Non-Clausal Redundancy Properties

Lee A. Barnett Armin Biere

Institute for Formal Models and Verification
Johannes Kepler University

July 12, 2021

JOHANNES KEPLER
UNIVERSITY LINZ

LIT
AI LAB

logics
LOGICAL METHODS IN
COMPUTER SCIENCE

Motivation

- SAT solvers are used where correctness matters
  - Verifying hardware and software [CBRZ01, GPB01, KSHK07]
  - Subroutines in other reasoning tools [BSST21, Vor14]
  - Search for solutions to math problems [HKM16, KL15]
- Solvers should produce externally-checkable certificates
  - **Example**: if F is UNSAT, produce a resolution refutation
- Most modern proof systems infer redundant clauses

Clause C is **redundant** w.r.t. formula F if $F \equiv_{\text{SAT}} F \wedge C$

- Examples: RUP [GN03], RAT [WHH14], PR [HKB17], ...

Motivation

- PR has short refutations for many **hard** problems (see [BT19])
  - Problems with no polynomial-length resolution refutation
- CDCL searches for resolution refutations [BKS04]
- PR presents the potential for major speed-ups in solving
  - Not obvious how to exploit this in practice (but see SDCL [HKB19])
- Established techniques already offer speed-up beyond CDCL

	`--gauss`	`--no-gauss`
`urquhart_s5_b3`	0.1 sec	$>$ 10 hrs

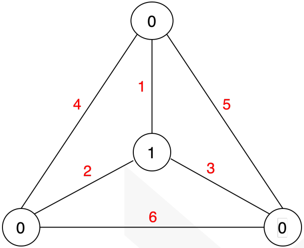
- Clausal proof systems struggle to express non-clausal reasoning
  - Typically disable XOR, cardinality reasoning if proofs required

* Want proofs that express these techniques as well

A hard problem

- Graph G, each v has a **charge** $\gamma(v) \in \{0, 1\}$, total charge is odd
- Variable $\boldsymbol{x_e}$ for each edge e in G
- Formula $\boldsymbol{F_{G,\gamma}}$: for each v, parity of the incident x_e equals $\gamma(v)$

```
p cnf 6 16
1 2 3 0
1 -2 -3 0
-1 2 -3 0
-1 -2 3 0
1 4 -5 0
1 -4 5 0
-1 4 5 0
-1 -4 -5 0
2 4 -6 0
2 -4 6 0
-2 4 6 0
-2 -4 -6 0
3 5 -6 0
3 -5 6 0
-3 5 6 0
-3 -5 -6 0
```



- For certain graphs, no short resolution proofs [Tse70, Urq95]
- **Tseitin formulas** have short DRAT, (D)PR proofs [BT19, HKB19]

Redundancy Properties

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

RUP	unit propagation on $F \wedge \neg C$ produces conflict: $\boldsymbol{F \vdash_1 C}$
RAT	for some $\ell \in C$ the clause $C \vee D$ is RUP for all $D \vee \neg \ell \in F$
PR	for some assignment ω, both $C\|_\omega = \top$ and $F\|_{\neg C} \vdash_1 F\|_\omega$

- Add clauses to F that meet the redundancy property
- Prove "UNSAT" by eventually adding the empty clause $\perp$
- Deciding whether a clause is PR is NP-complete [HKSB17]

XOR Reasoning

- Can Tseitin formulas be solved without looking for PR clauses?
- CryptoMiniSAT [SNC09], Lingeling [Bie18], . . . use **XOR reasoning**

```
p cnf 6 16
1 2 3 0          2 4 -6 0
1 -2 -3 0        2 -4 6 0
-1 2 -3 0        -2 4 6 0
-1 -2 3 0        -2 -4 -6 0
1 4 -5 0         3 5 -6 0
1 -4 5 0         3 -5 6 0
-1 4 5 0         -3 5 6 0
-1 -4 -5 0       -3 -5 -6 0
```

$$x_1 \oplus x_2 \oplus x_3 = 1$$
$$x_1 \oplus x_4 \oplus x_5 = 0$$
$$x_2 \oplus x_4 \oplus x_6 = 0$$
$$x_3 \oplus x_5 \oplus x_6 = 0$$

- Extract XOR constraints, solve efficiently with **Gaussian elimination**
- But expressing this is a challenge for RUP, RAT, PR, ...
- Not just XORs. Example: reasoning about **cardinality**

Non-clausal Redundancy

Function g is **redundant** w.r.t. function f if $f \equiv_{\mathsf{SAT}} f \wedge g$

Want non-clausal redundancy properties, proof systems

- Efficiently-checkable
- Easily express existing solver techniques
- Extend existing proof systems

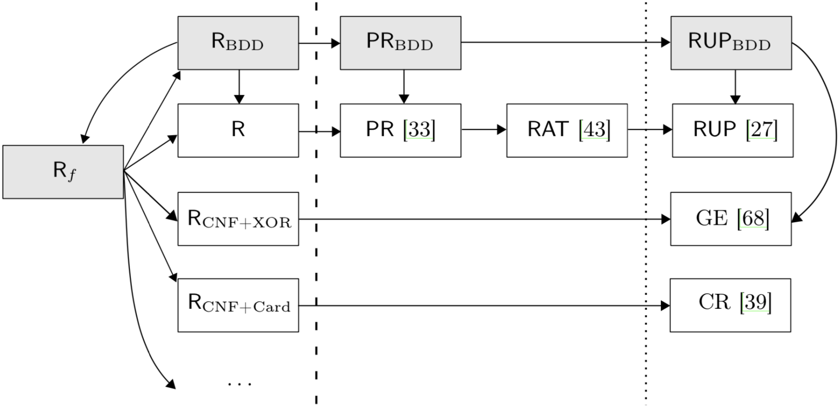
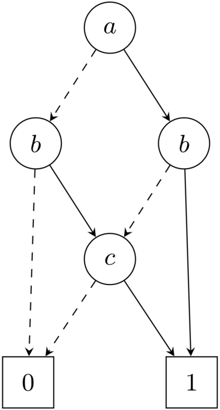


Fig. 1: Different notions of redundancy and their relationships. An arrow from A to B indicates A generalizes B. Properties to the right of the thick dashed line are polynomially checkable; those to the right of the thin dotted line only derive logical consequences. Novel properties defined in this paper are grey.

Binary Decision Diagrams

- BDDs [Ake78, Bry86, Lee59] compactly express Boolean functions
- Long history in SAT (e.g. [BH21, DK03, FKS+04, MM02, PV04])



- **Shannon decomposition**

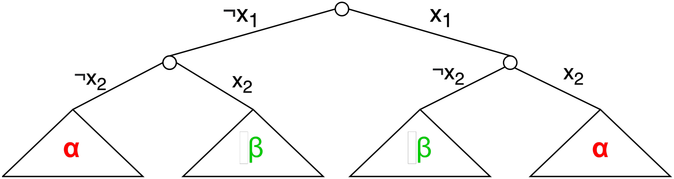
$$f = (\neg x \wedge f|_{\neg x}) \vee (x \wedge f|_{x})$$

- $a + b + c \geq 2$
- Clauses are easy to represent
- Formulas in general are not
- Conjunction of BDDs:

$$F = f_1 \wedge \cdots \wedge f_n$$

Redundancy for BDDs

- $x_1 \oplus x_2 = 1$ is redundant w.r.t. $F \iff$ not all F-solutions are in $\boldsymbol{\alpha}$



- Want a function $F|_\alpha$ such that
  - If assignment τ is in α then $F|_\alpha(\tau) = F(\tau)$
  - $F|_\alpha$ is simpler than F
- A **generalized cofactor** of F by $x_1 \oplus x_2 = 0$

Generalized Cofactor

- Can compute $f|_g$ using **constrain** operation [CM90, TSL+90]

Constrain(f, g), for $g \neq 0$, produces the BDD $f \circ \pi_g$, with π_g given by

$$\pi_g(\tau) = \begin{cases} \tau & \text{if } g(\tau) = 1 \\ \underset{\{\tau' \mid g(\tau')=1\}}{\arg\min} \; d(\tau, \tau') & \text{otherwise} \end{cases}$$

where $d(\tau, \tau') = \sum_{i=1}^{n} |\tau(x_i) - \tau'(x_i)| \cdot 2^{n-i}$ for variables $x_1 \prec \cdots \prec x_n$.

- Usually smaller than f and can be computed efficiently
- Distributes over $\wedge$, so $(f_1 \wedge \cdots \wedge f_n)|_g = f_1|_g \wedge \cdots \wedge f_n|_g$

Redundancy for BDDs

Can characterize redundancy as follows:

A BDD g is redundant w.r.t. $f_1 \wedge \cdots \wedge f_n$ iff there exists $\omega = \ell_1 \wedge \cdots \wedge \ell_k$:

1. $g|_\omega$ is the "always-true" BDD 1
2. $f_1|_{\neg g} \wedge \cdots \wedge f_n|_{\neg g} \models f_1|_\omega \wedge \cdots \wedge f_n|_\omega$

- Compare with the following characterization for clauses [HKB20]

A clause C is redundant w.r.t. formula F iff there exists an assignment ω:

1. ω satisfies C
2. $F|_{\neg C} \models F|_\omega$

- Can check redundancy by checking this implication
- Make this efficient by using a restricted implication relation
- For clause redundancy properties, use RUP

BDD Unit Propagation

UnitProp$(f_1, \ldots, f_n)$

repeat

  if $f_i = 0$ **or** $f_i = \neg f_j$ **for some** $1 \leq i, j \leq n$ **then**

    return "conflict"

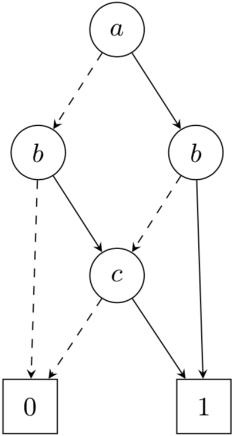
  if $U(f_i) \neq \emptyset$ **for some** $1 \leq i \leq n$ **then**

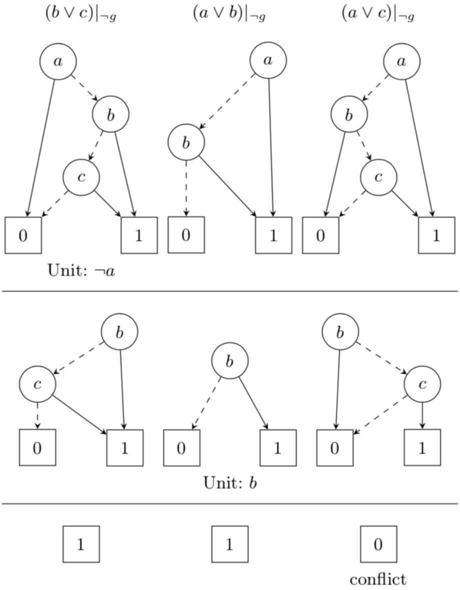
    $f_j := f_j|_{\bigwedge U(f_i)}$ **for all** $1 \leq j \leq n$

until no update to $f_1, \ldots, f_n$

- $U(f)$ = set of literals implied by the BDD f
- Propagates these implied literals through the collection
- If UnitProp$(f_1|_{\neg g}, \ldots, f_n|_{\neg g})$ = "conflict" then $f_1 \wedge \cdots \wedge f_n \models g$

Reverse UnitProp Example





Redundancy Properties for BDDs

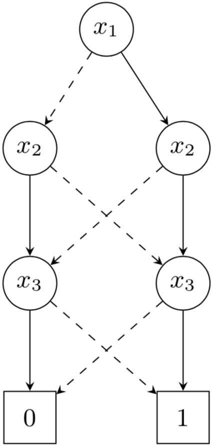
A BDD g is **RUP$_{\text{BDD}}$** w.r.t $f_1 \wedge \cdots \wedge f_n$ if
* UnitProp$(f_1|_{\neg g}, \ldots, f_n|_{\neg g})$ = "conflict", or
* UnitProp$(f_1|_p, \ldots, f_n|_p)$ = "conflict" for each 0-path p in g (**RUP$_{\text{path}}$**)

- Efficiently decidable, generalizes clausal RUP

A BDD g is **PR$_{\text{BDD}}$** w.r.t $f_1 \wedge \cdots \wedge f_n$ if for some ω
* $g|_\omega = 1$, and
* for all $1 \leq i \leq n$ either $f_i|_\omega = 1$ or $f_i|_\omega$ is RUP$_{\text{BDD}}$

- Efficiently checkable, given ω
- Generalizes clausal PR property

Gaussian Elimination with RUP_{BDD}

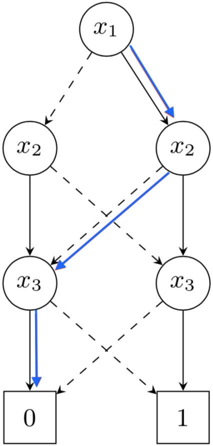


$x_1 \oplus x_2 \oplus x_3 = 1$

```
p cnf 6 16
1 2 3 0            2 4 -6 0
1 -2 -3 0          2 -4 6 0
-1 2 -3 0          -2 4 6 0
-1 -2 3 0          -2 -4 -6 0
1 4 -5 0           3 5 -6 0
1 -4 5 0           3 -5 6 0
-1 4 5 0           -3 5 6 0
-1 -4 -5 0         -3 -5 -6 0
```

- If $f_1 \wedge \cdots \wedge f_n$ includes the CNF of an XOR, the BDD is RUP_{path}

Gaussian Elimination with RUP_{BDD}



$p = x_1 \wedge \neg x_2 \wedge x_3$

```
p cnf 6 16
1 2 3 0
1 -2 -3 0
-1 2 -3 0
-1 -2 3 0
1 4 -5 0
1 -4 5 0
-1 4 5 0
-1 -4 -5 0
2 4 -6 0
2 -4 6 0
-2 4 6 0
-2 -4 -6 0
3 5 -6 0
3 -5 6 0
-3 5 6 0
-3 -5 -6 0
```

- If $f_1 \wedge \cdots \wedge f_n$ includes the CNF of an XOR, the BDD is RUP_{path}

Gaussian Elimination with RUP_{BDD}

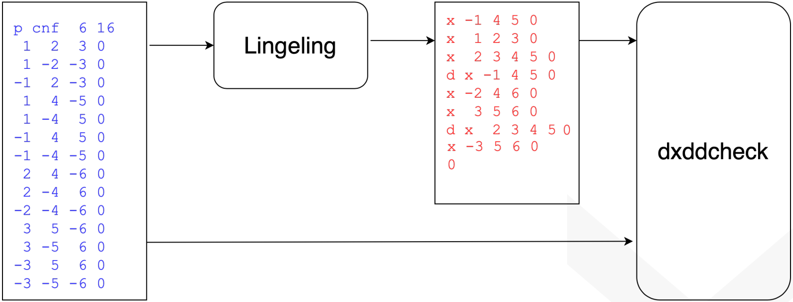
- From constraints X and Y, want to infer $X \oplus Y$

$$\begin{array}{rl} 1 \oplus 2 \oplus 3 & = 1 \\ 1 \oplus 4 \oplus 5 & = 0 \\ \hline 2 \oplus 3 \oplus 4 \oplus 5 & = 1 \end{array}$$

- We show $X|_{\neg(X \oplus Y)} = \neg Y|_{\neg(X \oplus Y)}$, so $X \oplus Y$ is RUP_{BDD}
- Proof system using RUP_{BDD} or PR_{BDD}
  - Easily expresses Gaussian elimination steps
  - Extends corresponding clausal property

Results

- **dxddcheck**: prototype implementation in Python
- Checks proofs in this subsystem capturing Gaussian elim.
  - Allows clause and XOR addition



- Proofs can be extracted from Lingeling output
- dxddcheck checks that the XOR constraint is redundant

Results

Formula	number of variables	number of clauses	solving time (s)	proof lines	proof size (KB)	checking time (s)
rpar_50	148	394	0.1	297	7	0.34
rpar_100	298	794	0.1	597	15	1.35
rpar_200	598	1594	0.2	1197	35	6.67
mchess_19	680	2291	0.0	1077	41	4.07
mchess_21	836	2827	0.1	1317	50	5.09
mchess_23	1008	3419	0.1	1581	63	6.42
urquhart-s5-b2	107	742	0.0	150	7	0.95
urquhart-s5-b3	121	1116	0.1	150	9	1.64
urquhart-s5-b4	114	888	0.0	150	8	1.20

- urquhart benchmarks: Tseitin formulas
- With Gaussian elim, Lingeling solves all almost instantly
- Without Gaussian elim, Lingeling and Kissat timeout
  - Only rpar_50 was solved in $<$ 10 hours
  - Proof in this case was $\approx$ 6911 MB

Conclusion

- Generalized redundancy beyond clauses
- Define properties, proof systems based on redundant BDDs
- Proof systems easily express Gaussian elimination
- Prototype results confirm this approach is practical

Future work: cardinality reasoning, (certified) $\mathrm{PR}_{\mathrm{BDD}}$ proof checker

References I

Sheldon B. Akers.
Binary decision diagrams.
IEEE Trans. Computers, 27(6):509–516, 1978.

Randal E. Bryant and Marijn J. H. Heule.
Generating extended resolution proofs with a BDD-based SAT solver.
In Jan Friso Groote and Kim Guldstrand Larsen, editors, *27th Intl. Conference on Tools and Algorithms for the Construction and Analysis of Systems – TACAS*, volume 12651 of *LNCS*, pages 76–93. Springer, 2021.

Armin Biere.
CaDiCaL, Lingeling, Plingeling, Treengeling and YalSAT entering the SAT competition 2018.
In Marijn J. H. Heule, Matti Järvisalo, and Martin Suda, editors, *Proc. of SAT Competition 2018*, Department of Computer Science Series of Publications B, pages 13–14. University of Helsinki, 2018.

References II

Paul Beame, Henry Kautz, and Ashish Sabharwal.
Towards understanding and harnessing the potential of clause learning.
Journal of Artificial Intelligence Research, 22(1):319–351, 2004.

Randal E. Bryant.
Graph-based algorithms for Boolean function manipulation.
IEEE Transactions on Computers, 35(8):677–691, 1986.

Clark Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli.
Satisfiability modulo theories.
In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, pages 1267–1329. IOS Press, 2021.

Sam Buss and Neil Thapen.
DRAT proofs, propagation redundancy, and extended resolution.
In Mikoláš Janota and Inês Lynce, editors, *22nd Intl. Conference on Theory and Applications of Satisfiability Testing – SAT*, volume 11628 of *LNCS*, pages 71–89. Springer, 2019.

References III

Edmund Clarke, Armin Biere, Richard Raimi, and Yunshan Zhu.
Bounded model checking using satisfiability solving.
Formal Methods in System Design, 19(1):7–34, 2001.

Olivier Coudert and Jean Christophe Madre.
A unified framework for the formal verification of sequential circuits.
In *IEEE Intl. Conference on Computer-Aided Design – ICCAD*, pages 126–129. IEEE Computer Society, 1990.

Robert F. Damiano and James H. Kukula.
Checking satisfiability of a conjunction of BDDs.
In *40th Design Automation Conference – DAC*, pages 818–823. ACM, 2003.

John Franco, Michal Kouril, John Schlipf, Jeffrey Ward, Sean Weaver, Michael Dransfield, and W. Mark Vanfleet.
SBSAT: a state-based, BDD-based satisfiability solver.
In Enrico Giunchiglia and Armando Tacchella, editors, *6th Intl. Conference on Theory and Applications of Satisfiability Testing – SAT*, volume 2919 of *LNCS*, pages 398–410. Springer, 2004.

References IV

Evguenii I. Goldberg and Yakov Novikov.
Verification of proofs of unsatisfiability for CNF formulas.
In *Conference on Design, Automation and Test in Europe– DATE*, pages 886–891. IEEE Computer Society, 2003.

Evguenii I. Goldberg, Mukul R. Prasad, and Robert K. Brayton.
Using SAT for combinational equivalence checking.
In Wolfgang Nebel and Ahmed Jerraya, editors, *Conference on Design, Automation and Test in Europe – DATE*, pages 114–121. IEEE Computer Society, 2001.

Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere.
Short proofs without new variables.
In Leonardo de Moura, editor, *26th Intl. Conference on Automated Deduction – CADE*, 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, *25th Intl. Conference on Tools and Algorithms for the Construction and Analysis of Systems – TACAS*, volume 11427 of *LNCS*, pages 41–58. Springer, 2019.

References V

Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere.
Strong extension-free proof systems.
Journal of Automated Reasoning, 64(3):533–554, 2020.

Marijn J. H. Heule, Oliver Kullmann, and Victor W. Marek.
Solving and verifying the Boolean Pythagorean triples problem via cube-and-conquer.
In Nadia Creignou and Daniel Le Berre, editors, *19th Intl. Conference on Theory and Applications of Satisfiability Testing – SAT*, volume 9710 of *LNCS*, pages 228–245. Springer, 2016.

Marijn J. H. Heule, Benjamin Kiesl, Martina Seidl, and Armin Biere.
PRuning through satisfaction.
In Ofer Strichman and Rachel Tzoref-Brill, editors, *13th Intl. Haifa Verification Conference – HVC*, volume 10629 of *LNCS*, pages 179–194. Springer, 2017.

Boris Konev and Alexei Lisitsa.
Computer-aided proof of Erdős discrepancy properties.
Artificial Intelligence, 224:103–118, 2015.

References VI

Daher Kaiss, Marcelo Skaba, Ziyad Hanna, and Zurab Khasidashvili.
Industrial strength SAT-based alignability algorithm for hardware equivalence verification.
In *7th Intl. Conference on Formal Methods in Computer Aided Design – FMCAD*, pages 20–26. IEEE Computer Society, 2007.

C. Y. Lee.
Representation of switching circuits by binary-decision programs.
The Bell System Technical Journal, 38(4):985–999, 1959.

DoRon B. Motter and Igor L. Markov.
A compressed breadth-first search for satisfiability.
In David M. Mount and Clifford Stein, editors, *4th Intl. Workshop on Algorithm Engineering and Experiments – ALENEX*, volume 2409 of *LNCS*, pages 29–42. Springer, 2002.

Guoqiang Pan and Moshe Y. Vardi.
Search vs. symbolic techniques in satisfiability solving.
In *7th Intl. Conference on Theory and Applications of Satisfiability Testing – SAT*, volume 3542 of *LNCS*, pages 235–250. Springer, 2004.

References VII

Mate Soos, Karsten Nohl, and Claude Castelluccia.
Extending SAT solvers to cryptographic problems.
In Oliver Kullmann, editor, *12th Intl. Conference on Theory and Applications of Satisfiability Testing – SAT*, LNCS, pages 244–257. Springer, 2009.

Grigorii Samuilovich Tseitin.
On the complexity of derivation in propositional calculus.
In Anatolii Olesevich Slissenko, editor, *Studies in Constructive Mathematics and Mathematical Logic*, volume 2, pages 115–125. Steklov Mathematical Institute, 1970.

Hervé J. Touati, Hamid Savoj, Bill Lin, Robert K. Brayton, and Alberto L. Sangiovanni-Vincentelli.
Implicit state enumeration of finite state machines using BDDs.
In *IEEE Intl. Conference on Computer-Aided Design – ICCAD*, pages 130–133. IEEE Computer Society, 1990.

Alasdair Urquhart.
The complexity of propositional proofs.
Bulletin of Symbolic Logic, 1(4):425–467, 12 1995.

References VIII

Andrei Voronkov.
AVATAR: The architecture for first-order theorem provers.
In Armin Biere and Roderick Bloem, editors, *26th Intl. Conference on Computer Aided Verification – CAV*, volume 8559 of *LNCS*, pages 696–710. Springer, 2014.

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, *17th Intl. Conference on Theory and Applications of Satisfiability Testing – SAT*, volume 8561 of *LNCS*, pages 422–429. Springer, 2014.