Learning Shorter Redundant Clauses in SDCL Using MaxSAT

Albert Oliveras[1], Chunxiao Li[2], Darryl Wu[2], Jonathan Chung[2], Vijay Ganesh[2]

[1]Universitat Politècnica de Catalunya

[2]University of Waterloo

SAT 2023, July 8th, l'Alguer

Outline of the Talk

Outline of the Talk

SAT, CDCL and Limitations

- SAT solving is a success story, with CDCL being the dominant method
- We like bragging: CDCL solves instances with millions of vars/clauses
- However, there are very small instances that CDCL will never solve efficiently (e.g. pigeon-hole, mutilated chessboard)
- These limitations are known due to:
  - Poly. equivalence between CDCL and resolution [AFT11, PD11]
  - Lower bounds on resolution proof sizes for some problems [Hak85]
- A natural research direction is to extend CDCL so that its underlying proof system is stronger than resolution.

Redundancy Notions

A proof system that is more powerful than resolution is based on redundancy:

Definition ([HKB17])

A clause C is **redundant** wrt F if and only if F and $F \wedge C$ are equisatisfiable

Theorem ([HKB17])

A clause C is **redundant** *wrt F if and only if there exists an assignment ω such that $\omega \models C$ and $F \wedge \neg C \models F|_\omega$ (where $F|_\omega$ is the result of removing from F all clauses satisfied by ω and all literals falsified by ω, i.e. simplify F wrt ω)*

But, even if we know ω (witness), it is hard to check that $F \wedge \neg C \models F|_\omega$.
This is why a weaker notion is considered:

Definition ([HKB17])

A clause C is **Propagation Redundant (PR)** wrt F if and only if there exists an assignment ω such that $\omega \models C$ and $F \wedge \neg C \vdash_1 F|_\omega$
(i.e. $\mathit{UnitProp}(F \wedge \neg C \wedge \neg F|_\omega)$ = conflict)

Redundancy Notions

A proof system that is more powerful than resolution is based on redundancy:

Definition ([HKB17])

A clause C is **redundant** wrt F if and only if F and $F \wedge C$ are equisatisfiable

Theorem ([HKB17])

A clause C is **redundant** *wrt F if and only if there exists an assignment ω such that $\omega \models C$ and $F \wedge \neg C \models F|_\omega$ (where $F|_\omega$ is the result of removing from F all clauses satisfied by ω and all literals falsified by ω, i.e. simplify F wrt ω)*

But, even if we know ω (witness), it is hard to check that $F \wedge \neg C \models F|_\omega$.
This is why a weaker notion is considered:

Definition ([HKB17])

A clause C is **Propagation Redundant (PR)** wrt F if and only if there exists an assignment ω such that $\omega \models C$ and $F \wedge \neg C \vdash_1 F|_\omega$
(i.e. $\mathit{UnitProp}(F \wedge \neg C \wedge \neg F|_\omega) = \text{conflict}$)

Redundancy Notions

A proof system that is more powerful than resolution is based on redundancy:

Definition ([HKB17])

A clause C is **redundant** wrt F if and only if F and $F \wedge C$ are equisatisfiable

Theorem ([HKB17])

A clause C is **redundant** *wrt F if and only if there exists an assignment ω such that $\omega \models C$ and $F \wedge \neg C \models F|_\omega$ (where $F|_\omega$ is the result of removing from F all clauses satisfied by ω and all literals falsified by ω, i.e. simplify F wrt ω)*

But, even if we know ω (witness), it is hard to check that $F \wedge \neg C \models F|_\omega$.
This is why a weaker notion is considered:

Definition ([HKB17])

A clause C is **Propagation Redundant (PR)** wrt F if and only if there exists an assignment ω such that $\omega \models C$ and $F \wedge \neg C \vdash_1 F|_\omega$
(i.e. $\mathit{UnitProp}(F \wedge \neg C \wedge \neg F|_\omega)$ = conflict)

Redundancy Notions (2)

Definition

A clause C is **Propagation Redundant (PR)** wrt F if and only if there exists an assignment ω such that $\omega \models C$ and $F \wedge \neg C \vdash_1 F|_\omega$
(i.e. $\mathit{UnitProp}(F \wedge \neg C \wedge \neg F|_\omega)$ = conflict)

Particular cases of PR clauses:

- SPR: ω assigns only variables of C
- LPR/RAT: ω assigns only vars of C and satisfies exactly one lit of C
- Set-blocked [KSTB16]: particular case of SPR, syntactic definition
- Blocked [JBH10]: particular case of set-blocked, syntactic definition

Outline of the Talk

From CDCL to SDCL

Conflict-Driven Clause-Learning (**CDCL**) can be seen as the right way to implement a resolution-based proof procedure:

$\alpha := \emptyset$
while *true* **do**
    $\alpha :=$ *unitPropagate*(F, α)
    if *conflict found* **then**
        $C :=$ *analyzeConflict*$()$
        $F := F \wedge C$
        if *C is the empty clause* **then** return UNSAT
        $\alpha :=$ *backjump*(C, α)
    else
        if *all variables are assigned* **then** return SAT
        $\alpha := \alpha \cup$ *Decide*$()$

From CDCL to SDCL

Satisfiability-Driven Clause-Learning (**SDCL**) [HKSB17]) is a way to implement a redundancy-based proof procedure:

$\alpha := \emptyset$
while *true* **do**
  $\alpha := \mathit{unitPropagate}(F, \alpha)$
  if *conflict found* **then**
    $C := \mathit{analyzeConflict}()$
    $F := F \wedge C$
    if *C is the empty clause* **then** return UNSAT
    $\alpha := \mathit{backjump}(C, \alpha)$
  else if $\neg\alpha$ *is redundant wrt F* **then**
    $C := \mathit{decisions}(\neg\alpha)$
    $F := F \wedge C$
    $\alpha := \mathit{backjump}(C, \alpha)$
  else
    if *all variables are assigned* **then** return SAT
    $\alpha := \alpha \cup \mathit{Decide}()$

From CDCL to SDCL

SDCL is a way to implement a redundancy-based proof procedure:

$\alpha := \emptyset$
while *true* **do**
  $\alpha := \mathit{unitPropagate}(F, \alpha)$
  if *conflict found* **then**
    $C := \mathit{analyzeConflict}()$
    $F := F \wedge C$
    if *C is the empty clause* **then** return UNSAT
    $\alpha := \mathit{backjump}(C, \alpha)$
  else if $P_\alpha(F)$ *is satisfiable* **then**
    $C := \mathit{decisions}(\neg\alpha)$
    $F := F \wedge C$
    $\alpha := \mathit{backjump}(C, \alpha)$
  else
    if *all variables are assigned* **then** return SAT
    $\alpha := \alpha \cup \mathit{Decide}()$

Definition ([HKSB17])

A **pruning predicate** for F and α is a formula $P_\alpha(F)$ such that, if $P_\alpha(F)$ is satisfiable then clause $\neg\alpha$ is redundant wrt F

Pruning Predicates

Definition

A **pruning predicate** for F and α is a formula $P_\alpha(F)$ such that, if $P_\alpha(F)$ is satisfiable then clause $\neg\alpha$ is redundant wrt F

Do we know how to construct pruning predicates?

- Purely positive reduct: formula $\neg\alpha \wedge G$, where $G = \{\mathit{satisfied}_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$.
- Positive reduct: formula $\neg\alpha \wedge G$, where $G = \{\mathit{touched}_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$. **[The one used in this work.]**
- Filtered positive reduct: formula $\neg\alpha \wedge G$, where $G = \{\mathit{touched}_\alpha(D) \mid D \in F \text{ and } F \wedge \alpha \not\vdash_1 \mathit{untouched}_\alpha(D)\}$.

Proposition (Contribution of this paper)

The purely positive reduct is satisfiable if and only if $\neg\alpha$ is blocked in F.

Pruning Predicates

Definition

A **pruning predicate** for F and α is a formula $P_\alpha(F)$ such that, if $P_\alpha(F)$ is satisfiable then clause $\neg\alpha$ is redundant wrt F

Do we know how to construct pruning predicates?

- Purely positive reduct: formula $\neg\alpha \wedge G$, where $G = \{\mathit{satisfied}_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$.
- Positive reduct: formula $\neg\alpha \wedge G$, where $G = \{\mathit{touched}_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$. **[The one used in this work.]**
- Filtered positive reduct: formula $\neg\alpha \wedge G$, where $G = \{\mathit{touched}_\alpha(D) \mid D \in F \text{ and } F \wedge \alpha \not\vdash_1 \mathit{untouched}_\alpha(D)\}$.

Proposition ([HKSB17])

The positive reduct is satisfiable if and only if $\neg\alpha$ is set-blocked in F.

Pruning Predicates

Definition

A **pruning predicate** for F and α is a formula $P_\alpha(F)$ such that, if $P_\alpha(F)$ is satisfiable then clause $\neg\alpha$ is redundant wrt F

Do we know how to construct pruning predicates?

- Purely positive reduct: formula $\neg\alpha \wedge G$, where $G = \{\mathit{satisfied}_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$.
- Positive reduct: formula $\neg\alpha \wedge G$, where $G = \{\mathit{touched}_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$. **[The one used in this work.]**
- Filtered positive reduct: formula $\neg\alpha \wedge G$, where $G = \{\mathit{touched}_\alpha(D) \mid D \in F \text{ and } F \wedge \alpha \not\vdash_1 \mathit{untouched}_\alpha(D)\}$.

Proposition ([HKB19])

The filtered positive reduct is satisfiable if and only if $\neg\alpha$ is SPR in F.

Outline of the Talk

Motivation

$\alpha := \emptyset$
while *true* **do**
  $\alpha := \mathit{unitPropagate}(F, \alpha)$
  if *conflict found* **then**
    $C := \mathit{analyzeConflict}()$
    $F := F \wedge C$
    if *C is the empty clause* **then** return UNSAT
    $\alpha := \mathit{backjump}(C, \alpha)$
  else if $P_\alpha(F)$ *is satisfiable* **then**
    $C := \mathit{decisions}(\neg\alpha)$
    $F := F \wedge C$
    $\alpha := \mathit{backjump}(C, \alpha)$
  else
    if *all variables are assigned* **then** return SAT
    $\alpha := \alpha \cup \mathit{Decide}()$

In the original SDCL design, the redundant clause to be learned is the negation of all decisions in α.

Our experience from CDCL suggests that this is not the best choice.

Motivation - Example

Positive reduct

$p_\alpha(F) = \neg\alpha \wedge G$, where $G = \{touched_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$.

Let us consider F and an assignment α built by the SAT solver:

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\}$$

F

$$\begin{array}{llllll} x_1 & \vee & x_2 & \vee & x_4 \\ x_1 & \vee & \neg x_2 & \vee & x_5 \\ \neg x_1 & \vee & \neg x_4 & \vee & x_5 \\ x_2 & \vee & x_4 & \vee & \neg x_5 \\ x_3 & \vee & x_6 & \vee & \neg x_5 \end{array}$$

$p_\alpha(F)$

$$\begin{array}{llllll} \neg x_1 & \vee & \neg x_4 & \vee & \neg x_5 & \vee & x_2 \\ x_1 & \vee & x_2 & \vee & x_4 \\ x_1 & \vee & \neg x_2 & \vee & x_5 \\ \neg x_1 & \vee & \neg x_4 & \vee & x_5 \\ x_2 & \vee & x_4 & \vee & \neg x_5 \end{array}$$

Since $p_\alpha(F)$ is satisfiable ($\{x_1, x_2\}$) we can learn a clause $\neg x_1 \vee \neg x_4 \vee x_2$

Motivation - Example

Positive reduct

$p_\alpha(F) = \neg\alpha \wedge G$, where $G = \{touched_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$.

Let us consider F and an assignment α built by the SAT solver:

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\}$$

F

$$\begin{array}{lllll} x_1 & \vee & x_2 & \vee & x_4 \\ x_1 & \vee & \neg x_2 & \vee & x_5 \\ \neg x_1 & \vee & \neg x_4 & \vee & x_5 \\ x_2 & \vee & x_4 & \vee & \neg x_5 \\ x_3 & \vee & x_6 & \vee & \neg x_5 \end{array}$$

$p_\alpha(F)$

$$\begin{array}{lllll} \neg x_1 & \vee & \neg x_4 & \vee & \neg x_5 \vee x_2 \\ x_1 & \vee & x_2 & \vee & x_4 \\ x_1 & \vee & \neg x_2 & \vee & x_5 \\ \neg x_1 & \vee & \neg x_4 & \vee & x_5 \\ x_2 & \vee & x_4 & \vee & \neg x_5 \end{array}$$

Since $p_\alpha(F)$ is satisfiable (e.g. $\{x_1, x_2\}$) we can learn a clause $\neg x_1 \vee \neg x_4 \vee x_2$

Motivation - Example

Positive reduct

$p_\alpha(F) = \neg\alpha \wedge G$, where $G = \{touched_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$.

Let us consider F and an assignment α built by the SAT solver:

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\}$$

F	$p_\alpha(F)$
$x_1 \vee x_2 \vee x_4$	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$
$x_1 \vee \neg x_2 \vee x_5$	$x_1 \vee x_2 \vee x_4$
$\neg x_1 \vee \neg x_4 \vee x_5$	$x_1 \vee \neg x_2 \vee x_5$
$x_2 \vee x_4 \vee \neg x_5$	$\neg x_1 \vee \neg x_4 \vee x_5$
$x_3 \vee x_6 \vee \neg x_5$	$x_2 \vee x_4 \vee \neg x_5$

Since $p_\alpha(F)$ is satisfiable (e.g. $\{x_1, x_2\}$) we can learn a clause $\neg x_1 \vee \neg x_2$

Motivation - Example

Positive reduct

$p_\alpha(F) = \neg\alpha \wedge G$, where $G = \{touched_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$.

Let us consider F and an assignment α built by the SAT solver:

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\}$$

F	$p_\alpha(F)$
	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$
$x_1 \vee x_2 \vee x_4$	$x_1 \vee x_2 \vee x_4$
$x_1 \vee \neg x_2 \vee x_5$	$x_1 \vee \neg x_2 \vee x_5$
$\neg x_1 \vee \neg x_4 \vee x_5$	$\neg x_1 \vee \neg x_4 \vee x_5$
$x_2 \vee x_4 \vee \neg x_5$	$x_2 \vee x_4 \vee \neg x_5$
$x_3 \vee x_6 \vee \neg x_5$	

Since $p_\alpha(F)$ is satisfiable ($\{x_1, x_2, x_5\}$) we can learn clause $\neg x_1 \vee \neg x_4 \vee x_2$

Motivation - Example

Positive reduct

$p_\alpha(F) = \neg\alpha \wedge G$, where $G = \{touched_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$.

Let us consider F and an assignment α built by the SAT solver:

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\}$$

F	$p_\alpha(F)$
	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$
$x_1 \vee x_2 \vee x_4$	$x_1 \vee x_2 \vee x_4$
$x_1 \vee \neg x_2 \vee x_5$	$x_1 \vee \neg x_2 \vee x_5$
$\neg x_1 \vee \neg x_4 \vee x_5$	$\neg x_1 \vee \neg x_4 \vee x_5$
$x_2 \vee x_4 \vee \neg x_5$	$x_2 \vee x_4 \vee \neg x_5$
$x_3 \vee x_6 \vee \neg x_5$	

Since $p_\alpha(F)$ is satisfiable ($\{x_1, x_2, x_5\}$) we can learn clause $\neg x_1 \vee \neg x_4 \vee x_2$

Motivation - Example

Positive reduct

$p_\alpha(F) = \neg\alpha \wedge G$, where $G = \{touched_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$.

Let us consider F and an assignment α built by the SAT solver:

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\}$$

F	$p_\alpha(F)$
	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$
$x_1 \vee x_2 \vee x_4$	$x_1 \vee x_2 \vee x_4$
$x_1 \vee \neg x_2 \vee x_5$	$x_1 \vee \neg x_2 \vee x_5$
$\neg x_1 \vee \neg x_4 \vee x_5$	$\neg x_1 \vee \neg x_4 \vee x_5$
$x_2 \vee x_4 \vee \neg x_5$	$x_2 \vee x_4 \vee \neg x_5$
$x_3 \vee x_6 \vee \neg x_5$	

Since $p_\alpha(F)$ is satisfiable ($\{x_1, x_2, x_5\}$) we can learn clause $\neg x_1 \vee \neg x_4 \vee x_2$

Motivation - Example

Positive reduct

$p_\alpha(F) = \neg\alpha \wedge G$, where $G = \{touched_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$.

Let us now consider another assignment $\gamma \subseteq \alpha$:

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\} \qquad \gamma = \{x_1, \neg x_2\}$$

F	$p_\alpha(F)$	$p_\gamma(F)$
$x_1 \vee x_2 \vee x_4$	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$	$\neg x_1 \vee x_2$
$x_1 \vee \neg x_2 \vee x_5$	$x_1 \vee x_2 \vee x_4$	$x_1 \vee x_2$
$\neg x_1 \vee \neg x_4 \vee x_5$	$x_1 \vee \neg x_2 \vee x_5$	$x_1 \vee \neg x_2$
$x_2 \vee x_4 \vee \neg x_5$	$\neg x_1 \vee \neg x_4 \vee x_5$	
$x_3 \vee x_6 \vee \neg x_5$	$x_2 \vee x_4 \vee \neg x_5$	

Since $p_\alpha(F)$ is satisfiable ($\{x_1, x_2, x_5\}$) we can learn clause $\neg x_1 \vee \neg x_4 \vee x_2$.

Motivation - Example

Positive reduct

$p_\alpha(F) = \neg\alpha \wedge G$, where $G = \{touched_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$.

Let us now consider another assignment $\gamma \subseteq \alpha$:

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\} \qquad \gamma = \{x_1, \neg x_2\}$$

F	$p_\alpha(F)$	$p_\gamma(F)$
$x_1 \vee x_2 \vee x_4$	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$	$\neg x_1 \vee x_2$
$x_1 \vee \neg x_2 \vee x_5$	$x_1 \vee x_2 \vee x_4$	$x_1 \vee x_2$
$\neg x_1 \vee \neg x_4 \vee x_5$	$x_1 \vee \neg x_2 \vee x_5$	$x_1 \vee \neg x_2$
$x_2 \vee x_4 \vee \neg x_5$	$\neg x_1 \vee \neg x_4 \vee x_5$	
$x_3 \vee x_6 \vee \neg x_5$	$x_2 \vee x_4 \vee \neg x_5$	

Since $p_\alpha(F)$ is satisfiable ($\{x_1, x_2, x_5\}$) we can learn clause $\neg x_1 \vee \neg x_4 \vee x_2$.

Since $p_\gamma(F)$ is satisfiable ($\{x_1, x_2\}$) we can learn clause $\neg x_1 \vee x_2$.

Motivation - Example

Positive reduct

$p_\alpha(F) = \neg\alpha \wedge G$, where $G = \{touched_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$.

Let us now consider another assignment $\gamma \subseteq \alpha$:

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\} \qquad \gamma = \{x_1, \neg x_2\}$$

F	$p_\alpha(F)$	$p_\gamma(F)$
$x_1 \vee x_2 \vee x_4$	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$	$\neg x_1 \vee x_2$
$x_1 \vee \neg x_2 \vee x_5$	$x_1 \vee x_2 \vee x_4$	$x_1 \vee x_2$
$\neg x_1 \vee \neg x_4 \vee x_5$	$x_1 \vee \neg x_2 \vee x_5$	$x_1 \vee \neg x_2$
$x_2 \vee x_4 \vee \neg x_5$	$\neg x_1 \vee \neg x_4 \vee x_5$	
$x_3 \vee x_6 \vee \neg x_5$	$x_2 \vee x_4 \vee \neg x_5$	

Since $p_\alpha(F)$ is satisfiable ($\{x_1, x_2, x_5\}$) we can learn clause $\neg x_1 \vee \neg x_4 \vee x_2$.

Since $p_\gamma(F)$ is satisfiable ($\{x_1, x_2\}$) we can learn clause $\neg x_1 \vee x_2$

Motivation - Example

Positive reduct

$p_\alpha(F) = \neg\alpha \wedge G$, where $G = \{touched_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$.

Let us now consider another assignment $\gamma \subseteq \alpha$:

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\} \qquad \gamma = \{x_1, \neg x_2\}$$

F	$p_\alpha(F)$	$p_\gamma(F)$
$x_1 \vee x_2 \vee x_4$	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$	$\neg x_1 \vee x_2$
$x_1 \vee \neg x_2 \vee x_5$	$x_1 \vee x_2 \vee x_4$	$x_1 \vee x_2$
$\neg x_1 \vee \neg x_4 \vee x_5$	$x_1 \vee \neg x_2 \vee x_5$	$x_1 \vee \neg x_2$
$x_2 \vee x_4 \vee \neg x_5$	$\neg x_1 \vee \neg x_4 \vee x_5$	
$x_3 \vee x_6 \vee \neg x_5$	$x_2 \vee x_4 \vee \neg x_5$	

Since $p_\alpha(F)$ is satisfiable ($\{x_1, x_2, x_5\}$) we can learn clause $\neg x_1 \vee \neg x_4 \vee x_2$.

Since $p_\gamma(F)$ is satisfiable ($\{x_1, x_2\}$) we can learn clause $\neg x_1 \vee x_2$

Motivation - Example

Positive reduct

$p_\alpha(F) = \neg\alpha \wedge G$, where $G = \{touched_\alpha(D) \mid D \in F \text{ and } \alpha \models D\}$.

Let us now consider another assignment $\gamma \subseteq \alpha$:

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\} \qquad \gamma = \{x_1, \neg x_2\}$$

F	$p_\alpha(F)$	$p_\gamma(F)$
$x_1 \vee x_2 \vee x_4$	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$	$\neg x_1 \vee x_2$
$x_1 \vee \neg x_2 \vee x_5$	$x_1 \vee x_2 \vee x_4$	$x_1 \vee x_2$
$\neg x_1 \vee \neg x_4 \vee x_5$	$x_1 \vee \neg x_2 \vee x_5$	$x_1 \vee \neg x_2$
$x_2 \vee x_4 \vee \neg x_5$	$\neg x_1 \vee \neg x_4 \vee x_5$	
$x_3 \vee x_6 \vee \neg x_5$	$x_2 \vee x_4 \vee \neg x_5$	

Since $p_\alpha(F)$ is satisfiable ($\{x_1, x_2, x_5\}$) we can learn clause $\neg x_1 \vee \neg x_4 \vee x_2$.

Since $p_\gamma(F)$ is satisfiable ($\{x_1, x_2\}$) we can learn clause $\neg x_1 \vee x_2$

Subsets $\gamma \subseteq \alpha$ for which $p_\gamma(F)$ is satisfiable give shorter redundant clauses.

Difficulty of the Problem

Definition

TRAIL-MINIMIZATION: given a formula F, an assignment α and an integer $k \geq 0$, we want to know whether there is a subset $\gamma \subseteq \alpha$ of size k such that $p_\gamma(F)$ is satisfiable.

Theorem

TRAIL-MINIMIZATION *is NP-hard*

A natural way to solve TRAIL-MINIMIZATION is via a MaxSAT encoding

A MaxSAT Encoding for TRAIL-MINIMIZATION

Let $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_k\}$. For each literal $\alpha_i \in \alpha$, we introduce three variables $\{r_i, p_i, n_i\}$. We will add soft clauses $[r_i]$ and hard clauses:

$$\begin{aligned} \neg r_i &\rightarrow p_i = \alpha_i \\ \neg r_i &\rightarrow n_i = \neg\alpha_i \\ r_i &\rightarrow \neg p_i \\ r_i &\rightarrow \neg n_i \end{aligned}$$

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\}$$

F	$p_\alpha(F)$	MaxSAT encoding
	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$	$n_1 \vee n_4 \vee n_5 \vee n_2$
$x_1 \vee x_2 \vee x_4$	$\mathbf{x_1} \vee x_2 \vee \mathbf{x_4}$	$p_1 \vee n_2 \vee p_4 \vee r_1$
		$p_1 \vee n_2 \vee p_4 \vee r_4$
$x_1 \vee \neg x_2 \vee x_5$	$\mathbf{x_1} \vee \neg\mathbf{x_2} \vee \mathbf{x_5}$	$p_1 \vee p_2 \vee p_5 \vee r_1$
		$p_1 \vee p_2 \vee p_5 \vee r_2$
		$p_1 \vee p_2 \vee p_5 \vee r_5$
$\neg x_1 \vee \neg x_4 \vee x_5$	$\neg x_1 \vee \neg x_4 \vee \mathbf{x_5}$	$n_1 \vee n_4 \vee p_5 \vee r_5$
$x_2 \vee x_4 \vee \neg x_5$	$x_2 \vee \mathbf{x_4} \vee \neg x_5$	$n_2 \vee p_4 \vee n_5 \vee r_4$
$x_3 \vee x_6 \vee \neg x_5$		

A MaxSAT Encoding for Trail-Minimization

For each variable x_i in $p_\alpha(F)$, we introduce three vars $\{r_i, p_i, n_i\}$. We will add soft clauses $[r_i]$ and hard clauses:

$$
\begin{array}{rcl}
r_i & \rightarrow & \neg p_i \\
r_i & \rightarrow & \neg n_i \\
\neg r_i & \rightarrow & p_i = \alpha_i \\
\neg r_i & \rightarrow & n_i = \neg \alpha_i
\end{array}
$$

If we remove x_4 and x_5 from α we obtain γ. Let us set r_4, r_5 to true and r_1, r_2 to false.

The implications set $\neg p_4, \neg n_4, \neg p_5, \neg n_5$ and $p_1 = x_1,\ n_1 = \neg x_1,\ p_2 = \neg x_2,\ n_2 = x_2$.

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\}$$

F	$p_\alpha(F)$	MaxSAT encoding
	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$	$n_1 \vee n_4 \vee n_5 \vee n_2$
$x_1 \vee x_2 \vee x_4$	$\mathbf{x_1} \vee x_2 \vee \mathbf{x_4}$	$p_1 \vee n_2 \vee p_4 \vee r_1$
		$p_1 \vee n_2 \vee p_4 \vee r_4$
$x_1 \vee \neg x_2 \vee x_5$	$\mathbf{x_1} \vee \neg \mathbf{x_2} \vee \mathbf{x_5}$	$p_1 \vee p_2 \vee p_5 \vee r_1$
		$p_1 \vee p_2 \vee p_5 \vee r_2$
		$p_1 \vee p_2 \vee p_5 \vee r_5$
$\neg x_1 \vee \neg x_4 \vee x_5$	$\neg x_1 \vee \neg x_4 \vee \mathbf{x_5}$	$n_1 \vee n_4 \vee p_5 \vee r_5$
$x_2 \vee x_4 \vee \neg x_5$	$x_2 \vee \mathbf{x_4} \vee \neg x_5$	$n_2 \vee p_4 \vee n_5 \vee r_4$
$x_3 \vee x_6 \vee \neg x_5$		

A MaxSAT Encoding for TRAIL-MINIMIZATION

For each variable x_i in $p_\alpha(F)$, we introduce three vars $\{r_i, p_i, n_i\}$. We will add soft clauses $[r_i]$ and hard clauses:

$$
\begin{array}{rcl}
r_i & \rightarrow & \neg p_i \\
r_i & \rightarrow & \neg n_i \\
\neg r_i & \rightarrow & p_i = \alpha_i \\
\neg r_i & \rightarrow & n_i = \neg\alpha_i
\end{array}
$$

If we remove x_4 and x_5 from α we obtain γ. Let us set r_4, r_5 to true and r_1, r_2 to false.

The implications set $\neg p_4, \neg n_4, \neg p_5, \neg n_5$ and $p_1 = x_1,\ n_1 = \neg x_1,\ p_2 = \neg x_2,\ n_2 = x_2$.

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\}$$

F	$p_\alpha(F)$	MaxSAT encoding
	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$	$\neg x_1 \vee \quad \vee x_2$
$x_1 \vee x_2 \vee x_4$	$\mathbf{x_1} \vee x_2 \vee \mathbf{x_4}$	$p_1 \vee n_2 \vee p_4 \vee r_1$
		$p_1 \vee n_2 \vee p_4 \vee r_4$
$x_1 \vee \neg x_2 \vee x_5$	$\mathbf{x_1} \vee \neg\mathbf{x_2} \vee \mathbf{x_5}$	$p_1 \vee p_2 \vee p_5 \vee r_1$
		$p_1 \vee p_2 \vee p_5 \vee r_2$
		$p_1 \vee p_2 \vee p_5 \vee r_5$
$\neg x_1 \vee \neg x_4 \vee x_5$	$\neg x_1 \vee \neg x_4 \vee \mathbf{x_5}$	$n_1 \vee n_4 \vee p_5 \vee r_5$
$x_2 \vee x_4 \vee \neg x_5$	$x_2 \vee \mathbf{x_4} \vee \neg x_5$	$n_2 \vee p_4 \vee n_5 \vee r_4$
$x_3 \vee x_6 \vee \neg x_5$		

A MaxSAT Encoding for TRAIL-MINIMIZATION

For each variable x_i in $p_\alpha(F)$, we introduce three vars $\{r_i, p_i, n_i\}$. We will add soft clauses $[r_i]$ and hard clauses:

$$
\begin{array}{rcl}
r_i & \rightarrow & \neg p_i \\
r_i & \rightarrow & \neg n_i \\
\neg r_i & \rightarrow & p_i = \alpha_i \\
\neg r_i & \rightarrow & n_i = \neg\alpha_i
\end{array}
$$

If we remove x_4 and x_5 from α we obtain γ. Let us set r_4, r_5 to true and r_1, r_2 to false.

The implications set $\neg p_4, \neg n_4, \neg p_5, \neg n_5$ and $p_1 = x_1,\ n_1 = \neg x_1,\ p_2 = \neg x_2,\ n_2 = x_2$.

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\}$$

F	$p_\alpha(F)$	MaxSAT encoding
	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$	$\neg x_1 \vee \quad \vee x_2$
$x_1 \vee x_2 \vee x_4$	$\mathbf{x_1} \vee x_2 \vee \mathbf{x_4}$	$x_1 \vee x_2$
$x_1 \vee \neg x_2 \vee x_5$	$\mathbf{x_1} \vee \neg\mathbf{x_2} \vee \mathbf{x_5}$	$p_1 \vee p_2 \vee p_5 \vee r_1$
		$p_1 \vee p_2 \vee p_5 \vee r_2$
		$p_1 \vee p_2 \vee p_5 \vee r_5$
$\neg x_1 \vee \neg x_4 \vee x_5$	$\neg x_1 \vee \neg x_4 \vee \mathbf{x_5}$	$n_1 \vee n_4 \vee p_5 \vee r_5$
$x_2 \vee x_4 \vee \neg x_5$	$x_2 \vee \mathbf{x_4} \vee \neg x_5$	$n_2 \vee p_4 \vee n_5 \vee r_4$
$x_3 \vee x_6 \vee \neg x_5$		

A MaxSAT Encoding for Trail-Minimization

For each variable x_i in $p_\alpha(F)$, we introduce three vars $\{r_i, p_i, n_i\}$. We will add soft clauses $[r_i]$ and hard clauses:

$$
\begin{array}{rcl}
r_i & \rightarrow & \neg p_i \\
r_i & \rightarrow & \neg n_i \\
\neg r_i & \rightarrow & p_i = \alpha_i \\
\neg r_i & \rightarrow & n_i = \neg \alpha_i
\end{array}
$$

If we remove x_4 and x_5 from α we obtain γ. Let us set r_4, r_5 to true and r_1, r_2 to false.

The implications set $\neg p_4, \neg n_4, \neg p_5, \neg n_5$ and $p_1 = x_1,\ n_1 = \neg x_1,\ p_2 = \neg x_2,\ n_2 = x_2$.

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\}$$

F	$p_\alpha(F)$	MaxSAT encoding
	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$	$\neg x_1 \vee \quad \vee x_2$
$x_1 \vee x_2 \vee x_4$	$\mathbf{x_1} \vee x_2 \vee \mathbf{x_4}$	$x_1 \vee x_2$
$x_1 \vee \neg x_2 \vee x_5$	$\mathbf{x_1} \vee \neg \mathbf{x_2} \vee \mathbf{x_5}$	$x_1 \vee \neg x_2$
		$x_1 \vee \neg x_2$
$\neg x_1 \vee \neg x_4 \vee x_5$	$\neg x_1 \vee \neg x_4 \vee \mathbf{x_5}$	$n_1 \vee n_4 \vee p_5 \vee r_5$
$x_2 \vee x_4 \vee \neg x_5$	$x_2 \vee \mathbf{x_4} \vee \neg x_5$	$n_2 \vee p_4 \vee n_5 \vee r_4$
$x_3 \vee x_6 \vee \neg x_5$		

A MaxSAT Encoding for TRAIL-MINIMIZATION

For each variable x_i in $p_\alpha(F)$, we introduce three vars $\{r_i, p_i, n_i\}$. We will add soft clauses $[r_i]$ and hard clauses:

$$
\begin{array}{rcl}
r_i & \rightarrow & \neg p_i \\
r_i & \rightarrow & \neg n_i \\
\neg r_i & \rightarrow & p_i = \alpha_i \\
\neg r_i & \rightarrow & n_i = \neg \alpha_i
\end{array}
$$

If we remove x_4 and x_5 from α we obtain γ. Let us set r_4, r_5 to true and r_1, r_2 to false.

The implications set $\neg p_4, \neg n_4, \neg p_5, \neg n_5$ and $p_1 = x_1,\ n_1 = \neg x_1,\ p_2 = \neg x_2,\ n_2 = x_2$.

$$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\}$$

F	$p_\alpha(F)$	MaxSAT encoding
	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$	$\neg x_1 \vee \quad \vee x_2$
$x_1 \vee x_2 \vee x_4$	$\mathbf{x_1} \vee x_2 \vee \mathbf{x_4}$	$x_1 \vee x_2$
$x_1 \vee \neg x_2 \vee x_5$	$\mathbf{x_1} \vee \neg \mathbf{x_2} \vee \mathbf{x_5}$	$x_1 \vee \neg x_2$
		$x_1 \vee \neg x_2$
$\neg x_1 \vee \neg x_4 \vee x_5$	$\neg x_1 \vee \neg x_4 \vee \mathbf{x_5}$	
$x_2 \vee x_4 \vee \neg x_5$	$x_2 \vee \mathbf{x_4} \vee \neg x_5$	
$x_3 \vee x_6 \vee \neg x_5$		

A MaxSAT Encoding for TRAIL-MINIMIZATION

For each variable x_i in $p_\alpha(F)$, we introduce three vars $\{r_i, p_i, n_i\}$. We will add soft clauses $[r_i]$ and hard clauses:

$$
\begin{array}{rcl}
r_i & \rightarrow & \neg p_i \\
r_i & \rightarrow & \neg n_i \\
\neg r_i & \rightarrow & p_i = \alpha_i \\
\neg r_i & \rightarrow & n_i = \neg\alpha_i
\end{array}
$$

If we remove x_4 and x_5 from α we obtain γ. Let us set r_4, r_5 to true and r_1, r_2 to false.

The implications set $\neg p_4, \neg n_4, \neg p_5, \neg n_5$ and $p_1 = x_1,\ n_1 = \neg x_1,\ p_2 = \neg x_2,\ n_2 = x_2$.

$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\}$ $\qquad$ $\gamma = \{x_1, \neg x_2\}$

F	$p_\alpha(F)$	$p_{\gamma(F)}$
	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$	$\neg x_1 \vee \vee x_2$
$x_1 \vee x_2 \vee x_4$	$\mathbf{x_1} \vee x_2 \vee \mathbf{x_4}$	$x_1 \vee x_2$
$x_1 \vee \neg x_2 \vee x_5$	$\mathbf{x_1} \vee \neg\mathbf{x_2} \vee \mathbf{x_5}$	$x_1 \vee \neg x_2$
		$x_1 \vee \neg x_2$
$\neg x_1 \vee \neg x_4 \vee x_5$	$\neg x_1 \vee \neg x_4 \vee \mathbf{x_5}$	
$x_2 \vee x_4 \vee \neg x_5$	$x_2 \vee \mathbf{x_4} \vee \neg x_5$	
$x_3 \vee x_6 \vee \neg x_5$		

A MaxSAT Encoding for TRAIL-MINIMIZATION

For each variable x_i in $p_\alpha(F)$, we introduce three vars $\{r_i, p_i, n_i\}$. We will add soft clauses $[r_i]$ and hard clauses:

$$
\begin{array}{rcl}
r_i & \rightarrow & \neg p_i \\
r_i & \rightarrow & \neg n_i \\
\neg r_i & \rightarrow & p_i = \alpha_i \\
\neg r_i & \rightarrow & n_i = \neg\alpha_i
\end{array}
$$

KEY IDEA: by setting the r_i's to appropriate values, the MaxSAT encoding allows us to explore the reducts of all possible subsets of α.
The soft clauses force the MaxSAT solver to pick the smallest subset with satisfiable reduct.

$\alpha = \{x_1^d, x_4^d, x_5, \neg x_2^d\}$ $\qquad$ $\gamma = \{x_1, \neg x_2\}$

F	$p_\alpha(F)$	$p_{\gamma(F)}$
	$\neg x_1 \vee \neg x_4 \vee \neg x_5 \vee x_2$	$\neg x_1 \vee \quad \vee x_2$
$x_1 \vee x_2 \vee x_4$	$\mathbf{x_1} \vee x_2 \vee \mathbf{x_4}$	$x_1 \vee x_2$
$x_1 \vee \neg x_2 \vee x_5$	$\mathbf{x_1} \vee \neg\mathbf{x_2} \vee \mathbf{x_5}$	$x_1 \vee \neg x_2$
		$x_1 \vee \neg x_2$
$\neg x_1 \vee \neg x_4 \vee x_5$	$\neg x_1 \vee \neg x_4 \vee \mathbf{x_5}$	
$x_2 \vee x_4 \vee \neg x_5$	$x_2 \vee \mathbf{x_4} \vee \neg x_5$	
$x_3 \vee x_6 \vee \neg x_5$		

SDCL with Redundant Clause Minimization

$\alpha := \emptyset$
while *true* **do**
  $\alpha := \mathit{unitPropagate}(F, \alpha)$
  if *conflict found* **then**
    $C := \mathit{analyzeConflict}()$
    $F := F \wedge C$
    if *C is the empty clause* **then** return UNSAT
    $\alpha := \mathit{backjump}(C, \alpha)$
  else if $P_\alpha(F)$ *is satisfiable* **then**
    $\gamma := \mathit{minimize}(\alpha)$
    $C := \mathit{analyzeConflict}(\neg\gamma)$
    $F := F \wedge C$
    $\alpha := \mathit{backjump}(C, \alpha)$
  else
    if *all variables are assigned* **then** return SAT
    $\alpha := \alpha \cup \mathit{Decide}()$

Outline of the Talk

Design Decisions - Order of calls

- There are two calls to be made:
  1. Check $p_\alpha(F)$ for satisfiability (SAT) [external SAT solver].
  2. Check whether $\exists\ \gamma \subseteq \alpha$ such that $p_\gamma(F)$ is satisfiable [external MaxSAT solver].
- It would suffice to only make the MaxSAT call.
  However, the MaxSAT call is more expensive than the SAT one.
- If we first make the SAT call and, **only if satisfiable**, we make the MaxSAT call, are we missing something?

  Could it be the case that the SAT call returns unsat ($p_\alpha(F)$ unsat), but the MaxSAT call would have found some $\gamma \subseteq \alpha$ with $p_\gamma(F)$ satisf.?

Proposition

Given assignments with $\gamma \subseteq \alpha$ if $p_\gamma(F)$ is satisfiable, $p_\alpha(F)$ is also satisfiable.

- This means that sometimes we will do the two calls but, since usually the SAT call will be unsat, we will mostly only make the SAT call (the cheaper one).

Design Decisions - Frequency of Calls

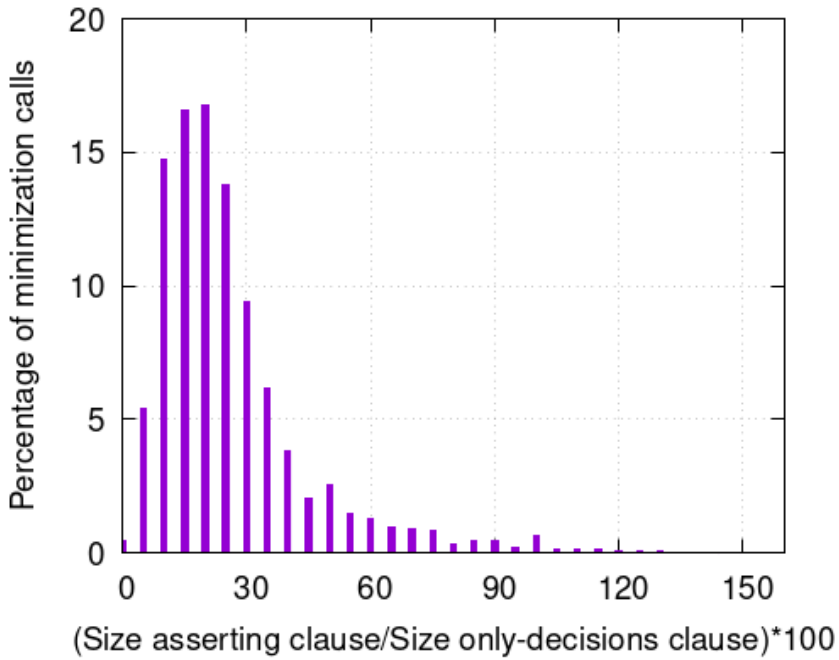
How often to check for the satisfiability of $p_\alpha(F)$?

- Without clause minimization, we should check it as soon as possible: the smaller α, the smaller the lemma to be learned
- With clause minimization, there is a compromise:
  - The sooner we find $p_\alpha(F)$ satisf., the sooner we prune the search
  - Smaller α have smaller $p_\alpha(F)$, and hence easier to solve
  - Larger α have higher chances of having $p_\alpha(F)$ satisfiable
  - Late calls lead to same-size lemmas (because of minimization)
- In our implementation:
  - Define a decision level D at which to check $p_\alpha(F)$ for satisfiability
  - If % of satisfiable checks is smaller than (e.g. 15%) increment D
  - If % of satisfiable checks is larger than (e.g. 15%) decrement D

Experimental Setting

- Implemented SDCL solver with clause minimization on top of CDCL solver MapleSAT [LGPC16]
- Used EvalMaxSAT [Ave20] as MaxSAT solver
- No modification to MapleSAT parameters (not even decision heuristics!)
- Run experiments on benchmarks that are known to be good for SDCL

Data on Lemma Size Reduction



On this benchmark, in about 70% of the minimization calls the size of the asserting lemma was at most 0.3 times the size of the all-decisions clause

Experimental Results

Results on mutilated chessboard and bipartite perfect matching problems:

		SaDiCaL		MapleSDCL		
Benchmark	Kissat	Positive	Filtered	CDCL	SDCL	**SDCL-min**
mchess14	4.6	5682	3.6	11.7	7.3	2.7
mchess15	30.1	$>$ 7200	13.8	54.3	24.7	5.5
mchess16	107	$>$ 7200	19.5	439	191	9
mchess17	2293	$>$ 7200	64.8	5038	517	25.8
mchess18	352	$>$ 7200	71.8	$>$ 7200	3803	52.8
mchess19	$>$ 7200	$>$ 7200	$>$ 7200	$>$ 7200	3578	128
mchess20	3720	$>$ 7200	$>$ 7200	$>$ 7200	$>$ 7200	369
mchess21	$>$ 7200	$>$ 7200	$>$ 7200	$>$ 7200	$>$ 7200	977
mchess22	$>$ 7200	$>$ 7200	$>$ 7200	$>$ 7200	$>$ 7200	4507
mchess23	$>$ 7200	$>$ 7200	$>$ 7200	$>$ 7200	$>$ 7200	6041
randomG-Mix-17	$>$ 7200	$>$ 7200	$>$ 7200	2837	1916	257
randomG-Mix-18	$>$ 7200	$>$ 7200	$>$ 7200	$>$ 7200	$>$ 7200	1683
randomG-n17	$>$ 7200	$>$ 7200	$>$ 7200	1266	688	157
randomG-n18	$>$ 7200	$>$ 7200	$>$ 7200	$>$ 7200	$>$ 7200	2350

Outline of the Talk

Conclusions

- SDCL is a promising approach for enhancing CDCL with more powerful reasoning power
- There is **A LOT** of space for improvement (SDCL is not as mature as CDCL, we should be patient!)
- In this work, we have studied one concrete aspect: how to learn smaller redundant clauses. Other research directions remain open.
- Need for MaxSAT solvers that are good at quick queries (at most 0.1s?)

Conclusions

- SDCL is a promising approach for enhancing CDCL with more powerful reasoning power
- There is **A LOT** of space for improvement (SDCL is not as mature as CDCL, we should be patient!)
- In this work, we have studied one concrete aspect: how to learn smaller redundant clauses. Other research directions remain open.
- Need for MaxSAT solvers that are good at quick queries (at most 0.1s?)

Thank you for your attention!

References I

[AFT11] Albert Atserias, Johannes Klaus Fichte, and Marc Thurley, *Clause-learning algorithms with many restarts and bounded-width resolution*, J. Artif. Intell. Res. **40** (2011), 353–373.

[Ave20] Florent Avellaneda, *A short description of the solver EvalMaxSAT*, Maxsat evaluation 2020, 2020, pp. 8–9.

[Hak85] Armin Haken, *The intractability of resolution*, Theor. Comput. Sci. **39** (1985), 297–308.

[HKB17] Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere, *Short proofs without new variables*, Automated deduction - CADE 26 - 26th international conference on automated deduction, gothenburg, sweden, august 6-11, 2017, proceedings, 2017, pp. 130–147.

[HKB19] Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere, *Encoding redundancy for satisfaction-driven clause learning*, Tools and algorithms for the construction and analysis of systems - 25th international conference, TACAS 2019, held as part of the european joint conferences on theory and practice of software, ETAPS 2019, prague, czech republic, april 6-11, 2019, proceedings, part I, 2019, pp. 41–58.

References II

[HKSB17] Marijn J. H. Heule, Benjamin Kiesl, Martina Seidl, and Armin Biere, *Pruning through satisfaction*, Hardware and software: Verification and testing - 13th international haifa verification conference, HVC 2017, haifa, israel, november 13-15, 2017, proceedings, 2017, pp. 179–194.

[JBH10] Matti Järvisalo, Armin Biere, and Marijn Heule, *Blocked clause elimination*, Tools and algorithms for the construction and analysis of systems, 16th international conference, TACAS 2010, held as part of the joint european conferences on theory and practice of software, ETAPS 2010, paphos, cyprus, march 20-28, 2010. proceedings, 2010, pp. 129–144.

[KSTB16] Benjamin Kiesl, Martina Seidl, Hans Tompits, and Armin Biere, *Super-blocked clauses*, Automated reasoning - 8th international joint conference, IJCAR 2016, coimbra, portugal, june 27 - july 2, 2016, proceedings, 2016, pp. 45–61.

[LGPC16] Jia Hui Liang, Vijay Ganesh, Pascal Poupart, and Krzysztof Czarnecki, *Learning rate based branching heuristic for SAT solvers*, Theory and applications of satisfiability testing - SAT 2016 - 19th international conference, bordeaux, france, july 5-8, 2016, proceedings, 2016, pp. 123–140.

References III

[PD11] Knot Pipatsrisawat and Adnan Darwiche, *On the power of clause-learning SAT solvers as resolution engines*, Artif. Intell. **175** (2011), no. 2, 512–525.