Exploitation des symétries dynamiques pour la résolution des problèmes SAT

Thèse de doctorat de Sorbonne Université

Hakan METIN

Jury Members:

PASCAL FONTAINE	Maître de conférences, Université de Liège
LAURE PETRUCCI	Professeur, Université Paris 13
JEAN-MICHEL COUVREUR	Professeur, Université d'Orléans
EMANUELLE ENCRENAZ	Maître de conférences, Sorbonne Université
SOUHEIB BAARIR	Maître de conférences, Université Paris Nanterre
FABRICE KORDON	Professeur, Sorbonne Université

Supervisors:

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

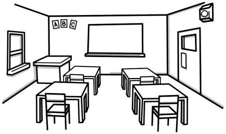
Motivation

SAT is widely used in different domains:

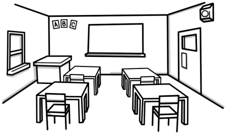
- Artificial intelligence (planning, games, ...)
- Bioinformatics (haplotype inference, ...)
- Security (cryptanalysis, inversion attack on hash function)
- Computationally hard problems (graph coloring, ...)
- Formal Methods (hardware model checking, ...)

Outline

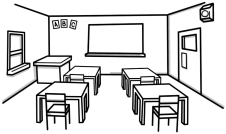
SAT an example



1

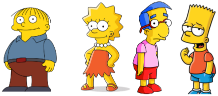


2



3

A

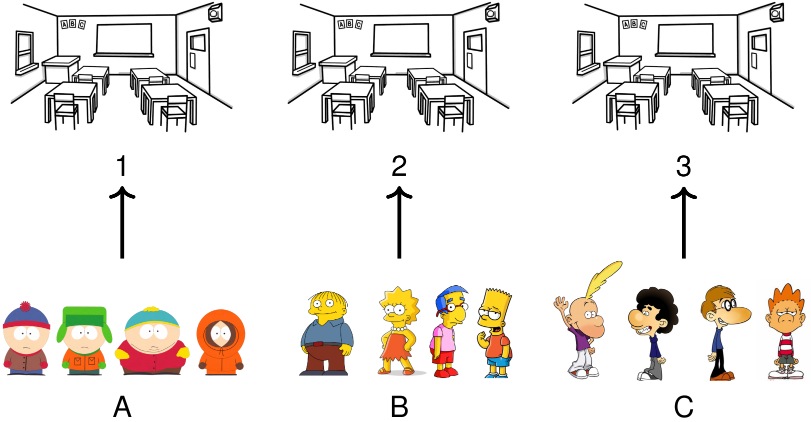


B

C

Is it possible to attribute each group to a classroom?

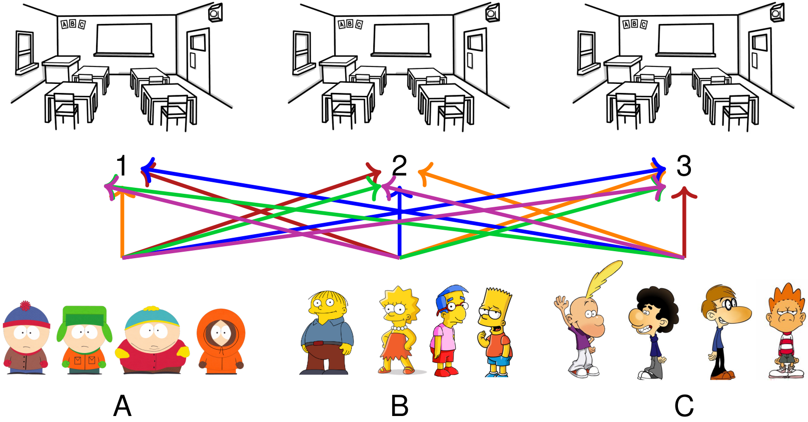
SAT an example



Is it possible to attribute each group to a classroom?

YES!

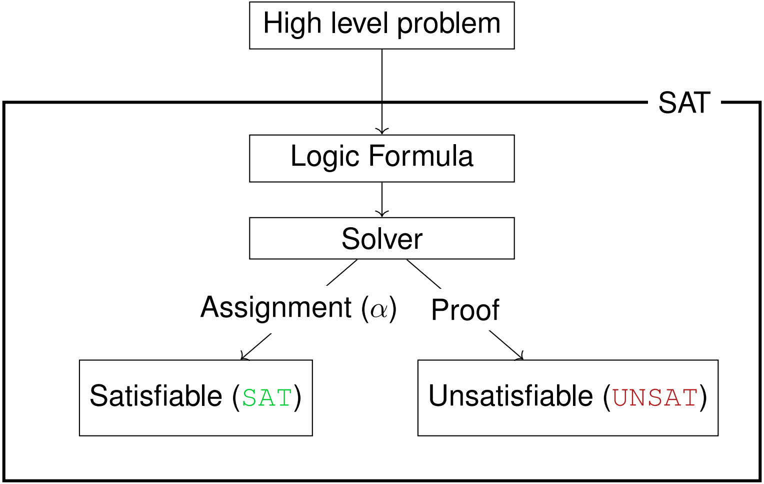
SAT an example



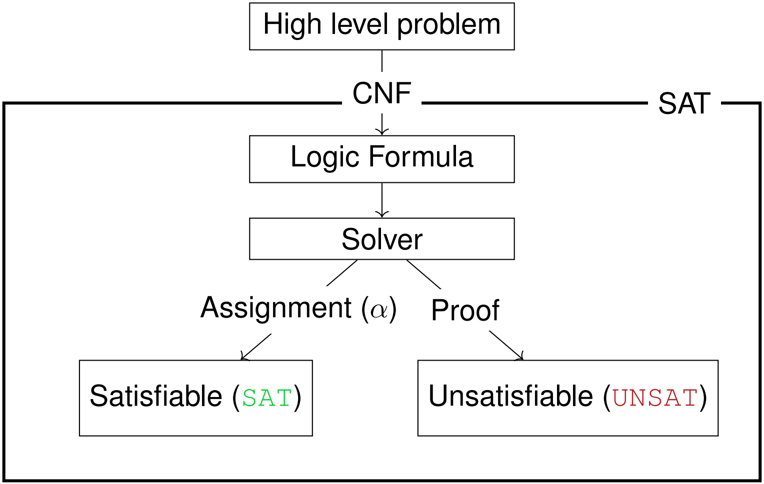
Is it possible to attribute each group to a classroom?

YES! Many solutions

SAT


High level problem
SAT
Logic Formula
Solver
Assignment (α)
Proof
Satisfiable (SAT)
Unsatisfiable (UNSAT)

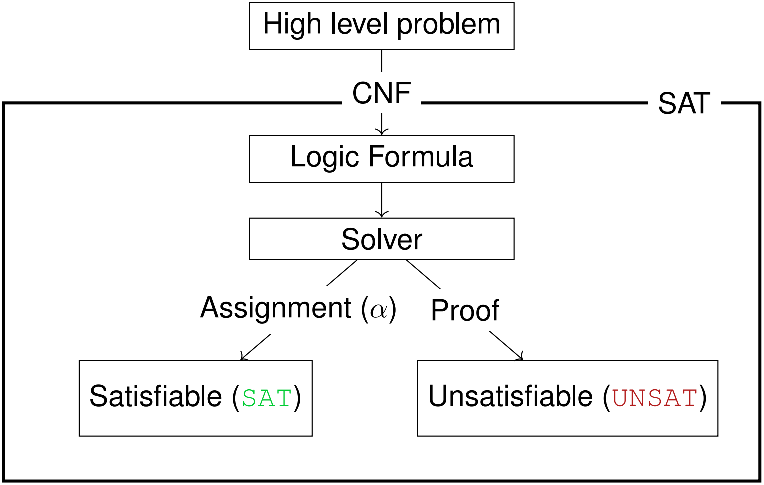
SAT



CNF Representation:

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

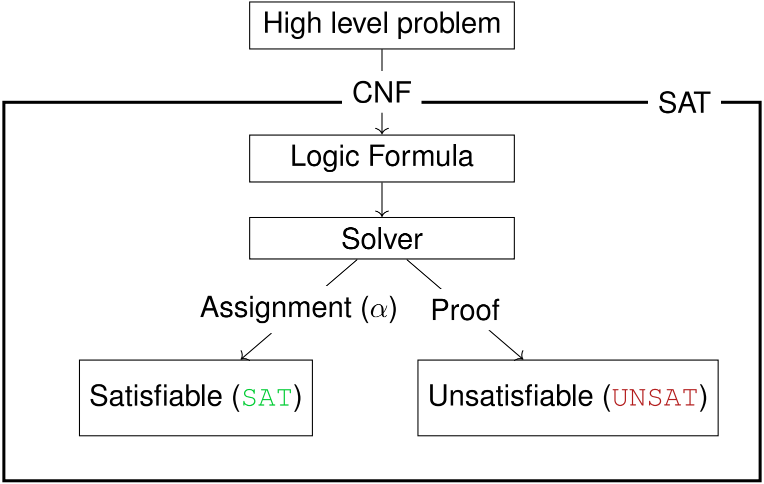
SAT



CNF Representation:

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

SAT



CNF Representation:

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

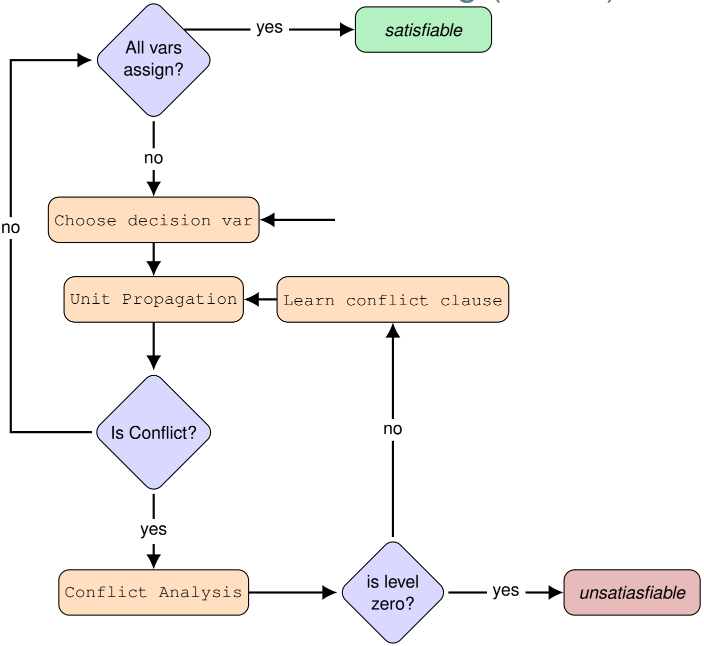
SAT Solving

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

Enumerative Algorithm:

- Davis, Putnam, Logemann, and Loveland (DPLL) [DLL62]
  - Boolean Constraint Propagation (BCP)
- Conflict Driven Clause Learning (CDCL) [MSS99]
  - derived from DPLL
  - clause learning

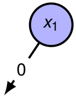
Conflict Driven Clause Learning (CDCL)



CDCL in action TODO

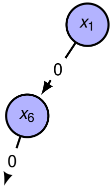
$$\omega_1 = \{x_1, x_2, x_3\}$$
$$\omega_2 = \{x_4, x_5, x_6\}$$
$$\omega_3 = \{\neg x_1, \neg x_5\}$$
$$\omega_4 = \{\neg x_2, \neg x_4\}$$
$$\omega_5 = \{\neg x_3, \neg x_4\}$$
$$\omega_6 = \{\neg x_3, \neg x_6\}$$

CDCL in action TODO



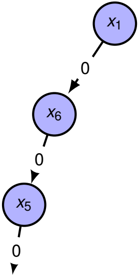
$$\omega_1 = \{x_1, x_2, x_3\}$$
$$\omega_2 = \{x_4, x_5, x_6\}$$
$$\omega_3 = \{\neg x_1, \neg x_5\}$$
$$\omega_4 = \{\neg x_2, \neg x_4\}$$
$$\omega_5 = \{\neg x_3, \neg x_4\}$$
$$\omega_6 = \{\neg x_3, \neg x_6\}$$

CDCL in action TODO



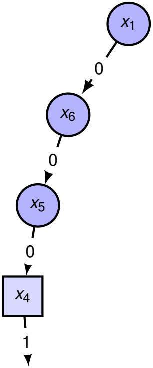
$$\omega_1 = \{x_1, x_2, x_3\}$$
$$\omega_2 = \{x_4, x_5, x_6\}$$
$$\omega_3 = \{\neg x_1, \neg x_5\}$$
$$\omega_4 = \{\neg x_2, \neg x_4\}$$
$$\omega_5 = \{\neg x_3, \neg x_4\}$$
$$\omega_6 = \{\neg x_3, \neg x_6\}$$

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



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

$$\omega_2 = \{x_4, x_5, x_6\}$$

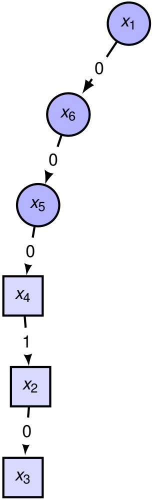
$$\omega_3 = \{\neg x_1, \neg x_5\}$$

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

$$\omega_5 = \{\neg x_3, \neg x_4\}$$

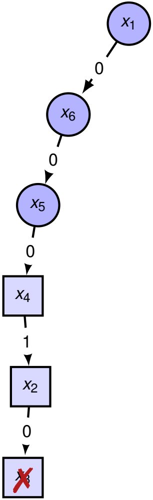
$$\omega_6 = \{\neg x_3, \neg x_6\}$$

CDCL in action TODO



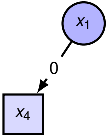
$$\omega_1 = \{x_1, x_2, x_3\}$$
$$\omega_2 = \{x_4, x_5, x_6\}$$
$$\omega_3 = \{\neg x_1, \neg x_5\}$$
$$\omega_4 = \{\neg x_2, \neg x_4\}$$
$$\omega_5 = \{\neg x_3, \neg x_4\}$$
$$\omega_6 = \{\neg x_3, \neg x_6\}$$

CDCL in action TODO



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

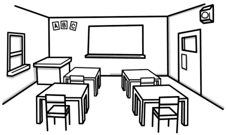
CDCL in action TODO



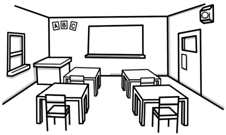
$$\omega_1 = \{x_1, x_2, x_3\}$$
$$\omega_2 = \{x_4, x_5, x_6\}$$
$$\omega_3 = \{\neg x_1, \neg x_5\}$$
$$\omega_4 = \{\neg x_2, \neg x_4\}$$
$$\omega_5 = \{\neg x_3, \neg x_4\}$$
$$\omega_6 = \{\neg x_3, \neg x_6\}$$

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

An UNSAT example

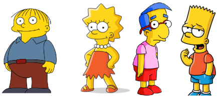


1

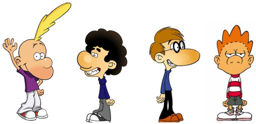


2

A



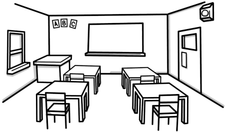
B



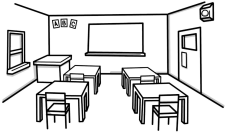
C

Is it possible to attribute each group to a classroom?

An UNSAT example

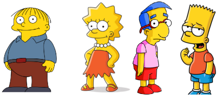


1

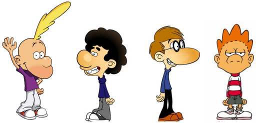


2

A



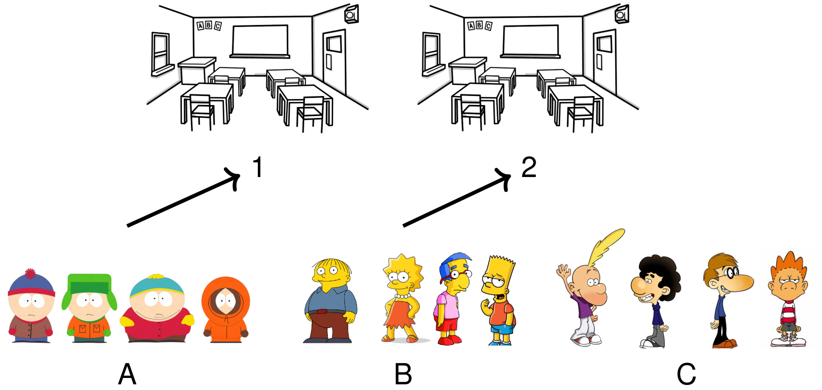
B



C

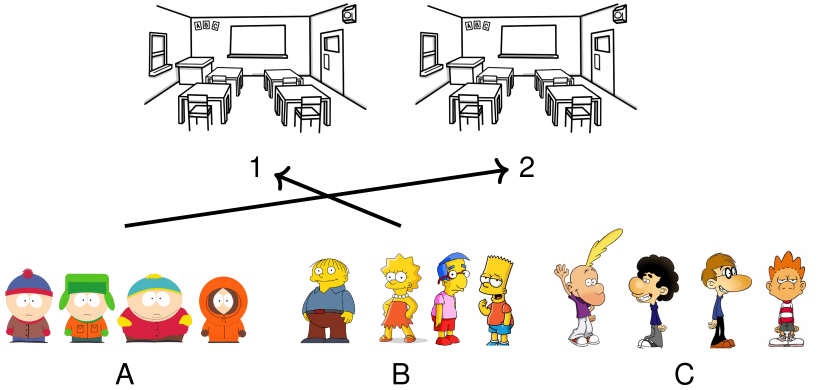
Is it possible to attribute each group to a classroom?

An UNSAT example



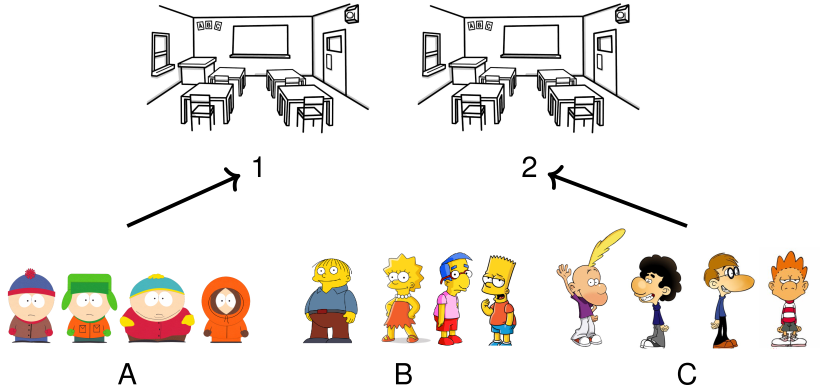
Is it possible to attribute each group to a classroom?

An UNSAT example



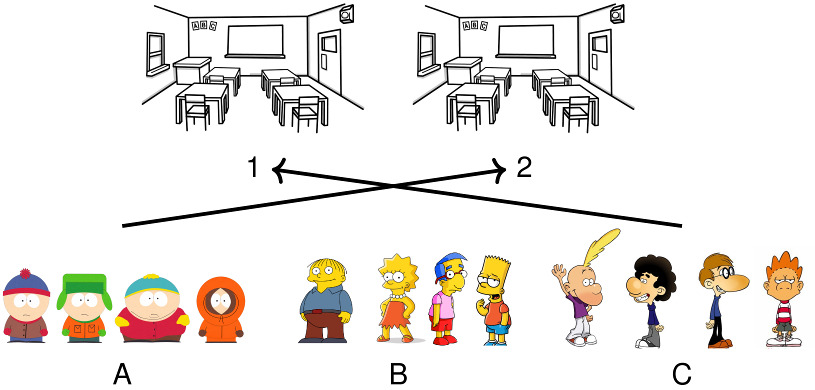
Is it possible to attribute each group to a classroom?

An UNSAT example



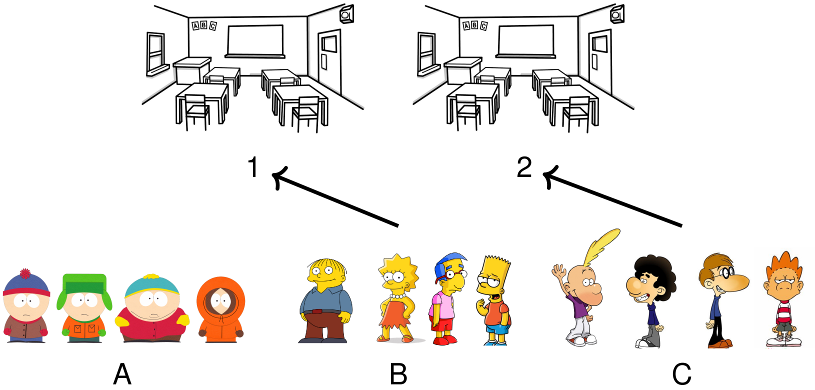
Is it possible to attribute each group to a classroom?

An UNSAT example



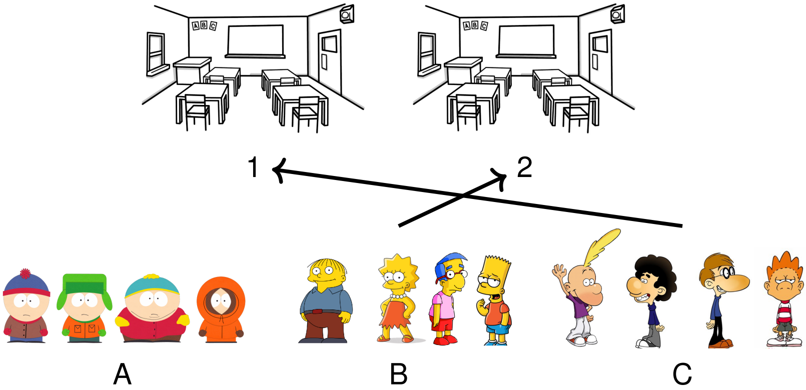
Is it possible to attribute each group to a classroom?

An UNSAT example



Is it possible to attribute each group to a classroom?

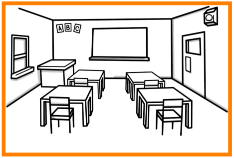
An UNSAT example



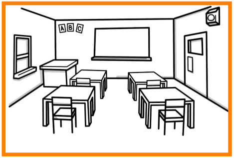
Is it possible to attribute each group to a classroom?

No!

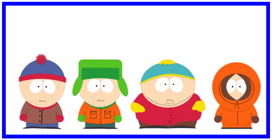
An UNSAT example



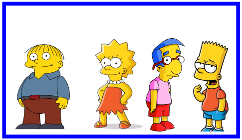
1



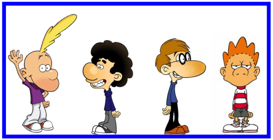
2



A



B



C

Is it possible to attribute each group to a classroom?

No!

Presence of symmetries hinders the performance of the solver

Outline

Symmetry

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

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

A symmetry (permuation) g is a bijective function (on variables) that leaves φ invariant.

Computing symmetries of a SAT problem

CNF formula

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

[1] http://www.tcs.hut.fi/Software/bliss/

[2] http://vlsicad.eecs.umich.edu/BK/SAUCY/

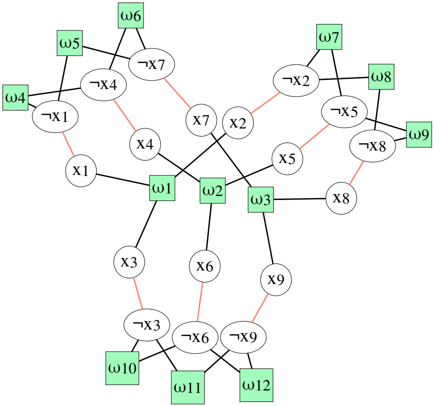
Computing symmetries of a SAT problem

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



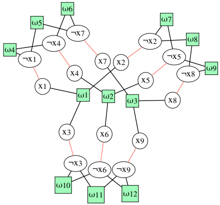
Computing symmetries of a SAT problem

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` [1] or `saucy` [2])

[1] http://www.tcs.hut.fi/Software/bliss/

[2] http://vlsicad.eecs.umich.edu/BK/SAUCY/

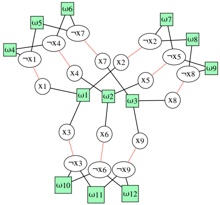
Computing symmetries of a SAT problem

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` [1] or `saucy` [2])

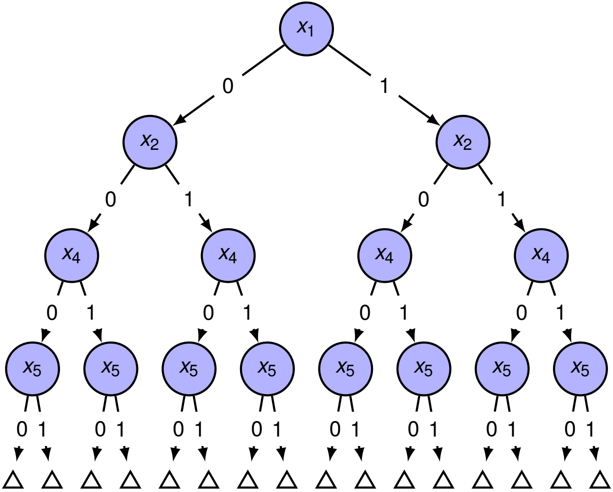
$\Downarrow$ $\Downarrow$

set of symmetries

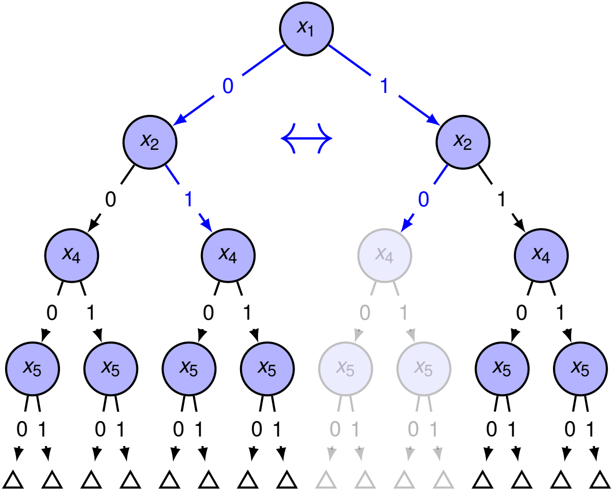
$$g_1 = (x_2\ \ x_3)(x_5\ \ x_6)(x_8\ \ x_9)$$
$$g_2 = (x_4\ \ x_7)(x_5\ \ x_8)(x_6\ \ x_9)$$
$$g_3 = (x_1\ \ x_2)(x_4\ \ x_5)(x_7\ \ x_8)$$
$$g_4 = (x_1\ \ x_4)(x_2\ \ x_5)(x_3\ \ x_6)$$

[1] http://www.tcs.hut.fi/Software/bliss/
[2] http://vlsicad.eecs.umich.edu/BK/SAUCY/

Using symmetries to prune search space



Using symmetries to prune search space



Adds additional constraints to prune search space.

Generates symmetry breaking predicates (SBP)

- Define lexicographic order
  - Define total order on variables
  - Define minimal value
- Forbid non minimal assignment with addition of SBP

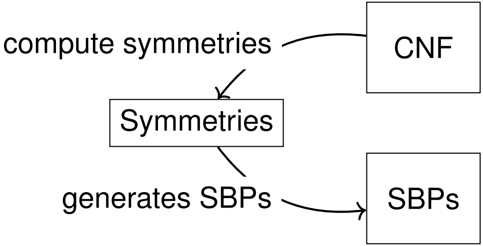
Example:

$$x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6 \leq x_7 \leq x_8; \texttt{false} < \texttt{true}$$

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

$x_1 \leq x_2$	$x_1 \vee \neg x_2$
$x_1 = x_2 \rightarrow x_4 \leq x_5$	$x_1 \vee x_2 \vee x_4 \vee \neg x_5$
	$\neg x_1 \vee \neg x_2 \vee x_4 \vee \neg x_5$
$x_1 = x_2 \wedge x_4 = x_5 \rightarrow x_8 \leq x_3$	$x_1 \vee x_2 \vee x_4 \vee x_5 \vee x_7 \vee \neg x_8$
	$\neg x_1 \vee \neg x_2 \vee x_4 \vee x_5 \vee x_7 \vee \neg x_8$
	$\ldots$

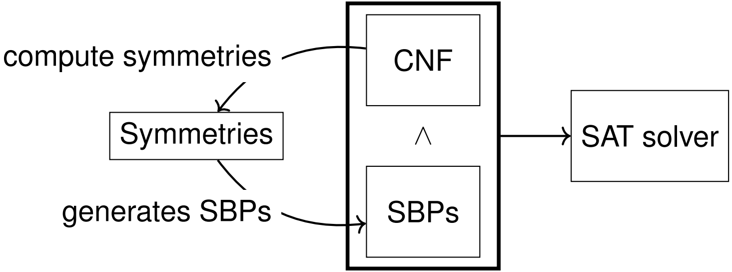
Static symmetry breaking



- Works well on many symmetric instances

- The solver can ”explode” instead of being helped

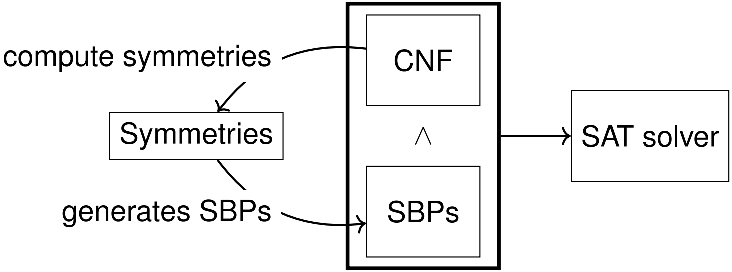
Static symmetry breaking



- Works well on many symmetric instances

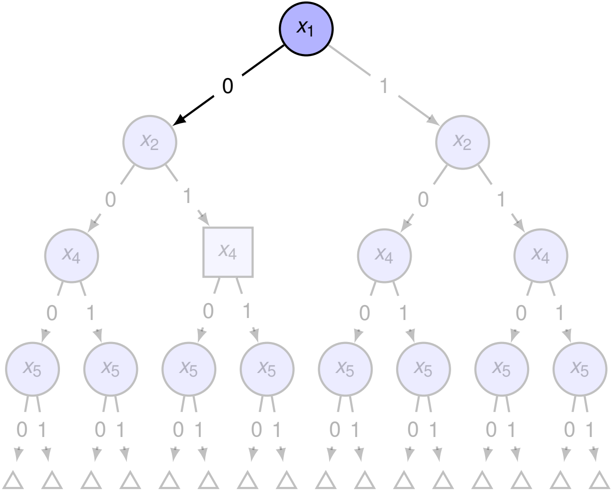
- The solver can ”explode” instead of being helped

Static symmetry breaking

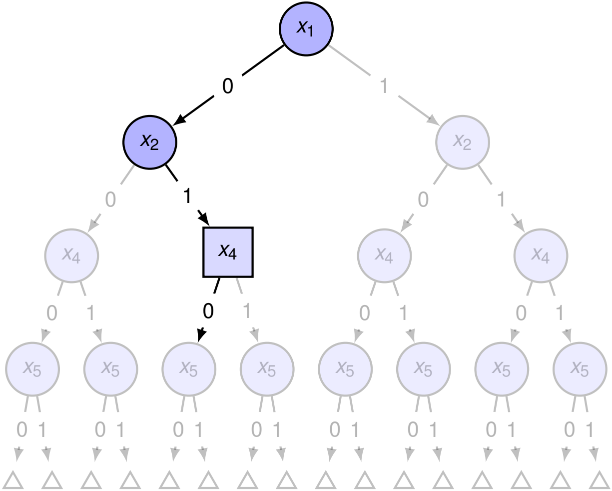


- Works well on many symmetric instances
- The solver can ”explode” instead of being helped

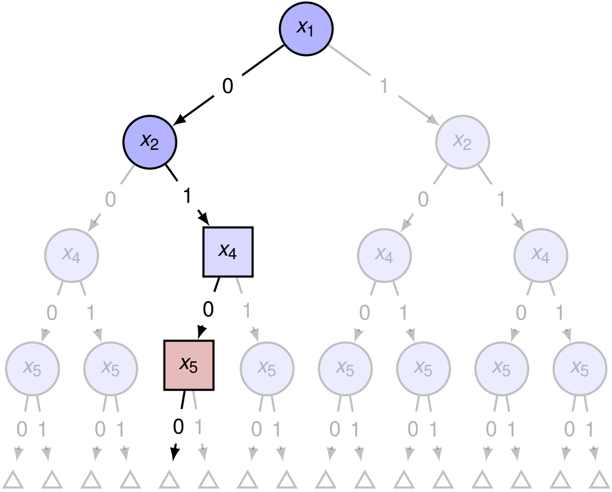
Using symmetries to accelerate tree traversal



Using symmetries to accelerate tree traversal



Using symmetries to accelerate tree traversal



Use symmetries to deduce symmetrical facts.

Dynamic Symmetry Breaking

- Accelerate SAT engine using symmetry properties
-

Modify solver behavior to accelerate tree traversal
modify solver
Different tools SP, SLS, SEL, ...

Symmetry Propagation

TODO
Present SP

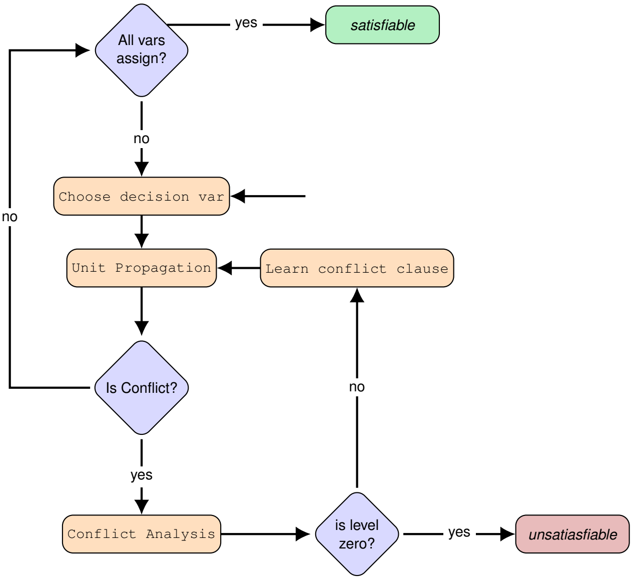
Example

TODO
Build an example

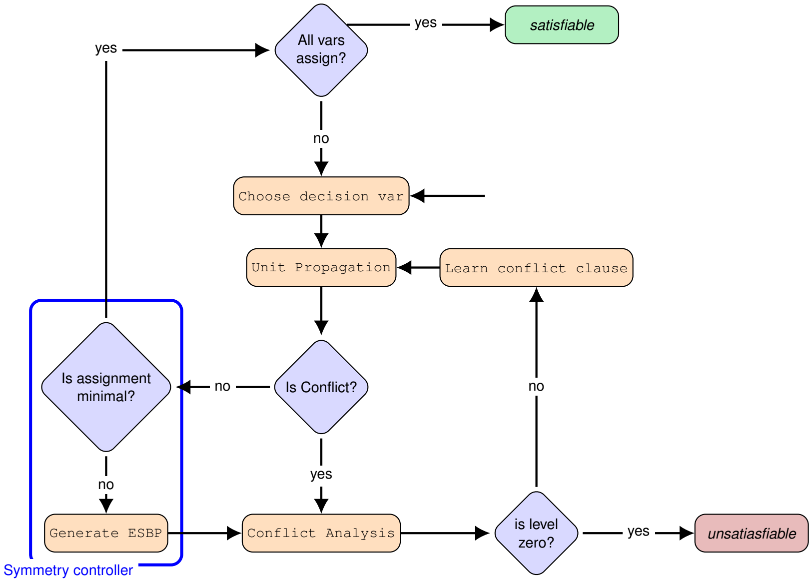
Outline

1. SAT overview
   - SAT basics
   - SAT and symmetries
2. Existing approaches
   - Static symmetry breaking
   - Dynamic symmetry breaking
3. Contribution and results
   - CDCL [Sym]
   - Combination of different approaches

Our contribution CDCL[Sym]



Our contribution CDCL[Sym]



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

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

$$x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6 \leq x_7 \leq x_8 \;;\; \texttt{false} < \texttt{true}$$

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

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

$$\ldots$$

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

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

$$x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6 \leq x_7 \leq x_8 \; ; \; \texttt{false} < \texttt{true}$$

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

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

$$\dots$$

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

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

$$x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6 \leq x_7 \leq x_8 \;\; ; \;\; \texttt{false} < \texttt{true}$$

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

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

$$\ldots$$

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

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

$$x_1 \leq x_2 \leq x_3 \leq x_4 \leq x_5 \leq x_6 \leq x_7 \leq x_8 \ ; \ \texttt{false} < \texttt{true}$$

$$g_1 = \quad (x_2 \quad x_3) \quad (\underset{\uparrow}{x_5} \quad x_6) \quad (x_8 \quad x_9) \quad \Big| \quad x = x_5 \quad g.x = x_6 \ \texttt{active}$$

$$g_2 = \quad (\underset{\uparrow}{x_1} \quad x_2) \quad (x_4 \quad x_5) \quad (x_7 \quad x_8) \quad \Big| \quad x = x_1 \quad g.x = x_2 \ \texttt{reducer}$$

$$\ldots$$

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

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

Experiments: benchmark

Benchmark:

- from SAT contests 2012 – 2017,
- retain only instances for which `bliss` finds significant symmetries in 1000s,
- 1350 symmetric instances (out of 3700)

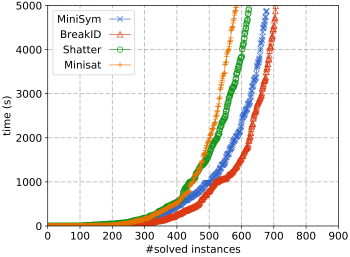
Setup:

- four tools
  - MiniSat (no symmetry, baseline)
  - MiniSat + breakID (state-of-the-art symmetry SAT solver)
  - MiniSat + Shatter (state-of-the-art symmetry SAT solver)
  - **MiniSym** = MiniSat + CDCLSym (our approach)
- 5000s timeout, 8GB memory,
- includes time to compute symmetries (except for MiniSat)

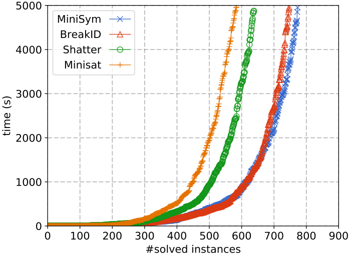
Experimental results

`bliss` gives more generators than `saucy3`

Figure: cactus plot total number of instances



(a) with `saucy3`



(b) with `bliss`

	MiniSAT	Shatter	BreakID	MiniSym
PAR-2 sum	8 074 348	7 770 434	**6 909 999**	7 229 700
PAR-2 avg	5 981	5 756	**5 119**	5 355

	MiniSAT	Shatter	BreakID	MiniSym
PAR-2 sum	8 074 348	7 517 556	6 444 954	**6 245 448**
PAR-2 avg	5 981	5 569	4 774	**4 626**

Table: time comparison

Experimental results (UNSAT versus SAT)

	MiniSAT	Shatter	BreakID	MiniSym
TOTAL (no dup)	261	302	**371**	345

(a) With saucy3

	MiniSAT	Shatter	BreakID	MiniSym
TOTAL (no dup)	261	324	415	**439**

(b) With bliss

Table: comparison on UNSAT instances

	MiniSAT	Shatter	BreakID	MiniSym
TOTAL (no dup)	325	323	**337**	335

(a) With saucy3

	MiniSAT	Shatter	BreakID	MiniSym
TOTAL (no dup)	325	316	334	**336**

(b) With bliss

Table: comparison on SAT instances

ESBP + SP

TODO
Symmetry propagation on top of ESBP
Compose both approaches
Is it possible?

Notion of local symmetries

TODO

Computation of local symmetries

TODO

Experimental results

TODO

Conclusion and Perspective

TODO
Conclusion:
Perspectives:

Thanks !

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.

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

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