Towards Faster Conflict-Driven Pseudo-Boolean Solving

Jakob Nordström

KTH Royal Institute of Technology
Stockholm, Sweden

Theory and Practice of Satisfiability Solving
Casa Matemática Oaxaca
August 28, 2018

Some Acknowledgements and Caveats Right Away...

Survey heavily indebted to:

- Jan Elffers
- Stephan Gocht
- Daniel Le Berre
- João Marques-Silva
- and many, many others...

Some Acknowledgements and Caveats Right Away. . .

Survey heavily indebted to:

- Jan Elffers
- Stephan Gocht
- Daniel Le Berre
- João Marques-Silva
- and many, many others. . .

By necessity, very selective—more material and references in:

- Dixon's thesis [Dix04]
- Pseudo-Boolean chapter [RM09] in *Handbook of Satisfiability*
- Full details of cited papers at end of slides (also online at `www.csc.kth.se/~jakobn/research/TalkCMO18.pdf`)

The Satisfiability Problem (SAT)

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

- Variables should be set to true $(= 1)$ or false $(= 0)$
- Constraint $(x \lor \overline{y} \lor z)$: means x or z should be true or y false
- $\land$ means all constraints should hold simultaneously

Is there a truth value assignment satisfying all constraints?

The Satisfiability Problem (SAT)

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

- Variables should be set to true $(= 1)$ or false $(= 0)$
- Constraint $(x \lor \overline{y} \lor z)$: means x or z should be true or y false
- $\land$ means all constraints should hold simultaneously

Is there a truth value assignment satisfying all constraints?

Can computers solve the SAT problem efficiently?

- Mentioned already in Gödel's famous letter in 1956 to von Neumann
- Intense research in TCS ever since early 1970s [Coo71, Lev73]
- Now one of Millennium Prize Problems in mathematics [Mil00]

SAT in Practice

- Dramatic progress last 15–20 years on SAT solvers using conflict-driven clause learning (CDCL) [MS96, BS97, MMZ+01]
- Today routinely used to solve large-scale real-world problems (100,000s or 1,000,000s of variables)
- But... There are also small formulas (just ∼100 variables) that are completely beyond reach of even the very best SAT solvers
- Limitations of CDCL
  1. Clauses weak formalism for encoding constraints
  2. Method of reasoning used (resolution) also weak

Pseudo-Boolean Reasoning to the Rescue?

- Pseudo-Boolean (PB) linear constraints are stronger than clauses

Compare

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 3$$

and

$$\begin{aligned}
&(x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee x_3 \vee x_5) \wedge (x_1 \vee x_2 \vee x_3 \vee x_6)\\
\wedge&(x_1 \vee x_2 \vee x_4 \vee x_5) \wedge (x_1 \vee x_2 \vee x_4 \vee x_6) \wedge (x_1 \vee x_2 \vee x_5 \vee x_6)\\
\wedge&(x_1 \vee x_3 \vee x_4 \vee x_5) \wedge (x_1 \vee x_3 \vee x_4 \vee x_6) \wedge (x_1 \vee x_3 \vee x_4 \vee x_6)\\
\wedge&(x_1 \vee x_4 \vee x_5 \vee x_6) \wedge (x_2 \vee x_3 \vee x_4 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4 \vee x_6)\\
\wedge&(x_2 \vee x_3 \vee x_5 \vee x_6) \wedge (x_2 \vee x_4 \vee x_5 \vee x_6) \wedge (x_3 \vee x_4 \vee x_5 \vee x_6)
\end{aligned}$$

Pseudo-Boolean Reasoning to the Rescue?

- Pseudo-Boolean (PB) linear constraints are stronger than clauses

Compare

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 3$$

and

$$\begin{aligned}
&(x_1 \lor x_2 \lor x_3 \lor x_4) \land (x_1 \lor x_2 \lor x_3 \lor x_5) \land (x_1 \lor x_2 \lor x_3 \lor x_6) \\
\land&(x_1 \lor x_2 \lor x_4 \lor x_5) \land (x_1 \lor x_2 \lor x_4 \lor x_6) \land (x_1 \lor x_2 \lor x_5 \lor x_6) \\
\land&(x_1 \lor x_3 \lor x_4 \lor x_5) \land (x_1 \lor x_3 \lor x_4 \lor x_6) \land (x_1 \lor x_3 \lor x_4 \lor x_6) \\
\land&(x_1 \lor x_4 \lor x_5 \lor x_6) \land (x_2 \lor x_3 \lor x_4 \lor x_5) \land (x_2 \lor x_3 \lor x_4 \lor x_6) \\
\land&(x_2 \lor x_3 \lor x_5 \lor x_6) \land (x_2 \lor x_4 \lor x_5 \lor x_6) \land (x_3 \lor x_4 \lor x_5 \lor x_6)
\end{aligned}$$

- And pseudo-Boolean reasoning exponentially more powerful in theory

Pseudo-Boolean Reasoning to the Rescue?

- Pseudo-Boolean (PB) linear constraints are stronger than clauses

Compare

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 3$$

and

$$\begin{aligned}
&(x_1 \lor x_2 \lor x_3 \lor x_4) \land (x_1 \lor x_2 \lor x_3 \lor x_5) \land (x_1 \lor x_2 \lor x_3 \lor x_6) \\
\land &(x_1 \lor x_2 \lor x_4 \lor x_5) \land (x_1 \lor x_2 \lor x_4 \lor x_6) \land (x_1 \lor x_2 \lor x_5 \lor x_6) \\
\land &(x_1 \lor x_3 \lor x_4 \lor x_5) \land (x_1 \lor x_3 \lor x_4 \lor x_6) \land (x_1 \lor x_3 \lor x_4 \lor x_6) \\
\land &(x_1 \lor x_4 \lor x_5 \lor x_6) \land (x_2 \lor x_3 \lor x_4 \lor x_5) \land (x_2 \lor x_3 \lor x_4 \lor x_6) \\
\land &(x_2 \lor x_3 \lor x_5 \lor x_6) \land (x_2 \lor x_4 \lor x_5 \lor x_6) \land (x_3 \lor x_4 \lor x_5 \lor x_6)
\end{aligned}$$

- And pseudo-Boolean reasoning exponentially more powerful in theory

- But PB solvers less efficient than CDCL in practice(!?)

Outline

1. Conflict-Driven Clause Learning
   - CDCL by Example
   - Pseudocode and Analysis

2. Conflict-Driven Pseudo-Boolean Solving
   - Some Preliminaries
   - Pseudo-Boolean Solving Using Saturation
   - Pseudo-Boolean Solving Using Division
   - More About Pseudo-Boolean Reasoning

3. Open Problems and Future Directions

Slides online at www.csc.kth.se/~jakobn/research/TalkCMO18.pdf

Modern SAT Solving

DPLL method [DP60, DLL62]

- Assign values to variables (in some smart way)
- Backtrack when conflict with falsified clause

Modern SAT Solving

DPLL method [DP60, DLL62]

- Assign values to variables (in some smart way)
- Backtrack when conflict with falsified clause

Conflict-driven clause learning (CDCL) [MS96, BS97, MMZ^{+}01]

- Analyse conflicts in more detail — add new clauses to formula
- More efficient backtracking
- Also let conflicts guide other heuristics

Modern SAT Solving

DPLL method [DP60, DLL62]

- Assign values to variables (in some smart way)
- Backtrack when conflict with falsified clause

Conflict-driven clause learning (CDCL) [MS96, BS97, MMZ⁺01]

- Analyse conflicts in more detail — add new clauses to formula
- More efficient backtracking
- Also let conflicts guide other heuristics

Variable Assignments

Two kinds of assignments — illustrate on our example formula:

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

Variable Assignments

Two kinds of assignments — illustrate on our example formula:

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

Decision

Free choice to assign value to variable

Notation $w \stackrel{\mathsf{d}}{=} 0$

Variable Assignments

Two kinds of assignments — illustrate on our example formula:

$$(u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{u} \vee w) \wedge (\overline{u} \vee \overline{w})$$

$\boxed{w \stackrel{\mathsf{d}}{=} 0}$

Decision

Free choice to assign value to variable

Notation $w \stackrel{\mathsf{d}}{=} 0$

Variable Assignments

Two kinds of assignments — illustrate on our example formula:

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$w \stackrel{\mathsf{d}}{=} 0$

Decision

Free choice to assign value to variable

Notation $w \stackrel{\mathsf{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $w = 0$, clause $\overline{u} \lor w$ forces $u = 0$

Notation $u \stackrel{\overline{u} \lor w}{=} 0$ ($\overline{u} \lor w$ is reason)

Variable Assignments

Two kinds of assignments — illustrate on our example formula:

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$w \stackrel{\mathsf{d}}{=} 0$

$u \stackrel{\overline{u} \lor w}{=} 0$

Decision

Free choice to assign value to variable

Notation $w \stackrel{\mathsf{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $w = 0$, clause $\overline{u} \lor w$ forces $u = 0$

Notation $u \stackrel{\overline{u} \lor w}{=} 0$ ($\overline{u} \lor w$ is reason)

Variable Assignments

Two kinds of assignments — illustrate on our example formula:

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$\boxed{w \stackrel{\mathsf{d}}{=} 0}$

$\boxed{u \stackrel{\overline{u} \lor w}{=} 0}$

$\boxed{x \stackrel{\mathsf{d}}{=} 0}$

Decision

Free choice to assign value to variable

Notation $w \stackrel{\mathsf{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $w = 0$, clause $\overline{u} \lor w$ forces $u = 0$

Notation $u \stackrel{\overline{u} \lor w}{=} 0$ ($\overline{u} \lor w$ is reason)

Always propagate if possible, otherwise decide

Until satisfying assignment or conflict clause

Variable Assignments

Two kinds of assignments — illustrate on our example formula:

$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$

$\boxed{w \stackrel{\mathsf{d}}{=} 0}$

$\boxed{u \stackrel{\overline{u} \lor w}{=} 0}$

$\boxed{x \stackrel{\mathsf{d}}{=} 0}$

$\boxed{y \stackrel{u \lor x \lor y}{=} 1}$

Decision

Free choice to assign value to variable

Notation $w \stackrel{\mathsf{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $w = 0$, clause $\overline{u} \lor w$ forces $u = 0$

Notation $u \stackrel{\overline{u} \lor w}{=} 0$ ($\overline{u} \lor w$ is reason)

Always propagate if possible, otherwise decide

Until satisfying assignment or conflict clause

Variable Assignments

Two kinds of assignments — illustrate on our example formula:

$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$

$\boxed{w \overset{\mathsf{d}}{=} 0}$

$\boxed{u \overset{\overline{u} \lor w}{=} 0}$

$\boxed{x \overset{\mathsf{d}}{=} 0}$

$\boxed{y \overset{u \lor x \lor y}{=} 1}$

$\boxed{z \overset{x \lor \overline{y} \lor z}{=} 1}$

Decision

Free choice to assign value to variable

Notation $w \overset{\mathsf{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $w = 0$, clause $\overline{u} \lor w$ forces $u = 0$

Notation $u \overset{\overline{u} \lor w}{=} 0$ ($\overline{u} \lor w$ is reason)

Always propagate if possible, otherwise decide

Until satisfying assignment or conflict clause

Variable Assignments

Two kinds of assignments — illustrate on our example formula:

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$\boxed{w \stackrel{\mathsf{d}}{=} 0}$

$\boxed{u \stackrel{\overline{u} \lor w}{=} 0}$

$\boxed{x \stackrel{\mathsf{d}}{=} 0}$

$\boxed{y \stackrel{u \lor x \lor y}{=} 1}$

$\boxed{z \stackrel{x \lor \overline{y} \lor z}{=} 1}$

$\boxed{\stackrel{\overline{y} \lor \overline{z}}{\bot}}$

Decision

Free choice to assign value to variable

Notation $w \stackrel{\mathsf{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $w = 0$, clause $\overline{u} \lor w$ forces $u = 0$

Notation $u \stackrel{\overline{u} \lor w}{=} 0$ ($\overline{u} \lor w$ is reason)

Always propagate if possible, otherwise decide

Until satisfying assignment or conflict clause

Conflict-Driven Clause Learning

Time to analyse this conflict!

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$\boxed{w \stackrel{\mathsf{d}}{=} 0}$

$\boxed{u \stackrel{\overline{u} \lor w}{=} 0}$

$\boxed{x \stackrel{\mathsf{d}}{=} 0}$

$\boxed{y \stackrel{u \lor x \lor y}{=} 1}$

$\boxed{z \stackrel{x \lor \overline{y} \lor z}{=} 1}$

$\boxed{\stackrel{\overline{y} \lor \overline{z}}{\bot}}$

Conflict-Driven Clause Learning

Time to analyse this conflict!

$$(u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{u} \vee w) \wedge (\overline{u} \vee \overline{w})$$

$w \overset{\mathsf{d}}{=} 0$

$u \overset{\overline{u} \vee w}{=} 0$

$x \overset{\mathsf{d}}{=} 0$

$y \overset{u \vee x \vee y}{=} 1$

$z \overset{x \vee \overline{y} \vee z}{=} 1$

$\overset{\overline{y} \vee \overline{z}}{\perp}$

Could backtrack by flipping last decision

Conflict-Driven Clause Learning

Time to analyse this conflict!

$$(u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{u} \vee w) \wedge (\overline{u} \vee \overline{w})$$

$w \stackrel{\mathsf{d}}{=} 0$

$u \stackrel{\overline{u} \vee w}{=} 0$

$x \stackrel{\mathsf{d}}{=} 0$

$y \stackrel{u \vee x \vee y}{=} 1$

$z \stackrel{x \vee \overline{y} \vee z}{=} 1$

$\stackrel{\overline{y} \vee \overline{z}}{\bot}$

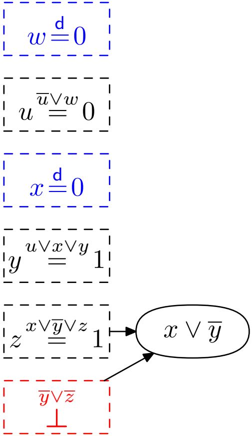
Could backtrack by flipping last decision

But want to learn from conflict and cut away as much of search space as possible

Conflict-Driven Clause Learning

Time to analyse this conflict!

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Could backtrack by flipping last decision

But want to learn from conflict and cut away as much of search space as possible

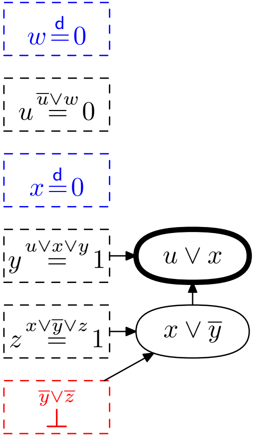
Case analysis over z for last two clauses:

- $x \lor \overline{y} \lor z$ wants $z = 1$
- $\overline{y} \lor \overline{z}$ wants $z = 0$
- Merge & remove z — must satisfy $x \lor \overline{y}$

Conflict-Driven Clause Learning

Time to analyse this conflict!

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Could backtrack by flipping last decision

But want to learn from conflict and cut away as much of search space as possible

Case analysis over z for last two clauses:

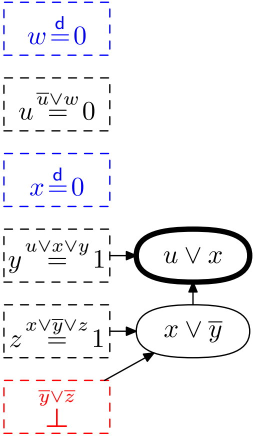
- $x \lor \overline{y} \lor z$ wants $z = 1$
- $\overline{y} \lor \overline{z}$ wants $z = 0$
- Merge & remove z — must satisfy $x \lor \overline{y}$

Repeat until only 1 variable after last decision — learn that clause (1UIP) and backjump

Complete Example of CDCL Execution

Backjump: roll back max #assignments so that last variable still flips

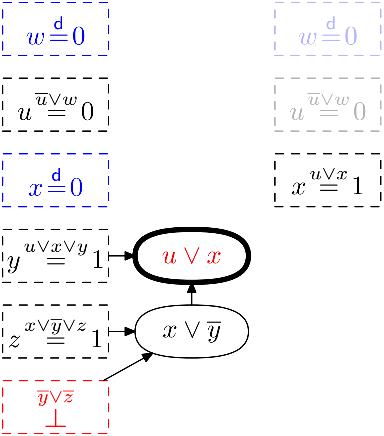
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Complete Example of CDCL Execution

Backjump: roll back max #assignments so that last variable still flips

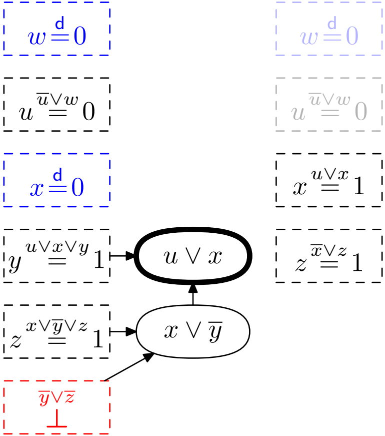
$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$



Complete Example of CDCL Execution

Backjump: roll back max #assignments so that last variable still flips

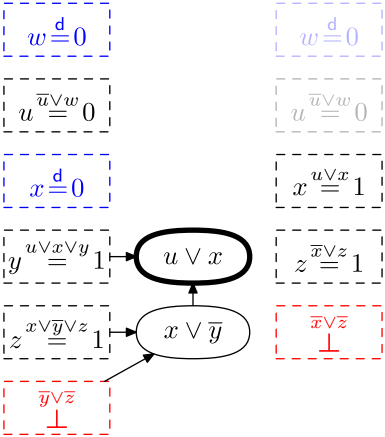
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Complete Example of CDCL Execution

Backjump: roll back max #assignments so that last variable still flips

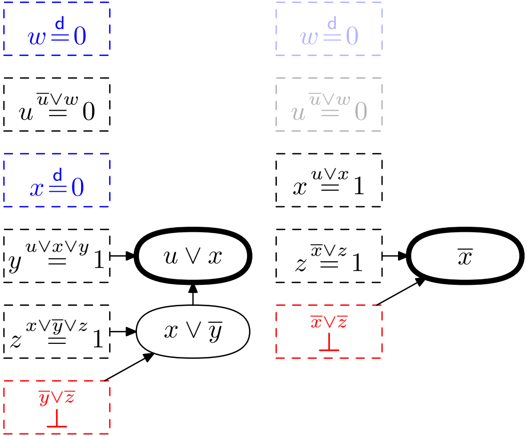
$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$



Complete Example of CDCL Execution

Backjump: roll back max #assignments so that last variable still flips

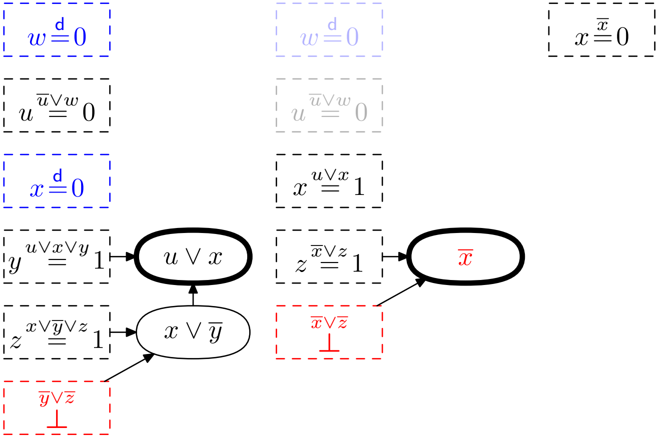
$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$



Complete Example of CDCL Execution

Backjump: roll back max #assignments so that last variable still flips

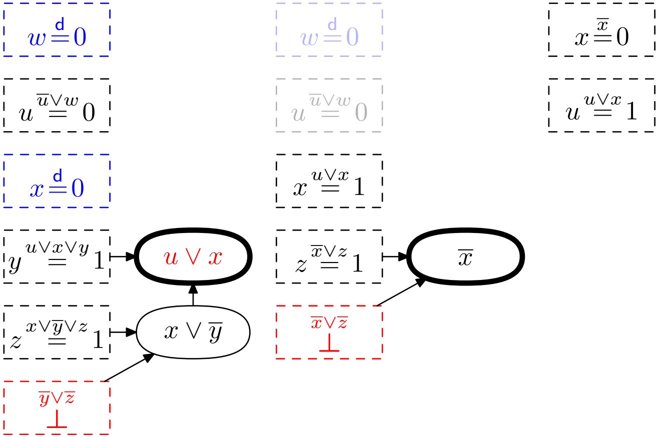
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Complete Example of CDCL Execution

Backjump: roll back max #assignments so that last variable still flips

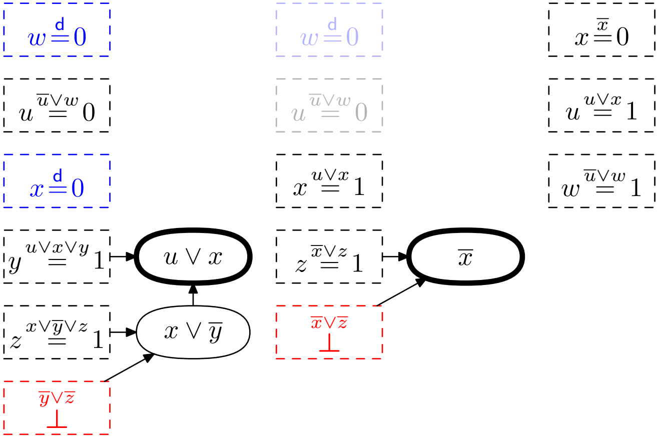
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Complete Example of CDCL Execution

Backjump: roll back max #assignments so that last variable still flips

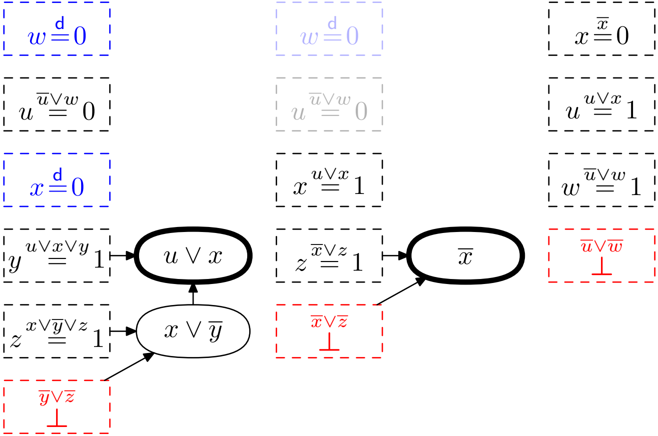
$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$



Complete Example of CDCL Execution

Backjump: roll back max #assignments so that last variable still flips

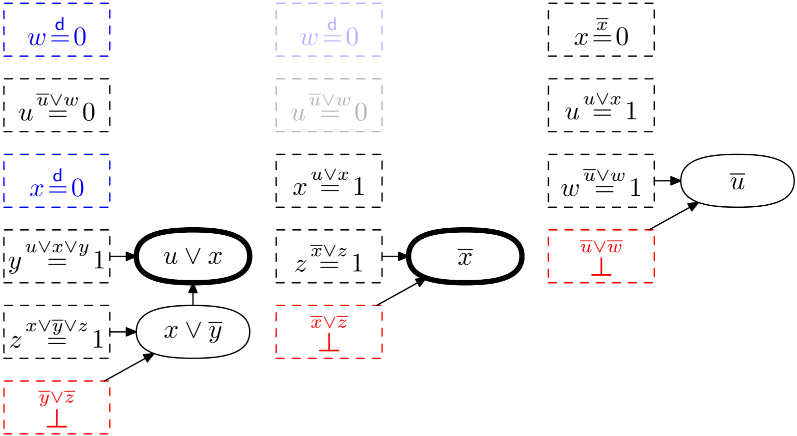
$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$



Complete Example of CDCL Execution

Backjump: roll back max #assignments so that last variable still flips

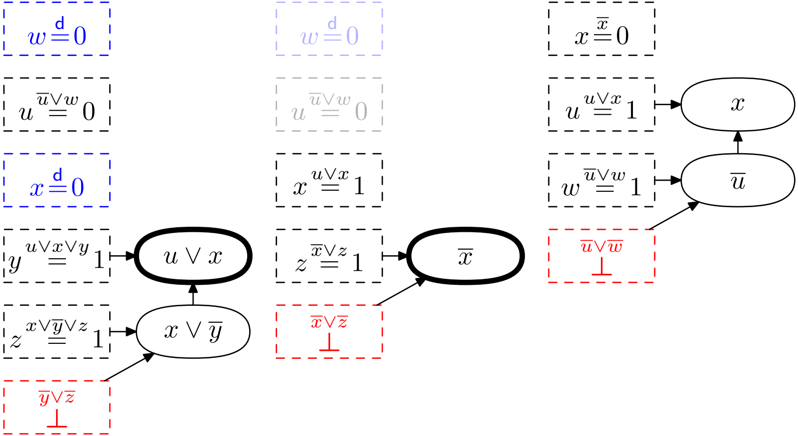
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Complete Example of CDCL Execution

Backjump: roll back max #assignments so that last variable still flips

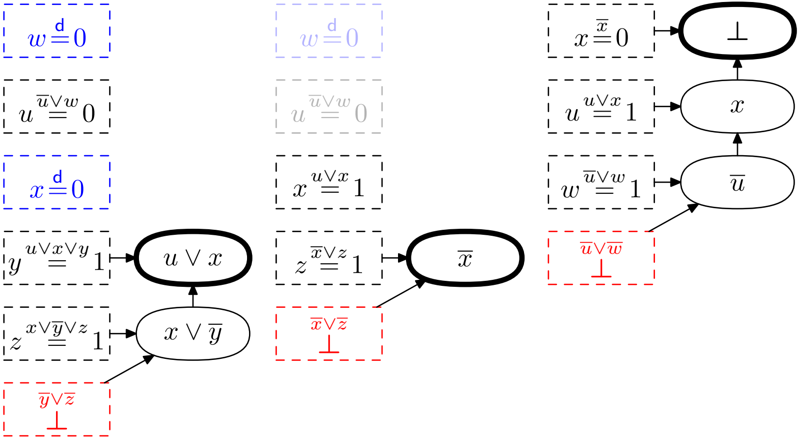
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Complete Example of CDCL Execution

Backjump: roll back max #assignments so that last variable still flips

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



CDCL Main Loop Pseudocode (High Level)

forever do
- **if** *current assignment falsifies clause* **then**
  - apply learning scheme to derive new clause;
  - **if** *learned clause empty* **then** output `UNSATISFIABLE` and exit;
  - **else**
    - add learned clause and backjump
  - **end**
- **else if** *all variables assigned* **then** output `SATISFIABLE` and exit;
- **else if** *exists unit clause C propagating x to value $b \in \{0, 1\}$* **then**
  - add propagated assignment $x \overset{C}{=} b$
- **else if** *time to restart* **then**
  - remove all variable assignments
- **else**
  - **if** *time for clause database reduction* **then**
    - erase (roughly) half of learned clauses in memory
  - **end**
  - use decision scheme to choose assignment $x \overset{\mathrm{d}}{=} b$;
- **end**

end

CDCL Main Loop Pseudocode (High Level)

forever do
  if *current assignment falsifies clause* **then**
    apply learning scheme to derive new clause;
    if *learned clause empty* **then** output `UNSATISFIABLE` and exit;
    else
      add learned clause and backjump
    end
  else if *all variables assigned* **then** output `SATISFIABLE` and exit;
  else if *exists unit clause* C *propagating* x *to value* $b \in \{0,1\}$ **then**
    add propagated assignment $x \stackrel{C}{=} b$
  else if *time to restart* **then**
    remove all variable assignments
  else
    if *time for clause database reduction* **then**
      erase (roughly) half of learned clauses in memory
    end
    use decision scheme to choose assignment $x \stackrel{\mathrm{d}}{=} b$;
  end
end

CDCL Analysis and the Resolution Proof System

How to analyse CDCL performance?

Many intricate, hard-to-understand heuristics

Best(?) rigorous method: Focus on underlying method of reasoning

CDCL Analysis and the Resolution Proof System

How to analyse CDCL performance?
Many intricate, hard-to-understand heuristics
Best(?) rigorous method: Focus on underlying method of reasoning

Resolution proof system

- Start with clauses of formula
- Derive new clauses by **resolution rule**

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

- Done when contradiction $\perp$ in form of empty clause derived

CDCL Analysis and the Resolution Proof System

How to analyse CDCL performance?
Many intricate, hard-to-understand heuristics
Best(?) rigorous method: Focus on underlying method of reasoning

Resolution proof system

- Start with clauses of formula
- Derive new clauses by **resolution rule**

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

- Done when contradiction $\perp$ in form of empty clause derived

When run on unsatisfiable formula, CDCL generates resolution proof*
So lower bounds on proof size $\Rightarrow$ lower bounds on running time

CDCL Analysis and the Resolution Proof System

How to analyse CDCL performance?
Many intricate, hard-to-understand heuristics
Best(?) rigorous method: Focus on underlying method of reasoning

Resolution proof system

- Start with clauses of formula
- Derive new clauses by **resolution rule**

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

- Done when contradiction $\perp$ in form of empty clause derived

When run on unsatisfiable formula, CDCL generates resolution proof*
So lower bounds on proof size $\Rightarrow$ lower bounds on running time

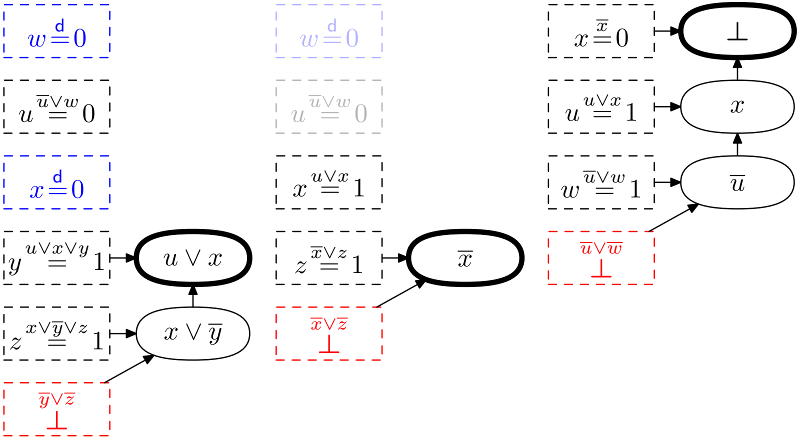
(*) Ignores preprocessing, but we don't have time to go into this

Resolution Proofs from CDCL Executions

Obtain resolution proof. . .

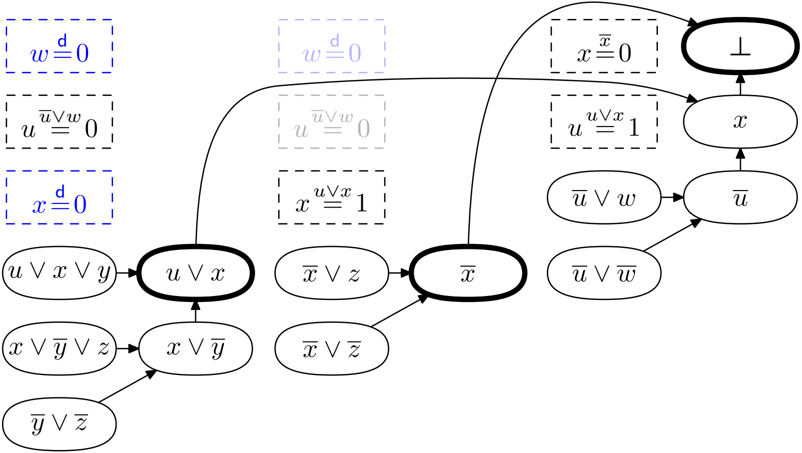
Resolution Proofs from CDCL Executions

Obtain resolution proof from our example CDCL execution. . .



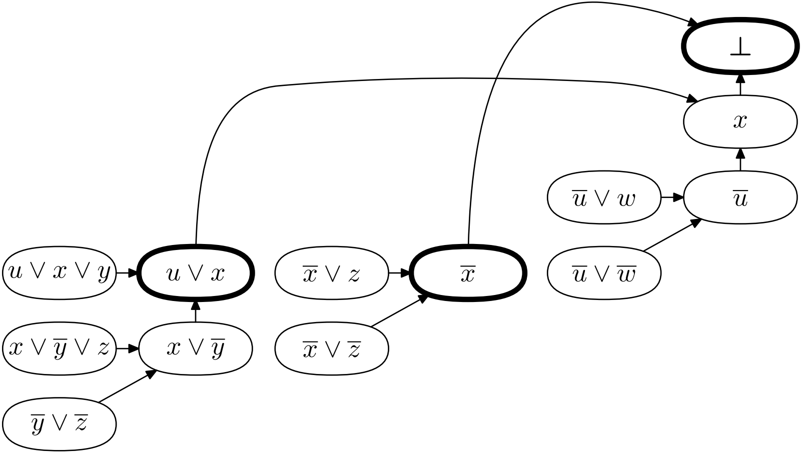
Resolution Proofs from CDCL Executions

Obtain resolution proof from our example CDCL execution by stringing together conflict analyses:



Resolution Proofs from CDCL Executions

Obtain resolution proof from our example CDCL execution by stringing together conflict analyses:



Current state of affairs

- State-of-the-art CDCL solvers often perform amazingly well ("SAT is easy in practice")
- Very poor theoretical understanding:
  - Why do heuristics work?
  - Why are applied instances easy?
- Paradox: resolution quite weak proof system; many strong lower bounds for "obvious" formulas, e.g., [Hak85, Urq87, BW01, MN14]
- Explore stronger reasoning methods (potential exponential speed-up)
- In particular, pseudo-Boolean solving (a.k.a. 0-1 integer programming) corresponding to cutting planes proof system
- Importantly, extends to pseudo-Boolean optimization (won't talk about that—instead listen to Daniel Le Berre's talk in the afternoon)

Pseudo-Boolean Constraints and Normalized Form

In this talk, "pseudo-Boolean" refers to 0-1 integer linear constraints

Pseudo-Boolean Constraints and Normalized Form

In this talk, "pseudo-Boolean" refers to 0-1 integer linear constraints

Convenient to use non-negative linear combinations of literals, a.k.a. normalized form

$$\sum_i a_i \ell_i \geq A$$

- coefficients a_i: non-negative integers
- degree (of falsity) A: positive integer
- literals ℓ_i: x_i or $\overline{x}_i$ (where $x_i + \overline{x}_i = 1$)

Pseudo-Boolean Constraints and Normalized Form

In this talk, "pseudo-Boolean" refers to 0-1 integer linear constraints

Convenient to use non-negative linear combinations of literals, a.k.a. normalized form

$$\sum_i a_i \ell_i \geq A$$

- coefficients a_i: non-negative integers
- degree (of falsity) A: positive integer
- literals ℓ_i: x_i or $\overline{x}_i$ (where $x_i + \overline{x}_i = 1$)

(All constraints in what follows assumed to be implicitly normalized)

Some Types of Pseudo-Boolean Constraints

1. Clauses are pseudo-Boolean constraints

$$x \vee \overline{y} \vee z \quad \Leftrightarrow \quad x + \overline{y} + z \geq 1$$

Some Types of Pseudo-Boolean Constraints

1. Clauses are pseudo-Boolean constraints

$$x \lor \overline{y} \lor z \quad \Leftrightarrow \quad x + \overline{y} + z \geq 1$$

2. Cardinality constraints

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 3$$

Some Types of Pseudo-Boolean Constraints

1. Clauses are pseudo-Boolean constraints

$$x \lor \overline{y} \lor z \quad \Leftrightarrow \quad x + \overline{y} + z \geq 1$$

2. Cardinality constraints

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 3$$

3. General constraints

$$x_1 + 2\overline{x}_2 + 3x_3 + 4\overline{x}_4 + 5x_5 \geq 7$$

Approaches to Pseudo-Boolean Solving

Conversion to disjunctive clauses

- Lazy approach: learn clauses from PB constraints
  - *Sat4j* [LP10] (one of versions in library)

Approaches to Pseudo-Boolean Solving

Conversion to disjunctive clauses

- Lazy approach: learn clauses from PB constraints
  - *Sat4j* [LP10] (one of versions in library)
- Eager approach: re-encode to clauses and run CDCL
  - *MiniSat+* [ES06]
  - *Open-WBO* [MML14]
  - *NaPS* [SN15]

Approaches to Pseudo-Boolean Solving

Conversion to disjunctive clauses

- Lazy approach: learn clauses from PB constraints
  - *Sat4j* [LP10] (one of versions in library)
- Eager approach: re-encode to clauses and run CDCL
  - *MiniSat+* [ES06]
  - *Open-WBO* [MML14]
  - *NaPS* [SN15]

Native reasoning with pseudo-Boolean constraints

- *PRS* [DG02]
- *Galena* [CK05]
- *Pueblo* [SS06]
- *Sat4j* [LP10]
- *RoundingSat* [EN18]

Approaches to Pseudo-Boolean Solving

Conversion to disjunctive clauses

- Lazy approach: learn clauses from PB constraints
  - *Sat4j* [LP10] (one of versions in library)
- Eager approach: re-encode to clauses and run CDCL
  - *MiniSat+* [ES06]
  - *Open-WBO* [MML14]
  - *NaPS* [SN15]

Native reasoning with pseudo-Boolean constraints

- *PRS* [DG02]
- *Galena* [CK05]
- *Pueblo* [SS06]
- *Sat4j* [LP10]
- *RoundingSat* [EN18]

Conflict-Driven Search in a Pseudo-Boolean Setting

Want to do "same thing" as CDCL but with linear constraints

- Variable assignments
  1. Always propagate forced assignment if possible
  2. Otherwise make assignment using decision heuristic
- At conflict
  1. Do conflict analysis to derive new constraint
  2. Add new constraint to instance
  3. Backjump by rolling back max #assignments so that variable flips

Propagation, Conflict, and Slack

Let ρ current assignment of solver (a.k.a. trail)
Represent as $\rho = \{$(ordered) set of literals assigned true$\}$

Propagation, Conflict, and Slack

Let ρ current assignment of solver (a.k.a. trail)
Represent as $\rho = \{$(ordered) set of literals assigned true$\}$

Slack measures how far ρ is from falsifying $\sum_i a_i \ell_i \geq A$

$$slack(\textstyle\sum_i a_i \ell_i \geq A; \rho) = \displaystyle\sum_{\ell_i \text{ not falsified by } \rho} a_i - A$$

Propagation, Conflict, and Slack

Let ρ current assignment of solver (a.k.a. trail)
Represent as $\rho = \{$(ordered) set of literals assigned true$\}$

Slack measures how far ρ is from falsifying $\sum_i a_i \ell_i \geq A$

$$slack(\textstyle\sum_i a_i \ell_i \geq A; \rho) = \displaystyle\sum_{\ell_i \text{ not falsified by } \rho} a_i - A$$

Consider $C : x_1 + 2\overline{x}_2 + 3x_3 + 4\overline{x}_4 + 5x_5 \geq 7$

ρ	$slack(C; \rho)$	comment

Propagation, Conflict, and Slack

Let ρ current assignment of solver (a.k.a. trail)
Represent as $\rho = \{$(ordered) set of literals assigned true$\}$

Slack measures how far ρ is from falsifying $\sum_i a_i\ell_i \geq A$

$$slack(\textstyle\sum_i a_i\ell_i \geq A; \rho) = \displaystyle\sum_{\ell_i \text{ not falsified by } \rho} a_i - A$$

Consider $C : x_1 + 2\overline{x}_2 + 3x_3 + 4\overline{x}_4 + 5x_5 \geq 7$

ρ	$slack(C;\rho)$	comment
$\{\}$	8	

Propagation, Conflict, and Slack

Let ρ current assignment of solver (a.k.a. trail)
Represent as $\rho = \{$(ordered) set of literals assigned true$\}$

Slack measures how far ρ is from falsifying $\sum_i a_i \ell_i \geq A$

$$slack(\textstyle\sum_i a_i \ell_i \geq A; \rho) = \displaystyle\sum_{\ell_i \text{ not falsified by } \rho} a_i - A$$

Consider $C : x_1 + 2\overline{x}_2 + 3x_3 + 4\overline{x}_4 + 5x_5 \geq 7$

ρ	$slack(C;\rho)$	comment
$\{\}$	8	
$\{\overline{x}_5\}$	3	propagates $\overline{x}_4$ (coefficient $>$ slack)

Propagation, Conflict, and Slack

Let ρ current assignment of solver (a.k.a. trail)
Represent as $\rho = \{$(ordered) set of literals assigned true$\}$

Slack measures how far ρ is from falsifying $\sum_i a_i \ell_i \geq A$

$$slack(\textstyle\sum_i a_i \ell_i \geq A; \rho) = \displaystyle\sum_{\ell_i \text{ not falsified by } \rho} a_i - A$$

Consider $C : x_1 + 2\overline{x}_2 + 3x_3 + 4\overline{x}_4 + 5x_5 \geq 7$

ρ	$slack(C;\rho)$	comment
$\{\}$	8	
$\{\overline{x}_5\}$	3	propagates $\overline{x}_4$ (coefficient $>$ slack)
$\{\overline{x}_5, \overline{x}_4\}$	3	propagation doesn't change slack

Propagation, Conflict, and Slack

Let ρ current assignment of solver (a.k.a. trail)
Represent as $\rho = \{$(ordered) set of literals assigned true$\}$

Slack measures how far ρ is from falsifying $\sum_i a_i \ell_i \geq A$

$$slack(\textstyle\sum_i a_i \ell_i \geq A; \rho) = \displaystyle\sum_{\ell_i \text{ not falsified by } \rho} a_i - A$$

Consider $C : x_1 + 2\overline{x}_2 + 3x_3 + 4\overline{x}_4 + 5x_5 \geq 7$

ρ	$slack(C;\rho)$	comment
$\{\}$	8	
$\{\overline{x}_5\}$	3	propagates $\overline{x}_4$ (coefficient > slack)
$\{\overline{x}_5, \overline{x}_4\}$	3	propagation doesn't change slack
$\{\overline{x}_5, \overline{x}_4, \overline{x}_3, x_2\}$	-2	conflict (slack < 0)

Propagation, Conflict, and Slack

Let ρ current assignment of solver (a.k.a. trail)
Represent as $\rho = \{$(ordered) set of literals assigned true$\}$

Slack measures how far ρ is from falsifying $\sum_i a_i \ell_i \geq A$

$$slack(\textstyle\sum_i a_i \ell_i \geq A; \rho) = \displaystyle\sum_{\ell_i \text{ not falsified by } \rho} a_i - A$$

Consider $C : x_1 + 2\overline{x}_2 + 3x_3 + 4\overline{x}_4 + 5x_5 \geq 7$

ρ	$slack(C; \rho)$	comment
$\{\}$	8	
$\{\overline{x}_5\}$	3	propagates $\overline{x}_4$ (coefficient $>$ slack)
$\{\overline{x}_5, \overline{x}_4\}$	3	propagation doesn't change slack
$\{\overline{x}_5, \overline{x}_4, \overline{x}_3, x_2\}$	-2	conflict (slack < 0)

Note that constraint can be conflicting though not all variables assigned

Conflict Analysis Invariant

Look at our example CDCL conflict analysis again

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$$\boxed{w \overset{\mathsf{d}}{=} 0}$$

$$\boxed{u \overset{\overline{u} \lor w}{=} 0}$$

$$\boxed{x \overset{\mathsf{d}}{=} 0}$$

$$\boxed{y \overset{u \lor x \lor y}{=} 1}$$

$$\boxed{z \overset{x \lor \overline{y} \lor z}{=} 1}$$

$$\boxed{\overset{\overline{y} \lor \overline{z}}{\bot}}$$

Conflict Analysis Invariant

Look at our example CDCL conflict analysis again

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$\boxed{w \stackrel{\mathsf{d}}{=} 0}$

$\boxed{u \stackrel{\overline{u} \lor w}{=} 0}$

$\boxed{x \stackrel{\mathsf{d}}{=} 0}$

$\boxed{y \stackrel{u \lor x \lor y}{=} 1}$

$\boxed{z \stackrel{x \lor \overline{y} \lor z}{=} 1}$

$\boxed{\stackrel{\overline{y} \lor \overline{z}}{\bot}}$

Assignment "left on trail" always falsifies derived clause

Conflict Analysis Invariant

Look at our example CDCL conflict analysis again

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$

$\boxed{w \stackrel{\mathsf{d}}{=} 0}$

$\boxed{u \stackrel{\overline{u} \lor w}{=} 0}$

$\boxed{x \stackrel{\mathsf{d}}{=} 0}$

$\boxed{y \stackrel{u \lor x \lor y}{=} 1}$

$\boxed{z \stackrel{x \lor \overline{y} \lor z}{=} 1}$

$\boxed{\stackrel{\overline{y} \lor \overline{z}}{\bot}}$

Assignment "left on trail" always falsifies derived clause

$\overline{y} \lor \overline{z}$ falsified by trail $\rho = \{\overline{w}, \overline{u}, \overline{x}, y, z\}$

Conflict Analysis Invariant

Look at our example CDCL conflict analysis again

$$(u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{u} \vee w) \wedge (\overline{u} \vee \overline{w})$$

$w \overset{\mathsf{d}}{=} 0$

$u \overset{\overline{u} \vee w}{=} 0$

$x \overset{\mathsf{d}}{=} 0$

$y \overset{u \vee x \vee y}{=} 1$

$z \overset{x \vee \overline{y} \vee z}{=} 1$

$\overset{\overline{y} \vee \overline{z}}{\bot}$

$x \vee \overline{y}$

Assignment "left on trail" always falsifies derived clause

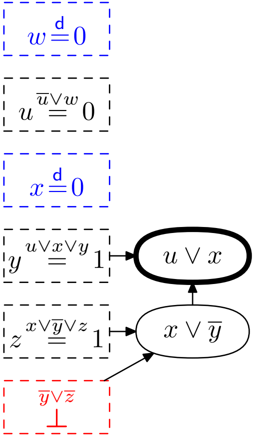
$x \vee \overline{y}$ falsified by trail $\rho' = \{\overline{w}, \overline{u}, \overline{x}, y\}$

$\overline{y} \vee \overline{z}$ falsified by trail $\rho = \{\overline{w}, \overline{u}, \overline{x}, y, z\}$

Conflict Analysis Invariant

Look at our example CDCL conflict analysis again

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



Assignment "left on trail" always falsifies derived clause

$u \lor x$ falsified by trail $\rho'' = \{\overline{w}, \overline{u}, \overline{x}\}$

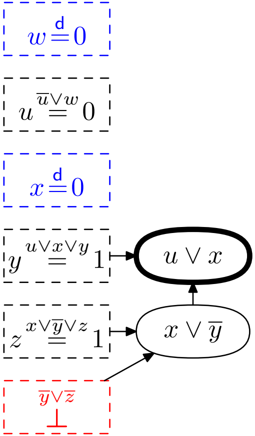
$x \lor \overline{y}$ falsified by trail $\rho' = \{\overline{w}, \overline{u}, \overline{x}, y\}$

$\overline{y} \lor \overline{z}$ falsified by trail $\rho = \{\overline{w}, \overline{u}, \overline{x}, y, z\}$

Conflict Analysis Invariant

Look at our example CDCL conflict analysis again

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



$u \lor x$ falsified by trail $\rho'' = \{\overline{w}, \overline{u}, \overline{x}\}$

$x \lor \overline{y}$ falsified by trail $\rho' = \{\overline{w}, \overline{u}, \overline{x}, y\}$

$\overline{y} \lor \overline{z}$ falsified by trail $\rho = \{\overline{w}, \overline{u}, \overline{x}, y, z\}$

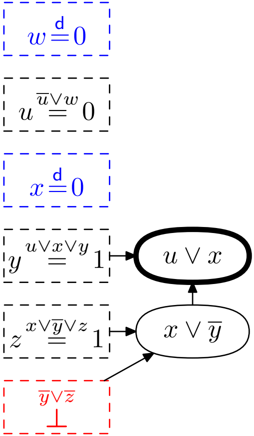
Assignment "left on trail" always falsifies derived clause

$\Rightarrow$ every derived constraint "explains" conflict

Conflict Analysis Invariant

Look at our example CDCL conflict analysis again

$$(u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{u} \vee w) \wedge (\overline{u} \vee \overline{w})$$



$u \vee x$ falsified by trail $\rho'' = \{\overline{w}, \overline{u}, \overline{x}\}$

$x \vee \overline{y}$ falsified by trail $\rho' = \{\overline{w}, \overline{u}, \overline{x}, y\}$

$\overline{y} \vee \overline{z}$ falsified by trail $\rho = \{\overline{w}, \overline{u}, \overline{x}, y, z\}$

Assignment "left on trail" always falsifies derived clause

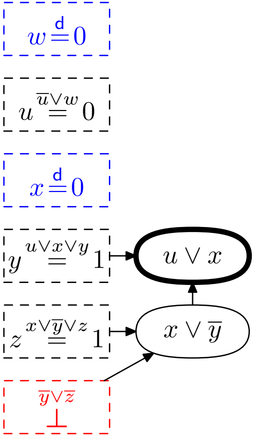
$\Rightarrow$ every derived constraint "explains" conflict

Terminate conflict analysis when explanation looks nice

Conflict Analysis Invariant

Look at our example CDCL conflict analysis again

$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



$u \lor x$ falsified by trail $\rho'' = \{\overline{w}, \overline{u}, \overline{x}\}$

$x \lor \overline{y}$ falsified by trail $\rho' = \{\overline{w}, \overline{u}, \overline{x}, y\}$

$\overline{y} \lor \overline{z}$ falsified by trail $\rho = \{\overline{w}, \overline{u}, \overline{x}, y, z\}$

Assignment "left on trail" always falsifies derived clause

$\Rightarrow$ every derived constraint "explains" conflict

Terminate conflict analysis when explanation looks nice

Learn asserting constraint: after backjump, some variable guaranteed to flip

Generalized Resolution

Can mimic resolution step

$$\frac{x \lor \overline{y} \lor z \qquad \overline{y} \lor \overline{z}}{x \lor \overline{y}}$$

Generalized Resolution

Can mimic resolution step

$$\frac{x \vee \overline{y} \vee z \qquad \overline{y} \vee \overline{z}}{x \vee \overline{y}}$$

by adding clauses as pseudo-Boolean constraints

$$\frac{x + \overline{y} + z \geq 1 \qquad \overline{y} + \overline{z} \geq 1}{x + 2\overline{y} \geq 1}$$

(Recall $z + \overline{z} = 1$)

Generalized Resolution

Can mimic resolution step

$$\frac{x \vee \overline{y} \vee z \qquad \overline{y} \vee \overline{z}}{x \vee \overline{y}}$$

by adding clauses as pseudo-Boolean constraints

$$\frac{x + \overline{y} + z \geq 1 \qquad \overline{y} + \overline{z} \geq 1}{x + 2\overline{y} \geq 1}$$

(Recall $z + \overline{z} = 1$)

Generalized resolution rule [Hoo88, Hoo92]

Positive linear combination so that some variable cancels

$$\frac{a_1 x_1 + \sum_{i \geq 2} a_i \ell_i \geq A \qquad b_1 \overline{x}_1 + \sum_{i \geq 2} b_i \ell_i \geq B}{\sum_{i \geq 2} (\frac{c}{a_1} a_i + \frac{c}{b_1} b_i) \ell_i \geq \frac{c}{a_1} A + \frac{c}{b_1} B - c} \; [c = \mathrm{lcm}(a_1, b_1)]$$

Saturation

Actually, don't get quite the right constraint in mimicking of resolution

$$\frac{x + \overline{y} + z \geq 1 \qquad \overline{y} + \overline{z} \geq 1}{x + 2\overline{y} \geq 1}$$

Saturation

Actually, don't get quite the right constraint in mimicking of resolution

$$\frac{x + \overline{y} + z \geq 1 \qquad \overline{y} + \overline{z} \geq 1}{x + 2\overline{y} \geq 1}$$

But clearly valid to conclude

$$\frac{x + 2\overline{y} \geq 1}{x + \overline{y} \geq 1}$$

Saturation

Actually, don't get quite the right constraint in mimicking of resolution

$$\frac{x + \overline{y} + z \geq 1 \qquad \overline{y} + \overline{z} \geq 1}{x + 2\overline{y} \geq 1}$$

But clearly valid to conclude

$$\frac{x + 2\overline{y} \geq 1}{x + \overline{y} \geq 1}$$

Saturation rule

$$\frac{\sum_i a_i \ell_i \geq A}{\sum_i \min\{a_i, A\} \cdot \ell_i \geq A}$$

Sound over integers, not over rationals (need such rules for SAT solving)

Analyze Conflict with Generalized Resolution + Saturation!

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Analyze Conflict with Generalized Resolution + Saturation!

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = (x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1)$ $\Rightarrow$ Conflict with C_2
(Note: same constraint can propagate several times!)

Analyze Conflict with Generalized Resolution + Saturation!

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = (x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1)$ $\Rightarrow$ Conflict with C_2
(Note: same constraint can propagate several times!)

- Resolve reason$(x_3, \rho) \doteq C_1$ with C_2 over x_3 to get resolve(C_1, C_2, x_3)

$$\frac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4 \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{x_4 \geq 1}$$

Analyze Conflict with Generalized Resolution + Saturation!

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = (x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1) \Rightarrow$ Conflict with C_2
(Note: same constraint can propagate several times!)

- Resolve reason$(x_3, \rho) \doteq C_1$ with C_2 over x_3 to get resolve(C_1, C_2, x_3)

$$\frac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4 \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{x_4 \geq 1}$$

- Applying saturate$(x_4 \geq 1)$ does nothing

Analyze Conflict with Generalized Resolution + Saturation!

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = (x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1) \Rightarrow$ Conflict with C_2

(Note: same constraint can propagate several times!)

- Resolve reason$(x_3, \rho) \doteq C_1$ with C_2 over x_3 to get resolve(C_1, C_2, x_3)

$$\frac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4 \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{x_4 \geq 1}$$

- Applying saturate$(x_4 \geq 1)$ does nothing
- Non-negative slack w.r.t. $\rho' = (x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1)$ — not conflicting!

Analyze Conflict with Generalized Resolution + Saturation!

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = (x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1)$ $\Rightarrow$ Conflict with C_2
(Note: same constraint can propagate several times!)

- Resolve reason$(x_3, \rho) \doteq C_1$ with C_2 over x_3 to get resolve(C_1, C_2, x_3)
$$\frac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4 \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{x_4 \geq 1}$$
- Applying saturate$(x_4 \geq 1)$ does nothing
- Non-negative slack w.r.t. $\rho' = (x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1)$ — not conflicting!

Fix (non-obvious): Apply weakening to reason constraints

$$\text{weaken}(\textstyle\sum_i a_i\ell_i \geq A, \ell_j) = \sum_{i \neq j} a_i\ell_i \geq A - a_j$$

Try to Reduce the Reason Constraint

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = (x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1)$ $\Rightarrow$ Conflict with C_2

Let's try to

1. Weaken reason on non-falsified literal (but not last propagated)
2. Saturate weakened constraint
3. Resolve with conflicting constraint over propagated literal

Try to Reduce the Reason Constraint

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = (x_1 \overset{\mathsf{d}}{=} 0, x_2 \overset{C_1}{=} 1, x_3 \overset{C_1}{=} 1)$ $\Rightarrow$ Conflict with C_2

Let's try to

1. Weaken reason on non-falsified literal (but not last propagated)
2. Saturate weakened constraint
3. Resolve with conflicting constraint over propagated literal

$$\text{resolve } x_3 \frac{\text{saturate} \dfrac{\text{weaken } x_2 \dfrac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4}{2x_1 + 2x_3 + x_4 \geq 2}}{2x_1 + 2x_3 + x_4 \geq 2} \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{2\overline{x}_2 + x_4 \geq 1}$$

Try to Reduce the Reason Constraint

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = (x_1 \overset{\mathsf{d}}{=} 0, x_2 \overset{C_1}{=} 1, x_3 \overset{C_1}{=} 1)$ $\Rightarrow$ Conflict with C_2

Let's try to

1. Weaken reason on non-falsified literal (but not last propagated)
2. Saturate weakened constraint
3. Resolve with conflicting constraint over propagated literal

$$\text{weaken } x_2 \frac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4}{\text{saturate } \dfrac{2x_1 + 2x_3 + x_4 \geq 2}{\text{resolve } x_3 \dfrac{2x_1 + 2x_3 + x_4 \geq 2 \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{2\overline{x}_2 + x_4 \geq 1}}}$$

Bummer! Still non-negative slack — not conflicting

Try Again to Reduce the Reason Constraint...

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = \left(x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1\right) \Rightarrow$ Conflict with C_2

Try Again to Reduce the Reason Constraint...

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = (x_1 \overset{\mathsf{d}}{=} 0, x_2 \overset{C_1}{=} 1, x_3 \overset{C_1}{=} 1)$ $\Rightarrow$ Conflict with C_2

$$\dfrac{\text{saturate}\ \dfrac{\text{weaken } \{x_2, x_4\}\ \dfrac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4}{2x_1 + 2x_3 \geq 1}}{x_1 + x_3 \geq 1} \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{2\overline{x}_2 \geq 1}\ \text{resolve } x_3$$

Try Again to Reduce the Reason Constraint...

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = (x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1)$ $\Rightarrow$ Conflict with C_2

$$\dfrac{\text{resolve } x_3 \dfrac{\text{saturate } \dfrac{\text{weaken } \{x_2, x_4\} \dfrac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4}{2x_1 + 2x_3 \geq 1}}{x_1 + x_3 \geq 1} \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{2\overline{x}_2 \geq 1}}{}$$

Negative slack — conflicting! Saturate and resolve with reason for x_2

Try Again to Reduce the Reason Constraint...

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = (x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1) \;\Rightarrow\;$ Conflict with C_2

$$\dfrac{\dfrac{\dfrac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4}{2x_1 + 2x_3 \geq 1}\text{ weaken } \{x_2, x_4\}}{x_1 + x_3 \geq 1}\text{ saturate} \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{2\overline{x}_2 \geq 1}\text{ resolve } x_3$$

Negative slack — conflicting! Saturate and resolve with reason for x_2

$$\dfrac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4 \qquad \dfrac{2\overline{x}_2 \geq 1}{\overline{x}_2 \geq 1}\text{ saturate}}{2x_1 + 2x_3 + x_4 \geq 4}\text{ resolve } x_2$$

Try Again to Reduce the Reason Constraint...

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = (x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1)$ $\Rightarrow$ Conflict with C_2

$$\text{resolve } x_3 \dfrac{\text{saturate} \dfrac{\text{weaken } \{x_2, x_4\} \dfrac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4}{2x_1 + 2x_3 \geq 1}}{x_1 + x_3 \geq 1} \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{2\overline{x}_2 \geq 1}$$

Negative slack — conflicting! Saturate and resolve with reason for x_2

$$\text{resolve } x_2 \dfrac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4 \qquad \dfrac{2\overline{x}_2 \geq 1}{\overline{x}_2 \geq 1} \text{ saturate}}{2x_1 + 2x_3 + x_4 \geq 4}$$

Asserting! Backjump propagates to conflict without decisions $\Rightarrow$ **done**

Reason Reduction Using Saturation [CK05]

$\mathrm{reduceSat}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

while slack(resolve$(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell); \rho) \geq 0$ **do**
  $\ell' \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ not falsified by ρ;
  $C_{\mathrm{reason}} \leftarrow$ saturate(weaken$(C_{\mathrm{reason}}, \ell')$);
end
return C_{reason};

Reason Reduction Using Saturation [CK05]

$\mathrm{reduceSat}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

while $\mathit{slack}(\mathsf{resolve}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell); \rho) \geq 0$ **do**
  $\ell' \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ not falsified by ρ;
  $C_{\mathrm{reason}} \leftarrow \mathsf{saturate}(\mathsf{weaken}(C_{\mathrm{reason}}, \ell'))$;
end
return C_{reason};

Why does this work?

- Slack is subadditive

$$\mathit{slack}(c \cdot C + d \cdot D; \rho) \leq c \cdot \mathit{slack}(C; \rho) + d \cdot \mathit{slack}(D; \rho)$$

Reason Reduction Using Saturation [CK05]

$\mathrm{reduceSat}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

while $\mathit{slack}(\mathsf{resolve}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell); \rho) \geq 0$ **do**
  $\ell' \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ not falsified by ρ;
  $C_{\mathrm{reason}} \leftarrow \mathsf{saturate}(\mathsf{weaken}(C_{\mathrm{reason}}, \ell'))$;
end
return C_{reason};

Why does this work?

- Slack is subadditive

$$\mathit{slack}(c \cdot C + d \cdot D; \rho) \leq c \cdot \mathit{slack}(C; \rho) + d \cdot \mathit{slack}(D; \rho)$$

- By invariant have $\mathit{slack}(C_{\mathrm{confl}}; \rho) < 0$

Reason Reduction Using Saturation [CK05]

$\mathrm{reduceSat}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

while $\mathit{slack}(\mathsf{resolve}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell); \rho) \geq 0$ **do**
    $\ell' \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ not falsified by ρ;
    $C_{\mathrm{reason}} \leftarrow \mathsf{saturate}(\mathsf{weaken}(C_{\mathrm{reason}}, \ell'))$;
end
return C_{reason};

Why does this work?

- Slack is subadditive

$$\mathit{slack}(c \cdot C + d \cdot D; \rho) \leq c \cdot \mathit{slack}(C; \rho) + d \cdot \mathit{slack}(D; \rho)$$

- By invariant have $\mathit{slack}(C_{\mathrm{confl}}; \rho) < 0$
- Weakening leaves $\mathit{slack}(C_{\mathrm{reason}}; \rho)$ unchanged

Reason Reduction Using Saturation [CK05]

$\mathrm{reduceSat}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

while $\mathit{slack}(\mathsf{resolve}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell); \rho) \geq 0$ **do**
  $\ell' \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ not falsified by ρ;
  $C_{\mathrm{reason}} \leftarrow \mathsf{saturate}(\mathsf{weaken}(C_{\mathrm{reason}}, \ell'))$;
end
return C_{reason};

Why does this work?

- Slack is subadditive

$$\mathit{slack}(c \cdot C + d \cdot D; \rho) \leq c \cdot \mathit{slack}(C; \rho) + d \cdot \mathit{slack}(D; \rho)$$

- By invariant have $\mathit{slack}(C_{\mathrm{confl}}; \rho) < 0$
- Weakening leaves $\mathit{slack}(C_{\mathrm{reason}}; \rho)$ unchanged
- Saturation decreases slack — reach 0 when max #literals weakened

Pseudo-Boolean Conflict Analys

analyzePBconflict(C_{confl}, ρ)

while C_{confl} *not asserting* **do**
  $\ell \leftarrow$ literal assigned last on trail ρ;
  if $\overline{\ell}$ *occurs in* C_{confl} **then**
    $C_{\mathrm{reason}} \leftarrow$ reason(ℓ, ρ);
    $\boldsymbol{C_{\mathrm{reason}} \leftarrow \mathbf{reduceSat}(C_{\mathrm{reason}}, C_{\mathrm{confl}}, \ell, \rho)}$;
    $C_{\mathrm{confl}} \leftarrow$ resolve$(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell)$;
    $C_{\mathrm{confl}} \leftarrow$ saturate(C_{confl});
  end
  $\rho \leftarrow \mathit{removeLast}(\rho)$;
end
return C_{confl};

The need to reduce the reason is new compared to CDCL
Everything else is the same

Some Problems Compared to CDCL

- Compared to clauses harder to detect propagation for constraints like

$$\sum_{i=1}^{n} x_i \geq n - 1$$

Some Problems Compared to CDCL

- Compared to clauses harder to detect propagation for constraints like

$$\sum_{i=1}^{n} x_i \geq n-1$$

- Generalized resolution for general pseudo-Boolean constraints
  $\Rightarrow$ lots of lcm computations
  $\Rightarrow$ coefficient sizes can explode (expensive arithmetic)

Some Problems Compared to CDCL

- Compared to clauses harder to detect propagation for constraints like

$$\sum_{i=1}^{n} x_i \geq n - 1$$

- Generalized resolution for general pseudo-Boolean constraints
  $\Rightarrow$ lots of lcm computations
  $\Rightarrow$ coefficient sizes can explode (expensive arithmetic)

- For CNF inputs, degenerates to resolution!
  $\Rightarrow$ CDCL but with super-expensive data structures

The Cutting Planes Proof System

Cutting planes as defined in [CCT87] doesn't use saturation but instead division (a.k.a. Chvátal-Gomory cut)

Literal axioms $$\frac{}{\ell_i \geq 0}$$

Linear combination $$\frac{\sum_i a_i \ell_i \geq A \qquad \sum_i b_i \ell_i \geq B}{\sum_i (c_A a_i + c_B b_i)\ell_i \geq c_A A + c_B B}$$

Division $$\frac{\sum_i a_i \ell_i \geq A}{\sum_i \lceil a_i / c \rceil \ell_i \geq \lceil A/c \rceil}$$

The Cutting Planes Proof System

Cutting planes as defined in [CCT87] doesn't use saturation but instead division (a.k.a. Chvátal-Gomory cut)

Literal axioms $$\frac{}{\ell_i \geq 0}$$

Linear combination $$\frac{\sum_i a_i \ell_i \geq A \qquad \sum_i b_i \ell_i \geq B}{\sum_i (c_A a_i + c_B b_i)\ell_i \geq c_A A + c_B B}$$

Division $$\frac{\sum_i a_i \ell_i \geq A}{\sum_i \lceil a_i / c \rceil \ell_i \geq \lceil A/c \rceil}$$

- Cutting planes with division implicationally complete
- Cutting planes with saturation is **not** [VEG+18]
- Can division yield stronger conflict analysis?

The Cutting Planes Proof System

Cutting planes as defined in [CCT87] doesn't use saturation but instead division (a.k.a. Chvátal-Gomory cut)

Literal axioms $$\frac{}{\ell_i \geq 0}$$

Linear combination $$\frac{\sum_i a_i \ell_i \geq A \qquad \sum_i b_i \ell_i \geq B}{\sum_i (c_A a_i + c_B b_i)\ell_i \geq c_A A + c_B B}$$

Division $$\frac{\sum_i a_i \ell_i \geq A}{\sum_i \lceil a_i / c \rceil \ell_i \geq \lceil A/c \rceil}$$

- Cutting planes with division implicationally complete
- Cutting planes with saturation is **not** [VEG⁺18]
- Can division yield stronger conflict analysis?
  (Used for general integer linear programming in *CutSat* [JdM13])

Using Division to Reduce the Reason

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = (x_1 \overset{\mathsf{d}}{=} 0, x_2 \overset{C_1}{=} 1, x_3 \overset{C_1}{=} 1)$ $\Rightarrow$ Conflict with C_2

Using Division to Reduce the Reason

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = (x_1 \overset{\mathsf{d}}{=} 0, x_2 \overset{C_1}{=} 1, x_3 \overset{C_1}{=} 1)$ $\Rightarrow$ Conflict with C_2

1. Weaken reason on non-falsified literal(s) with coefficient not divisible by propagating literal coefficient
2. Divide weakened constraint by propagating literal coefficient
3. Resolve with conflicting constraint over propagated literal

Using Division to Reduce the Reason

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = (x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1)$ $\Rightarrow$ Conflict with C_2

1. Weaken reason on non-falsified literal(s) with coefficient not divisible by propagating literal coefficient
2. Divide weakened constraint by propagating literal coefficient
3. Resolve with conflicting constraint over propagated literal

$$\dfrac{\text{resolve } x_3 \; \dfrac{\text{divide by } 2 \; \dfrac{\text{weaken } x_4 \; \dfrac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4}{2x_1 + 2x_2 + 2x_3 \geq 3}}{x_1 + x_2 + x_3 \geq 2} \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{0 \geq 1}$$

Using Division to Reduce the Reason

$$C_1 \doteq 2x_1 + 2x_2 + 2x_3 + x_4 \geq 4$$
$$C_2 \doteq 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3$$

Trail $\rho = (x_1 \stackrel{\mathsf{d}}{=} 0, x_2 \stackrel{C_1}{=} 1, x_3 \stackrel{C_1}{=} 1)$ $\Rightarrow$ Conflict with C_2

1. Weaken reason on non-falsified literal(s) with coefficient not divisible by propagating literal coefficient
2. Divide weakened constraint by propagating literal coefficient
3. Resolve with conflicting constraint over propagated literal

$$\dfrac{\text{divide by } 2\ \dfrac{\text{weaken } x_4\ \dfrac{2x_1 + 2x_2 + 2x_3 + x_4 \geq 4}{2x_1 + 2x_2 + 2x_3 \geq 3}}{x_1 + x_2 + x_3 \geq 2} \qquad 2\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3}{0 \geq 1}\ \text{resolve } x_3$$

Conflict immediately!

Reason Reduction Using Division [EN18]

$\mathrm{reduceDiv}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

$c \leftarrow \mathit{coeff}(C_{\mathrm{reason}}, \ell)$;
while $\mathit{slack}(\mathsf{resolve}(C_{\mathrm{confl}}, \mathsf{divide}(C_{\mathrm{reason}}, c), \ell); \rho) \geq 0$ **do**
  $\ell_j \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ such that $\overline{\ell}_j \notin \rho$ and $c \nmid \mathit{coeff}(C, \ell_j)$;
  $C_{\mathrm{reason}} \leftarrow \mathsf{weaken}(C_{\mathrm{reason}}, \ell_j)$;
end
return $\mathsf{divide}(C_{\mathrm{reason}}, c)$;

Reason Reduction Using Division [EN18]

$\mathrm{reduceDiv}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

$c \leftarrow \mathit{coeff}(C_{\mathrm{reason}}, \ell)$;
while $\mathit{slack}(\mathsf{resolve}(C_{\mathrm{confl}}, \mathsf{divide}(C_{\mathrm{reason}}, c), \ell); \rho) \geq 0$ **do**
  $\ell_j \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ such that $\overline{\ell}_j \notin \rho$ and $c \nmid \mathit{coeff}(C, \ell_j)$;
  $C_{\mathrm{reason}} \leftarrow \mathsf{weaken}(C_{\mathrm{reason}}, \ell_j)$;
end
return $\mathsf{divide}(C_{\mathrm{reason}}, c)$;

So now why does **this** work?

- Sufficient to get reason with slack 0 since
  1. $\mathit{slack}(C_{\mathrm{confl}}; \rho) < 0$
  2. slack is subadditive

Reason Reduction Using Division [EN18]

$\mathrm{reduceDiv}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

$c \leftarrow \mathit{coeff}(C_{\mathrm{reason}}, \ell)$;
while $\mathit{slack}(\mathsf{resolve}(C_{\mathrm{confl}}, \mathsf{divide}(C_{\mathrm{reason}}, c), \ell); \rho) \geq 0$ **do**
  $\ell_j \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ such that $\overline{\ell}_j \notin \rho$ and $c \nmid \mathit{coeff}(C, \ell_j)$;
  $C_{\mathrm{reason}} \leftarrow \mathsf{weaken}(C_{\mathrm{reason}}, \ell_j)$;
end
return $\mathsf{divide}(C_{\mathrm{reason}}, c)$;

So now why does **this** work?

- Sufficient to get reason with slack 0 since
  1. $\mathit{slack}(C_{\mathrm{confl}}; \rho) < 0$
  2. slack is subadditive
- Weakening doesn't change slack $\Rightarrow$ always $0 \leq \mathit{slack}(C_{\mathrm{reason}}; \rho) < c$

Reason Reduction Using Division [EN18]

$\mathrm{reduceDiv}(C_{\mathrm{confl}}, C_{\mathrm{reason}}, \ell, \rho)$

$c \leftarrow \mathit{coeff}(C_{\mathrm{reason}}, \ell)$;
while $\mathit{slack}(\text{resolve}(C_{\mathrm{confl}}, \text{divide}(C_{\mathrm{reason}}, c), \ell); \rho) \geq 0$ **do**
  $\ell_j \leftarrow$ literal in $C_{\mathrm{reason}} \setminus \{\ell\}$ such that $\overline{\ell}_j \notin \rho$ and $c \nmid \mathit{coeff}(C, \ell_j)$;
  $C_{\mathrm{reason}} \leftarrow \text{weaken}(C_{\mathrm{reason}}, \ell_j)$;
end
return divide(C_{reason}, c);

So now why does **this** work?

- Sufficient to get reason with slack 0 since
  1. $\mathit{slack}(C_{\mathrm{confl}}; \rho) < 0$
  2. slack is subadditive
- Weakening doesn't change slack $\Rightarrow$ always $0 \leq \mathit{slack}(C_{\mathrm{reason}}; \rho) < c$
- After max #weakenings have $0 \leq \mathit{slack}(\text{divide}(C_{\mathrm{reason}}, c); \rho) < 1$

Round-to-1 Reduction used in *RoundingSat*

Reduction method used in *RoundingSat* does max weakening right away

roundToOne(C, ℓ, ρ)

$c \leftarrow \mathit{coeff}(C, \ell)$;
foreach *literal* ℓ_j *in* C **do**
  if $\overline{\ell}_j \notin \rho$ *and* $c \nmid \mathit{coeff}(C, \ell_j)$ **then**
    $C \leftarrow$ weaken(C, ℓ_j);
  end
end
return divide(C, c);

And roundToOne used more aggressively in conflict analysis

RoundingSat Conflict Analysis

$\text{analyzePBconflict}(C_{\text{confl}}, \rho)$

while C_{confl} *contains no or multiple falsified literals on last level* **do**
  if *no current solver decisions* **then**
    output `UNSATISFIABLE` and terminate
  end
  $\ell \leftarrow$ literal assigned last on trail ρ;
  if $\overline{\ell}$ *occurs in* C_{confl} **then**
    $C_{\text{confl}} \leftarrow \text{roundToOne}(C_{\text{confl}}, \overline{\ell}, \rho)$;
    $C_{\text{reason}} \leftarrow \text{roundToOne}(\text{reason}(\ell, \rho), \ell, \rho)$;
    $C_{\text{confl}} \leftarrow \text{resolve}(C_{\text{confl}}, C_{\text{reason}}, \ell)$;
  end
  $\rho \leftarrow \mathit{removeLast}(\rho)$;
end
$\ell \leftarrow$ literal in C_{confl} last falsified by ρ;
return $\text{roundToOne}(C_{\text{confl}}, \ell, \rho)$;

Division vs. Saturation

- Higher conflict speed when PB reasoning doesn't help [EN18]
- Seems to perform better when PB reasoning crucial [EGNV18]
- Keeps coefficients small — can do fixed-precision integer arithmetic
- But still equally hard to detect propagation
- And still degenerates to resolution for CNF inputs

Other PB Rules I: Cardinality Constraint Reduction

Given PB constraint

$$3x_1 + 2x_2 + x_3 + x_4 \geq 4$$

can compute least #literals that have to be true

Other PB Rules I: Cardinality Constraint Reduction

Given PB constraint

$$3x_1 + 2x_2 + x_3 + x_4 \geq 4$$

can compute least #literals that have to be true

$$x_1 + x_2 + x_3 + x_4 \geq 2$$

Other PB Rules I: Cardinality Constraint Reduction

Given PB constraint

$$3x_1 + 2x_2 + x_3 + x_4 \geq 4$$

can compute least #literals that have to be true

$$x_1 + x_2 + x_3 + x_4 \geq 2$$

Galena [CK05] only learns cardinality constraints — easier to deal with

Other PB Rules I: Cardinality Constraint Reduction

Given PB constraint

$$3x_1 + 2x_2 + x_3 + x_4 \geq 4$$

can compute least #literals that have to be true

$$x_1 + x_2 + x_3 + x_4 \geq 2$$

Galena [CK05] only learns cardinality constraints — easier to deal with

---

Cardinality constraint reduction rule

$$\frac{\sum_i a_i \ell_i \geq A}{\sum_{i\,:\,a_i>0} \ell_i \geq T} \quad T = \min\{|I| \,:\, I \subseteq [n],\, \sum_{i \in I} a_i \geq A\}$$

Other PB Rules II: Strengthening

Strengthening by example:

- Set $x = 0$ and propagate on constraints

$$x + y \geq 1 \qquad x + z \geq 1 \qquad y + z \geq 1$$

Other PB Rules II: Strengthening

Strengthening by example:

- Set $x = 0$ and propagate on constraints

$$x + y \geq 1 \qquad x + z \geq 1 \qquad y + z \geq 1$$

- $y \overset{x+y\geq 1}{=} 1$ and $z \overset{x+z\geq 1}{=} 1 \Rightarrow y + z \geq 1$ oversatisfied by margin 1

Other PB Rules II: Strengthening

Strengthening by example:

- Set $x = 0$ and propagate on constraints

$$x + y \geq 1 \qquad x + z \geq 1 \qquad y + z \geq 1$$

- $y \overset{x+y \geq 1}{=} 1$ and $z \overset{x+z \geq 1}{=} 1 \Rightarrow y + z \geq 1$ oversatisfied by margin 1
- Hence, can deduce constraint $x + y + z \geq 2$

Other PB Rules II: Strengthening

Strengthening by example:

- Set $x = 0$ and propagate on constraints

$$x + y \geq 1 \qquad x + z \geq 1 \qquad y + z \geq 1$$

- $y \overset{x+y\geq 1}{=} 1$ and $z \overset{x+z\geq 1}{=} 1 \Rightarrow y + z \geq 1$ oversatisfied by margin 1
- Hence, can deduce constraint $x + y + z \geq 2$

Strengthening rule (imported by [DG02] from operations research)

- Suppose $\ell = 0 \Rightarrow \sum_i a_i \ell_i \geq A$ oversatisfied by amount K
- Then can deduce $K\ell + \sum_i a_i \ell_i \geq A + K$

Other PB Rules II: Strengthening

Strengthening by example:

- Set $x = 0$ and propagate on constraints

$$x + y \geq 1 \qquad x + z \geq 1 \qquad y + z \geq 1$$

- $y \overset{x+y\geq 1}{=} 1$ and $z \overset{x+z\geq 1}{=} 1 \Rightarrow y + z \geq 1$ oversatisfied by margin 1
- Hence, can deduce constraint $x + y + z \geq 2$

Strengthening rule (imported by [DG02] from operations research)

- Suppose $\ell = 0 \Rightarrow \sum_i a_i \ell_i \geq A$ oversatisfied by amount K
- Then can deduce $K\ell + \sum_i a_i \ell_i \geq A + K$

In theory, can recover from bad encodings (e.g., CNF)
In practice, seems inefficient and hard to get to work. . .

Other PB Rules III: "Fusion Resolution"

Suppose have constraints

$$2x + 3y + 2z + w \geq 3 \qquad 2\overline{x} + 3y + 2z + w \geq 3$$

Other PB Rules III: "Fusion Resolution"

Suppose have constraints

$$2x + 3y + 2z + w \geq 3 \qquad 2\overline{x} + 3y + 2z + w \geq 3$$

Then by eyeballing can conclude

$$3y + 2z + w \geq 3$$

Other PB Rules III: “Fusion Resolution”

Suppose have constraints

$$2x + 3y + 2z + w \geq 3 \qquad 2\overline{x} + 3y + 2z + w \geq 3$$

Then by eyeballing can conclude

$$3y + 2z + w \geq 3$$

But only get from resolution

$$6y + 4z + 2w \geq 4$$

Other PB Rules III: “Fusion Resolution”

Suppose have constraints

$$2x + 3y + 2z + w \geq 3 \qquad 2\overline{x} + 3y + 2z + w \geq 3$$

Then by eyeballing can conclude

$$3y + 2z + w \geq 3$$

But only get from resolution + saturation

$$4y + 4z + 2w \geq 4$$

Other PB Rules III: “Fusion Resolution”

Suppose have constraints

$$2x + 3y + 2z + w \geq 3 \qquad 2\overline{x} + 3y + 2z + w \geq 3$$

Then by eyeballing can conclude

$$3y + 2z + w \geq 3$$

But only get from resolution + saturation + division

$$2y + 2z + \ \ w \geq 2$$

Other PB Rules III: "Fusion Resolution"

Suppose have constraints

$$2x + 3y + 2z + w \geq 3 \qquad 2\overline{x} + 3y + 2z + w \geq 3$$

Then by eyeballing can conclude

$$3y + 2z + w \geq 3$$

But only get from resolution + saturation + division

$$2y + 2z + \ w \geq 2$$

"Fusion resolution" [Goc17]

$$\frac{a\ell + \sum_i b_i \ell_i \geq B \qquad a\overline{\ell} + \sum_i b_i \ell_i \geq B'}{\sum_i b_i \ell_i \geq \min\{B, B'\}}$$

No obvious way for cutting planes to immediately derive this

Shows up in some tricky benchmarks in [EGNV18]

Open Problems I: Some Implementation Challenges

1. Degrees of freedom in PB conflict analysis
   - Skip resolution steps when slack very negative?
   - How much to weaken?
   - Learn general PB constraints or more limited form?
2. Efficient propagation detection for PB constraints
3. Assessment of quality of learned constraints
4. Distance to backjump? (Constraint can be asserting at several levels)

Open Problems II: Some PB Reasoning Challenges

1. Better conflict analysis (also for CDCL)
   Is trivial resolution optimal, or can it pay to be smarter?
2. Natural way to recover from bad encodings (e.g., CNF)
3. Efficient and concise PB proof logging
4. Theoretical potential and limitations poorly understood [VEG$^+$18]
   - Separations of subsystems of cutting planes?
   - In particular, is division strictly stronger than saturation?

   Hear more from Marc Vinyals this afternoon

Open Problems III: Beyond PB Reasoning

- Sometimes very poor performance even on LPs that are rationally infeasible! (And trivial for mixed integer linear programming solvers)
- But sometimes MIP solvers lost when learning from PB constraints crucial (and when conflict-driven PB solvers shine)
- Borrow techniques from (or merge with) MIP?
  Stay tuned for Ambros Gleixner's talk...

Summing up

- Conflict-driven search hugely successful SAT solving paradigm
- This talk: Survey how to port from CDCL to PB constraints
- Potential exponential performance gains haven't materialized so far
- Instead highly nontrivial challenges regarding
  - Efficient implementation
  - Theoretical understanding
- But no obvious reason why efficient PB solvers should not be possible (remember CDCL took 50 years)
- And in any case lots of fun questions to work on! ☺

Summing up

- Conflict-driven search hugely successful SAT solving paradigm
- This talk: Survey how to port from CDCL to PB constraints
- Potential exponential performance gains haven't materialized so far
- Instead highly nontrivial challenges regarding
  - Efficient implementation
  - Theoretical understanding
- But no obvious reason why efficient PB solvers should not be possible (remember CDCL took 50 years)
- And in any case lots of fun questions to work on! ☺

Thank you for your attention!

References I

[BS97] Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pages 203–208, July 1997.

[BW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, March 2001. Preliminary version in *STOC '99*.

[CCT87] William Cook, Collette Rene Coullard, and György Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, November 1987.

[CK05] Donald Chai and Andreas Kuehlmann. A fast pseudo-Boolean constraint solver. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(3):305–317, March 2005. Preliminary version in *DAC '03*.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC '71)*, pages 151–158, 1971.

[DG02] Heidi E. Dixon and Matthew L. Ginsberg. Inference methods for a pseudo-Boolean satisfiability solver. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI '02)*, pages 635–640, July 2002.

References II

[Dix04] Heidi E. Dixon. *Automating Psuedo-Boolean Inference within a DPLL Framework*. PhD thesis, University of Oregon, 2004. Available at `http://www.cirl.uoregon.edu/dixon/papers/dixonDissertation.pdf`.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, July 1962.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.

[EGNV18] Jan Elffers, Jesús Giráldez-Cru, Jakob Nordström, and Marc Vinyals. Using combinatorial benchmarks to probe the reasoning power of pseudo-Boolean solvers. In *Proceedings of the 21st International Conference on Theory and Applications of Satisfiability Testing (SAT '18)*, volume 10929 of *Lecture Notes in Computer Science*, pages 75–93. Springer, July 2018.

[EN18] Jan Elffers and Jakob Nordström. Divide and conquer: Towards faster pseudo-Boolean solving. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI '18)*, pages 1291–1299, July 2018.

[ES06] Niklas Eén and Niklas Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1-4):1–26, 2006.

References III

[Goc17] Stephan Gocht. Personal communication, 2017.

[Hak85] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39(2-3):297–308, August 1985.

[Hoo88] John N. Hooker. Generalized resolution and cutting planes. *Annals of Operations Research*, 12(1):217–239, 1988.

[Hoo92] John N. Hooker. Generalized resolution for 0-1 linear inequalities. *Annals of Mathematics and Artificial Intelligence*, 6(1):271–286, 1992.

[JdM13] Dejan Jovanovic and Leonardo de Moura. Cutting to the chase solving linear integer arithmetic. *Journal of Automated Reasoning*, 51(1):79–108, June 2013. Preliminary version in *CADE-23* 2011.

[Lev73] Leonid A. Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973. In Russian. Available at `http://mi.mathnet.ru/ppi914`.

[LP10] Daniel Le Berre and Anne Parrain. The Sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, 7:59–64, 2010.

References IV

[Mil00] The Millennium Problems of the Clay Mathematics Institute, May 2000. See `http://www.claymath.org/millennium-problems`.

[MML14] Ruben Martins, Vasco M. Manquinho, and Inês Lynce. Open-WBO: A modular MaxSAT solver. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14)*, volume 8561 of *Lecture Notes in Computer Science*, pages 438–445. Springer, July 2014.

[MMZ+01] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (DAC '01)*, pages 530–535, June 2001.

[MN14] Mladen Mikša and Jakob Nordström. Long proofs of (seemingly) simple formulas. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14)*, volume 8561 of *Lecture Notes in Computer Science*, pages 121–137. Springer, July 2014.

[MS96] João P. Marques-Silva and Karem A. Sakallah. GRASP—a new search algorithm for satisfiability. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '96)*, pages 220–227, November 1996.

References V

[RM09] Olivier Roussel and Vasco M. Manquinho. Pseudo-Boolean and cardinality constraints. In Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 22, pages 695–733. IOS Press, February 2009.

[SN15] Masahiko Sakai and Hidetomo Nabeshima. Construction of an ROBDD for a PB-constraint in band form and related techniques for PB-solvers. *IEICE Transactions on Information and Systems*, 98-D(6):1121–1127, June 2015.

[SS06] Hossein M. Sheini and Karem A. Sakallah. Pueblo: A hybrid pseudo-Boolean SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1-4):165–189, March 2006. Preliminary version in *DATE '05*.

[Urq87] Alasdair Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1):209–219, January 1987.

[VEG+18] Marc Vinyals, Jan Elffers, Jesús Giráldez-Cru, Stephan Gocht, and Jakob Nordström. In between resolution and cutting planes: A study of proof systems for pseudo-Boolean SAT solving. In *Proceedings of the 21st International Conference on Theory and Applications of Satisfiability Testing (SAT '18)*, volume 10929 of *Lecture Notes in Computer Science*, pages 292–310. Springer, July 2018.