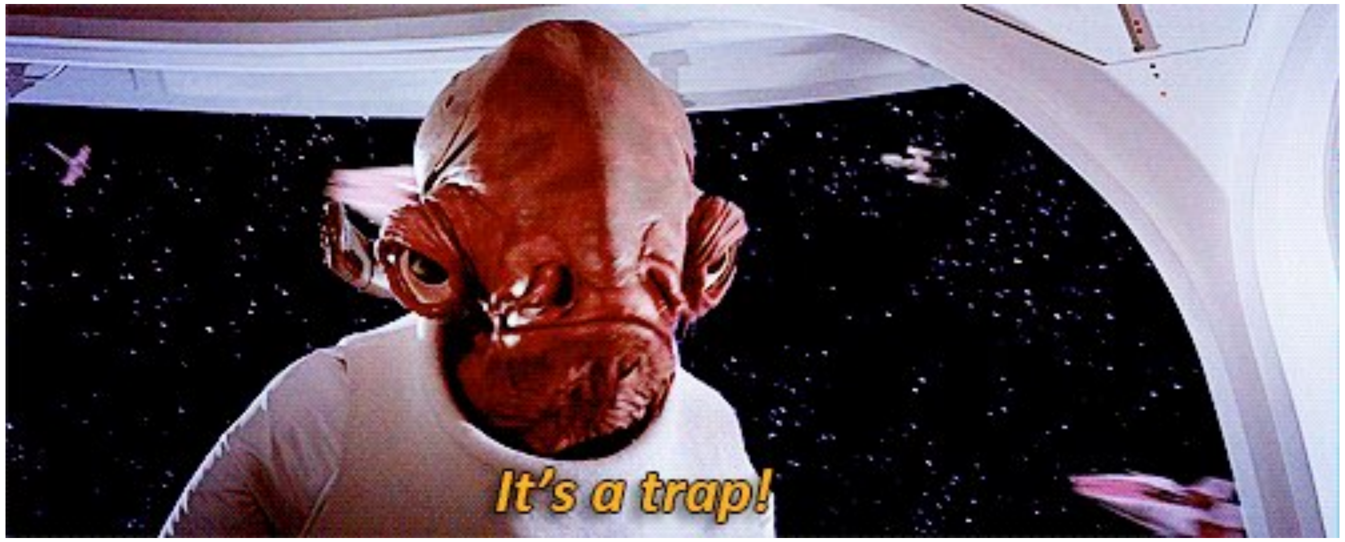
Isabelle/HOL



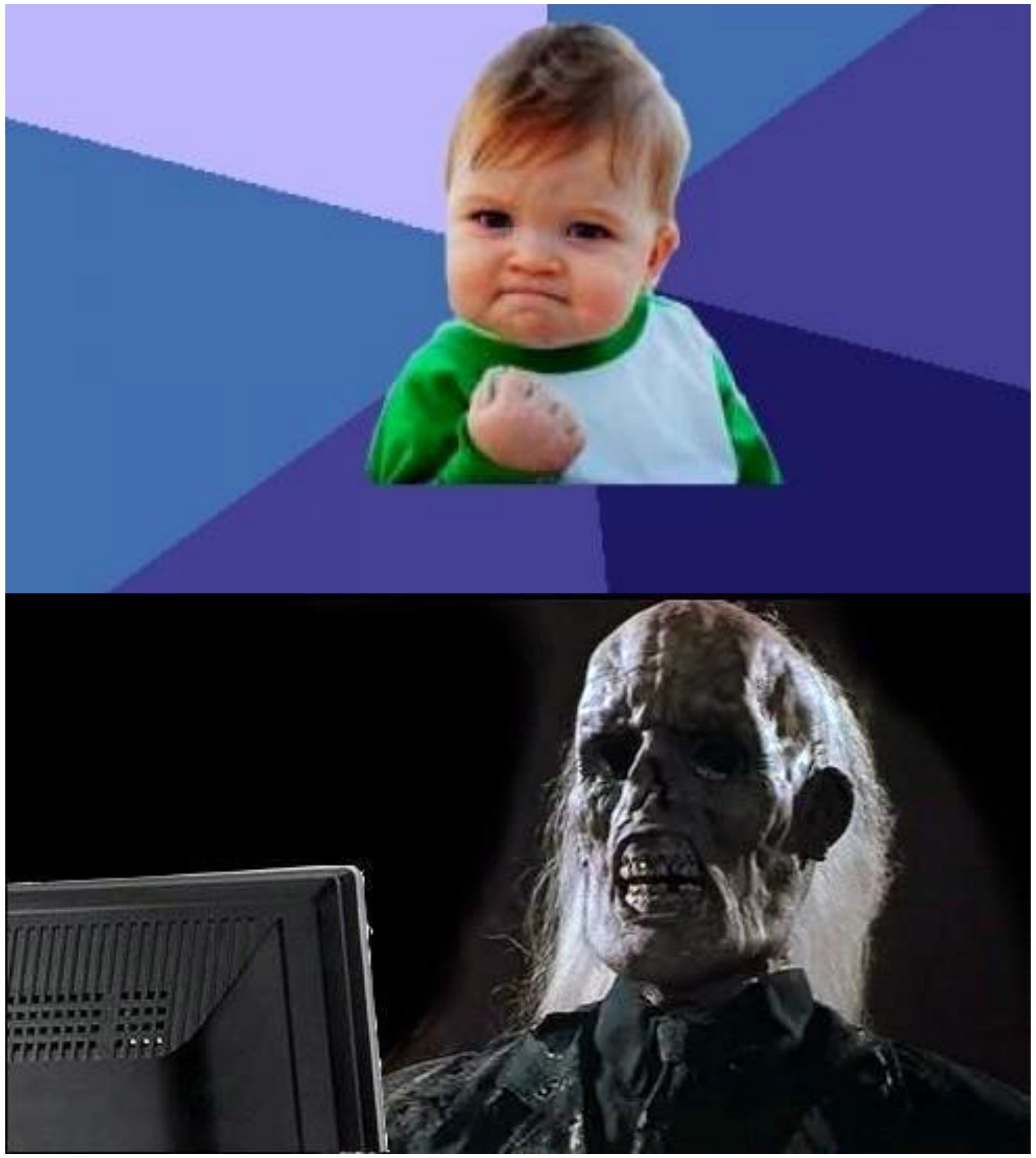
Sledgehammer:
proof finder

Nitpick, Quickcheck:
counterexample finders


You mean
not everyone is
using that?


It's a trap!

When you start



When you start

When you finish

State of the art

Paper

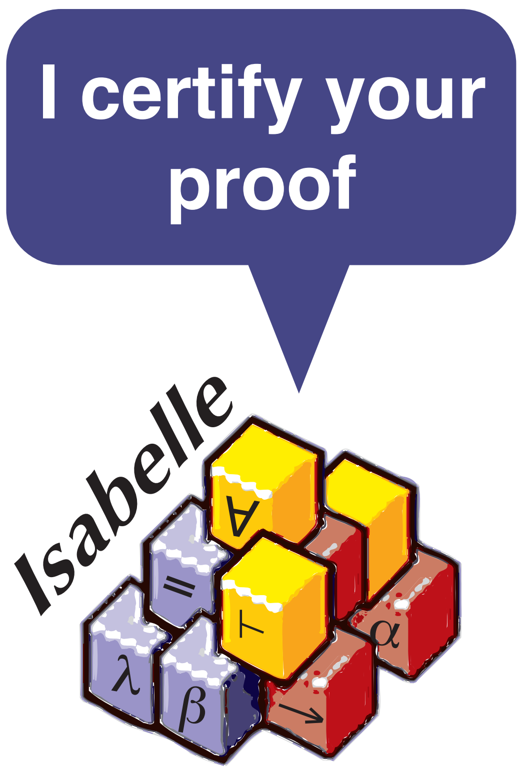
Proof Assistant

IsaFoL project

Isabelle Formalisation of Logic

IsaFoL



Motivation

- Eat our own dog food

  case study for proof assistants and automatic provers

- Build libraries for state-of-the-art research

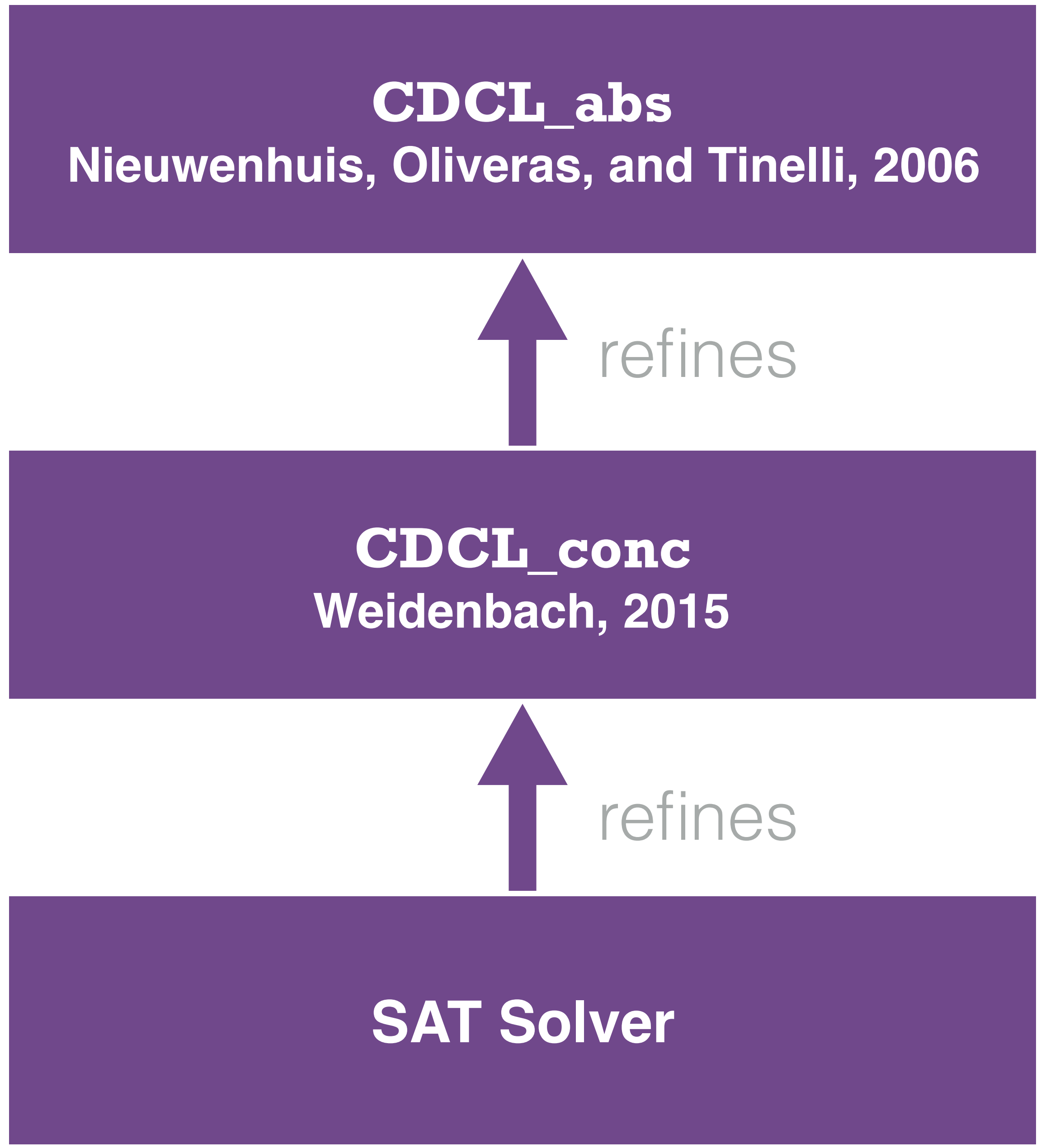
  Automated Reasoning:
  The Art of Generic Problem Solving
  (forthcoming textbook by Weidenbach)

IsaFoL

- FO tableau
  by Blanchette, Popescu, Traytel (IJCAR 2014)
- FO resolution
  by Schlichtkrull (ITP 2016)
- CDCL with learn, forget, restart, and incrementality
  by Blanchette, Fleury, Weidenbach (IJCAR 2016)
- FO ordered resolution with selection
  by Blanchette, Schlichtkrull, Traytel (ongoing)

IsaFoL

- FO tableau
  by Blanchette, Popescu, Traytel (IJCAR 2014)
- FO resolution
  by Schlichtkrull (ITP 2016)
- CDCL with learn, forget, restart, and incrementality
  by Blanchette, Fleury, Weidenbach (IJCAR 2016)
- FO ordered resolution with selection
  by Blanchette, Schlichtkrull, Traytel (ongoing)


CDCL_abs
Nieuwenhuis, Oliveras, and Tinelli, 2006
refines
CDCL_conc
Weidenbach, 2015
refines
SAT Solver

CDCL_abs

Nieuwenhuis, Oliveras, and Tinelli, 2006

Core of CDCL: **DPLL+BJ**

DPLL+BJ = Propagate + Decide + Backjump

Core of CDCL: **DPLL+BJ**

DPLL+BJ = Propagate + Decide + Backjump

DPLL = Propagate + Decide + Backtrack

Core of CDCL: **DPLL+BJ**

$$
\begin{array}{rcl}
\text{DPLL+BJ} & = & \text{Propagate} + \text{Decide} + \boxed{\text{Backjump}} \\
 & & \qquad\qquad\qquad\qquad\quad \cup\!\!| \\
\text{DPLL} & = & \text{Propagate} + \text{Decide} + \boxed{\text{Backtrack}}
\end{array}
$$

Core of CDCL: **DPLL+BJ**

$$\begin{array}{rcl} \textbf{DPLL+BJ} & = & \textbf{Propagate} + \textbf{Decide} + \boxed{\begin{array}{c}\textbf{Backjump} \\ \cup\!\mid \\ \textbf{Backtrack}\end{array}} \\ \textbf{DPLL} & = & \textbf{Propagate} + \textbf{Decide} + \end{array}$$

Backtrack $\Rightarrow$ **Backjump**

- deduce termination from DPLL+BJ

Backtrack = **Parametrised Backjump**(BT_cond)

- get all theorems for free from DPLL+BJ

Trail
(candidate model)

(M, N)

Current clauses

Backjump

on paper

if $C \in N$ and $M \models \neg C$ and
there is C' such that ...
then $(M, N) \Rightarrow_{\text{DPLL+BJ}} (L^{\dagger} M', N)$

Parametrised Backjump(BJ_cond)

if $C \in N$ and $M \models \neg C$ and
there is C' such that ...
and $BJ\_cond\ C'$
then $(M, N) \Rightarrow_{\mathbf{DPLL+BJ}} (L^{\dagger} M', N)$

on paper

$\cap|$

Backjump

on paper

if $C \in N$ and $M \models \neg C$ and
there is C' such that ...
then $(M, N) \Rightarrow_{\mathbf{DPLL+BJ}} (L^{\dagger} M', N)$

Parametrised Backjump(BJ_cond)

if $C \in N$ and $M \models \neg C$ and
there is C' such that ...
and $BJ\_cond\ C'$
then $(M, N) \Rightarrow_{\text{DPLL+BJ}} (L^{\dagger} M', N)$

on paper

under some assumptions

Backjump

if $C \in N$ and $M \models \neg C$ and
there is C' such that ...
then $(M, N) \Rightarrow_{\text{DPLL+BJ}} (L^{\dagger} M', N)$

on paper

Development hierarchy

specialises

DPLL → DPLL+BJ

```
sublocale DPLL ⊆ DPLL+BJ where
  BJ_cond = BT_cond
```

in Isabelle

Backtrack = **Parametrised Backjump**(BT_cond)

DPLL = **DPLL+BJ**(BT_cond)

Development hierarchy

DPLL+BJ

specialises

DPLL

```
sublocale DPLL ⊆ DPLL+BJ where
  BJ_cond = BT_cond
```

in Isabelle

prove those assumptions

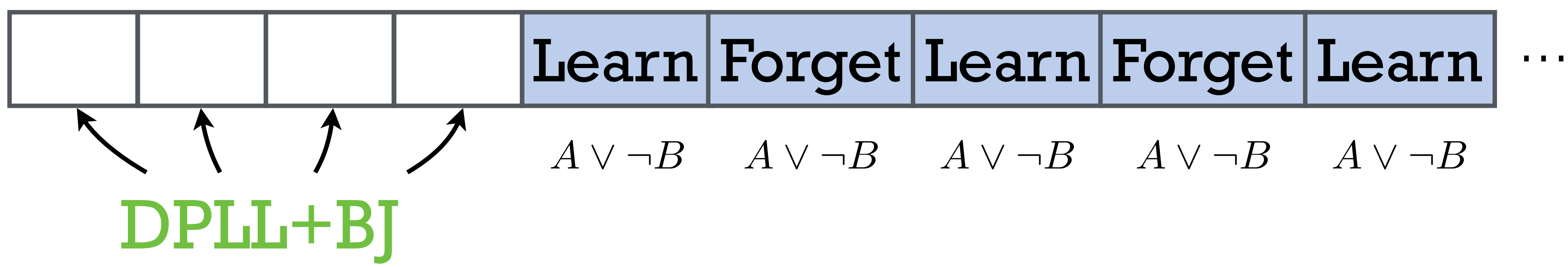
Backtrack = Parametrised Backjump(BT_cond)

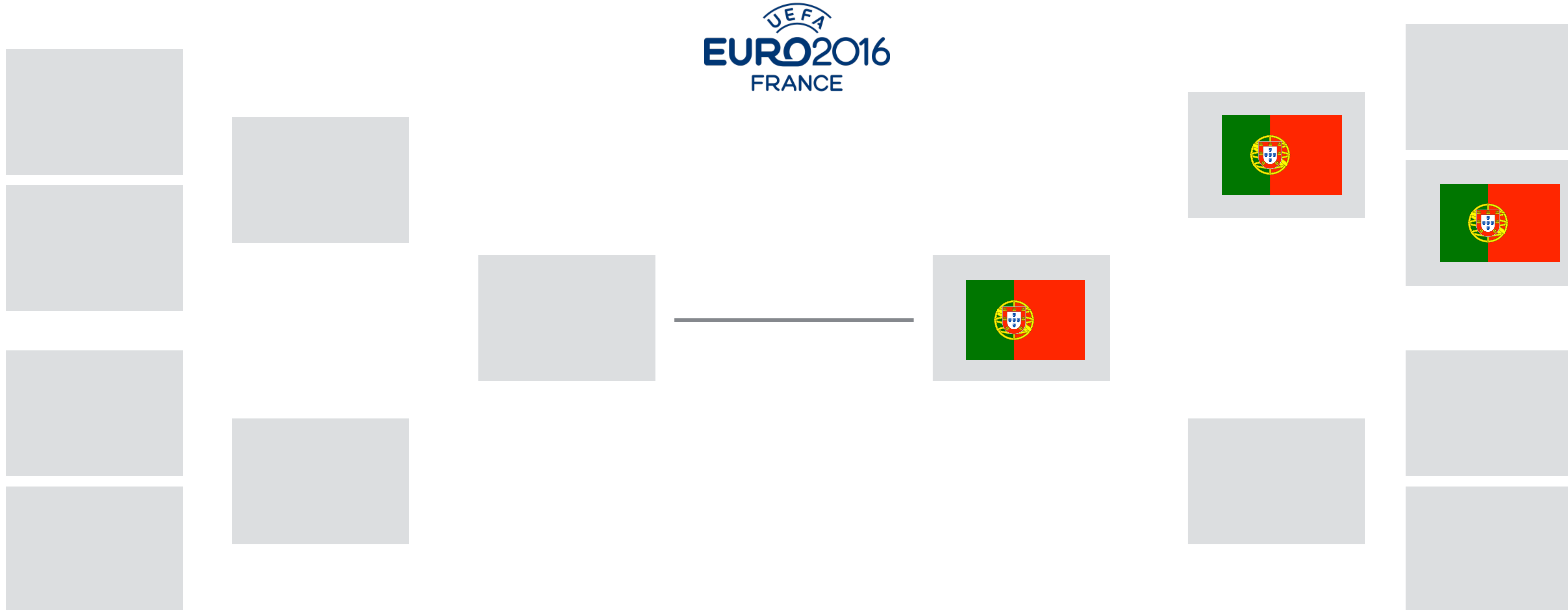
DPLL = DPLL+BJ(BT_cond)

Development hierarchy

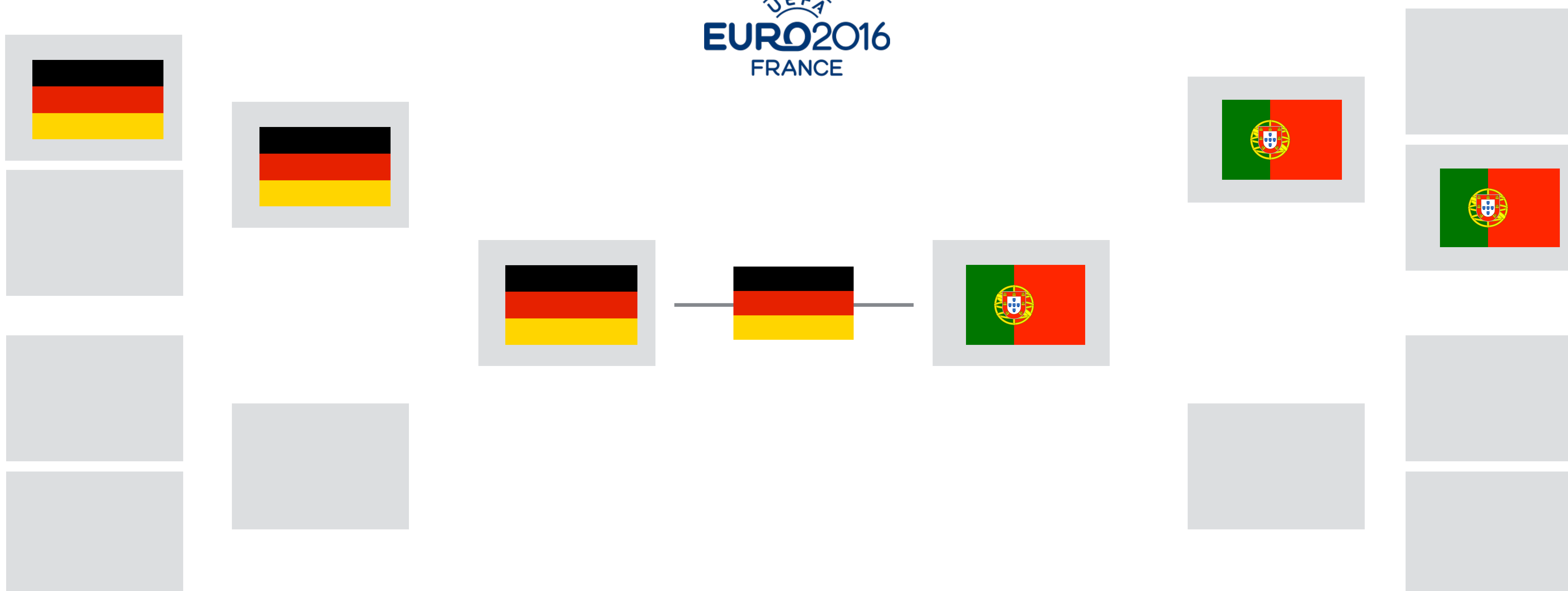
CDCL_abs = DPLL+BJ + Learn + Forget

CDCL_abs = DPLL+BJ + Learn + Forget





Learn



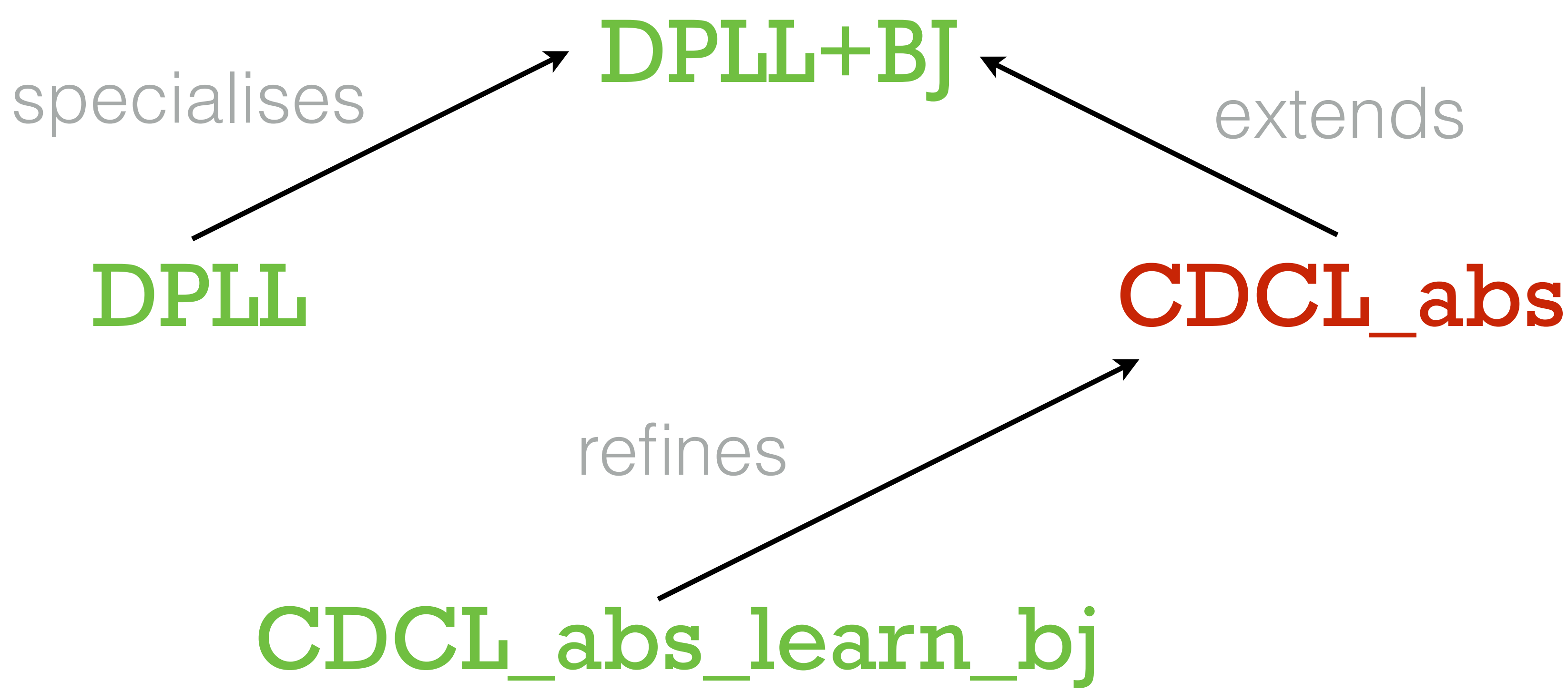
Backjump

Learn

Backjump

Learn

Development hierarchy

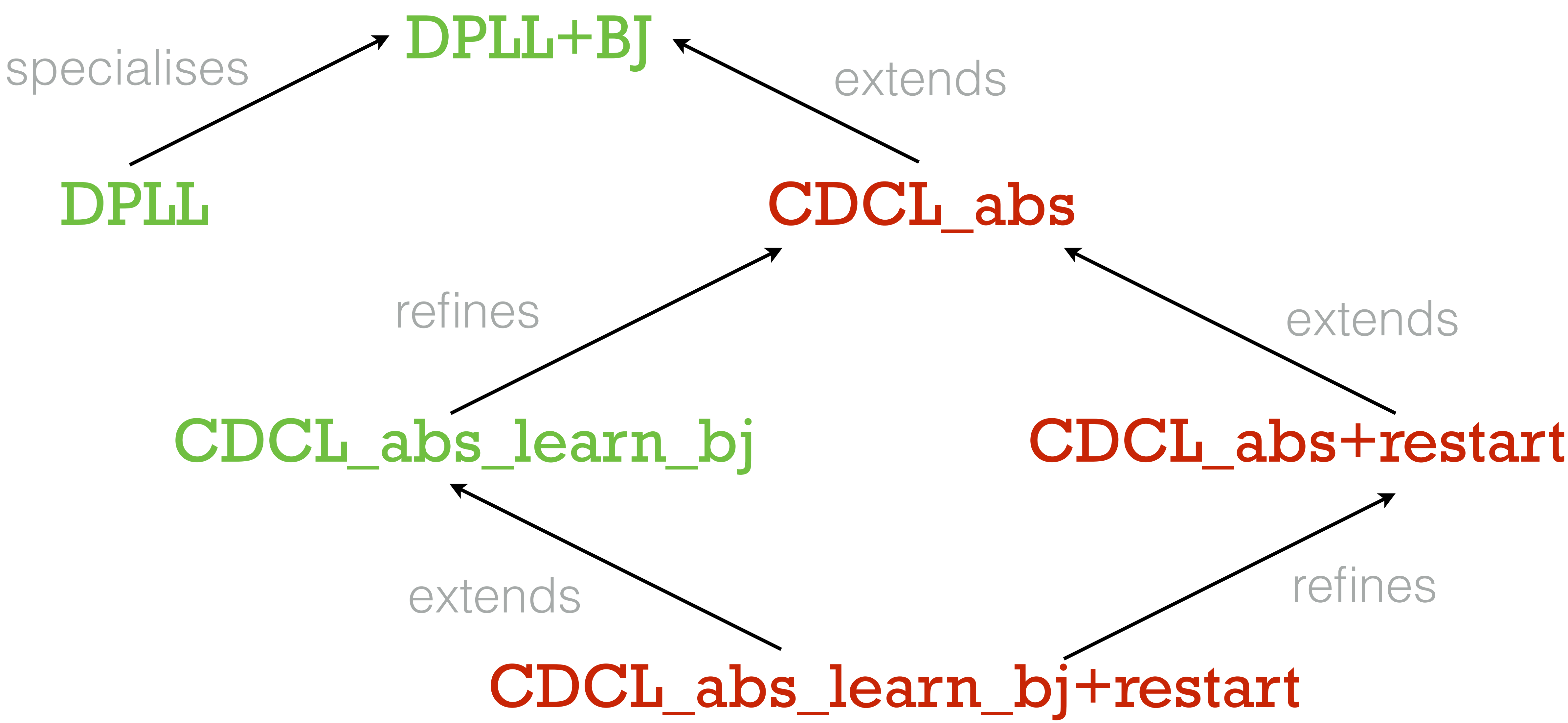


CDCL_abs_learn_bj

learn only clause of backjump

successful in real implementations

Development hierarchy



CDCL_abs+restart = CDCL_abs + Restart

CDCL_abs_learn_bj+restart = CDCL_abs_learn_bj + Restart

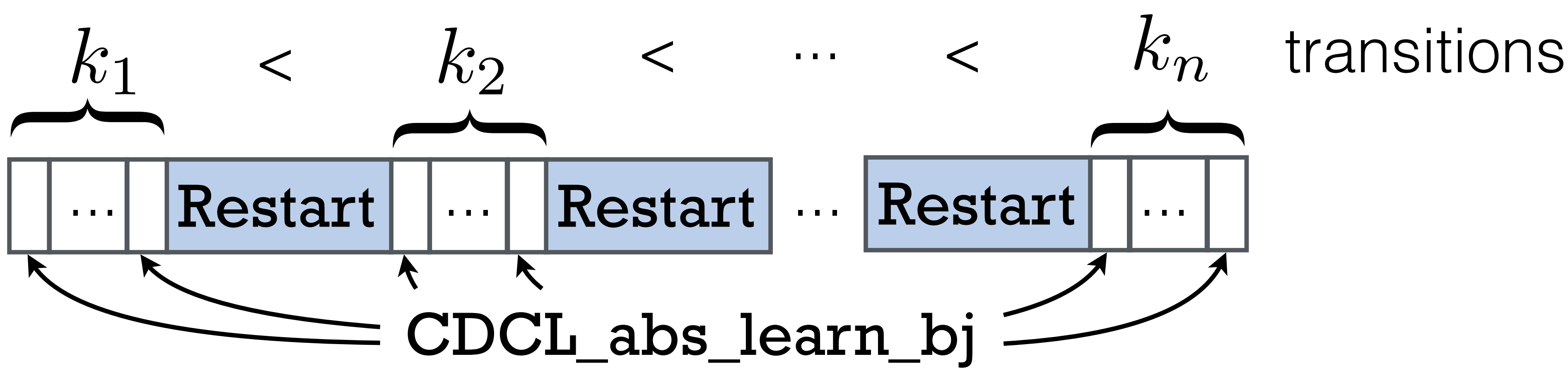
Restart Restart Restart Restart ···

CDCL_abs+restart = CDCL_abs + Restart

CDCL_abs_learn_bj+restart = CDCL_abs_learn_bj + Restart

Restart Restart Restart Restart ⋯

CDCL_abs_learn_bj+restart_T

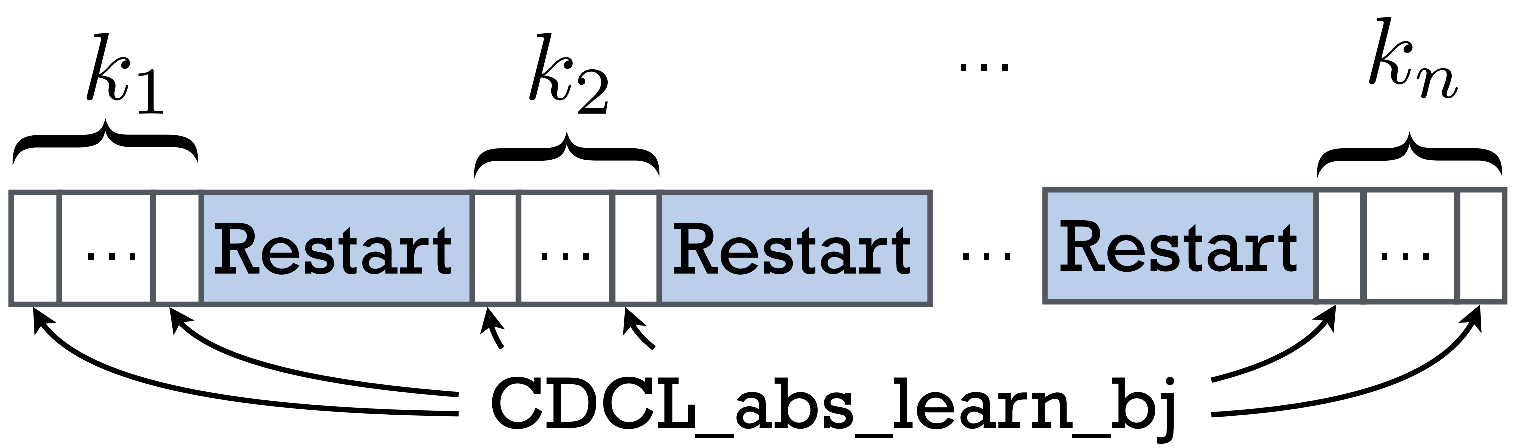


CDCL_abs+restart = CDCL_abs + Restart

CDCL_abs_learn_bj+restart = CDCL_abs_learn_bj + Restart

Restart Restart Restart Restart ⋯

CDCL_abs_learn_bj+restart_T



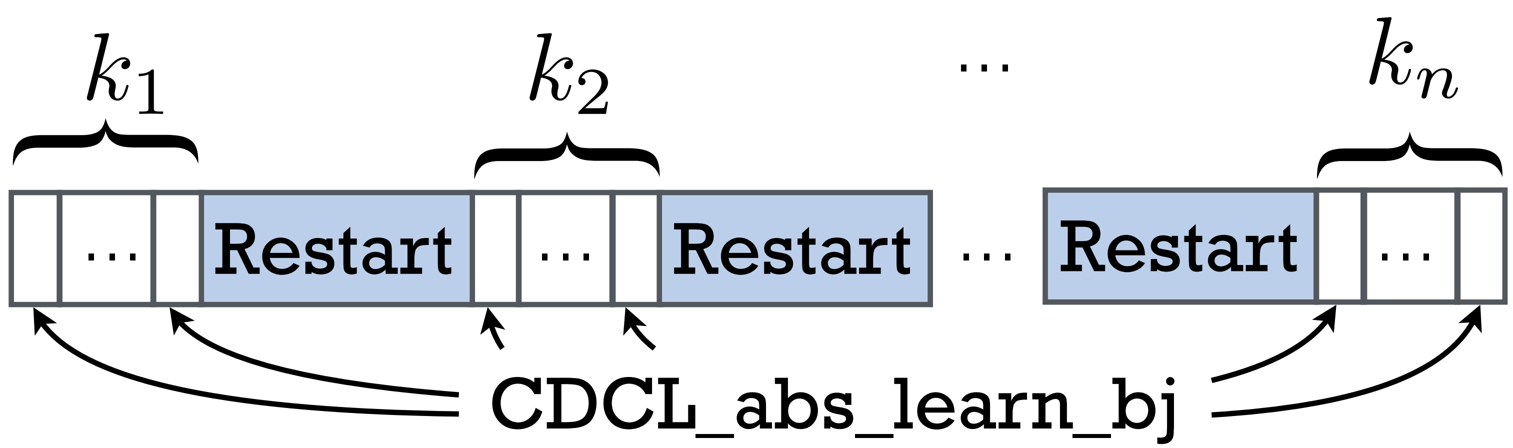
$(k_i)_i$ unbounded

CDCL_abs+restart = CDCL_abs + Restart

CDCL_abs_learn_bj+restart = CDCL_abs_learn_bj + Restart

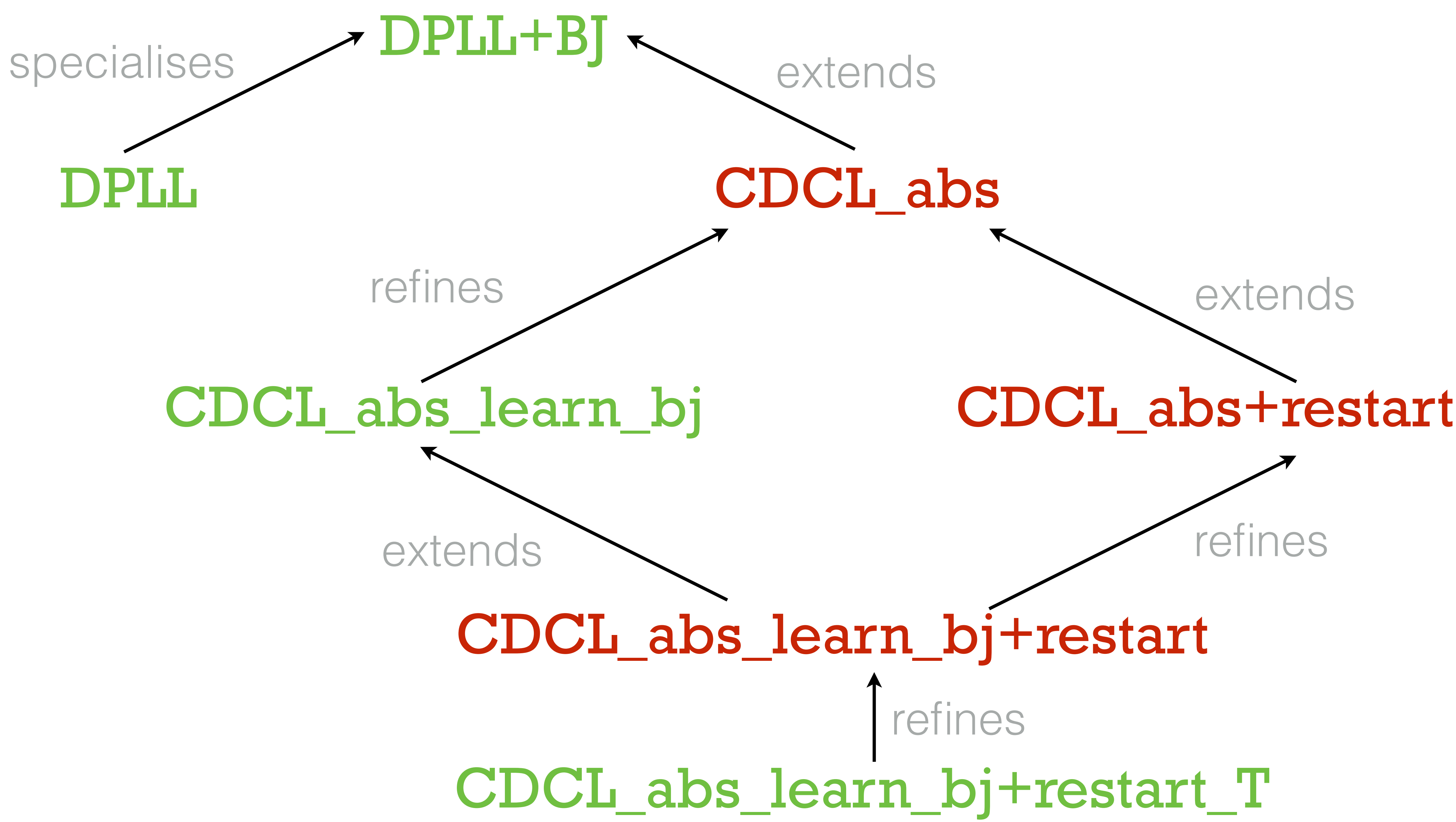
Restart Restart Restart Restart ...

CDCL_abs_learn_bj+restart_T



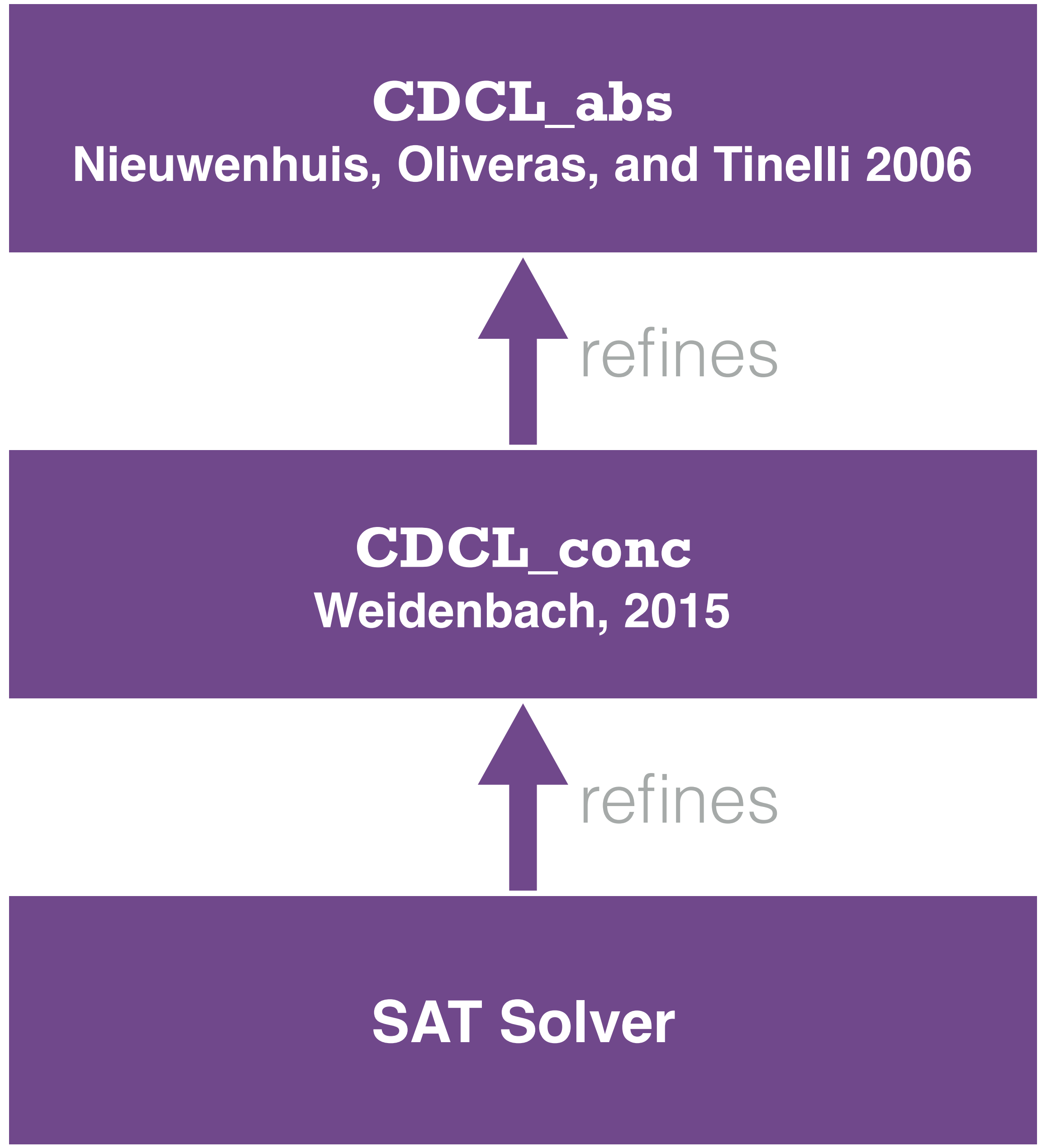
$(k_i)_i$ unbounded e.g., Luby sequence: 1, 1, 2, 1, 1, 2, 4, ...

Development hierarchy



Inria INVENTORS FOR THE DIGITAL WORLD

max planck institut informatik


CDCL_abs
Nieuwenhuis, Oliveras, and Tinelli 2006
refines
CDCL_conc
Weidenbach, 2015
refines
SAT Solver

CDCL_conc
Weidenbach, 2015

Parametrised Backjump(True)

on paper

if $C \in N$ and $M \models \neg C$ and
there is C' such that ...
then $(M, N) \Rightarrow_{\mathbf{DPLL+BJ}} (L^{\dagger} M', N)$

How do we get a suitable C'?

Inria INVENTORS FOR THE DIGITAL WORLD

max planck institut informatik

Parametrised Backjump(True)

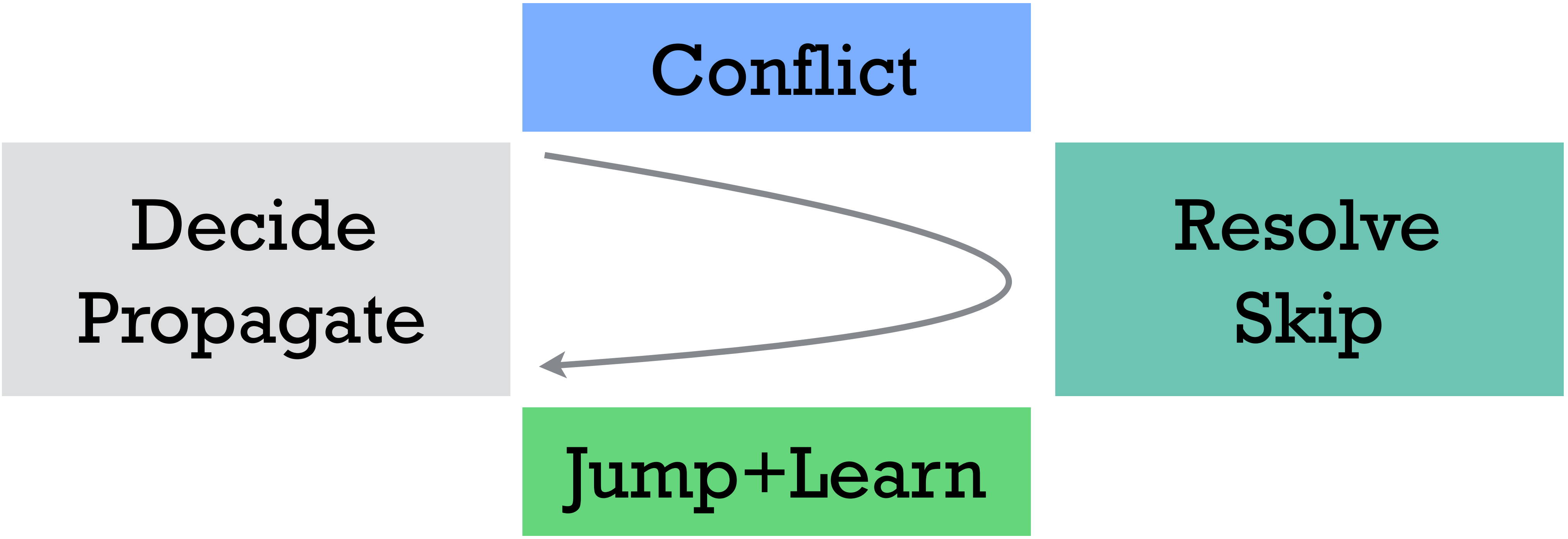
on paper

if $C \in N$ and $M \models \neg C$ and
there is C' such that ...
then $(M, N) \Rightarrow_{\text{DPLL+BJ}} (L^{\dagger} M', N)$

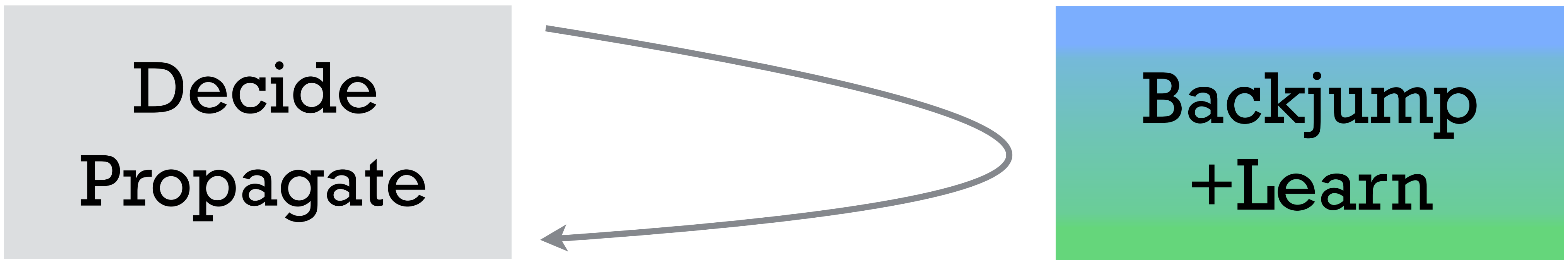
How do we get a suitable C'?

- First unique implication point

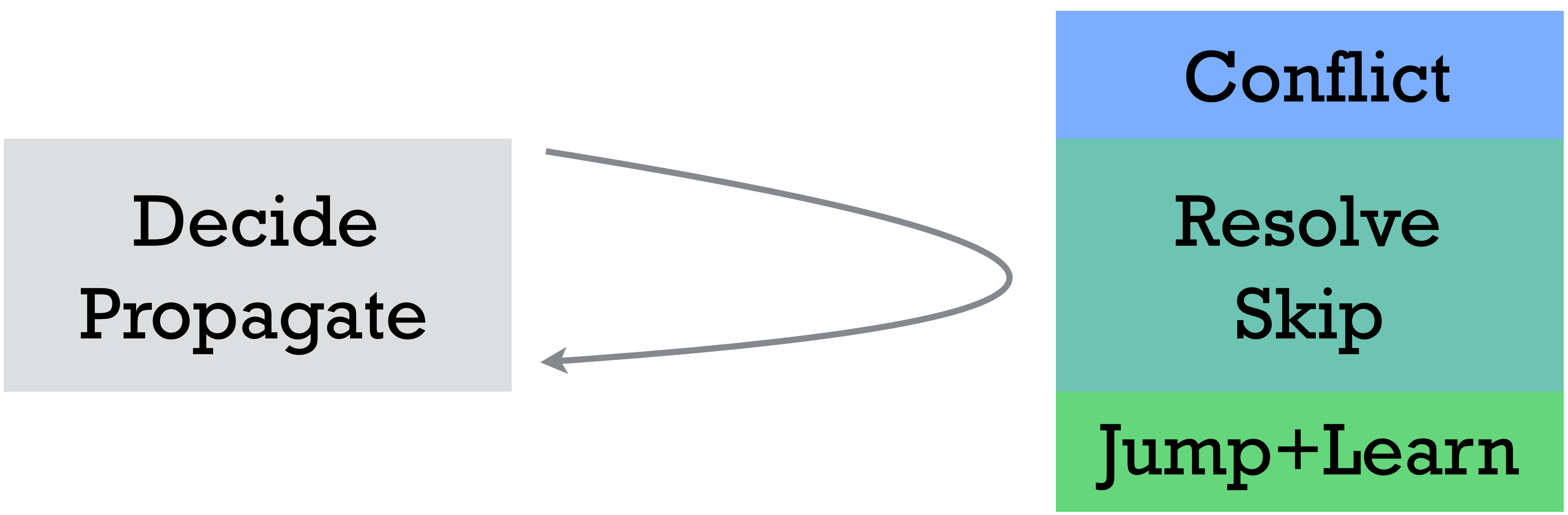
CDCL_conc



CDCL_abs_learn_bj



CDCL_conc



CDCL_abs_learn_bj

state: (M, N)

CDCL_conc

state: (M, N, U, k, C)

CDCL_abs_learn_bj terminates

state: (M, N)

$\Downarrow$

CDCL_conc terminates

state: (M, N, U, k, C)

Incremental SAT Solving

User

Calculus

Incremental SAT Solving

User

Start with these clauses

Satisfiable?

Calculus

Incremental SAT Solving

User

Start with these clauses

Satisfiable?

Calculus

Incremental SAT Solving

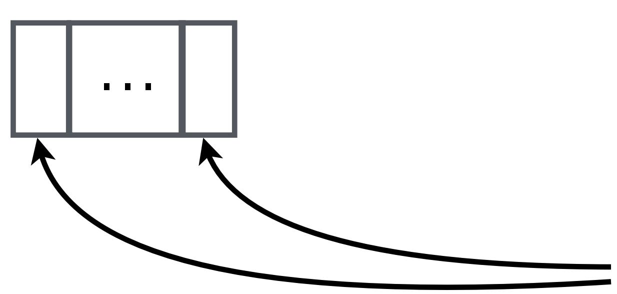
User

Start with these clauses

Satisfiable?

Calculus

I have a model



CDCL_conc

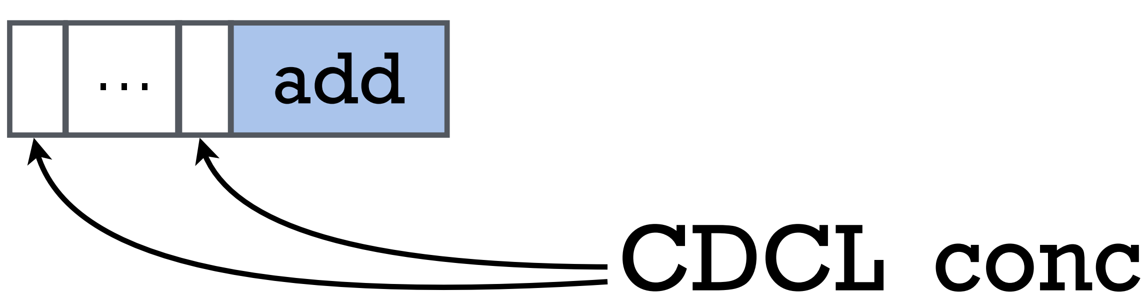
Incremental SAT Solving

User

Add C to the clauses

Still satisfiable?

Calculus



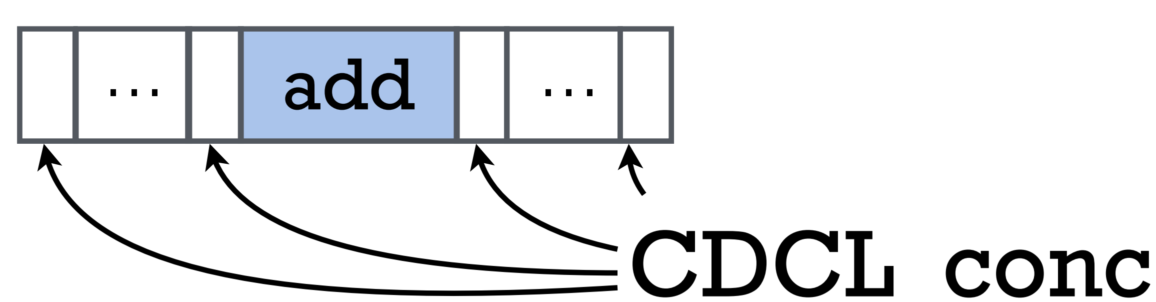
Incremental SAT Solving

User

Add C to the clauses

Still satisfiable?

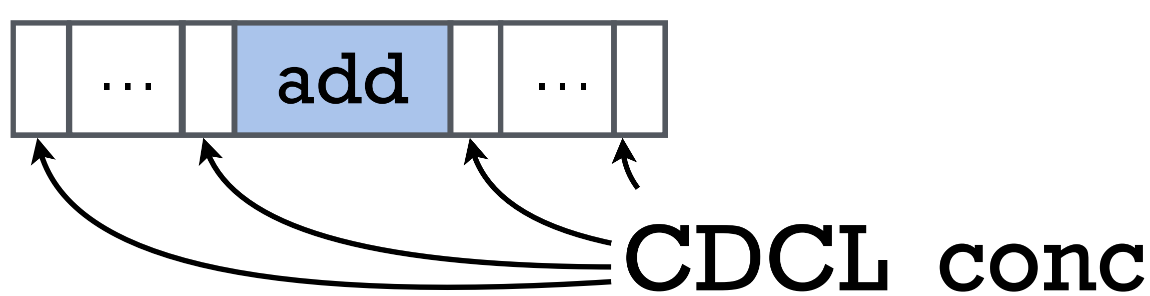
Calculus



Incremental SAT Solving

User — **Add C to the clauses** — **Still satisfiable?**

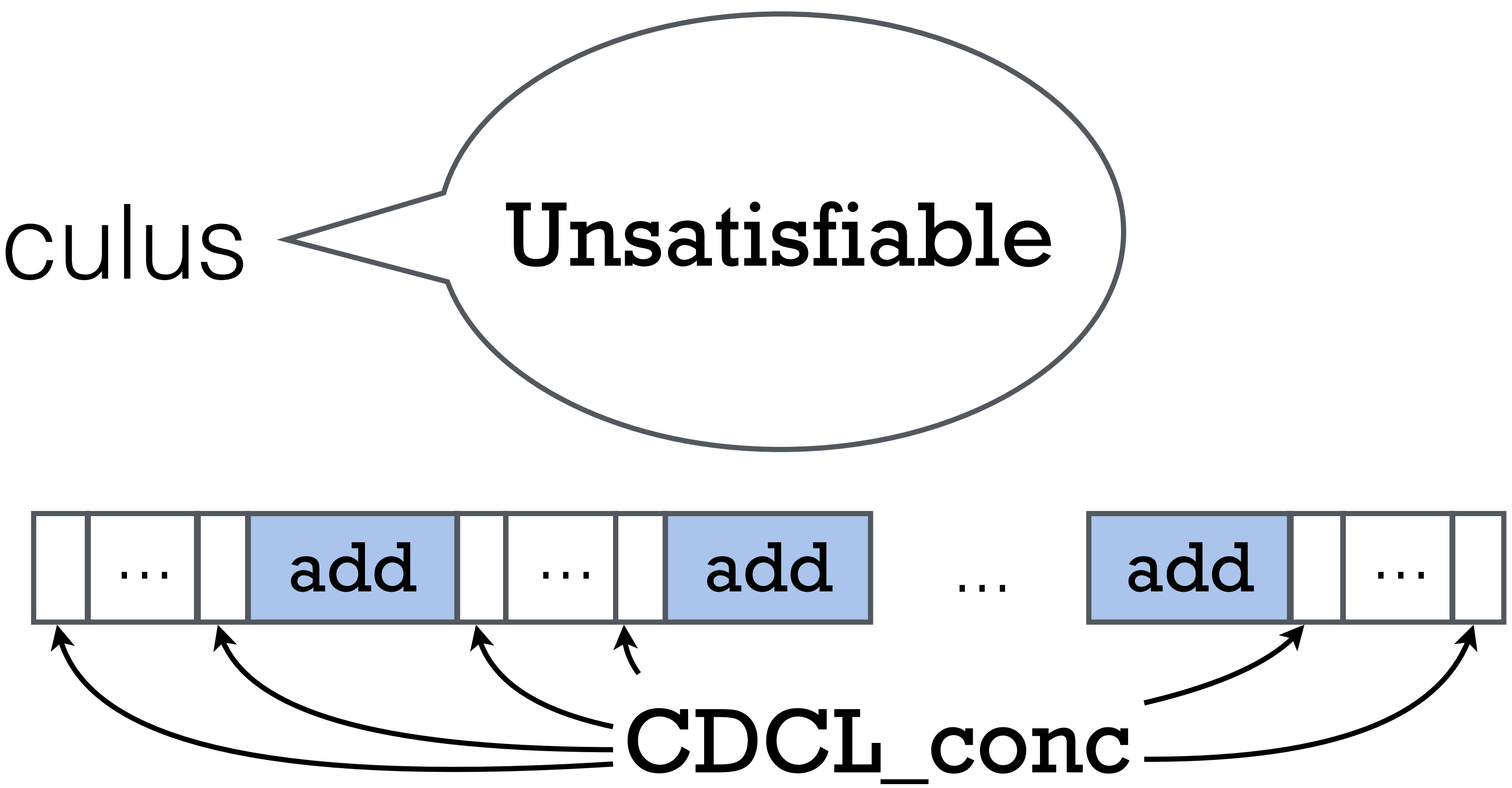
Calculus — **I have a model**



Incremental SAT Solving

User

Calculus

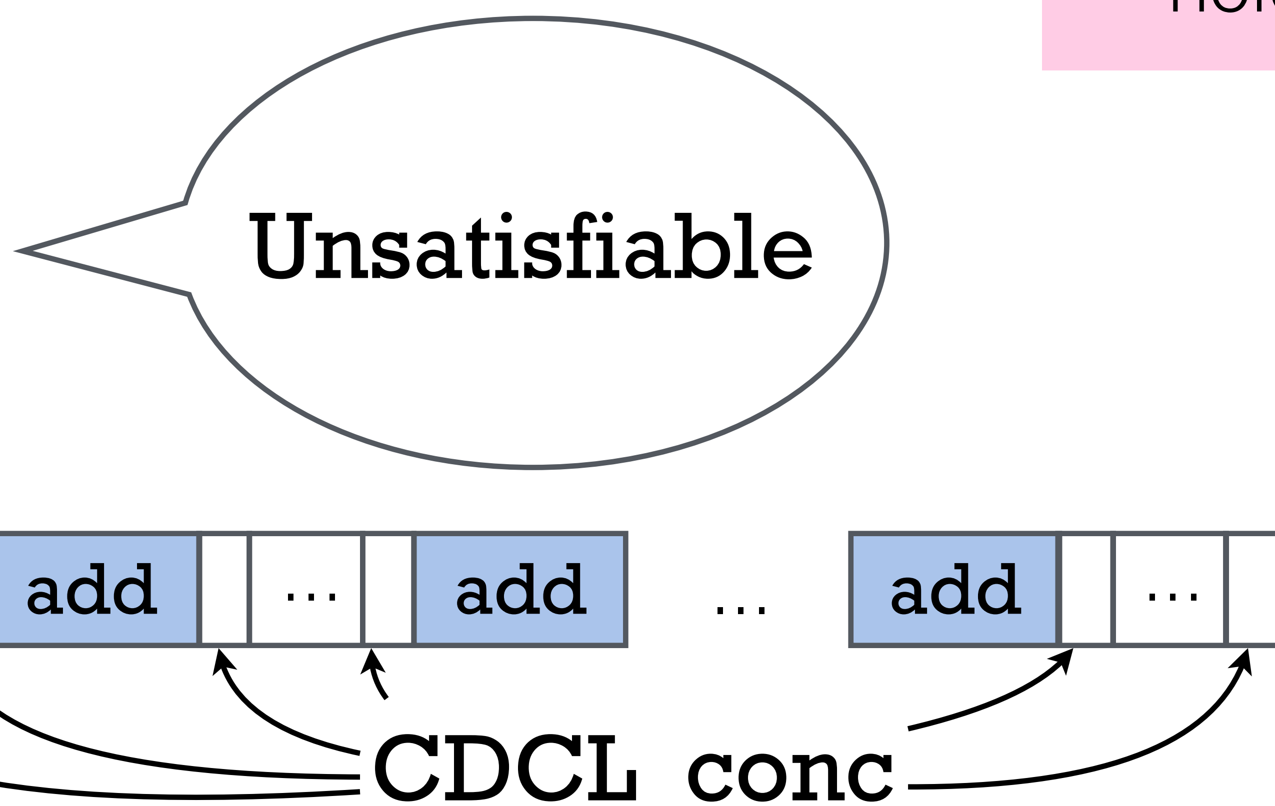


Incremental SAT Solving

User

Theorem: in Isabelle
‹invariants still hold after **add**›

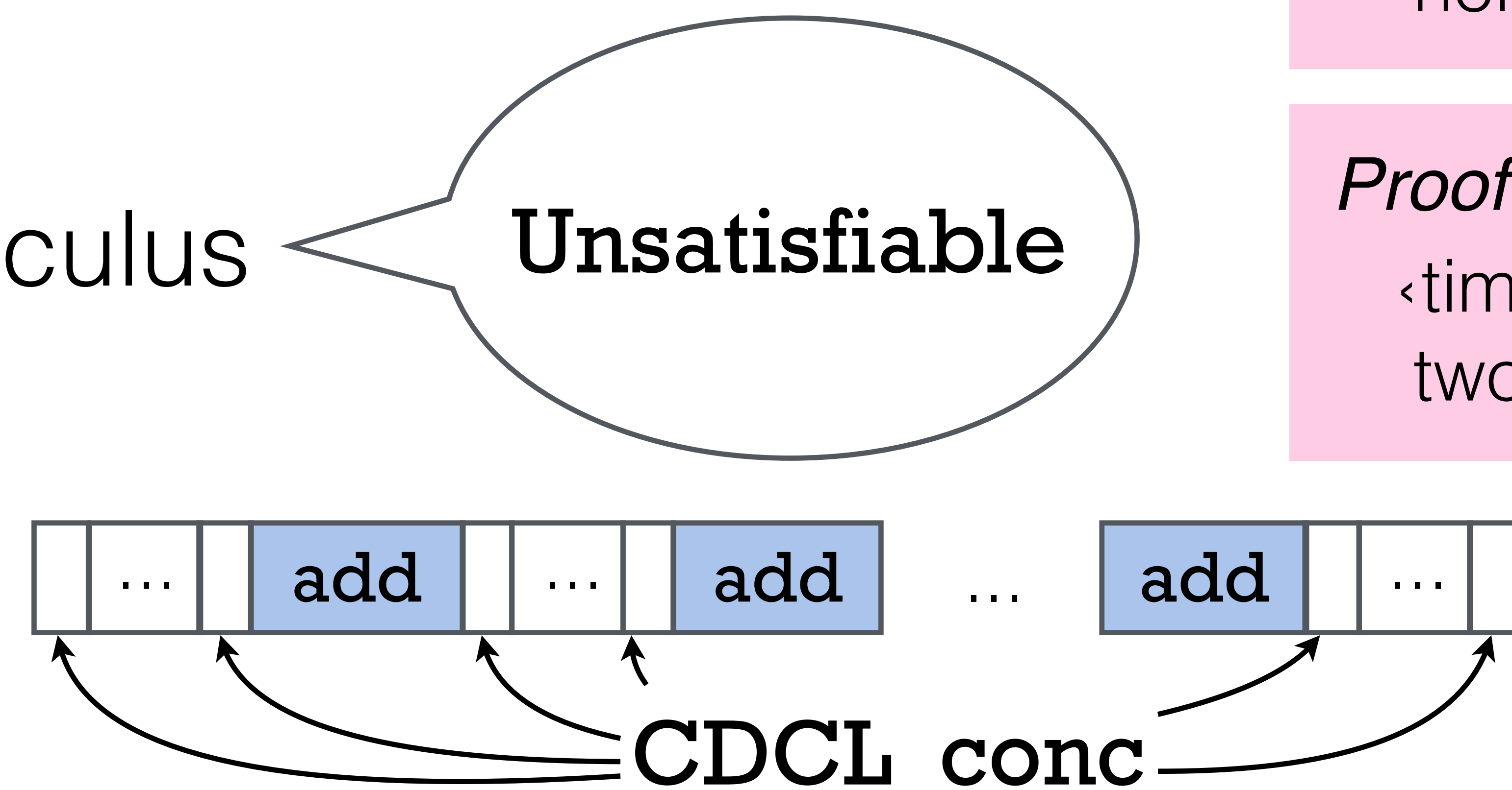
Calculus



Incremental SAT Solving

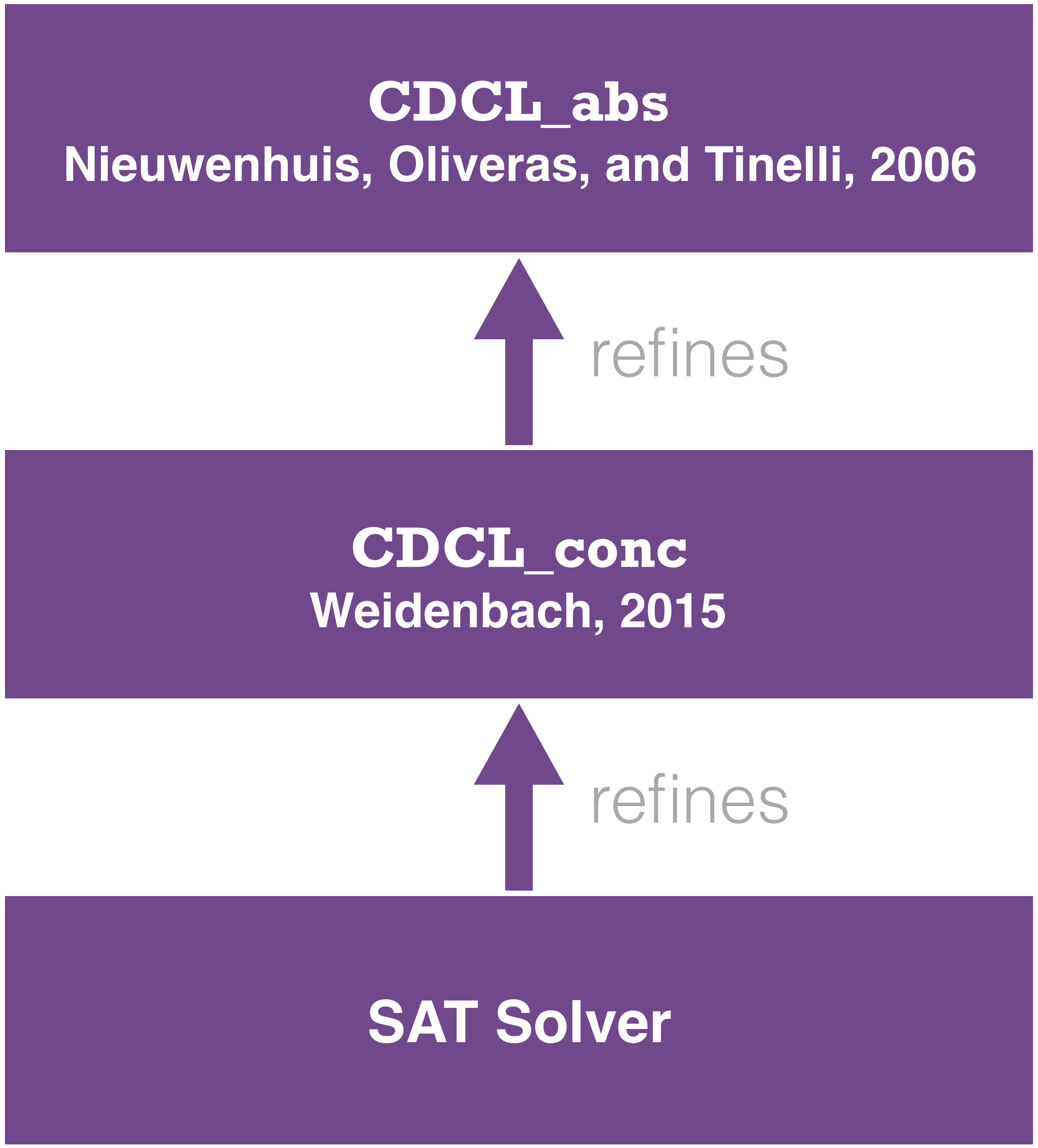
User

Calculus

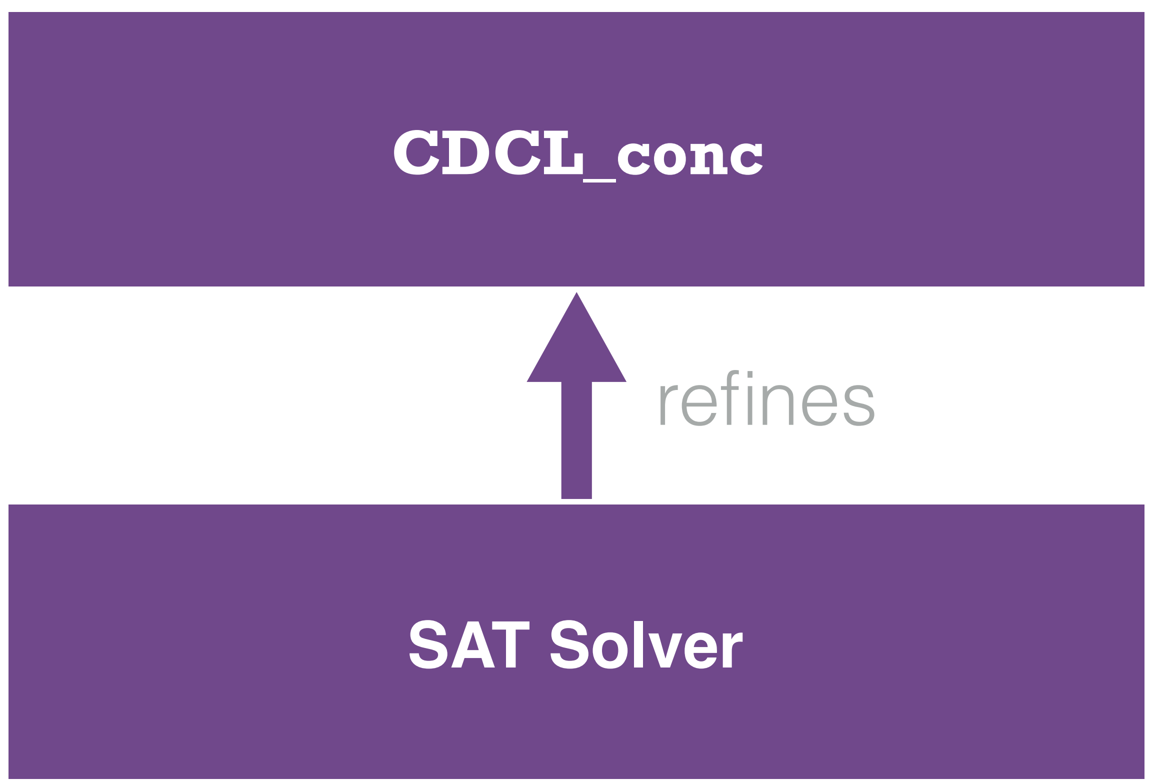


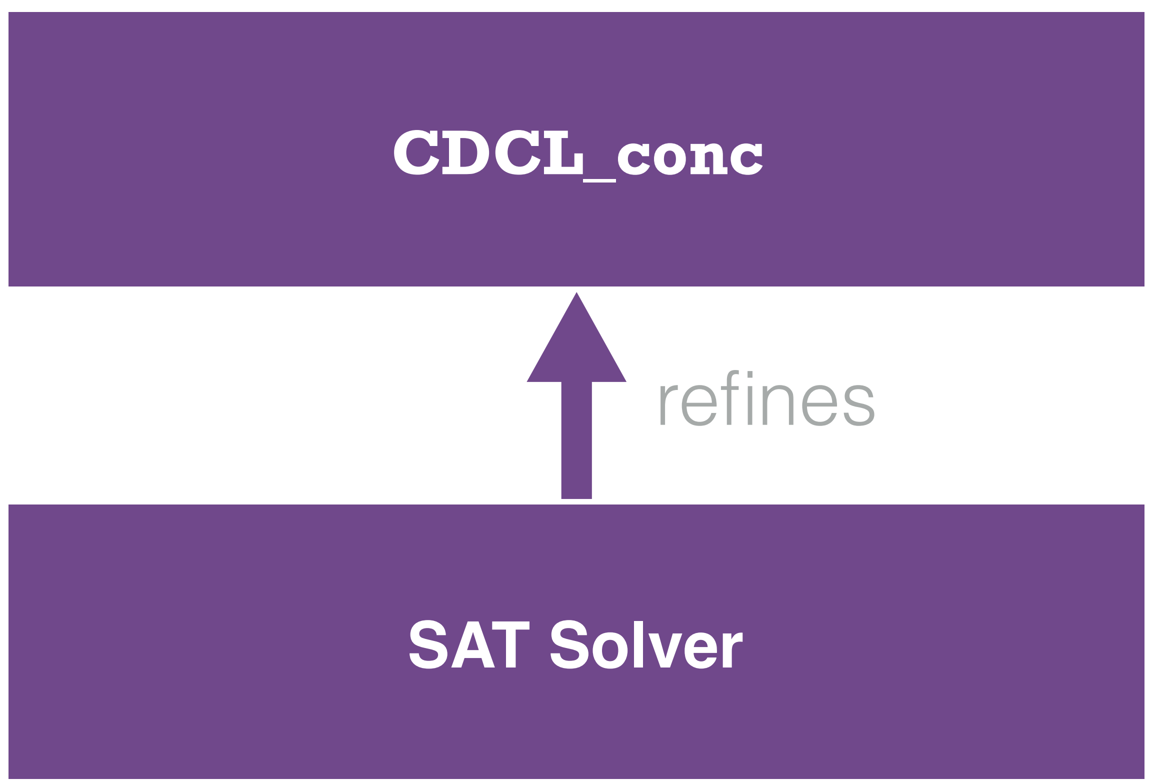
Theorem: in Isabelle
‹invariants still hold after **add**›

Proof. in Isabelle
‹time needed: two days›


CDCL_abs
Nieuwenhuis, Oliveras, and Tinelli, 2006
refines
CDCL_conc
Weidenbach, 2015
refines
SAT Solver

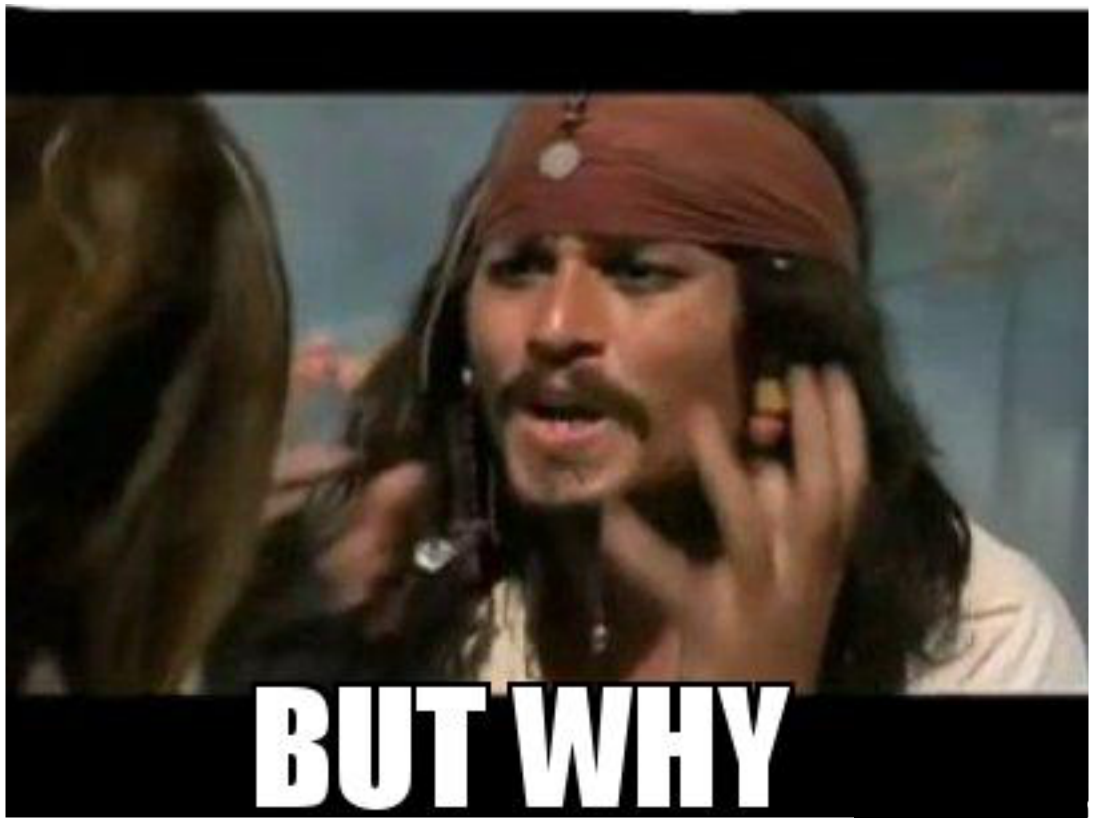
SAT Solver


CDCL_conc
refines
SAT Solver

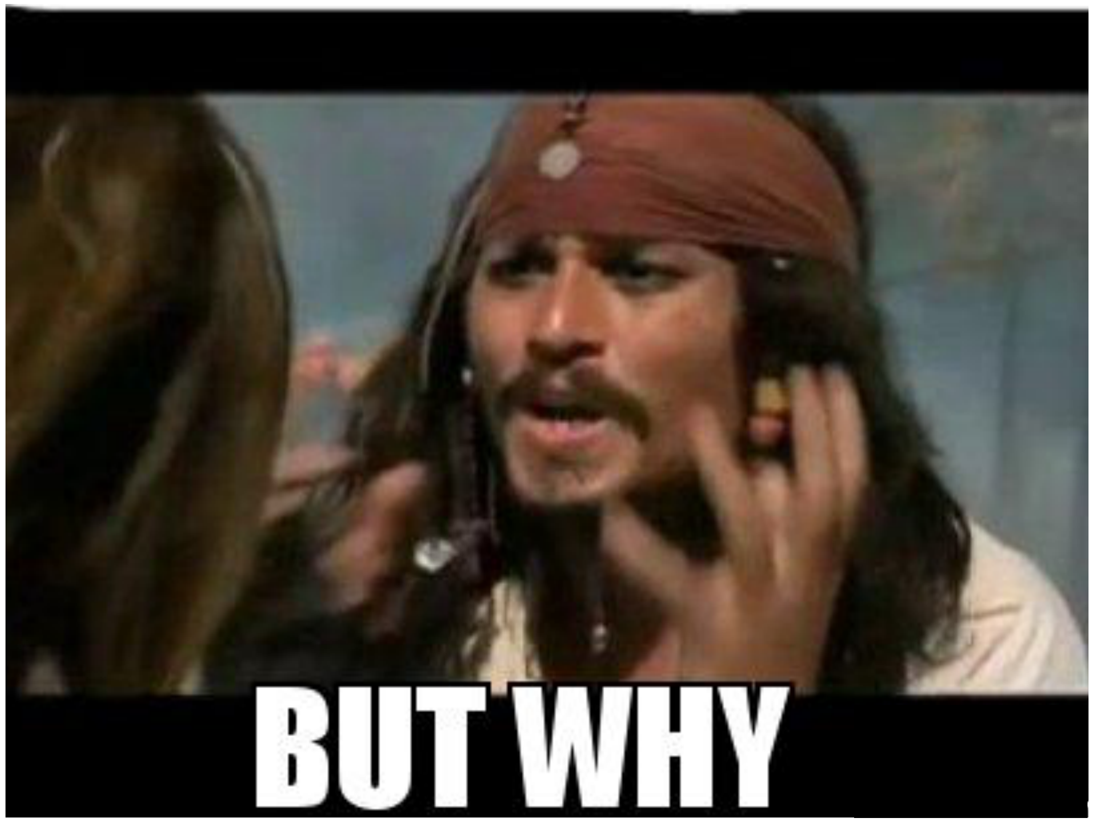


- inductive predicate (cf. Prolog)
- multiset based

- recursive (cf. ML)
- list based



The SAT community
right now



The SAT community
right now

- possible to continue refinement
- test for our framework
- rapid prototyping

What was hard?

Theorem (no relearning):
No clause can be learned twice.

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 $(M_1 K^{i+1} M'_2 K_1^k K_2 \ldots K_n; N; U; k; D \vee L)$ 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 $M_1 K^{i+1} M'_2 K_1^k K_2 \ldots K_n$ 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 $D \vee L$. □

on paper

‹700 lines of proof›

in Isabelle

Inria
INVENTORS FOR THE DIGITAL WORLD

max planck institut
informatik

Recall assumptions

Proof. By contradiction. Assume CDCL learns the same clause twice, i.e., it reaches a state $(M; N; U; k; D \lor L)$ where Backtracking is applicable and $D \lor L \in (N \cup U)$.

Speak about the current state

More precisely, the state has the form $(M_1 K^{i+1} M_2' K_1^k K_2 \ldots K_n; N; U; k; D \lor L)$ 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.

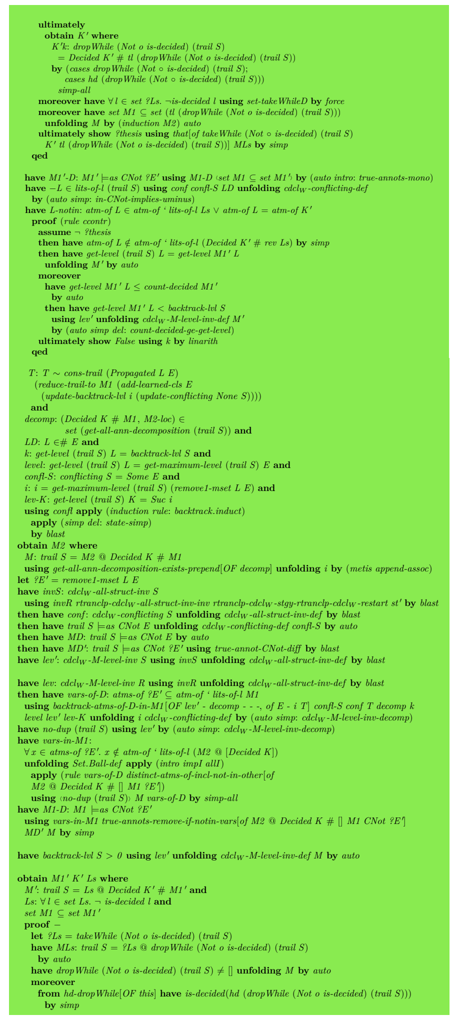
Speak about the past transitions

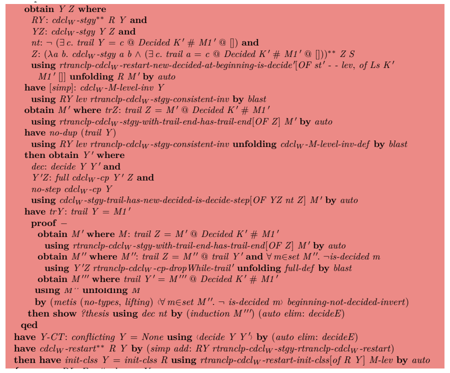
But now, because $D \lor L$ is false in $M_1 K^{i+1} M_2' K_1^k K_2 \ldots K_n$ and $D \lor L \in (N \cup U)$

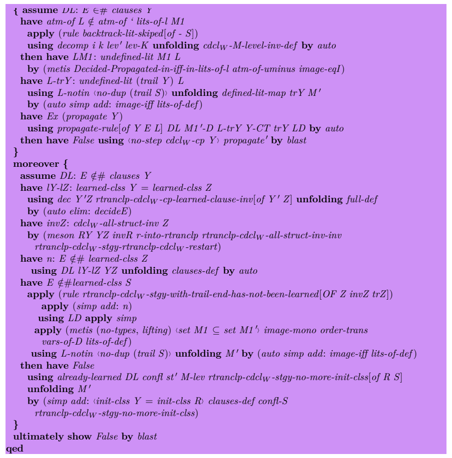
Find a Contradiction

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 $D \lor L$. □

```
lemma cdcl_W-stgy-no-relearned-clause:
  assumes
    invR: cdcl_W-all-struct-inv R and
    st': cdcl_W-stgy** R S and
    bt: backtrack S T and
    confl: conflicting S = Some E and
    already-learned: E ∈# clauses S and
    R: trail R = []
```





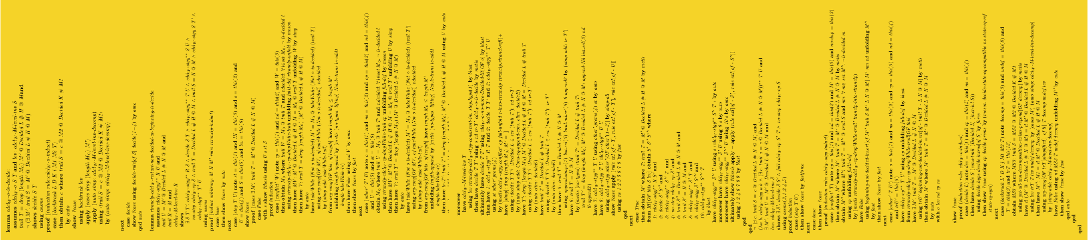


Assumptions | **Current state** | **Previous transitions** | **Contradiction**

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 $D \vee L \in (N \cup U)$

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 $D \vee L \in (N \cup U)$

There is a state such that...



But now, because $D \vee L$ is false in $M_1 K^{i+1} M_2' K_1^k K_2 \ldots K_n$ and $D \vee L \in (N \cup U)$

There is a state such that...

This is the last state such that...

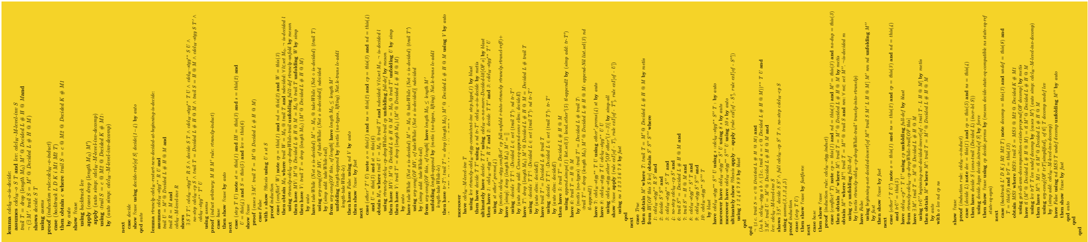
INVENTORS FOR THE DIGITAL WORLD

max planck institut informatik

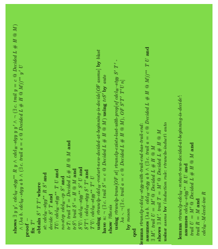
But now, because

$D \vee L$ is false in $M_1 K^{i+1} M_2' K_1^k K_2 \ldots K_n$ and $D \vee L \in (N \cup U)$

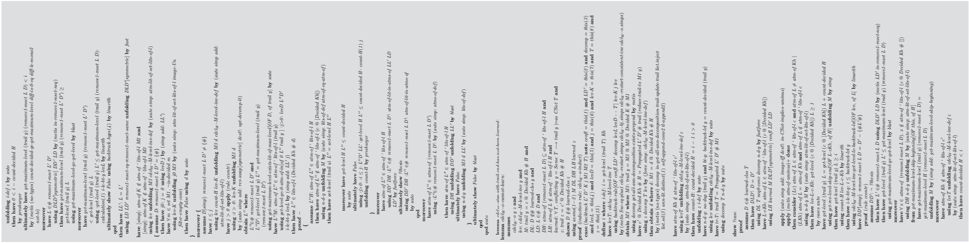
There is a state such that...



This is the last state such that...



It exists even in our strategy



How to reduce the effort?

The effort

	Paper	Proof assistant
CDCL_abs	13 pages	50 pages
CDCL_conc	9 pages (½ month)	90 pages (5 months)

How to reduce the effort?

The effort

	Paper	Proof assistant
CDCL_abs	13 pages	50 pages
CDCL_conc	9 pages (½ month)	90 pages (5 months)

Develop better automatic provers

- induction needed

How to reduce the effort?

The effort

	Paper	Proof assistant
CDCL_abs	13 pages	50 pages
CDCL_conc	9 pages (½ month)	90 pages (5 months)

Develop better automatic provers

- induction needed

Future work for audience

Inria INVENTORS FOR THE DIGITAL WORLD

max planck institut informatik

Related Work

	Maric 2008 Isabelle	Lescuyer 2011 Coq	Shankar and Vaucher 2011 PVS	Oe et al. 2012 Guru	This work 2016 Isabelle
Backjumping	✔	✔	✔	✔	✔
Learning	✘	✔	✔	✔	✔
Soundness	✔	✔	✔	✔	✔
Completeness	✔	✘	✔	✘	✔
Implementation	✔	✔	✘	✔	✔
Termination	✔	✔	✔	✘	✔
Restart + forget	✘	✘	✘	✘	✔
Incremental solving	✘	✘	✘	✘	✔

Related Work

	Maric 2008 Isabelle	Lescuyer 2011 Coq	Shankar and Vaucher 2011 PVS	Oe et al. 2012 Guru	This work 2016 Isabelle
Backjumping	✔	✔	✔	✔	✔
Learning	✘	✔	✔	✔	✔
Soundness	✔	✔	✔	✔	✔
Completeness	✔	✘	✔	✘	✔
Implementation	✔	✔	✘	✔	✔
Termination	✔	✔	✔	✘	✔
Restart + forget	✘	✘	✘	✘	✔
Incremental solving	✘	✘	✘	✘	✔
Two watched literals	✔	✘	✘	✔	(✘)

Conclusion

Concrete outcome

- verified SAT solver framework
- basic logic libraries (multisets, clauses)
- improve book draft

Methodology

- Isabelle: locales, Isar, Sledgehammer

Future work

- two watched literals, CDCL(T)