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

by

Zhengyu Li

A research project
presented to the University of Waterloo
in fulfillment of the
research paper requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Waterloo, Ontario, Canada, 2023

## Abstract

One of the fundamental results in quantum foundations is the Kochen–Specker (KS) theorem, which states that any theory whose predictions agree with quantum mechanics must be contextual, i.e., a quantum observation cannot be understood as revealing a pre-existing value. The theorem hinges on the existence of a mathematical object called a KS vector system. While many KS vector systems are known, the problem of finding the minimum KS vector system in three dimensions has remained stubbornly open for over 55 years. In this paper, we present a new method based on a combination of a Boolean satisfiability (SAT) solver and a computer algebra system (CAS) to address this problem. Our approach shows that a KS system in three dimensions must contain at least 24 vectors. Our SAT+CAS method is over 35,000 times faster at deriving the previously known lower bound of 22 vectors than the prior CAS-based searches. More importantly, we generate certificates that allow verifying our results without trusting either the SAT solver or the CAS. The increase in efficiency is due to the fact we are able to exploit the powerful combinatorial search-with-learning capabilities of SAT solvers, together with the CAS-based isomorph-free exhaustive method of orderly generation of graphs. To the best of our knowledge, our work is the first application of a SAT+CAS method to a problem in the realm of quantum foundations and the first lower bound in the minimum Kochen–Specker problem with a computer-verifiable proof certificate.

## Acknowledgements

I am deeply grateful and honored to have the opportunity to express my appreciation to all those who have contributed to the completion of this Master research paper. This journey has been both challenging and rewarding, and I owe my success to the incredible support and guidance I received from many individuals.

First and foremost, I would like to extend my sincere gratitude to my supervisors Professor Vijay Ganesh and Professor Curtis Bright for their exceptional mentorship and unwavering support throughout my research. Their profound knowledge, insightful advice, and continuous encouragement have been invaluable in shaping this paper and my academic journey as a whole. Their constant feedback and constructive criticism played a crucial role in refining my work and driving me to excel in my research.

I extend my thanks to the entire Computational Mathematics faculty, staff, and fellow students for creating a stimulating academic environment. The collaborative atmosphere and valuable discussions within the department have been instrumental in shaping my ideas and strengthening my research.

I am grateful to Conor Duggan, Piyush Jha, and Jonathan Chung for their thoughtful contributions and insights to the project. The brainstorming sessions and exchange of ideas with them significantly enriched the outcome of this paper.

Lastly, I want to acknowledge the support of all those whose names may not be mentioned but whose contributions and encouragement have been significant in shaping my academic path.

## Dedication

This is dedicated to my parents, my aunt and uncle, and my partner Scarlett Tran. I wouldn't have reached this milestone without their support.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 The Kochen–Specker Theorem and SPIN Axiom

Quantum Mechanics (QM) is often described as one of the most successful physical theories of all time, and yet many questions regarding the very foundations of QM remain unresolved. To address these foundational issues, many interpretations of QM (i.e., mappings from mathematical formalisms of QM to physical phenomena) have been proposed. Hidden-variable theories are attempts at understanding counterintuitive QM phenomena through a deterministic lens by positing the existence of (possibly) unobservable physical entities or hidden variables [40] that 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 [7]. In a similar vein, Simon Kochen and Ernst Specker [61] proved their famous Kochen–Specker (KS) theorem in 1967 (and independently by John Bell in 1966 [6]) that essentially asserts that non-contextual hidden variable theories cannot reproduce the empirical predictions of QM.

Informally, the KS theorem states that there is a contradiction between the SPIN axiom of standard QM and the assumption of non-contextuality. The Stanford Encyclopedia of Philosophy provides a comprehensive background to the KS theorem and stresses its importance in the foundations of QM [40]. The proof of the KS theorem crucially relies on the existence of a KS vector system (see Figure 1.1). More precisely, exhibiting the existence of a KS vector system proves the KS theorem, which essentially states that the unit sphere is not 010-colorable (defined in Section 3). The two following terms play a crucial role in motivating the formal definition of the KS vector system in Section 3:

**Spin of an Elementary Particle:** Spin is an intrinsic form of angular momentum carried by elementary particles. Its existence can be inferred from the Stern–Gerlach experiment [37]. In the context of this paper, a spin-1 particle is shot through a magnetic field in a given direction and continues undisturbed, deflects up, or deflects down—corresponding to 3 possible angular momentum states, namely 0, 1, and $-1$. Thus, the square of this measurement is 0 or 1.

**SPIN axiom:** The SPIN axiom of QM states that the squared spin components of a spin-1 particle are 1, 0, 1 in three pairwise orthogonal directions of measurement. Thus, the observable corresponding to the question "is the squared spin 0?" measured in three mutually orthogonal direction always produce *yes* in exactly one direction and *no* in the other two orthogonal directions. We use the dual of the above form in the present work, i.e., the '010' convention rather than '101', following Uijlen and Westerbaan [79]. The SPIN axiom follows from the postulates of QM and is experimentally verifiable [48].

## 1.2 3-Dimensional Kochen–Specker Problem

The KS theorem rules out non-contextual hidden-variable theories via the existence of a finite set of three-dimensional vectors, referred to as a *KS vector system* [61]. A KS vector system (or simply a KS system) is a combinatorial object that witnesses a contradiction between non-contextuality (i.e., the assumption that observables can be assigned values prior to measurement and independent of measurement context) and the SPIN axiom of QM. The first KS vector system, discovered in 1967, contains 117 vectors [61]. Another theorem that relies on the existence of KS systems in an essential way is the "Free Will" theorem of John Conway and Simon Kochen [28].

Since the publication of Kochen and Specker's theorem in 1967, physicists and mathematicians have wondered about the cardinality of the smallest-sized KS vector system (see Table 1.1 and Section 1.3). Finding the minimum KS system, referred to as the minimum KS problem, is not only of scientific and historical interest, but also has direct applications in quantum information processing [23]. For example, finding a minimum KS system could enable applications in the security of quantum cryptographic protocols based on complementarity [21], zero-error classical communication [31], and dimension witnessing [38]. Further, the large size of all known KS systems has hindered physicists from using them for empirical tests of the KS theorem, similar to the empirical tests of Bell's theorem [75].

| Authors | Year | Bound |
|---|---|---|
| Kochen, Specker | 1967 | $\leq 117$ |
| Jost | 1976 | $\leq 109$ |
| Conway, Kochen | 1990 | $\leq 31$ |
| Arends, Ouaknine, Wampler | 2009 | $\geq 18$ |
| Uijlen, Westerbaan | 2016 | $\geq 22$ |
| Li, Bright, Ganesh | 2022 | $\geq 23$ |
| Li, Bright, Ganesh / Kirchweger, Peitl, Szeider | 2023 | $\geq 24$ |

Table 1.1: A chronology of the bounds on the size of the minimum KS vector system in three dimensions. The present work (presented at CanaDAM 2023) was performed independently of Kirchweger, Peitl, Szeider (presented at IJCAI 2023).

## 1.3 Related Works

Over the last 55 or more years, many mathematicians and physicists such as Roger Penrose, Asher Peres, and John Conway have attempted to find a minimum 3-dimensional KS system (see Table 1.1). The first KS system was constructed in 1967 and it contained 117 vectors [61]. A KS system with 109 vectors was found by Res Jost [50]. The current smallest known KS system in three dimensions contains 31 vectors and was discovered by John Conway and Simon Kochen circa 1990 (see Figure 1.1). All these discoveries were made analytically, without the assistance of computational methods.

In 2011, Arends, Ouaknine, and Wampler proved several interesting properties of KS graphs and leveraged them to computationally establish that a KS system must contain at least 18 vectors [4]. Seven years later, Uijlen and Westerbaan showed that a KS system must have at least 22 vectors [79]. This computational effort used around 300 CPU cores for three months and relied on the *nauty* software package [68] to exhaustively search for KS graphs. Pavičić, Merlet, McKay, and Megill [72] have improved a variation of the KS problem, one in which each vector is part of a mutually orthogonal triple. Under this restriction they show a KS system must have at least 30 vectors in three dimensions and in four or higher dimensions the minimum KS system has 18 vectors. However, in three dimensions the gap between the lower and upper bounds on a KS system remains significant and the minimum size remains unknown.

Another way of measuring the size of a $d$-dimensional KS system is the number of mutually orthogonal "contexts" (cliques of size $d$ in the orthogonality graph). Lisoněk, Badziąg, Portillo, and Cabello [65] found a six-dimensional KS system with seven contexts

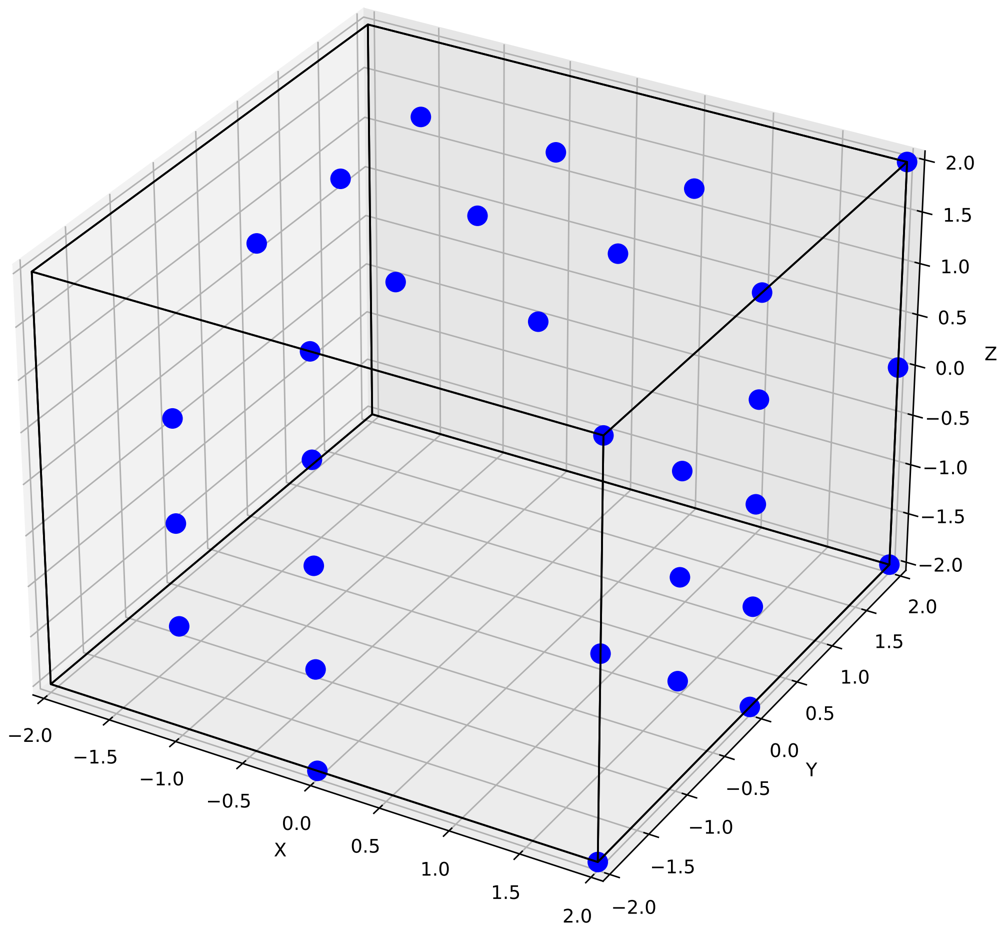


Figure 1.1: The 31 vectors of the smallest known KS system (discovered by John Conway and Simon Kochen circa 1990).

and showed this is the simplest possible KS system allowing a symmetry parity proof of the KS theorem. This KS system was later experimentally used by Cañas et al. [20] to perform measurements verified to arise from a quantum system rather than a classical system.

Preliminary versions of the present work were announced at the *2022 SC-Square workshop*, as well as at the *2023 Southeastern International Conference on Combinatorics, Graph Theory and Computing*, and at *CanaDAM 2023*. At the former two venues we presented searches for KS systems with up to 22 vectors, and at CanaDAM 2023 we presented our work that extends this to a search for KS systems with up to 23 vectors. In each case the searches were exhaustive and no KS systems were found. Thus, a KS system in three dimensions must contain at least 24 vectors.

We recently became aware of the contemporaneous work of Kirchweger, Peitl and Szeider [56] who completed an independent search for KS systems with up to 23 vectors with a similar approach as our technical report [62] but with a SAT modulo symmetries (SMS) solver and an alternate definition of canonicity. They do not use orderly generation, as their definition of canonical does not satisfy property (2) from Sec. 4.1, but otherwise the SMS approach is similar in that it combines a SAT solver with a canonical checking routine [58]. Their approach can also be used to generate proof certificates, though the certificate verification was not performed with the exception of 5% of the certificates in the order 23 search. Going forward, we would like to determine the strengths and weaknesses of the SAT+CAS and SMS approaches and how they can be improved. Having said that, one big difference is that CAS systems are not limited to symmetry breaking. For example, discrete fourier transform (DFT) is applied in the SAT+CAS pipeline for the smallest counterexample of the Williamson conjecture [16].

# Chapter 2

# The SAT+CAS Paradigm for Combinatorics

## 2.1 Boolean Satisfiability Problem

A basic understanding of Boolean logic is necessary to understand the encoding described in Section 3. Each variable in Boolean logic can assume only one of two values denoted by true or false. The variables are joined by Boolean operators. We summarize the most commonly used Boolean operators in Table 2.1.

| Name | Symbol | Arity |
|---|---|---|
| Negation | $\neg$ | 1 |
| Conjunction | $\wedge$ | $n$-ary |
| Disjunction | $\vee$ | $n$-ary |
| Implication | $\rightarrow$ | 2 |
| Biconditional | $\leftrightarrow$ | 2 |

Table 2.1: Some commonly used Boolean operators and their arities.

The interpretation of $\vee$ (disjunction/or), $\wedge$ (conjunction/and), $\neg$ (negation/not) operators follows their intuitive meanings: $x_1 \vee x_2 \vee \cdots \vee x_n$ is evaluated to be true exactly when at least one $x_i$ is true. $x_1 \wedge x_2 \wedge \cdots \wedge x_n$ is true exactly when all $x_i$ are true. $\neg x_i$ is true exactly when $x_i$ is false. We will define the terms “literal”, “clause”, and “formula”, which will be used throughout this paper. A literal is a propositional variable or its negation. A clause is a disjunction of literals. A formula in conjunctive normal form (CNF) is a conjunction of clauses. For example, given variables $x_1, x_2, x_3, x_4$, we can construct the formula $(x_1 \vee x_2) \wedge (\neg x_3 \vee x_4)$, which is in CNF, but a formula like $(x_1 \wedge x_2) \vee (x_3 \wedge x_4)$ is not.

Boolean satisfiability (SAT) is one of the most influential problems in computer science and mathematics, as it has been studied intensively since it was shown to be NP-complete [29]. Over the last two decades, the design and implementation of conflict-driven clause-learning (CDCL) SAT solving algorithms are able to solve multimillion variable instances efficiently [36]. Even more surprisingly, SAT solvers frequently outperform special-purpose algorithms designed for software engineering [22], verification [24], and AI planning [55].

## 2.2 Computer Algebra Systems

Computer algebra systems are powerful tools that have been used for decades by mathematicians to perform symbolic computation in different branches of mathematics such as abstract algebra, graph theory, combinatorics, number theory, etc. However, when applied to problems with a large search space, computer algebra systems (CAS) lack the search capabilities of SAT/SMT solvers. CAS contain state-of-the-art algorithms that can be intergrated as subroutines to encode predicates relevant for a SAT solver. The key concept of the SAT + CAS paradigm mentioned below is to embed the functionality of a computer algebra system (CAS) within the inner loop of a CDCL SAT solver, thus achieving powerful searching capabilities along with strong mathematical context and knowledge.

## 2.3 The SAT+CAS Paradigm for Combinatorics

In recent years we have witnessed the dramatic impact of satisfiability (SAT) solvers—computer programs that take as input Boolean logic formulas and decide whether they have solutions—in areas as diverse as AI, software engineering, program verification, program synthesis, and computer security [9, 36]. Unfortunately, despite these fantastic achievements of SAT solvers, they struggle on certain problems such as those containing many symmetries [69] or those requiring the usage of more advanced mathematical theories than propositional logic [17]. Much work has been done to remedy these drawbacks, including the development of sophisticated symmetry breaking techniques [3] and the development of solvers that support richer logic such as "SAT modulo theories" or SMT solvers [5]. However, the mathematical support of SMT solvers is quite limited 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 CASs, a new kind of solving methodology was developed in 2015 by Zulkoski, Ganesh, and Czarnecki [85] and independently by Ábrahám [1]. This SAT+CAS solving methodology has been successfully applied to many diverse problems, including circuit verification [53, 66], automatic debugging [67], finding circuits for matrix multiplication [43], computing directed Ramsey numbers [70], and verifying mathematical conjectures [13]. For other work in the intersection of symbolic computation and satisfiability checking, see Matthew England's summary [34] of the SC-Square project. In short, the SAT+CAS methodology has found wide application in diverse fields that somehow require solving hard combinatorial problems.

In this paper, we use the SAT+CAS solving methodology (see Figure 2.1) to dramatically improve the performance of the search for KS systems compared to all previous approaches developed to prove lower bounds for the minimum KS problem (see Section 2.5). This is made possible via a combination of the powerful search and learning algorithms used in modern SAT solvers with an "isomorph-free exhaustive generation" approach that prevents the duplicate exploration of isomorphic parts of the search space by the solver. For example, such an approach was recently used to resolve the Lam's problem from projective geometry [12]. Although isomorph-free exhaustive generation has been used extensively in combinatorial enumeration, it has only recently been combined with SAT solving [52, 76].

The traditional approach to prevent a SAT solver repeatedly explore isomorphic sections of a search space is through the use of *symmetry breaking* techniques. [69]. One such symmetry breaking approach is to add "static" constraints to the input formula at the beginning of the search aimed at reducing the size of the search space [30, 42]. Unfortunately, such an approach can be quite expensive in the sense that the number of added constraints can be large (e.g., exponential in the number of variables of the formula that encodes the problem-at-hand). Another approach is to "dynamically" break symmetries during the solver's search [77, 69] such as in the SAT modulo symmetries (SMS) paradigm [59, 57]. Our approach is similar in that it also dynamically adds constraints to the problem during the solving process. However, an important difference is that the SAT+CAS paradigm is more general since the functionality of CAS goes beyond breaking symmetries and can be used to incorporate a wide range of mathematical domain knowledge. For example, in the resolution of the smallest counterexample of the Williamson conjecture, we used the Discrete Fourier Transform (DFT) as part of the CAS computations [16].

## 2.4 Automated Verification of Results

Verification is of utmost importance in the context of computer-assisted proofs, given the mathematical nature of such computations—especially for nonexistence proofs. Fortunately, the SAT+CAS paradigm naturally lends itself to automated verification, given the fact that all modern solvers produce verifiable proofs. By contrast, all previous computer-assisted proofs of lower bounds for the minimum KS problem are not verifiable.

Since our problem requires the solver to perform an exhaustive search, the validity of our nonexistence result is crucially dependent on the encodings and the computational tools that we use. For example, our nonexistence result crucially relies on the correctness of the SAT solver's search and the computer algebra system's isomorph-free exhaustive generation routine. Fortunately, our SAT+CAS method generates verifiable certificates that allow an independent third party to certify that the SAT solver's search is exhaustive and also that the facts provided by the isomorph-free generation are correct. Thus, one does not need to trust either the SAT solver or the CAS to trust that our results are correct—instead, one only needs to trust the correctness of the proof verifier. This is quite significant, as SAT solvers and CASs are complicated pieces of software that typically cannot be guaranteed to be bug-free. By contrast, a proof verifier is a much simpler piece of software that can be formally checked. In Section 5, we provide details on the verification techniques that we used to certify our results.

## 2.5 Our Contributions

In this paper, we present the first successful design and implementation of a SAT+CAS system with extensive verification aimed at problems in the realm of quantum foundations. Specifically, we do so by leveraging and improving the SAT+CAS paradigm to incorporate an isomorph-free generation method (as part of a new SAT+CAS tool, PhysicsCheck[1]) to obtain tighter lower bounds on the minimum KS problem with four orders of magnitude speedup over previous computational methods developed for this problem. We managed to get this improvement in spite of the fact that we also generate verifiable proofs.

In more detail, we implement a robust push-and-run pipeline that incorporates a version of the MapleSAT Boolean SAT solver [63], the SMT solver Z3 [32], and a CAS-based

[1]We provide an easy-to-use open source repository at https://github.com/curtisbright/PhysicsCheck for readers to reproduce our results.

isomorph-free exhaustive generation method known as orderly generation that we implemented and integrated into MapleSAT. We also describe new encoding techniques that enabled an efficient reduction of the minimum KS problem into a SAT problem. Finally, we propose an extension of the standard Boolean UNSAT proof certificate format DRAT (deletion, reverse asymmetric tautology) that enables us to construct certificates of nonexistence for KS systems without needing to trust either the SAT solver or the CAS. We provide a modified DRAT-trim proof checker that can check the nonexistence proof certificates we produced [81].

Our new approach establishes a lower bound of 24 for the minimum size of a KS system, as opposed to the previous best of 22.[2] Our approach is over 35,000 times more efficient than the previous best approach [79] and we solidify previous results by finding candidates missing in previous results (see Section 5). We also verified all certificates computed by the SAT and CAS solvers (see Section 5.2) in contrast to Kirchweger et al. [56] who verified 5% of the certificates in order 23.[3]

In order to make the report relatively self-contained, we provide a thorough background on the KS problem and previous work (Sec. 1.3). Following this, we motivate our SAT encoding of the KS problem (Sec. 3), provide a detailed explanation of orderly generation in the context of the SAT+CAS method (Sec. 4.1), describe our usage of an SMT solver for embeddability checking (Sec. 4.2), and describe how we exploit parallelism (Sec. 4.3). We also provide comparison of our results and runtime with previous work (Sec. 5).

[2]At the SC-Square workshop in 2022, we presented a preliminary version of this work in which we improved the lower bound to 23. The lower bound has also been improved to 24 independently by Kirchweger et al. [56] (see the remarks in Sec. 1.3).

[3]We have fully verified all results for KS systems up to order 22. The results in order 23 have been fully verified except for a small number of cubes (see Section 4.3) whose verification is ongoing.

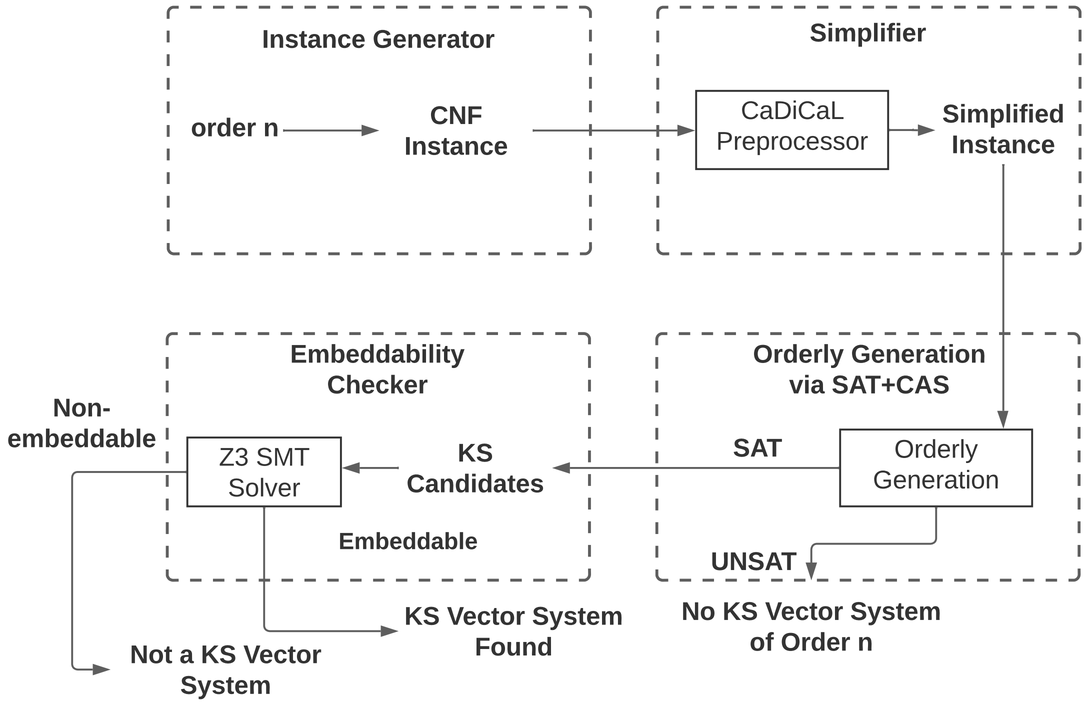


Figure 2.1: A flowchart of our SAT+CAS based toolPhysicsCheck for solving the KS problem. The instance generator generates the SAT instance encoding the KS problem (see Section 3), and the instance is simplified using CaDiCaL [8]. The simplified instance is passed to the MapleSAT+CAS tool (see Section 4.1) either sequentially or in parallel using cube-and-conquer [45]. Finally, an embeddability checker applies the SMT solver Z3 to determine whether the candidates are embeddable (see Section 4.2).

# Chapter 3

# Boolean Encoding of the Kochen–Specker Problem

## 3.1 Encoding Vector Systems as Undirected Graphs

**KS Vector System:** A KS vector system can be represented in multiple ways and 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, two collinear vectors are considered to be equivalent. To define a KS vector system, we first formally define a vector system and the notion of 010-colorability. 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 higher dimensions is already known [72].

**Definition 1 (Vector System)** *A **vector system** is a finite set of non-collinear points on the unit sphere in* $\mathbf{R}^3$*.*

A $\{0, 1\}$-coloring of a vector system is an assignment of 0 and 1 to each vector in the system. The colorings of interest to us are described in the following definition.

**Definition 2 (010-Colorability of Vector Systems)** *A vector system is **010-colorable** if there exists an assignment of 0 and 1 to each vector such that:*

1. *No two orthogonal vectors are assigned 1.*

2. *Three mutually orthogonal vectors are not all assigned 0.*

**Definition 3 (KS Vector System)** *A* ***Kochen–Specker (KS) vector system*** *is a vector system that is not 010-colorable.*

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

Essentially, the vertices of $G_{\mathcal{K}}$ are the vectors in $\mathcal{K}$, and there is an edge between two vertices exactly when their corresponding vectors are orthogonal. Similarly, the notion of 010-colorability can be translated from a vector system to an orthogonality graph.

**Definition 5 (010-colorability of Graphs)** *A graph $G$ is* ***010-colorable*** *if there is a $\{0, 1\}$-coloring of the vertices such that the following two conditions are satisfied simultaneously:*

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

It is not always the case that an arbitrary graph has a corresponding vector system, but if one does exist then we say that such a graph is *embeddable*.

**Definition 6 (Embeddable Graph)** *A graph $G = (V, E)$ is* ***embeddable*** *if it is a subgraph of an orthogonality graph for some vector system.*

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 adjacent vertices are assigned to orthogonal vectors. An example of an unembeddable graph is the cyclic graph $C_4$ on 4 vertices, as the orthogonality constraints force a pair of opposite vertices to be mapped to collinear vectors (which are not allowed in a vector system).

**Definition 7 (KS Graph)** *An embeddable and non-010-colorable graph is called a* ***KS graph****.*

**Observation 1** *There exists a KS vector system if and only if there exists a KS graph.*

## 3.2 Encoding Undirected Graphs as Boolean Formulas

As stated earlier, every 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.

We say a KS graph is minimal if the only subgraph that is also a KS graph is itself. Arends, Ouaknine, and Wampler [4] proved that a three-dimensional minimal KS graph must satisfy the following properties:

1. The graph does not contain the 4-cycle graph $C_4$ as a subgraph.
2. Each vertex of the graph has minimum degree 3.
3. Every vertex is part of a 3-cycle triangle graph $C_3$.

We encode these three properties and the non-010-colorability of the KS graph in conjunctive normal form (CNF), as described below. If a SAT solver produces solutions for such an encoding, then these solutions are equivalent to graphs that satisfy all of the above-mentioned four constraints.

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 since the graphs we consider are undirected. 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 $t_{ijk} \leftrightarrow (e_{ij} \land e_{ik} \land e_{jk})$ which in conjunctive normal form is expressed via the four clauses $\neg t_{ijk} \lor e_{ij}$, $\neg t_{ijk} \lor e_{ik}$, $\neg t_{ijk} \lor e_{jk}$, and $\neg e_{ij} \lor \neg e_{ik} \lor \neg e_{jk} \lor t_{ijk}$. Again, the indices $i$, $j$, and $k$ of the variable $t_{ijk}$ may be reordered arbitrarily for notational convenience.

### 3.2.1 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. Observe that three squares can be formed on four vertices. Therefore, for every choice of four vertices $i$, $j$, $k$, $l$, we use clauses $\neg e_{ij} \lor \neg e_{jk} \lor \neg e_{kl} \lor \neg e_{li}$, $\neg e_{ij} \lor \neg e_{jl} \lor \neg e_{lk} \lor \neg e_{ki}$, and $\neg e_{il} \lor \neg e_{lj} \lor \neg e_{jk} \lor \neg e_{ki}$ to encode the fact that a solution produced by the solver must be squarefree. By enumerating over all possible choices of four vertices and constructing the above CNF formula, we force the graph to be squarefree. The total number of clauses used is $3 \cdot \binom{n}{4}$.

### 3.2.2 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 $1 \leq i \leq n$, enforces that any vertex in the graph has a degree of at least 3. The total number of clauses used is therefore $n \cdot \binom{n-1}{n-3} = n \cdot \binom{n-1}{2}$.

### 3.2.3 Encoding the Triangle Constraint

We encode the property that every vertex is part of a triangle as follows: 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 KS graph—this is encoded 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 $1 \leq i \leq n$ ensures that every vertex is part of a triangle and hence there are $n$ triangle clauses.

### 3.2.4 Encoding the Noncolorability Constraint

Recall that the key property of a KS graph is that it is non-010-colorable. As stated earlier, a graph is non-010-colorable if and only if for all $\{0, 1\}$-colorings of the graph a pair of color-1 vertices is connected or a set of three color-0 vertices are mutually connected.

For each $\{0, 1\}$-coloring, a KS graph has a set $V_0$ of color-0 vertices and a set $V_1$ of color-1 vertices. Given a specific such coloring, the clause

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

encodes that this coloring is not a 010-coloring of a graph—since either a pair of color-1 vertices is connected or three color-0 vertices are mutually connected. Note that we have to generate such a clause for all possible colorings, and conjunct them together to obtain a non-colorability constraint for graphs of order $n$. An assignment that satisfies such a constraint corresponds to a graph that is not 010-colorable under any possible coloring. Observe that in order $n$ the total number of such clauses is $2^n$.

Fortunately, an empirical observation allows cutting the size of formula dramatically: $\{0, 1\}$-colorings with more than $\lceil \frac{n}{2} \rceil$ color-1 vertices are unlikely to be 010-colourings and in practice are not useful in blocking 010-colourable graphs. Put differently, by dropping the constraints with $|V_1| \geq \lceil \frac{n}{2} \rceil$ we reduce the formula size drastically (making the formula easier to solve) and the corresponding increase in the number of satisfying assignments is small enough that these candidates can be ruled out via post-processing (Section 5.2). In fact, for graphs up to order 23, no additional satisfying assignments (or candidate KS graphs) were generated.

### 3.2.5 Encoding Static Isomorphism Blocking Clauses

Following [26], we use symmetry breaking constraints that enforce a lexicographical order among rows of the graph's adjacency matrix. These small number of additional constraint enable us to *statically block* many isomorphic graphs.

Given an adjacency matrix $A$ of a graph, define $A_{i,j}$ as the $i$th 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 $a_1$, $\ldots$, $a_{n-1}$ [60]. The clauses are $\neg x_k \lor y_k \lor \neg a_{k-1}$, $\neg x_k \lor a_k \lor \neg a_{k-1}$, and $y_k \lor a_k \lor \neg a_{k-1}$ for $k = 1, \ldots, n - 1$. The literal $\neg a_0$ is omitted and the clause $\neg x_n \lor y_n \lor \neg a_{n-1}$ is also included.

# Chapter 4

# The PhysicsCheck Pipeline

## 4.1 Orderly Generation via SAT+CAS

The symmetry breaking constraints described in Section 3.2.5 do not block all isomorphic copies of adjacency matrices. Thus, a crucial part of the PhysicsCheck pipeline is the use of a SAT+CAS combination of a SAT solver and an isomorph-free generation routine (the CAS part). The orderly isomorph-free generation approach was developed independently by Read et al. [74] and Faradvzev et al. [35]. It relies on the notion of a *canonical representation* of an adjacency matrix.

**Definition 8 (Canonical Graph)** *An adjacency matrix $M$ of a graph is canonical if every permutation of the graph's vertices 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 starting from the left.*

An *intermediate* matrix of $A$ is a square upper-left submatrix of $A$. If $A$ is of order $n$ then its intermediate matrix of order $n-1$ is said to be its *parent*, and $A$ is said to be a *descendant* of its intermediate matrices.

The orderly generation method is based on the following two consequences of Definition 8:

1. Every isomorphic class of graphs only has exactly one canonical representative.

2. If a matrix is canonical, then its parent is also canonical.

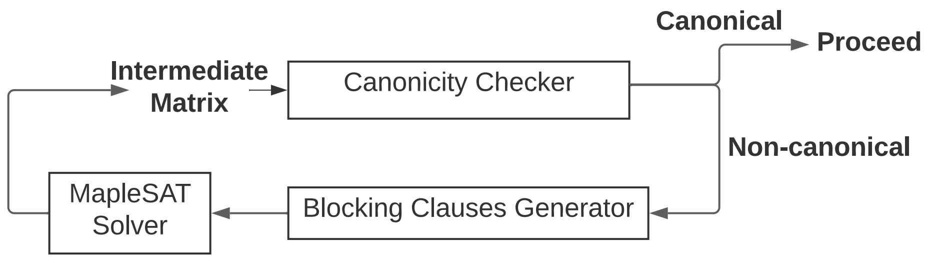


Figure 4.1: A flowchart of the orderly generation algorithm implemented as part of PhysicsCheck's SAT+CAS architecture.

Note that the contrapositive of the second property implies that if a matrix is not canonical, then all of its descendants are not canonical. The orderly generation process only generates canonical matrices and they are built starting from the upper-left. Therefore, any noncanonical intermediate matrix that is encountered during an orderly generation exhaustive search can be discarded, as none of its descendants will be canonical.

As described in Figure 4.1, 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 in the PhysicsCheck system. If the matrix is noncanonical, then a "blocking" clause is learned which removes this matrix (and all of its descendants) from the search. Otherwise, the matrix is canonical and the SAT solver proceeds as normal.

When a matrix is noncanonical, the canonicity-checking routine also provides a "witness" of this fact (a permutation of the vertices that produces a lex-smaller adjacency matrix). We combine this process with the symmetry breaking clauses of Codish et al. that canonical matrices can be shown to satisfy [26, Def. 8].

The orderly generation technique provides a speedup that seems to increase exponentially in the order $n$ of the KS graph—see Table 4.1, which provides experimental running times comparing the SAT+CAS approach against SAT-only and CAS-only approaches. These timings were run on an AMD EPYC 7502P CPU running at 1.5 GHz and the CAS compared against was the *nauty* graph generator [68] with the same configuration from [79].

As described in Figure 2.1, we simplify the SAT instance using the SAT solver CaDiCaL [8] before solving the instance using MapleSAT [64]. 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.

| $n$ | SAT+O.G. | Speedup over SAT | Speedup over CAS |
|---|---|---|---|
| 17 | 0.99 m | 8.9× | 25.6× |
| 18 | 1.65 m | 161.6× | 276.1× |
| 19 | 14.03 m | 834.3× | 677.6× |

Table 4.1: The solving time for SAT + orderly generation and the speedup factor provided in each order $17 \leq n \leq 19$ when compared against SAT-only and CAS-only approaches. We did not provide the speedup factor for $n > 19$ since the SAT-only and CAS-only instances could not be solved within 12,000 minutes.

## 4.2 Embeddability Checker using SMT Solver

We refer to the solutions generated by the SAT solver as *KS candidates*. Note that we have to additionally check whether a KS candidate is embeddable in order to detect whether it is a KS graph (and hence corresponds to a KS vector system). Hence, we perform an embeddability check on every KS candidate generated by the SAT+CAS solver of PhysicsCheck.

Operationally, a graph $G$ is said to be embeddable if every pair of adjacent $G$-vertices can be mapped to two orthogonal vectors on the unit sphere in $\mathbf{R}^3$ (refer to Definition 6). Otherwise, we say that $G$ is unembeddable.

Our embeddability checking algorithm consists of two parts. The first part is an integration of the vector assignment algorithm of [79] that finds all possible vector assignments describing the orthogonal relations between the vectors $v_i$ in a KS candidate defined by a set of edges $E$. A vector assignment is a set of edge pairs $C = \{(e_{ij}, e_{ik}), (e_{lm}, e_{ln}), \dots\} \subseteq E^2$ where each pair of edges share one common vertex and each pair is disjoint from each other, meaning the same edge cannot exist in more than one pair. Each pair $(e_{ij}, e_{ik})$ in $C$ can be interpreted as a cross product relationship between the vectors $v_i$, $v_j$, and $v_k$, since the presence of $e_{ij}$ and $e_{ik}$ in the KS graph means that vector $v_i$ must be orthogonal to both $v_j$ and $v_k$ in any embedding of the candidate.

The second part of the algorithm applies an SMT solver to determine the satisfiability of a system of nonlinear equations generated from a particular vector assignment as described below. More precisely, an assignment 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 [32] that solves the equations over the real numbers. We denote the vector corresponding to vertex $v_i$ as $V_i$ in Z3, where $V_i$ is a 3-tuple of real numbers.

Given a specific vector assignment generated by the previous algorithm, the system of constraints is as follows:

1. If $(e_{ij}, e_{ik}) \in C$, we add the cross product constraint $V_i = V_j \times V_k$.
2. If $e_{ij}$ is one of the edges of $E$ that is not contained in any of the pairs of $C$, we add the dot product constraint $V_i \cdot V_j = 0$.
3. If $i \neq j$ then $V_i$ must not be collinear with $V_j$, so we add the noncollinearity constraint $V_i \times V_j \neq \vec{0}$.

A *free vector* is one that has not been fixed as the cross product of two other vectors. Of all possible assignments, we first choose the one with the least number of free vectors, since in practice such an assignment is likely to be solved more quickly.

It is important to note that for any vector assignment, each edge of the KS graph is encoded into either constraint 1 or 2, making the encoding shorter and more efficient than a naive encoding. 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.

We also fix two orthogonal vectors to be the standard vectors $(1, 0, 0)$ and $(0, 1, 0)$ to cut down on the number of free variables. 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 systems [51]. If a solution is found, it is an assignment of vertices to vectors that satisfies all orthogonality constraints and the graph is therefore embeddable.

Embeddability checking of large graphs can be further optimized by precomputing minimal unembeddable graphs, as defined below.

**Definition 9 (Minimal Unembeddable Graph)** *An unembeddable graph $G$ is said to be a **minimal unembeddable graph** if any proper subgraph of $G$ is embeddable.*

A graph is unembeddable if and only if it contains a minimal unembeddable subgraph. To optimize embeddability checking, we precomputed all minimal unembeddable graphs of orders up to and including 12. It suffices to only consider squarefree graphs in this enumeration, as the square graph $C_4$ is minimally unembeddable itself. Moreover, we only

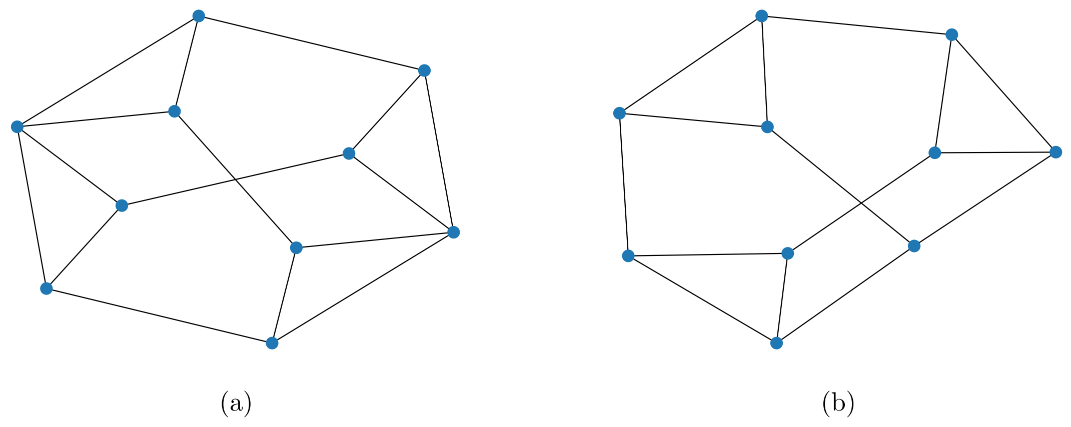


Figure 4.2: The only two minimal nonembeddable graphs of order 10. These are the smallest squarefree graphs that are not embeddable.

consider squarefree graphs with a minimum degree of 2 or greater, as a graph containing a vertex of degree 0 or 1 is not minimally unembeddable. If it was, removing that vertex from the graph yields another unembeddable subgraph—in contradiction to the supposition the graph was minimally unembeddable.

The embeddability of most graphs can be determined using the first assignment (with the fewest free vectors) in less than 1 second. If the satisfiability of an assignment is not determined within 10 seconds, we move on to a different orthogonality assignment and attempt the satisfiability check again 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, since nearly all candidates contain an unembeddable subgraph of order 10, 11, or 12 (see Section 5). In Figure 4.2 we provide the two minimal nonembeddable graphs of order 10 which appear frequently as subgraphs of KS candidates.

## 4.3 Parallization using Cube-and-Conquer

The cube-and-conquer SAT solving paradigm was developed in [46] to solve hard combinatorial problems. The method applies two (possibly) different 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, a "conquering solver" solves each subproblem under the assumption that its associated cube is true (more precisely, the conjunction of the original instance and the cube).

The cube-and-conquer method has empirically been shown to be effective at quickly solving large satisfiability problems when the cubing solver generates many cubes encoding subproblems of similar difficulty. It has since been applied to solve huge combinatorial problems such as the Boolean Pythagorean triples problem [44], the computation of Schur number five [41], and a SAT-based resolution of Lam's problem [12].

In this Section, we discuss parallelization of the SAT+CAS solving process. In our implementation, parallelization is applied by dividing the SAT instance into smaller subproblems using the cube-and-conquer approach [46]. The approach applies the lookahead solver march_cu [46] to partition a hard problem into many cubes and offers very efficient solving time for some combinatorial problems.

During the splitting, the lookahead solver tries to find the next variable that will split the search space the most evenly. Each splitting variable will be added to the SAT instance as a new unit clause, generating two subproblems (one with a positive unit clause and one with a negative unit clause) that can be solved in parallel.

In prior applications of the cube-and-conquer technique [45], the cubing solver generates a collection of cubes before the conquering solver is invoked. Subsequently, each of these subproblems is solved using the conquering solver in parallel. However, this approach presents two primary challenges. Firstly, the generated cubes might exhibit imbalanced solving times, especially since the cubing solver does not have the ability to call the CAS to incorporate isomorph-free generation. Secondly, the proof size for each subproblem will also vary, making it difficult to allocate an appropriate amount of memory to individual cores. In PhysicsCheck, we implement a slight modification of traditional cube-and-conquer practices to resolve the above challenges.

In our proposed method, the cubing solver operates on the CNF instance, generating a set of cubes until a fixed number of edge variables $e_{ij}$ in each subproblem have determined values (either through added unit clauses or unit propagation). Subsequently, each subproblem is passed to the conquering solver (MapleSAT with orderly generation) and solved in parallel. To manage the termination of each subproblem, we set a condition such that if the proof size exceeds 7 GiB, the subproblem is further cubed and solved by increasing the number of edge variables $e_{ij}$ to determine. In addition, we augment instances that need to be cubed further with learned clauses derived from MapleSAT, encompassing noncanonical blocking clauses, unit clauses, and candidate blocking clauses. This iterative process continues until all subproblems can be solved with corresponding proofs of size less than 7 GiB. We overcome the challenges posed by varying proof sizes by implementing this slight modification and it allows us to verify all generated proof certificates with at most 4 GiB of memory allocation.

# Chapter 5

# Results

## 5.1 A New Lower Bound of the KS Problem

Given the CNF file with the encoded constraints, we use the aforementioned encoding techniques combined with the SAT+CAS approach to verify all previous results on KS systems up to order 21 with a speedup factor of over four order-of-magnitudes. Moreover, we improve on the best known lower bound for a minimum KS system (see Table 5.1). The computations up to order 22 were done on an Intel Xeon E5-2667 CPU and the computations in order 23 were done on Intel E5-2683 CPUs that belong to Canada's national advanced research computing platform. All computations are measured in the total CPU time reported by the solver. Our search in order 21 is about 35,000 times faster than the previous computational search of Uijlen and Westerbaan which was distributed on approximately 300 CPU cores and took roughly three months [79]. Furthermore, we achieve comparable runtime to Kirchweger, Peitl and Szeider's search [56] using a SAT modulo symmetries (SMS) solver. Uijlen and Westerbaan were unable to determine the embeddability of one particular graph of order 14. Using our embeddability checking approach, 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[1], we find our minimal unembeddable subgraphs from order 10 to 12 to be identical to theirs up to isomorphism.

We compared our Kochen–Specker candidates with Uijlen and Westerbaan's findings, and verified their conclusion that there is no KS system with strictly less than 22 vectors.

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

| $n$ | Candidates | Simplifying | Solving |
|---|---|---|---|
| 17 | 1 | 0.01 h | 0.00 h |
| 18 | 0 | 0.01 h | 0.02 h |
| 19 | 8 | 0.07 h | 0.15 h |
| 20 | 147 | 0.06 h | 1.25 h |
| 21 | 2,497 | 0.31 h | 18.36 h |
| 22 | 88,282 | 0.44 h | 360.75 h |
| 23 | 3,747,950 | 963.28 h | 52,619.16 h |

Table 5.1: A summary of our results on orders $17 \leq n \leq 23$ (in hours). The second column lists the number of KS candidates generated by PhysicsCheck.

In order 20, we found four additional KS candidates that were not present in the collection of Uijlen and Westerbaan, indicating that their search missed some KS candidates. We present one of the missing graphs in Figure 5.1. We verified that these four additional graphs satisfy the constraints of a KS candidate and therefore would be KS systems were they embeddable, but unfortunately they are not.

In order 23, we adjust the colorability encoding in Section 3.2.4 by reducing the maximum number of color-1 vertices from $\lceil n/2 \rceil$ to $\lceil n/3 \rceil$ to counter the exponential blowup in the number of clauses. As a result, the SAT solver found 5,160,001 solutions in order 23, but 1,412,051 solutions could be 010-colored (using more than $\lceil n/3 \rceil$ color-1 vertices). After these 010-colorable graphs were removed, we were left with 3,747,950 candidates and this matches the count of Kirchweger et al. [56].

All KS candidates of order less than 24 are not embeddable. The embeddability check is done quickly since over 99.99% of the candidates contain one of the minimal nonembeddable subgraph up to order 12 and this can be checked cheaply. Even though all minimal nonembeddable subgraphs up to order 14 are found by Uijlen and Westerbaan [79], only minimal unembeddable graphs up to order 12 are used in the PhysicsCheck pipeline since we computed them ourselves (see Table 5.2). Specifically, there is a single order-22 candidate and there are 41 order-23 candidates that do not contain a minimal nonembeddable subgraph up to order 12.

For verification purposes, we also check if those graphs contain a minimal unembeddable subgraph up to order 14 using the list computed by Uijlen and Westerbaan [79]. We found that all candidates less than order 23 contain a minimal unembeddable subgraph up to order 14, and there are two order-23 candidates that do not contain any known minimal unembeddable subgraphs (see Figure 5.2), corroborating the findings obtained by Kirchweger et al. [56]. The candidates that do not contain any minimal nonembeddable

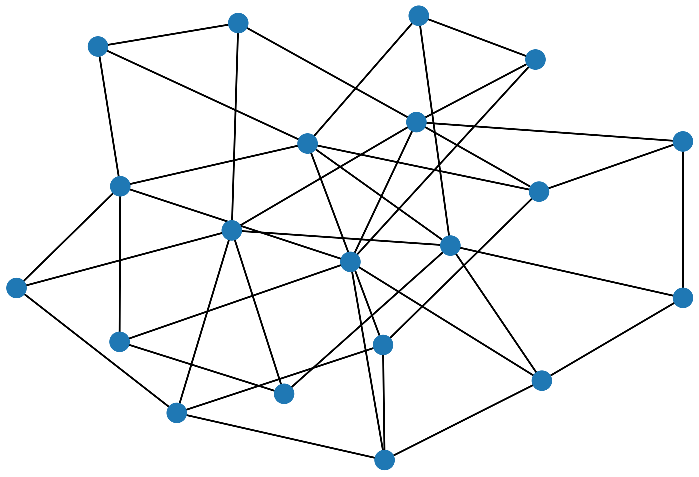


Figure 5.1: One of the four graphs with 20 vertices that was not present in Uijlen and Westerbaan's enumeration. The four graphs satisfy all constraints mentioned in Section 3, but are not embeddable, and therefore do not constitute a KS system.

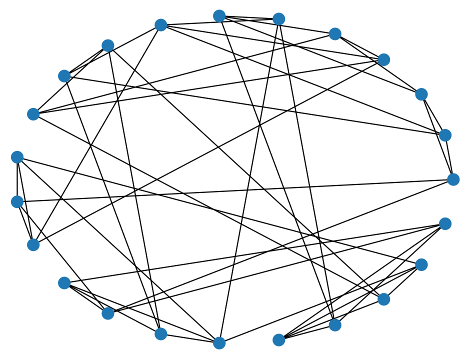
subgraph up to order 12 are determined to be unembeddable using Z3. Therefore, we conclude that the minimum size of a KS system is at least 24.

## 5.2 Formal Verification of Results

In order to verify the computations produced by the SAT solver, we enabled DRAT proof logging in the SAT solver so that nonexistence certificates are generated. This makes it possible for a proof verifier to provide an independent certification of the correctness of the

| Order | Squarefree + Min. Degree 2 | Min. Unembed. | Runtime |
|---|---|---|---|
| 4–9 | 164 | 0 | 30 s |
| 10 | 563 | 2 | 4.1 m |
| 11 | 3,257 | 5 | 1.3 h |
| 12 | 23,699 | 10 | 27 h |

Table 5.2: Counts for the number of minimal unembeddable graphs in orders up to 12 and the computation time for the embeddability check.


(a)

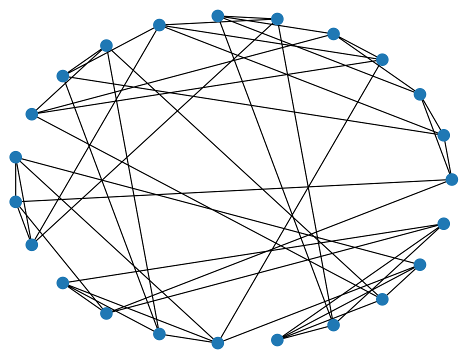

(b)

Figure 5.2: The only two KS candidates up to order 23 that do not contain minimal nonembeddable graphs up to and including order 14. These two graphs are determined to be unembeddable by Z3.

solver's conclusion (assuming the correctness of the constraints in the SAT instance).

A DRAT proof consists of a trace of the clauses learned by the solver during its execution. A proof verifier checks that each clause can be derived from the previous clauses using simple rules known to be logically consistent. The CAS-derived noncanonical blocking clauses cannot be verified using the normal rules, so they were specially tagged to be verified separately. Instead, they are justified via a CAS-derived permutation that, when applied to the blocked adjacency matrix, produces a lex-smaller adjacency matrix—and therefore provides a witness that the blocked matrix is noncanonical and is safe to block.

The CAS-derived clauses in the DRAT proof were prefixed by the character '`t`' to signify they should be trusted and we modified DRAT-trim [81] to trust such clauses. The trusted CAS-derived clauses were separately verified by a permutation-applying Python script that applied the witnesses produced by the CAS to verify the blocked matrices were noncanonical. Similarly, when a KS candidate was found the solver learns a trusted clause blocking the candidate (so that the search continues until all candidates have been found). The DRAT proof ends with the empty clause which by definition is not satisfiable. If the verifier is indeed able to verify the empty clause then we can have confidence that the SAT solver's search missed no candidates without needing to trust the solver itself.

We have fully certified the results up to and including order 22. The uncompressed proofs are about 1.9 TiB in total. For order 23, we have verified more than 95% of the cubes, and we estimate that the uncompressed proof size for order 23 would exceed 40 TiB.

The certification in orders 22 and 23 required using cube-and-conquer (as described in Sec. 4.3) to ensure that each DRAT proof could be verified with at most 4 GiB of memory. While the certification of the non-existence proofs of order 23 is ongoing, we have done many cross-checks on it as described below.

We have conducted extensive cross-verification on all the results (KS candidates) produced by the SAT solver. For example, each KS candidate is passed into a verification script implemented using the NetworkX [39] graph package to verify that they satisfy all encoded constraints (see Section 3). In the order 23 search, some 010-colorable graphs that are colorable with more than $\lceil n/3 \rceil$ color-1s were discarded during this step.

We also test the embeddability pipeline by performing a verification on all embeddable subgraphs. 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.

The candidate counts of the previous search [79] were larger than ours because the previous search did not require each vertex to be part of a triangle. However, we cross-verified that all our KS candidates from order 17 to 21 (except for the four new candidates that we discovered) are isomorphic to the previously known candidates. The previously known candidates were discovered using the graph theory package nauty—a very different approach than ours. Our candidates were verified to be isomorphic to the previously known candidates using both SageMath [78] and NetworkX [39].

# Chapter 6

# Future Work and Conclusion

## 6.1 Future Work

With our current methods, we predict that order 24 can also be solved given proper computation power and parallization, though it is unlikely that an exhaustive search can be performed up to order 31. Consequently, the pursuit of further improvements in the PhysicsCheck pipeline becomes imperative. At present, three of these strategies are under rigorous investigation.

One approach involves a deliberate construction of the exploration domain through the expansion of 'special' subgraphs into KS graphs by adding vertices and edges. This is motivated by the observation that certain subgraphs appear frequently in many KS graphs; thus, limiting the search space by fixing the existence of such subgraphs could allow us to find an embeddable candidates between order 24 to 30 more quickly even though the search is not exhaustive.

Another approach is to introduce techniques that produce more balanced cubes. Currently, the cube-and-conquer process generates cubes that exhibit imbalances in terms of their solving complexity. By generating cubes that are similar in solving time, we minimize the solving time for the hardest cube, and therefore improve the wall-clock solving time of the KS problem.

The last approach is to investigate the additional graph-theretical properties that a KS graph must satisfy, then encode them in conjunctive normal form to reduce the search space of SAT solving.

## 6.2 Conclusion

We give a computer-assisted proof showing that a Kochen–Specker vector system in three dimensions must contain at least 24 vectors. In addition, we provide a computational speedup of over four orders of magnitude over the previously used approach of Uijlen and Westerbaan [79]. For the first time, we successfully implemented and applied the SAT+CAS paradigm along with orderly isomorph-free generation to provide a robust pipeline for problems in quantum foundations. The validity of our work is further confirmed by Kirchweger, Peitl, Szeider [56], who performed an independent search for KS systems with up to 23 vectors with a similar approach as our technical report [62] using a SAT modulo symmetries (SMS) solver. Compared to previous work, our approach is less error-prone since we use heavily-tested proof-generating SAT solvers such as MapleSAT and CaDiCaL. We verified the produced proofs using independent proof checkers, meaning our result does not rely on the correctness of MapleSAT or CaDiCaL.

Finding the minimum KS system has remained stubbornly open for over 55 years. It is not only a problem of great importance to quantum foundations, but has direct applications to various fields of quantum information processing, such as quantum cryptographic protocols [21], zero-error classical communication [31], and dimension witnessing [38]. 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.

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 [13, 43, 12]. With this work we extend the reach of the SAT+CAS paradigm, for the first time, to resolving combinatorial questions in the realm of quantum foundations.

# References

[1] Erika Ábrahám. Building bridges between symbolic computation and satisfiability checking, 2015.

[2] Fadi A Aloul, Arathi Ramani, Igor L Markov, and Karem A Sakallah. Solving difficult SAT instances in the presence of symmetry, 2002.

[3] Fadi A Aloul, Karem A Sakallah, and Igor L Markov. Efficient symmetry breaking for Boolean satisfiability. *IEEE Transactions on Computers*, 55(5):549–558, 2006.

[4] Felix Arends, Joël Ouaknine, and Charles W Wampler. On searching for small Kochen-Specker vector systems, 2011.

[5] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org`, 2016.

[6] John S. Bell. On the problem of hidden variables in quantum mechanics. *Rev. Mod. Phys.*, 38:447–452, Jul 1966.

[7] John Stewart Bell. *Speakable and unspeakable in quantum mechanics: Collected papers on quantum philosophy*. Cambridge University Press, Cambridge, 2004.

[8] Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximillian Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020, 2020.

[9] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*. IOS Press, Amsterdam, February 2021.

[10] David Bohm and Basil J Hiley. *The undivided universe: An ontological interpretation of quantum theory*. Routledge, 2006.

[11] Curtis Bright, Kevin Cheung, Brett Stevens, Dominique Roy, Ilias Kotsireas, and Vijay Ganesh. A nonexistence certificate for projective planes of order ten with weight 15 codewords. *Applicable Algebra in Engineering, Communication and Computing*, 31(3):195–213, 2020.

[12] Curtis Bright, Kevin K. H. Cheung, Brett Stevens, Ilias Kotsireas, and Vijay Ganesh. A SAT-based resolution of Lam's problem, 2021.

[13] Curtis Bright, Vijay Ganesh, Albert Heinle, Ilias Kotsireas, Saeed Nejati, and Krzysztof Czarnecki. MathCheck2: A SAT+CAS verifier for combinatorial conjectures, 2016.

[14] Curtis Bright, Jürgen Gerhard, Ilias Kotsireas, and Vijay Ganesh. Effective problem solving using SAT solvers, 2020.

[15] Curtis Bright, Ilias Kotsireas, and Vijay Ganesh. SAT solvers and computer algebra systems: A powerful combination for mathematics, 2019.

[16] Curtis Bright, Ilias Kotsireas, and Vijay Ganesh. Applying computer algebra systems with SAT solvers to the Williamson conjecture. *Journal of Symbolic Computation*, 100:187–209, 2020.

[17] Curtis Bright, Ilias S. Kotsireas, and Vijay Ganesh. When satisfiability solving meets symbolic computation. *Commun. ACM*, 65(7):64–72, 2022.

[18] Curtis Bright, Ilias S. Kotsireas, Albert Heinle, and Vijay Ganesh. Enumeration of complex Golay pairs via programmatic SAT, 2018.

[19] Costantino Budroni, Adán Cabello, Otfried Gühne, Matthias Kleinmann, and Jan-Åke Larsson. Quantum contextuality. *arXiv preprint arXiv:2102.13036*, 2021.

[20] Gustavo Cañas, Mauricio Arias, Sebastián Etcheverry, Esteban S. Gómez, Adán Cabello, Guilherme B. Xavier, and Gustavo Lima. Applying the simplest Kochen-Specker set for quantum information processing. *Physical Review Letters*, 113(9), August 2014.

[21] Adán Cabello, Vincenzo D'Ambrosio, Eleonora Nagali, and Fabio Sciarrino. Hybrid ququart-encoded quantum cryptography protected by Kochen-Specker contextuality. *Physical Review A*, 84(3):030302, 2011.

[22] Cristian Cadar, Vijay Ganesh, Peter M Pawlowski, David L Dill, and Dawson R Engler. EXE: Automatically generating inputs of death. *ACM Transactions on Information and System Security (TISSEC)*, 12(2):1–38, 2008.

[23] Gustavo Canas, Mauricio Arias, Sebastián Etcheverry, Esteban S Gómez, Adán Cabello, Guilherme B Xavier, and Gustavo Lima. Applying the simplest Kochen-Specker set for quantum information processing. *Physical review letters*, 113(9):090404, 2014.

[24] Edmund Clarke, Armin Biere, Richard Raimi, and Yunshan Zhu. Bounded model checking using satisfiability solving. *Formal methods in system design*, 19(1):7–34, 2001.

[25] Michael Codish, Alice Miller, Patrick Prosser, and A Stuckey. Breaking symmetries in graph representation. 2013.

[26] Michael Codish, Alice Miller, Patrick Prosser, and Peter J. Stuckey. Constraints for symmetry breaking in graph representation. *Constraints*, 24(1):1–24, August 2019.

[27] John Conway and Simon Kochen. The geometry of the quantum paradoxes, 2002.

[28] John Conway and Simon Kochen. The free will theorem. *Foundations of Physics*, 36:1441–1473, 2006.

[29] Stephen A Cook. The complexity of theorem-proving procedures, 1971.

[30] James M. Crawford, Matthew L. Ginsberg, Eugene M. Luks, and Amitabha Roy. Symmetry-breaking predicates for search problems. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*, KR'96, page 148–159, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[31] Toby S Cubitt, Debbie Leung, William Matthews, and Andreas Winter. Improving zero-error classical communication with entanglement. *Physical Review Letters*, 104(23):230503, 2010.

[32] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver, 2008.

[33] Albert Einstein, Boris Podolsky, and Nathan Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical review*, 47(10):777, 1935.

[34] Matthew England. SC-Square: Overview to 2021. In Curtis Bright and James Davenport, editors, *Proceedings of the 6th SC-Square Workshop*, pages 1–6, 2022.

[35] I A Faradžev. Constructive enumeration of combinatorial objects, 1978.

[36] Vijay Ganesh and Moshe Y. Vardi. On the unreasonable effectiveness of SAT solvers, December 2020.

[37] Walther Gerlach and Otto Stern. Der experimentelle nachweis der richtungsquantelung im magnetfeld. *Zeitschrift für Physik*, 9:349–352, 1922.

[38] Otfried Gühne, Costantino Budroni, Adán Cabello, Matthias Kleinmann, and Jan-Åke Larsson. Bounding the quantum dimension with contextuality. *Physical Review A*, 89(6):062107, 2014.

[39] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX, 2008.

[40] Carsten Held. The Kochen-Specker Theorem. https://plato.stanford.edu/archives/spr2018/entries/kochen-specker/, 2000.

[41] Marijn Heule. Schur number five, 2018.

[42] Marijn J. H. Heule. Optimal symmetry breaking for graph problems. *Mathematics in Computer Science*, 13(4):533–548, May 2019.

[43] Marijn J. H. Heule, Manuel Kauers, and Martina Seidl. New ways to multiply $3 \times 3$-matrices. *Journal of Symbolic Computation*, 104:899–916, May 2021.

[44] Marijn J. H. Heule, Oliver Kullmann, and Victor W. Marek. Solving and verifying the boolean pythagorean triples problem via cube-and-conquer, 2016.

[45] Marijn J H Heule, Oliver Kullmann, and Victor W Marek. Solving very hard problems: Cube-and-conquer, a hybrid SAT solving method. In *IJCAI*, volume 17, pages 228–245, 2017.

[46] Marijn J. H. Heule, Oliver Kullmann, Siert Wieringa, and Armin Biere. Cube and conquer: Guiding CDCL SAT solvers by lookaheads, 2011.

[47] Mark Howard, Joel Wallman, Victor Veitch, and Joseph Emerson. Contextuality supplies the 'magic' for quantum computation. *Nature*, 510(7505):351–355, 2014.

[48] Yun-Feng Huang, Chuan-Feng Li, Yong-Sheng Zhang, Jian-Wei Pan, and Guang-Can Guo. Experimental test of the Kochen-Specker theorem with single photons. *Physical Review Letters*, 90(25), June 2003.

[49] Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge university press, 2004.

[50] Res Jost. 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*, pages 209–228, 1976.

[51] Dejan Jovanović and Leonardo de Moura. Solving non-linear arithmetic, 2012.

[52] Tommi Junttila, Matti Karppa, Petteri Kaski, and Jukka Kohonen. An adaptive prefix-assignment technique for symmetry reduction. *Journal of Symbolic Computation*, 99:21–49, July 2020.

[53] Daniela Kaufmann and Armin Biere. Improving AMulet2 for verifying multiplier circuits using SAT solving and computer algebra. *International Journal on Software Tools for Technology Transfer*, January 2023.

[54] Daniela Kaufmann, Armin Biere, and Manuel Kauers. Verifying large multipliers by combining SAT and computer algebra, October 2019.

[55] Henry A Kautz, Bart Selman, et al. Planning as satisfiability, 1992.

[56] Markus Kirchweger, Tomáš Peitl, and Stefan Szeider. Co-certificate learning with SAT Modulo Symmetries, 2023. Main Track, to appear.

[57] Markus Kirchweger, Manfred Scheucher, and Stefan Szeider. A SAT attack on Rota's basis conjecture, 2022.

[58] Markus Kirchweger, Manfred Scheucher, and Stefan Szeider. SAT-based generation of planar graphs, 2023. To appear at the 26th International Conference on Theory and Applications of Satisfiability Testing (SAT 2023), July 04–08, 2023, Alghero, Italy.

[59] Markus Kirchweger and Stefan Szeider. SAT modulo symmetries for graph generation, 2021.

[60] Donald E Knuth. *The art of computer programming, Volume 4, Fascicle 6: Satisfiability*. Addison-Wesley Professional, Massachusetts, 2015.

[61] Simon Kochen and E. P. Specker. The Problem of Hidden Variables in Quantum Mechanics. *Journal of Mathematics and Mechanics*, 17:59–87, 1967.

[62] Zhengyu Li, Curtis Bright, and Vijay Ganesh. A SAT solver + computer algebra attack on the minimum Kochen–Specker problem, 2022. Technical report, https://cs.curtisbright.com/reports/nmi-ks-preprint.pdf.

[63] Jia Liang, Vijay Ganesh, Pascal Poupart, and Krzysztof Czarnecki. Exponential recency weighted average branching heuristic for SAT solvers, 2016.

[64] Jia Hui Liang, Vijay Ganesh, Pascal Poupart, and Krzysztof Czarnecki. Learning rate based branching heuristic for SAT solvers, 2016.

[65] Petr Lisoněk, Piotr Badziąg, José R. Portillo, and Adán Cabello. Kochen-Specker set with seven contexts. *Physical Review A*, 89(4), April 2014.

[66] Alireza Mahzoon, Daniel Große, Christoph Scholl, Alexander Konrad, and Rolf Drechsler. Formal verification of modular multipliers using symbolic computer algebra and boolean satisfiability. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, New York, July 2022. ACM.

[67] Alireza Mahzoon, Daniel Große, and Rolf Drechsler. Combining symbolic computer algebra and Boolean satisfiability for automatic debugging and fixing of complex multipliers, July.

[68] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, 2014.

[69] Hakan Metin, Souheib Baarir, Maximilien Colange, and Fabrice Kordon. CDCLSym: Introducing effective symmetry breaking in SAT solving, 2018.

[70] David Neiman, John Mackey, and Marijn Heule. Tighter bounds on directed Ramsey number $R(7)$. *Graphs and Combinatorics*, 38(5), September 2022.

[71] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[72] Mladen Pavičić, Jean-Pierre Merlet, Brendan McKay, and Norman D. Megill. Kochen–Specker vectors. *Journal of Physics A: Mathematical and General*, 38(7):1577–1592, February 2005.

[73] Asher Peres. *Quantum theory: concepts and methods*. Springer, 2002.

[74] Ronald C Read. Every one a winner or how to avoid isomorphism search when cataloguing combinatorial configurations, 1978.

[75] Emilio Santos. Critical analysis of the empirical tests of local hidden-variable theories. *Phys. Rev. A*, 46:3646–3656, Oct 1992.

[76] Jarkko Savela, Emilia Oikarinen, and Matti Järvisalo. Finding periodic apartments via Boolean satisfiability and orderly generation, 2020.

[77] Meinolf Sellmann and Pascal Van Hentenryck. Structural symmetry breaking, 2005.

[78] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.6)*, 2022. https://www.sagemath.org.

[79] Sander Uijlen and Bas Westerbaan. A Kochen-Specker system has at least 22 vectors. *New Generation Computing*, 34(1):3–23, 2016.

[80] Peter van der Tak, Marijn J. H. Heule, and Armin Biere. Concurrent cube-and-conquer, 2012.

[81] Nathan Wetzler, Marijn J. H. Heule, and Warren A. Hunt Jr. DRAT-trim: Efficient checking and trimming using expressive clausal proofs, 2014.

[82] Robert S. Wolf. *A Tour through Mathematical Logic*, volume 30. Mathematical Association of America, 1 edition, 2005.

[83] Jason Zimba and Roger Penrose. On bell non-locality without probabilities: More curious geometry. *Studies in History and Philosophy of Science Part A*, 24(5):697–720, December 1993.

[84] Edward Zulkoski, Curtis Bright, Albert Heinle, Ilias Kotsireas, Krzysztof Czarnecki, and Vijay Ganesh. Combining SAT solvers with computer algebra systems to verify combinatorial conjectures. *Journal of Automated Reasoning*, 58(3):313–339, 2017.

[85] Edward Zulkoski, Vijay Ganesh, and Krzysztof Czarnecki. Mathcheck: A math assistant via a combination of computer algebra systems and SAT solvers, 2015.