

On Symmetries and Transformations

CP '26, Lissabon, Portugal

Daimy Van Caudenberg¹, Markus Anders², Bart Bogaerts^{1,3}

*KU Leuven*¹, *RPTU Kaiserslautern-Landau*², *Vrije Universiteit Brussel*³

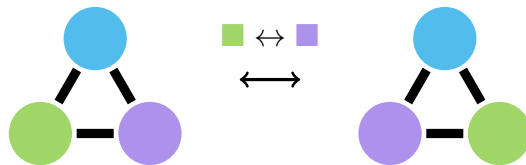
July 23, 2026



KU LEUVEN

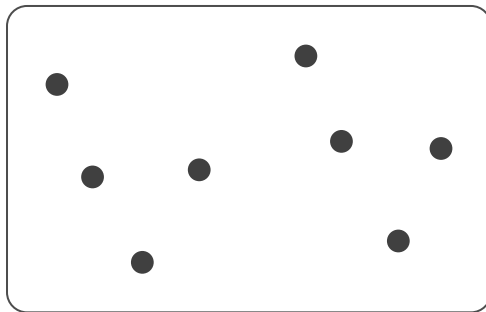


WHAT IS A SYMMETRY?



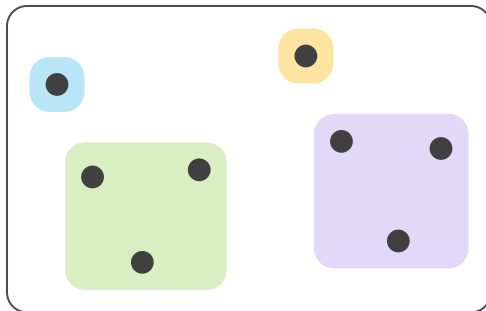
Intuition: Symmetries are permutations between **structurally equivalent** solutions

Solution Space



sols = candidate solutions

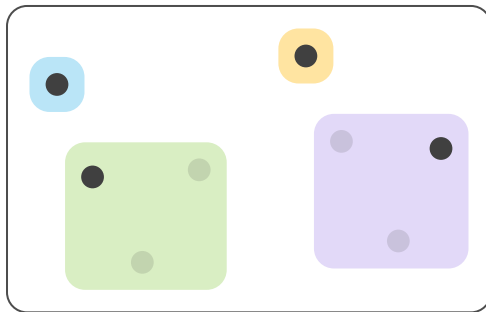
Solution Space



sols = candidate solutions

Coloured regions = equivalence classes

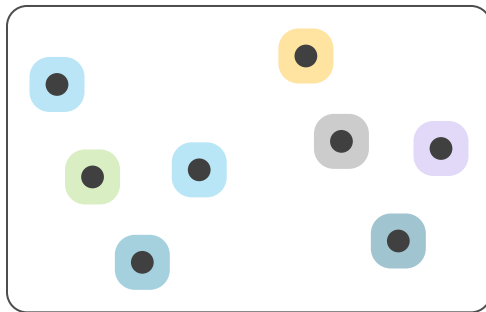
Solution Space



sols = candidate solutions

Coloured regions = equivalence classes

Solution Space



sols = candidate solutions

Coloured regions = equivalence classes

Some transformations break all symmetry...

Example: Structure-based extended resolution adds **auxiliary variables** [CS13]; shortens proofs, but can **destroy symmetries**...

Decompose a sum constraint using a prefix-sum decomposition:

Sum Constraint

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

All variables are interchangeable

TRANSFORMATIONS DESTROY SYMMETRY

Example: Structure-based extended resolution adds **auxiliary variables** [CS13]; shortens proofs, but can **destroy symmetries**...

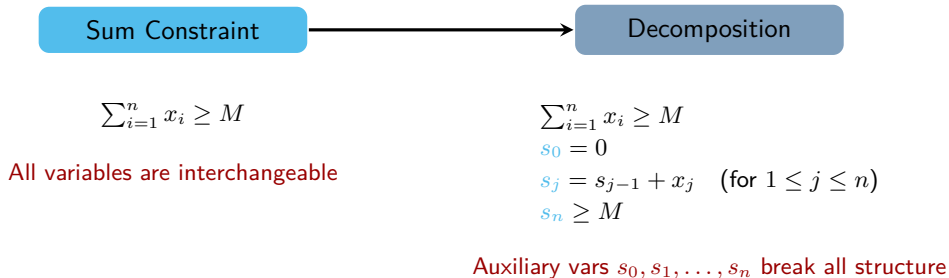
Decompose a sum constraint using a prefix-sum decomposition:



TRANSFORMATIONS DESTROY SYMMETRY

Example: Structure-based extended resolution adds **auxiliary variables** [CS13]; shortens proofs, but can **destroy symmetries**...

Decompose a sum constraint using a prefix-sum decomposition:



Goal

A framework for **symmetry-aware transformations**: study how symmetries behave as problems pass through a transformation pipeline.

Current situation:

- ▶ Break symmetries before the transformations,
- ▶ or *hope* that transformations preserve symmetry...

With this framework:

- ▶ Preserve symmetry during transformations,
- ▶ Such that **final solver** can:
 - ▶ Choose its own lex order [GMR09]
 - ▶ Use dynamic symmetry breaking [MGDW14, DBB17]
 - ▶ Exploit symmetries during search [GHK02]

Definition

A *symmetry-aware problem* is a triple $\langle \mathcal{P}, \mathcal{V}, G \rangle$ with

- ▶ $\mathcal{P}$, a **constraint problem**, consisting of:
 - ▶ $\text{voc}(\mathcal{P})$, the variables of P
 - ▶ $\text{form}(\mathcal{P})$, the formula of P
- ▶ $\mathcal{V} \subseteq \text{voc}(\mathcal{P})$, a set of **projection variables** (the ones that “matter”)
- ▶ G , a group of **projected symmetries** acting only on $\mathcal{V}$

Intuition: “Project” away encoding details

Definition

A *symmetry-aware problem* is a triple $\langle \mathcal{P}, \mathcal{V}, G \rangle$ with

- ▶ $\mathcal{P}$, a **constraint problem**, consisting of:
 - ▶ $\text{voc}(\mathcal{P})$, the variables of P
 - ▶ $\text{form}(\mathcal{P})$, the formula of P
- ▶ $\mathcal{V} \subseteq \text{voc}(\mathcal{P})$, a set of **projection variables** (the ones that “matter”)
- ▶ G , a group of **projected symmetries** acting only on $\mathcal{V}$

Intuition: “Project” away encoding details

$$\text{voc}(\mathcal{P}) = \underbrace{\{x_1, \dots, x_n\}}_{\mathcal{V}}, s_0, \dots, s_n\}$$

Definition

A permutation π on $\mathcal{V}$ is a **projected ($\mathcal{V}$ -)symmetry** of $\mathcal{P}$ iff for every $\mathcal{V}$ -assignment α :

$$\alpha \text{ extends to a solution of } \mathcal{P} \iff \pi(\alpha) \text{ extends to a solution of } \mathcal{P}$$

Note: If $\mathcal{V} = \text{voc}(\mathcal{P})$: π preserves satisfiability of α . Otherwise, π preserves **extensibility** to a solution.

Definition

A permutation π on $\mathcal{V}$ is a **projected ($\mathcal{V}$ -)symmetry** of $\mathcal{P}$ iff for every $\mathcal{V}$ -assignment α :

$$\alpha \text{ extends to a solution of } \mathcal{P} \iff \pi(\alpha) \text{ extends to a solution of } \mathcal{P}$$

Note: If $\mathcal{V} = \text{voc}(\mathcal{P})$: π preserves satisfiability of α . Otherwise, π preserves **extensibility** to a solution.

Example: sum decomposition with auxiliary variables $s_0, \dots, s_n$

- ▶ $\mathcal{V} = \text{voc}(\mathcal{P})$: Permuting $x_1, \dots, x_n$ is **not a symmetry** of the decomposed problem (the s_i impose a specific order)
- ▶ $\mathcal{V} = \text{voc}(\mathcal{P}) \setminus \{s_0, \dots, s_n\}$: It **is a $\mathcal{V}$ -symmetry**: any satisfying x -assignment remains satisfying under permutation (just re-run the decomposition for the new order)

Definition

A **symmetry-aware transformation**:

- ▶ maps $\langle \mathcal{P}, \mathcal{V}, G \rangle \rightarrow \langle \mathcal{P}', \mathcal{V}', G' \rangle$
- ▶ defines a reconstruction function ρ to map solutions of $\mathcal{P}'$ to solutions of $\mathcal{P}$

Definition

A **symmetry-aware transformation**:

- ▶ maps $\langle \mathcal{P}, \mathcal{V}, G \rangle \rightarrow \langle \mathcal{P}', \mathcal{V}', G' \rangle$
- ▶ defines a reconstruction function ρ to map solutions of $\mathcal{P}'$ to solutions of $\mathcal{P}$

Example: $\langle \mathcal{P}, \mathcal{V}, G \rangle \rightarrow \langle \mathcal{P}, \{\}, \{\} \rangle$ is a transformation, but not a very good one...

GOOD SYMMETRY-AWARE TRANSFORMATIONS

Property	Intuition
Reconstructing	$\mathcal{P}'$ does not contain extra solutions (preserves UNSAT)
Solution-preserving	No solutions of $\mathcal{P}$ are lost (modulo G) (preserves SAT)

GOOD SYMMETRY-AWARE TRANSFORMATIONS

Property	Intuition
Reconstructing	$\mathcal{P}'$ does not contain extra solutions (preserves UNSAT)
Solution-preserving	No solutions of $\mathcal{P}$ are lost (modulo G) (preserves SAT)
Backwards-symmetric	The transformation does not split equivalence classes
Forwards-symmetric	The transformation does not merge equivalence classes

GOOD SYMMETRY-AWARE TRANSFORMATIONS

Property	Intuition
Reconstructing	$\mathcal{P}'$ does not contain extra solutions (preserves UNSAT)
Solution-preserving	No solutions of $\mathcal{P}$ are lost (modulo G) (preserves SAT)
Backwards-symmetric	The transformation does not split equivalence classes
Forwards-symmetric	The transformation does not merge equivalence classes

Perfect transformation

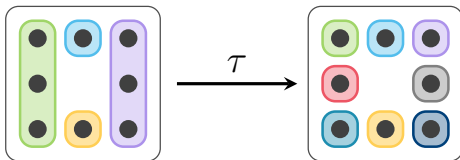
Satisfies *all four* properties — induces a **bijection between equivalence classes**.

Safe for enumeration modulo G .

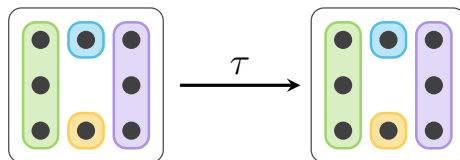
CLASSIFYING EXISTING TECHNIQUES

Technique	Rec.	Sol-pres.	Bwd-sym.	Fwd-sym.	Perfect?
Naive sum decomposition	✓	✓	✓	✗	No
Symm.-aware sum decomposition	✓	✓	✓	✓	Yes

Naive



Symm.-Aware



CLASSIFYING EXISTING TECHNIQUES

Technique	Rec.	Sol-pres.	Bwd-sym.	Fwd-sym.	Perfect?
Naive sum decomposition	✓	✓	✓	✗	No
Symm.-aware sum decomposition	✓	✓	✓	✓	Yes

Sum decomposition: the punchline

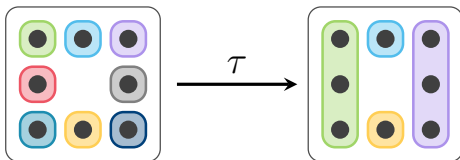
Simply **projecting out auxiliary variables** from the decomposition encoding makes it a **perfect** transformation — symmetries are fully preserved.

Same holds for **cardinality encodings** [BB03], **BVA preprocessing** [HGH23]...

CLASSIFYING EXISTING TECHNIQUES

Technique	Rec.	Sol-pres.	Bwd-sym.	Fwd-sym.	Perfect?
Naive sum decomposition	✓	✓	✓	✗	No
Symm.-aware sum decomposition	✓	✓	✓	✓	Yes
Symmetry detection [MP14, AS21]	✓	✓	✗	✓	No

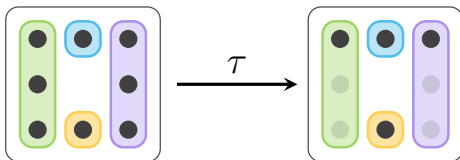
Symm. Detection



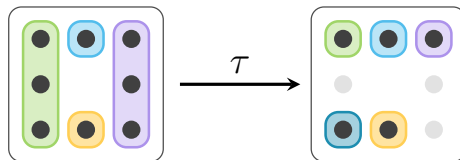
CLASSIFYING EXISTING TECHNIQUES

Technique	Rec.	Sol-pres.	Bwd-sym.	Fwd-sym.	Perfect?
Naive sum decomposition	✓	✓	✓	✗	No
Symm.-aware sum decomposition	✓	✓	✓	✓	Yes
Symmetry detection [MP14, AS21]	✓	✓	✗	✓	No
Complete SB [CGLR96]	✓	✓	✓	✓	Yes
Partial SB	✓	✓	✓	✗	No

Complete Symm. Breaking



Partial Symm. Breaking

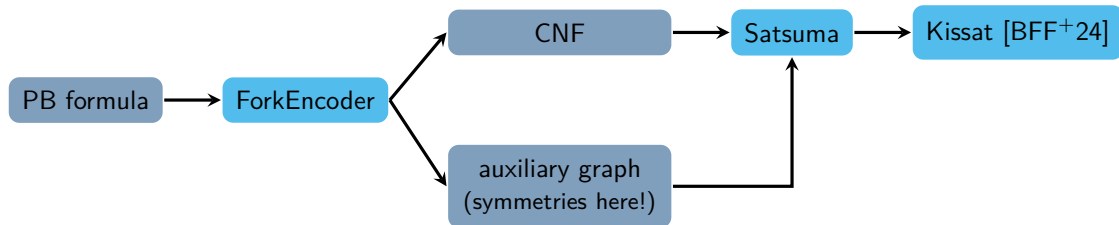


PRACTICAL REALISATION: FORKENCODER

Goal: Implement a perfect symmetry-aware transformation for PB to CNF encoding.

ForkEncoder

- ▶ **Input:** PB formula (OPB format)
- ▶ **Output:** CNF encoding + auxiliary graph encoding original PB symmetries
- ▶ Connects to [Satsuma](#) [ABR26] (symmetry-breaking preprocessor), which reads the graph instead of detecting symmetries from the CNF



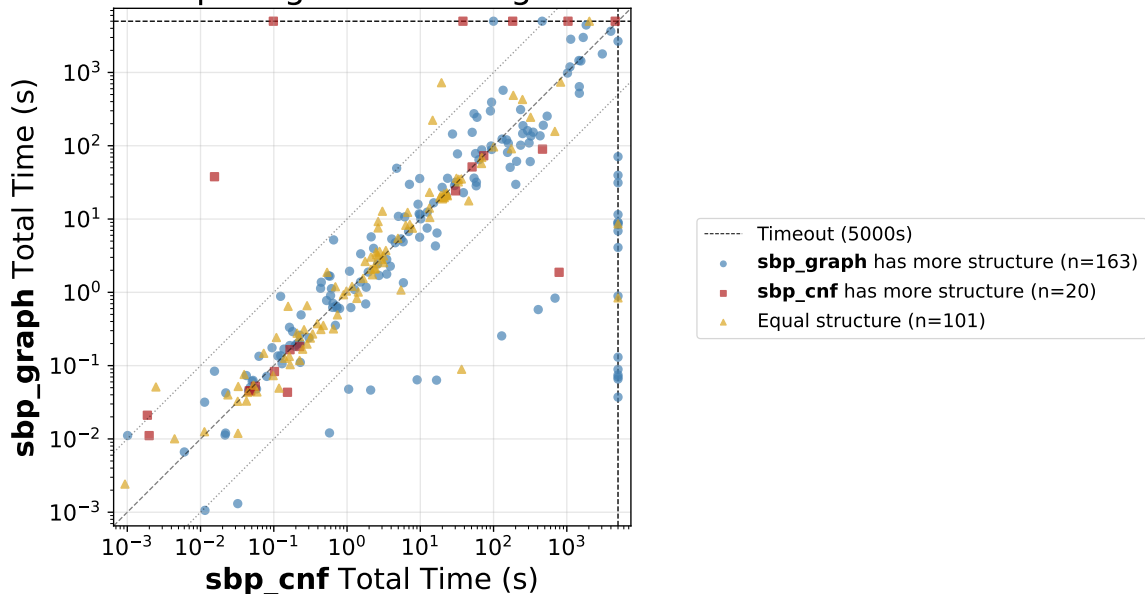
EXPERIMENTAL RESULTS

Benchmark: Instances from PB Competition 2025 [Rou25] — 354 instances with non-trivial symmetries

Configuration	Solved	Avg. time	PAR-2
No symmetry breaking	247	1602 s	3114 s
SBP from CNF (baseline)	265	1372 s	2629 s
SBP from auxiliary graph	276	1199 s	2301 s

- ▶ +11 instances solved vs. CNF-based SBP
- ▶ 12.6% speed-up in average solving time vs. CNF-based SBP
- ▶ Auxiliary graph indeed preserves structural symmetries that are lost in the CNF encoding

Comparing Total Solving Time



What this paper gives you:

1. A **clean formal framework** for tracking symmetry through transformations
2. **Projection variables** are the key mechanism for handling auxiliary variables
3. **Perfect transformations** are safe for isomorph-free enumeration and counting
4. **Practical validation** of key concepts: 12.6% speed-up on real benchmarks

What this paper gives you:

1. A **clean formal framework** for tracking symmetry through transformations
2. **Projection variables** are the key mechanism for handling auxiliary variables
3. **Perfect transformations** are safe for isomorph-free enumeration and counting
4. **Practical validation** of key concepts: 12.6% speed-up on real benchmarks

Thank you!

Questions?

ACKNOWLEDGEMENTS



Co-funded by the European Union (ERC, CertiFOX, 101122653). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

- [ABR26] Markus Anders, Sofia Brenner, and Gaurav Rattan. Satsuma: Structure-based symmetry breaking in SAT. *J. Artif. Intell. Res.*, 85, 2026.
- [AS21] Markus Anders and Pascal Schweitzer. Engineering a fast probabilistic isomorphism test. In *ALENEX*, pages 73–84. SIAM, 2021.
- [BB03] Olivier Bailleux and Yacine Boufkhad. Efficient CNF encoding of Boolean cardinality constraints. In *CP*, volume 2833 of *LNCS*, pages 108–122. Springer, 2003.
- [BFF⁺24] Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleys, and Florian Pollitt. CaDiCaL, Gimsatul, IsaSAT and Kissat entering the SAT Competition 2024. In Marijn Heule, Ashlin Iser, Matti Järvisalo, and Martin Suda, editors, *Proc. of SAT Competition 2024 – Solver, Benchmark and Proof Checker Descriptions*, volume B-2024-1 of *Department of Computer Science Report Series B*, pages 8–10. University of Helsinki, 2024.

- [CGLR96] James M. Crawford, Matthew L. Ginsberg, Eugene M. Luks, and Amitabha Roy. Symmetry-breaking predicates for search problems. In *KR*, pages 148–159. Morgan Kaufmann, 1996.
- [CS13] Geoffrey Chu and Peter J. Stuckey. Structure based extended resolution for constraint programming. *CoRR*, abs/1306.4418, 2013.
- [DBB17] Jo Devriendt, Bart Bogaerts, and Maurice Bruynooghe. Symmetric explanation learning: Effective dynamic symmetry handling for SAT. In *SAT*, volume 10491 of *LNCS*, pages 83–100. Springer, 2017.
- [GHK02] Ian P. Gent, Warwick Harvey, and Tom W. Kelsey. Groups and constraints: Symmetry breaking during search. In *CP*, *LNCS*, pages 415–430. Springer, 2002.

- [GMR09] Andrew Grayland, Ian Miguel, and Colva M. Roney-Dougal. Snake lex: An alternative to double lex. In *Principles and Practice of Constraint Programming - CP 2009, 15th International Conference, CP 2009, Lisbon, Portugal, September 20-24, 2009, Proceedings*, pages 391–399, 2009.
- [HGH23] Andrew Haberlandt, Harrison Green, and Marijn J. H. Heule. Effective auxiliary variables via structured reencoding. In *SAT*, volume 271 of *LIPIcs*, pages 11:1–11:19. Schloss Dagstuhl, 2023.
- [MGDW14] Christopher Mears, Maria Garcia de la Banda, Bart Demoen, and Mark Wallace. Lightweight dynamic symmetry breaking. *Constraints An Int. J.*, 19(3):195–242, 2014.
- [MP14] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *J. Symb. Comput.*, 60:94–112, 2014.
- [Rou25] Olivier Roussel. Pseudo-Boolean Competition 2025, 2025.