Search in SAT

Marijn Heule

Delft University of Technology, The Netherlands
Johannes Kepler University, Austria

May 23, 2011 @ CPAIOR

The Satisfiability (SAT) problem

$$
\begin{array}{llll}
(x_5 \vee x_8 \vee \bar{x}_2) \wedge & (x_2 \vee \bar{x}_1 \vee \bar{x}_3) \wedge & (\bar{x}_8 \vee \bar{x}_3 \vee \bar{x}_7) \wedge & (\bar{x}_5 \vee x_3 \vee x_8) \wedge \\
(\bar{x}_6 \vee \bar{x}_1 \vee \bar{x}_5) \wedge & (x_8 \vee \bar{x}_9 \vee x_3) \wedge & (x_2 \vee x_1 \vee x_3) \wedge & (\bar{x}_1 \vee x_8 \vee x_4) \wedge \\
(\bar{x}_9 \vee \bar{x}_6 \vee x_8) \wedge & (x_8 \vee x_3 \vee \bar{x}_9) \wedge & (x_9 \vee \bar{x}_3 \vee x_8) \wedge & (x_6 \vee \bar{x}_9 \vee x_5) \wedge \\
(x_2 \vee \bar{x}_3 \vee \bar{x}_8) \wedge & (x_8 \vee \bar{x}_6 \vee \bar{x}_3) \wedge & (x_8 \vee \bar{x}_3 \vee \bar{x}_1) \wedge & (\bar{x}_8 \vee x_6 \vee \bar{x}_2) \wedge \\
(x_7 \vee x_9 \vee \bar{x}_2) \wedge & (x_8 \vee \bar{x}_9 \vee x_2) \wedge & (\bar{x}_1 \vee \bar{x}_9 \vee x_4) \wedge & (x_8 \vee x_1 \vee \bar{x}_2) \wedge \\
(x_3 \vee \bar{x}_4 \vee \bar{x}_6) \wedge & (\bar{x}_1 \vee \bar{x}_7 \vee x_5) \wedge & (\bar{x}_7 \vee x_1 \vee x_6) \wedge & (\bar{x}_5 \vee x_4 \vee \bar{x}_6) \wedge \\
(\bar{x}_4 \vee x_9 \vee \bar{x}_8) \wedge & (x_2 \vee x_9 \vee x_1) \wedge & (x_5 \vee \bar{x}_7 \vee x_1) \wedge & (\bar{x}_7 \vee \bar{x}_9 \vee \bar{x}_6) \wedge \\
(x_2 \vee x_5 \vee x_4) \wedge & (x_8 \vee \bar{x}_4 \vee x_5) \wedge & (x_5 \vee x_9 \vee x_3) \wedge & (\bar{x}_5 \vee \bar{x}_7 \vee x_9) \wedge \\
(x_2 \vee \bar{x}_8 \vee x_1) \wedge & (\bar{x}_7 \vee x_1 \vee x_5) \wedge & (x_1 \vee x_4 \vee x_3) \wedge & (x_1 \vee \bar{x}_9 \vee \bar{x}_4) \wedge \\
(x_3 \vee x_5 \vee x_6) \wedge & (\bar{x}_6 \vee x_3 \vee \bar{x}_9) \wedge & (\bar{x}_7 \vee x_5 \vee x_9) \wedge & (x_7 \vee \bar{x}_5 \vee \bar{x}_2) \wedge \\
(x_4 \vee x_7 \vee x_3) \wedge & (x_4 \vee \bar{x}_9 \vee \bar{x}_7) \wedge & (x_5 \vee \bar{x}_1 \vee x_7) \wedge & (x_5 \vee \bar{x}_1 \vee x_7) \wedge \\
(x_6 \vee x_7 \vee \bar{x}_3) \wedge & (\bar{x}_8 \vee \bar{x}_6 \vee \bar{x}_7) \wedge & (x_6 \vee x_2 \vee x_3) \wedge & (\bar{x}_8 \vee x_2 \vee x_5)
\end{array}
$$

Does there exist an assignment satisfying all clauses?

Search for a satisfying assignment (or proof none exists)

$$
\begin{array}{l}
(x_5 \vee x_8 \vee \bar{x}_2) \wedge (x_2 \vee \bar{x}_1 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_3 \vee \bar{x}_7) \wedge (\bar{x}_5 \vee x_3 \vee x_8) \wedge \\
(\bar{x}_6 \vee \bar{x}_1 \vee \bar{x}_5) \wedge (x_8 \vee \bar{x}_9 \vee x_3) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\bar{x}_1 \vee x_8 \vee x_4) \wedge \\
(\bar{x}_9 \vee \bar{x}_6 \vee x_8) \wedge (x_8 \vee x_3 \vee \bar{x}_9) \wedge (x_9 \vee \bar{x}_3 \vee x_8) \wedge (x_6 \vee \bar{x}_9 \vee x_5) \wedge \\
(x_2 \vee \bar{x}_3 \vee \bar{x}_8) \wedge (x_8 \vee \bar{x}_6 \vee \bar{x}_3) \wedge (x_8 \vee \bar{x}_3 \vee \bar{x}_1) \wedge (\bar{x}_8 \vee x_6 \vee \bar{x}_2) \wedge \\
(x_7 \vee x_9 \vee \bar{x}_2) \wedge (x_8 \vee \bar{x}_9 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_9 \vee x_4) \wedge (x_8 \vee x_1 \vee \bar{x}_2) \wedge \\
(x_3 \vee \bar{x}_4 \vee \bar{x}_6) \wedge (\bar{x}_1 \vee \bar{x}_7 \vee x_5) \wedge (\bar{x}_7 \vee x_1 \vee x_6) \wedge (\bar{x}_5 \vee x_4 \vee \bar{x}_6) \wedge \\
(\bar{x}_4 \vee x_9 \vee \bar{x}_8) \wedge (x_2 \vee x_9 \vee x_1) \wedge (x_5 \vee \bar{x}_7 \vee x_1) \wedge (\bar{x}_7 \vee \bar{x}_9 \vee \bar{x}_6) \wedge \\
(x_2 \vee x_5 \vee x_4) \wedge (x_8 \vee \bar{x}_4 \vee x_5) \wedge (x_5 \vee x_9 \vee x_3) \wedge (\bar{x}_5 \vee \bar{x}_7 \vee x_9) \wedge \\
(x_2 \vee \bar{x}_8 \vee x_1) \wedge (\bar{x}_7 \vee x_1 \vee x_5) \wedge (x_1 \vee x_4 \vee x_3) \wedge (x_1 \vee \bar{x}_9 \vee \bar{x}_4) \wedge \\
(x_3 \vee x_5 \vee x_6) \wedge (\bar{x}_6 \vee x_3 \vee \bar{x}_9) \wedge (\bar{x}_7 \vee x_5 \vee x_9) \wedge (x_7 \vee \bar{x}_5 \vee \bar{x}_2) \wedge \\
(x_4 \vee x_7 \vee x_3) \wedge (x_4 \vee \bar{x}_9 \vee \bar{x}_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge \\
(x_6 \vee x_7 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_6 \vee \bar{x}_7) \wedge (x_6 \vee x_2 \vee x_3) \wedge (\bar{x}_8 \vee x_2 \vee x_5)
\end{array}
$$

Play the SAT game:

http://www.cril.univ-artois.fr/~roussel/satgame/satgame.php

Motivation

From 100 variables, 200 constraints (early 90s)
to 1,000,000 vars. and 20,000,000 cls. in 20 years.

Applications:
Hardware and Software Verification, Planning, Scheduling, Optimal Control, Protocol Design, Routing, Combinatorial problems, Equivalence Checking, etc.

SAT used to solve many other problems!

Motivation

From 100 variables, 200 constraints (early 90s)
to 1,000,000 vars. and 20,000,000 cls. in 20 years.

Applications:
Hardware and Software Verification, Planning, Scheduling, Optimal Control, Protocol Design, Routing, Combinatorial problems, Equivalence Checking, etc.

SAT used to solve many other problems!

Motivation

From 100 variables, 200 constraints (early 90s)
to 1,000,000 vars. and 20,000,000 cls. in 20 years.

Applications:
Hardware and Software Verification, Planning, Scheduling, Optimal Control, Protocol Design, Routing, Combinatorial problems, Equivalence Checking, etc.

SAT used to solve many other problems!

Overview

Search for Lemmas

Depth-first search

- Learning Lemmas
- Data-structures
- Heuristics

Search for Simplification

Breadth-first search

- Variable elimination
- Blocked clause elimination
- Unhiding redundancy

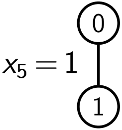
Conflict-driven SAT solvers: Search and Analysis

$(x_1 \vee x_4) \wedge$
$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$
$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge$
$\mathcal{F}_{\text{extra}}$

0

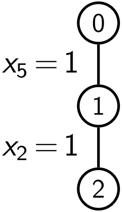
Conflict-driven SAT solvers: Search and Analysis

$$(x_1 \vee x_4) \wedge$$
$$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$$
$$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge$$
$$\mathcal{F}_{\text{extra}}$$



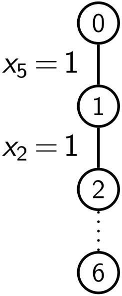
Conflict-driven SAT solvers: Search and Analysis

$(x_1 \vee x_4) \wedge$
$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$
$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge$
$\mathcal{F}_{\text{extra}}$



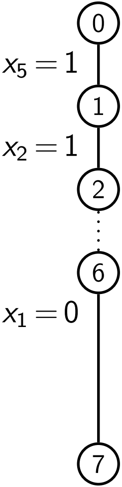
Conflict-driven SAT solvers: Search and Analysis

$$
\begin{aligned}
&(x_1 \vee x_4) \wedge \\
&(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge \\
&(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge \\
&\mathcal{F}_{\text{extra}}
\end{aligned}
$$



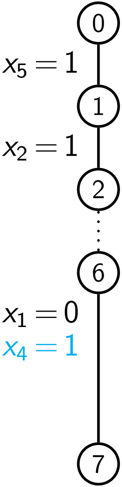
Conflict-driven SAT solvers: Search and Analysis

$(x_1 \lor x_4) \land$
$(x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land$
$(\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land$
$\mathcal{F}_{\text{extra}}$



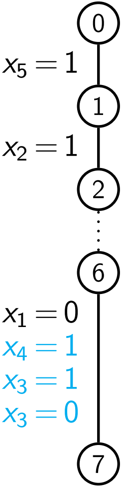
Conflict-driven SAT solvers: Search and Analysis

$(x_1 \vee x_4) \wedge$
$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$
$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge$
$\mathcal{F}_{\text{extra}}$



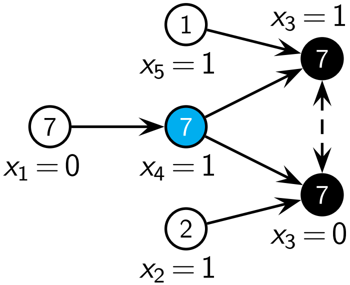
Conflict-driven SAT solvers: Search and Analysis

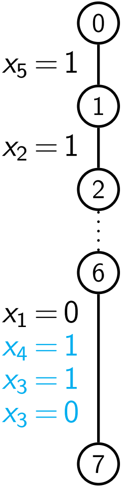
$$
\begin{aligned}
&(x_1 \lor x_4) \land \\
&(x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land \\
&(\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land \\
&\mathcal{F}_{\text{extra}}
\end{aligned}
$$



Conflict-driven SAT solvers: Search and Analysis

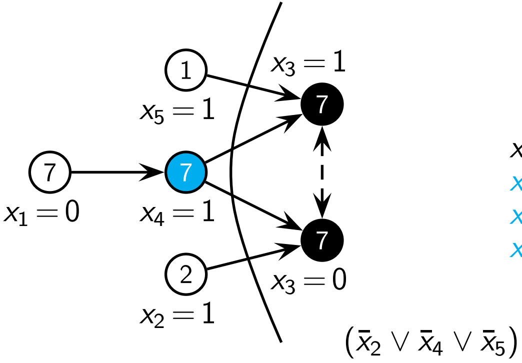
$(x_1 \lor x_4) \land$
$(x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land$
$(\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land$
$\mathcal{F}_{\text{extra}}$

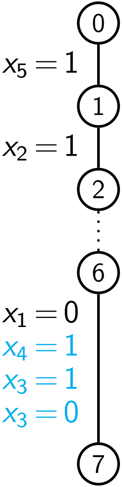




Conflict-driven SAT solvers: Search and Analysis

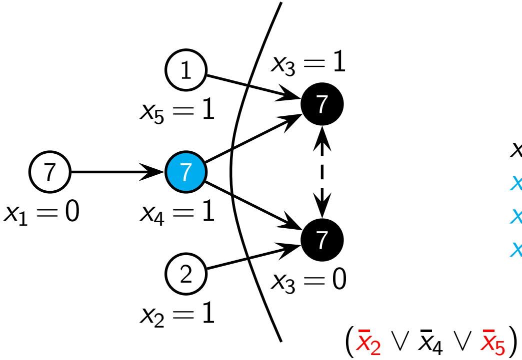
$(x_1 \vee x_4) \wedge$
$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$
$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge$
$\mathcal{F}_{\text{extra}}$

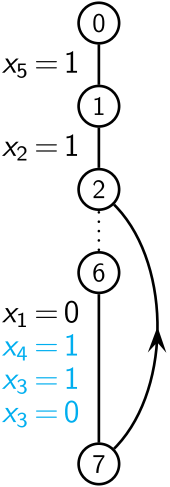




Conflict-driven SAT solvers: Search and Analysis

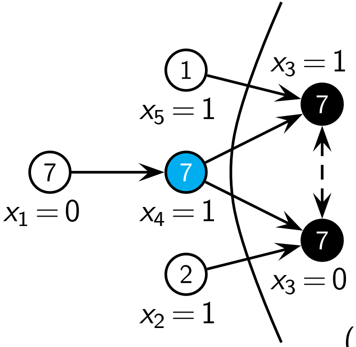
$(x_1 \vee x_4) \wedge$
$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$
$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge$
$\mathcal{F}_{\text{extra}}$

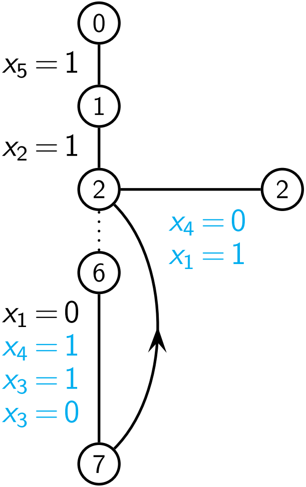




Conflict-driven SAT solvers: Search and Analysis

$$(x_1 \vee x_4) \wedge$$
$$(x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$$
$$(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_4) \wedge$$
$$\mathcal{F}_{\text{extra}}$$

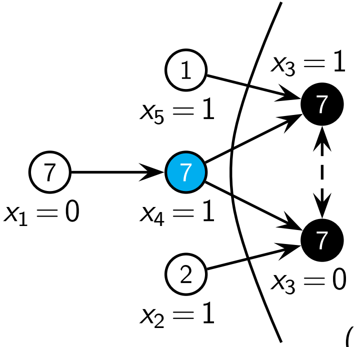


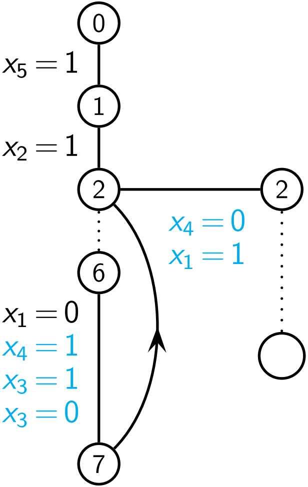


$$(\bar{x}_2 \vee \bar{x}_4 \vee \bar{x}_5)$$

Conflict-driven SAT solvers: Search and Analysis

$$(x_1 \lor x_4) \land$$
$$(x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land$$
$$(\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land$$
$$\mathcal{F}_{\text{extra}}$$





$$(\bar{x}_2 \lor \bar{x}_4 \lor \bar{x}_5)$$

Conflict-driven SAT solvers: Pseudo-code

while TRUE **do**

    $l_{\text{decision}} := \textsc{GetDecisionLiteral}(\ )$

    If no l_{decision} **then return** `satisfiable`

    $\mathcal{F} := \textsc{Simplify}(\ \mathcal{F}(l_{\text{decision}} \leftarrow 1)\ )$

    while $\mathcal{F}$ contains C_{conflict} **do**

        $C_{\text{conflict}} := \textsc{AnalyzeConflicts}(\ )$

        If $C_{\text{conflict}} \neq \emptyset$ **then return** `unsatisfiable`

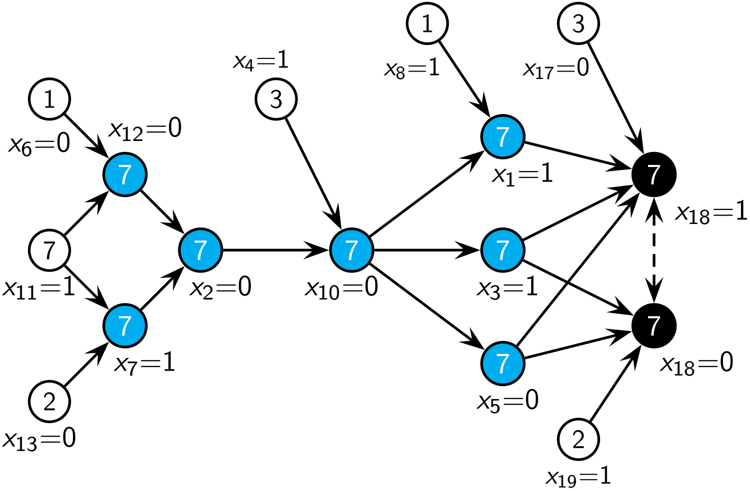
        $\textsc{BackTrack}(\ C_{\text{conflict}}\ )$

        $\mathcal{F} := \textsc{Simplify}(\ \mathcal{F} \cup C_{\text{conflict}}\ )$

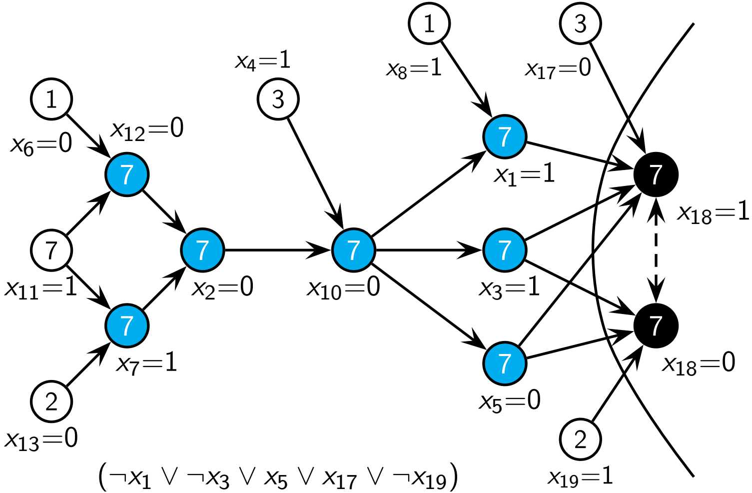
    end while

end while

Learning conflict clauses (lemma's)



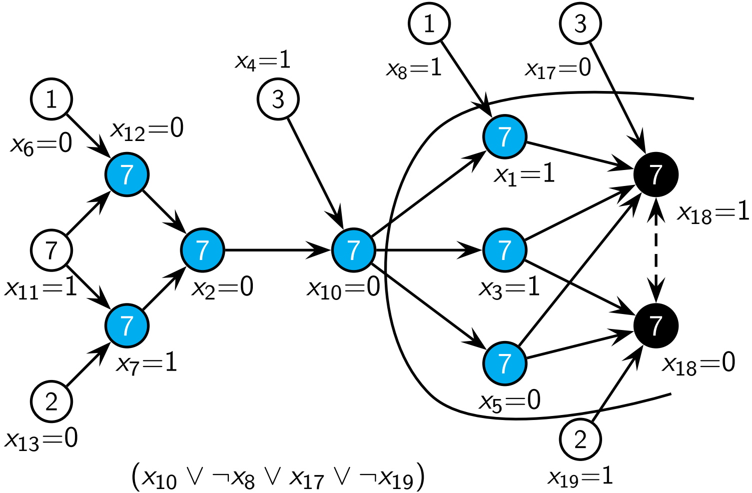
Learning conflict clauses (lemma's)



$(\neg x_1 \vee \neg x_3 \vee x_5 \vee x_{17} \vee \neg x_{19})$

tri-asserting clause

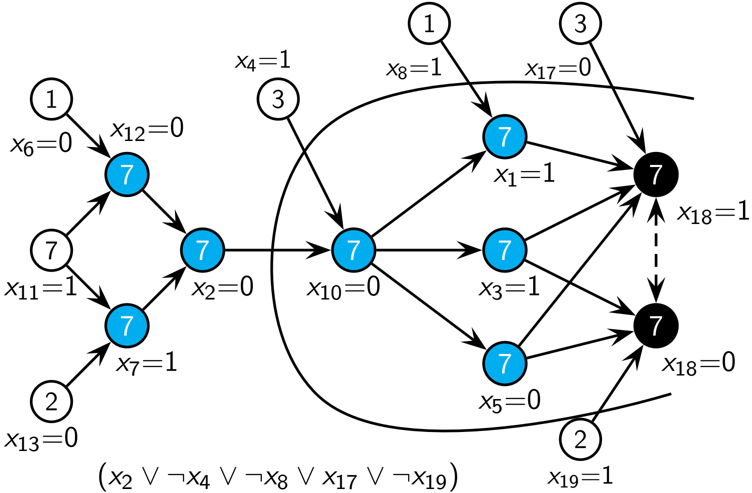
Learning conflict clauses (lemma's)



$(x_{10} \vee \neg x_8 \vee x_{17} \vee \neg x_{19})$

first unique implication point

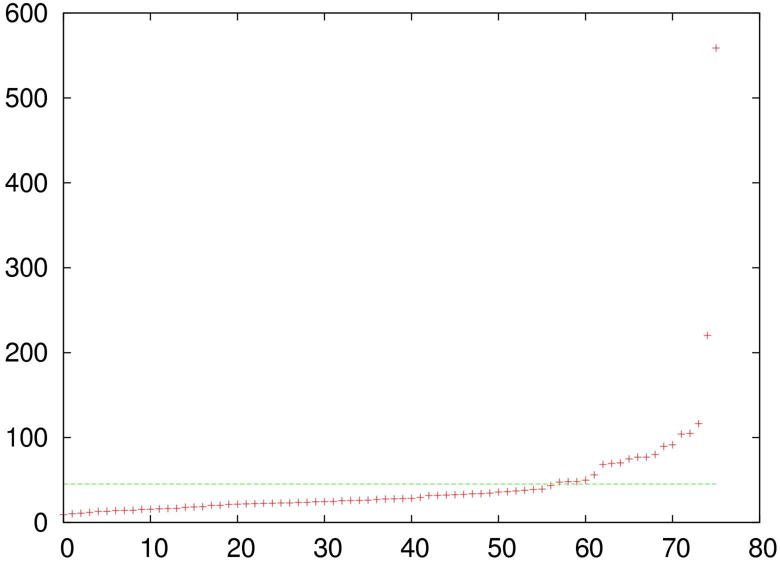
Learning conflict clauses (lemma's)



$(x_2 \vee \neg x_4 \vee \neg x_8 \vee x_{17} \vee \neg x_{19})$

second unique implication point

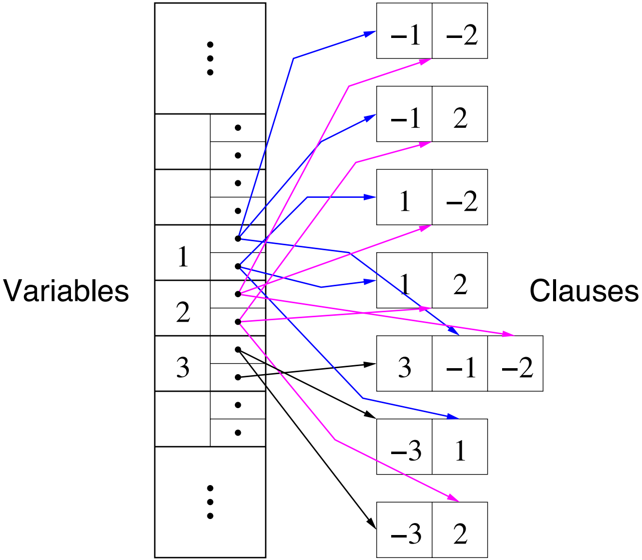
Average Learned Clause Length



Data-structures

Watch pointers

Simple data structure for unit propagation



Conflict-driven: Watch pointers (1)

$$\varphi = \{x_1 = *, x_2 = *, x_3 = *, x_4 = *, x_5 = *, x_6 = *\}$$

$\neg x_1$	x_2	$\neg x_3$	$\neg x_5$	x_6

x_1	$\neg x_3$	x_4	$\neg x_5$	$\neg x_6$

Conflict-driven: Watch pointers (1)

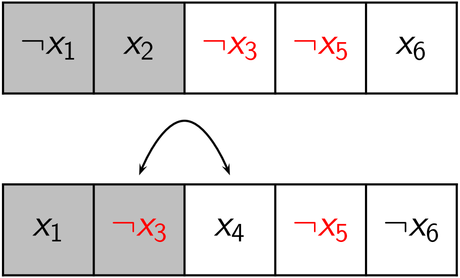
$$\varphi = \{x_1 = *, x_2 = *, x_3 = *, x_4 = *, x_5 = \mathbf{1}, x_6 = *\}$$

$\neg x_1$	x_2	$\neg x_3$	$\neg x_5$	x_6

x_1	$\neg x_3$	x_4	$\neg x_5$	$\neg x_6$

Conflict-driven: Watch pointers (1)

$$\varphi = \{x_1 = *, x_2 = *, x_3 = \mathbf{1}, x_4 = *, x_5 = 1, x_6 = *\}$$



Conflict-driven: Watch pointers (1)

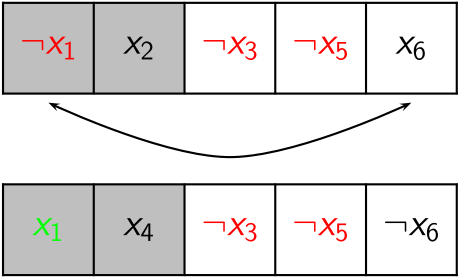
$$\varphi = \{x_1 = *, x_2 = *, x_3 = 1, x_4 = *, x_5 = 1, x_6 = *\}$$

$\neg x_1$	x_2	$\neg x_3$	$\neg x_5$	x_6

x_1	x_4	$\neg x_3$	$\neg x_5$	$\neg x_6$

Conflict-driven: Watch pointers (1)

$$\varphi = \{x_1 = \mathbf{1}, x_2 = *, x_3 = 1, x_4 = *, x_5 = 1, x_6 = *\}$$



Conflict-driven: Watch pointers (1)

$$\varphi = \{x_1 = 1, x_2 = *, x_3 = 1, x_4 = *, x_5 = 1, x_6 = *\}$$

x_6	x_2	$\neg x_3$	$\neg x_5$	$\neg x_1$

x_1	x_4	$\neg x_3$	$\neg x_5$	$\neg x_6$

Conflict-driven: Watch pointers (1)

$$\varphi = \{x_1 = 1, x_2 = *, x_3 = 1, x_4 = \mathbf{0}, x_5 = 1, x_6 = *\}$$

x_6	x_2	$\neg x_3$	$\neg x_5$	$\neg x_1$

x_1	x_4	$\neg x_3$	$\neg x_5$	$\neg x_6$

Conflict-driven: Watch pointers (1)

$$\varphi = \{x_1 = 1, x_2 = \mathbf{0}, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = *\}$$

x_6	x_2	$\neg x_3$	$\neg x_5$	$\neg x_1$

x_1	x_4	$\neg x_3$	$\neg x_5$	$\neg x_6$

Conflict-driven: Watch pointers (1)

$$\varphi = \{x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = \mathbf{1}\}$$

x_6	x_2	$\neg x_3$	$\neg x_5$	$\neg x_1$

x_1	x_4	$\neg x_3$	$\neg x_5$	$\neg x_6$

Conflict-driven: Watch pointers (1)

$$\varphi = \{x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 1, x_6 = 1\}$$

x_6	x_2	$\neg x_3$	$\neg x_5$	$\neg x_1$

x_1	x_4	$\neg x_3$	$\neg x_5$	$\neg x_6$

Conflict-driven: Watch pointers (2)

Only examine (get in the cache) a clause when both

- a watch pointer gets falsified
- the other one is not satisfied

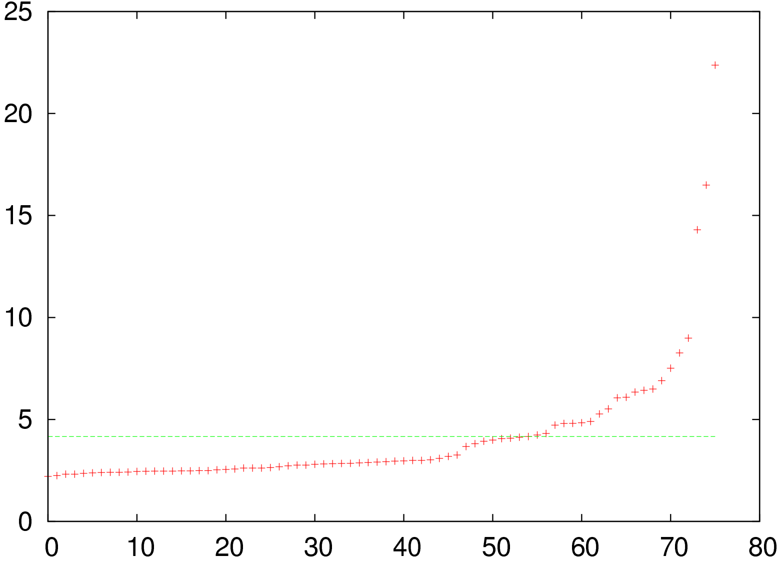
While backjumping, just unassign variables

Conflict clauses $\rightarrow$ watch pointers

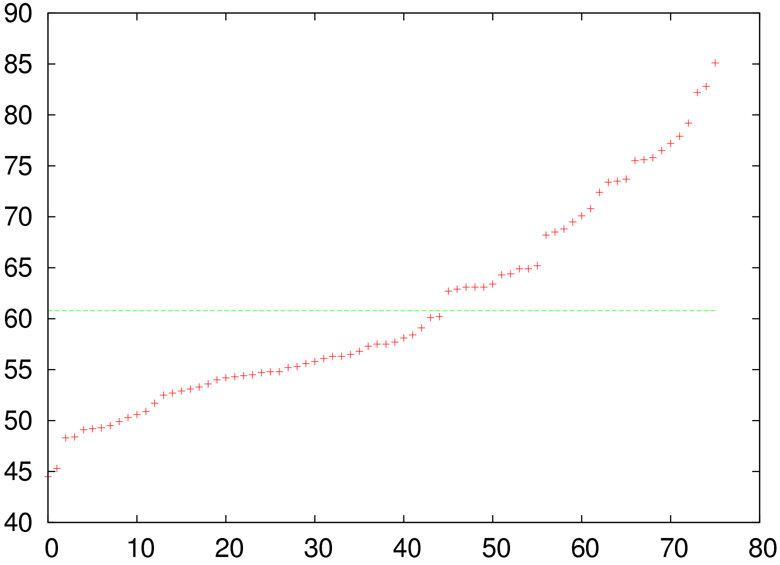
No detailed information available

Not used for binary clauses

Average Number Clauses Visited Per Propagation



Percentage visited clauses with other watched literal true



Heuristics

Most important CDCL heuristics

Variable selection heuristics

- aim: minimize the search space
- plus: could compensate a bad value selection

Value selection heuristics

- aim: guide search towards a solution (or conflict)
- plus: could compensate a bad variable selection, cache solutions of subproblems [PipatsrisawatDarwiche'07]

Restart strategies

- aim: avoid heavy-tail behavior [GomesSelmanCrato'97]
- plus: focus search on recent conflicts when combined with dynamic heuristics

Most important CDCL heuristics

Variable selection heuristics

- aim: minimize the search space
- plus: could compensate a bad value selection

Value selection heuristics

- aim: guide search towards a solution (or conflict)
- plus: could compensate a bad variable selection, cache solutions of subproblems [PipatsrisawatDarwiche'07]

Restart strategies

- aim: avoid heavy-tail behavior [GomesSelmanCrato'97]
- plus: focus search on recent conflicts when combined with dynamic heuristics

Most important CDCL heuristics

Variable selection heuristics

- aim: minimize the search space
- plus: could compensate a bad value selection

Value selection heuristics

- aim: guide search towards a solution (or conflict)
- plus: could compensate a bad variable selection, cache solutions of subproblems [PipatsrisawatDarwiche'07]

Restart strategies

- aim: avoid heavy-tail behavior [GomesSelmanCrato'97]
- plus: focus search on recent conflicts when combined with dynamic heuristics

Variable selection heuristics

Based on the occurrences in the (reduced) formula

- examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads
- not practical for CDCL solver due to watch pointers

Variable State Independent Decaying Sum (VSIDS)

- original idea (zChaff): for each conflict, increase the score of involved variables by 1, half all scores each 256 conflicts [MoskewiczMZZM2001]
- improvement (MiniSAT): for each conflict, increase the score of involved variables by δ and increase $\delta := 1.05\delta$ [EenSörensson2003]

Variable selection heuristics

Based on the occurrences in the (reduced) formula

- examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads
- not practical for CDCL solver due to watch pointers

Variable State Independent Decaying Sum (VSIDS)

- original idea (zChaff): for each conflict, increase the score of involved variables by 1, half all scores each 256 conflicts [MoskewiczMZZM2001]
- improvement (MiniSAT): for each conflict, increase the score of involved variables by δ and increase $\delta := 1.05\delta$ [EenSörensson2003]

Visualization of VSIDS in PicoSAT

http://www.youtube.com/watch?v=MOjhFywLre8

Value selection heuristics

Based on the occurrences in the (reduced) formula

- examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads
- not practical for CDCL solver due to watch pointers

Based on the encoding / consequently

- negative branching (early MiniSAT) [EenSörensson2003]

Based on the last implied value (phase-saving)

- introduced to CDCL [PipatsrisawatDarwiche2007]
- already used in local search [HirschKojevnikov2001]

Value selection heuristics

Based on the occurrences in the (reduced) formula

- examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads
- not practical for CDCL solver due to watch pointers

Based on the encoding / consequently

- negative branching (early MiniSAT) [EenSörensson2003]

Based on the last implied value (phase-saving)

- introduced to CDCL [PipatsrisawatDarwiche2007]
- already used in local search [HirschKojevnikov2001]

Value selection heuristics

Based on the occurrences in the (reduced) formula

- examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads
- not practical for CDCL solver due to watch pointers

Based on the encoding / consequently

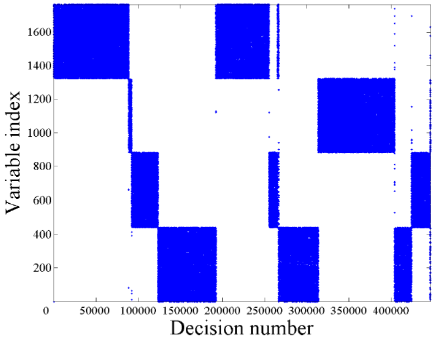
- negative branching (early MiniSAT) [EenSörensson2003]

Based on the last implied value (phase-saving)

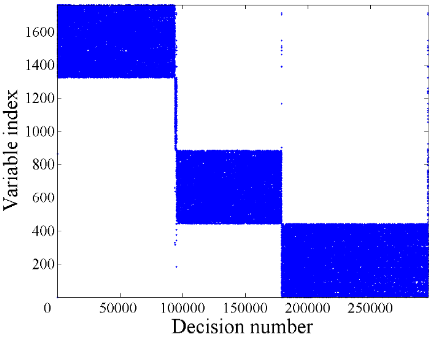
- introduced to CDCL [PipatsrisawatDarwiche2007]
- already used in local search [HirschKojevnikov2001]

Heuristics: Phase-saving

Selecting the last implied value remembers solved components



negative branching



phase-saving

Restarts

Restarts in CDCL solvers:

- Counter heavy-tail behavior [GomesSelmanCrato'97]
- Unassign all variables but keep the (dynamic) heuristics

Restart strategies: [Walsh'99, LubySinclairZuckerman'93]

- Geometrical restart: e.g. 100, 150, 225, 333, 500, 750, ...
- Luby sequence: e.g. 100, 100, 200, 100, 100, 200, 400, ...

Rapid restarts by reusing trail: [vanderTakHeuleRamos'11]

- Partial restart same effect as full restart
- Optimal strategy Luby-1: 1, 1, 2, 1, 1, 2, 4, ...

Restarts

Restarts in CDCL solvers:

- Counter heavy-tail behavior [GomesSelmanCrato'97]
- Unassign all variables but keep the (dynamic) heuristics

Restart strategies: [Walsh'99, LubySinclairZuckerman'93]

- Geometrical restart: e.g. $100, 150, 225, 333, 500, 750, \ldots$
- Luby sequence: e.g. $100, 100, 200, 100, 100, 200, 400, \ldots$

Rapid restarts by reusing trail: [vanderTakHeuleRamos'11]

- Partial restart same effect as full restart
- Optimal strategy Luby-1: $1, 1, 2, 1, 1, 2, 4, \ldots$

Restarts

Restarts in CDCL solvers:

- Counter heavy-tail behavior [GomesSelmanCrato'97]
- Unassign all variables but keep the (dynamic) heuristics

Restart strategies: [Walsh'99, LubySinclairZuckerman'93]

- Geometrical restart: e.g. $100, 150, 225, 333, 500, 750, \ldots$
- Luby sequence: e.g. $100, 100, 200, 100, 100, 200, 400, \ldots$

Rapid restarts by reusing trail: [vanderTakHeuleRamos'11]

- Partial restart same effect as full restart
- Optimal strategy Luby-1: $1, 1, 2, 1, 1, 2, 4, \ldots$

Variable Elimination

Variable Elimination [DavisPutnam'60]

Definition (Resolution)

Given two clauses $C = (x \vee a_1 \vee \cdots \vee a_i)$ and $D = (\bar{x} \vee b_1 \vee \cdots \vee b_j)$, the *resolvent* of C and D on variable x (denoted by $C \otimes_x D$) is $(a_1 \vee \cdots \vee a_i \vee b_1 \vee \cdots \vee b_j)$

Resolution on sets of clauses F_x and $F_{\bar{x}}$ (denoted by $F_x \otimes_x F_{\bar{x}}$) generates all (non-tautological) resolvents of $C \in F_x$ and $D \in F_{\bar{x}}$.

Definition (Variable elimination (VE))

Given a CNF formula F, *variable elimination* (or DP resolution) removes a variable x by replacing F_x and $F_{\bar{x}}$ by $F_x \otimes_x F_{\bar{x}}$

Proof procedure [DavisPutnam60]

VE is a complete proof procedure. Applying VE until fixpoint results in the empty formula (satisfiable) or empty clause (unsatisfiable)

Variable Elimination [DavisPutnam'60]

Definition (Resolution)

Given two clauses $C = (x \vee a_1 \vee \cdots \vee a_i)$ and $D = (\bar{x} \vee b_1 \vee \cdots \vee b_j)$, the *resolvent* of C and D on variable x (denoted by $C \otimes_x D$) is $(a_1 \vee \cdots \vee a_i \vee b_1 \vee \cdots \vee b_j)$

Resolution on sets of clauses F_x and $F_{\bar{x}}$ (denoted by $F_x \otimes_x F_{\bar{x}}$) generates all (non-tautological) resolvents of $C \in F_x$ and $D \in F_{\bar{x}}$.

Definition (Variable elimination (VE))

Given a CNF formula F, *variable elimination* (or DP resolution) removes a variable x by replacing F_x and $F_{\bar{x}}$ by $F_x \otimes_x F_{\bar{x}}$

Proof procedure [DavisPutnam60]

VE is a complete proof procedure. Applying VE until fixpoint results in the empty formula (satisfiable) or empty clause (unsatisfiable)

Variable Elimination [DavisPutnam'60]

Definition (Resolution)

Given two clauses $C = (x \vee a_1 \vee \dots \vee a_i)$ and $D = (\bar{x} \vee b_1 \vee \dots \vee b_j)$, the *resolvent* of C and D on variable x (denoted by $C \otimes_x D$) is $(a_1 \vee \dots \vee a_i \vee b_1 \vee \dots \vee b_j)$

Resolution on sets of clauses F_x and $F_{\bar{x}}$ (denoted by $F_x \otimes_x F_{\bar{x}}$) generates all (non-tautological) resolvents of $C \in F_x$ and $D \in F_{\bar{x}}$.

Definition (Variable elimination (VE))

Given a CNF formula F, *variable elimination* (or DP resolution) removes a variable x by replacing F_x and $F_{\bar{x}}$ by $F_x \otimes_x F_{\bar{x}}$

Proof procedure [DavisPutnam60]

VE is a complete proof procedure. Applying VE until fixpoint results in the empty formula (satisfiable) or empty clause (unsatisfiable)

Example VE by clause distribution [DavisPutnam'60]

Definition (Variable elimination (VE))

Given a CNF formula F, *variable elimination* (or DP resolution) removes a variable x by replacing F_x and $F_{\bar{x}}$ by $F_x \otimes_x F_{\bar{x}}$

Example of clause distribution

	F_x: $(x \lor c)$	F_x: $(x \lor \bar{d})$	F_x: $(x \lor \bar{a} \lor \bar{b})$
$F_{\bar{x}}$: $(\bar{x} \lor a)$	$(a \lor c)$	$(a \lor d)$	$(a \lor \bar{a} \lor \bar{b})$
$F_{\bar{x}}$: $(\bar{x} \lor b)$	$(b \lor c)$	$(b \lor d)$	$(b \lor \bar{a} \lor \bar{b})$
$F_{\bar{x}}$: $(\bar{x} \lor \bar{e} \lor f)$	$(c \lor \bar{e} \lor f)$	$(d \lor \bar{e} \lor f)$	$(\bar{a} \lor \bar{b} \lor \bar{e} \lor f)$

example: $|F_x \otimes F_{\bar{x}}| > |F_x| + |F_{\bar{x}}|$; in general: exponential growth of clauses

Example VE by clause distribution [DavisPutnam'60]

Definition (Variable elimination (VE))

Given a CNF formula F, *variable elimination* (or DP resolution) removes a variable x by replacing F_x and $F_{\bar{x}}$ by $F_x \otimes_x F_{\bar{x}}$

Example of clause distribution

$$
\begin{array}{rl|ccc}
 & & \multicolumn{3}{c}{\overbrace{\hspace{14em}}^{F_x}} \\
 & & (x \vee c) & (x \vee \bar{d}) & (x \vee \bar{a} \vee \bar{b}) \\
\hline
\multirow{3}{*}{$F_{\bar{x}}$} & (\bar{x} \vee a) & (a \vee c) & (a \vee d) & (a \vee \bar{a} \vee \bar{b}) \\
 & (\bar{x} \vee b) & (b \vee c) & (b \vee d) & (b \vee \bar{a} \vee \bar{b}) \\
 & (\bar{x} \vee \bar{e} \vee f) & (c \vee \bar{e} \vee f) & (d \vee \bar{e} \vee f) & (\bar{a} \vee \bar{b} \vee \bar{e} \vee f)
\end{array}
$$

example: $|F_x \otimes F_{\bar{x}}| > |F_x| + |F_{\bar{x}}|$; in general: exponential growth of clauses

Example VE by clause distribution [DavisPutnam'60]

Definition (Variable elimination (VE))

Given a CNF formula F, *variable elimination* (or DP resolution) removes a variable x by replacing F_x and $F_{\bar{x}}$ by $F_x \otimes_x F_{\bar{x}}$

Example of clause distribution

		F_x	
	$(x \vee c)$	$(x \vee \bar{d})$	$(x \vee \bar{a} \vee \bar{b})$
$F_{\bar{x}}$: $(\bar{x} \vee a)$	$(a \vee c)$	$(a \vee d)$	$\cancel{(a \vee \bar{a} \vee \bar{b})}$
$(\bar{x} \vee b)$	$(b \vee c)$	$(b \vee d)$	$\cancel{(b \vee \bar{a} \vee \bar{b})}$
$(\bar{x} \vee \bar{e} \vee f)$	$(c \vee \bar{e} \vee f)$	$(d \vee \bar{e} \vee f)$	$(\bar{a} \vee \bar{b} \vee \bar{e} \vee f)$

example: $|F_x \otimes F_{\bar{x}}| > |F_x| + |F_{\bar{x}}|$; in general: exponential growth of clauses

Example VE by clause distribution [DavisPutnam'60]

Definition (Variable elimination (VE))

Given a CNF formula F, *variable elimination* (or DP resolution) removes a variable x by replacing F_x and $F_{\bar{x}}$ by $F_x \otimes_x F_{\bar{x}}$

Example of clause distribution

	$(x \vee c)$	$(x \vee \bar{d})$	$(x \vee \bar{a} \vee \bar{b})$
$(\bar{x} \vee a)$	$(a \vee c)$	$(a \vee d)$	$\cancel{(a \vee \bar{a} \vee \bar{b})}$
$(\bar{x} \vee b)$	$(b \vee c)$	$(b \vee d)$	$\cancel{(b \vee \bar{a} \vee \bar{b})}$
$(\bar{x} \vee \bar{e} \vee f)$	$(c \vee \bar{e} \vee f)$	$(d \vee \bar{e} \vee f)$	$(\bar{a} \vee \bar{b} \vee \bar{e} \vee f)$

(Columns: F_x; rows: $F_{\bar{x}}$)

example: $|F_x \otimes F_{\bar{x}}| > |F_x| + |F_{\bar{x}}|$; in general: exponential growth of clauses

VE by substitution [EenBiere07]

General idea

Detect gates (or definitions) $x = \mathrm{GATE}(a_1, \ldots, a_n)$ in the formula and use them to reduce the number of added clauses

Possible gates

gate	G_x	$G_{\bar{x}}$
$\mathrm{AND}(a_1, \ldots, a_n)$	$(x \vee \bar{a}_1 \vee \cdots \vee \bar{a}_n)$	$(\bar{x} \vee a_1), \ldots, (\bar{x} \vee a_n)$
$\mathrm{OR}(a_1, \ldots, a_n)$	$(x \vee \bar{a}_1), \ldots, (x \vee \bar{a}_n)$	$(\bar{x} \vee a_1 \vee \cdots \vee a_n)$
$\mathrm{ITE}(c, t, f)$	$(x \vee \bar{c} \vee \bar{t}), (x \vee c \vee \bar{f})$	$(\bar{x} \vee \bar{c} \vee t), (\bar{x} \vee c \vee f)$

Variable elimination by substitution [EenBiere07]

Let $R_x = F_x \setminus G_x$; $R_{\bar{x}} = F_{\bar{x}} \setminus G_{\bar{x}}$.

Replace $F_x \wedge F_{\bar{x}}$ by $G_x \otimes_x R_{\bar{x}} \wedge G_{\bar{x}} \otimes_x R_x$.

VE by substitution [EenBiere07]

General idea

Detect gates (or definitions) $x = \mathrm{GATE}(a_1, \ldots, a_n)$ in the formula and use them to reduce the number of added clauses

Possible gates

gate	G_x	$G_{\bar{x}}$
$\mathrm{AND}(a_1, \ldots, a_n)$	$(x \vee \bar{a}_1 \vee \cdots \vee \bar{a}_n)$	$(\bar{x} \vee a_1), \ldots, (\bar{x} \vee a_n)$
$\mathrm{OR}(a_1, \ldots, a_n)$	$(x \vee \bar{a}_1), \ldots, (x \vee \bar{a}_n)$	$(\bar{x} \vee a_1 \vee \cdots \vee a_n)$
$\mathrm{ITE}(c, t, f)$	$(x \vee \bar{c} \vee \bar{t}), (x \vee c \vee \bar{f})$	$(\bar{x} \vee \bar{c} \vee t), (\bar{x} \vee c \vee f)$

Variable elimination by substitution [EenBiere07]

Let $R_x = F_x \setminus G_x$; $R_{\bar{x}} = F_{\bar{x}} \setminus G_{\bar{x}}$.

Replace $F_x \wedge F_{\bar{x}}$ by $G_x \otimes_x R_{\bar{x}} \wedge G_{\bar{x}} \otimes_x R_x$.

VE by substitution [EenBiere07]

General idea

Detect gates (or definitions) $x = \mathrm{GATE}(a_1, \dots, a_n)$ in the formula and use them to reduce the number of added clauses

Possible gates

gate	G_x	$G_{\bar{x}}$
$\mathrm{AND}(a_1, \dots, a_n)$	$(x \vee \bar{a}_1 \vee \dots \vee \bar{a}_n)$	$(\bar{x} \vee a_1), \dots, (\bar{x} \vee a_n)$
$\mathrm{OR}(a_1, \dots, a_n)$	$(x \vee \bar{a}_1), \dots, (x \vee \bar{a}_n)$	$(\bar{x} \vee a_1 \vee \dots \vee a_n)$
$\mathrm{ITE}(c, t, f)$	$(x \vee \bar{c} \vee \bar{t}), (x \vee c \vee \bar{f})$	$(\bar{x} \vee \bar{c} \vee t), (\bar{x} \vee c \vee f)$

Variable elimination by substitution [EenBiere07]

Let $R_x = F_x \setminus G_x$; $R_{\bar{x}} = F_{\bar{x}} \setminus G_{\bar{x}}$.

Replace $F_x \wedge F_{\bar{x}}$ by $G_x \otimes_x R_{\bar{x}} \wedge G_{\bar{x}} \otimes_x R_x$.

VE by substitution [EenBiere'07]

Example of gate extraction: $x = \mathrm{AND}(a, b)$

$$F_x = (x \lor c) \land (x \lor \bar{d}) \land (x \lor \bar{a} \lor \bar{b})$$
$$F_{\bar{x}} = (\bar{x} \lor a) \land (\bar{x} \lor b) \land (\bar{x} \lor \bar{e} \lor f)$$

Example of substitution

		R_x		G_x
		$(x \lor c)$	$(x \lor \bar{d})$	$(x \lor \bar{a} \lor \bar{b})$
$G_{\bar{x}}$	$(\bar{x} \lor a)$	$(a \lor c)$	$(a \lor d)$	
	$(\bar{x} \lor b)$	$(b \lor c)$	$(b \lor d)$	
$R_{\bar{x}}$	$(\bar{x} \lor \bar{e} \lor f)$			$(\bar{a} \lor \bar{b} \lor \bar{e} \lor f)$

using substitution: $|F_x \otimes F_{\bar{x}}| < |F_x| + |F_{\bar{x}}|$

VE by substitution [EenBiere'07]

Example of gate extraction: $x = \text{AND}(a, b)$

$$F_x = (x \vee c) \wedge (x \vee \bar{d}) \wedge (x \vee \bar{a} \vee \bar{b})$$
$$F_{\bar{x}} = (\bar{x} \vee a) \wedge (\bar{x} \vee b) \wedge (\bar{x} \vee \bar{e} \vee f)$$

Example of substitution

		R_x: $(x \vee c)$	R_x: $(x \vee \bar{d})$	G_x: $(x \vee \bar{a} \vee \bar{b})$
$G_{\bar{x}}$	$(\bar{x} \vee a)$	$(a \vee c)$	$(a \vee d)$	
$G_{\bar{x}}$	$(\bar{x} \vee b)$	$(b \vee c)$	$(b \vee d)$	
$R_{\bar{x}}$	$(\bar{x} \vee \bar{e} \vee f)$			$(\bar{a} \vee \bar{b} \vee \bar{e} \vee f)$

using substitution: $|F_x \otimes F_{\bar{x}}| < |F_x| + |F_{\bar{x}}|$

VE by substitution [EenBiere'07]

Example of gate extraction: $x = \mathrm{AND}(a, b)$

$$F_x = (x \vee c) \wedge (x \vee \bar{d}) \wedge \color{red}{(x \vee \bar{a} \vee \bar{b})}$$
$$F_{\bar{x}} = \color{red}{(\bar{x} \vee a)} \wedge \color{red}{(\bar{x} \vee b)} \wedge (\bar{x} \vee \bar{e} \vee f)$$

Example of substitution

		R_x		G_x
		$(x \vee c)$	$(x \vee \bar{d})$	$(x \vee \bar{a} \vee \bar{b})$
$G_{\bar{x}}$	$(\bar{x} \vee a)$	$(a \vee c)$	$(a \vee d)$	
	$(\bar{x} \vee b)$	$(b \vee c)$	$(b \vee d)$	
$R_{\bar{x}}$	$(\bar{x} \vee \bar{e} \vee f)$			$(\bar{a} \vee \bar{b} \vee \bar{e} \vee f)$

using substitution: $|F_x \otimes F_{\bar{x}}| < |F_x| + |F_{\bar{x}}|$

Blocked Clause Elimination

Blocked Clauses [Kullmann'99]

Definition (Blocking literal)

A literal l in a clause C of a CNF F blocks C w.r.t. F if for every clause $C' \in F$ with $\bar{l} \in C'$, the resolvent $(C \setminus \{l\}) \cup (C' \setminus \{\bar{l}\})$ obtained from resolving C and C' on l is a tautology.

With respect to a fixed CNF and its clauses we have:

Definition (Blocked clause)

A clause is blocked if it contains a literal that blocks it.

Example

Consider the formula $(a \vee b) \wedge (a \vee \bar{b} \vee \bar{c}) \wedge (\bar{a} \vee c)$.
First clause is not blocked.
Second clause is blocked by both a and $\bar{c}$. Third clause is blocked by c

Proposition

Removal of an arbitrary blocked clause preserves satisfiability.

Blocked Clauses [Kullmann'99]

Definition (Blocking literal)

A literal l in a clause C of a CNF F blocks C w.r.t. F if for every clause $C' \in F$ with $\bar{l} \in C'$, the resolvent $(C \setminus \{l\}) \cup (C' \setminus \{\bar{l}\})$ obtained from resolving C and C' on l is a tautology.

With respect to a fixed CNF and its clauses we have:

Definition (Blocked clause)

A clause is blocked if it contains a literal that blocks it.

Example

Consider the formula $(a \vee b) \wedge (a \vee \bar{b} \vee \bar{c}) \wedge (\bar{a} \vee c)$.
First clause is not blocked.
Second clause is blocked by both a and $\bar{c}$. Third clause is blocked by c

Proposition

Removal of an arbitrary blocked clause preserves satisfiability.

Blocked Clauses [Kullmann'99]

Definition (Blocking literal)

A literal l in a clause C of a CNF F blocks C w.r.t. F if for every clause $C' \in F$ with $\bar{l} \in C'$, the resolvent $(C \setminus \{l\}) \cup (C' \setminus \{\bar{l}\})$ obtained from resolving C and C' on l is a tautology.

With respect to a fixed CNF and its clauses we have:

Definition (Blocked clause)

A clause is blocked if it contains a literal that blocks it.

Example

Consider the formula $(a \vee b) \wedge (a \vee \bar{b} \vee \bar{c}) \wedge (\bar{a} \vee c)$.
First clause is not blocked.
Second clause is blocked by both a and $\bar{c}$. Third clause is blocked by c

Proposition

Removal of an arbitrary blocked clause preserves satisfiability.

Blocked Clause Elimination (BCE)

Definition (BCE)

While there is a blocked clause C in a CNF F, remove C from F.

Example

Consider $(a \vee b) \wedge (a \vee \bar{b} \vee \bar{c}) \wedge (\bar{a} \vee c)$.
After removing either $(a \vee \bar{b} \vee \bar{c})$ or $(\bar{a} \vee c)$,
the clause $(a \vee b)$ becomes blocked (*no clause with either $\bar{b}$ or $\bar{a}$*).
An extreme case in which BCE removes all clauses of a formula!

Proposition

BCE is confluent, i.e., has a unique fixpoint

- Blocked clauses stay blocked w.r.t. removal

Blocked Clause Elimination (BCE)

Definition (BCE)

While there is a blocked clause C in a CNF F, remove C from F.

Example

Consider $(a \vee b) \wedge (a \vee \bar{b} \vee \bar{c}) \wedge (\bar{a} \vee c)$.
After removing either $(a \vee \bar{b} \vee \bar{c})$ or $(\bar{a} \vee c)$,
the clause $(a \vee b)$ becomes blocked (*no clause with either $\bar{b}$ or $\bar{a}$*).
An extreme case in which BCE removes all clauses of a formula!

Proposition

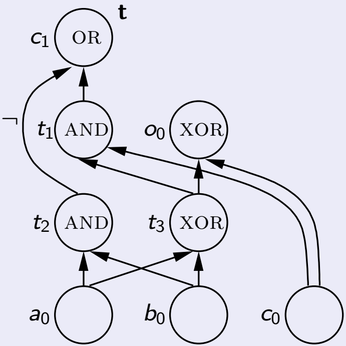
BCE is confluent, i.e., has a unique fixpoint

- Blocked clauses stay blocked w.r.t. removal

BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseiting encoding to Plaisted Greenbaum
BCE simulates Pure literal elimination, Cone of influence and much more

Example of circuit simplification by BCE on CNF



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseiting encoding to Plaisted Greenbaum
BCE simulates Pure literal elimination, Cone of influence and much more

Example of circuit simplification by BCE on CNF

(c_1)

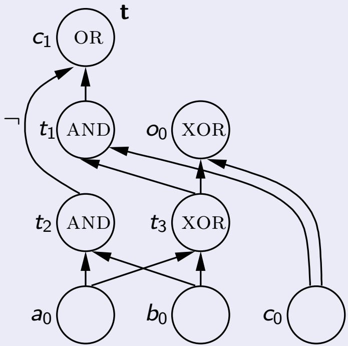
$(\neg c_1 \lor t_1 \lor t_2)$
$(c_1 \lor \neg t_1)$
$(c_1 \lor \neg t_2)$

$(\neg o_0 \lor t_3 \lor c_0)$
$(\neg o_0 \lor \neg t_3 \lor \neg c_0)$
$(o_0 \lor t_3 \lor \neg c_0)$
$(o_0 \lor \neg t_3 \lor c_0)$

$(t_1 \lor \neg t_3 \lor \neg c_0)$
$(\neg t_1 \lor t_3)$
$(\neg t_1 \lor c_0)$

$(t_2 \lor \neg a_0 \lor \neg b_0)$
$(\neg t_2 \lor a_0)$
$(\neg t_2 \lor b_0)$

$(\neg t_3 \lor a_0 \lor b_0)$
$(\neg t_3 \lor \neg a_0 \lor \neg b_0)$
$(t_3 \lor a_0 \lor \neg b_0)$
$(t_3 \lor \neg a_0 \lor b_0)$



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseiting encoding to Plaisted Greenbaum
BCE simulates Pure literal elimination, Cone of influence and much more

Example of circuit simplification by BCE on CNF

(c_1)

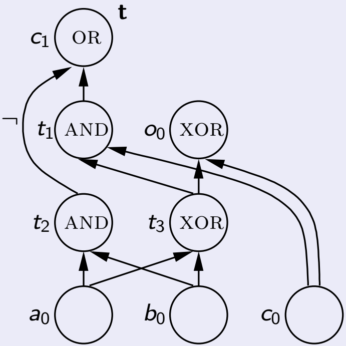
$(\neg c_1 \lor t_1 \lor t_2)$
$\cancel{(c_1 \lor \neg t_1)}$
$\cancel{(c_1 \lor \neg t_2)}$

$(\neg o_0 \lor t_3 \lor c_0)$
$(\neg o_0 \lor \neg t_3 \lor \neg c_0)$
$(o_0 \lor t_3 \lor \neg c_0)$
$(o_0 \lor \neg t_3 \lor c_0)$

$(t_1 \lor \neg t_3 \lor \neg c_0)$
$(\neg t_1 \lor t_3)$
$(\neg t_1 \lor c_0)$

$(t_2 \lor \neg a_0 \lor \neg b_0)$
$(\neg t_2 \lor a_0)$
$(\neg t_2 \lor b_0)$

$(\neg t_3 \lor a_0 \lor b_0)$
$(\neg t_3 \lor \neg a_0 \lor \neg b_0)$
$(t_3 \lor a_0 \lor \neg b_0)$
$(t_3 \lor \neg a_0 \lor b_0)$



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseiting encoding to Plaisted Greenbaum
BCE simulates Pure literal elimination, Cone of influence and much more

Example of circuit simplification by BCE on CNF

(c_1)

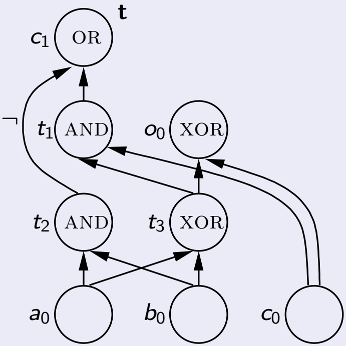
$(\neg c_1 \vee t_1 \vee t_2)$
$\cancel{(c_1 \vee \neg t_1)}$
$\cancel{(c_1 \vee \neg t_2)}$

$\cancel{(\neg o_0 \vee t_3 \vee c_0)}$
$\cancel{(\neg o_0 \vee \neg t_3 \vee \neg c_0)}$
$\cancel{(o_0 \vee t_3 \vee \neg c_0)}$
$\cancel{(o_0 \vee \neg t_3 \vee c_0)}$

$(t_1 \vee \neg t_3 \vee \neg c_0)$
$(\neg t_1 \vee t_3)$
$(\neg t_1 \vee c_0)$

$(t_2 \vee \neg a_0 \vee \neg b_0)$
$(\neg t_2 \vee a_0)$
$(\neg t_2 \vee b_0)$

$(\neg t_3 \vee a_0 \vee b_0)$
$(\neg t_3 \vee \neg a_0 \vee \neg b_0)$
$(t_3 \vee a_0 \vee \neg b_0)$
$(t_3 \vee \neg a_0 \vee b_0)$



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseiting encoding to Plaisted Greenbaum
BCE simulates Pure literal elimination, Cone of influence and much more

Example of circuit simplification by BCE on CNF

(c_1)

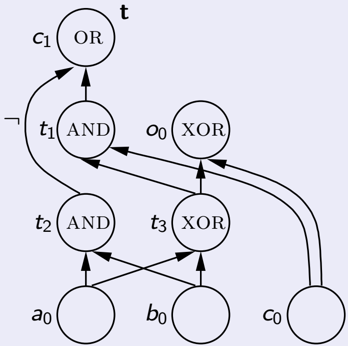
$(\neg c_1 \vee t_1 \vee t_2)$
$\cancel{(c_1 \vee \neg t_1)}$
$\cancel{(c_1 \vee \neg t_2)}$

$\cancel{(\neg o_0 \vee t_3 \vee c_0)}$
$\cancel{(\neg o_0 \vee \neg t_3 \vee \neg c_0)}$
$\cancel{(o_0 \vee t_3 \vee \neg c_0)}$
$\cancel{(o_0 \vee \neg t_3 \vee c_0)}$

$\cancel{(t_1 \vee \neg t_3 \vee \neg c_0)}$
$(\neg t_1 \vee t_3)$
$(\neg t_1 \vee c_0)$

$(t_2 \vee \neg a_0 \vee \neg b_0)$
$(\neg t_2 \vee a_0)$
$(\neg t_2 \vee b_0)$

$(\neg t_3 \vee a_0 \vee b_0)$
$(\neg t_3 \vee \neg a_0 \vee \neg b_0)$
$(t_3 \vee a_0 \vee \neg b_0)$
$(t_3 \vee \neg a_0 \vee b_0)$



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseiting encoding to Plaisted Greenbaum
BCE simulates Pure literal elimination, Cone of influence and much more

Example of circuit simplification by BCE on CNF

(c_1)

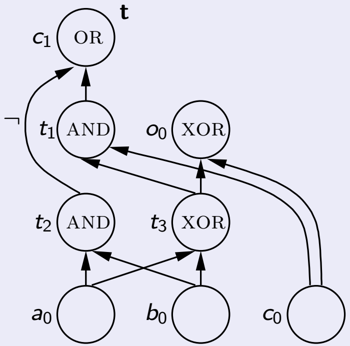
$(\neg c_1 \vee t_1 \vee t_2)$
$\cancel{(c_1 \vee \neg t_1)}$
$\cancel{(c_1 \vee \neg t_2)}$

$\cancel{(\neg o_0 \vee t_3 \vee c_0)}$
$\cancel{(\neg o_0 \vee \neg t_3 \vee \neg c_0)}$
$\cancel{(o_0 \vee t_3 \vee \neg c_0)}$
$\cancel{(o_0 \vee \neg t_3 \vee c_0)}$

$\cancel{(t_1 \vee \neg t_3 \vee \neg c_0)}$
$(\neg t_1 \vee t_3)$
$(\neg t_1 \vee c_0)$

$(t_2 \vee \neg a_0 \vee \neg b_0)$
$\cancel{(\neg t_2 \vee a_0)}$
$\cancel{(\neg t_2 \vee b_0)}$

$(\neg t_3 \vee a_0 \vee b_0)$
$(\neg t_3 \vee \neg a_0 \vee \neg b_0)$
$(t_3 \vee a_0 \vee \neg b_0)$
$(t_3 \vee \neg a_0 \vee b_0)$



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseiting encoding to Plaisted Greenbaum
BCE simulates Pure literal elimination, Cone of influence and much more

Example of circuit simplification by BCE on CNF

(c_1)

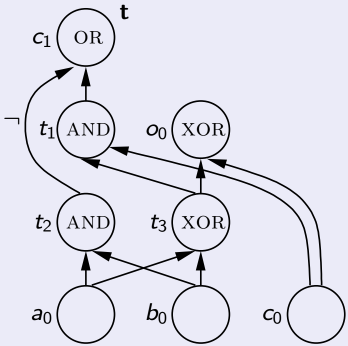
$(\neg c_1 \lor t_1 \lor t_2)$
$\cancel{(c_1 \lor \neg t_1)}$
$\cancel{(c_1 \lor \neg t_2)}$

$\cancel{(\neg o_0 \lor t_3 \lor c_0)}$
$\cancel{(\neg o_0 \lor \neg t_3 \lor \neg c_0)}$
$\cancel{(o_0 \lor t_3 \lor \neg c_0)}$
$\cancel{(o_0 \lor \neg t_3 \lor c_0)}$

$\cancel{(t_1 \lor \neg t_3 \lor \neg c_0)}$
$(\neg t_1 \lor t_3)$
$(\neg t_1 \lor c_0)$

$(t_2 \lor \neg a_0 \lor \neg b_0)$
$\cancel{(\neg t_2 \lor a_0)}$
$\cancel{(\neg t_2 \lor b_0)}$

$(\neg t_3 \lor a_0 \lor b_0)$
$(\neg t_3 \lor \neg a_0 \lor \neg b_0)$
$\cancel{(t_3 \lor a_0 \lor \neg b_0)}$
$\cancel{(t_3 \lor \neg a_0 \lor b_0)}$



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseiting encoding to Plaisted Greenbaum
BCE simulates Pure literal elimination, Cone of influence and much more

Example of circuit simplification by BCE on CNF

(c_1)

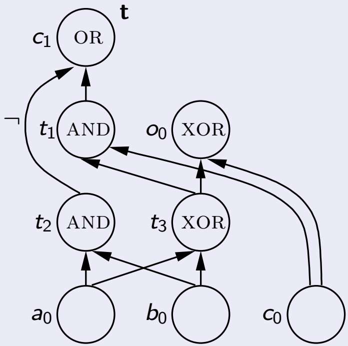
$(\neg c_1 \vee t_1 \vee t_2)$
$\cancel{(c_1 \vee \neg t_1)}$
$\cancel{(c_1 \vee \neg t_2)}$

$\cancel{(\neg o_0 \vee t_3 \vee c_0)}$
$\cancel{(\neg o_0 \vee \neg t_3 \vee \neg c_0)}$
$\cancel{(o_0 \vee t_3 \vee \neg c_0)}$
$\cancel{(o_0 \vee \neg t_3 \vee c_0)}$

$\cancel{(t_1 \vee \neg t_3 \vee \neg c_0)}$
$(\neg t_1 \vee t_3)$
$(\neg t_1 \vee c_0)$

$(t_2 \vee \neg a_0 \vee \neg b_0)$
$\cancel{(\neg t_2 \vee a_0)}$
$\cancel{(\neg t_2 \vee b_0)}$

$(\neg t_3 \vee a_0 \vee b_0)$
$(\neg t_3 \vee \neg a_0 \vee \neg b_0)$
$\cancel{(t_3 \vee a_0 \vee \neg b_0)}$
$\cancel{(t_3 \vee \neg a_0 \vee b_0)}$



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseiting encoding to Plaisted Greenbaum

BCE simulates Pure literal elimination, Cone of influence and much more

Example of circuit simplification by BCE on CNF

(c_1)

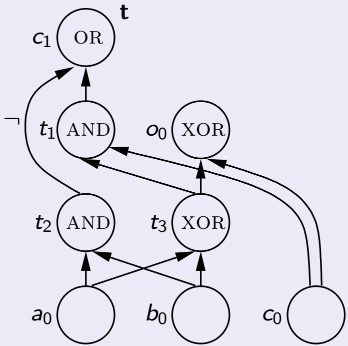
$\cancel{(\neg c_1 \vee t_1 \vee t_2)}$
$\cancel{(c_1 \vee \neg t_1)}$
$\cancel{(c_1 \vee \neg t_2)}$

$\cancel{(\neg o_0 \vee t_3 \vee c_0)}$
$\cancel{(\neg o_0 \vee \neg t_3 \vee \neg c_0)}$
$\cancel{(o_0 \vee t_3 \vee \neg c_0)}$
$\cancel{(o_0 \vee \neg t_3 \vee c_0)}$

$\cancel{(t_1 \vee \neg t_3 \vee \neg c_0)}$
$(\neg t_1 \vee t_3)$
$(\neg t_1 \vee c_0)$

$\cancel{(t_2 \vee \neg a_0 \vee \neg b_0)}$
$\cancel{(\neg t_2 \vee a_0)}$
$\cancel{(\neg t_2 \vee b_0)}$

$(\neg t_3 \vee a_0 \vee b_0)$
$(\neg t_3 \vee \neg a_0 \vee \neg b_0)$
$\cancel{(t_3 \vee a_0 \vee \neg b_0)}$
$\cancel{(t_3 \vee \neg a_0 \vee b_0)}$



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseiting encoding to Plaisted Greenbaum
BCE simulates Pure literal elimination, Cone of influence and much more

Example of circuit simplification by BCE on CNF

$\cancel{(c_1)}$

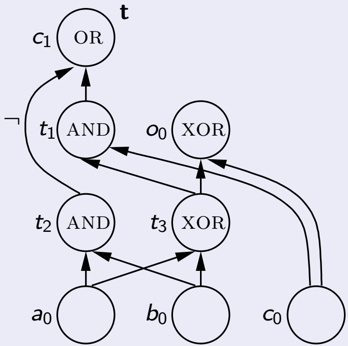
$\cancel{(\neg c_1 \vee t_1 \vee t_2)}$
$\cancel{(c_1 \vee \neg t_1)}$
$\cancel{(c_1 \vee \neg t_2)}$

$\cancel{(\neg o_0 \vee t_3 \vee c_0)}$
$\cancel{(\neg o_0 \vee \neg t_3 \vee \neg c_0)}$
$\cancel{(o_0 \vee t_3 \vee \neg c_0)}$
$\cancel{(o_0 \vee \neg t_3 \vee c_0)}$

$\cancel{(t_1 \vee \neg t_3 \vee \neg c_0)}$
$(\neg t_1 \vee t_3)$
$(\neg t_1 \vee c_0)$

$\cancel{(t_2 \vee \neg a_0 \vee \neg b_0)}$
$\cancel{(\neg t_2 \vee a_0)}$
$\cancel{(\neg t_2 \vee b_0)}$

$(\neg t_3 \vee a_0 \vee b_0)$
$(\neg t_3 \vee \neg a_0 \vee \neg b_0)$
$\cancel{(t_3 \vee a_0 \vee \neg b_0)}$
$\cancel{(t_3 \vee \neg a_0 \vee b_0)}$



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseiting encoding to Plaisted Greenbaum
BCE simulates Pure literal elimination, Cone of influence and much more

Example of circuit simplification by BCE on CNF

$\cancel{(c_1)}$

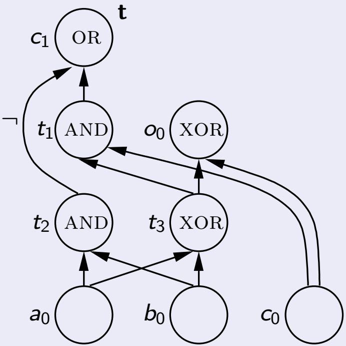
$\cancel{(\neg c_1 \vee t_1 \vee t_2)}$
$\cancel{(c_1 \vee \neg t_1)}$
$\cancel{(c_1 \vee \neg t_2)}$

$\cancel{(\neg o_0 \vee t_3 \vee c_0)}$
$\cancel{(\neg o_0 \vee \neg t_3 \vee \neg c_0)}$
$\cancel{(o_0 \vee t_3 \vee \neg c_0)}$
$\cancel{(o_0 \vee \neg t_3 \vee c_0)}$

$\cancel{(t_1 \vee \neg t_3 \vee \neg c_0)}$
$\cancel{(\neg t_1 \vee t_3)}$
$\cancel{(\neg t_1 \vee c_0)}$

$\cancel{(t_2 \vee \neg a_0 \vee \neg b_0)}$
$\cancel{(\neg t_2 \vee a_0)}$
$\cancel{(\neg t_2 \vee b_0)}$

$(\neg t_3 \vee a_0 \vee b_0)$
$(\neg t_3 \vee \neg a_0 \vee \neg b_0)$
$\cancel{(t_3 \vee a_0 \vee \neg b_0)}$
$\cancel{(t_3 \vee \neg a_0 \vee b_0)}$



BCE very effective on circuits [JärvisaloBiereHeule'10]

BCE converts the Tseiting encoding to Plaisted Greenbaum
BCE simulates Pure literal elimination, Cone of influence and much more

Example of circuit simplification by BCE on CNF

$\cancel{(c_1)}$

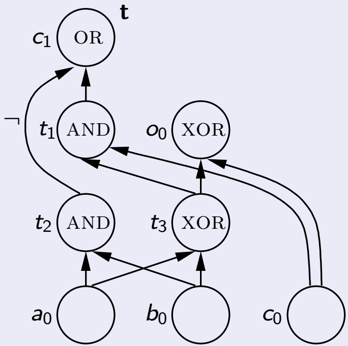
$\cancel{(\neg c_1 \vee t_1 \vee t_2)}$
$\cancel{(c_1 \vee \neg t_1)}$
$\cancel{(c_1 \vee \neg t_2)}$

$\cancel{(\neg o_0 \vee t_3 \vee c_0)}$
$\cancel{(\neg o_0 \vee \neg t_3 \vee \neg c_0)}$
$\cancel{(o_0 \vee t_3 \vee \neg c_0)}$
$\cancel{(o_0 \vee \neg t_3 \vee c_0)}$

$\cancel{(t_1 \vee \neg t_3 \vee \neg c_0)}$
$\cancel{(\neg t_1 \vee t_3)}$
$\cancel{(\neg t_1 \vee c_0)}$

$\cancel{(t_2 \vee \neg a_0 \vee \neg b_0)}$
$\cancel{(\neg t_2 \vee a_0)}$
$\cancel{(\neg t_2 \vee b_0)}$

$\cancel{(\neg t_3 \vee a_0 \vee b_0)}$
$\cancel{(\neg t_3 \vee \neg a_0 \vee \neg b_0)}$
$\cancel{(t_3 \vee a_0 \vee \neg b_0)}$
$\cancel{(t_3 \vee \neg a_0 \vee b_0)}$



Unhiding redundancy

Redundancy

Redundant clauses:

- Removal of $C \in F$ preserves unsatisfiability of F
- Assign $l \in C$ to false and check for a conflict in $F \setminus \{C\}$

Redundant literals:

- Removal of $l \in C$ preserves satisfiability of F
- Assign $l' \in C \setminus \{l\}$ to false and check if l is forced to false

Redundancy elimination during pre- and in-processing

- Distillation [JinSomenzi2005]
- ReVivAl [PietteHamadiSaïs2008]
- Unhiding [HeuleJärvisaloBiere2011]

Redundancy

Redundant clauses:

- Removal of $C \in F$ preserves unsatisfiability of F
- Assign $l \in C$ to false and check for a conflict in $F \setminus \{C\}$

Redundant literals:

- Removal of $l \in C$ preserves satisfiability of F
- Assign $l' \in C \setminus \{l\}$ to false and check if l is forced to false

Redundancy elimination during pre- and in-processing

- Distillation [JinSomenzi2005]
- ReVivAl [PietteHamadiSaïs2008]
- Unhiding [HeuleJärvisaloBiere2011]

Redundancy

Redundant clauses:

- Removal of $C \in F$ preserves unsatisfiability of F
- Assign $l \in C$ to false and check for a conflict in $F \setminus \{C\}$

Redundant literals:

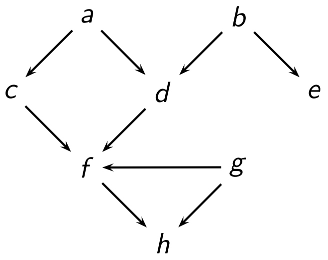
- Removal of $l \in C$ preserves satisfiability of F
- Assign $l' \in C \setminus \{l\}$ to false and check if l is forced to false

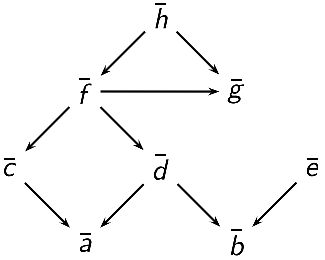
Redundancy elimination during pre- and in-processing

- Distillation [JinSomenzi2005]
- ReVivAl [PietteHamadiSaïs2008]
- Unhiding [HeuleJärvisaloBiere2011]

Unhide: Binary implication graph (BIG)

unhide: use the binary clauses to detect redundant clauses and literals

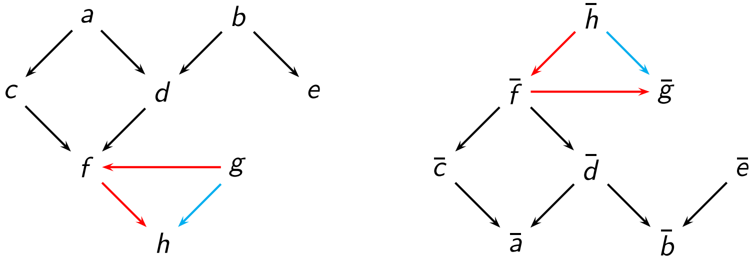




$$(\bar{a} \vee c) \wedge (\bar{a} \vee d) \wedge (\bar{b} \vee d) \wedge (\bar{b} \vee e) \wedge$$
$$(\bar{c} \vee f) \wedge (\bar{d} \vee f) \wedge (\bar{g} \vee f) \wedge (\bar{f} \vee h) \wedge$$
$$(\bar{g} \vee h) \wedge \underbrace{(\bar{a} \vee \bar{e} \vee h) \wedge (\bar{b} \vee \bar{c} \vee h) \wedge (a \vee b \vee c \vee d \vee e \vee f \vee g \vee h)}_{\text{non binary clauses}}$$

Unhide: Transitive reduction (TRD)

transitive reduction: remove shortcuts in the binary implication graph

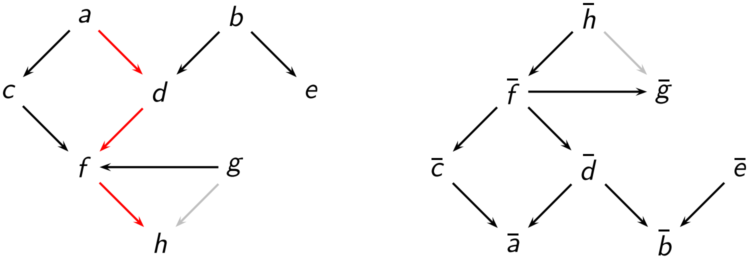


$$(\bar{a} \vee c) \wedge (\bar{a} \vee d) \wedge (\bar{b} \vee d) \wedge (\bar{b} \vee e) \wedge$$
$$(\bar{c} \vee f) \wedge (\bar{d} \vee f) \wedge (\bar{g} \vee f) \wedge (\bar{f} \vee h) \wedge$$
$$\underset{\text{TRD}}{\cancel{(\bar{g} \vee h)}} \wedge (\bar{a} \vee \bar{e} \vee h) \wedge (\bar{b} \vee \bar{c} \vee h) \wedge (a \vee b \vee c \vee d \vee e \vee f \vee g \vee h)$$

$g \rightarrow f \rightarrow h$

Unhide: Hidden tautology elimination (HTE) (1)

HTE removes clauses that are subsumed by an implication in BIG



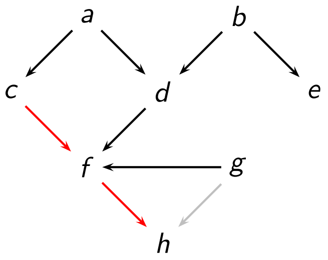
$$(\bar{a} \vee c) \wedge (\bar{a} \vee d) \wedge (\bar{b} \vee d) \wedge (\bar{b} \vee e) \wedge$$
$$(\bar{c} \vee f) \wedge (\bar{d} \vee f) \wedge (\bar{g} \vee f) \wedge (\bar{f} \vee h) \wedge$$
$$\cancel{(\bar{a} \vee \bar{e} \vee h)} \wedge (\bar{b} \vee \bar{c} \vee h) \wedge (a \vee b \vee c \vee d \vee e \vee f \vee g \vee h)$$

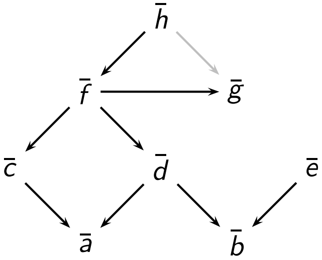
HTE

$a \rightarrow d \rightarrow f \rightarrow h$

Unhide: Hidden tautology elimination (HTE) (2)

HTE removes clauses that are subsumed by an implication in BIG





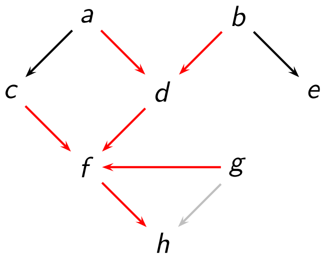
$$(\bar{a} \vee c) \wedge (\bar{a} \vee d) \wedge (\bar{b} \vee d) \wedge (\bar{b} \vee e) \wedge$$
$$(\bar{c} \vee f) \wedge (\bar{d} \vee f) \wedge (\bar{g} \vee f) \wedge (\bar{f} \vee h) \wedge$$
$$\cancel{(\bar{b} \vee \bar{c} \vee h)} \wedge (a \vee b \vee c \vee d \vee e \vee f \vee g \vee h)$$

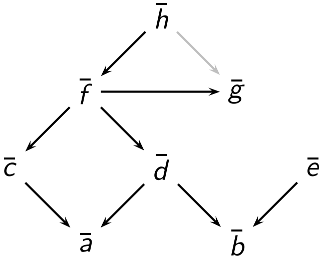
HTE

$c \rightarrow f \rightarrow h$

Unhide: Hidden literal elimination (HLE)

HLE removes literal using the implication in BIG





$$(\bar{a} \vee c) \wedge (\bar{a} \vee d) \wedge (\bar{b} \vee d) \wedge (\bar{b} \vee e) \wedge$$
$$(\bar{c} \vee f) \wedge (\bar{d} \vee f) \wedge (\bar{g} \vee f) \wedge (\bar{f} \vee h) \wedge$$

$$(\cancel{a \vee b \vee c \vee d \vee}\ e \vee \cancel{f \vee g \vee}\ h)$$

HLE

all but e imply h

also b implies e

Conclusions: state-of-the-art SAT solver

Key contributions to SAT search engine:

- adding conflict clauses (grasp) [Marques-Silva'96]
- restart strategies [GomesSC'97,LubySZ'93]
- 2-watch pointers and VSIDS (zChaff) [MoskewiczMZZM'01]
- efficient implementation (Minisat) [EenSörensson'03]
- variable elimination (SatElite) [EenBiere'05]
- phase-saving (Rsat) [PipatsrisawatDarwiche'07]

Recent progress: pre- and in-processing

- removal of redundant clauses and literals [JinSomenzi'05]
- removal of blocked clauses [JärvisaloBiereHeule'10]
- unhiding redundancy [HeuleJärvisaloBiere'11]

Conclusions: state-of-the-art SAT solver

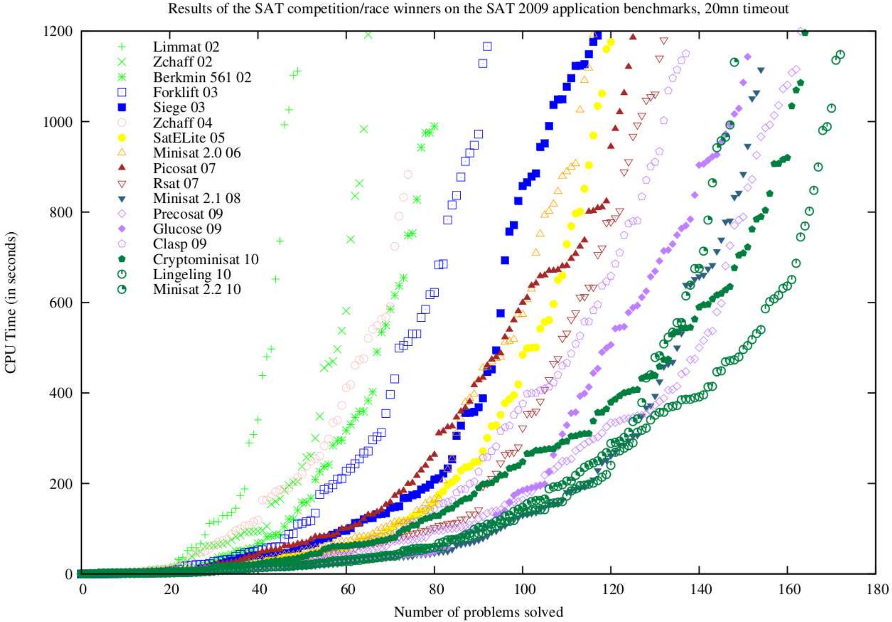
Key contributions to SAT search engine:

- adding conflict clauses (grasp) [Marques-Silva'96]
- restart strategies [GomesSC'97,LubySZ'93]
- 2-watch pointers and VSIDS (zChaff) [MoskewiczMZZM'01]
- efficient implementation (Minisat) [EenSörensson'03]
- variable elimination (SatElite) [EenBiere'05]
- phase-saving (Rsat) [PipatsrisawatDarwiche'07]

Recent progress: pre- and in-processing

- removal of redundant clauses and literals [JinSomenzi'05]
- removal of blocked clauses [JärvisaloBiereHeule'10]
- unhiding redundancy [HeuleJärvisaloBiere'11]

Cactus plot: Lingeling [Biere'10] contains all features



Call for PhD Students and Postdocs

The RiSE project is a National Research Network funded by the Austrian National Science Foundation (FWF) on the topic of Rigorous Systems Engineering.

RiSE is looking for several postdocs and PhD students in formal methods, systems engineering and related fields such as programming languages and distributed systems, to work on topics such as:

- design and verification of concurrent and real-time software;
- reactive synthesis and game theory;
- logical decision procedures.

Armin Biere - JKU Linz (biere@jku.at)
Roderick Bloem - TU Graz (roderick.bloem@iaik.tugraz.at)
Krishnendu Chatterjee - IST Austria (krishnendu.chatterjee@ist.ac.at)
Uwe Egly - TU Wien (uwe@kr.tuwien.ac.at)
Thomas A. Henzinger - IST Austria (tah@ist.ac.at)
Christoph Kirsch - PLU Salzburg (ck@cs.uni-salzburg.at)
Laura Kovcs - TU Wien (lkovacs@complang.tuwien.ac.at)
Ulrich Schmid - TU Wien (s@ecs.tuwien.ac.at)
Helmut Veith - TU Wien (veith@forsyte.tuwien.ac.at)