Covered Clauses Are Not Propagation Redundant

Lee A. Barnett David Cerna Armin Biere

Institute for Formal Models and Verification
Johannes Kepler University

July 2, 2020

Overview

- SAT success by reasoning about **redundancy**
  - Non-equivalence-preserving solving techniques
  - Strong propositional proof systems
- Revisit **covered clauses** [5]
  - Generalization of blocked clauses [14]
  - Used for clause elimination, preprocessing
- Consider CCs in more recent context of redundancy
  - Redundancy characterized via **witnesses**
  - Proof systems like SPR and PR [7]

Overview

- SAT success by reasoning about **redundancy**
  - Non-equivalence-preserving solving techniques
  - Strong propositional proof systems
- Revisit **covered clauses** [5]
  - Generalization of blocked clauses [14]
  - Used for clause elimination, preprocessing
- Consider CCs in more recent context of redundancy
  - Redundancy characterized via **witnesses**
  - Proof systems like SPR and PR [7]

- Present an algorithm to identify CCs
- Show CCs are not generalized by PR
- Prove witnesses for CCs are hard to compute
- Deciding clause redundancy is co-DP-complete

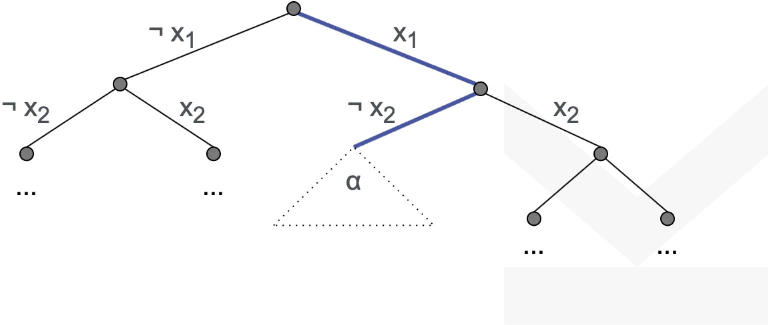
Background

Redundancy [13]

Clause C is **redundant** w.r.t. formula F if:

F and $F \wedge C$ are **equisatisfiable**

- Suppose $(\neg x_1 \vee x_2)$ is redundant w.r.t. F



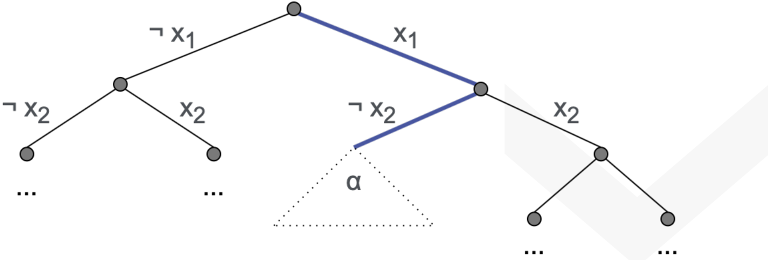
Background

Redundancy [13]

Clause C is **redundant** w.r.t. formula F if:

F and $F \wedge C$ are **equisatisfiable**

- Suppose $(\neg x_1 \vee x_2)$ is redundant w.r.t. F



- If α includes solutions, there are solutions elsewhere too

Background

- Efficiently decidable **redundancy properties** $P(F, C)$

$$P(F, C) \Rightarrow C \text{ is redundant w.r.t. } F$$

Background

- Efficiently decidable **redundancy properties** $P(F, C)$

$$P(F, C) \Rightarrow C \text{ is redundant w.r.t. } F$$

(1) Clause Elimination: iteratively remove $C \in F$ s.t. $P(F, C)$

- Subsumption, blocked clauses, covered clauses (see [6])
- Strong preprocessing, inprocessing techniques

Background

- Efficiently decidable **redundancy properties** $P(F, C)$

$$P(F, C) \Rightarrow C \text{ is redundant w.r.t. } F$$

(1) Clause Elimination: iteratively remove $C \in F$ s.t. $P(F, C)$

- Subsumption, blocked clauses, covered clauses (see [6])
- Strong preprocessing, inprocessing techniques

(2) Proof systems: add $C \notin F$ such that $P(F, C)$, eventually add $\perp$

- (Resolution), DRAT [16], SPR, **PR** (Propagation Redundancy)
- PR has simple, short proofs of **pigeonhole formulas** [7] (shortest res. proofs are exponential in size [3])

Background

- Efficiently decidable **redundancy properties** $P(F, C)$

$$P(F, C) \Rightarrow C \text{ is redundant w.r.t. } F$$

(1) Clause Elimination: iteratively remove $C \in F$ s.t. $P(F, C)$

- Subsumption, blocked clauses, covered clauses (see [6])
- Strong preprocessing, inprocessing techniques

(2) Proof systems: add $C \notin F$ such that $P(F, C)$, eventually add $\bot$

- (Resolution), DRAT [16], SPR, **PR** (Propagation Redundancy)
- PR has simple, short proofs of **pigeonhole formulas** [7] (shortest res. proofs are exponential in size [3])

* Strong even without extension (new variables), deletion
* Seem to be somewhat "automatizable"

Background

Witnesses [7, 9]

HKB [7]: C is redundant w.r.t. $F \iff$ for $\alpha = \neg C$, there exists ω:

$$\begin{array}{lrcl} (1) & \boldsymbol{\omega} & \models & \boldsymbol{C} \\ (2) & \boldsymbol{F}|_\alpha & \models & \boldsymbol{F}|_\omega \end{array}$$

- ω is a **witness** for C

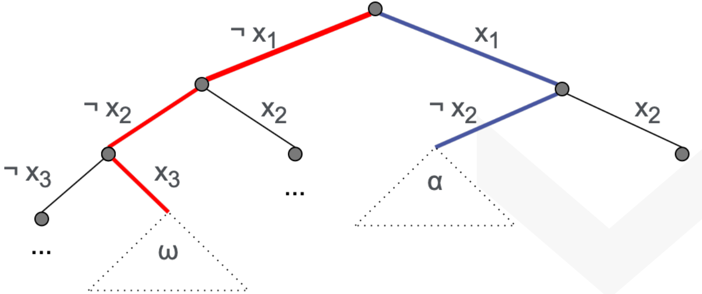
Background

Witnesses [7, 9]

HKB [7]: C is redundant w.r.t. $F \iff$ for $\alpha = \neg C$, there exists ω:

$$\begin{array}{lc} (1) & \boldsymbol{\omega} \models \boldsymbol{C} \\ (2) & \boldsymbol{F}|_\alpha \models \boldsymbol{F}|_\omega \end{array}$$

- ω is a **witness** for C



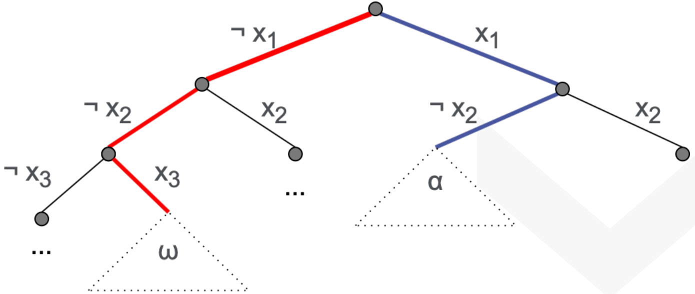
Background

Witnesses [7, 9]

HKB [7]: C is redundant w.r.t. F $\iff$ for $\alpha = \neg C$, there exists ω:

$$\begin{array}{lrcl} (1) & \boldsymbol{\omega} & \models & \boldsymbol{C} \\ (2) & \boldsymbol{F}|_{\alpha} & \models & \boldsymbol{F}|_{\omega} \end{array}$$

- ω is a **witness** for C



- $C = (\neg x_1 \vee x_2)$
- $\omega = \neg x_1, \neg x_2, x_3$ is a witness if $F|_{x_1,\neg x_2} \models F|_{\neg x_1,\neg x_2,x_3}$

Background

Witnesses [7, 9]

Clause elimination: witnesses needed for **reconstruction** [4]

- Remove redundant clauses C_1, C_2, ..., C_N from F
- Solution τ for $F \setminus \{C_1, \dots, C_N\}$ may not satisfy F
- Record witness ω_i for each C_i, apply **reconstruction function**

$$\mathcal{R}_\epsilon(\tau) = \tau \qquad \mathcal{R}_{\sigma\cdot(\omega:C)}(\tau) = \begin{cases} \mathcal{R}_\sigma(\tau) & \text{if } \tau \vDash C \\ \mathcal{R}_\sigma(\tau \circ \omega) & \text{otherwise} \end{cases}$$

Background

Witnesses [7, 9]

Clause elimination: witnesses needed for **reconstruction** [4]

- Remove redundant clauses C_1, C_2, ..., C_N from F
- Solution τ for $F \setminus \{C_1, \ldots, C_N\}$ may not satisfy F
- Record witness ω_i for each C_i, apply **reconstruction function**

$$\mathcal{R}_\epsilon(\tau) = \tau \qquad \mathcal{R}_{\sigma\cdot(\omega:C)}(\tau) = \begin{cases} \mathcal{R}_\sigma(\tau) & \text{if } \tau \vDash C \\ \mathcal{R}_\sigma(\tau \circ \omega) & \text{otherwise} \end{cases}$$

Witnesses crucial for PR proofs as well

C is PR w.r.t. F if $F|_\alpha \vdash_1 F|_\omega$ for some $\omega \vDash C$

- Deciding if C is PR w.r.t. F is NP-complete [10]
- Must record witness for each added C for polynomial proof check

Background

Witnesses [7, 9]

Property	Witness ω	Implication check
Subsumption	any*	$F\|_\alpha \ni \bot$
RUP [2]	any*	$F\|_\alpha \vdash_1 \bot$
Blocking	$\alpha \circ \ell$ for some $\ell \in C$	$F\|_\alpha \supseteq F\|_\omega$
RAT	$\alpha \circ \ell$ for some $\ell \in C$	$F\|_\alpha \vdash_1 F\|_\omega$
Set-blocking [13]	$\alpha \circ L$ for some $L \subseteq C$	$F\|_\alpha \supseteq F\|_\omega$
SPR	$\alpha \circ L$ for some $L \subseteq C$	$F\|_\alpha \vdash_1 F\|_\omega$
Global-blocking [12]	$\alpha \circ L$ for some $L \cap C \neq \emptyset$	$F\|_\alpha \supseteq F\|_\omega$
PR	any	$F\|_\alpha \vdash_1 F\|_\omega$
R	any	$F\|_\alpha \vDash F\|_\omega$

- $F|_\alpha \vdash_1 \bot$ means unit propagation on $F|_\alpha$ produces $\bot$
- $F|_\alpha \vdash_1 F|_\omega$ means $F|_\alpha \wedge \neg D \vdash_1 \bot$ for every $D \in F|_\omega$

Covered Clauses

Definition [5, 6]

- Consider the resolvents of $C = (a \vee b)$ on a:

$$\boldsymbol{C \otimes_a (D \vee \neg a)} \text{ for all } (D \vee \neg a) \in F$$

All resolvents tautological	$\Rightarrow$	a blocks C
All non-taut. resolvents include x	$\Rightarrow$	a **covers** x

- Extend C by adding covered literals: $C_{\text{ext}} = (a \vee b \vee x)$

$$(F \wedge C_{\text{ext}})|_{\neg a, \neg b} \ \vdash_1 \ (F \wedge C_{\text{ext}})|_{a, \neg b}$$

- C is redundant w.r.t. $F \wedge C_{\text{ext}}$, with witness $\boldsymbol{\omega = a, \neg b}$
- Iteratively add covered literals to C_{ext}

C is **covered** if some extension C_{ext} is blocked, or subsumed.

Covered Clauses

Example

(1) a covers x	$(a \vee b)$
	$\Downarrow$
(2) b covers y	$(a \vee b \vee x)$
	$\Downarrow$
(3) C_{ext} blocked by x	$(a \vee b \vee x \vee y)$
	$\Downarrow$
(4) $(a \vee b)$ is covered	$\top$

Covered Clauses

Example

(1) a covers x	$(a \vee b)$
	$\Downarrow$
(2) b covers y	$(a \vee b \vee x)$
	$\Downarrow$
(3) C_{ext} blocked by x	$(a \vee b \vee x \vee y)$
	$\Downarrow$
(4) $(a \vee b)$ is covered	$\top$

- Reconstruction for CC-Elimination: use the sequence of witnesses

Covered Clauses

Example

(1) a covers x	$(a \vee b)$	$\omega_1 = a, \neg b$
	$\Downarrow$	
(2) b covers y	$(a \vee b \vee x)$	$\omega_2 = \neg a, b, \neg x$
	$\Downarrow$	
(3) C_{ext} blocked by x	$(a \vee b \vee x \vee y)$	$\omega_3 = \neg a, \neg b, x, \neg y$
	$\Downarrow$	
(4) $(a \vee b)$ is covered	$\top$	

- Reconstruction for CC-Elimination: use the sequence of witnesses

Covered Clauses

Example

(1) a covers x	$(a \vee b)$ $\Downarrow$	$\omega_1 = a, \neg b$
(2) b covers y	$(a \vee b \vee x)$ $\Downarrow$	$\omega_2 = \neg a, b, \neg x$
(3) C_{ext} blocked by x	$(a \vee b \vee x \vee y)$ $\Downarrow$	$\omega_3 = \neg a, \neg b, x, \neg y$
(4) $(a \vee b)$ is covered	$\top$	

- Reconstruction for CC-Elimination: use the sequence of witnesses

$$\begin{array}{ll} (\omega_3 : & a \vee b \vee x \vee y) \\ (\omega_2 : & a \vee b \vee x) \\ (\omega_1 : & a \vee b) \end{array}$$

Identifying Covered Clauses

- Actually identify asymmetric covered clauses [6]
- Can extend C by both covered and **asymmetric** literals ℓ:

$$(D \vee \neg\ell) \in F \text{ for some } D \subseteq C$$

Identifying Covered Clauses

- Actually identify asymmetric covered clauses [6]
- Can extend C by both covered and **asymmetric** literals ℓ:

$$(D \vee \neg\ell) \in F \text{ for some } D \subseteq C$$

- Adding asymmetric literals is equivalence-preserving
- ACC is strictly more general than CC

Identifying Covered Clauses

$\mathsf{ACC}(F, C)$

$\sigma := \varepsilon$

$E := C$

$\alpha := \neg C$

repeat

  if $\bot \in F|_\alpha$ **then return** $(\textbf{true}, \sigma)$

  if there are unit clauses in $F|_\alpha$ **then**

    $\alpha := \alpha \cup \{u\}$ for each unit u

  else

    for each $l \in E$

      $G := \{D|_\alpha \mid (D \vee \neg l) \in F \text{ and } D|_\alpha \neq \top\}$

      if $G = \emptyset$ **then return** $\big(\textbf{true}, \sigma \cdot (\neg E_l : E)\big)$

      $\Phi := \bigcap G$

      if $\Phi \neq \emptyset$ **then**

        $\sigma := \sigma \cdot (\neg E_l : E)$

        $E := E \cup \Phi$

        $\alpha := \alpha \cup \neg \Phi$

until no updates to α

return $(\textbf{false}, \varepsilon)$

Identifying Covered Clauses

ACC(F, C)

$\sigma := \varepsilon$ — witness sequence

$E := C$ — C with covered literals

$\alpha := \neg C$ — C with all added literals (negated)

repeat

  if $\perp \in F|_\alpha$ **then return** (**true**, σ) — Subsumption Check

  if there are unit clauses in $F|_\alpha$ **then** — Add all asymmetric literals

    $\alpha := \alpha \cup \{u\}$ for each unit u

  else

    for each $l \in E$

      $G := \{D|_\alpha \mid (D \vee \neg l) \in F \text{ and } D|_\alpha \neq \top\}$

      if $G = \emptyset$ **then return** (**true**, $\sigma \cdot (\neg E_l : E)$) — Blocking Check on l

      $\Phi := \bigcap G$

      if $\Phi \neq \emptyset$ **then**

        $\sigma := \sigma \cdot (\neg E_l : E)$ — Record to witness seq.

        $E := E \cup \Phi$ — Add all literals covered by l

        $\alpha := \alpha \cup \neg\Phi$

until no updates to α

return (**false**, ε)

Witnesses for Covered Clauses

- Witness sequence can be quadratic in the size of C_{ext}

$$|\omega_n| + \cdots + |\omega_1|$$

Witnesses for Covered Clauses

- Witness sequence can be quadratic in the size of C_{ext}

$$|C_{\text{ext}}| + \cdots + |C|$$

Witnesses for Covered Clauses

- Witness sequence can be quadratic in the size of C_{ext}

$$|C_{\text{ext}}| + \cdots + |C|$$

- C is covered in $F \Rightarrow C$ is redundant w.r.t. F
- **HKB**: C has a single witness ω. Can we compute it?

Witnesses for Covered Clauses

- Witness sequence can be quadratic in the size of C_{ext}

$$|C_{\text{ext}}| + \cdots + |C|$$

- C is covered in $F \Rightarrow C$ is redundant w.r.t. F
- **HKB**: C has a single witness ω. Can we compute it?

Thm. Computing ω can be as hard as finding satisfying assignments.

Witnesses for Covered Clauses

Thm. Computing ω can be as hard as finding satisfying assignments.

- Proof sketch: Take any CNF formula G. Let $C = (a \vee b)$

Witnesses for Covered Clauses

Thm. Computing ω can be as hard as finding satisfying assignments.

- Proof sketch: Take any CNF formula G. Let $C = (a \vee b)$
- Define $F = (\neg a \vee x) \wedge (\neg b \vee y) \wedge (\neg x \vee \neg y) \wedge \mathcal{S}(G, x, y)$, and

Witnesses for Covered Clauses

Thm. Computing ω can be as hard as finding satisfying assignments.

- Proof sketch: Take any CNF formula G. Let $C = (a \vee b)$
- Define $F = (\neg a \vee x) \wedge (\neg b \vee y) \wedge (\neg x \vee \neg y) \wedge \mathcal{S}(G, x, y)$, and

$$\mathcal{S}(G, x, y) = \bigwedge_{D \in G} (x \vee D) \bigwedge_{D' \in G'} (y \vee D')$$

where $G' = \bigwedge_{D \in G} D'$ is a variable-renamed copy of G.

Witnesses for Covered Clauses

Thm. Computing ω can be as hard as finding satisfying assignments.

- Proof sketch: Take any CNF formula G. Let $C = (a \vee b)$
- Define $F = (\neg a \vee x) \wedge (\neg b \vee y) \wedge (\neg x \vee \neg y) \wedge \mathcal{S}(G, x, y)$, and

$$\mathcal{S}(G, x, y) = \bigwedge_{D \in G} (x \vee D) \bigwedge_{D' \in G'} (y \vee D')$$

where $G' = \bigwedge_{D \in G} D'$ is a variable-renamed copy of G.

- C is covered: $(a \vee b) \rightarrow (a \vee b \vee x) \rightarrow (a \vee b \vee x \vee y) \rightarrow \top$

Witnesses for Covered Clauses

$$C = (a \vee b)$$

$$F = (\neg a \vee x) \wedge (\neg b \vee y) \wedge (\neg x \vee \neg y) \wedge \mathcal{S}(G, x, y)$$

$$\mathcal{S}(G, x, y) = \bigwedge_{D \in G} (x \vee D) \bigwedge_{D' \in G'} (y \vee D')$$

Witnesses for Covered Clauses

$$C = (a \vee b)$$

$$F = (\neg a \vee x) \wedge (\neg b \vee y) \wedge (\neg x \vee \neg y) \wedge \mathcal{S}(G, x, y)$$

$$\mathcal{S}(G, x, y) = \bigwedge_{D \in G} (x \vee D) \bigwedge_{D' \in G'} (y \vee D')$$

- G is satisfiable $\iff$ any ω for C includes a solution for G or G'

Witnesses for Covered Clauses

$$C = (a \vee b)$$

$$F = (\neg a \vee x) \wedge (\neg b \vee y) \wedge (\neg x \vee \neg y) \wedge \mathcal{S}(G, x, y)$$

$$\mathcal{S}(G, x, y) = \bigwedge_{D \in G} (x \vee D) \bigwedge_{D' \in G'} (y \vee D')$$

- G is satisfiable $\iff$ any ω for C includes a solution for G or G'

$$F|_{\alpha} = (\neg x \vee \neg y) \wedge \mathcal{S}(G, x, y)$$

Witnesses for Covered Clauses

$$C = (a \vee b)$$

$$F = (\neg a \vee x) \wedge (\neg b \vee y) \wedge (\neg x \vee \neg y) \wedge \mathcal{S}(G, x, y)$$

$$\mathcal{S}(G, x, y) = \bigwedge_{D \in G} (x \vee D) \bigwedge_{D' \in G'} (y \vee D')$$

- G is satisfiable $\iff$ any ω for C includes a solution for G or G'

$$F|_\alpha = (\neg x \vee \neg y) \wedge \mathcal{S}(G, x, y)$$

- If $a \in \omega$ then $(\neg a \vee x)|_\omega = (x) \in F|_\omega \Rightarrow x \in \omega$ as well
- If $x \in \omega$ then $\neg y \in \omega$, and ω satisfies G'

Witnesses for Covered Clauses

$$C = (a \vee b)$$

$$F = (\neg a \vee x) \wedge (\neg b \vee y) \wedge (\neg x \vee \neg y) \wedge \mathcal{S}(G, x, y)$$

$$\mathcal{S}(G, x, y) = \bigwedge_{D \in G} (x \vee D) \bigwedge_{D' \in G'} (y \vee D')$$

- G is satisfiable $\iff$ any ω for C includes a solution for G or G'

$$F|_{\alpha} = (\neg x \vee \neg y) \wedge \mathcal{S}(G, x, y)$$

- If $a \in \omega$ then $(\neg a \vee x)|_{\omega} = (x) \in F|_{\omega} \Rightarrow x \in \omega$ as well
- If $x \in \omega$ then $\neg y \in \omega$, and ω satisfies G'
- If $b \in \omega$ then ω satisfies G

Covered Clauses are not PR

Cor. Not all covered clauses are PR

Same set up as before:

$$C = (a \lor b)$$

$$F = (\neg a \lor x) \land (\neg b \lor y) \land (\neg x \lor \neg y) \land \mathcal{S}(G, x, y)$$

$$\mathcal{S}(G, x, y) = \bigwedge_{D \in G} (x \lor D) \bigwedge_{D' \in G'} (y \lor D')$$

Covered Clauses are not PR

Cor. Not all covered clauses are PR

Same set up as before:

$$C = (a \vee b)$$

$$F = (\neg a \vee x) \wedge (\neg b \vee y) \wedge (\neg x \vee \neg y) \wedge \mathcal{S}(G, x, y)$$

$$\mathcal{S}(G, x, y) = \bigwedge_{D \in G} (x \vee D) \bigwedge_{D' \in G'} (y \vee D')$$

- Take $G = (c \vee d) \wedge (\neg c \vee d) \wedge (c \vee \neg d) \wedge (\neg c \vee \neg d)$
- No ω for C satisfies $F|_\alpha \vdash_1 F|_\omega$

Deciding PR vs. R

- Deciding whether a clause is PR is NP-complete [10]

- Main goal: find useful PR clauses while solving
- SDCL [8] use **reducts** to detect PR clauses
  - Small propositional formulas (ideally easy to solve)
  - (Un-)satisfiable reduct $\Rightarrow$ C is (not) PR w.r.t. F

Deciding PR vs. R

- Deciding whether a clause is PR is NP-complete [10]

- Main goal: find useful PR clauses while solving
- SDCL [8] use **reducts** to detect PR clauses
  - Small propositional formulas (ideally easy to solve)
  - (Un-)satisfiable reduct $\Rightarrow$ C is (not) PR w.r.t. F

- Can we detect **R** clauses the same way?

Complexity of R

Difference Polynomial Time

Difference Polynomial Time or **DP** [15] is the class:

$$\mathrm{DP} = \{L_1 \setminus L_2 \mid L_1, L_2 \in \mathrm{NP}\}$$

- Defined to classify various "exact" or "critical problems"
- Ex: can a graph G be colored using exactly four colors?

$$\mathrm{NP}, \text{co-NP} \subseteq \mathbf{DP} \subseteq \Theta_2^{\mathrm{P}} = \mathrm{P}^{\mathrm{NP}[\mathcal{O}(\log)]}$$

- Second-level of the **Boolean hierarchy** over NP [1]
- BH collapse $\Rightarrow$ PH collapse [11]

Complexity of R

Thm. Deciding R is co-DP-complete

- Proof sketch: show complement is complete for DP

$$(F, C) \in \overline{\mathrm{R}} \text{ if } C \text{ is not redundant w.r.t. } F$$

Complexity of R

Thm. Deciding R is co-DP-complete

- Proof sketch: show complement is complete for DP

$$(F, C) \in \overline{\mathrm{R}} \text{ if } C \text{ is not redundant w.r.t. } F$$

- $\overline{\mathrm{R}} = \{(F, C) \mid F \text{ is SAT but } F \wedge C \text{ is UNSAT}\}$
  $= \{(F, C) \mid F \in \mathrm{SAT}\} \setminus \{(F, C) \mid F \wedge C \in \mathrm{SAT}\}$
  $\in \mathrm{DP}$

Complexity of R

Thm. Deciding R is co-DP-complete

- Reduction from SAT-UNSAT: (F, G) s.t. F is SAT, G is UNSAT
- Given F and G, let $C' = x$ and construct:

$$F' = \bigwedge_{C \in F} (C \vee x) \bigwedge_{D \in G} (D \vee \neg x)$$

Complexity of R

Thm. Deciding R is co-DP-complete

- Reduction from SAT-UNSAT: (F, G) s.t. F is SAT, G is UNSAT
- Given F and G, let $C' = x$ and construct:

$$F' = \bigwedge_{C \in F} (C \vee x) \bigwedge_{D \in G} (D \vee \neg x)$$

- C' is not redundant w.r.t. F' $\iff$ $(F, G) \in$ SAT-UNSAT
- Notice:

$$F'|_x = \bigwedge_{D \in G} (D) = G$$

$$F'|_{\neg x} = \bigwedge_{C \in F} (C) = F$$

Conclusion

- Presented algorithm to identify ACCs
- Proved witnesses for CCs are hard to compute
- Showed PR does not generalize CC
- Deciding clause redundancy is co-DP-complete

Conclusion

- Presented algorithm to identify ACCs
- Proved witnesses for CCs are hard to compute
- Showed PR does not generalize CC
- Deciding clause redundancy is co-DP-complete

∗ Future work: different witness structures, non-clausal redundancy

Conclusion

- Presented algorithm to identify ACCs
- Proved witnesses for CCs are hard to compute
- Showed PR does not generalize CC
- Deciding clause redundancy is co-DP-complete

* Future work: different witness structures, non-clausal redundancy

Thanks!

References I

Jin-Yi Cai and Lane Hemachandra.
The Boolean hierarchy: hardware over NP.
In *Structure in complexity theory*, volume 223 of *LNCS*, pages 105–124. Springer, 1986.

Evgueni Goldberg and Yakov Novikov.
Verification of proofs of unsatisfiability for CNF formulas.
In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1*, DATE '03, page 10886, USA, 2003. IEEE Computer Society.

Armin Haken.
The intractability of resolution.
Theoretical Computer Science, 39:297 – 308, 1985.
Third Conference on Foundations of Software Technology and Theoretical Computer Science.

References II

Marijn J. H. Heule, Matti Järvisalo, and Armin Biere.
Clause elimination procedures for CNF formulas.
In Christian G. Fermüller and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning – LPAR 17*, volume 6397 of *LNCS*, pages 357–371. Springer, 2010.

Marijn J. H. Heule, Matti Järvisalo, and Armin Biere.
Covered clause elimination.
In Andrei Voronkov, Geoff Sutcliffe, Matthias Baaz, and Christian Fermüller, editors, *Logic for Programming, Artificial Intelligence and Reasoning – LPAR 2013 (Short)*, volume 13 of *EPiC Series in Computing*, pages 41–46. EasyChair, 2013.

Marijn J. H. Heule, Matti Järvisalo, Florian Lonsing, Martina Seidl, and Armin Biere.
Clause elimination for SAT and QSAT.
Journal of Artificial Intelligence Research, 53(1):127–168, 2015.

References III

Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere.
Short proofs without new variables.
In Leonardo de Moura, editor, *Automated Deduction – CADE 26*, volume 10395 of *LNCS*, pages 130–147. Springer, 2017.

Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere.
Encoding redundancy for satisfaction-driven clause learning.
In Tomás Vojnar and Lijun Zhang, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference – TACAS 2019*, volume 11427 of *LNCS*, pages 41–58. Springer, 2019.

Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere.
Strong extension-free proof systems.
Journal of Automated Reasoning, 64:533–554, 2020.

Marijn J. H. Heule, Benjamin Kiesl, Martina Seidl, and Armin Biere.
PRuning through satisfaction.
In Ofer Strichman and Rachel Tzoref-Brill, editors, *Haifa Verification Conference – HVC 2017*, volume 10629 of *LNCS*, pages 179–194. Springer, 2017.

References IV

Jim Kadin.
The polynomial time hierarchy collapses if the boolean hierarchy collapses.
SIAM Journal on Computing, 17(6):1263–1282, 1988.

Benjamin Kiesl, Marijn J. H. Heule, and Armin Biere.
Truth assignments as conditional autarkies.
In Yu-Fang Chen, Chih-Hong Cheng, and Javier Esparza, editors, *Automated Technology for Verification and Analysis - 17th International Symposium, ATVA 2019, Taipei, Taiwan, October 28-31, 2019, Proceedings*, volume 11781 of *Lecture Notes in Computer Science*, pages 48–64. Springer, 2019.

Benjamin Kiesl, Martina Seidl, Hans Tompits, and Armin Biere.
Super-blocked clauses.
In Nicola Olivetti and Ashish Tiwari, editors, *International Joint Conference on Automated Reasoning – IJCAR 2016*, volume 9706 of *LNCS*, pages 45–61. Springer, 2016.

Oliver Kullmann.
On a generalization of xended resolution.
Discrete Appl. Math., 96/97:149–176, 1999.

References V

Christos Papadimitriou and Mihalis Yannakakis.
The complexity of facets (and some facets of complexity).
Journal of Computer and System Sciences, 28(2):244–259, 1984.

Nathan Wetzler, Marijn J. H. Heule, and Warren A. Hunt.
DRAT-trim: Efficient checking and trimming using expressive clausal proofs.
In Carsten Sinz and Uwe Egly, editors, *Theory and applications of satisfiability testing – SAT 2014*, volume 8561 of *LNCI*, pages 422–429. Springer, 2014.