Exploitation of dynamic symmetries for solving SAT problems

Thèse de doctorat de Sorbonne Université

Hakan METIN

Rapporteurs:

PASCAL FONTAINE	Maître de conférences, Université de Liège
LAURE PETRUCCI	Professeur, Université Paris 13

Reviewers:

BART BOGAERTS	Assistant Professor, Vrije Universiteit Brussel
JEAN-MICHEL COUVREUR	Professeur, Université d'Orléans
EMMANUELLE ENCRENAZ	Maître de conférences, Sorbonne Université

Directors:

SOUHEIB BAARIR	Maître de conférences, Université Paris Nanterre
FABRICE KORDON	Professeur, Sorbonne Université

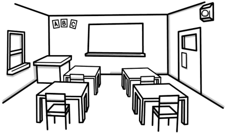
SCIENCES SORBONNE UNIVERSITÉ

Motivation

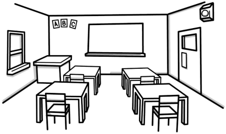
Boolean SATisfiability is widely used in different domains

- Artificial intelligence (planning [KS$^+$92], ...)
- Bioinformatics (haplotype inference [LMS06], ...)
- Security (cryptanalysis [MM00], ...)
- Computationally hard problems (ramsey numbers, graph coloring, ...)
- Formal methods,(bounded model checking [BCCZ99], ...)

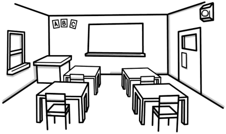
SAT an example



1

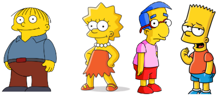


2

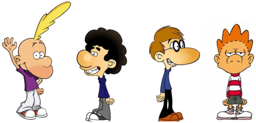


3

A



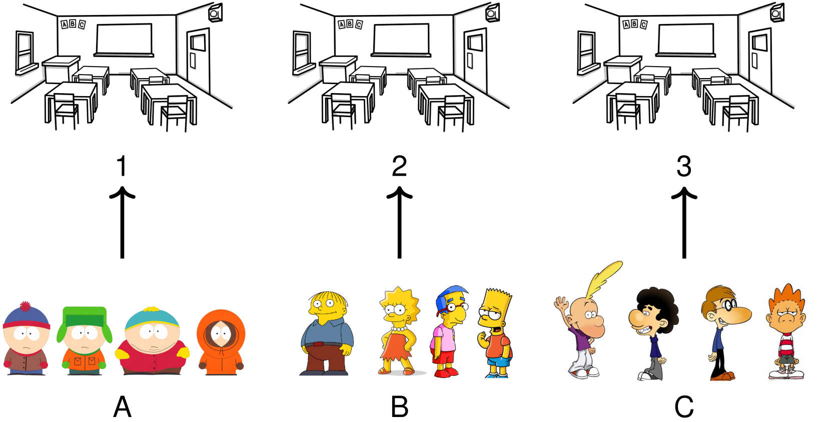
B



C

Is it possible to attribute each group to a classroom?

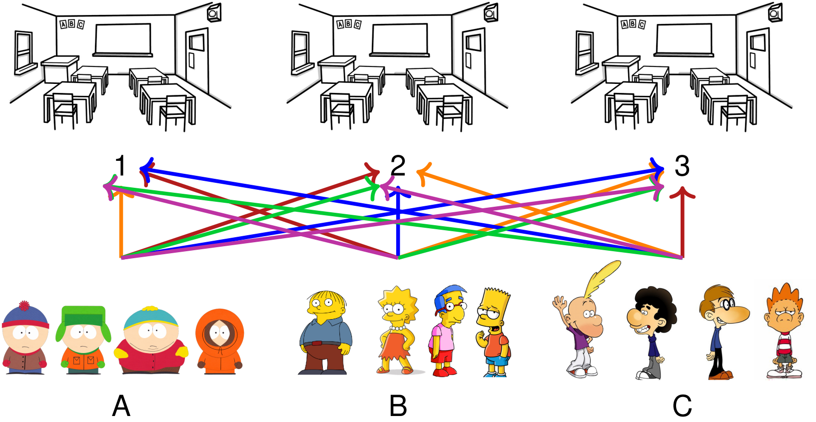
SAT an example



Is it possible to attribute each group to a classroom?

YES!

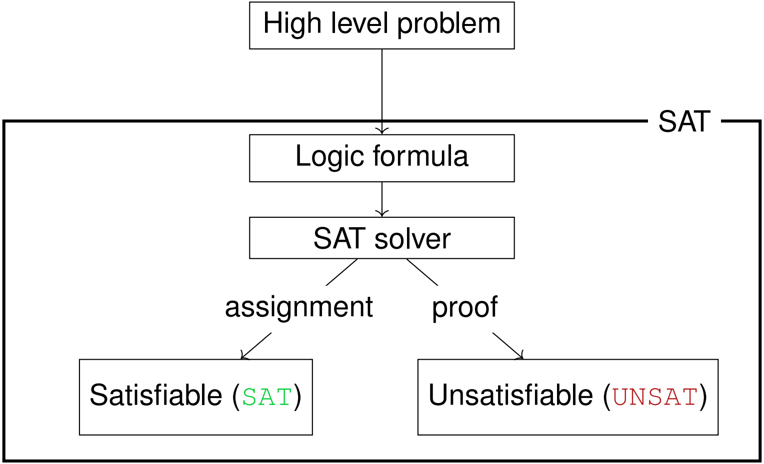
SAT an example



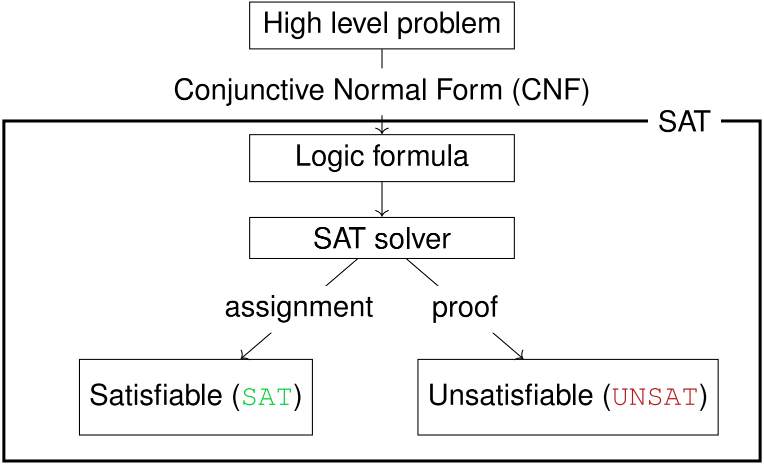
Is it possible to attribute each group to a classroom?

YES! Many solutions

Boolean SATisfiability problem



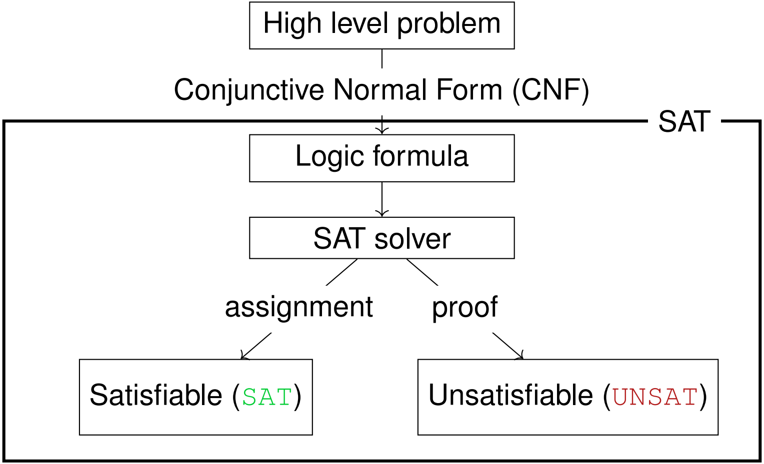
Boolean SATisfiability problem



CNF representation:

$$\underbrace{(x_1 \lor x_2 \lor \neg x_3)}_{\text{Clause with literals } x_1, x_2, \neg x_3}$$

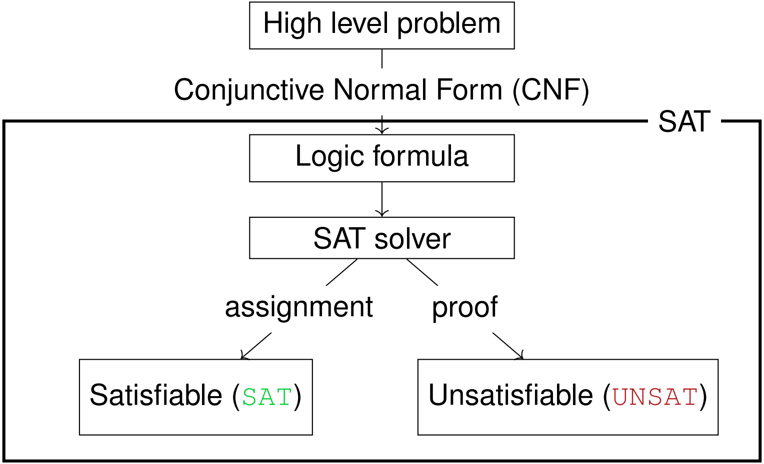
Boolean SATisfiability problem



CNF representation:

$$\overbrace{\underbrace{(x_1 \lor x_2 \lor \neg x_3)}_{\textit{Clause}} \land (\neg x_1 \lor \neg x_2) \land (x_2 \lor \neg x_4)}^{\text{Formula (CNF)}}$$

Boolean SATisfiability problem



Clause representation as a set:

$$(x_1 \vee x_2 \vee \neg x_3) \to \{x_1, x_2, \neg x_3\}$$

SAT Solving

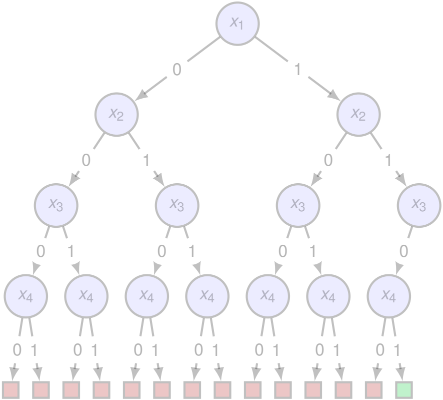
Solving SAT formula is known to be **NP-complete** [Coo71]

Enumerative algorithms:

- Davis, Putnam, Logemann, and Loveland (DPLL) [DLL62]
  - Boolean Constraint Propagation (BCP)

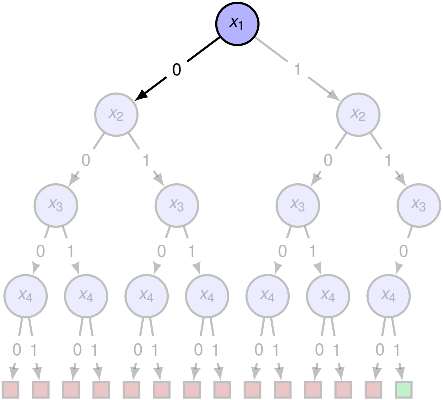
- Conflict Driven Clause Learning (CDCL) [MSS99]
  - Derived from DPLL
  - Clause learning

CDCL in action



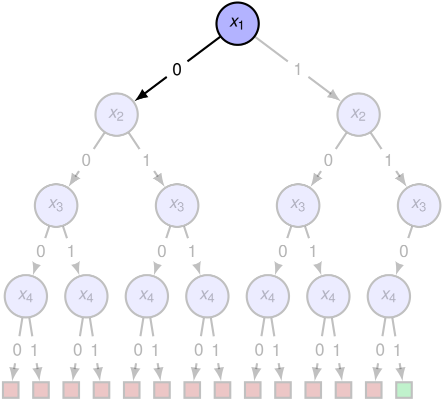
$$\omega_1 = \{x_1, x_2, x_3, x_4\}$$
$$\omega_2 = \{x_1, \neg x_4\}$$
$$\omega_3 = \{x_1, x_4\}$$
$$\omega_4 = \{x_2, \neg x_4\}$$
$$\omega_5 = \{x_2, x_4\}$$
$$\omega_6 = \{x_3, x_4\}$$

CDCL in action



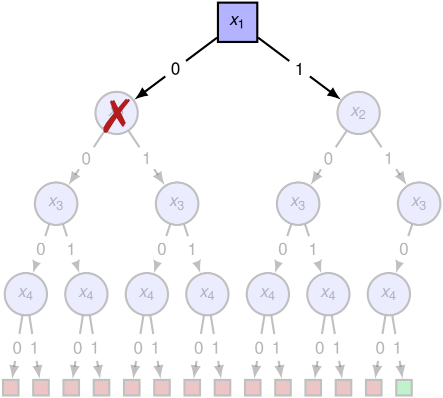
$$\omega_1 = \{x_1, x_2, x_3, x_4\}$$
$$\omega_2 = \{x_1, \neg x_4\}$$
$$\omega_3 = \{x_1, x_4\}$$
$$\omega_4 = \{x_2, \neg x_4\}$$
$$\omega_5 = \{x_2, x_4\}$$
$$\omega_6 = \{x_3, x_4\}$$

CDCL in action



$$\omega_1 = \{x_1, x_2, x_3, x_4\}$$
$$\omega_2 = \{x_1, \neg x_4\}$$
$$\omega_3 = \{x_1, x_4\}$$
$$\omega_4 = \{x_2, \neg x_4\}$$
$$\omega_5 = \{x_2, x_4\}$$
$$\omega_6 = \{x_3, x_4\}$$

CDCL in action



$\omega_1 = \{x_1, x_2, x_3, x_4\}$

$\omega_2 = \{x_1, \neg x_4\}$

$\omega_3 = \{x_1, x_4\}$

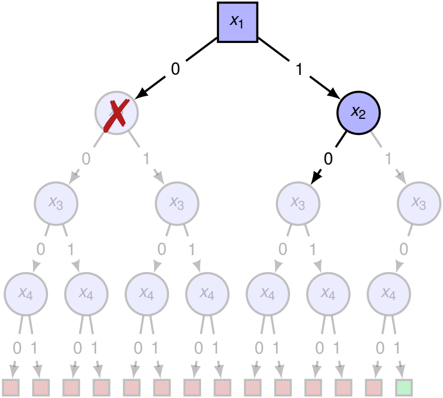
$\omega_4 = \{x_2, \neg x_4\}$

$\omega_5 = \{x_2, x_4\}$

$\omega_6 = \{x_3, x_4\}$

$\omega_7 = \{x_1\}$

CDCL in action



$$\omega_1 = \{x_1, x_2, x_3, x_4\}$$

$$\omega_2 = \{x_1, \neg x_4\}$$

$$\omega_3 = \{x_1, x_4\}$$

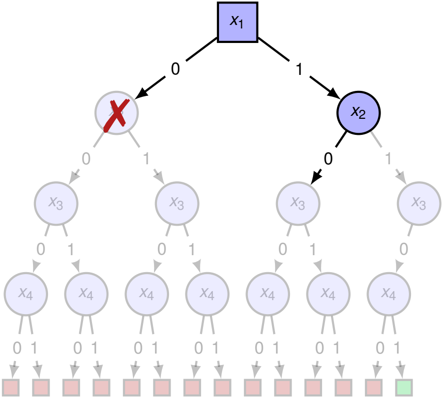
$$\omega_4 = \{x_2, \neg x_4\}$$

$$\omega_5 = \{x_2, x_4\}$$

$$\omega_6 = \{x_3, x_4\}$$

$$\omega_7 = \{x_1\}$$

CDCL in action



$\omega_1 = \{x_1, x_2, x_3, x_4\}$

$\omega_2 = \{x_1, \neg x_4\}$

$\omega_3 = \{x_1, x_4\}$

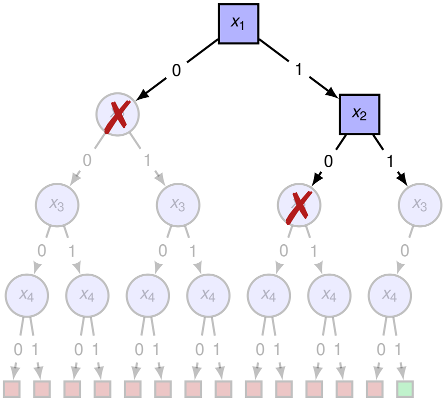
$\omega_4 = \{x_2, \neg x_4\}$

$\omega_5 = \{x_2, x_4\}$

$\omega_6 = \{x_3, x_4\}$

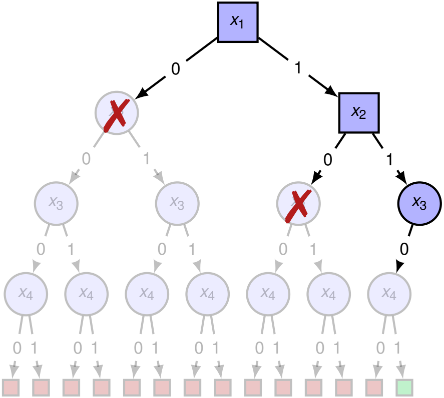
$\omega_7 = \{x_1\}$

CDCL in action



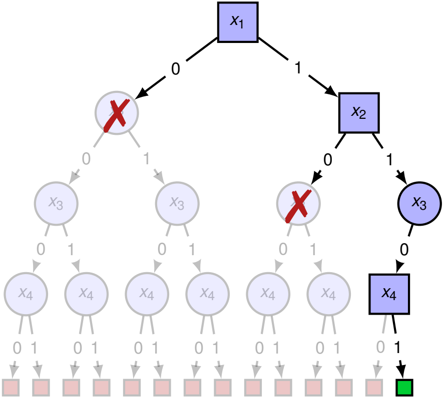
$$\omega_1 = \{x_1, x_2, x_3, x_4\}$$
$$\omega_2 = \{x_1, \neg x_4\}$$
$$\omega_3 = \{x_1, x_4\}$$
$$\omega_4 = \{x_2, \neg x_4\}$$
$$\omega_5 = \{x_2, x_4\}$$
$$\omega_6 = \{x_3, x_4\}$$
$$\omega_7 = \{x_1\}$$
$$\omega_8 = \{x_2\}$$

CDCL in action



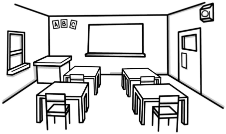
$$\omega_1 = \{x_1, x_2, x_3, x_4\}$$
$$\omega_2 = \{x_1, \neg x_4\}$$
$$\omega_3 = \{x_1, x_4\}$$
$$\omega_4 = \{x_2, \neg x_4\}$$
$$\omega_5 = \{x_2, x_4\}$$
$$\omega_6 = \{x_3, x_4\}$$
$$\omega_7 = \{x_1\}$$
$$\omega_8 = \{x_2\}$$

CDCL in action

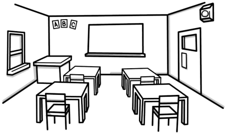


$$\omega_1 = \{x_1, x_2, x_3, x_4\}$$
$$\omega_2 = \{x_1, \neg x_4\}$$
$$\omega_3 = \{x_1, x_4\}$$
$$\omega_4 = \{x_2, \neg x_4\}$$
$$\omega_5 = \{x_2, x_4\}$$
$$\omega_6 = \{x_3, x_4\}$$
$$\omega_7 = \{x_1\}$$
$$\omega_8 = \{x_2\}$$

An UNSAT example

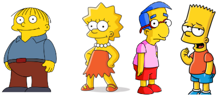


1



2

A

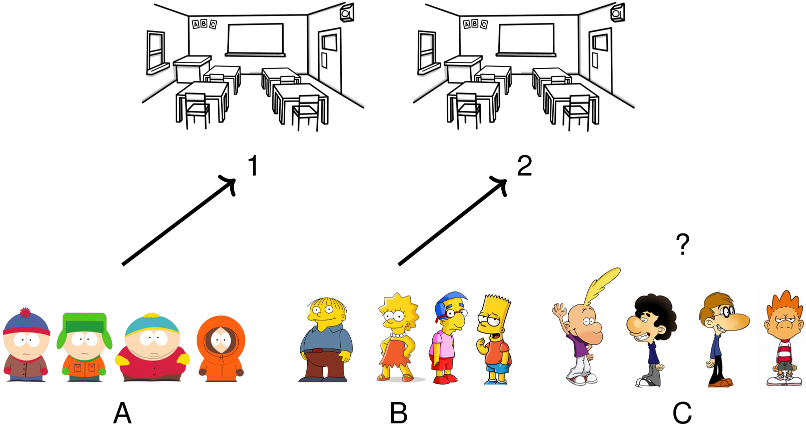


B

C

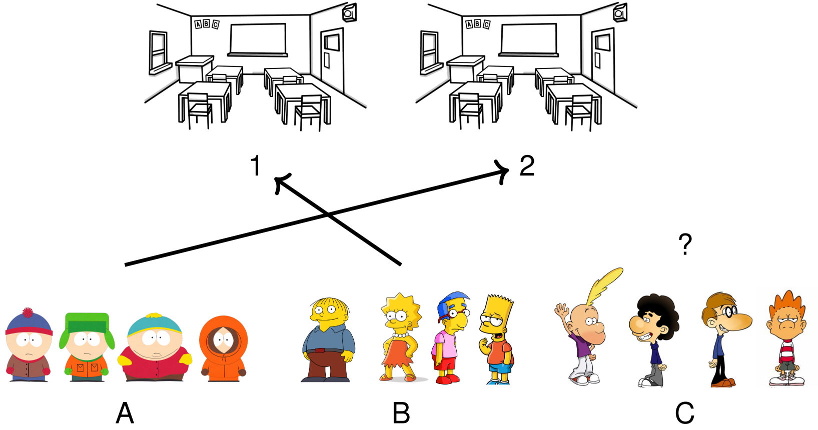
Is it possible to attribute each group to a classroom?

An UNSAT example



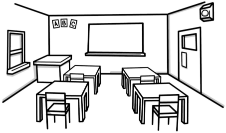
Is it possible to attribute each group to a classroom?

An UNSAT example

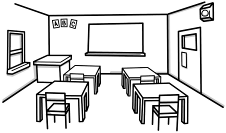


Is it possible to attribute each group to a classroom?

An UNSAT example



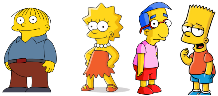
1



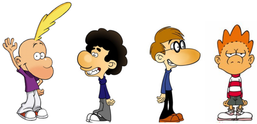
2

. . .

A



B

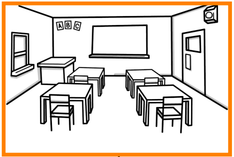


C

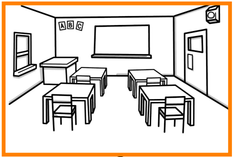
Is it possible to attribute each group to a classroom?

No!

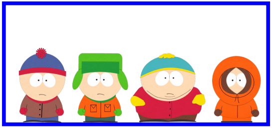
An UNSAT example



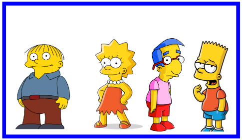
1



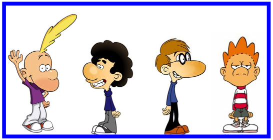
2



A



B



C

Is it possible to attribute each group to a classroom?

No!

Presence of symmetries hinders the performance of the solver

Symmetry

A symmetry (permuation) g is a bijective function (on variables) that leaves the formula invariant

$$g = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \\ x_2 & x_1 & x_3 & x_5 & x_4 & x_6 & x_8 & x_7 & x_9 \end{pmatrix} \rightarrow (x_1 \ \ x_2)(x_4 \ \ x_5)(x_7 \ \ x_8)$$

$$\begin{array}{lcl}
\omega_1 = \{x_1, x_2, x_3\} & \longleftrightarrow & \omega_1 = \{x_2, x_1, x_3\} \\
\omega_2 = \{x_4, x_5, x_6\} & \longleftrightarrow & \omega_2 = \{x_5, x_4, x_6\} \\
\omega_3 = \{x_7, x_8, x_9\} & \longleftrightarrow & \omega_3 = \{x_8, x_7, x_9\} \\
\omega_4 = \{\neg x_1, \neg x_4\} & & \omega_7 = \{\neg x_2, \neg x_5\} \\
\omega_5 = \{\neg x_1, \neg x_7\} & & \omega_8 = \{\neg x_2, \neg x_8\} \\
\omega_6 = \{\neg x_4, \neg x_7\} & & \omega_9 = \{\neg x_5, \neg x_8\} \\
\omega_7 = \{\neg x_2, \neg x_5\} & & \omega_4 = \{\neg x_1, \neg x_4\} \\
\omega_8 = \{\neg x_2, \neg x_8\} & & \omega_5 = \{\neg x_1, \neg x_7\} \\
\omega_9 = \{\neg x_5, \neg x_8\} & & \omega_6 = \{\neg x_4, \neg x_7\} \\
\omega_{10} = \{\neg x_3, \neg x_6\} & \longleftrightarrow & \omega_{10} = \{\neg x_3, \neg x_6\} \\
\omega_{11} = \{\neg x_3, \neg x_9\} & \longleftrightarrow & \omega_{11} = \{\neg x_3, \neg x_9\} \\
\omega_{12} = \{\neg x_6, \neg x_9\} & \longleftrightarrow & \omega_{12} = \{\neg x_6, \neg x_9\}
\end{array}$$

The set of symmetries of a formula is a group noted G

Computing symmetries of a SAT problem

CNF formula

$$\begin{aligned}&(x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_7 \vee x_8 \vee x_9)\\ &\wedge(\neg x_1 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_7) \wedge (\neg x_4 \vee \neg x_7)\\ &\wedge(\neg x_2 \vee \neg x_5) \wedge (\neg x_2 \vee \neg x_8) \wedge (\neg x_5 \vee \neg x_8)\\ &\wedge(\neg x_3 \vee \neg x_6) \wedge (\neg x_3 \vee \neg x_9) \wedge (\neg x_6 \vee \neg x_9)\end{aligned}$$

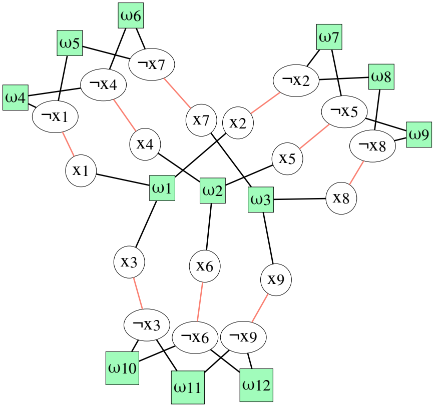
Computing symmetries of a SAT problem

CNF formula

$$(x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_7 \vee x_8 \vee x_9)$$
$$\wedge(\neg x_1 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_7) \wedge (\neg x_4 \vee \neg x_7)$$
$$\wedge(\neg x_2 \vee \neg x_5) \wedge (\neg x_2 \vee \neg x_8) \wedge (\neg x_5 \vee \neg x_8)$$
$$\wedge(\neg x_3 \vee \neg x_6) \wedge (\neg x_3 \vee \neg x_9) \wedge (\neg x_6 \vee \neg x_9)$$

$\Downarrow$ $\Downarrow$

colored graph



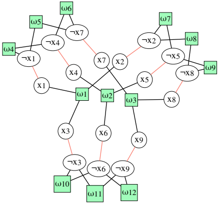
Computing symmetries of a SAT problem

CNF formula

$$(x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_7 \vee x_8 \vee x_9)$$
$$\wedge(\neg x_1 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_7) \wedge (\neg x_4 \vee \neg x_7)$$
$$\wedge(\neg x_2 \vee \neg x_5) \wedge (\neg x_2 \vee \neg x_8) \wedge (\neg x_5 \vee \neg x_8)$$
$$\wedge(\neg x_3 \vee \neg x_6) \wedge (\neg x_3 \vee \neg x_9) \wedge (\neg x_6 \vee \neg x_9)$$

$\Downarrow$ $\Downarrow$

colored graph



$\|$ $\|$

graph automorphism

(`bliss`, `saucy`, ...)

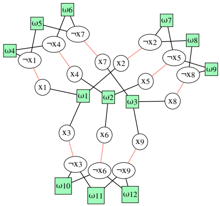
Computing symmetries of a SAT problem

CNF formula

$$(x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_7 \vee x_8 \vee x_9)$$
$$\wedge(\neg x_1 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_7) \wedge (\neg x_4 \vee \neg x_7)$$
$$\wedge(\neg x_2 \vee \neg x_5) \wedge (\neg x_2 \vee \neg x_8) \wedge (\neg x_5 \vee \neg x_8)$$
$$\wedge(\neg x_3 \vee \neg x_6) \wedge (\neg x_3 \vee \neg x_9) \wedge (\neg x_6 \vee \neg x_9)$$

$\Downarrow$ $\Downarrow$

colored graph



$\|$ $\|$

graph automorphism

(`bliss`, `saucy`, ...)

$\Downarrow$ $\Downarrow$

set of symmetries

$$g_1 = (x_2\ \ x_3)(x_5\ \ x_6)(x_8\ \ x_9)$$
$$g_2 = (x_4\ \ x_7)(x_5\ \ x_8)(x_6\ \ x_9)$$
$$g_3 = (x_1\ \ x_2)(x_4\ \ x_5)(x_7\ \ x_8)$$
$$g_4 = (x_1\ \ x_4)(x_2\ \ x_5)(x_3\ \ x_6)$$

Orbit

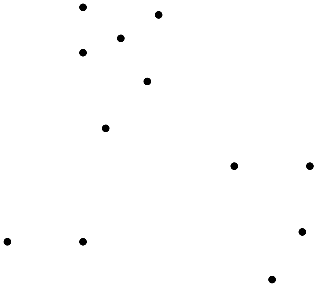
Orbit of an assignment $\alpha = G.\alpha = \{g.\alpha \mid g \in G\}$

Orbit

Orbit of an assignment $\alpha = G.\alpha = \{g.\alpha \mid g \in G\}$

Example:

- full assignment

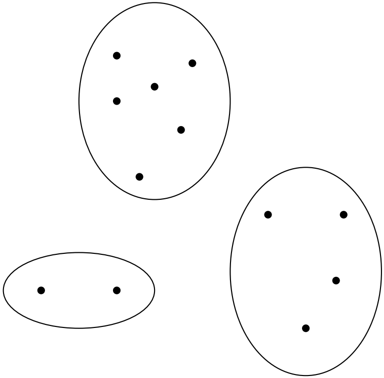


Orbit

Orbit of an assignment $\alpha = G.\alpha = \{g.\alpha \mid g \in G\}$

Example:

- full assignment
- orbit

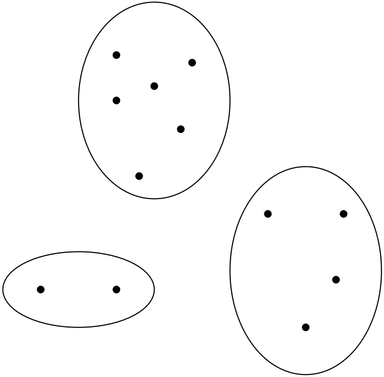


Orbit

Orbit of an assignment $\alpha = G.\alpha = \{g.\alpha \mid g \in G\}$

Example:

- full assignment
- orbit

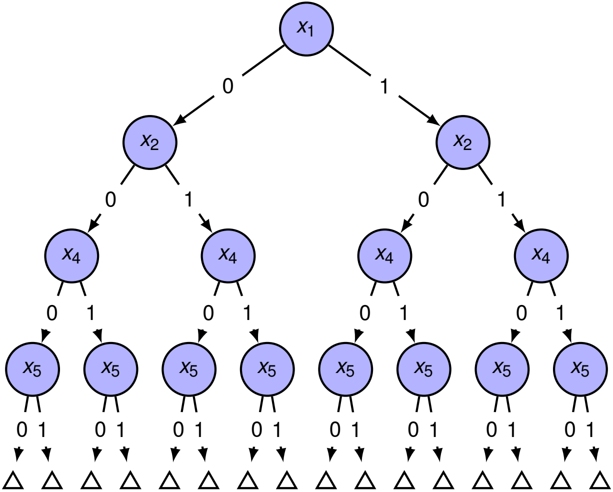


Equivalence relation with respect to SAT:

- Either $G.\alpha$ contains no solution
- Or all elements of $G.\alpha$ are solutions

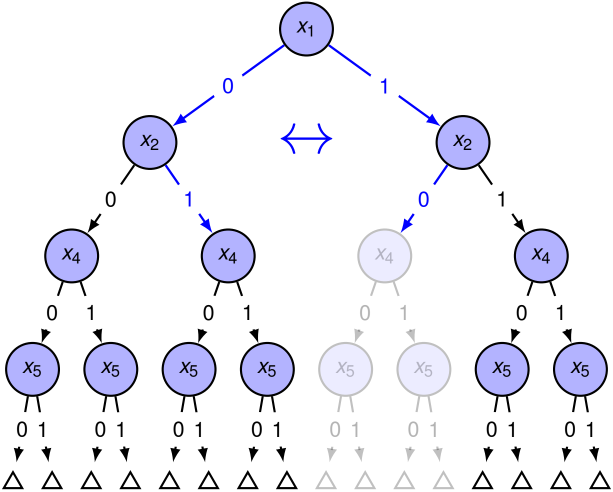
Using symmetries to prune the search space

$g = (x_1 \;\; x_2)(x_4 \;\; x_5)(x_7 \;\; x_8)$



Using symmetries to prune the search space

$g = (x_1 \;\; x_2)(x_4 \;\; x_5)(x_7 \;\; x_8)$



Generates symmetry breaking predicates (SBP)

- Define lexicographic order
  - Define total order on variables
  - Define minimal value
- Forbid non minimal assignment for each orbit

Example:

$$x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6 \leq x_7 \leq x_8; \mathtt{F} < \mathtt{T}$$

$$g = (x_1 \ \ x_2)(x_4 \ \ x_5)(x_7 \ \ x_8)$$

	x_1	x_2	x_3	x_4	x_5	$\cdots$	lex-leader	SBP

Generates symmetry breaking predicates (SBP)

- Define lexicographic order
  - Define total order on variables
  - Define minimal value
- Forbid non minimal assignment for each orbit

Example:

$$x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6 \leq x_7 \leq x_8; \mathtt{F} < \mathtt{T}$$

$$g = (x_1\ \ x_2)(x_4\ \ x_5)(x_7\ \ x_8)$$

	x_1	x_2	x_3	x_4	x_5	$\cdots$	lex-leader	SBP
O_1	F	T	–	–	–	$\cdots$	✓	

Generates symmetry breaking predicates (SBP)

- Define lexicographic order
  - Define total order on variables
  - Define minimal value
- Forbid non minimal assignment for each orbit

Example:

$$x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6 \leq x_7 \leq x_8; \mathtt{F} < \mathtt{T}$$

$$g = (x_1\ \ x_2)(x_4\ \ x_5)(x_7\ \ x_8)$$

	x_1	x_2	x_3	x_4	x_5	$\cdots$	lex-leader	SBP
O_1	F	T	–	–	–	$\cdots$	✓	
	T	F	–	–	–	$\cdots$	✗	$\rightarrow \neg x_1 \vee x_2$

Generates symmetry breaking predicates (SBP)

- Define lexicographic order
  - Define total order on variables
  - Define minimal value
- Forbid non minimal assignment for each orbit

Example:

$$x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6 \leq x_7 \leq x_8; \mathtt{F} < \mathtt{T}$$

$$g = (x_1 \ \ x_2)(x_4 \ \ x_5)(x_7 \ \ x_8)$$

	x_1	x_2	x_3	x_4	x_5	$\cdots$	lex-leader	SBP
O_1	F	T	–	–	–	$\cdots$	✓	
	T	F	–	–	–	$\cdots$	✗	$\rightarrow \neg x_1 \vee x_2$
O_2	F	F	–	F	T	$\cdots$	✓	

Generates symmetry breaking predicates (SBP)

- Define lexicographic order
  - Define total order on variables
  - Define minimal value
- Forbid non minimal assignment for each orbit

Example:

$$x_1 \le x_2 \le x_3 \le x_4 \le x_5 \le x_6 \le x_7 \le x_8 ; \texttt{F} < \texttt{T}$$

$$g = (x_1 \ \ x_2)(x_4 \ \ x_5)(x_7 \ \ x_8)$$

	x_1	x_2	x_3	x_4	x_5	$\cdots$	lex-leader	SBP
O_1	F	T	–	–	–	$\cdots$	✓	
	T	F	–	–	–	$\cdots$	✗	$\rightarrow \neg x_1 \vee x_2$
O_2	F	F	–	F	T	$\cdots$	✓	
	F	F	–	T	F	$\cdots$	✗	$\rightarrow x_1 \vee x_2 \vee \neg x_4 \vee x_5$

Generates symmetry breaking predicates (SBP)

- Define lexicographic order
  - Define total order on variables
  - Define minimal value
- Forbid non minimal assignment for each orbit

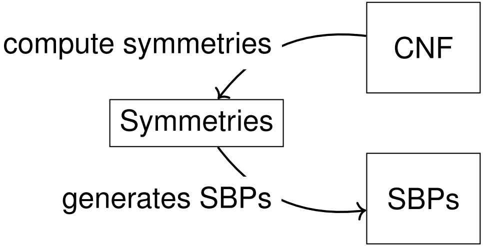
Example:

$$x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6 \leq x_7 \leq x_8; \mathtt{F} < \mathtt{T}$$

$$g = (x_1 \ \ x_2)(x_4 \ \ x_5)(x_7 \ \ x_8)$$

	x_1	x_2	x_3	x_4	x_5	$\cdots$	lex-leader	SBP
O_1	F	T	–	–	–	$\cdots$	✓	
	T	F	–	–	–	$\cdots$	✗	$\rightarrow \neg x_1 \vee x_2$
O_2	F	F	–	F	T	$\cdots$	✓	
	F	F	–	T	F	$\cdots$	✗	$\rightarrow x_1 \vee x_2 \vee \neg x_4 \vee x_5$
				$\cdots$				

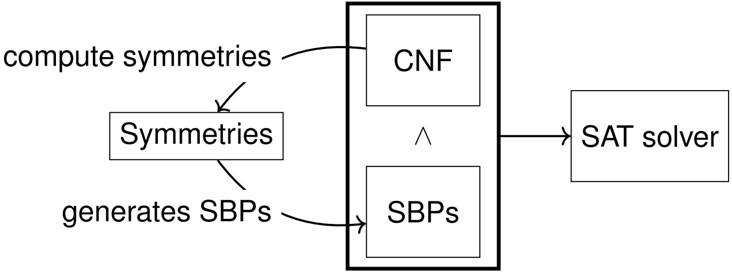
Static symmetry breaking



Different approaches:

- Shatter [ASM06]
- BreakID [DBBD16]
- ...

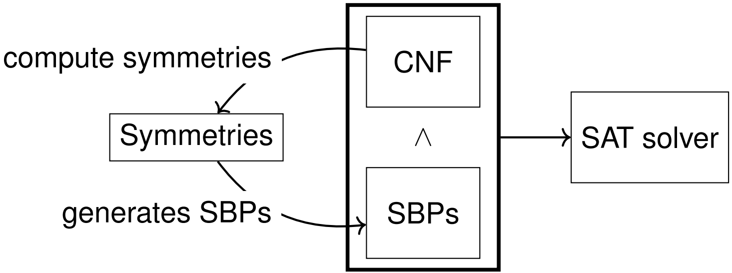
Static symmetry breaking



Different approaches:

- Shatter [ASM06]
- BreakID [DBBD16]
- ...

Static symmetry breaking



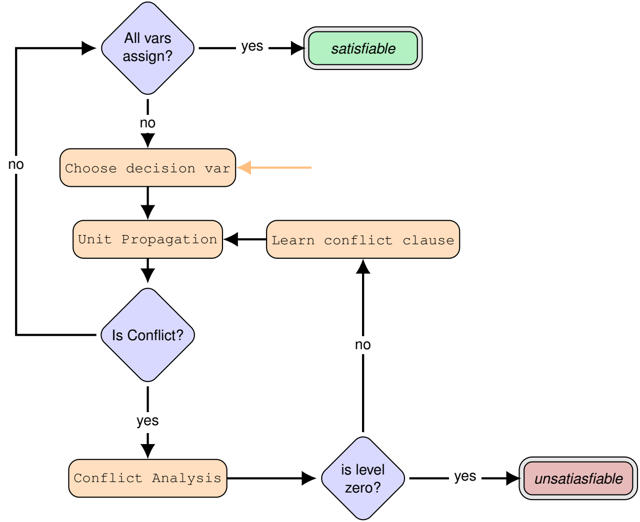
Different approaches:

- Shatter [ASM06]
- BreakID [DBBD16]
- ...

The solver can ”explode” instead of being helped

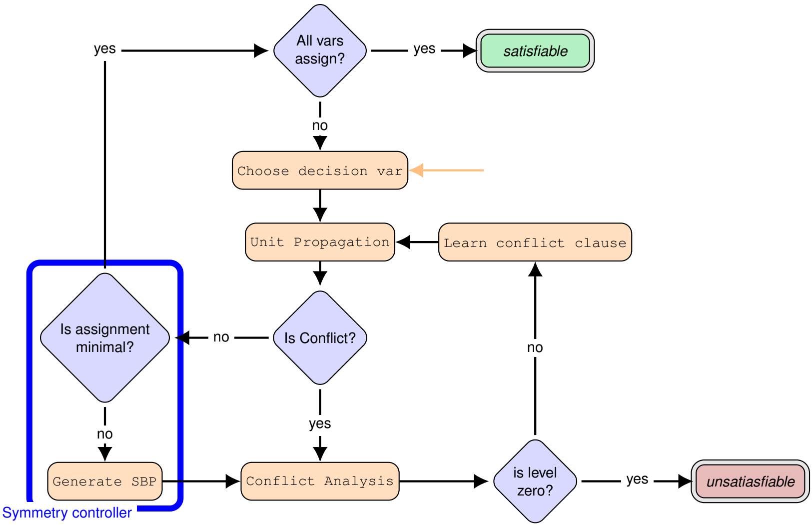
Our contribution CDCL[Sym] [MBCK18]

Compute and inject SBP opportunistically, during the solving



Our contribution CDCL[Sym] [MBCK18]

Compute and inject SBP opportunistically, during the solving



Symmetry status

- `reducer`: $g.\alpha \prec \alpha$
- `inactive` : $\alpha \prec g.\alpha$
- `active` : *not enough information*

Efficient implementation of symmetry status

Keep track the smallest unassigned variable x :

1. $\alpha(g.x) \leq \alpha(x)$, then g is `reducer` $\Rightarrow$ Effective SBP (ESBP)
2. $\alpha(x) \leq \alpha(g.x)$, then g is `inactive` $\Rightarrow$ g cannot reduce α
3. $\alpha(g.x)$ or $\alpha(x)$ is unassigned then g is `active`

Update whenever variables are assigned / unassigned

Example

$$g_1 = (x_1 \;\; x_2)(x_4 \;\; x_5)(x_7 \;\; x_8)$$
$$g_2 = (x_2 \;\; x_3)(x_5 \;\; x_6)(x_8 \;\; x_9)$$

$$\alpha = \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_8 \quad x_9$$

Example

1. `reducer`: $\alpha(g.x) \leq \alpha(x)$
2. `inactive`: $\alpha(x) \leq \alpha(g.x)$
3. `active`: $\alpha(g.x)$ or $\alpha(x)$ is unassigned

$$x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6 \leq x_7 \leq x_8 \; ; \; \mathtt{F} < \mathtt{T}$$

$$g_1 = \begin{array}{cccccc|cc} (x_2 & x_3) & (x_5 & x_6) & (x_8 & x_9) & x = x_2 & g.x = x_3 \\ \uparrow & & & & & & & \mathtt{active} \end{array}$$

$$g_2 = \begin{array}{cccccc|cc} (x_1 & x_2) & (x_4 & x_5) & (x_7 & x_8) & x = x_1 & g.x = x_2 \\ \uparrow & & & & & & & \mathtt{active} \end{array}$$

$$\dots$$

$$\alpha = \{ \qquad\qquad\qquad \}$$

Example

1. `reducer`: $\alpha(g.x) \leq \alpha(x)$
2. `inactive`: $\alpha(x) \leq \alpha(g.x)$
3. `active`: $\alpha(g.x)$ or $\alpha(x)$ is unassigned

$$x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6 \leq x_7 \leq x_8 \;\; ; \;\; \mathtt{F} < \mathtt{T}$$

$$g_1 = \;\; (\underset{\uparrow}{x_2} \quad x_3) \quad (x_5 \quad x_6) \quad (x_8 \quad x_9) \;\Big|\; x = x_2 \quad \begin{array}{l} g.x = x_3 \\ \mathtt{active} \end{array}$$

$$g_2 = \;\; (\underset{\uparrow}{x_1} \quad x_2) \quad (x_4 \quad x_5) \quad (x_7 \quad x_8) \;\Big|\; x = x_1 \quad \begin{array}{l} g.x = x_2 \\ \mathtt{active} \end{array}$$

$$\ldots$$

$$\alpha = \{\neg x_2 \qquad\qquad \}$$

Example

1. `reducer`: $\alpha(g.x) \leq \alpha(x)$
2. `inactive`: $\alpha(x) \leq \alpha(g.x)$
3. `active`: $\alpha(g.x)$ or $\alpha(x)$ is unassigned

$$x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6 \leq x_7 \leq x_8 \;\; ; \;\; \mathtt{F} < \mathtt{T}$$

$$\begin{array}{llllllll|ll} g_1 = & (x_2 & x_3) & (x_5 & x_6) & (x_8 & x_9) & & x = x_5 & g.x = x_6 \\ & & & \uparrow & & & & & & \mathtt{active} \end{array}$$

$$\begin{array}{llllllll|ll} g_2 = & (x_1 & x_2) & (x_4 & x_5) & (x_7 & x_8) & & x = x_1 & g.x = x_2 \\ & \uparrow & & & & & & & & \mathtt{reducer} \end{array}$$

$$\dots$$

$$\alpha = \{\neg x_2, \neg x_3, x_1\}$$

Example

1. `reducer`: $\alpha(g.x) \leq \alpha(x)$
2. `inactive`: $\alpha(x) \leq \alpha(g.x)$
3. `active`: $\alpha(g.x)$ or $\alpha(x)$ is unassigned

$$x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6 \leq x_7 \leq x_8 \;\; ; \;\; \mathtt{F} < \mathtt{T}$$

$$\begin{array}{lcccccc|ll} g_1 = & (x_2 & x_3) & (x_5 & x_6) & (x_8 & x_9) & x = x_5 & g.x = x_6 \\ & & & \uparrow & & & & & \mathtt{active} \end{array}$$

$$\begin{array}{lcccccc|ll} g_2 = & (x_1 & x_2) & (x_4 & x_5) & (x_7 & x_8) & x = x_1 & g.x = x_2 \\ & \uparrow & & & & & & & \mathtt{reducer} \end{array}$$

$$\ldots$$

$$\alpha = \{\neg x_2, \neg x_3, x_1\}$$

g_2 generates $\omega = \{\neg x_1, x_2\}$

CDCL[Sym] Implementation

- Packaged as a library **cosy**[1]
- Lightweight
- Fast Update and low memory
- Follows symmetry status

- Works with any enumerative SAT solver

[1] `https://github.com/lip6/cosy`

Experiments

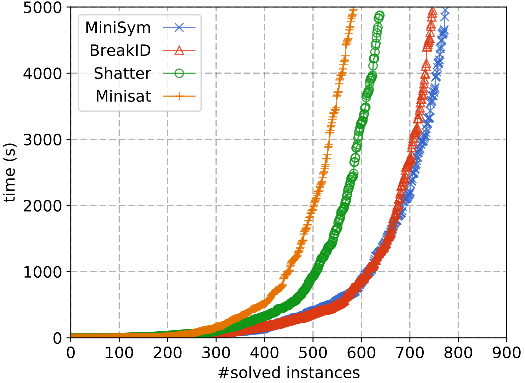
Benchmark:

- from SAT contests 2012 – 2017
- filter: `bliss` finds significant symmetries in 1000s
- 36 % of instances, $1\,350/3\,700$

Setup:

- four tools
  - MiniSat (no symmetry, baseline)
  - MiniSat + BreakID (SOTA SAT solver using symmetries)
  - MiniSat + Shatter (SOTA SAT solver using symmetries)
  - **MiniSym** = MiniSat + CDCL[Sym] (our approach)
- 5000s timeout, 8GB memory
- includes time to compute symmetries (except for MiniSat)

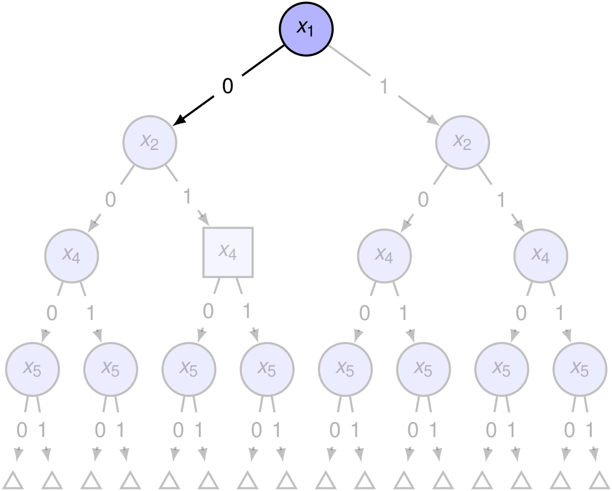
Experimental results



Solver	PAR-2	ALL	SAT	UNSAT
MiniSAT	2243h	586	325	261
Shatter	2088h	640	316	324
BreakID	1790h	749	334	415
MiniSym	**1735h**	**775**	**336**	**439**

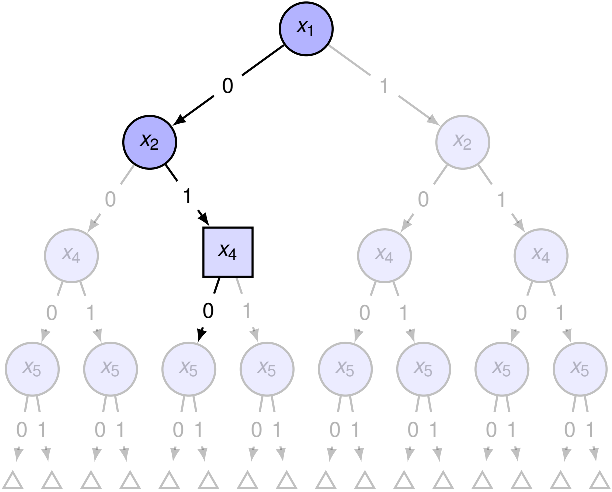
Using symmetries to accelerate the tree traversal

$g = (x_1 \;\; x_2)(x_4 \;\; x_5)(x_7 \;\; x_8)$



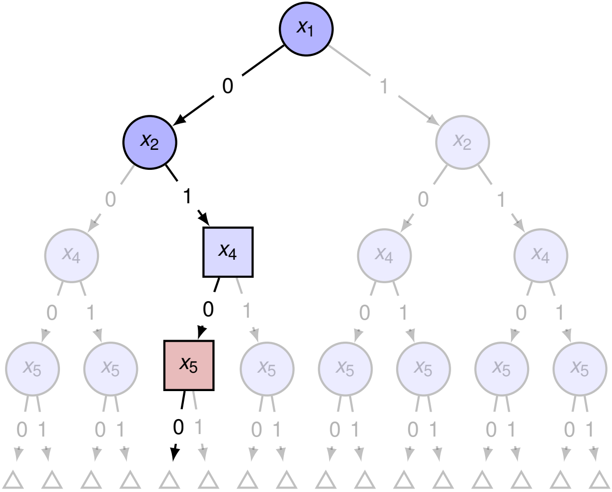
Using symmetries to accelerate the tree traversal

$g = (x_1 \;\; x_2)(x_4 \;\; x_5)(x_7 \;\; x_8)$



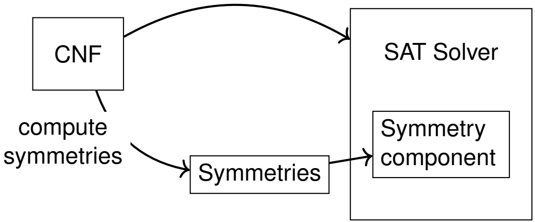
Using symmetries to accelerate the tree traversal

$g = (x_1 \;\; x_2)(x_4 \;\; x_5)(x_7 \;\; x_8)$



Use symmetries to deduce symmetrical facts.

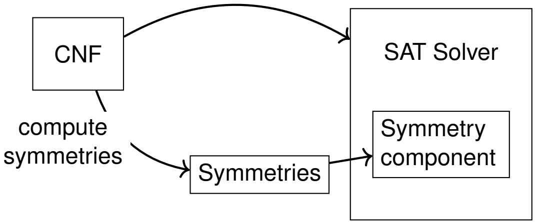
Dynamic Symmetry Breaking



Different approaches:

- Symmchaff [Sab05]
- Symmetry Propagation (SP) [DBdC$^+$12]
- Symmetry Learning Scheme (SLS) [BNOS10]
- Symmetry Explanation Leaning (SEL) [DBB17]
- ...

Dynamic Symmetry Breaking



Different approaches:

- Symmchaff [Sab05]
- Symmetry Propagation (SP) [DBdC$^+$12]
- Symmetry Learning Scheme (SLS) [BNOS10]
- Symmetry Explanation Leaning (SEL) [DBB17]
- ...

Cannot handle some instances solved by static approach

ESBP + SP [MBK19]

Compose the symmetry propagation and the ESBP

prune the decision tree while accelerating its traversal

Problems:

- ESBP breaks symmetries (incrementally)
- SP considers the manipulated symmetries valid all time

In a hybrid approach, SP must be able to identify **valid symmetries**

Local symmetries

$$\text{Formula} \leftarrow \text{(Symmetries)} \qquad \begin{aligned} \omega_1 &\leftarrow \text{(Local symmetries)} \\ \omega_2 &\leftarrow \text{(Local symmetries)} \\ \omega_3 &\leftarrow \text{(Local symmetries)} \\ \omega_4 &\leftarrow \text{(Local symmetries)} \end{aligned}$$

Macro level $\rightarrow$ Micro level

Local symmetries

$$\omega_1 \leftarrow \text{(Local symmetries)}$$
$$\omega_2 \leftarrow \text{(Local symmetries)}$$
$$\text{Formula} \leftarrow \text{(Symmetries)} \qquad \omega_3 \leftarrow \text{(Local symmetries)}$$
$$\omega_4 \leftarrow \text{(Local symmetries)}$$
$$\omega_5$$

Macro level $\rightarrow$ Micro level

Local symmetries

$$\text{Formula} \leftarrow (\text{Symmetries}) \qquad \begin{aligned} \omega_1 &\leftarrow (\text{Local symmetries}) \\ \omega_2 &\leftarrow (\text{Local symmetries}) \\ \omega_3 &\leftarrow (\text{Local symmetries}) \\ \omega_4 &\leftarrow (\text{Local symmetries}) \\ \omega_5 &\leftarrow (\text{Local symmetries}) \end{aligned}$$

Macro level $\rightarrow$ Micro level

Compute valid local symmetries on-the-fly at a minimal cost.

Local symmetries

$\omega_1 \leftarrow$ (Local symmetries)

$\omega_2 \leftarrow$ (Local symmetries)

Formula $\leftarrow$ (Symmetries)

$\omega_3 \leftarrow$ (Local symmetries)

$\omega_4 \leftarrow$ (Local symmetries)

ω_5

Macro level $\rightarrow$ Micro level

Compute valid local symmetries on-the-fly at a minimal cost.

- Inductive construction of the valid symmetries
- During the solving
- At a minimal cost

Experimental results

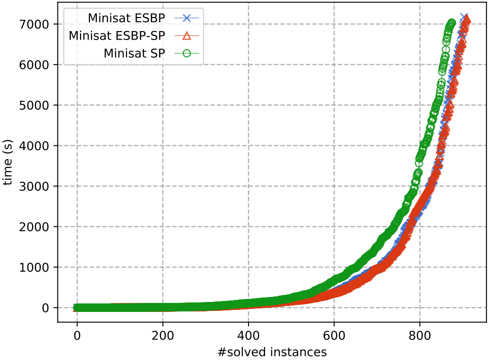
Benchmark:

- from SAT contests 2012 – 2018
- retain only instances for which `bliss` finds significant symmetries in 1000s
- 1400 symmetric instances (out of 4000)

Setup:

- three tools
  - MiniSat SP (Minisat with Symmetry Propagation)
  - MiniSat ESBP (Minisat with CDCL[Sym])
  - **Minisat ESBP-SP** (our approach)
- 7200s timeout

Experimental results



Solver	PAR-2	ALL	SAT	UNSAT
SP	1674h00	876	406	470
ESBP	1578h30	904	416	488
ESBP-SP	**1570h15**	**911**	**420**	**491**

Conclusion

- A new dynamic symmetry breaking approach
  - Generation of SBP on the fly
  - Package as a library cosy usable with any CDCL solver
  - Overcomes drawbacks of the existing approaches

- A new hybrid approach (ESBP-SP)
  - Take advantage of static and dynamic approach
  - Introduce local symmetries

Perspectives

- Combination of CDCL[Sym] with other dynamic symmetry breaking approach

- Exploitation of partial symmetries

Perspectives

- Combination of CDCL[Sym] with other dynamic symmetry breaking approach

- Exploitation of partial symmetries

Thanks !

Fadi A. Aloul, Karem A. Sakallah, and Igor L. Markov.
Efficient symmetry breaking for boolean satisfiability.
IEEE Trans. Computers, 55(5):549–558, 2006.

Armin Biere, Alessandro Cimatti, Edmund Clarke, and Yunshan Zhu.
Symbolic model checking without bdds.
Tools and Algorithms for the Construction and Analysis of Systems, pages 193–207, 1999.

Belaid Benhamou, Tarek Nabhani, Richard Ostrowski, and Mohamed Reda Saidi.
Enhancing clause learning by symmetry in sat solvers.
In *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, volume 1, pages 329–335. IEEE, 2010.

Stephen A Cook.
The complexity of theorem-proving procedures.
In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

Jo Devriendt, Bart Bogaerts, and Maurice Bruynooghe.
Symmetric explanation learning: Effective dynamic symmetry handling for sat.
In *International Conference on Theory and Applications of Satisfiability Testing*, pages 83–100. Springer, 2017.

Jo Devriendt, Bart Bogaerts, Maurice Bruynooghe, and Marc Denecker.
Improved static symmetry breaking for sat.
In *International Conference on Theory and Applications of Satisfiability Testing*, pages 104–122. Springer, 2016.

Jo Devriendt, Bart Bogaerts, Broes de Cat, Marc Denecker, and Christopher Mears.
Symmetry propagation: Improved dynamic symmetry breaking in SAT.
In *IEEE 24th International Conference on Tools with Artificial Intelligence, ICTAI 2012, Athens, Greece, November 7-9, 2012*, pages 49–56, 2012.

Martin Davis, George Logemann, and Donald Loveland.
A machine program for theorem-proving.
Commun. ACM, 5(7):394–397, July 1962.

Henry A Kautz, Bart Selman, et al.
Planning as satisfiability.
In *ECAI*, volume 92, pages 359–363, 1992.

Inês Lynce and Joao Marques-Silva.
Sat in bioinformatics: Making the case with haplotype inference.
In *International Conference on Theory and Applications of Satisfiability Testing*, pages 136–141. Springer, 2006.

Hakan Metin, Souheib Baarir, Maximilien Colange, and Fabrice Kordon.
Cdclsym: Introducing effective symmetry breaking in sat solving.
In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 99–114. Springer, 2018.

Hakan Metin, Souheib Baarir, and Fabrice Kordon.
Composing symmetry propagation and effective symmetry breaking for sat solving.
In *NASA Formal Methods Symposium*, pages 316–332. Springer, 2019.

Fabio Massacci and Laura Marraro.
Logical cryptanalysis as a sat problem.
Journal of Automated Reasoning, 24(1):165–203, 2000.

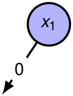
Joao P Marques-Silva and Karem A Sakallah.
Grasp: A search algorithm for propositional satisfiability.
IEEE Transactions on Computers, 48(5):506–521, 1999.

Ashish Sabharwal.
Symchaff: A structure-aware satisfiability solver.
In *AAAI*, volume 5, pages 467–474, 2005.

CDCL in action TODO

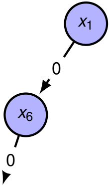
$$
\begin{aligned}
\omega_1 &= \{x_1, x_2, x_3\} \\
\omega_2 &= \{x_4, x_5, x_6\} \\
\omega_3 &= \{\neg x_1, \neg x_5\} \\
\omega_4 &= \{\neg x_2, \neg x_4\} \\
\omega_5 &= \{\neg x_3, \neg x_4\} \\
\omega_6 &= \{\neg x_3, \neg x_6\}
\end{aligned}
$$

CDCL in action TODO



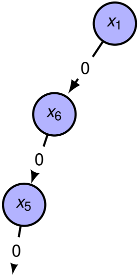
$$\omega_1 = \{x_1, x_2, x_3\}$$
$$\omega_2 = \{x_4, x_5, x_6\}$$
$$\omega_3 = \{\neg x_1, \neg x_5\}$$
$$\omega_4 = \{\neg x_2, \neg x_4\}$$
$$\omega_5 = \{\neg x_3, \neg x_4\}$$
$$\omega_6 = \{\neg x_3, \neg x_6\}$$

CDCL in action TODO



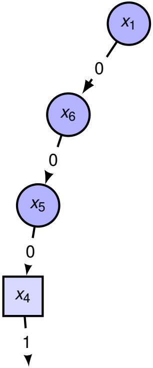
$$\omega_1 = \{x_1, x_2, x_3\}$$
$$\omega_2 = \{x_4, x_5, x_6\}$$
$$\omega_3 = \{\neg x_1, \neg x_5\}$$
$$\omega_4 = \{\neg x_2, \neg x_4\}$$
$$\omega_5 = \{\neg x_3, \neg x_4\}$$
$$\omega_6 = \{\neg x_3, \neg x_6\}$$

CDCL in action TODO



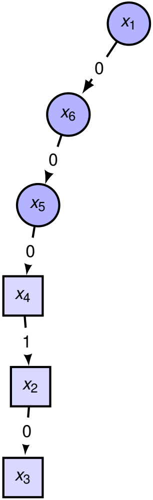
$$\omega_1 = \{x_1, x_2, x_3\}$$
$$\omega_2 = \{x_4, x_5, x_6\}$$
$$\omega_3 = \{\neg x_1, \neg x_5\}$$
$$\omega_4 = \{\neg x_2, \neg x_4\}$$
$$\omega_5 = \{\neg x_3, \neg x_4\}$$
$$\omega_6 = \{\neg x_3, \neg x_6\}$$

CDCL in action TODO



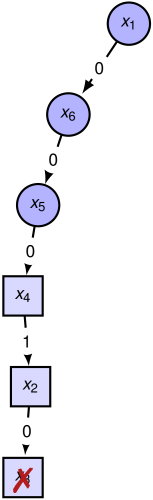
$$\omega_1 = \{x_1, x_2, x_3\}$$
$$\omega_2 = \{x_4, x_5, x_6\}$$
$$\omega_3 = \{\neg x_1, \neg x_5\}$$
$$\omega_4 = \{\neg x_2, \neg x_4\}$$
$$\omega_5 = \{\neg x_3, \neg x_4\}$$
$$\omega_6 = \{\neg x_3, \neg x_6\}$$

CDCL in action TODO



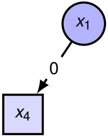
$$\omega_1 = \{x_1, x_2, x_3\}$$
$$\omega_2 = \{x_4, x_5, x_6\}$$
$$\omega_3 = \{\neg x_1, \neg x_5\}$$
$$\omega_4 = \{\neg x_2, \neg x_4\}$$
$$\omega_5 = \{\neg x_3, \neg x_4\}$$
$$\omega_6 = \{\neg x_3, \neg x_6\}$$

CDCL in action TODO



$$
\begin{aligned}
\omega_1 &= \{x_1, x_2, x_3\} \\
\omega_2 &= \{x_4, x_5, x_6\} \\
\omega_3 &= \{\neg x_1, \neg x_5\} \\
\omega_4 &= \{\neg x_2, \neg x_4\} \\
\omega_5 &= \{\neg x_3, \neg x_4\} \\
\omega_6 &= \{\neg x_3, \neg x_6\}
\end{aligned}
$$

CDCL in action TODO



$$\omega_1 = \{x_1, x_2, x_3\}$$
$$\omega_2 = \{x_4, x_5, x_6\}$$
$$\omega_3 = \{\neg x_1, \neg x_5\}$$
$$\omega_4 = \{\neg x_2, \neg x_4\}$$
$$\omega_5 = \{\neg x_3, \neg x_4\}$$
$$\omega_6 = \{\neg x_3, \neg x_6\}$$

$$\omega_7 = \{x_1, \neg x_4\}$$

Weakly active symmetries

Logical consequence

When ω is satisfied in all satisfying assignments of φ, we say that ω is a logical consequence of φ, and we denote this by $\varphi \vdash \omega$.

Weakly active symmetries

Logical consequence

When ω is satisfied in all satisfying assignments of φ, we say that ω is a logical consequence of φ, and we denote this by $\varphi \vdash \omega$.

Weakly active symmetries

Let a subset $\delta \subseteq \alpha$, a symmetry σ of φ such that $\varphi \cup \delta \vdash \varphi \cup \alpha \wedge \sigma.\delta \subseteq \alpha$ then σ is weakly active symmetry.

Weakly active symmetries

Logical consequence

When ω is satisfied in all satisfying assignments of φ, we say that ω is a logical consequence of φ, and we denote this by $\varphi \vdash \omega$.

Weakly active symmetries

Let a subset $\delta \subseteq \alpha$, a symmetry σ of φ such that $\varphi \cup \delta \vdash \varphi \cup \alpha \wedge \sigma.\delta \subseteq \alpha$ then σ is weakly active symmetry.

Symmetry propagation

Let σ a weakly active symmetry, then

$$\varphi \cup \alpha \vdash \{l\} \Leftrightarrow \varphi \cup \alpha \vdash \sigma.\{l\}$$

Local symmetries

Logical consequence

When ω is satisfied in all satisfying assignments of φ, we say that ω is a logical consequence of φ, and we denote this by $\varphi \vdash \omega$.

Local Symmetries

Let φ be a formula. We define $L_{\omega,\varphi}$, the set of *local symmetries* for a clause ω, and with respect to a formula φ, as follows:

$$L_{\omega,\varphi} = \{\sigma \in \mathfrak{S} \mid \varphi \vdash \sigma.\omega\}$$

Local symmetries

Logical consequence

When ω is satisfied in all satisfying assignments of φ, we say that ω is a logical consequence of φ, and we denote this by $\varphi \vdash \omega$.

Local Symmetries

Let φ be a formula. We define $L_{\omega,\varphi}$, the set of *local symmetries* for a clause ω, and with respect to a formula φ, as follows:

$$L_{\omega,\varphi} = \{\sigma \in \mathfrak{S} \mid \varphi \vdash \sigma.\omega\}$$

We can state that:

$$\bigcap_{\omega \in \varphi} L_{\omega,\varphi} \subseteq G.$$

Computing local symmetries

Formula can be decomposed as : $\varphi = \varphi_o \cup \varphi_e \cup \varphi_d$ where

- φ_o is the set of the original clauses
- φ_e is the set of ESBPs
- φ_d is the set of deduced clauses.

Local symmetries

- $\omega \in \varphi_o, L_{\omega,\varphi} \supseteq G$
- $\omega \in \varphi_e, L_{\omega,\varphi} \supseteq Stab(\omega) = \{\sigma \in G \mid \omega = \sigma.\omega\}$
- $\omega \in \varphi_d, L_{\omega,\varphi} \supseteq (\bigcap_{\omega' \in \varphi_1} L_{\omega',\varphi}) \cup Stab(\omega)$

  where φ_1 is the set of clauses that derives ω.