DRAT Proofs and Extensions

Sam Buss
U.C. San Diego

(joint work with Neil Thapen)
April 12, 2021

5^{th} St. Petersburg Days of Logic and Computability (Online)
Dedicated to the 80^{th} Birthday of V. P. Orevkov

Outline. *Setting: Propositional proof complexity*

- CDCL solvers: DPLL with clause learning. These widely developed solvers decide instances of satisfiability. CDCL solvers seek to either produce a refutation of a set of clauses, or find a satisfying assignment.
- The resolution rule is the primary basis for CDCL refutations.
- Recently, non-implicational inferences rules are used more-and-more. These include (D)RAT, an inference rule which admits inferring non-implied clauses, and many other inference rules including (D)SPR, (D)PR, and (D)SR.
- We focus on proofs that do not use new variables.
- New results reported in this talk include a simulation of DPR by DRAT, short proofs of PHP and other tautologies without new extension variables, and an exponential separation of SPR and RAT via the Bit-PHP (Bit Pigeonhole Principle) when no new variables are permitted.

SAT and SAT Solvers

Setting: Propositional logic.

Satisfiability (SAT)
Input: A set Γ of clauses
- A clause is a disjunction of literals.
- Γ is a CNF formula.

Output one of:
- *Satisfiable*, with a satisfying assignment.
- *Unsatisfiable*.
- Or, *Unknown*.

SAT is NP complete; nonetheless SAT Solvers have been remarkably (!) successful in recent decades, routinely solving instances with 100,000's or 1,000,000's of variables.
Many practitioners even view SAT as being tractable (feasible).

CDCL – Conflict Driven Clause Learning

CDCL = DPLL with Clause Learning. [Marques-Silva,Sakallah'96]
Central ingredients of CDCL:

- Depth-first search plus unit propagationm
- Clause learning (inference) upon conflict
- Restarts

Input: A set Γ of clauses
Output one of:
- *Satisfiable*, with a satisfying assignment.
- *Unsatisfiable*.
- *Unknown*.

For *Unsatisfiable*, the run of the SAT solver can serve as a refutation.
But this presupposes the solver is sound. (Often not true!)
The DRAT systems were developed to provide succinct, quickly checkable refutations in a form that can be readily generated by most SAT solvers.

CDCL solvers use sophisticated decision branching, unit propagation (UP), backtracking, restarts, UIP's, clause learning, etc. But the main inference rule (for learning) is "asymmetric tautology" (AT), based on contradictions from unit propagation:

Notation: $C, D, \ldots$ are clauses. Γ is a set of clauses.

Definition

$\Gamma \models C$ holds iff every truth assignment satisfying Γ also satisfies C.

Definition ($\models_1$)

$\Gamma \models_1 \bot$ holds iff unit propagation derives a conflict from Γ.
For $C = \{p_1, \ldots, p_k\}$ a clause, $\Gamma \models_1 C$ holds iff $\Gamma, \overline{p_1}, \ldots \overline{p_k} \models_1 \bot$.

Note that $\Gamma \models_1 C$ implies $\Gamma \models C$.
The condition $\Gamma \models_1 C$ is polynomial time checkable.

Unit propagation and Input resolution

Unit Propagation: Do repeatedly: If all but one of the literals in a clause C have been set false, set the remaining literal true.

Input Resolution/Trivial Resolution Derivation from Γ:

- Every resolution inference has at least one hypothesis from Γ
- No literal is resolved on more than once. ("Trivial" resolution.)

Theorem (Chang'70, BKS'04)

$\Gamma \models_1 C$ holds iff there is an input/trivial resolution derivation of C from Γ.

Thus, CDCL proofs of unsatisfiability lead to resolution refutations.

Inference with $\models_1$ (Asymmetric Tautology):

Definition (AT - Asymmetric Tautology [Järvisalo-Heule-Biere'12])

A clause C is an AT w.r.t. Γ iff $\Gamma \models_1 C$.

The AT rule is sound since $\Gamma \models C$.

A resolution inference

$$\frac{C \vee p \qquad D \vee \overline{p}}{C \vee D}$$

can be simulated by an AT inference.

Clause Learning

Most CDCL solvers can infer C from Γ with clause learning exactly if $\Gamma \models_1 C$.

Non-implicational inferences

CDCL solvers also frequently infer clauses C that are *not* implied by Γ. For example:

Pure literal: If p appears in Γ but $\overline{p}$ does not, then infer p.

Extension rule: For a new variable x infer three new clauses expressing $x \leftrightarrow q \wedge r$:

$$\overline{q} \vee \overline{r} \vee x, \qquad q \vee \overline{x}, \qquad r \vee \overline{x}.$$

A useful way to think about these are as "wlog" inferences.
Namely, "wlog p is true"
or "wlog it is the case that $x \leftrightarrow q \wedge r$ holds".

Equisatisfiability: These inferences do not change the (un)satisfiability of the set of clauses.

Resolution Blocked Clauses – the inference rule

Definition (Resolution Blocked Clause (RBC))

Let $C := C' \vee p$. Then C is RBC wrt p and Γ if, for each clause $\overline{p} \vee D'$ in Γ, the resolvent $C' \vee D'$ is a tautology.

Definition (BC inference [Kullmann'99])

If C is RBC w.r.t. Γ, then C may be inferred by a BC inference.

Theorem (Equisatisfiability under BC)

In this case, Γ *is satisfiable iff* $\Gamma \cup \{C\}$ *is satisfiable.*

Proof idea: Consider the first step of the Davis-Putnam procedure (applied to p).

RAT - Resolution Asymmetric Tautology

Definition (Resolution Asymmetric Tautology (RAT))

Let $C := C' \vee p$. Then C is RAT wrt p and Γ if, for each clause $\overline{p} \vee D'$ in Γ, the resolvent $C' \vee D'$ is an asymmetric tautology; i.e., $\Gamma \vDash_1 C' \vee D'$.

Definition (RAT inference [Heule-Hunt-Wetzler'13])

If C is RAT w.r.t. Γ, then C may be inferred by a RAT inference.

Theorem (Equisatisfiability under RAT)

In this case, Γ *is satisfiable iff* $\Gamma \cup \{C\}$ *is satisfiable.*

Proof idea: Consider the first step of the Davis-Putnam procedure (applied to p).

BC ≡ RAT ≡ Extended Resolution [Tseitin'68]

Theorem (BC simulates ER [Kullmann'99])

An extension rule can be polynomially simulated by BC inferences. Hence also by RAT inferences.

Proof: For x new, the extension clauses are blocked:

$$\overline{q} \vee \overline{r} \vee x, \qquad q \vee \overline{x}, \qquad r \vee \overline{x}. \qquad \square$$

Theorem ([Kiesl-Rebola-Pardo-Heule'18])

Extended resolution polynomially simulates RAT proofs.

Proofs:
(1) [KRPH'18] give a direct simulation. Or
(2) [Using Bounded Arithmetic]. S_2^1 proves that RAT inferences preserve satisfiability. Thus, follows by Cook's PV theorem.

Proof Traces

As CDCL solvers become more complicated, soundness is a serious problem. Even without "bugs", solvers use many techniques and many optimizations; they interact in subtle ways that can be unsound.

Hence: desirable for SAT solvers to output refutations.

A **RAT proof trace** lists the inferred clauses, each one a RAT consequence of earlier clauses. It ends with the empty clause ($\emptyset$) demonstrating unsatisfiability.
[Heule-Hunt-Wetzer'13]

Earlier approaches used AT (aka RUP, Reverse Unit Propagation). I.e., using just $\models_1$ for inferences.
[Van Gelder'03; Goldberg-Novikov'08]

Proof Traces [HHW'13, WHH'14]

DRAT Proof Trace system: [Heule-Hunt-Wetzler'13; Wetzler-Heule-Hunt'14]

DRAT (= 'D' + 'RAT') Proof Trace (Refutation) consists of a sequence of clauses updating the current set of clauses with two rules:

- RAT inferences: Introduce C by RAT.
- Deletion (D): Remove any clause C.

Inferences preserve satisfiability, so the system is sound.

Often takes longer to verify refutations than generate them. (!)
Deletions help prune the unit propagation search space.

And: Deletions make the RAT system stronger!

World's Longest Maths Proof

A parallel (cube-and-conquer) SAT solver resolved the Boolean Pythagorean Triples Problem (unsatisfiable for $n = 7825$)
DRAT proof size 200TB; compressed to 14TB (clause compression plus bzip2), then to 68GB by special encoding.
Run time: 2 days wall clock time, 37100 CPU hours.
Verification time: About 16000 CPU hours.
[Heule-Kullmann-Marek'16]

LPR - Literal Propagation Redundancy $\equiv$ RAT

RAT can be reformulated as LPR — easier to generalize:
For next definitions: Γ is a set of clauses and

- $C := p \vee C'$ is a clause.
- α is $\overline{C}$: the minimal partial truth assignment falsifying C.

Definition (LPR - Literal Propagation Redundant [HKB'17])

C is Literal Propagation Redundant (LPR) wrt p and Γ if

$$\Gamma|\!\upharpoonright_{\alpha} \vDash_1 (\{C\} \cup \Gamma)|\!\upharpoonright_{\tau},$$

where τ is the same as α, except setting p true. I.e. $\tau(p) = \text{True}$.

Notation: $\Gamma \vDash_1 \Delta$ means that, for each $D \in \Delta$, $\Gamma \vDash_1 D$.

Theorem (LPR $\equiv$ RAT [HKB'17])

A clause C is LPR wrt p and Γ iff it is RAT wrt p and Γ.

Proof of soundness of LPR:

Need show that if Γ is satisfiable, then so is $\Gamma \cup C$.

Suppose $\sigma \vDash \Gamma$, but $\sigma \nvDash C$.
Since $\sigma \nvDash C$, we have $\sigma \supseteq \alpha$.
Let σ' be σ except setting p to True.
Then $\sigma' \supseteq \tau$.
By $\Gamma\restriction\alpha \vDash (\{C\} \cup \Gamma)\restriction\tau$, and since σ satisfies Γ, we have that

$$\sigma' \text{ satisfies } \Gamma \cup C.$$

q.e.d.

Note the proof only needs "$\Gamma\restriction\alpha \vDash (\{C\} \cup \Gamma)\restriction\tau$", not "$\Gamma\restriction\alpha \vDash_1 (\{C\} \cup \Gamma)\restriction\tau$". But "$\vDash_1$" makes the LPR condition polynomial time checkable.

PR, SPR - (Subset) Propagation Redundancy

Recall $C := p \vee C'$ is a clause, and α is $\overline{C}$.

Definition (PR - Propagation Redundant [Heule-Kiesl-Biere'17])

C is *Propagation Redundant (PR)* wrt Γ if, for some partial assignment τ,

$$\Gamma\restriction\alpha \vDash_1 (\{C\} \cup \Gamma)\restriction\tau.$$

Notation: $\Gamma \vDash_1 \Delta$ means that, for each $D \in \Delta$, $\Gamma \vDash_1 D$.

Definition (SPR - Subset Propagation Redundant [HKB'17])

C is *Subset Propagation Redundant (SPR)* wrt Γ if, it is PR with $dom(\tau) = dom(\alpha)$.

SR - Substitution Redundancy [B.-Thapen'20]

Recall $C := p \vee C'$ is a clause, and α is $\overline{C}$.

Definition (SR - Substitution Redundant [B.-Thapen'20])

C is *Substitution Redundant (SR)* wrt Γ if, for some partial substitution τ,

$$\Gamma\restriction\alpha \vDash_1 (\{C\} \cup \Gamma)\restriction\tau.$$

A **substitution** maps a variable to 0 or 1 or to a literal x.

Theorem

BC, LPR/RAT, SPR, PR, SR are increasing in applicability. They all preserve (un)satisfiability.

Proof systems

Using the inference rules BC, RAT, SPR, PR, SR, define proof systems

- Without deletion:

$$\text{BC}, \quad \text{RAT}, \quad \text{SPR}, \quad \text{PR}, \quad \text{SR}$$

- With deletion (D):

$$\text{DBC}, \quad \text{DRAT}, \quad \text{DSPR}, \quad \text{DPR}, \quad \text{DSR}.$$

Theorem

All of these systems are polynomially equivalent to extended resolution.

Proof: BC is the weakest, and polynomially simulates extended resolution. Conversely, S_2^1 proves the soundness of DSR. □

Not using new variables

The strength of extended resolution depends strongly on the ability to introduce new variables.

Likewise the simulations of extended resolution by systems BC through DSR depend on the ability to introduce new variables.

However, for practical SAT solvers, we do not yet have any good hueristics for how to introduce new variables with extension.

This raises the question: **What are the power of systems such as BC, RAT, PR, SR etc. when restricted to not allow new variables to be introduced?**

Prior results

Theorem ([Kiesl-Rebola-Pardo-Heule'18])

Without new variables, DBC *polynomially simulates* DRAT.

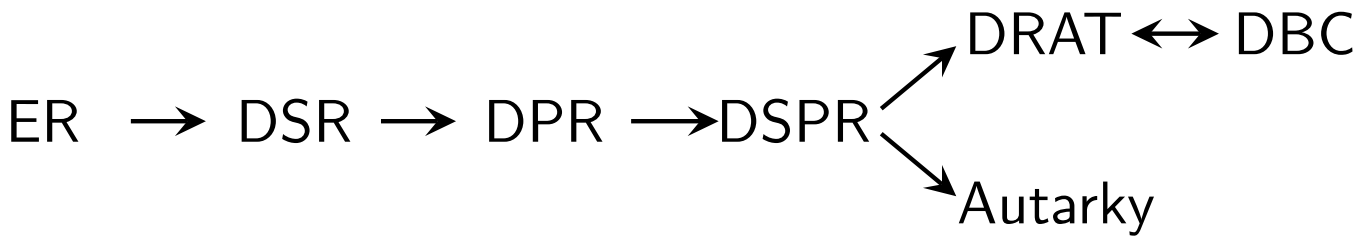
Proof requires introducing and deleting clauses to make the "blocked" condition hold, then undoing the extra introductions and deletions. □

Theorem ([Heule-Biere'18])

With only one additional new variable, DRAT *polynomially simulates* DPR.

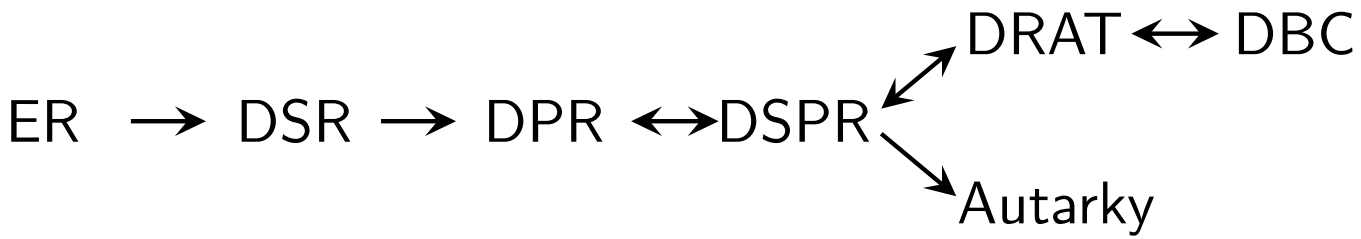
Proof involves first introducing a new variable equivalent to the clause to be inferred. □

Prior results



All systems except ER (extended resolution) are not allowed to introduce new variables. Arrows indicate polynomial simulation.

New simulations



Theorem ([B.-Thapen'19])

Without new variables, DRAT *polynomially simulates* DPR.

Proof idea: Use one step of the Davis-Putnam procedure to eliminate the use of one variable from a PR refutation. Then use the simulation of [Heule-Biere'18]. Result is complex, but still polynomial size. □

Short proofs without new variables

Theorem ([Heule-Kiesl-Biere'17, Heule-Biere'18])

The PHP_n^{n+1} *and Tseitin clauses have polynomial size refutations in* SPR *(resp.,* PR*) without introducing new variables.*

Theorem ([B.-Thapen'19])

The following have short proofs in SPR *(hence* DBC*) without introducing new variables:*

- *Parity principles*
- *Clique-Coloring principles*
- *Tseitin tautologies on degree d expander graphs*
- *Bit pigeonhole principles (*$\mathrm{Bit\text{-}PHP}$*)*
- *Or-ification and Xor-ification and Index Gadget-ification.*

Bit Pigeonhole Principle (Bit-PHP$_n$)

Informally: No 1-1 function from $[n]$ to $[n-1]$ with function values encoded in binary.

Let $n = 2^k$.

Variables:

$p_1^x, \ldots p_k^x$, for all $x \in [n]$.

These variables encode, in binary, integers i_x.

Clauses express:

$i_x \neq a \vee i_y \neq a$, for each $x, y \in [n]$, each $a \in [n] \setminus \{0\}$.

$i_x \neq 0$, for each $x \in [n]$.

These clauses are unsatisfiable.

Some lower bounds without Deletion

Theorem ([Kullmann'99])

BC *without new variables requires exponential size refutations for* PHP_n^{n+1}.

Theorem ([B.-Thapen'19])

RAT *without new variables requires exponential size refutations for* Bit-PHP_n^{n+1}.

This gives an exponential separation between PR and RAT when it is not allowed to introduce new variables.

Proof idea: A random restriction applied to a short RAT refutation gives a narrow width refutation. In a narrow refutation, RAT inferences can be replaced by narrow width resolution derivations. □

Corollary ([B.-Thapen'19])

Without new variables:

- RAT *does not polynomially simulate* DRAT.
- RAT *does not polynomially simulate* SPR.

(Separations are exponential.)

Open Questions

- Exponential lower bounds on proof length for DSPR or DPR or DSR when not allowed to introduce new variables? Candidates:
  - Graph PHP [Beame,pc], or
  - Random 3-SAT, or
  - Even Coloring Principle [Nordström,pc]
- Can DSR without new variables polynomially simulate extended resolution?
- [Heule-Kiesl-Seidel-Biere'17] have been able to automatically generate short refutations of the PHP using SDSL (Satisfaction-Driven Clause Learning). Can this method be made more broadly applicable?

Thank you!