Extending SAT Solvers with Extended Resolution

Sam Buss

In honor of Eduard Čech

Mathematics Institute
Czech Academy of Sciences
April 17, 2024

Collaborative grants 1989-1999

N.S.F. and Czechoslovakian Academy of Science Cooperative Research Grant 1996-1999. Investigators: S. Buss (P.I.), R. Impagliazzo, J. Krajíček, T. Pitassi, P. Pudlák, S. Rudich, J. Sgall.

U.S. and Czech Republic Joint Research Grant 1993-1996. Investigators are: S. Buss, P. Clote, R. Impagliazzo, J. Krajíček(P.I.), P. Pudlák, J. Sgall, and G. Takeuti. Funds cooperative research in the U.S. and in Czechoslovakia.

N.S.F. and Czechoslovakian Academy of Science Cooperative Research Grant 1989-1991. Investigators: J. Bečvář (P.I.), S. Buss (P.I.), P. Clote, P. Hájek, J. Krajíček, P. Pudlák, G. Takeuti.

San Diego, Summer 1990

Colloque Takeuti, 1993

Logic Colloquium, Prague, 1998

This talk discusses:

- **CDCL SAT solvers and proof systems.**
  - CDCL solvers are remarkably successful in solving very large instances of SAT, routinely solving SAT instances with 100,000's or 1,000,000's of variables.
  - CDCL solvers find an instance of SAT to be unsatisfiable, (mostly!) by implicitly finding a ***resolution refutation***.
- **DRAT, substitution propagation redundant (SR) and related inference systems.**
  These extend CDCL solvers to potentially (but indirectly) simulate the full strength of ***extended resolution*** which is strictly stronger than resolution.
- **Dual Implication Points (DIPs).** A proposal for strategies for directly incorporating extended resolution into SAT solvers.

"SAT" = "Satisfiability" of CNF formulas
"CDCL" = "Conflict Driven Clause Learning"
"DRAT" = "Deletion & Reverse Asymmetric Tautologies"

Satisfiability (SAT) problem: Given a conjunctive normal form (CNF) formula, determine if there is Boolean truth assignment that makes it true.

CNF formula: Variables range over *True* and *False*.
A formula is a conjunction (Boolean and) of clauses (disjunctions, i.e. a Boolean or, of literals).

SAT is NP-complete. Furthermore, it is "efficiently" NP-complete in that common NP-complete problems are reducible to near-linear size SAT instances.

Thm: For M a non-deterministic Turing machine, the problem of whether it halts in k steps can be reduced to an instance of SAT for a formula of size $k(\log k)^{O(1)}$. ("quasilinear size").

SAT solvers have many applications in software and hardware verification, scheduling, optimization, (combinatorial) theorem proving, etc.

I. Conflict Driven Clause Learning (CDCL)

CDCL is the most commonly used algorithm for SAT. Its core consists of:

- **Depth-first search (DFS) + Unit propagation.**
  - Picks a variable to set true or false.
  - Sets all unit propagation consequences.
  - Repeats as long as possible.
- **Conflict/Learning:** When a some clause is falsified (a "***conflict***)", some new clause is learned (inferred), and the DFS backtracks.

CDCL (with restarts) simulates Resolution:[BKS'04, PD'11, AFT'11]

A **resolution inference** is:

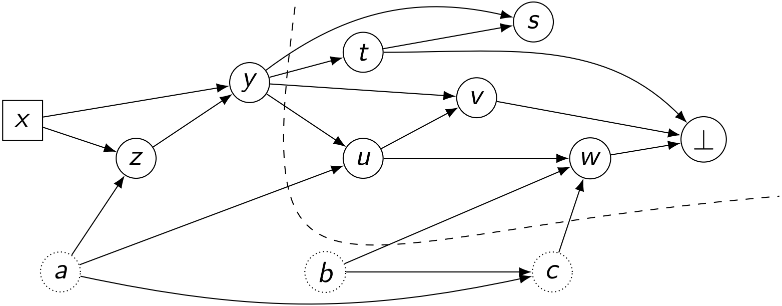
$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D}$$

A resolution refutation ends with $\emptyset$.

Example: $\{\ x,\ \overline{x} \vee y,\ \overline{x} \vee \overline{y}\ \}$ is unsatisfiable.

$$\cfrac{\cfrac{x \qquad \overline{x} \vee y}{y} \qquad \cfrac{x \qquad \overline{x} \vee \overline{y}}{\overline{y}}}{\emptyset}$$

Example of a conflict graph and first-UIP learning



The CNF contains the clauses $\overline{x} \vee \overline{a} \vee z$, $\overline{x} \vee \overline{z} \vee y$, $\overline{y} \vee t$, $\overline{y} \vee v$, $\overline{y} \vee \overline{a} \vee u$, $\overline{y} \vee \overline{u} \vee v$, $\overline{u} \vee \overline{b} \vee \overline{c} \vee w$, $\overline{t} \vee \overline{v} \vee \overline{w}$ and $\overline{a} \vee \overline{b} \vee c$.

x is the latest **decision literal** (i.e, last chosen by the DFS)

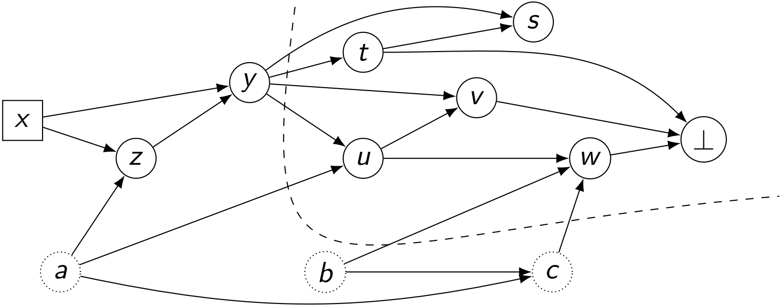
a, b, c were set at earlier decision levels.

The **first-UIP** literal is y. ("UIP" = "Unique Implication Point".)

The **learned clause** is $\overline{\boldsymbol{a}} \vee \overline{\boldsymbol{b}} \vee \overline{\boldsymbol{c}} \vee \overline{\boldsymbol{y}}$.

It can be inferred by a "trivial" resolution refutation.

Example of a conflict graph and first-UIP learning



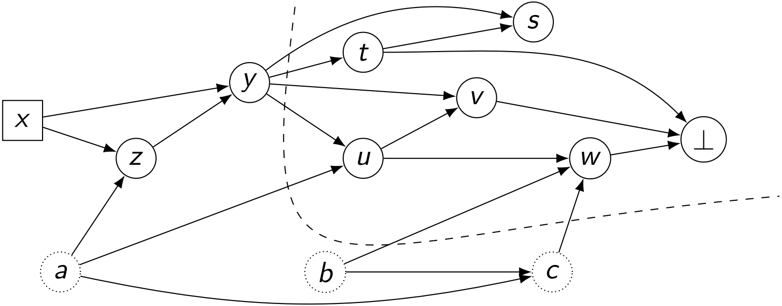
The CNF contains the clauses $\overline{x} \vee \overline{a} \vee z$, $\overline{x} \vee \overline{z} \vee y$, $\overline{y} \vee t$, $\overline{y} \vee v$, $\overline{y} \vee \overline{a} \vee u$, $\overline{y} \vee \overline{u} \vee v$, $\overline{u} \vee \overline{b} \vee \overline{c} \vee w$, $\overline{t} \vee \overline{v} \vee \overline{w}$ and $\overline{a} \vee \overline{b} \vee c$.

Once x, a, b, c have been set, unit propagation gives successively z, y, t, s, u, v, w, and finally $\perp$.

The **first-UIP** literal is y.

The **learned clause** is $\overline{\boldsymbol{a}} \vee \overline{\boldsymbol{b}} \vee \overline{\boldsymbol{c}} \vee \overline{\boldsymbol{y}}$.

Example of a conflict graph and first-UIP learning



The CNF contains the clauses $\overline{x} \vee \overline{a} \vee z$, $\overline{x} \vee \overline{z} \vee y$, $\overline{y} \vee t$, $\overline{y} \vee v$, $\overline{y} \vee \overline{a} \vee u$, $\overline{y} \vee \overline{u} \vee v$, $\overline{u} \vee \overline{b} \vee \overline{c} \vee w$, $\overline{t} \vee \overline{v} \vee \overline{w}$ and $\overline{a} \vee \overline{b} \vee c$.

By backtracking to the maximum decision level of a, b, c, the learned clause $\boldsymbol{\overline{a} \vee \overline{b} \vee \overline{c} \vee \overline{y}}$ becomes ***asserting***, allowing $\overline{y}$ to be inferred by unit propagation.
This in turn can trigger further unit propagation.

II. Propositional Proof Systems (for SAT Solvers)

Resolution: proofs reason with clauses.

Frege proofs reason with arbitrary Boolean formulas $(\land, \lor, \neg, \rightarrow)$.
Uses Modus Ponens, e.g.

Extended Frege / Extended Resolution: Extends Frege by allowing ***new*** variables u to be introduced as abbreviations, e.g.

$$u \leftrightarrow x \land y \qquad \text{(or more complicated formulas).}$$

II. Propositional Proof Systems (for SAT Solvers)

Resolution: proofs reason with clauses.

Frege proofs reason with arbitrary Boolean formulas $(\wedge, \vee, \neg, \rightarrow)$.
Uses Modus Ponens, e.g.

Extended Frege / Extended Resolution: Extends Frege by allowing ***new*** variables u to be introduced as abbreviations, e.g.

$$u \leftrightarrow x \wedge y \qquad \text{(or more complicated formulas).}$$

DRAT / Propagation redundancy / SR proofs - in theory, have the same strength as extended Frege.

II. Propositional Proof Systems (for SAT Solvers)

Weaker systems: Tree-like resolution; Regular resolution.

Resolution: proofs reason with clauses.

Frege proofs reason with arbitrary Boolean formulas ($\land, \lor, \neg, \to$). Uses Modus Ponens, e.g.

Extended Frege / Extended Resolution: Extends Frege by allowing ***new*** variables u to be introduced as abbreviations, e.g.

$$u \leftrightarrow x \land y \qquad \text{(or more complicated formulas).}$$

DRAT / Propagation redundancy / SR proofs - in theory, have the same strength as extended Frege.

II. Propositional Proof Systems (for SAT Solvers)

Weaker systems: Tree-like resolution; Regular resolution.

Resolution: proofs reason with clauses.

Intermediate (?) systems: AC^0-Frege, Cutting Planes, Nullstellensatz, Polynomial Calculus, Sum-of-Squares, MaxSat, Computer Algebra Systems, Algebraic proof systems, Semi-algebraic proof systems.

Frege proofs reason with arbitrary Boolean formulas $(\land, \lor, \neg, \rightarrow)$. Uses Modus Ponens, e.g.

Extended Frege / Extended Resolution: Extends Frege by allowing ***new*** variables u to be introduced as abbreviations, e.g.

$$u \leftrightarrow x \land y \qquad \text{(or more complicated formulas).}$$

DRAT / Propagation redundancy / SR proofs - in theory, have the same strength as extended Frege.

II. Propositional Proof Systems (for SAT Solvers)

Weaker systems: Tree-like resolution; Regular resolution.

Resolution: proofs reason with clauses.

Intermediate (?) systems: AC^0-Frege, Cutting Planes, Nullstellensatz, Polynomial Calculus, Sum-of-Squares, MaxSat, Computer Algebra Systems, Algebraic proof systems, Semi-algebraic proof systems.

Frege proofs reason with arbitrary Boolean formulas $(\land, \lor, \neg, \rightarrow)$. Uses Modus Ponens, e.g.

Extended Frege / Extended Resolution: Extends Frege by allowing ***new*** variables u to be introduced as abbreviations, e.g.

$$u \leftrightarrow x \land y \qquad \text{(or more complicated formulas).}$$

DRAT / Propagation redundancy / SR proofs - in theory, have the same strength as extended Frege.

Potentially stronger: Arbitrary symmetry breaking.

Pigeonhole Principle (PHP_n)**:** $\neg\exists$ injective $f : [n+1] \to [n]$:

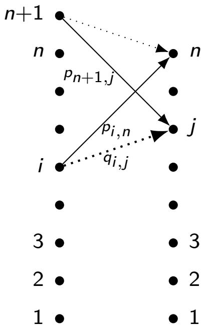
$$\begin{array}{cl} p_{i,1} \vee p_{i,2} \vee \cdots \vee p_{i,n} & \text{for } i = 1, \ldots, n+1 \\ \overline{p_{i,j}} \vee \overline{p_{i',j}} & \text{for } i < i' \leq n+1 \text{ and } j = 1, \ldots, n. \end{array}$$

Theorem [Haken'86, Cook-Reckhow'79, B'87,...] PHP_n has polynomial size Frege and extended Frege proofs, but requires exponential size resolution proofs.

Extended resolution proof uses extension axioms:

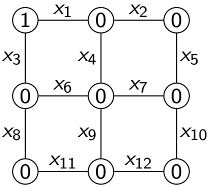
$$q_{i,j} \leftrightarrow p_{i,j} \vee (p_{i,n} \wedge p_{n+1,j})$$

to reduce $\mathrm{PHP}_n(\vec{p})$ to $\mathrm{PHP}_{n-1}(\vec{q})$. Iterate to reduce to $\mathrm{PHP}_2(\cdots)$.



The **Tseitin principle** states that the following is impossible:

In a (low degree) graph G each node has a fixed 0/1 charge. The total charge is odd. Variables label the edges of G. For each node u; the parity of the incident edges equals the node's charge.



e.g., $x_1 \oplus x_3 = 1$
is expressed as
$(x_1 \vee x_3) \;\wedge\; (\overline{x_1} \vee \overline{x_3})$

Theorem: [Urquhart'87] For G a connected expander graph, the Tseitin principles require exponential size resolution refutations.

Corollary: PHP and Tseitin do not have short CDCL refutations.

Theorem: [Tseitin'66; $\approx$B'87] The Tseitin principles have polynomial size Frege and extended Frege proofs.

III. DRAT, Propagation Redundancy, SR

DRAT and Propagation Redundancy (PR) are extensions to CDCL designed to

- Allow checking the correctness of CDCL generated proofs, including special CDCL techniques that go beyond resolution. (E.g., "without loss of generality" reasoning, or symmetries.)
- Allow inferring ***non-implied*** clauses,
- Extend CDCL to have the full power of extended resolution.

The next slides define two of the more powerful versions...

Kullmann, *On a Generalization of Extended Resolution*, Discrete Applied Math., 1999
Järvisalo, Heule, Biere, *Inprocessing Rules*, IJCAR '2012.
Heule, Hunt, Wetzler, *Verifying Refutations with Extended Resolution*, CADE 2013.
Heule, Hunt, Wetzler, *Trimming while Checking Clausal Proofs*, FMCAD, 2013.
Wetzler, Huele, Hunt; *DRAT-trim: Efficient Checking and Trimming Using Expressive Clausal Proofs*, SAT 2014.
Heule, Kiesl, Seidel, Biere, *PRuning Through Satisfaction*, HVC 2017.
Heule, Kiesl, Biere, *Short Proofs Without New Variables*, CADE 2017.
Huele, Biere, *What a Difference a Variable Makes*, TACAS 2018.
Kiesl-Rebola-Pardo-Heule, *Extended Resolution Simulates DRAT*, IJCAR 2018
B., Thapen, *DRAT and Propagation Redundancy Without New Variables*, LMCS 2021.

The Largest Math Proof is a DRAT proof

[Heule-Kullmann-Marek'16]

- Resolved the Boolean Pythogorean Triples Problem (false for $n = 7825$)
  (Thm: Every 2-coloring of $\{1, \ldots, 7825\}$ has a monochromatic Pathagorean triple.)
- 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.

SAT Competitions now routinely require SAT solvers to produce DRAT or DPR proofs of unsatisfiability.

Definition: Let Γ be a CNF formula, and C a clause. A ***Propagation Redundancy (PR) inference*** can derive C from Γ provided, there is a partial truth assignment τ such that

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

- $\overline{C}$ is the conjunction of the negations of the literals in C.
- "$\restriction\tau$" means apply the truth assignment τ and simplifying.
- We do not have that $\Gamma \vDash C$,
  only that **Γ is satisfiable iff $\Gamma \wedge C$ is satisfiable**.
- $\Pi \vDash_1 \Delta$ means that for each clause $D \in \Delta$,
  the CNF $\Gamma \wedge \overline{D}$ yields a contradiction by unit propagation.
- The "$\vDash_1$" condition is polynomial time checkable
  (since unit propagation can be carried out efficiently, in fact in linear time).

Definition: ***SPR (Subset PR)*** is PR with the additional condition that the domain of τ is the variables in C.

Definition: [B-Thapen] Let Γ be a CNF formula, and C a clause. A ***Substitution Propagation Redundancy (SR) inference*** can derive C from Γ provided, there is a ***substitution*** τ such that

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

- The only difference is that now τ is a substitution, namely it maps variables to a constant 0 or 1 (False) or (True) or to a literal.
- The condition is still satisfiability-preserving and polynomial-time checkable.

—

Remark: DPR and DSR add a Clause Deletion rule.

Theorem. [Kullman] The extension rule can be simulated by PR.

Proof sketch: To infer $u \leftrightarrow x \vee y$:

- To infer the clauses $\overline{x} \vee u$ and $\overline{y} \vee u$, use the truth assignment $\tau(u) = \mathit{True}$.
- To infer the clause $\overline{u} \vee x \vee y$, use the truth assignment $\tau(u) = \mathit{False}$.

Theorem. [$\approx$ Kiesl,Rebola Pardo,Heule'18]
Extended resolution simulates PR and SR.

One disadvantage of extended resolution is that it complicates proof search: namely, there are too many options for what formulas to abbreviate with new extension variables.

Accordingly: one can define systems that ***disallow new variables***:

Definition: The inferences SPR^{-}, PR^{-} and SR^{-} defined like SPR, PR and SR, but restricting the inferred clause C to not contain any new variables.

Theorem: [B.-Thapen'21] The system SPR^{-} has polynomial size proofs of:

- Pigeonhole principles [Heule-Kiesl-Biere'17] and Bit-PHP.
- Tseitin tautologies.
- Parity principles.
- Or-fication and Xor-ification obfuscations of easy formulas.

Example of how prove PHP_n with SR$^-$.

- Infer (one at a time, for $i = 1, \ldots, n$) the unit clauses

  $$\overline{p}_{i,n} \qquad \text{(expressing that pigeon } i \text{ is not in hole } n\text{).}$$

  by using the substitution τ that maps $p_{i,j}$'s to $p_{n+1,j}$'s and vice-versa. (Interchanging pigeons i and n.)
- Deduce (by unit propagation) the PHP_{n-1} clauses.
- Iterate!

IV. Dual Implication Points [B.-Chung-Ganesh-Oliveras'24]

A new proposal for choosing pairs of variables x and y for introducing new variables by extended resolution as

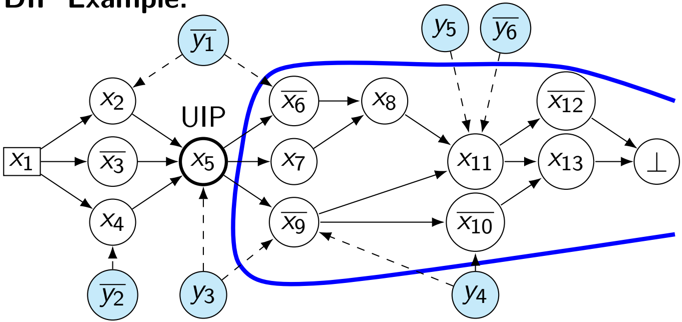
$$u \leftrightarrow x \wedge y.$$

Based on examination of the conflict graph to finds pairs of variables that form a ***dual implication point (DIP)***.

- The notion of DIP generalizes the notion of UIP.
- A DIP is a pair of variables x and y that together with literals from lower levels imply a contradiction.
- DIP's occur very frequently in conflict graphs.
- There can be quadratically many DIP's, but all such pairs can be identified in linear time using a compressed representation.
- Finding DIP's in linear time is based on an effective version of Menger's theorem for 3-connected vertices in a graph.

Prior work: GlucoseER [Audemard-Katsirelos-Simon'10] & TiniSatX [Huang'10].)

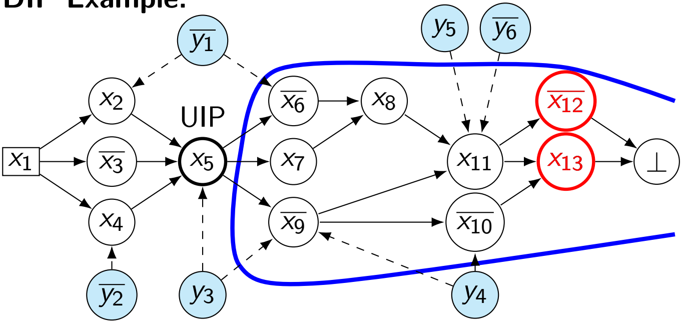
DIP Example:



Extension axiom	Learned clauses (post-DIP and pre-DIP)
$z \leftrightarrow (\overline{x_{12}} \wedge x_{13})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (x_{11} \wedge x_{13})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg x_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (\overline{x_{10}} \wedge x_{11})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (\overline{x_9} \wedge x_{11})$	$\neg z \vee \neg y_4$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (x_8 \wedge \overline{x_9})$	$\neg z \vee \neg y_4 \vee \neg y_5 \vee y_6$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee z$

One criterion: The choice of DIP should make z "useful for unit propagation" by appearing both negatively and especially positively in learned clauses.

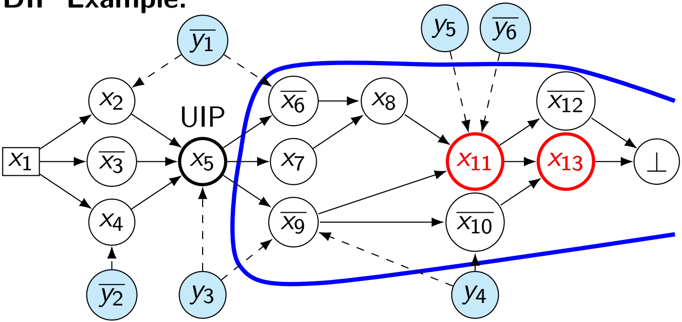
DIP Example:



Extension axiom	Learned clauses (post-DIP and pre-DIP)
$z \leftrightarrow (\overline{x_{12}} \wedge x_{13})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (x_{11} \wedge x_{13})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg x_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (\overline{x_{10}} \wedge x_{11})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (\overline{x_9} \wedge x_{11})$	$\neg z \vee \neg y_4$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (x_8 \wedge \overline{x_9})$	$\neg z \vee \neg y_4 \vee \neg y_5 \vee y_6$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee z$

One criterion: The choice of DIP should make z "useful for unit propagation" by appearing both negatively and especially positively in learned clauses.

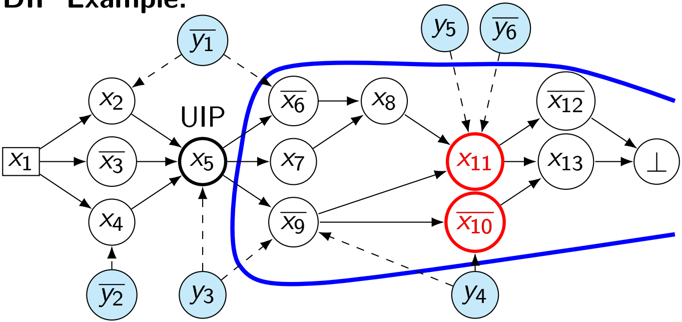
DIP Example:



Extension axiom	Learned clauses (post-DIP and pre-DIP)
$z \leftrightarrow (\overline{x_{12}} \wedge x_{13})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (x_{11} \wedge x_{13})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg x_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (\overline{x_{10}} \wedge x_{11})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (\overline{x_9} \wedge x_{11})$	$\neg z \vee \neg y_4$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (x_8 \wedge \overline{x_9})$	$\neg z \vee \neg y_4 \vee \neg y_5 \vee y_6$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee z$

One criterion: The choice of DIP should make z "useful for unit propagation" by appearing both negatively and especially positively in learned clauses.

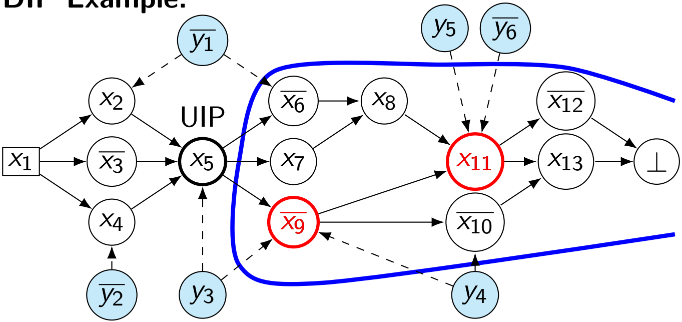
DIP Example:



Extension axiom	Learned clauses (post-DIP and pre-DIP)
$z \leftrightarrow (\overline{x_{12}} \wedge x_{13})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (x_{11} \wedge x_{13})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg x_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (\overline{x_{10}} \wedge x_{11})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (\overline{x_9} \wedge x_{11})$	$\neg z \vee \neg y_4$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (x_8 \wedge \overline{x_9})$	$\neg z \vee \neg y_4 \vee \neg y_5 \vee y_6$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee z$

One criterion: The choice of DIP should make z "useful for unit propagation" by appearing both negatively and especially positively in learned clauses.

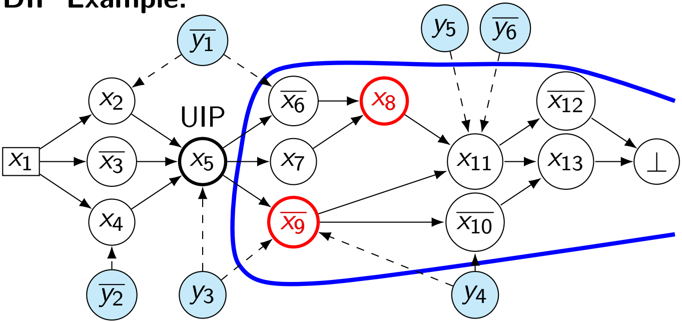
DIP Example:



Extension axiom	Learned clauses (post-DIP and pre-DIP)
$z \leftrightarrow (\overline{x_{12}} \wedge x_{13})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (x_{11} \wedge x_{13})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg x_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (\overline{x_{10}} \wedge x_{11})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (\overline{x_9} \wedge x_{11})$	$\neg z \vee \neg y_4$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (x_8 \wedge \overline{x_9})$	$\neg z \vee \neg y_4 \vee \neg y_5 \vee y_6$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee z$

One criterion: The choice of DIP should make z "useful for unit propagation" by appearing both negatively and especially positively in learned clauses.

DIP Example:



Extension axiom	Learned clauses (post-DIP and pre-DIP)
$z \leftrightarrow (\overline{x_{12}} \wedge x_{13})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (x_{11} \wedge x_{13})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg x_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (\overline{x_{10}} \wedge x_{11})$	$\neg z$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (\overline{x_9} \wedge x_{11})$	$\neg z \vee \neg y_4$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee \neg y_5 \vee y_6 \vee z$
$z \leftrightarrow (x_8 \wedge \overline{x_9})$	$\neg z \vee \neg y_4 \vee \neg y_5 \vee y_6$ and $\neg x_5 \vee y_1 \vee \neg y_3 \vee \neg y_4 \vee z$

One criterion: The choice of DIP should make z "useful for unit propagation" by appearing both negatively and especially positively in learned clauses.

First rounds of experiments on DIP-based extension rules (xMapleSatLcm), using prior SAT competition problems:

- Heuristics for choosing DIP's tried so far include "closest", " middle" and "random". Filtered optionally by activity or LBD (glue). Best results obtained without the optional filtering.
- There are *a lot* of DIP's, so we choose a DIP only after it has arisen multiple times (either 5 or 20 times)
- Decision variable selection uses the usual VSIDS — and seems to work well also for extension variables.
- Deletion of inactive extension variables is also useful.
- Comparison with GlucoseER, an earlier ER-based solver [Audemard-Katsirelos-Simon'10].

Experimental results:

- DIP-based extension and GlucoseER both perform very well on Tseitin principles, random XOR formulas and "intersecting interval" tautologies — much better than traditional CDCL. The Tseitin formulas considered are both grid-based and on random graphs (degree 4 and degree 6).
- The DIP method produces polynomial size refutations for Tseitin, albeit in (slow-growing) exponential time.
- DIP-based extension performs comparably to CDCL on a wide range of other problems, with an overhead of $\approx$ 2%-5%.

Open Problems / Future Work

- Explore more broadly the capabilities of the DIP-based extension framework. This framework provides a great deal of flexibility in extension formulas, and there remain main possibilities to explore. Can it be useful across a wider range of SAT problems?
- Is the DIP-based extension system capable of simulating the full extension resolution system? Similarly for the restrictive LER system used by GlucoseER?
- Explain how the DIP-based extension can discover small proofs of Tseitin principles, and random XORs. Show explicitly how such proofs are possible (e.g., by hand). So far, this is completely open.
- Investigate using DIP's to learn more 2-clauses.
- Give superpolynomial lower bounds for strong redundancy proof systems without new variables, such as SPR^-, PR^- or SR^-. So far, this is done only for RAT^- [B.-Thapen'21].

Thank you!