A SAT Solver + Computer Algebra Attack on the Minimum Kochen–Specker Problem

Zhengyu Li[1⋆], Curtis Bright[2], and Vijay Ganesh[3]

[1] University of Waterloo, Canada, brian.li@uwaterloo.ca
[2] University of Windsor, Canada, cbright@uwindsor.ca
[3] University of Waterloo, Canada, vganesh@uwaterloo.ca

Abstract. One of the most fundamental results in the foundations of quantum mechanics is the Kochen–Specker (KS) theorem (also known as the Bell–Kochen–Specker theorem), which essentially states that *contextuality* is an inevitable feature of any hidden-variable theory whose predictions agree with quantum mechanics. The theorem hinges on the existence of a mathematical object called a KS vector system. Although the existence of a KS vector system was first established by Kochen and Specker, the problem of the minimum size of such a system has stubbornly remained open for over 55 years. In this paper, we present a new method based on a combination of a SAT solver and a computer algebra system (CAS) to address this problem. Our SAT+CAS approach improves the lower bound on the minimum number of vectors in a KS system from 22 to 23, and, more importantly, is three orders of magnitude more efficient when compared to the previous best computational methods. Finding a minimum KS system would simplify experimental tests of the KS theorem and have direct applications in quantum information processing, specifically in the security of quantum cryptographic protocols based on complementarity, zero-error classical communication, and dimension witnessing.

Keywords: Satisfiability (SAT) solving · symbolic computation · symmetry breaking · isomorph-free generation · Kochen–Specker systems

1 Introduction

Quantum Mechanics (QM) is often described as one of the most successful physical theories of all time, and yet many questions regarding the foundations of QM continue to be hotly contested. Many interpretations of QM, i.e., mappings from mathematical formalisms of QM to physical phenomena, have been proposed in order to resolve these foundational questions. Perhaps the most well-studied among them are the Copenhagen interpretation, hidden-variable, many-worlds, and quantum information theories [52]. Of these, hidden-variable theories are attempts to understand counterintuitive QM phenomena through a *realist* and deterministic lens by positing the existence of unobservable entities or hidden

⋆ Corresponding author

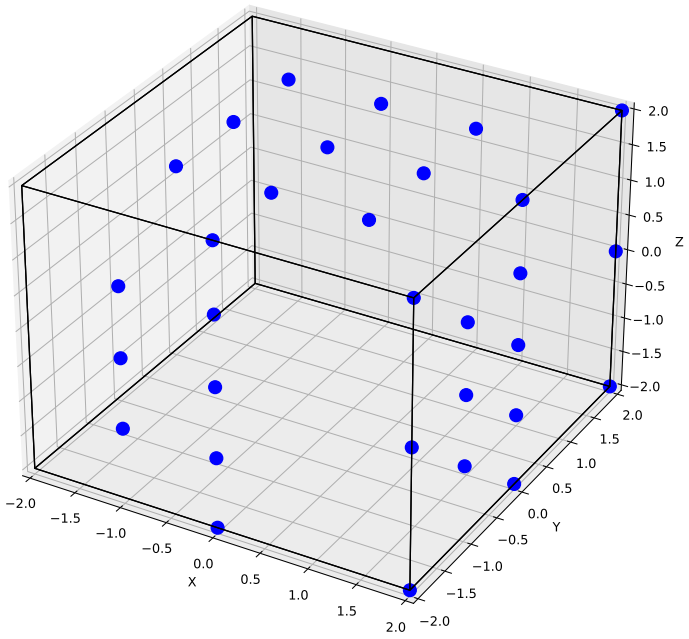


Fig. 1: The 31 vectors of the smallest known KS system (discovered by John Conway and Simon Kochen around 1990).

variables [31] that the standard QM theory does not account for (and hence is deemed incomplete).

Over the years, many constraints have been imposed on hidden-variable theories, e.g., Bell's inequalities that rule out the possibility of *local* hidden-variable theories that are also in agreement with the predictions of QM [8]. In a similar vein, in 1967, Kochen and Specker [45] (and independently, Bell [7] in 1966) proved their famous theorem that essentially asserts that non-contextual hidden variable theories cannot reproduce the empirical predictions of quantum mechanics—thus severely limiting the kind of hidden-variable theories one can posit. Contextuality refers to a peculiar feature of QM observables, wherein any value assignment to all observables must depend on the measurement context and how those observables are measured.

Authors	Year	KS
Kochen, Specker	1967	≤ 117
Jost	1976	≤ 109
Conway, Kochen	1990	≤ 31
Arends, Ouaknine, Wampler	2009	≥ 18
Uijlen, Westerbaan	2016	≥ 22
Li, Bright, Ganesh	2022	≥ 23

Table 1: A chronology of the bounds on the size of the minimum KS system.

1.1 Motivation for Finding The Minimum Kochen–Specker System

As stated above, the Kochen–Specker (KS) theorem rules out noncontextual hidden-variable theories by establishing the existence of a set of three-dimensional vectors (simply referred to as a KS system) that witnesses a contradiction between the assumption of non-contextuality (i.e., observables can be assigned values even prior to measurement and independent of measurement context) and the SPIN axiom of QM. The KS vector system found by Kochen and Specker in 1967 contains 117 vectors [45].

Since the publication of Kochen and Specker's theorem in 1967, physicists and mathematicians have wondered how many vectors the smallest-sized KS vector system contains (see Table 1). This problem of finding the minimum-sized set of three-dimensional[4] KS vectors is often referred to as the (minimum) KS problem. Around 1990, Kochen and Conway found a KS vector system of order 31 via purely mathematical means (see Figure 1) and this is currently the smallest known KS vector system.

Subsequently, many mathematicians and computer scientists have worked on lower bounds to the minimum KS problem via a combination of mathematical and computational approaches. In 2009, Arends et al. provided a computer-aided proof that the smallest KS system must have at least 18 vectors via a reduction of the minimum KS problem to a graph colouring problem [5]. This lower bound was improved to 22 by Uijlen and Westerbaan, who used a combination of analytical results and several graph software packages [61] and this is remains the best known lower bound until our result presented in this paper.

Finding the minimum KS system is not only of scientific and historical interest, but also has direct applications in quantum information processing [20]. So far, the large size of all known KS systems has prevented physicists from using them for applications, including empirical tests of the KS theorem. Finding a minimum KS system could enable applications in the security of quantum cryptographic protocols based on complementarity [18], zero-error classical communication [24], and dimension witnessing [29].

[4] For readability and brevity, we shall leave out the term "3-dimensional" in the rest of the paper.

1.2 SAT Solvers and Combinatorial Problems

In recent years, Boolean satisfiability (SAT) solvers have been widely used to solve a variety of combinatorial mathematical problems [10, 34]. Given that the minimum KS problem is fundamentally combinatorial in nature, and given that there is a reduction from the KS problem to graph colouring—an NP-complete problem for which SAT solvers have been shown to be effective [12]—it is only natural to consider a reduction from the KS problem to the SAT problem. Such an approach enables a SAT solver to aid in the determination of the cardinality of the minimum-sized KS system.

SAT is one of the most influential problems in computer science and mathematics, and it has been studied extensively since it was shown to be NP-complete by Stephen Cook in 1971 [23]. Over the last two decades, thanks to highly efficient conflict-driven clause-learning (CDCL) SAT solving algorithms, we are now able to solve Boolean formulas, derived from real-world applications or combinatorial problems, with tens or even hundreds of millions of variables and clauses efficiently [27]. Even more surprisingly, SAT solvers frequently outperform special-purpose algorithms designed for software engineering [19], verification [21], AI planning [43], and combinatorial mathematics [15].

1.3 The SAT+CAS Paradigm for Combinatorial Mathematics

Despite these fantastic achievements, SAT solvers struggle on certain problems such as those containing many symmetries [12] or those requiring the usage of more advanced mathematical theories than propositional logic [2, 64]. Much work has been done to remedy these drawbacks, including the development of sophisticated symmetry breaking techniques [3, 4] and the development of solvers that support richer logic [6] ("SAT modulo theories" or SMT solvers). However, the mathematical support of SMT solvers is quite primitive when compared with the vast mathematical functionality available in a modern computer algebra system (CAS).

In response to this need for a solver that combines the efficient search capabilities of SAT solvers with the mathematical knowledge available in CAS, a new kind of solving methodology was developed in 2015 by Zulkoski, Ganesh and others [63, 64] and independently by Abraham [1], and has been further expanded since then by Bright et al. [13, 15]. This "SAT+CAS" solving methodology has been successfully applied to many diverse problems, including circuit verification [42, 49], automatic debugging [48], finding circuits for matrix multiplication [33], computing directed Ramsey numbers [51], and finding special kinds of sequences and matrices [14, 16].

In this paper, we use the SAT+CAS solving methodology to dramatically improve the performance of searching for KS systems compared to an out-of-the-box SAT solver and all previous approaches developed to prove lower bounds in the minimum KS problem. More precisely, our SAT+CAS approach is over 1000 times more efficient than the previous best approach by Uijlen and Westerbaan [61] for the minimum KS problem. This is made possible by combining the powerful

search and learning algorithms used in modern SAT solvers with an "isomorph-free exhaustive generation" approach preventing duplicate exploration of isomorphic parts of the search space. Such an approach was recently used to resolve Lam's problem from projective geometry of confirming the nonexistence of a projective plane of order ten [10, 15]. Although isomorph-free exhaustive generation has been extensively used in combinatorial enumeration, it has only recently been combined with SAT-based methods [41, 58].

1.4 Our Contributions

In this paper, we leverage and refine the SAT+CAS paradigm [1,64] to incorporate the isomorph-free generation method of orderly generation (as part of a new SAT+CAS tool, called PhysicsCheck), to obtain tighter lower bounds on the minimum KS problem with three orders of magnitude speedup over previous computational methods developed for this problem.[5]

Specifically, we implement a robust push-and-run pipeline that incorporates a parallel version of MapleSAT [46] coupled with an SMT solver (Z3 [25]) to solve nonlinear real systems and a computer algebraic isomorph-free generation method. We also describe new encoding techniques that enabled an efficient reduction of the minimum KS problem into the Boolean satisfiability problem.

Our new approach establishes that the lower bound for the minimum KS system is 23, as opposed to the previous best of 22. Moreover, we discover missing candidates from Uijlen and Westerbaan's search [61] in order 20, solidifying previous search results (see Section 7). In addition, our approach is over 1000 times more efficient than the previous best approach by Uijlen and Westerbaan.

In order to make the paper relatively self-contained, we provide a thorough background on the KS problem (Section 2) and previous work (Section 3). We motivate our SAT encoding of the KS problem in Section 4, describe our usage of an SMT solver in Section 5, and provided a detailed explanation of our orderly generation technique in the context of the SAT+CAS method in Section 6. We also provide extensive comparison of our results and runtime with previous work (Section 7).

2 Background

In this section we introduce the SPIN axiom, 010-colourability, the KS theorem, and the KS vector system. For a deeper dive into these topics, please refer to the Stanford Encyclopedia of Philosophy [31], and the article "Quantum Contextuality" [17] by Budroni, Cabello, Gühne, Kleinman, and Larsson on the current status, proofs, experimental testings, and applications of contextuality.

[5] We provide an easy-to-use open source repository (`https://github.com/BrianLi009/PhysicsCheck`) for readers to reproduce our results.

2.1 The KS Theorem

Informally, the KS theorem states that there is a contradiction between standard quantum mechanics and certain hidden variable interpretations of quantum phenomena. That is, no non-contextual hidden-variable model can reproduce the predictions of quantum theory in three or more dimensions. Formally, the KS theorem states that by accepting the theory of QM and rules of logical deduction, one of the two premises must be renounced:

- All observables of QM have definite values at all times (Value Definiteness)
- The value of an observable of a QM system are independent of any measurement context and how those properties are measured (Non-contextuality)

Spin of an Elementary Particle: In QM, spin is an intrinsic and discrete form of angular momentum carried by elementary particles. Its existence can be inferred from the Stern–Gerlach experiment [28]. In the context of this paper, a spin-1 particle is shot through a fixed inhomogeneous magnetic field and remains non-deflected, deflects up, or deflects down, corresponding to 3 possible angular momentum states, namely 0, 1, and -1. Thus, the squared result of such measurements along any direction is always 0 or 1. The **SPIN axiom** states that given three pairwise orthogonal directions of measurement, the squared spin components of a spin-1 particle are 1, 0, 1 in these three directions. Thus, the observable corresponding to the question "is the squared spin 0?" measured in three mutually orthogonal directions will always produce *yes* (or 1) in exactly one direction and *no* (or 0) in the other two orthogonal directions in 3-dimensional Euclidean space. The SPIN axiom follows from the postulates of quantum mechanics and is experimentally verifiable [37].

KS Vector System: A KS vector system can be represented in multiple ways—we describe it as a finite set of points on a sphere. As a consequence of the SPIN axiom, the squared-spin measurements along opposite directions must yield the same outcome, therefore we can restrict the domain to the closed northern hemisphere. To define a KS vector system, we first formally define a vector system and the notion of 010-colourability. For the purposes of this paper, we limit ourselves to the 3-dimensional version of the KS problem as the size of the minimum Kochen–Specker system in dimension 4 and higher has already been found [53], while the three dimensional case is still open despite extensive attempts as stated in section 1.

Definition 1. *A **vector system** is a finite subset of the closed northern hemisphere.*

Definition 2. *A vector system is **010-colourable** if there exists an assignment of 0 and 1 to each vector such that:*

1. *No two orthogonal vectors are assigned to 1.*
2. *Three mutually orthogonal vectors are not all assigned to 0.*

Definition 3. *The term Kochen–Specker (KS) vector system refers to a vector system that is not 010-colourable.*

Exhibiting the existence of a KS vector system proves the KS theorem, which states that the closed northern hemisphere is not 010-colourable.

Definition 4. *For a vector system $\mathcal{K}$, define its* ***orthogonality graph*** $G_{\mathcal{K}} = (V, E)$*, where* $V = \mathcal{K}$*,* $E = \{ (v_1, v_2) : v_1, v_2 \in \mathcal{K}$ *and* $v_1 \cdot v_2 = 0 \}$*.*

Essentially, the vertices of $G_{\mathcal{K}}$ are the vectors in $\mathcal{K}$, and there exists an edge between two vertices if and only if their corresponding vectors are orthogonal. Then similarly, we can reduce the notion of 010-colourability from a vector system to an orthogonality graph.

Definition 5. *A graph G is* ***010-colourable*** *if there is a $\{0, 1\}$-colouring of the vertices such that the following two conditions are satisfied simultaneously:*

1. *No two adjacent vertices are coloured 1.*
2. *For each triangle in G, the vertices are not all coloured 0.*

Given an arbitrary graph, it is not guaranteed that there exists a corresponding vector system. If a graph has a corresponding vector system, we say that the graph is embeddable.

Definition 6. *A graph $G = (V, E)$ is* ***embeddable*** *if it is a subgraph of an orthogonality graph $G_{\mathcal{K}}$ for some vector system $\mathcal{K}$.*

Essentially, being embeddable implies the existence of a vector system $\mathcal{K}$ whose vectors have a one-to-one correspondence with the vertices of G in such a way that the adjacent vertices are assigned to orthogonal vectors. For an embeddable graph G, it is not necessary for non-adjacent vertices of G to go to non-orthogonal vectors, since G could be a subgraph of an orthogonality graph with some edges removed (by Definition 6). However, it is necessary for distinct vertices to be mapped to distinct vectors. An example of an unembeddable graph would be the cycle graph of order 4, as the orthogonality constraints would force a pair of opposite vertices of C_4 to be mapped to two collinear vectors.

Definition 7. *An embeddable and non-010-colourable graph is called a* ***KS graph****. Note that every KS vector system can be reduced to a KS graph.*

2.2 Minimum KS Problem

Definition 8. *The cardinality or size of a KS vector system (respectively, a KS graph) is the number of vectors (respectively, the number of nodes) in it.*

The minimum KS problem is defined as the problem of finding the minimum cardinality KS system (respectively, KS graph). That is, with the fewest number of vectors in 3-dimensional space (respectively, with the fewest number of vertices). Every KS system has a KS graph, so if a KS graph with cardinality n does not exist then the lower bound on the minimum KS problem is at least $n + 1$.

2.3 Conjunctive Normal Form

We assume that the reader is familiar with Boolean logic [38] and SAT solvers [36]. We provide a brief description here of Conjunctive Normal Form (CNF) for Boolean formulas. A literal is a propositional variable or its negation. A clause is a disjunction of literals. A formula in CNF is a conjunction of clauses. A formula is said to be satisfiable if there is a variable assignment such that every clause has at least one literal assigned true. Modern SAT solvers only accept Boolean formulas in CNF.

2.4 Cube-and-conquer

The cube-and-conquer satisfiability solving paradigm was developed [35] to solve hard combinatorial problems. The method applies two types of SAT solvers in two stages: First, a "cubing solver" splits a SAT instance into a large number of distinct subproblems specified by cubes—formulas of the form $x_1 \wedge \cdots \wedge x_n$ where x_i are literals. Second, for each cube a "conquering solver" solves the original instance under the assumption that the cube is true. The cube-and-conquer method tends to be effective at quickly solving large satisfiability instances when the cubing solver can generate many cubes encoding subproblems of similar difficulty and solving time. It has since been applied to solve huge combinatorial problems such as the Boolean Pythagorean triples problem [34], the computation of Schur number five [32], and a SAT-based resolution of Lam's problem [10].

3 Related Work

Over the last 50+ years, many mathematicians and physicists such as Roger Penrose, Asher Peres, and John Conway have attempted to find a minimum three-dimensional KS system (see Table 1). Kochen and Specker [45] found the first KS system in 1967 that contained 117 vectors. A KS system with 109 vectors was found by Jost [39] in 1976. The current smallest known KS system contains 31 vectors and was discovered by John Conway and Simon Kochen around 1990. They did not publish their discovery, but they communicated it to Peres [55], who found a more symmetric system of 33 vectors [56]. Shortly thereafter, Penrose [54, 62] found another system of 33 vectors.

In 2011, Arends, Ouaknine, and Wampler [5] proved several interesting properties of KS graphs and leveraged them to computationally prove that a KS system must contain at least 18 vectors. Seven years later, Uijlen and Westerbaan [61] showed that a KS system must have at least 22 vectors. This computational effort used around 300 CPU cores for three months and relied on the *nauty* software package [50] to exhaustively search for KS vector systems. Uijlen and Westerbaan's approach takes some properties of the KS system into consideration during computation, and marks the beginning of computation-intensive resolution to the KS problem. Pavičić, Merlet, McKay, and Megill [53] have improved a slight variation of the KS problem, proving that a KS system in which each vector

is part of a mutually orthogonal triple must have at least 30 vectors. They also studied higher-dimensional generalisations of the KS problem. However, despite these extensive searches, the gap between the lower and upper bounds remains significant and the minimum size of a 3-dimensional KS system remains unknown.

4 SAT Encoding of the Minimum KS Problem

A KS vector system $\mathcal{K}$ can be converted into a KS graph $G_{\mathcal{K}}$. Each vector in $\mathcal{K}$ is assigned to a vertex in $G_{\mathcal{K}}$, so that if two vectors are orthogonal, then their corresponding vertices are connected. Therefore, to find a KS vector system, it is sufficient to find a Kochen–Specker graph. A KS graph is minimal if the only subgraph that is a KS graph is itself. Arends, Ouaknine, and Wampler [5] proved that a three-dimensional minimal KS graph must satisfy the following properties:

1. The graph must not contain a subgraph isomorphic to C_4.
2. Each vertex of the graph must have minimum degree 3.
3. Every vertex is part of a triangle graph C_3.

We encode these three properties above and the non-010-colourability of the KS graph in conjunctive normal form (CNF). If a SAT solver run on this encoding produces solutions, then these solutions are equivalent to graphs that satisfy all four properties.

A simple undirected graph of order n has $\binom{n}{2}$ potential edges, and we represent each edge as a Boolean variable. The edge variable e_{ij} is true exactly when the vertices i and j are connected, where $1 \leq i < j \leq n$. For convenience, we let both e_{ij} and e_{ji} denote the same variable. We also use the $\binom{n}{3}$ triangle variables t_{ijk} denoting that distinct vertices i, j, and k are mutually connected. In Boolean logic this is expressed as $t_{ijk} \leftrightarrow (e_{ij} \wedge e_{ik} \wedge e_{jk})$ which in conjunctive normal form is expressed via the four clauses $\neg t_{ijk} \vee e_{ij}$, $\neg t_{ijk} \vee e_{ik}$, $\neg t_{ijk} \vee e_{jk}$, and $\neg e_{ij} \vee \neg e_{ik} \vee \neg e_{jk} \vee t_{ijk}$. Again, the indices i, j, and k of the variable t_{ijk} may be reordered arbitrarily for notational convenience.

4.1 Encoding the Squarefree Constraint

To encode the property that a Kochen–Specker graph must be squarefree, we construct encodings that prevent the existence of any squares in the graph. Three squares can be formed on four vertices. Therefore, for each choice of four vertices i, j, k, l, we use the three clauses

$$\neg e_{ij} \vee \neg e_{jk} \vee \neg e_{kl} \vee \neg e_{li}, \quad \neg e_{ij} \vee \neg e_{jl} \vee \neg e_{lk} \vee \neg e_{ki}, \quad \neg e_{il} \vee \neg e_{lj} \vee \neg e_{jk} \vee \neg e_{ki}.$$

By enumerating over all possible choices of four vertices and constructing the above CNF formula, we force the graph to be squarefree.

4.2 Encoding the Minimum Degree Constraint

For each vertex i, to ensure that i is connected to at least three other vertices, we take each subset S of $\{1,\dots,i-1,i+1,\dots,n\}$ with cardinality $n-3$ and construct the clause $\bigvee_{j\in S} e_{ij}$. By enumerating over all such subsets we enforce a minimum degree of 3 on vertex i. Thus, constructing similar formulae for all vertices $1 \le i \le n$, enforces that any vertex in the graph has a degree of at least 3.

4.3 Encoding the Triangle Constraint

Now we encode the property that every vertex is part of a triangle. For each vertex i, we require 2 other distinct vertices to form a triangle, and there are $\binom{n-1}{2}$ possible triangles containing i. At least one of those triangles must be present in the graph—this is ensured by the clause $\bigvee_{j,k\in S} t_{ijk}$ where S is $\{1,\dots,i-1,i+1,\dots,n\}$ and $j<k$. Using this clause for each $1 \le i \le n$ ensures that every vertex is part of a triangle.

4.4 Encoding the Noncolourability Constraint

We generate clauses to block as many 010-colourable graphs as possible (ideally all of them, leaving only the non-010-colourable graphs). A graph is non-010-colourable if and only if for all $\{0,1\}$-colourings of the graph a pair of colour-1 vertices is connected or a set of three colour-0 vertices are mutually connected. The idea is to consider many $\{0,1\}$-colourings and construct clauses that block the graphs for which those colourings form a 010-colouring.

For each $\{0,1\}$-colouring, we have a set of colour-0 vertices V_0 and a set of colour-1 vertices V_1. Given a specific such colouring, the clause

$$\bigvee_{\substack{i,j\in V_1\\ i<j}} e_{ij} \vee \bigvee_{\substack{i,j,k\in V_0\\ i<j<k}} t_{ijk}$$

enforces that the colouring is not a 010-colouring of the graph since either a pair of colour-1 vertices is connected or a set of three colour-0 vertices is mutually connected.

Due to the large number of possible $\{0,1\}$-colourings, we only consider colourings with less than or equal to $\lceil \frac{n}{2} \rceil$ colour-1 vertices. Colourings with more than $\lceil \frac{n}{2} \rceil$ colour-1 vertices are unlikely to be 010-colourings and in practice were not useful in blocking 010-colourable graphs.

4.5 Encoding Isomorphism Blocking Clauses

We want to block as many isomorphic graphs using a small number of clauses before passing the instance to the SAT solver. Following Codish et al. [22] we use symmetry breaking constraints that enforce a certain lexicographical order between rows of the graph's adjacency matrix. We denote this as the cubic

method as it adds $O(n^3)$ clauses for graphs of order n. Given an adjacency matrix A of a graph, we define $A_{i,j}$ as the ith row of A without columns i and j. Codish et al. prove that up to isomorphism every graph can be represented by an adjacency matrix A for which $A_{i,j}$ is lexicographically equal or smaller than $A_{j,i}$ for all $1 \leq i < j \leq n$.

We express that $A_{i,j} = [x_1, x_2, \ldots, x_n]$ is lexicographically equal or less than $A_{j,i} = [y_1, y_2, \ldots, y_n]$ using $3n-2$ clauses and auxiliary variables $a_1, \ldots, a_{n-1}$ [44]. The clauses are $\neg x_k \vee y_k \vee \neg a_{k-1}$, $\neg x_k \vee a_k \vee \neg a_{k-1}$, and $y_k \vee a_k \vee \neg a_{k-1}$ for $k = 1, \ldots, n-1$. The literal $\neg a_0$ is omitted and the clause $\neg x_n \vee y_n \vee \neg a_{n-1}$ is also included.

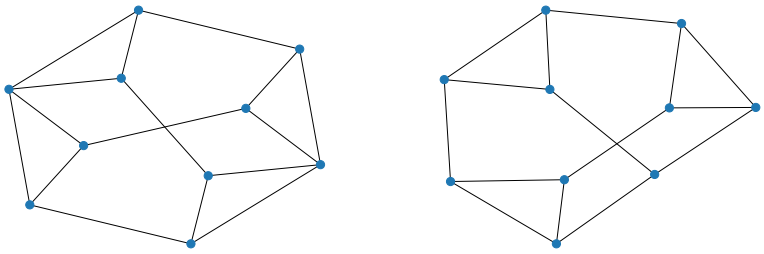
5 Embeddability Checking

We refer to the solutions generated by the SAT solver as *KS candidates* and we check the embeddability of a KS candidate using an SMT solver. A KS candidate that is embeddable is a KS graph and its embedding is a KS system. Our embeddability checking algorithm consists of two parts. The first part is a direct integration of Uijlen and Westerbaan's vector assignment algorithm [61], which finds all possible interpretations to describe the orthogonal relations between the vectors. We define *free vectors* as vectors that have not been fixed as the cross product of two vectors. Of all possible interpretations, we first choose the one with the least number of free vectors, since such an assignment is likely to be solved in the least amount of time. The second part of the algorithm applies an SMT solver to determine the satisfiability of an interpretation. An interpretation generated by Uijlen and Westerbaan's algorithm is converted into a set of cross and dot product equations, and these equations are passed to the theorem prover Z3 [25]. Denoting each vertex in a KS candidate as v_i and its corresponding vector representation as V_i, we form the following constraints:

1. Vector $V_i = (x_i, y_i, z_i) \in \mathbb{R}^3$ has nonnegative z_i.
2. If v_i is connected to v_j and v_k, then $V_i = (V_j \times V_k)$ or $V_i = (V_k \times V_j)$.
3. If v_i are v_j are not connected, then V_i is not collinear to V_j, and $V_i \times V_j \neq \vec{0}$.
4. If v_i and v_j are connected, then V_i and V_j are orthogonal, and $V_i \cdot V_j = 0$.

Each orthogonal relation of the graph is expressed either by constraint 2 or constraint 4, depending on the orthogonality interpretation used. Constraint 3 requires two vectors to be noncollinear rather than only being nonequal, since we do not enforce vectors to have unit length (for reasons of efficiency). This is a harmless optimization, since vectors can be projected onto the unit sphere without disturbing these constraints. To check whether a graph is embeddable, we use Z3 to determine whether these nonlinear arithmetic constraints are satisfiable over the real numbers. Z3 applies a CDCL-style algorithm to decide the satisfiability of such equations [40].

Without loss of generality, to accelerate this process we fix two orthogonal vectors to be the standard vectors $(1, 0, 0)$ and $(0, 1, 0)$. Given a system of equations, Z3 attempts to find a solution for all variables. If a solution is found,



(a) (b)

Fig. 2: The only two minimal nonembeddable graphs of order 10. These are the smallest squarefree graphs that are not embeddable.

it is an assignment of vertices to vectors satisfying all orthogonality constraints—therefore proving the embeddability of the graph. Embeddability checking of large graphs can be further optimized by precomputing minimal unembeddable graphs, as defined below.

Definition 9. *A proper subgraph is a subgraph that removes at least one vertex or edge relative to the whole graph.*

Definition 10. *A graph G is said to be a* ***minimal unembeddable graph*** *if any proper subgraph of G is embeddable.*

A graph is unembeddable if it contains a minimal unembeddable subgraph. Therefore, to optimize embeddability checking, we precomputed all minimal unembeddable squarefree graphs up to order 12. To find all minimal unembeddable subgraphs of size n, we use the PhysicsCheck pipeline with only the squarefree constraints to generate all graphs of order n that are squarefree. We then perform an embeddability check on each graph to determine if it is embeddable. After completing the embeddability check on all smaller orders, we were able to complete the check on order n quickly using the following proposition.

Proposition 1. *A graph is not minimally unembeddable if it has a vertex of degree less than 2.*

We provide the following justification for Proposition 1: If a graph has a vertex of degree less than 2, it either has a vertex of degree 1 or 0. In either case, assuming that it is minimally unembeddable generates a contradiction since its subgraph without the degree-1 or degree-0 vertex must also be unembeddable. We generate all squarefree graphs up to order 12 that have minimum vertex degree 2 and decide their embeddability. The embeddability of most graphs can be determined on the first assignment in less than 1 second. If the satisfiability of an interpretation is not determined within 10 seconds, we move on to a different orthogonality interpretation and attempt the satisfiability check again. This

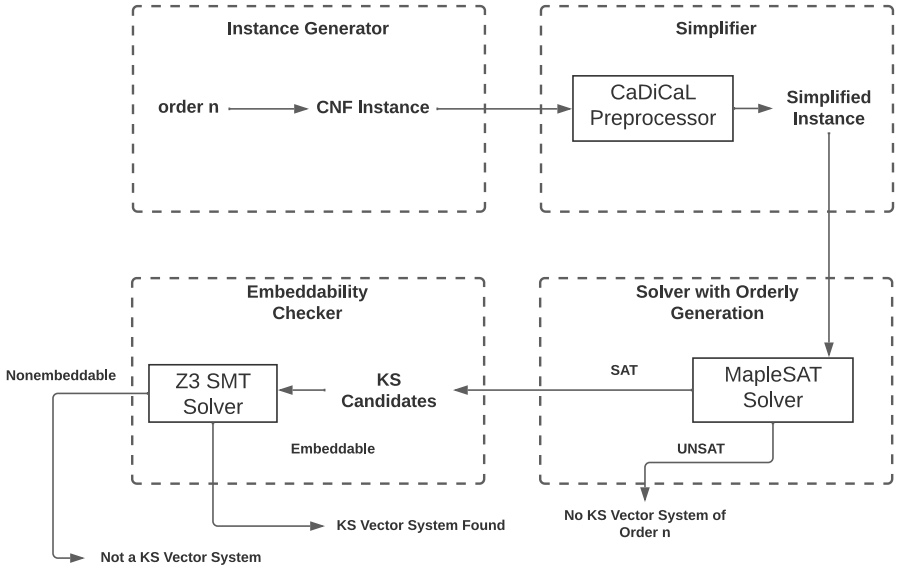


Fig. 3: A flowchart of our method, PhysicsCheck, for solving the KS problem. The instance generator generates the SAT instance, encoding the KS problem, based on the encodings given in Section 4. The formula is then simplified using the CaDiCaL [9] SAT solver. The simplified instance is then passed to the SAT+CAS tool (i.e., MapleSAT with orderly generation) either sequentially or in parallel using the cube-and-conquer technique. Finally, an embeddability checker applies the SMT solver Z3 to determine whether the candidate KS systems found are indeed embeddable.

process continues until we determine the embeddability of a graph. Given a KS candidate, if the candidate contains a minimal unembeddable subgraph, then the candidate must be unembeddable. Using this property significantly speeds up the embeddability checking process for KS candidates, since most candidates contain an unembeddable subgraph of order 10, 11, or 12. We provide the two minimal nonembeddable graphs of order 10 which appears frequently as subgraphs of KS candidates in Figure 2. The result of the embeddability check will be further discussed in Section 7.

6 Implementation

Directly solving the SAT instances with the encoding from Section 4 is feasible only for smaller orders, since the number of graphs in the search increases exponentially with the order. We implement two effective techniques to reduce runtime. One is an orderly generation technique, which generates graphs in the search space up to isomorphism, and the other is a parallelization technique. The integration of these techniques is shown in Figure 3.

Order	Speedup
16	6.5
17	13.6
18	37.8
19	104.5

Table 2: The implementation of SAT+CAS orderly generation provides the above speedup. As the search space increases, orderly generation becomes more effective since the number of isomorphic graphs in the search space is also increasing. We did not provide the speedup factor for order greater than 19, since solving these instances without the orderly generation technique would take too long.

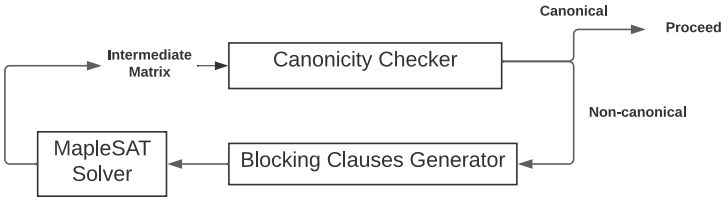


Fig. 4: A flowchart of the orderly generation algorithm implemented as part of PhysicsCheck's SAT+CAS architecture.

6.1 Orderly Generation

Since the cubic isomorphism blocking constraints does not block all isomorphic copies of the graph, the SAT solver still generates many copies of the same graph up to isomorphism. Thus, we use a hybrid SAT and isomorphic-free generation approach. The orderly isomorphic-free generation approach was developed independently by Igor Faradžev [26] and Ronald Read [57] in 1978. To describe it, we first define the following canonical representation of a graph.

Definition 11. *An adjacency matrix M is canonical if every permutation of its rows produces a matrix lexicographically greater than or equal to M, where the lexicographical order is defined by concatenating the above-diagonal entries of the columns of the adjacency matrix in order.*

The *parent* of an $n \times n$ matrix A is the upper-left $(n-1) \times (n-1)$ submatrix of A. Similarly, the *children* of an $n \times n$ matrix A is any extensions of A with size $(n+1) \times (n+1)$ by adding an additional row and column. The orderly generation method is based on the following two consequences of Definition 11:

1. Every isomorphic class of graphs only has one canonical representative.
2. If a matrix is canonical, then its parent is also canonical.

Note that the second property implies that if a matrix is not canonical, then all of its children and grandchildren are not canonical. Therefore, we can reject

all intermediate noncanonical matrices, as they cannot lead to a canonical matrix in the search tree. Orderly generation works by recording intermediate canonical objects and iteratively extending them by a row/column at a time until the matrices have been extended to a full canonical matrix.

As described in Figure 4, in our SAT+CAS implementation, when the SAT solver finds an intermediate matrix the canonicity of this matrix is determined by a canonicity-checking routine implemented using the MathCheck system [11]. If the matrix is noncanonical, then a "blocking" clause is learned which removes this matrix (and all of its children and grandchildren) from the search. Otherwise, the matrix is canonical and the SAT solver proceeds as normal. We combine this process with the symmetry breaking clauses of Codish et al. that canonical matrices can be shown to satisfy [22, Def. 8]. The orderly generation technique provides a speedup which experimentally increases exponentially as the order increases, as demonstrated in Table 2.

As described in Figure 3, we simplify the SAT instance using the SAT solver CaDiCaL [9] before solving the instance using MapleSAT [47]. As a preprocessing step, we also run the orderly generation process on graphs with up to 12 vertices and add the generated blocking clauses directly into the instance provided to CaDiCaL—this allows the simplification to incorporate some of the knowledge derived from the orderly generation process.

6.2 Parallelization

For orders greater than 20, parallelization is applied by dividing the instance into smaller subproblems using the cube-and-conquer approach [35]. The approach applies a lookahead solver [36] to partition a hard problem into many cubes and offers very efficient solving time for some combinatorial problems.

During the partitioning step, a lookahead solver attempts to find the variables that split the search space the most evenly. In our work, March_cu is used as the lookahead solver [59]. The SAT instance is then solved under the assumption that each cube is true, generating many subproblems that can be solved in parallel. We terminate the splitting process once the number of undetermined edge variables in each subproblem drops below a bound specified in advance.

7 Results

Given the CNF file with the encoded constraints, we use the aforementioned techniques combined with the SAT+CAS approach to verify all previous results on KS systems up to order 21 with an orders-of-magnitude speedup factor. Moreover, we improve on the best known lower bound for a minimum KS system (see Table 3). All computations were done on Intel E5-2683 CPUs @ 2.1GHz administrated by Canada's national advanced research computing (ARC) platform, and measured in total CPU time. Our search in order 21 is over 1000 times faster than the previous computational search of Uijlen and Westerbaan which was distributed on approximately 300 CPU cores and took roughly three months [61].

Order	Candidates	Simplification	Cubing	Cube Simplification	Solving
17	1	0.02 hrs	N/A	N/A	0.02 hrs
18	0	0.02 hrs	N/A	N/A	0.13 hrs
19	8	0.31 hrs	N/A	N/A	2.46 hrs
20	147	0.54 hrs	N/A	N/A	39.71 hrs
21	2,497	1.50 hrs	38 hrs	19.4 hrs	1,019 hrs
22	88,282	2.54 hrs	953.7 hrs	253.3 hrs	46,079 hrs

Table 3: A summary of our results in the Kochen–Specker problem on orders $17 \leq n \leq 22$.

We apply cube-and-conquer and naive parallel SAT solving on order 21 and 22 due to the combinatorial explosion caused by the large order. We eliminate 75 edge variables from subproblems in order 21 and 90 edge variables in order 22 during the cubing process.

We simplify the SAT instance of each order by eliminating variables to speed up solving. While simplifying the SAT instance, we aim to eliminate 60% of the variables; otherwise, CaDiCaL stops simplifying after being called 100 times. For order 21 and 22, additional simplification is applied on each subproblem after the cubing process. Specifically, if more than 50% of the variables have been eliminated from the instance already, we only call CaDiCaL once more on each subproblem with a conflict limit of 200,000 since the instance is sufficiently simplified. Otherwise, we decrease the amount of variables to eliminate to 50%, or the simplification terminates if CaDiCaL is called 100 times once again. Some cubes of order 22 with 90 edge variables eliminated define instances that are not solved within 72 hours, so we perform additional cubing on these instances until at least 125 edge variables have been eliminated.

As discussed in Section 5, we precompute all minimal nonembeddable subgraphs with less than 12 vertices. Uijlen and Westerbaan were unable to determine the embeddability of one particular graph of order 14. Using our embeddability checking approach along with the Z3 SMT solver, this graph is quickly shown to be unembeddable. By comparing our sets of minimal unembeddable subgraphs with Uijlen and Westerbaan's online dataset of small graphs[6], we find our minimal unembeddable subgraphs from order 10 to 12 to be identical to theirs up to isomorphism. This provides us with high confidence in the robustness of our embeddability checking pipeline.

We compared our Kochen–Specker candidates with Uijlen and Westerbaan's findings, and have verified their conclusion that there is no KS system with less than 22 vectors. It is natural that we obtained fewer candidates for each order because Uijlen and Westerbaan did not require every vertex of a candidate to be part of a triangle. In order 20, we found four additional KS candidates that were not present in the collection of Uijlen and Westerbaan, indicating their

[6] https://kochen-specker.info/smallGraphs/

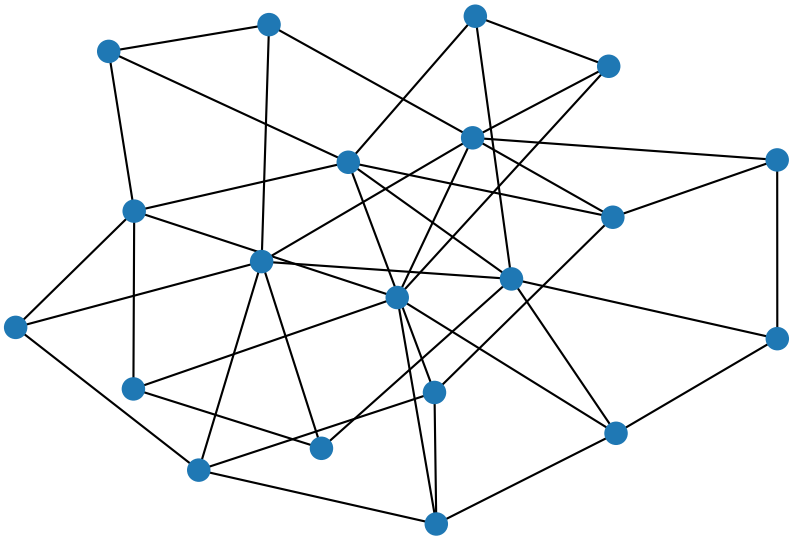


Fig. 5: One of the four graphs with 20 vertices that was not present in Uijlen and Westerbaan's enumeration. The four graphs satisfy all constraints mentioned in Section 4, but are not embeddable, and therefore do not constitute a KS system.

search was incomplete. We present one of the missing graphs in Figure 5. We have verified that these four additional graphs satisfy the constraints of a KS candidate and therefore would be KS systems were they embeddable. Note that not all KS candidates are minimal; some KS candidates contain a KS candidate of smaller order as a subgraph.

All 90,935 KS candidates of order less than 23 are not embeddable.[7] The embeddability check is done quickly since all candidates (with the exception of a single candidate in order 22) contain a minimal nonembeddable subgraph of orders 10–12 (summarized in Table 4). Therefore, we conclude that the minimum size of the KS system is at least 23.

7.1 Correctness of Results

It is natural to ask how confident we are that our pipeline and results are correct. Whenever a complex set of encoders, solvers, CAS, and provers are used to prove a mathematical result, it is important to have reasonable verification procedures in place to ensure the correctness of said result. A complete verifiable (machine-checkable) proof of our result is infeasible at this stage given the complexity

[7] We provide another open source repository (`https://github.com/BrianLi009/PhysicsCheck_log`) for log files, results, and additional scripts used to run PhysicsCheck on Canada's national advanced research computing (ARC) platform.

Order	Canonical squarefree	Minimal unembeddable	Runtime
10	5,069	2	2.39 hrs
11	25,181	5	9.07 hrs
12	152,045	10	78.8 hrs

Table 4: Counts for the number of canonical squarefree graphs, the number of minimal unembeddable graphs in order 10–12, and the computation time to conduct embeddability check on canonical squarefree graphs.

of the CAS and solvers used and the difficulty of proving their correctness via existing theorem provers or proof assistants. Further, while solvers are known to produce verifiable proofs of UNSAT results, that by itself is not sufficient to provide a high degree of confidence in a pipeline such as ours. We need a different way to approach this problem.

Our approach is built on three ideas: first, extensive testing of all parts of our systems; second, extensive end-to-end tests; and finally, and most importantly, cross-verification of our pipeline against previously known results by Uijlen and Westerbaan. These significant measures that we have taken considerably improve our confidence in the results presented here.

Providing more context, we used our entire pipeline to cross-verify all KS candidates from order 17 to 21 (except for the four new candidates that we discovered) are isomorphic to the ones discovered by Uijlen and Westerbaan using SageMath [60] and the NetworkX [30] graph theory package, a very different set of systems from the ones we used.

We also conducted extensive cross-verification on the results (KS candidates) produced by the SAT solver. For example, each solver-generated candidate is passed into a verification script implemented using the NetworkX [30] graph package to verify that they satisfy all encoded constraints (see Section 4), namely the squarefree constraint, the minimum degree constraint, the triangle constraint, and the noncolourability constraint. Moreover, we test the embeddability pipeline by conducting verification on all embeddable subgraphs found in orders 10 to 12. Specifically, if a graph is embeddable and corresponds to a set of vectors, we check that no pair of vectors in the set are collinear, and a pair of vectors are orthogonal if their corresponding vertices are connected. These extensive cross-verification steps provide us with high confidence in the robustness of the PhysicsCheck pipeline.

8 Conclusion

In this paper, we improve the lower bound of the minimum KS problem from 22 to 23, and provide a computational speedup by three orders of magnitude over the previous best approach by Uijlen and Westerbaan (see Section 7). Moreover, we leverage the new SAT+CAS paradigm along with orderly isomorph-free

generation to provide a robust pipeline for the minimum KS problem. Compared to previous work, our approach is less error-prone as it reduces the need for custom-purpose search algorithms. Instead, we use heavily-tested SAT solvers such as MapleSAT. Moreover, our method has extensive cross-verification in place (see Section 7.1).

Finding the minimum KS system is not only a problem of great importance to the foundations of quantum mechanics, but has direct applications to various fields of quantum information processing, such as quantum cryptographic protocols [18], zero-error classical communication [24], and dimension witnessing [29]. As a consequence, a wide variety of techniques have been developed to address this question over the past several decades. We add a novel class of techniques to this body of work.

As previously mentioned, the SAT+CAS paradigm has been successfully used to resolve a number of mathematical problems in combinatorics, number theory, and geometry that had previously remained unsolved for many decades. With this work we extend the reach of the SAT+CAS paradigm, for the first time, to resolving combinatorial questions in the realm of foundations of quantum mechanics.

9 Methods

9.1 Exhaustive Generation of KS Candidates.

Generating KS candidates of order n is an exhaustive task, meaning that we want the SAT solver to output all possible solutions of the SAT instance. To accomplish that, when we find a solution to the instance, we add a "blocking" clause blocking this solution from the instance. We repeat such a procedure until the instance is not satisfiable (UNSAT), which means that we have generated all solutions exhaustively.

9.2 Cube-and-conquer and Parallelization.

We use March_cu as the cubing solver on the instance while specifying a certain number of edge variables to eliminate. This procedure splits the instance into subproblems by generating a set of cubes. To formulate each subproblem, we adjoin the previously simplified instance with an individual cube, in other words, the cube is being added as unit clauses in the instance. In our paper, each subproblem is run on an individual core, though readers can customize parallelization strategies based on the computational resources available.

9.3 Uijlen and Westerbaan's Vector Assignment Algorithm

As described in Section 5, we use Uijlen and Westerbaan's vector assignment algorithm in our embeddability checking pipeline. The algorithm first fixes some vectors, then derives cross-product relations for the remaining vectors, and finally,

it creates dot-product equations to relate pairs of cross-product expressions. Depending on the choices of vectors in each step, different interpretations are generated. The pseudocode is provided in Uijlen and Westerbaan's paper, and Python code is made available by them as well.

9.4 Running PhysicsCheck for the KS Problem

We provide a detailed overview of the methodology of our work. To find all KS systems of order n, we first generate constraints in Section 4 for order n and adjoin them together to form a SAT instance. Then we add the blocking clauses generated by running orderly generation on graphs with up to 12 vertices to the instance. The instance is then simplified using CaDiCaL until more than 60% of the edge variables are eliminated, or until CaDiCaL is called 100 times. If cube-and-conquer is enabled, we proceed with the cubing process on the instance by specifying the number of variables to eliminate in every cube, then further simplify and solve each subproblem. Otherwise, we pass the instance directly to MapleSAT. Assuming that MapleSAT outputs some KS candidates, we proceed to check the embeddability of each candidate. If a candidate contains one of the precomputed minimal unembeddable subgraphs, we immediately conclude that this candidate is unembeddable and not a KS system. If no minimal unembeddable subgraph is found in a candidate, we then call the script to generate an orthogonality assignment and use Z3 to determine the satisfiability of the corresponding system of equations.

Our repository provides a driver script connecting all components mentioned above. Each component of PhysicsCheck can be run independently; specifically, users can call the instance generator, instance simplifier, MapleSAT solver (with orderly generation), cube-and-conquer, and the embeddability checker. Users can also customize the parameters of PhysicsCheck to optimize the runtime. Some key parameters include the amount of simplification, when to simplify, and the number of edge variables to eliminate during cubing. We discuss the exact parameters used in the KS problem in Subsection 9.5. If the instances associated with some cubes time out, those cubes can be adjoined into a single file and 'cube.sh' can be used to split those cubes deeper by increasing the number of edge variables to remove.

9.5 Parameters Used for PhysicsCheck

Here we describe the exact parameters used to generate the results in Section 7. While running the driver script 'main.sh', we set the order of KS systems to solve for (using parameter n) and simplify until 60% of the edge variables are eliminated (using parameter o). We call CaDiCaL after adding the noncanonical blocking clauses (using parameter s). We also generate these noncanonical blocking clauses in real-time (using parameter b). For order 21 and 22, cube-and-conquer is enabled before calling MapleSAT. We eliminate 75 variables for order 21 and 90 variables for order 22 (using parameter r). Finally, we set parameter p to 1 to enable parallel cubing. All KS candidates of order n are generated to the

file n.exhaust by MapleSAT, and running the check_embedability.sh script will call the embeddability check on every candidate in n.exhaust. Since all minimal unembeddable subgraph of order 10 to 12 have been pre-generated, the script will first check whether each candidate contains any of the minimal unembeddable subgraph. While calling check_embedability.sh, we set parameter n to be the order, then enable minimal nonembeddable subgraph (using parameter s).

Author declarations

The authors declare no competing interests.

References

1. Ábrahám, E.: Building bridges between symbolic computation and satisfiability checking. In: Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation. pp. 1–6 (2015). https://doi.org/10.1145/2755996.2756636
2. Ábrahám, E., Kremer, G.: Satisfiability checking: Theory and applications. In: International Conference on Software Engineering and Formal Methods. pp. 9–23. Springer (2016). https://doi.org/10.1007/978-3-319-41591-8_2
3. Aloul, F.A., Ramani, A., Markov, I.L., Sakallah, K.A.: Solving difficult SAT instances in the presence of symmetry. In: Proceedings 2002 Design Automation Conference (IEEE Cat. No. 02CH37324). pp. 731–736. IEEE (2002). https://doi.org/10.1109/DAC.2002.1012719
4. Aloul, F.A., Sakallah, K.A., Markov, I.L.: Efficient symmetry breaking for Boolean satisfiability. IEEE Transactions on Computers **55**(5), 549–558 (2006). https://doi.org/10.1109/TC.2006.75
5. Arends, F., Ouaknine, J., Wampler, C.W.: On searching for small Kochen-Specker vector systems. In: International Workshop on Graph-Theoretic Concepts in Computer Science. pp. 23–34. Springer (2011). https://doi.org/10.1007/978-3-642-25870-1_4
6. Barrett, C., Fontaine, P., Tinelli, C.: The Satisfiability Modulo Theories Library (SMT-LIB). www.SMT-LIB.org (2016)
7. BELL, J.S.: On the problem of hidden variables in quantum mechanics. Rev. Mod. Phys. **38**, 447–452 (Jul 1966). https://doi.org/10.1103/RevModPhys.38.447, https://link.aps.org/doi/10.1103/RevModPhys.38.447
8. Bell, J.S., Bell, J.S.: Speakable and unspeakable in quantum mechanics: Collected papers on quantum philosophy. Cambridge university press (2004). https://doi.org/10.1017/CBO9780511815676
9. Biere, A., Fazekas, K., Fleury, M., Heisinger, M.: CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In: Balyo, T., Froleyks, N., Heule, M.J.H., Iser, M., Järvisalo, M., Suda, M. (eds.) Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions. Department of Computer Science Report Series B, vol. B-2020-1, pp. 51–53. University of Helsinki (2020)
10. Bright, C., Cheung, K.K.H., Stevens, B., Kotsireas, I., Ganesh, V.: A SAT-based resolution of Lam's problem. In: Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence. pp. 3669–3676 (2021), https://ojs.aaai.org/index.php/AAAI/article/view/16483

11. Bright, C., Ganesh, V., Heinle, A., Kotsireas, I., Nejati, S., Czarnecki, K.: MathCheck2: A SAT+CAS verifier for combinatorial conjectures. In: International Workshop on Computer Algebra in Scientific Computing. pp. 117–133. Springer (2016). https://doi.org/10.1007/978-3-319-45641-6_9
12. Bright, C., Gerhard, J., Kotsireas, I., Ganesh, V.: Effective problem solving using SAT solvers. In: Maple in Mathematics Education and Research, Communications in Computer and Information Science, vol. 1125, pp. 205–219. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-41258-6_15
13. Bright, C., Kotsireas, I., Ganesh, V.: SAT solvers and computer algebra systems: A powerful combination for mathematics. In: Proceedings of the 29th International Conference on Computer Science and Software Engineering. pp. 323–328 (2019), https://dl.acm.org/doi/abs/10.5555/3370272.3370309
14. Bright, C., Kotsireas, I., Ganesh, V.: Applying computer algebra systems with SAT solvers to the Williamson conjecture. Journal of Symbolic Computation **100**, 187–209 (2020). https://doi.org/10.1016/j.jsc.2019.07.024
15. Bright, C., Kotsireas, I., Ganesh, V.: When satisfiability solving meets symbolic computation. Communications of the ACM **65**(7), 64–72 (Jul 2022). https://doi.org/10.1145/3500921
16. Bright, C., Kotsireas, I.S., Heinle, A., Ganesh, V.: Enumeration of complex Golay pairs via programmatic SAT. In: Proceedings of the 2018 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2018, New York, NY, USA, July 16–19, 2018. pp. 111–118 (2018). https://doi.org/10.1145/3208976.3209006
17. Budroni, C., Cabello, A., Gühne, O., Kleinmann, M., Larsson, J.Å.: Quantum contextuality. arXiv preprint arXiv:2102.13036 (2021). https://doi.org/10.48550/arXiv.2102.13036
18. Cabello, A., D'Ambrosio, V., Nagali, E., Sciarrino, F.: Hybrid ququart-encoded quantum cryptography protected by Kochen-Specker contextuality. Physical Review A **84**(3), 030302 (2011). https://doi.org/10.1103/PhysRevA.84.030302
19. Cadar, C., Ganesh, V., Pawlowski, P.M., Dill, D.L., Engler, D.R.: EXE: Automatically generating inputs of death. ACM Transactions on Information and System Security (TISSEC) **12**(2), 1–38 (2008). https://doi.org/10.1145/1455518.1455522
20. Canas, G., Arias, M., Etcheverry, S., Gómez, E.S., Cabello, A., Xavier, G.B., Lima, G.: Applying the simplest Kochen-Specker set for quantum information processing. Physical review letters **113**(9), 090404 (2014). https://doi.org/10.1103/PhysRevLett.113.090404
21. Clarke, E., Biere, A., Raimi, R., Zhu, Y.: Bounded model checking using satisfiability solving. Formal methods in system design **19**(1), 7–34 (2001). https://doi.org/10.1023/A:1011276507260
22. Codish, M., Miller, A., Prosser, P., Stuckey, P.J.: Constraints for symmetry breaking in graph representation. Constraints **24**(1), 1–24 (Aug 2019). https://doi.org/10.1007/s10601-018-9294-5
23. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the third annual ACM symposium on Theory of computing. pp. 151–158 (1971). https://doi.org/10.1145/800157.805047
24. Cubitt, T.S., Leung, D., Matthews, W., Winter, A.: Improving zero-error classical communication with entanglement. Physical Review Letters **104**(23), 230503 (2010). https://doi.org/10.1103/PhysRevLett.104.230503
25. de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: International conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer (2008). https://doi.org/10.1007/978-3-540-78800-3_24

26. Faradžev, I.: Constructive enumeration of combinatorial objects. In: Problèmes combinatoires et théorie des graphes. pp. 131–135 (1978)
27. Ganesh, V., Vardi, M.Y.: On the unreasonable effectiveness of SAT solvers (2020), https://www.cs.rice.edu/~vardi/papers/SATSolvers21.pdf
28. Gerlāch, W., Stern, O.: Der experimentelle nachweis der richtungsquantelung im magnetfeld. Zeitschrift für Physik **9**, 349–352 (1922). https://doi.org/10.1007/BF01326983
29. Gühne, O., Budroni, C., Cabello, A., Kleinmann, M., Larsson, J.Å.: Bounding the quantum dimension with contextuality. Physical Review A **89**(6), 062107 (2014). https://doi.org/10.1103/PhysRevA.89.062107
30. Hagberg, A.A., Schult, D.A., Swart, P.J.: Exploring network structure, dynamics, and function using NetworkX. In: Varoquaux, G., Vaught, T., Millman, J. (eds.) Proceedings of the 7th Python in Science Conference. pp. 11 – 15. Pasadena, CA USA (2008)
31. Held, C.: The Kochen-Specker Theorem. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, Spring 2018 edn. (2018)
32. Heule, M.: Schur number five. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018). https://doi.org/10.1609/aaai.v32i1.12209
33. Heule, M.J.H., Kauers, M., Seidl, M.: New ways to multiply 3×3-matrices. Journal of Symbolic Computation **104**, 899–916 (May 2021). https://doi.org/10.1016/j.jsc.2020.10.003
34. Heule, M.J.H., Kullmann, O., Marek, V.W.: Solving and verifying the boolean pythagorean triples problem via cube-and-conquer. In: Creignou, N., Le Berre, D. (eds.) Theory and Applications of Satisfiability Testing – SAT 2016. pp. 228–245. Springer International Publishing, Cham (2016)
35. Heule, M.J.H., Kullmann, O., Wieringa, S., Biere, A.: Cube and conquer: Guiding CDCL SAT solvers by lookaheads. In: Haifa Verification Conference. pp. 50–65. Springer (2011). https://doi.org/10.1007/978-3-642-34188-5_8
36. Heule, M.J.H., van Maaren, H.: Look-ahead based SAT solvers. Handbook of satisfiability **185**, 155–184 (2009). https://doi.org/10.3233/FAIA200988
37. Huang, Y.F., Li, C.F., Zhang, Y.S., Pan, J.W., Guo, G.C.: Experimental test of the Kochen-Specker theorem with single photons. Physical Review Letters **90**(25) (Jun 2003). https://doi.org/10.1103/physrevlett.90.250401
38. Huth, M., Ryan, M.: Logic in Computer Science: Modelling and reasoning about systems. Cambridge university press (2004). https://doi.org/10.1017/CBO9780511810275
39. Jost, R.: Measures on the finite dimensional subspaces of a Hilbert space: remarks to a theorem by A. M. Gleason. Studies in Mathematical Physics: Essays in Honour of Valentine Bergmann pp. 209–228 (1976). https://doi.org/10.1515/9781400868940-011
40. Jovanović, D., de Moura, L.: Solving non-linear arithmetic. In: International Joint Conference on Automated Reasoning. pp. 339–354. Springer (2012). https://doi.org/10.1007/978-3-642-31365-3_27
41. Junttila, T., Karppa, M., Kaski, P., Kohonen, J.: An adaptive prefix-assignment technique for symmetry reduction. Journal of Symbolic Computation **99**, 21–49 (Jul 2020). https://doi.org/10.1016/j.jsc.2019.03.002
42. Kaufmann, D., Biere, A., Kauers, M.: Verifying large multipliers by combining SAT and computer algebra. In: 2019 Formal Methods in Computer Aided Design (FMCAD). IEEE (Oct 2019). https://doi.org/10.23919/fmcad.2019.8894250

43. Kautz, H.A., Selman, B., et al.: Planning as satisfiability. In: ECAI. vol. 92, pp. 359–363. Citeseer (1992), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.35.9443&rep=rep1&type=pdf
44. Knuth, D.E.: The art of computer programming, Volume 4, Fascicle 6: Satisfiability. Addison-Wesley Professional (2015), https://dl.acm.org/doi/abs/10.5555/2898950
45. Kochen, S., Specker, E.P.: The Problem of Hidden Variables in Quantum Mechanics. Journal of Mathematics and Mechanics **17**, 59–87 (1967). https://doi.org/10.1007/978-94-010-1795-4_17
46. Liang, J., Ganesh, V., Poupart, P., Czarnecki, K.: Exponential recency weighted average branching heuristic for SAT solvers. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 30 (2016). https://doi.org/10.1609/aaai.v30i1.10439
47. Liang, J.H., Ganesh, V., Poupart, P., Czarnecki, K.: Learning rate based branching heuristic for SAT solvers. In: Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5–8, 2016, Proceedings. pp. 123–140 (2016). https://doi.org/10.1007/978-3-319-40970-2_9
48. Mahzoon, A., Große, D., Drechsler, R.: Combining symbolic computer algebra and Boolean satisfiability for automatic debugging and fixing of complex multipliers. In: 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE (Jul 2018). https://doi.org/10.1109/isvlsi.2018.00071
49. Mahzoon, A., Große, D., Scholl, C., Konrad, A., Drechsler, R.: Formal verification of modular multipliers using symbolic computer algebra and Boolean satisfiability. In: 59th Design Automation Conference (DAC) (2022), https://www.informatik.uni-bremen.de/agra/doc/konf/Formal_Verification_of_Modular_Multipliers.pdf
50. McKay, B.D., Piperno, A.: Practical graph isomorphism, II. Journal of Symbolic Computation **60**, 94–112 (2014). https://doi.org/10.1016/j.jsc.2013.09.003
51. Neiman, D., Mackey, J., Heule, M.J.H.: Tighter bounds on directed Ramsey number $R(7)$ (2020). https://doi.org/10.48550/ARXIV.2011.00683
52. Nielsen, M.A., Chuang, I.: Quantum computation and quantum information (2002). https://doi.org/10.1119/1.1463744
53. Pavičić, M., Merlet, J.P., McKay, B., Megill, N.D.: Kochen–Specker vectors. Journal of Physics A: Mathematical and General **38**(7), 1577–1592 (Feb 2005). https://doi.org/10.1088/0305-4470/38/7/013
54. Penrose, R.: On Bell non-locality without probabilities: some curious geometry. Quantum Reflections, Cambridge University Press, Cambridge pp. 1–27 (2000), https://people.maths.ox.ac.uk/lmason/Tn/34/TN34-09.pdf
55. Peres, A.: Two simple proofs of the Kochen–Specker theorem. Journal of Physics A: Mathematical and General **24**(4), L175–L178 (Feb 1991). https://doi.org/10.1088/0305-4470/24/4/003
56. Peres, A.: Quantum theory: concepts and methods. Springer (2002). https://doi.org/10.1007/0-306-47120-5
57. Read, R.C.: Every one a winner or how to avoid isomorphism search when cataloguing combinatorial configurations. In: Annals of Discrete Mathematics, vol. 2, pp. 107–120. Elsevier (1978), https://doi.org/10.1016/S0167-5060(08)70325-X
58. Savela, J., Oikarinen, E., Järvisalo, M.: Finding periodic apartments via Boolean satisfiability and orderly generation. In: EPiC Series in Computing. EasyChair (2020). https://doi.org/10.29007/k8jd

59. van der Tak, P., Heule, M.J.H., Biere, A.: Concurrent cube-and-conquer. In: Theory and Applications of Satisfiability Testing – SAT 2012, pp. 475–476. Springer Berlin Heidelberg (2012). https://doi.org/10.1007/978-3-642-31612-8_42
60. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 9.6) (2022), https://www.sagemath.org
61. Uijlen, S., Westerbaan, B.: A Kochen-Specker system has at least 22 vectors. New Generation Computing **34**(1), 3–23 (2016). https://doi.org/10.1007/s00354-016-0202-5
62. Zimba, J., Penrose, R.: On bell non-locality without probabilities: More curious geometry. Studies in History and Philosophy of Science Part A **24**(5), 697–720 (Dec 1993). https://doi.org/10.1016/0039-3681(93)90061-n
63. Zulkoski, E., Bright, C., Heinle, A., Kotsireas, I., Czarnecki, K., Ganesh, V.: Combining SAT solvers with computer algebra systems to verify combinatorial conjectures. Journal of Automated Reasoning **58**(3), 313–339 (2017). https://doi.org/10.1007/s10817-016-9396-y
64. Zulkoski, E., Ganesh, V., Czarnecki, K.: Mathcheck: A math assistant via a combination of computer algebra systems and SAT solvers. In: Felty, A.P., Middeldorp, A. (eds.) Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1–7, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9195, pp. 607–622. Springer (2015). https://doi.org/10.1007/978-3-319-21401-6_41