# SAT Modulo Symmetries: A Survey

Stefan Szeider

*Algorithms and Complexity Group, TU Wien, Vienna, Austria*

**Abstract**

The generation of combinatorial objects with a given property is a fundamental task in discrete mathematics and computer science. The combinatorial explosion of candidate objects renders simple generate-and-test approaches infeasible. SAT-based synthesis methods offer an alternative, but they require effective symmetry breaking to prevent the solver from exploring isomorphic parts of the search space. This paper surveys SAT Modulo Symmetries (SMS), a framework that integrates dynamic symmetry breaking directly into a Conflict-Driven Clause Learning (CDCL) SAT solver. The approach relies on a propagator that checks for the canonicity of partially defined objects during the search. If a partial assignment cannot be extended to a canonical object, the propagator provides a learned clause to the solver, pruning the corresponding part of the search tree. We outline the core components of the SMS framework and review its application to several open problems in areas such as extremal graph theory, Ramsey theory, matroid theory, and quantum foundations, where it has been used to obtain new mathematical results.

**Keywords**

SAT solving, dynamic symmetry breaking, graph generation, isomorph-free generation, extremal combinatorics, computational mathematics

## 1. Introduction

Many problems in discrete mathematics concern the existence or properties of combinatorial objects. For example, what is the smallest triangle-free graph that is non-2-colorable? One can deduce that this is a cycle of length 5. If we ask for the smallest triangle-free graph that is non-3-colorable, the answer is not obvious: it is the Grötzsch graph (Figure 1), an 11-vertex graph discovered in the 1950s.

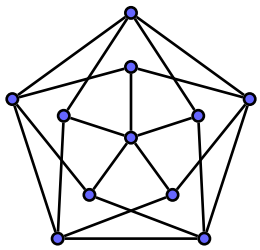


**Figure 1:** The Grötzsch graph.

Such questions fall into a general class of problems that includes enumerating all objects of a certain size with a given property, finding extremal objects, or searching for counterexamples to mathematical conjectures.

A principal difficulty in tackling these problems computationally is the combinatorial explosion. For example, the number of graphs with vertices labeled $1, \ldots, n$ grows from 1024 for $n = 5$ to approximately $3.6 \times 10^{16}$ for $n = 11$.[1] A simple generate-and-test methodology is not feasible for such quantities.

A key technique to manage this complexity is *isomorph-free generation*, which ensures that only one representative from each isomorphism class is generated. Two graphs are *isomorphic* if they have the

*10th International Workshop on Satisfiability Checking and Symbolic Computation, August 2, 2025, Stuttgart, Germany*

sz@ac.tuwien.ac.at (S. Szeider)

0000-0001-8994-1656 (S. Szeider)



[1] OEIS sequence A006125: https://oeis.org/A006125

same structure, differing only in how vertices are labeled. A graph $G$ is considered *canonical* if it is selected as the unique representative of its isomorphism class; for instance, if its adjacency matrix is lexicographically minimal among all labelings. For $n = 11$ the number of non-isomorphic graphs is approximately $1.02 \times 10^9$,[2] compared to $3.6 \times 10^{16}$ labeled graphs—a reduction by seven orders of magnitude. However, this is still a very large number. The standard way to answer the above question about the smallest triangle-free graph that is non-3-colorable would be to enumerate all canonical graphs with $n \leq 11$ vertices and check for each whether it is triangle-free and non-3-colorable. This approach is limited to $n$ where it is still feasible to enumerate.

An alternative that avoids enumeration is to formulate the problem as a *synthesis task* using propositional satisfiability (SAT) solvers. Here, one defines propositional variables to represent the components of the object (e.g., $e_{u,v}$ stating whether the edge between vertices $u$ and $v$ exists) and encodes the desired property as a propositional formula. A satisfying assignment to the formula corresponds to an object with the property. Modern SAT solvers based on Conflict-Driven Clause Learning (CDCL) can explore vast search spaces [1]. However, without a mechanism to handle symmetries, a SAT solver will redundantly explore different labelings of the same underlying structure.

Static symmetry breaking, where canonicity constraints are added to the initial propositional formula, provides a partial solution. However, a polynomial-size encoding for full canonicity is not known, so these approaches must rely on incomplete symmetry breaking, which may still generate multiple isomorphic copies [2].

This paper surveys *SAT Modulo Symmetries (SMS)*, a framework for isomorph-free generation that implements dynamic and complete symmetry breaking within a CDCL SAT solver. It avoids the limitations of static methods by interacting with the solver during its search, pruning branches that are guaranteed not to lead to canonical solutions. SMS was originally introduced at CP 2021 [3]. A more recent journal paper [4] provides a refined technical exposition, details and updates.

We describe the SMS framework and review its application to a range of problems in combinatorics and related fields.

## 2. The SAT Modulo Symmetries Framework

The central idea of SMS is to augment a CDCL SAT solver with a custom propagator that performs canonicity tests on-the-fly.

### 2.1. System Architecture

The SMS architecture consists of two main components: (1) a standard CDCL SAT solver that operates on a formula encoding the desired property of the combinatorial object, and (2) symmetry-breaking propagator MinCheck, or *canonicity tester*, that communicates with the solver through a callback-based interface.

During its search, the SAT solver makes decisions, assigning truth values to variables. These assignments correspond to a partially constructed object. At certain points, the solver queries the propagator. The propagator analyzes the current partially defined object for canonicity. If it determines that the partially defined object cannot be extended to any canonical solution, it learns a blocking clause. This clause is passed back to the SAT solver, which uses it to backtrack and prune that entire region of the search space.

### 2.2. Partially Defined Graphs and Canonicity

A key concept in SMS is the *partially defined graph*. At any point during the solver's execution, the set of edge variables is partitioned into those assigned *true*, those assigned *false*, and those that are unassigned. This corresponds to a graph $G$ where the edge set is partitioned into a set $D(G)$ of defined

[2] OEIS sequence A000088: https://oeis.org/A000088

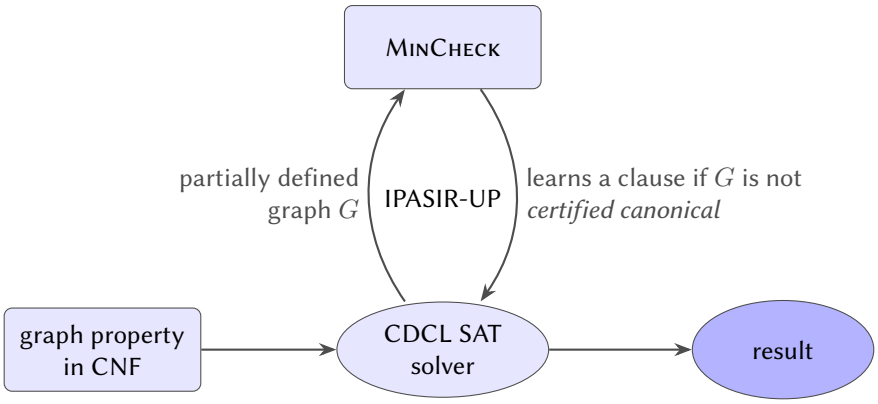
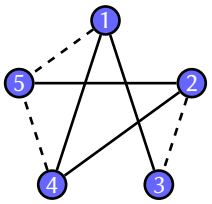



**Figure 2:** The SMS system architecture showing the interaction between the SAT solver and MinCheck via the IPASIR-UP interface.

edges, a set $U(G)$ of undefined edges, and the remaining defined non-edges. Such a graph $G$ represents the set $\mathcal{X}(G)$ of all fully defined graphs that can be obtained by assigning the variables corresponding to edges in $U(G)$. In the adjacency matrix representation shown in Figure 3, entries are 1 for edges, 0 for non-edges, and $\star$ for undefined edges.




| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **1** | 0 | 0 | 1 | 1 | $\star$ |
| **2** | 0 | 0 | $\star$ | 1 | 1 |
| **3** | 1 | $\star$ | 0 | 0 | 0 |
| **4** | 1 | 1 | 0 | 0 | $\star$ |
| **5** | $\star$ | 1 | 0 | $\star$ | 0 |

**Figure 3:** A partially defined graph and its adjacency matrix.

A partially defined graph $G_1$ is a *refinement* of $G_2$ (written $G_1 \sqsupseteq G_2$) if more edges are defined in $G_1$, meaning $\mathcal{X}(G_1) \subseteq \mathcal{X}(G_2)$.

The goal is to prune the search as soon as the current partially defined graph $G$ is provably non-canonical. Ideally, one would terminate a search branch if for every possible extension $H \in \mathcal{X}(G)$, there exists a permutation $\pi$ such that $\pi(H)$ is lexicographically smaller than $H$. (Here, lexicographic order compares adjacency matrices row-major, reading the full symmetric matrix including the diagonal.) However, checking this property is computationally difficult, as it involves quantifiers over exponentially large sets of graphs and permutations.

SMS employs a more tractable relaxation of this condition. It searches for a single permutation $\pi$ that acts as a *certificate of non-canonicity* (or *nc-certificate*) for the entire set $\mathcal{X}(G)$. That is, it checks if:

> There exists a permutation $\pi$ such that for all extensions $H \in \mathcal{X}(G)$, the adjacency matrix of $\pi(H)$ is lexicographically smaller than the adjacency matrix of $H$.

This condition, with its swapped quantifiers, is sufficient to prove that no extension of $G$ can be canonical. While this check is a relaxation, it becomes equivalent to the full canonicity test when the graph is fully defined (i.e., $U(G) = \emptyset$). Thus, the method is sound (it never prunes canonical solutions) and complete (it finds all canonical solutions).

## 2.3. Finding a Certifying Permutation

We now describe how SMS searches for an nc-certificate $\pi$ by exploring a refined space of permutations using ordered partitions. The heart of SMS is the canonicity tester MinCheck , which searches for an nc-certificate $\pi$. This search is structured using *ordered partitions* of the vertex set $V = \{1, \ldots, n\}$. An

ordered partition $P = [V_1, \ldots, V_k]$ is a sequence of disjoint vertex sets whose union is $V$. It represents the set of all permutations $\pi$ where for any $u \in V_i$ and $v \in V_j$ with $i < j$, it holds that $\pi(u) < \pi(v)$.

The algorithm proceeds recursively:

1. It starts with the trivial partition $[\{1, \ldots, n\}]$, which represents all $n!$ permutations.
2. It refines the partition by making decisions. For instance, if the current partition is $[\{1, 2, 3\}, \{4, 5\}]$ and we decide to map vertex 2 to position 1, we obtain the refined partition $[\{2\}, \{1, 3\}, \{4, 5\}]$.
3. After each decision, it propagates constraints derived from the structure of the partially defined graph to further refine the other sets in the partition, reducing the set of candidate permutations.

This structured search continues until an nc-certificate is found or the search space of permutations is exhausted for the current branch. In the worst case, the search is exponential, but due to the propagation, it often runs quickly.

If an nc-certificate $\pi$ is found for a partially defined graph $G$, the algorithm identifies a minimal non-canonical subgraph $G'$ with $G \sqsupseteq G'$, called an *obstruction*. From this obstruction, a blocking clause is generated. For instance, the clause

$$\bigvee_{uv \in D(G')} \neg e_{u,v} \vee \bigvee_{uv \notin D(G') \cup U(G')} e_{u,v}$$

is added to the solver. This clause blocks $G'$ and all its extensions, effectively pruning the search.

## 3. Extensions

The core SMS framework described in the previous section has been extended in several directions to enhance its capabilities and broaden its applicability.

### 3.1. Specialized Propagators

The SMS framework can be extended with domain-specific propagators that encode graph-theoretic lemmas directly into the search process, building on the core MinCheck propagator (Section 2). These specialized propagators augment the canonicity tester with additional constraints derived from structural properties of the target objects. By pruning branches based on domain knowledge, they can reduce the search space beyond what pure symmetry breaking achieves.

This approach was used to partially verify the 3-Decomposition Conjecture—that every connected cubic graph can be decomposed into a spanning tree, a collection of cycles, and a matching. Using SMS with a specialized propagator encoding structural properties of cubic graphs, all graphs up to 28 vertices were verified, extending the previously verified bound from 20 vertices [5].

### 3.2. Handling coNP Properties with Co-Certificate Learning

Many mathematical properties, such as non-3-colorability, are coNP-complete and difficult to encode directly with SAT clauses. To address this class of problems, SMS was extended with *Co-Certificate Learning* (CCL) [6]. The technique applies to *alternating search problems*, where the goal is to find a combinatorial object satisfying two graph properties: property $A$ (typically an NP property) and the negation of property $B$ (where $B$ is an NP property, making its negation a coNP property). Our example from above, finding the smallest triangle-free graph that is non-3-colorable is of that type: property $A$ is triangle-freeness (even in $P$) and property $B$ is 3-colorability (whose negation, non-3-colorability, is the desired coNP property).

The CCL framework uses a dual-solver architecture (Figure 4), complementing SMS's symmetry pruning. SAT solver 1 generates a candidate graph satisfying a property $A$ (e.g., triangle-free). This candidate is passed to SAT solver 2, which tests whether the graph satisfies a property $B$ (e.g., 3-colorability). If SAT solver 2 reports unsatisfiability (the graph does not satisfy $B$), then the candidate

satisfies both $A$ and $\neg B$, and a solution is found. If SAT solver 2 reports satisfiability (the graph satisfies $B$), it returns a *co-certificate*—a satisfying assignment that witnesses property $B$. For 3-colorability, a co-certificate is a valid 3-coloring.

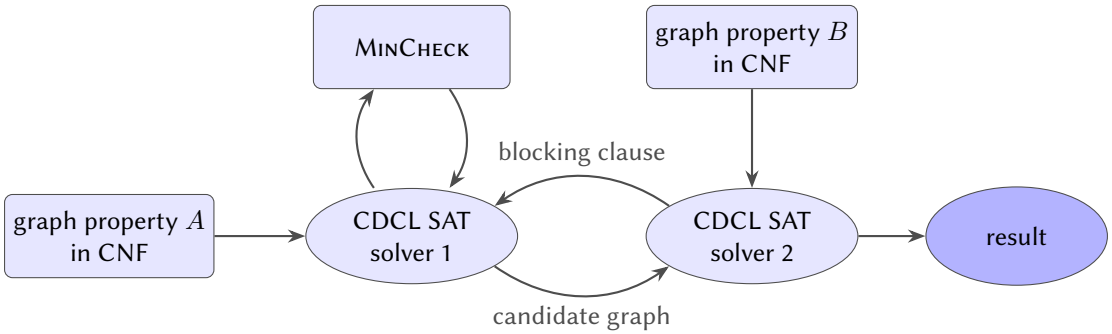



**Figure 4:** The CCL architecture extending SMS with a second SAT solver for co-certificate learning.

From this co-certificate, a blocking clause is constructed and learned by SAT solver 1. For a 3-coloring $c$, the clause requires that any graph considered in subsequent search must contain at least one edge between vertices that were assigned the same color by $c$:

$$\bigvee_{u<v,\, c(u)=c(v)} e_{u,v}$$

This clause eliminates all graphs that satisfy the property $B$ via the same coloring $c$, and SAT solver 1 resumes its search for a graph satisfying $A$ but not $B$.

A key observation is that a single co-certificate can significantly reduce the search space. For triangle-free graphs on 14 vertices, while tools like Nauty must check over 467 million canonical graphs, SMS with CCL visits only 85,668 candidates by learning just 9,407 3-colorings from SAT solver 2. On average, each 3-coloring eliminates approximately 50,000 canonical graphs [6].

A further refinement, Q-SMS, unifies this dual-solver architecture by formulating the entire problem as a Quantified Boolean Formula (QBF) and using a circuit-based QBF solver [7].

### 3.3. Proofs

The SMS framework can produce independently verifiable proofs that certify both the correctness and completeness of its results [4]. The verification architecture is separated into two components that correspond to the two main parts of the system: the SAT solver's logical reasoning and the dynamic symmetry breaking.

For the symmetry-breaking component, SMS outputs an *nc-certificate* (a permutation $\pi$) for each learned symmetry-breaking clause. An independent checker can verify in polynomial time that this permutation indeed witnesses the non-canonicity of the pruned partial graph. This ensures that no canonical solution was incorrectly discarded.

For the SAT reasoning component, the framework can generate standard DRAT (Deletion Resolution Asymmetric Tautology) proofs [8]. After an initial run that generates all necessary symmetry-breaking clauses, these clauses are added to the problem encoding. A second run with a proof-producing SAT solver outputs a DRAT proof, which can be checked by existing proof verification tools. This certifies the correctness of unsatisfiability results. Implementation details for proof generation are provided in Section 5.

To certify the completeness of an enumeration, the framework adds blocking clauses to exclude all found solutions and verifies the unsatisfiability of the resulting formula. The DRAT proof for this extended formula certifies that no additional non-isomorphic solutions exist. Individual solutions can be certified as canonical by encoding the canonicity test as a CNF formula; a DRAT proof of unsatisfiability for this formula confirms that the solution is lexicographically minimal among all isomorphic copies.

# 4. Applications

The SMS framework has been applied to computational problems across several areas of discrete mathematics, yielding new results and confirming existing conjectures for problem sizes previously beyond the reach of prior methods.

## 4.1. Extremal Graph Theory

Extremal graph theory seeks graphs that maximize or minimize a parameter subject to given constraints. One such problem concerns *diameter-2-critical* graphs. A graph has diameter $d$ if the longest shortest paths between any two vertices have length $d$. A graph is diameter-$d$-critical if its diameter is $d$, but deleting any edge increases the diameter. The Simon-Murty Conjecture (1979) states that a diameter-2-critical graph on $n$ vertices has at most $\lfloor n^2/4 \rfloor$ edges.

Prior computational searches were limited to graphs with at most 11 vertices. Using a direct SAT encoding of the diameter-2-critical property, SMS enumerated all such graphs for larger sizes. For $n = 12$, it found all 40,866 graphs in under a minute, and for $n = 13$, it enumerated all 688,120 graphs in approximately 20 minutes. Static symmetry breaking methods did not terminate for $n \geq 12$. By incorporating known degree constraints from the literature to partition the search space, the conjecture was further verified for all $n \leq 19$ [4].

Beyond diameter-2-critical graphs, we studied extremal problems in planar graphs via planarity encodings. The planar Turán number $\mathrm{ex}_P(n, H)$ is the maximum number of edges in an $n$-vertex planar graph containing no subgraph isomorphic to $H$. Computing these numbers requires generating planar graphs with specified properties. We compared three planarity encodings: a lazy Kuratowski-based propagator that detects forbidden Kuratowski subdivisions (topological $K_5$ or $K_{3,3}$) using the lazy propagator approach introduced in Section 3.1, an eager encoding using Schnyder's planarity criterion [9], and an eager encoding based on universal point sets [10]. The lazy Kuratowski-based approach proved orders of magnitude faster, particularly for unsatisfiable instances. SMS determined the exact values $\mathrm{ex}_P(n, C_4)$ and $\mathrm{ex}_P(n, C_5)$ for $n \leq 18$. For $\mathrm{ex}_P(17, C_4)$, the lazy encoding completed the search over 100 times faster than the eager encodings [11].

## 4.2. Graph Decompositions and Coloring

This area concerns partitioning graph edges or coloring vertices under complex structural constraints. The 3-Decomposition Conjecture, as mentioned in Section 3.1, states that every connected cubic graph can be partitioned into a spanning tree, a collection of cycles, and a matching. Using specialized propagators, SMS exploits theoretical properties of minimal counterexamples. A forbidden subgraph checker, integrating the Glasgow Subgraph Solver [12], dynamically prunes the search space based on reducible templates—subgraphs that cannot appear in a minimal counterexample. This approach verified the conjecture for all connected cubic graphs up to 28 vertices, extending the previous bound from 20 vertices.

A related coloring problem concerns biplanar graphs—graphs whose edges can be partitioned into two planar graphs. The Earth-Moon problem [13] asks for the chromatic number of biplanar graphs, generalizing the Four-Color Theorem. SMS was adapted to handle directed graphs: we represent one planar layer by anti-parallel arcs and the other by single arcs, and controlling crossings per layer then enforces biplanarity. This approach proved that all biplanar graphs on up to 13 vertices are 9-colorable, settling several open cases. The method also demonstrated that a specific 19-vertex candidate graph is not biplanar, ruling it out as a potential counterexample [11].

## 4.3. Ramsey and Universality Problems

These problems concern the unavoidable appearance of substructures in large combinatorial objects. The Ramsey number $R(x, y)$ is the smallest integer $n$ such that any graph on $n$ vertices contains either a clique of size $x$ or an independent set of size $y$. The set $\mathcal{R}(x, y, n)$ consists of all non-isomorphic

graphs on $n$ vertices with no $x$-clique and no $y$-independent set. Using a direct encoding of these properties, SMS enumerated the complete sets for $\mathcal{R}(3,5,n)$ and $\mathcal{R}(4,4,n)$. The framework's ability to produce verifiable proofs of completeness is particularly valuable in this context. The DRAT proof from the SAT solver certifies the search, while nc-certificates for each learned symmetry clause can be independently verified [4].

A related universality problem asks for the smallest graph containing all graphs of a given size. A graph is *$k$-universal* if it contains all graphs on $k$ vertices as induced subgraphs. The problem is to find the smallest such graph, denoted $f(k)$. A similar problem exists for tournaments, $t(k)$. SMS was applied using both a direct SAT encoding and a lazy approach with the Glasgow Subgraph Solver as an external propagator. A templates method, where pre-computed interactions of small pattern graphs partition the search space for parallel execution, further improved performance. This work improved the lower bound for 7-universal graphs, proving $f(7) > 16$, and determined exact values $t(k)$ for $k \leq 6$ and bounded the 7-universal tournament size to $13 \leq t(7) \leq 15$ [14].

### 4.4. Abstract Combinatorial Structures

The SMS framework extends beyond simple graphs to more abstract combinatorial objects. For *hypergraphs*, the Erdős-Faber-Lovász Conjecture (1972) concerns the edge-colorability of linear hypergraphs (hypergraphs where each pair of edges shares at most one vertex). To apply SMS, hypergraphs are modeled via their bipartite incidence graphs. The coNP property of non-colorability is handled using Co-Certificate Learning. This approach verified the conjecture for all linear hypergraphs with up to 12 vertices, extending previous computational results and providing the first such verification with proof-generation capabilities [15].

For *matroids*, Rota's Basis Conjecture (1989) concerns decompositions into disjoint rainbow bases. We developed a specialized MinCheck propagator, tailored to the basis exchange axioms of matroids of bounded rank. This led to a proof of the conjecture for all matroids of rank 4. The work also provided the first framework for producing DRAT proofs for matroid problems [16].

We also extended the framework to *CNF formulas* by encoding them as graphs. Our goal was to find smallest unsatisfiable $(k, s)$-CNF formulas, where each clause has exactly $k$ literals and each variable occurs at most $s$ times. The size of these formulas relates to inapproximability results for MaxSAT. We encoded CNF formulas as 2-colored graphs (clause-literal graphs) to apply SMS. The search combined SMS with theoretical techniques like disjunctive splitting. This determined exact minimum sizes of unsatisfiable formulas for all 3-CNF subclasses and improved the upper bound for the smallest unsatisfiable (4,5)-CNF formula from Knuth's 257 to 235 clauses [17, 18].

### 4.5. Applications in Other Scientific Domains

We also applied SMS to problems from fields outside pure combinatorics. In *quantum foundations*, Kochen-Specker systems are sets of vectors that demonstrate quantum contextuality, a key distinction between quantum and classical physics. Finding the smallest such system is a long-standing problem. We formulated the search for a Kochen-Specker system as finding a graph with a coNP property (non-010-colorability), which was handled via CCL (Section 3.2). A 010-coloring assigns colors 0, 1 to vectors respecting orthogonality [6]. Our SMS approach improved the lower bound from 23 to 24 vectors [6], significantly faster than a SAT+CAS (Computer Algebra System) method that had established the 23-vector bound; the SAT+CAS approach subsequently confirmed the 24-vector bound independently [19].

In *computational social choice*, the rainbow cycle number $R_f(d)$ provides bounds on the feasibility of achieving envy-free-up-to-any-good (EFX) allocations in the fair division of indivisible goods. Variants $R_i(d)$ and $R_p(d)$ use edge labelings restricted to functions $x \mapsto x + k \bmod d$ and permutations, respectively, with $R_i(d) \leq R_p(d) \leq R_f(d)$. We introduced the new technique of *invariant pruning*: to branch on the value of a graph invariant not known a priori, such as maximum degree, reducing symmetry [20].. Using SMS with invariant pruning, we determined the exact values $R_f(4) = 4$ and

$R_p(5) = 5$, improving guarantees for EFX allocations. Invariant pruning provided a speedup of almost two orders of magnitude in some cases.

# 5. Software

To facilitate adoption and enable further research, the SMS framework is implemented as open source software. This section outlines its architecture and practical usage, connecting the theoretical concepts from Section 2 to their implementation. The SMS framework is available at

`https://github.com/markirch/sat-modulo-symmetries`,

with documentation at

`https://sat-modulo-symmetries.readthedocs.io/`.

The software consists of a C++ core implementation and a Python wrapper that provides a high-level interface for encoding graph properties.

## 5.1. C++ Implementation

The core implementation consists of the solver `smsg`, which is built on CaDiCaL, an award-winning CDCL SAT solver, extended with the MinCheck propagator for symmetry breaking. The integration utilizes the IPASIR-UP interface [21], which provides callback functions for communication between the solver and the propagator. When the propagator detects that a partially defined graph cannot be extended to a canonical solution, it constructs a conflict clause and passes it to the solver via `cb_add_external_clause_lit`.

The software also provides C++ libraries (`libsms_static.a` and `libsms.so`) for integration into other applications.

## 5.2. Python Wrapper: PySMS

The Python module PySMS (`pysms`) provides a declarative interface for specifying graph properties without directly manipulating propositional variables. The module `pysms.graph_builder` can be invoked from the command line or imported into custom scripts. It generates CNF encodings in DIMACS format [22] , and, by default, automatically invokes `smsg` to solve them.

For command-line use, the basic syntax is:

```
python -m pysms.graph_builder --vertices n [options]
```

Common options include `–all-graphs` to enumerate all solutions, `–delta-low d` for minimum degree, `–num-edges-upp e` for maximum edges, and `–directed` for directed graphs. To output CNF without solving, add `–no-solve`. For better performance, it is recommended to use `–cutoff 20000` to limit the minimality check due to the exponential worst-case complexity of the ordered partition search in MinCheck (Section 2).

Examples of command-line usage:
The Grötzsch graph: triangle-free, non-3-colorable, 11 vertices:

```
python -m pysms.graph_builder -v 11 --ck-free 3 --min-chromatic-number 4
```

All triangle-free graphs on 7 vertices:

```
python -m pysms.graph_builder -v 7 --ck-free 3 --all-graphs
```

Diameter-2-critical graphs on 12 vertices:

```
python -m pysms.graph_builder -v 12 --diam2-critical --all-graphs
```

Ramsey graphs R(3,5) on 13 vertices:

```
python -m pysms.graph_builder -v 13 --ramsey 3 5 --all-graphs
```

For scripting, the `GraphEncodingBuilder` class provides methods for common constraints and direct clause manipulation. A simple example that enumerates maximal triangle-free graphs:

```
from pysms.graph_builder import GraphEncodingBuilder
from itertools import combinations
builder = GraphEncodingBuilder(7, directed=False)
builder.ckFree(3)  # triangle-free
for i, j in combinations(builder.V, 2):
    # non-neighbors must have a common neighbor
    builder.append([-builder.var_edge(i,j)] +
        [builder.CNF_AND([builder.var_edge(i,k),
                          builder.var_edge(j,k)])
         for k in builder.V if k != i and k != j])
builder.solve(allGraphs=True)
```

The `smsg` solver can also be invoked directly with DIMACS input. Key arguments include `-v` for vertex count, `--all-graphs` for enumeration, `--cutoff` to limit minimality check time, and `--dimacs FILE` to read constraints from a file. For proof generation (Section 3.3), the solver supports outputting nc-certificates for symmetry-breaking clauses and can be configured to enable DRAT proof logging for subsequent verification. For large-scale computations, SMS supports cube-and-conquer parallelization [23] via `--simple-assignment-cutoff` for problem decomposition and `--cube-file` with `--cube-line` for solving individual cubes.

## 6. Conclusion

SAT Modulo Symmetries provides a general and effective framework for certified combinatorial search. By integrating dynamic symmetry breaking directly into the search process of a CDCL SAT solver, it avoids the exhaustive enumeration of generate-and-test methods and overcomes the incompleteness of static symmetry breaking. The core mechanism relies on testing the canonicity of partially defined objects and learning clauses to prune non-canonical branches of the search tree.

The framework's adaptability has been demonstrated through its successful application to a wide range of problems involving graphs, hypergraphs, and matroids. Furthermore, the extension to Co-Certificate Learning enables the efficient handling of problems with coNP properties. By producing verifiable certificates for both the SAT search and the symmetry breaking, SMS contributes to the growing field of computer-assisted mathematical discovery. The source code and documentation for the framework are publicly available, providing researchers with a tool to tackle other open problems in combinatorics and beyond.

## Acknowledgments

The author acknowledges support from the Austrian Science Fund (FWF), project 10.55776/P36688.

## Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to: Grammar and spelling check, paraphrase and reword. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

## References


[1] J. K. Fichte, D. L. Berre, M. Hecher, S. Szeider, The silent (r)evolution of SAT, Communications of the ACM 66 (2023) 64–72. doi:10.1145/3560469.

[2] M. Codish, A. Miller, P. Prosser, P. J. Stuckey, Constraints for symmetry breaking in graph representation, Constraints An Int. J. 24 (2019) 1–24. doi:10.1007/S10601-018-9294-5.

[3] M. Kirchweger, S. Szeider, SAT modulo symmetries for graph generation, in: L. D. Michel (Ed.), 27th International Conference on Principles and Practice of Constraint Programming, CP 2021, Montpellier, France (Virtual Conference), October 25-29, 2021, volume 210 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 34:1–34:16. doi:10.4230/LIPICS.CP.2021.34.

[4] M. Kirchweger, S. Szeider, SAT modulo symmetries for graph generation and enumeration, ACM Trans. Comput. Log. 25 (2024) 1–30. doi:10.1145/3670405.

[5] T. Zhang, S. Szeider, The 3-decomposition conjecture: A SAT-based approach with specialized propagators, in: M. G. de la Banda (Ed.), 31st International Conference on Principles and Practice of Constraint Programming, CP 2025, August 10-15, 2025, Glasgow, Scotland, volume 340 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025, pp. 39:1–39:19. doi:10.4230/LIPICS.CP.2025.39.

[6] M. Kirchweger, T. Peitl, S. Szeider, Co-certificate learning with SAT modulo symmetries, in: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China, ijcai.org, 2023, pp. 1944–1953. doi:10.24963/IJCAI.2023/216.

[7] M. Janota, M. Kirchweger, T. Peitl, S. Szeider, Breaking symmetries in quantified graph search: A comparative study, in: T. Walsh, J. Shah, Z. Kolter (Eds.), AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA, AAAI Press, 2025, pp. 11246–11254. doi:10.1609/AAAI.V39I11.33223.

[8] M. J. H. Heule, The DRAT format and DRAT-trim checker, CoRR abs/1610.06229 (2016). URL: http://arxiv.org/abs/1610.06229. arXiv:1610.06229.

[9] W. Schnyder, Embedding planar graphs on the grid, in: D. S. Johnson (Ed.), Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, 22-24 January 1990, San Francisco, California, USA, SIAM, 1990, pp. 138–148. URL: http://dl.acm.org/citation.cfm?id=320176.320191.

[10] H. de Fraysseix, J. Pach, R. Pollack, Small sets supporting Fáry embeddings of planar graphs, in: J. Simon (Ed.), Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA, ACM, 1988, pp. 426–433. doi:10.1145/62212.62254.

[11] M. Kirchweger, M. Scheucher, S. Szeider, SAT-based generation of planar graphs, in: M. Mahajan, F. Slivovsky (Eds.), 26th International Conference on Theory and Applications of Satisfiability Testing, SAT 2023, July 4-8, 2023, Alghero, Italy, volume 271 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, pp. 14:1–14:18. doi:10.4230/LIPICS.SAT.2023.14.

[12] C. McCreesh, P. Prosser, J. Trimble, The Glasgow subgraph solver: Using constraint programming to tackle hard subgraph isomorphism problem variants, in: F. Gadducci, T. Kehrer (Eds.), Graph Transformation - 13th International Conference, ICGT 2020, Held as Part of STAF 2020, Bergen, Norway, June 25-26, 2020, Proceedings, volume 12150 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 316–324. doi:10.1007/978-3-030-51372-6_19.

[13] G. Ringel, Färbungsprobleme auf Flächen und Graphen, volume 2 of *Mathematische Monographien*, VEB Deutscher Verlag der Wissenschaften, Berlin, 1959.

[14] T. Zhang, S. Szeider, Searching for smallest universal graphs and tournaments with SAT, in: R. H. C. Yap (Ed.), 29th International Conference on Principles and Practice of Constraint Programming, CP 2023, August 27-31, 2023, Toronto, Canada, volume 280 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, pp. 39:1–39:20. doi:10.4230/LIPICS.CP.2023.39.

[15] M. Kirchweger, T. Peitl, S. Szeider, A SAT solver's opinion on the Erdős-Faber-Lovász conjecture, in: M. Mahajan, F. Slivovsky (Eds.), 26th International Conference on Theory and Applications of Satisfiability Testing, SAT 2023, July 4-8, 2023, Alghero, Italy, volume 271 of *LIPIcs*, Schloss Dagstuhl

- Leibniz-Zentrum für Informatik, 2023, pp. 13:1–13:17. doi:10.4230/LIPICS.SAT.2023.13.

[16] M. Kirchweger, M. Scheucher, S. Szeider, A SAT attack on Rota's basis conjecture, in: K. S. Meel, O. Strichman (Eds.), 25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel, volume 236 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, pp. 4:1–4:18. doi:10.4230/LIPICS.SAT.2022.4.

[17] D. E. Knuth, The art of computer programming. Vol. 4B. Combinatorial algorithms. Part 2, Addison-Wesley, Upper Saddle River, NJ, 2023.

[18] T. Zhang, T. Peitl, S. Szeider, Small unsatisfiable k-CNFs with bounded literal occurrence, in: S. Chakraborty, J. R. Jiang (Eds.), 27th International Conference on Theory and Applications of Satisfiability Testing, SAT 2024, August 21-24, 2024, Pune, India, volume 305 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, pp. 31:1–31:22. doi:10.4230/LIPICS.SAT.2024.31.

[19] Z. Li, C. Bright, V. Ganesh, A SAT solver and computer algebra attack on the minimum Kochen–Specker problem, in: Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, February 20-27, 2024, Vancouver, Canada, AAAI Press, 2024, pp. 23130–23131. Student Abstract.

[20] M. Kirchweger, S. Szeider, Computing small rainbow cycle numbers with SAT modulo symmetries (short paper), in: P. Shaw (Ed.), 30th International Conference on Principles and Practice of Constraint Programming, CP 2024, September 2-6, 2024, Girona, Spain, volume 307 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, pp. 37:1–37:11. doi:10.4230/LIPICS.CP.2024.37.

[21] K. Fazekas, A. Niemetz, M. Preiner, M. Kirchweger, S. Szeider, A. Biere, IPASIR-UP: user propagators for CDCL, in: M. Mahajan, F. Slivovsky (Eds.), 26th International Conference on Theory and Applications of Satisfiability Testing, SAT 2023, July 4-8, 2023, Alghero, Italy, volume 271 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, pp. 8:1–8:13. doi:10.4230/LIPICS.SAT.2023.8.

[22] DIMACS, The second DIMACS implementation challenge: 1992-1993. NP hard problems: Maximum clique, graph coloring, and satisfiability, 1992. http://dimacs.rutgers.edu/Challenges.

[23] M. Heule, O. Kullmann, S. Wieringa, A. Biere, Cube and conquer: Guiding CDCL SAT solvers by lookaheads, in: K. Eder, J. Lourenço, O. Shehory (Eds.), Hardware and Software: Verification and Testing - 7th International Haifa Verification Conference, HVC 2011, Haifa, Israel, December 6-8, 2011, Revised Selected Papers, volume 7261 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 50–65. doi:10.1007/978-3-642-34188-5_8.