When Satisfiability Solving Meets Symbolic Computation

The Science of Less-Than-Brute Force

CURTIS BRIGHT, University of Windsor, Canada
ILIAS KOTSIREAS, Wilfrid Laurier University, Canada
VIJAY GANESH, University of Waterloo, Canada

Recent advances in satisfiability (SAT) solving and computer algebra systems (CAS) have led to the development of search tools that can solve mathematical problems significantly larger than ever before—and in a faster, more verifiable way. Many long-standing open mathematical problems have been solved using these methods in the past few years and in this article we provide an overview of this "SAT+CAS" paradigm by describing some of its recent successes. In particular, we describe how it produced the first certifiable solution of Lam's problem from geometry (uncovering bugs in previous solutions in the process) and how it produced new results on Williamson's conjecture from combinatorics—such as the determination of its minimal counterexample and the discovery of many new Williamson matrices, including a new infinite family. The SAT+CAS paradigm is well-suited for solving problems that require both efficient search and sophisticated mathematics.

ACM Reference Format:
Curtis Bright, Ilias Kotsireas, and Vijay Ganesh. 2022. When Satisfiability Solving Meets Symbolic Computation: The Science of Less-Than-Brute Force. 1, 1 (June 2022), 12 pages. https://doi.org/10.1145/3500921

INTRODUCTION

Mathematicians have long been fascinated by objects that exhibit exceptionally nice combinatorial properties. However, it is often difficult to determine whether or not objects satisfying a given combinatorial property exist. Sometimes, the only feasible method of definitively answering the question of existence is simply to perform a systematic search. A famous example of this is the proof of the *four colour theorem*—that four colours suffice to colour the regions of a planar map with adjacent regions are coloured differently [3]. The theorem has been known to be true since 1977, but every known proof relies on computer calculations in an essential way. Mathematical arguments are used to reduce the search for counterexamples to a finite number of cases, and the cases are then exhaustively checked using a custom-written computer program in order to rule out any counterexamples.

Independently, over the last fifty years computer scientists have made significant progress on developing general-purpose programs that can automatically solve many kinds of mathematical problems. *Satisfiability solving* and *symbolic computation* are two important branches of computer science that each specialize in solving mathematical problems. Both of these fields have long histories and have produced impressive tools—satisfiability (SAT) solvers in the former and computer algebra systems (CASs) in the latter. Originally, SAT solvers were designed to solve problems in logic,

Authors' addresses: Curtis Bright, University of Windsor, Windsor, Canada; Ilias Kotsireas, Wilfrid Laurier University, Waterloo, Canada; Vijay Ganesh, University of Waterloo, Waterloo, Canada.

Manuscript submitted to ACM

and CASs were tools to manipulate and simplify algebraic expressions. As we will see, these tools have since found an abundance of new applications outside of these original domains.

Despite their common specialization in solving mathematical problems, the SAT and CAS communities have developed independently of each other [1]. Broadly speaking, the SAT community has focused on effective search methods, while the CAS community has focused on effective mathematical algorithms. Recently, these two communities have started to collaborate in crossover initiatives like the SC-square project[1] [2]. Since the insights of these communities are largely complementary, bringing them together has resulted in new solutions to problems that were out-of-reach of either community separately and has produced advances in problems involving nonlinear real arithmetic [13], linear integer arithmetic [12], and Boolean polynomials [28] (to name a few). In this overview, we focus on our own contribution to this ongoing project—a hybrid SAT and CAS system called MathCheck[2] that we have applied to mathematical problems in graph theory [45], finite geometry [5], combinatorics [9], and number theory [11]. For more applications of the SAT+CAS paradigm see the overview article [17] appearing in a special issue of the Journal of Symbolic Computation devoted to SAT and CAS synergies.

Satisfiability Solving

A satisfiability (SAT) solver is a program that solves the satisfiability problem from Boolean logic—given a formula in conjunctive normal form, is there an assignment to its variables that makes the expression true? At first glance, SAT solvers seem disconnected from the kinds of problems that most mathematicians and engineers care about. However, stunning progress in applied SAT solving over the last several decades [43] has lead to a surprising diversity of applications for SAT solvers—from generating or solving puzzles like Sudoku [8] to software verification [30] and hardware design [36]. This "SAT revolution" has even led to the resolution of decades-old mathematical problems such as the Boolean Pythagorean triples problem [26] and the determination of the fifth Schur number [25]. These problems are solved by first reducing them to a set of constraints in Boolean logic. The constraints are provided to SAT solvers that search for solutions of the constraints and provide nonexistence certificates when no solutions exist.

The search spaces in the aforementioned problems are enormous and could never be exhaustively searched without very clever and powerful search methods. The fact that SAT solvers are currently the only tools that can solve these problems speak to their exceptional search ability. However, SAT solvers do struggle with some kinds of problems—particularly those with an underlying mathematical structure that is unknown to the solver.

As a simple example of this phenomenon, consider the following problem: Find a way to put n pigeons into $n-1$ holes given that each hole is only large enough to contain a single pigeon. A moment's thought reveals that the problem is not solvable and this can be justified by a simple counting argument. The problem is also straightforwardly expressed in Boolean logic—but embarrassingly, SAT solvers are known to take an infeasible amount of time to solve it using the most straightforward encoding [23]. The issue is that during the reduction to Boolean logic the mathematical context of the problem is lost; SAT solvers simply don't realize that a counting argument suffices to rule out the existence of a solution. In this case, there are additional mathematical facts that can be encoded into Boolean logic and that allow SAT solvers to effectively show the problem is unsatisfiable for large n. However, in many problems—such as the ones we consider in this article—the mathematical facts that greatly reduce the search space are not easily expressible in Boolean logic. It would seem as if we either have to ignore the mathematical facts or avoid using a SAT solver entirely.

[1] www.sc-square.org
[2] www.uwaterloo.ca/mathcheck

Symbolic Computation

A computer algebra system (CAS) is a program that can manipulate and simplify mathematical expressions and objects. They have a long history and often incorporate functionality from wide-ranging mathematical topics—including polynomial arithmetic and factorization, computational algebraic geometry, algorithmic number theory, symbolic combinatorics, symbolic integration, the solution of differential equations, exact linear algebra, and quantifier elimination. A stunning amount of research—including the 1999 Nobel prize in physics[3]—relies on the precise calculations made possible by computer algebra systems. For example, they were essential in the resolution of Kepler's 1611 conjecture that the most efficient way of packing spheres is in a pyramid shape [24]. This proof made use of numerous CAS functionalities including linear programming, global nonlinear optimization, and interval arithmetic.

CASs are particularly effective at solving problems with so much structure that they can be solved by an algorithm that does not have to resort to a general search.[4] Because of this, CASs are often much more efficient at solving problems whenever they apply. For example, many CASs have routines for finding the factorization of a polynomial that work much faster than finding factors through a general search. The CASs are able to exploit the algebraic structure inherent in the factorization problem and thereby sidestep the need for a general search.

On the other hand, CASs are not typically optimized to perform search with learning as done by modern SAT solvers [1]. Note that the search spaces for mathematical objects are typically of an exponential size and this will quickly overwhelm any system that has not been designed to cope with this immenseness. Simply put, finding a needle in a exponentially-large haystack requires tools specifically designed to find clever ways of pruning the search space down to a manageable size and this has not traditionally been in the domain of symbolic computation research.

The Best of Both Worlds

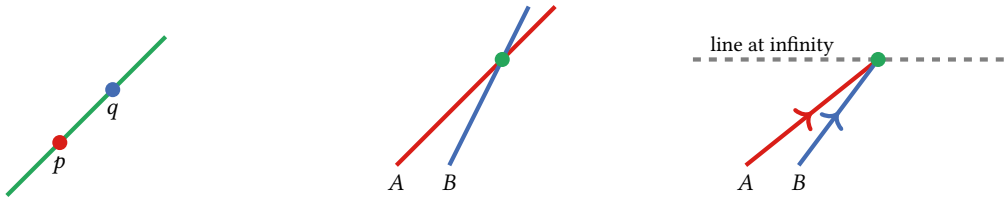
Although the SAT and CAS worlds have both been applied to solve very hard mathematical problems, they have traditionally been applied in isolation—with SAT solvers applied to Boolean constraint problems involving huge search spaces and CASs applied to complicated mathematical problems involving minimal search. While each tool is very effective at solving problems in their respective worlds, a driving motivation of our work is that *there are many problems whose solutions demand effectiveness in both worlds.*

In 2015, Ábrahám proposed to join the techniques of the SAT and CAS worlds together in order to solve problems that are intractable using the techniques of either world in isolation [1]. Independently, the MathCheck project was started and demonstrated the effectiveness of this combination by improving the best known bounds in some graph theoretic conjectures [46]. However, MathCheck can be applied much more generally and over the past five years MathCheck has solved problems from a variety of branches of mathematics. In the remainder of this article we describe how MathCheck has been applied to two long-standing problems in finite geometry and combinatorial matrix theory. This article draws inspiration from Heule and Kullmann's *The Science of Brute Force* [26] which describes how SAT solvers can use "brute force" reasoning to solve problems with enormous search spaces. Within this framework the reasoning of a SAT+CAS solver then becomes "less-than-brute force".

LAM'S PROBLEM

The roots of Lam's problem date back to 300 BC, when Euclid defined five postulates that he believed characterized geometry. While all five postulates were considered obvious, the fifth or "parallel" postulate was significantly more

[3] www.nobelprize.org/prizes/physics/1999/press-release/
[4] We thank a reviewer for raising this point.



(a) Unique line through points p and q. (b) Unique point on skew lines A and B. (c) Unique point on "parallel" lines A and B.

Fig. 1. Visual demonstrations of the axioms of a projective plane.

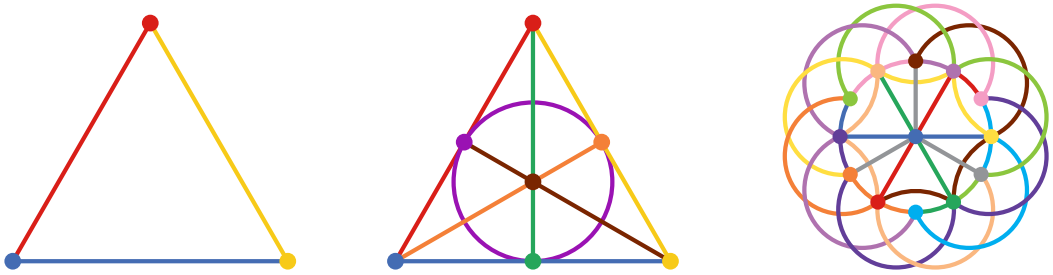
complicated than his other postulates. For over 2000 years this bothered some mathematicians and many attempts were made to prove the parallel postulate from Euclid's other postulates.

Surprisingly, it was not until the 1800s when it was finally realized that this task was impossible—because there are alternate geometries that satisfy Euclid's first four postulates but in which the parallel postulate does not hold. For example, this happens in a *projective plane* which is a collection of points and lines satisfying the following two axioms:

(1) There is exactly one line through any two points.
(2) Any two lines intersect at a unique point.

These axioms are visually demonstrated in Figure 1. The first axiom holds in the usual Euclidean plane and is visualized in Figure 1a. The second axiom does not hold in the usual Euclidean plane—if two lines are skew (shown in Figure 1b) they *will* intersect in a unique point, but if the lines are parallel they will not meet at all, let alone in a unique point. To remedy this situation, the usual Euclidean plane may be augmented with a "line at infinity" on which any two formerly parallel lines will intersect (shown in Figure 1c). The resulting structure satisfies both axioms (1) and (2).

An interesting question can now be raised: are there any other structures that also satisfy these axioms? For example, are there examples of projective planes with a finite number of points? There are some trivial ways of satisfying the axioms—for example, if the plane consists of a single line containing every point. Disqualifying these trivial examples, counting arguments can be used to show that if a finite projective plane exists then it contains the same number of lines and points and every point lies on the same number of lines. If each point lies on $n + 1$ lines then the plane is said to be of *order* n and it will contain exactly $n^2 + n + 1$ points. Determining the set of possible orders for projective planes has been of significant mathematical interest for over 200 years and today it remains a major open problem.



(a) Projective plane of order 1. (b) Projective plane of order 2. (c) Projective plane of order 3.

Fig. 2. Visual depictions of projective planes in orders 1–3. Each point and line are drawn in a separate colour.

Projective planes in the orders up to three may be explicitly visualized through the structures shown in Figure 2. By inspection, one can verify the axioms that any two points lie on a unique line and any two lines intersect in a unique point. Some might protest that the "lines" in these planes are not necessarily represented by straight lines—but so long as the lines satisfy the axioms they can be drawn however we like!

Figure 3 demonstrates the small orders for which projective planes are known to exist. Immediately what jumps out is that projective planes *do* generally exist in small orders—except in order six, when a projective plane would contain 43 points. What is special about this order? In 1949, Bruck and Ryser proved that if the order of a projective plane is of the form $4k + 1$ or $4k + 2$ then it must be the sum of two integer squares [14]. Since six is of the form $4k + 2$ but it is not the sum of two squares it follows that no projective plane of order six can exist. However, the Bruck–Ryser theorem cannot be used to rule out order ten, as ten is the sum of two squares (3^2 and 1^2). This case has attracted a huge amount of interest since no projective plane of order ten is known—yet no one knows any theoretical reason why one shouldn't exist. *Lam's problem* is to resolve this dilemma and determine how to correctly complete the missing entry in Figure 3.

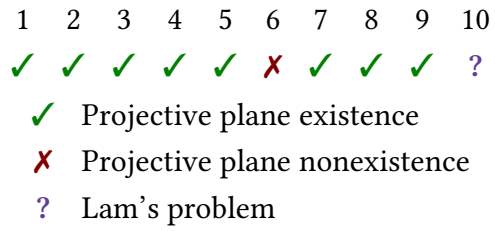


Fig. 3. A summary of when projective planes exist in orders up to ten. Lam's problem is to determine if a projective plane of order ten exists.

A Computational Solution

Lam's problem was eventually resolved in the late 1980s by using a sophisticated case breakdown and an enormous amount of computing to separately search each case [32]. No cases were found to lead to a solution, thereby disproving the existence of a projective plane of order ten.

The case breakdown was based on properties that a binary error-correcting code associated with a hypothetical projective plane of order ten must satisfy. This code is derived from the projective plane's *incidence matrix*—the $\{0, 1\}$-matrix that contains a 1 in its (i, j)th entry exactly when the ith line is incident with the jth point (see Figure 4). A *codeword* is a $\{0, 1\}$-vector in the row space (mod 2) of this incidence matrix and a codeword's *weight* is how many nonzero entries it has. Analyzing the properties of these codewords through some powerful theorems of coding theory shows that a projective plane of order ten must produce at least one codeword of weight 15, 16, or 19—thus splitting the search into three cases.

1	1	0	1	0	0	0
0	1	1	0	1	0	0
0	0	1	1	0	1	0
0	0	0	1	1	0	1
1	0	0	0	1	1	0
0	1	0	0	0	1	1
1	0	1	0	0	0	1

Fig. 4. The incidence matrix of a projective plane of order 2.

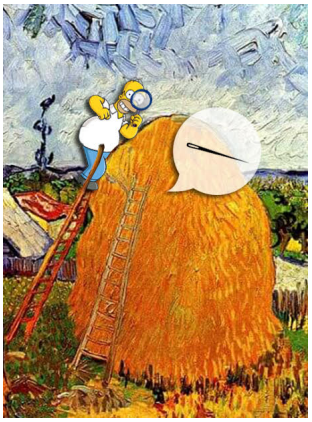
> *"I want to emphasize that this is only an experimental result and it desperately needs an independent verification."* C. W. H. Lam [32]

The weight 15 case was resolved in 1973 using several hours on a mainframe computer [35], the weight 16 case was resolved in 1986 using 2,000 hours on a supermini computer [33] and 100 hours on a supercomputer [15], and the weight 19 case was resolved in 1989 using about 20,000 hours on a supermini computer and 2,000 hours on a supercomputer [34]. These searches were performed using code specifically written for each case and the authors acknowledged that mistakes were a real possibility. The source codes are not publicly available, and even if they were, they were designed to be run on specialized computers that

are no longer produced. In order to believe that Lam's problem has in fact been resolved, we now have to resolve a different and serious problem—how can these searches be verified?

A New Hope

Developments in the fields of satisfiability checking and symbolic computation offer the promise that Lam's problem can be resolved more efficiently and also resolved to a higher standard of rigour. This is because SAT solvers—though complicated pieces of software—are well-tested and produce certificates that allow their results to be validated by external certificate verifiers. This means that it is no longer necessary to trust the implementation of a complicated search algorithm. Instead, one need only trust the certificate verifier.



Brute reasoning with discerning tools can search exponentially large haystacks.

Naturally, the SAT solver will be used as the "combinatorial workhorse" that searches for a projective plane in each possible case. But of what use is the CAS in the search? To answer this, imagine you have a haystack for which you know the left and right sides are perfectly symmetrical. If you want to find a needle in this haystack a brilliant insight is to split the haystack down the middle and only search through one side—since anything in the left side appears on the right side and vice versa.

The same reasoning also applies more generally. Whenever a symmetry is detected in the search space, the space should ideally be split up and reduced to a single nonsymmetric component. In many problems solved by SAT solvers [25, 26] this is done through the introduction of additional "symmetry breaking" constraints which prevent the SAT solver from exploring identical parts of the search space. This method of adding "static" constraints requires the blocked symmetries to be known in advance and works best when there is a concise set of constraints which block those symmetries. Unfortunately, Lam's problem contains some complicated symmetries that are not so easy to block. However, if these symmetries can be detected during the search then it is possible to block them dynamically and such an approach has successfully been used in various SAT solvers [18, 38, 42]. Computing symmetries is one of the things that CASs excel at—including the complicated symmetries that arise in Lam's problem. Thus, a SAT+CAS system is perfectly suited to take advantage of the strengths of both SAT solvers and CASs.

How should the SAT solver and CAS communicate? The basic connection is outlined in Figure 5. As the SAT solver is searching it finds "partial" projective planes—structures that satisfy the projective plane constraints up to some point. These are provided to the CAS and the CAS will send back additional symmetry blocking constraints that remove other partial projective planes (symmetric to the one that was found) from the search space.

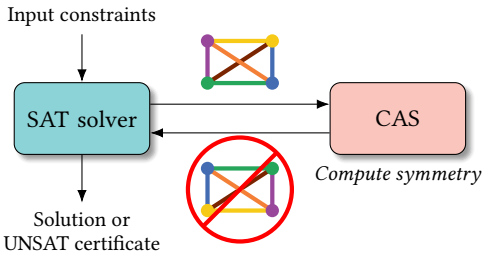


Fig. 5. An outline of the SAT+CAS method as applied to searching for projective planes. The SAT solver provides a partial projective plane to the CAS which computes the symmetries present. This example shows blocking the symmetry of rotating the partial plane by 180 degrees.

This kind of connection takes inspiration from the Davis–Putnam–Logemann–Loveland DPLL(T) algorithm [39] with the first-order logic T-solver replaced by

a CAS. This connection also exploits the conflict-driven clause learning used by modern SAT solvers, since after a blocking constraint is generated it forms a *conflict* that the SAT solver can use to control how much the search should backtrack. We stress that this connection is very general and can be applied in many other circumstances than symmetry breaking (as we will see in the next section). Indeed, one of the strengths of CASs is their huge amount of mathematical functionality—something that can be exploited to learn a variety of mathematical facts that the SAT solver would otherwise have no way of knowing. This also allows the SAT+CAS method to be applied on a wider variety of problems than "SAT modulo theory" (SMT) solvers which are similar in that they also solve problems using the DPLL(T) algorithm. However, SMT solvers are typically limited to solving problems that are specified in one of a fixed number of theories like the theory of integers, reals, or strings.

The Effectiveness of SAT+CAS

How does the effectiveness of the SAT+CAS approach compare with previous approaches? The weight 15 search that was first solved in 1973 [35] has since been confirmed by at least four independent implementations, some requiring up to 80 minutes on a modern desktop [6]. The fastest previous implementation that we are aware of uses highly optimized C code and requires about 30 seconds to complete [16]. Using a straightforward SAT reduction this case can be solved in about 6 minutes with a modern SAT solver—already faster than some searches *specifically written* to perform this search. This speaks to the efficiency of modern SAT solvers, as a mathematical analysis of the search space reveals many symmetries that SAT solvers ignore. Using a SAT+CAS solver that does this (as described in Figure 5) the search completes in about 7 seconds, the fastest approach yet [6].

The weight 16 search is significantly more involved and has a much larger search space compared with the weight 15 search. Much of the symmetry in the weight 16 search space can be analyzed by theoretical means and removed by splitting the search into ten distinct cases. However, some cases still have over a thousand symmetries in their search space which are harder to deal with—leading to dramatic speedups in the running time when these symmetries can be detected and removed by a CAS. Following the exhaustive searches of the space completing in 1986 [15, 33] we are aware of only a single previous independent confirmation: an optimized C implementation using the CAS library nauty [37] that required 16,000 hours on a cluster of desktops in 2011 [41]. In comparison, the SAT approach with CAS symmetry removal resolves this case in 30 hours [5].

The weight 19 search is even more involved than the weight 16 search, and following the exhaustive search in 1989 [34] the only independent confirmation we are aware of required 19,000 hours in 2011 [41]. The initial step of this case splits the search into about 650,000 distinct cases. The previous CAS-based search required about 7 hours to complete the initialization step while our SAT-based approach without CAS symmetry breaking used 62 hours. Adding in CAS symmetry breaking sped up the computation by a factor of 150 and resulted in the initialization being completed in 25 minutes. Unfortunately, after the initialization step the search space is usually not very symmetric, meaning we cannot expect to achieve a big speedup through symmetry breaking. Even still, without any special-purpose search algorithm the SAT+CAS method completed the search in about 16,000 hours [4] and ran an average of 25% faster than the special-purpose code used in the 2011 confirmation of Lam's problem when benchmarked on the same hardware.

A final advantage of the SAT+CAS method is that the certificates produced by the solver may be verified by an independent party. They only need to trust the SAT encoding and the CAS-generated constraints—not the actual procedure used to generate the certificates. This does not prove our searches are error-free but does significantly reduce the amount of trust necessary. This is a particularly important consideration for "experimental" results lacking formal proof because we can be almost certain that custom-written programs contain bugs. This is not a sleight on the

authors but a simple reality of software development: without extensive formal verification even the most well-used and well-tested software cannot be made bug-free.

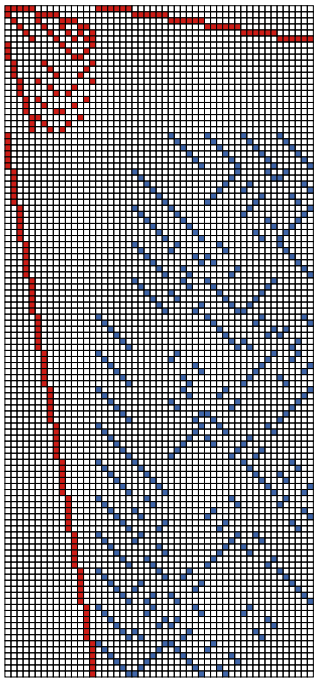


Fig. 6. A 51-column partial projective plane previously claimed to not exist. Red entries denote 1s known in advance and blue entries denote 1s determined by MathCheck.

Indeed, our searches uncovered errors in several previous searches, including both the original 1989 search and its 2011 confirmation. Errors can be detected when the intermediate results of these searches (e.g., the claimed nonexistence of a partial solution) contradict the partial solutions that are found by the SAT solver. For example, the 2011 confirmation of the weight 15 case was based upon proving the nonexistence of a certain 51-column partial projective plane—which in fact actually exists and is explicitly shown in Figure 6. Note that it is easy to check that such a matrix is in fact a partial projective plane, as every column contains the same number of 1s and any two distinct columns share exactly a single 1 in the same location.

In summary, the SAT+CAS method works well in Lam's problem because it allows searching for projective planes using the powerful search routines of a modern SAT solver while simultaneously allowing the mathematical properties revealed by a computer algebra system to greatly constrain the search space. Furthermore, it removes the need to write a custom special-purpose search algorithm and therefore ultimately makes executing the search a more straightforward and trustworthy process.

WILLIAMSON CONJECTURE

Next, we discuss some of the impact the SAT+CAS method has had on combinatorial matrix theory. We begin with a motivating example: say you need to communicate with someone over a noisy channel. In other words, the data that you send may not be the data that is received. How can you increase the likelihood the recipient can perfectly recover your original message? One way to do this is to first *encode* your message using a set of codewords that you and your recipient agree on in advance. Because the codewords may be corrupted while being sent it makes sense to choose the codewords to be as different as possible. For example, if you have n binary codewords of length n you could try to maximize the minimum pairwise Hamming distance of the codewords. Plotkin's bound from coding theory [40] says that this minimum distance cannot be strictly larger than $n/2$. In certain cases it is possible to reach this bound and find a set of n binary codewords of length n such that the Hamming distance of any two distinct codewords is exactly $n/2$.

In this context it is convenient to represent a binary codeword as a $\{\pm 1\}$-vector. Then two codewords of length n have a Hamming distance of $n/2$ exactly when they are orthogonal vectors and a set of n such codewords with pairwise Hamming distances of $n/2$ form the rows of a matrix A such that the off-diagonal entries of AA^T are zero. Such a matrix is called a *Hadamard matrix* after the French mathematician Jacques Hadamard who studied them in 1893 [22]. Even though they have been extensively studied for over 125 years there remain many open problems concerning them—we still don't even know all the orders in which they exist. Hadamard was able to prove that if a Hadamard matrix has order larger than two then its order must necessarily be a multiple of four. The *Hadamard conjecture* is that this necessary condition is also sufficient.

Since a proof of the Hadamard conjecture has not been forthcoming, mathematicians and computer scientists have constructed Hadamard matrices in as many orders as possible; currently $n = 167$ is the smallest case for which a Hadamard matrix of order $4n$ is not known. Because the search space for general Hadamard matrices is so enormous it makes sense to focus on special constructions that have more structure to exploit. One such construction was presented by the mathematician John Williamson in 1944 [44]. In this construction we have four symmetric $\{\pm 1\}$-matrices A, B, C, D of order n which for simplicity we assume are circulant (each row is a cyclic shift of the previous row). If $A^2 + B^2 + C^2 + D^2$ is the identity matrix scaled by a factor of $4n$ then a Hadamard matrix of order $4n$ exists and is explicitly presented in Figure 7. In this case the matrices A, B, C, D are known as a *set of Williamson matrices*.

In the 1960s, while developing codes for spacecraft communication, scientists at NASA's Jet Propulsion Laboratory discovered the first set of Williamson matrices of order twenty-three and conjectured that Williamson matrices exist for all values of n [21]. This conjecture was disproven in 1993 when the counterexample $n = 35$ was discovered by exhaustive computer search [19]. At the time it was noted that this is the smallest *odd* counterexample but not much was known about the even orders—though Williamson himself found some examples in even orders including all powers of two up to $n = 32$. He expressed interest in proving an existence theorem which could generalize this pattern but was unable to do so. The search space for the next power of two $n = 64$ was out of the range of feasibility for computational searches, so for 75 years it was unknown if Williamson matrices of order 64 existed or not.

A	B	C	D
$-B$	A	$-D$	C
$-C$	D	A	$-B$
$-D$	$-C$	B	A

Fig. 7. The general form of Hadamard matrices constructed via the Williamson construction.

"It would be interesting to determine whether the results of this paper are isolated results or are particular cases of some general theorem."

J. Williamson [44]

A SAT+CAS Solution

MathCheck was able to greatly improve our knowledge of Williamson matrices by exhaustively searching for Williamson matrices in many orders $n \leq 70$ (see Figure 8 for one example). Two immediate surprises were uncovered: first, Williamson matrices exist in all even orders $n \leq 70$ (including $n = 64$), and second, Williamson matrices are much more plentiful when their order is divisible by a large power of two. For example, there are over 70,000 sets of Williamson matrices in order 64 alone and fewer than 100 sets of Williamson matrices in all the odd orders up to 64. Moreover, analyzing the structure of the Williamson matrices in order 64 revealed a structure that can be generalized to all powers of two. Thus, as a result of the searches of MathCheck we have improved Williamson's result that Williamson matrices of order 2^k exist with $k \leq 5$; we now know that Williamson matrices of this form actually exist for all k [10]. These results raise the possibility that Williamson matrices actually exist in all even orders and this has become known as the *even Williamson conjecture*.

SAT solvers can be used to search for Williamson matrices because the arithmetical defining relationship of Williamson matrices may be directly encoded using straightforward arithmetic circuits. However, this encoding is not at all competitive with special-purpose solvers and tops out around order thirty [7]. The reason special-purpose solvers are much more effective is because they can exploit mathematical properties that Williamson matrices are known to satisfy. One example of this is known as the power spectral density or *PSD condition*.

Briefly, the PSD of a vector $X = [x_0, \ldots, x_{n-1}]$ is the vector of squared magnitudes of the discrete Fourier transform of X. In other words, the kth entry of $\mathrm{PSD}(X)$ is $\left|\sum_{j=0}^{n-1} x_j \omega^{jk}\right|^2$ where ω is a primitive nth root of unity. Amazingly, if A is the first row of a Williamson matrix then all entries of $\mathrm{PSD}(A)$ are at most $4n$. This is a very useful condition since

the vast majority of $\{\pm 1\}$-vectors do not satisfy it—thus exploiting it dramatically reduces the size of the search space. The problem, of course, is that SAT solvers have no conception of what a PSD value is and the condition is not easily encoded into Boolean logic.

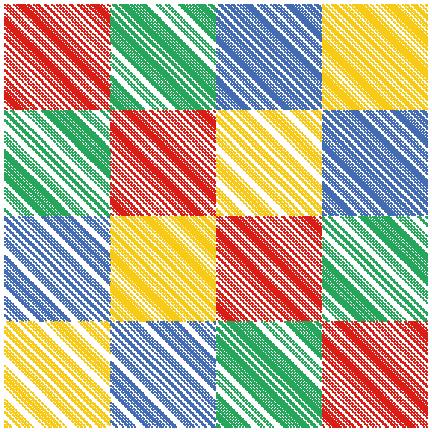


Fig. 8. A Hadamard matrix of order 280 generated from a set of Williamson matrices of order 70. Coloured entries represent 1s and white entries represent −1s.

However, CASs are adept at computing PSD values using the fast Fourier transform. Thus, when a SAT solver finds a matrix that it thinks might appear in a set of Williamson matrices it can provide this matrix to a CAS. The CAS checks if it can be ruled out using the PSD condition, and if so—because the first row of the matrix has a PSD value larger than $4n$—then the matrix is removed from the search space. The SAT solver will continue to search for matrices that satisfy the constraints until either a set of Williamson matrices is found or it is able to certify that no Williamson matrices exist in the given order. This general method is outlined in Figure 9.

The SAT+CAS method using PSD filtering easily outperforms approaches using either SAT solvers alone or CASs alone—particularly in large orders where the SAT+CAS method can perform orders of magnitude faster [9]. The size of the search space for Williamson matrices of order 70 is about $2^{4(70/2)} \approx 10^{42}$ making it rather surprising that such a space can be searched *exhaustively*. But using the powerful search routines of SAT solvers coupled with the sophisticated mathematical capabilities available in CASs (and some additional powerful filtering criteria [20]) make this search tractable to complete on a modern CPU in under 1,000 hours. The minimal counterexample of the Williamson conjecture ($n = 35$) can be verified in a few minutes.

CONCLUSION

The coupling of SAT solvers with computer algebra systems can effectively search large spaces specified via mathematical constraints—combining the search capabilities of SAT solvers with the expressiveness and rich mathematical knowledge of CASs. Although in this overview we've focused on applications to finite geometry and combinatorics, the paradigm is quite general and we have also used MathCheck to improve the known bounds on conjectures in graph theory [45] and number theory [11]. The graph theory and number theory applications provide even more of a taste of the variety of possibilities—relying on CAS functionality like a travelling salesperson solver, a shortest path solver, and a nonlinear real optimizer. Most recently, we've used MathCheck to derive a new lower bound on the size of Kochen–Specker systems—an object from quantum mechanics used to prove the "Free Will Theorem" that if humans have free will then so do elementary particles.

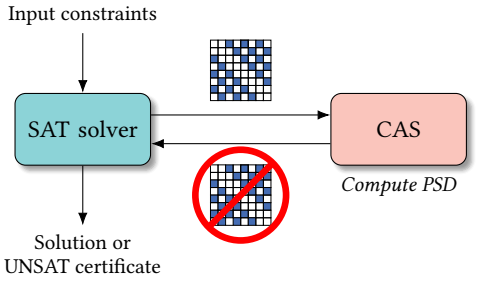


Fig. 9. A outline of the SAT+CAS method as applied to searching for Williamson matrices. The matrix provided to the CAS is blocked when it has a PSD value that is too large.

The type of problems particularly attractive for pursuing with SAT+CAS methods have two primary characteristics. First, the problem is somehow "Booleanizable"—in other words, a large part of the search space can be specified in Boolean logic, allowing the search power of SAT solvers to be exploited. Second, there are mathematical properties, theorems, or invariants that *cannot* easily be specified in Boolean logic but significantly decrease the search space size. These kinds of constraints lie perfectly in the wheelhouse of a CAS and as we've seen can dramatically improve the efficiency of a solver. Best of all, they are not limited to any particular branch of mathematics, and indeed there have been applications of SAT+CAS methods to a surprising variety of problems in areas like cryptanalysis [27], program synthesis [29], and circuit verification [31]. These applications along with those pioneered by initiatives like the SC-square project [17] have convinced us that SAT+CAS methods will be with us for a long time to come—solving problems larger and larger than previously thought possible.

ACKNOWLEDGEMENTS

We would like to thank the reviewers for their insightful comments which improved the presentation and clarity of this article. The needle-in-a-haystack image is by Athanasios Gounaris. The visualization of the projective plane of order three is based off of a representation originally presented by Petr Vojtěchovský.

REFERENCES

[1] E. Ábrahám. Building bridges between symbolic computation and satisfiability checking. In S. Linton, editor, *Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation*, pages 1–6. ACM, 2015.

[2] E. Ábrahám, J. Abbott, B. Becker, A. M. Bigatti, M. Brain, A. Cimatti, J. H. Davenport, M. England, P. Fontaine, S. Forrest, V. Ganesh, A. Griggio, D. Kroening, and W. M. Seiler. SC^2: when satisfiability checking and symbolic computation join forces. In G. Reger and D. Traytel, editors, *ARCADE 2017, 1st International Workshop on Automated Reasoning: Challenges, Applications, Directions, Exemplary Achievements*, volume 51 of *EPiC Series in Computing*, pages 6–10. EasyChair, 2017.

[3] K. Appel and W. Haken. Every planar map is four colorable. Part I: Discharging. *Illinois Journal of Mathematics*, 21(3):429–490, 1977.

[4] C. Bright, K. Cheung, B. Stevens, I. Kotsireas, and V. Ganesh. A SAT-based resolution of Lam's problem. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 3669–3676, 2021.

[5] C. Bright, K. K. H. Cheung, B. Stevens, I. Kotsireas, and V. Ganesh. Unsatisfiability proofs for weight 16 codewords in Lam's problem. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 1460–1466, 2020.

[6] C. Bright, K. K. H. Cheung, B. Stevens, D. Roy, I. Kotsireas, and V. 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.

[7] C. Bright, V. Ganesh, A. Heinle, I. Kotsireas, S. Nejati, and K. Czarnecki. MathCheck2: A SAT+CAS verifier for combinatorial conjectures. In V. P. Gerdt, W. Koepf, W. M. Seiler, and E. V. Vorozhtsov, editors, *Proceedings of the 18th International Workshop on Computer Algebra in Scientific Computing*, pages 117–133. Springer, 2016.

[8] C. Bright, J. Gerhard, I. Kotsireas, and V. Ganesh. Effective problem solving using SAT solvers. In *Maple in Mathematics Education and Research*, volume 1125 of *Communications in Computer and Information Science*, pages 205–219. Springer, 2020.

[9] C. Bright, I. Kotsireas, and V. Ganesh. Applying computer algebra systems with SAT solvers to the Williamson conjecture. *Journal of Symbolic Computation*, 100:187–209, 2020.

[10] C. Bright, I. Kotsireas, and V. Ganesh. New infinite families of perfect quaternion sequences and Williamson sequences. *IEEE Transactions on Information Theory*, 66(12):7739–7751, 2020.

[11] C. Bright, I. Kotsireas, A. Heinle, and V. Ganesh. Complex Golay pairs up to length 28: A search via computer algebra and programmatic SAT. *Journal of Symbolic Computation*, 102:153–172, 2021.

[12] M. Bromberger, T. Sturm, and C. Weidenbach. A complete and terminating approach to linear integer solving. *Journal of Symbolic Computation*, 100:102–136, Sept. 2020.

[13] C. W. Brown and F. Vale-Enriquez. From simplification to a partial theory solver for non-linear real polynomial constraints. *Journal of Symbolic Computation*, 100:72–101, Sept. 2020.

[14] R. H. Bruck and H. J. Ryser. The nonexistence of certain finite projective planes. *Canadian Journal of Mathematics*, 1(1):88–93, 1949.

[15] J. L. Carter. *On the existence of a projective plane of order ten.* PhD thesis, University of California, Berkeley, 1974.

[16] K. Clarkson and S. Whitesides. On the non-existence of maximal 6-arcs in projective planes of order 10. In *Poster session at IWOCA 2014, the 25th International Workshop on Combinatorial Algorithms*, 2014.

[17] J. H. Davenport, M. England, A. Griggio, T. Sturm, and C. Tinelli. Symbolic computation and satisfiability checking. *Journal of Symbolic Computation*, 100:1–10, 2020.

[18] J. Devriendt, B. Bogaerts, and M. Bruynooghe. Symmetric explanation learning: Effective dynamic symmetry handling for SAT. In *Theory and Applications of Satisfiability Testing – SAT 2017*, pages 83–100. Springer International Publishing, 2017.

[19] D. Ž. Đoković. Williamson matrices of order $4n$ for $n = 33, 35, 39$. *Discrete Mathematics*, 115(1-3):267–271, 1993.

[20] D. Ž. Đoković and I. S. Kotsireas. Compression of periodic complementary sequences and applications. *Designs, Codes and Cryptography*, 74(2):365–377, 2015.

[21] S. W. Golomb and L. D. Baumert. The search for Hadamard matrices. *The American Mathematical Monthly*, 70(1):12–17, 1963.

[22] J. Hadamard. Résolution d'une question relative aux déterminants. *Bulletin des Sciences Mathematiques*, 17(1):240–246, 1893.

[23] A. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[24] T. C. Hales and S. P. Ferguson. *The Kepler Conjecture: The Hales-Ferguson Proof.* Springer Science & Business Media, 2011.

[25] M. J. H. Heule. Schur number five. In S. A. McIlraith and K. Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 6598–6606. AAAI Press, 2018.

[26] M. J. H. Heule and O. Kullmann. The science of brute force. *Communications of the ACM*, 60(8):70–79, 2017.

[27] J. Horáček. *Algebraic and Logic Solving Methods for Cryptanalysis.* PhD thesis, University of Passau, 2020.

[28] J. Horáček and M. Kreuzer. On conversions from CNF to ANF. *Journal of Symbolic Computation*, 100:164–186, Sept. 2020.

[29] J. P. Inala, S. Gao, S. Kong, and A. Solar-Lezama. REAS: Combining numerical optimization with SAT solving. *arXiv preprint arXiv:1802.04408*, 2018.

[30] R. Jhala and R. Majumdar. Software model checking. *ACM Computing Surveys (CSUR)*, 41(4):1–54, 2009.

[31] D. Kaufmann, A. Biere, and M. Kauers. SAT, computer algebra, multipliers. In L. Kovács and A. Voronkov, editors, *Vampire 2018 and Vampire 2019. The 5th and 6th Vampire Workshops*, volume 71 of *EPiC Series in Computing*, pages 1–18. EasyChair, 2020.

[32] C. W. H. Lam. The search for a finite projective plane of order 10. *The American Mathematical Monthly*, 98(4):305–318, 1991.

[33] C. W. H. Lam, L. Thiel, and S. Swiercz. The nonexistence of code words of weight 16 in a projective plane of order 10. *Journal of Combinatorial Theory, Series A*, 42(2):207–214, 1986.

[34] C. W. H. Lam, L. Thiel, and S. Swiercz. The non-existence of finite projective planes of order 10. *Canadian Journal of Mathematics*, 41(6):1117–1123, 1989.

[35] F. J. MacWilliams, N. J. A. Sloane, and J. G. Thompson. On the existence of a projective plane of order 10. *Journal of Combinatorial Theory, Series A*, 14(1):66–78, 1973.

[36] J. P. Marques-Silva and K. A. Sakallah. Boolean satisfiability in electronic design automation. In *Proceedings of the 37th Annual Design Automation Conference*, pages 675–680, 2000.

[37] B. D. McKay and A. Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, Jan. 2014.

[38] H. Metin, S. Baarir, M. Colange, and F. Kordon. CDCLSym: Introducing effective symmetry breaking in SAT solving. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 99–114. Springer International Publishing, 2018.

[39] R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM (JACM)*, 53(6):937–977, 2006.

[40] M. Plotkin. Binary codes with specified minimum distance. *IRE Transactions on Information Theory*, 6(4):445–450, 1960.

[41] D. J. Roy. Confirmation of the non-existence of a projective plane of order 10. Master's thesis, Carleton University, 2011.

[42] A. Sabharwal. `SymChaff`: Exploiting symmetry in a structure-aware satisfiability solver. *Constraints*, 14(4):478–505, Oct. 2008.

[43] M. Y. Vardi. Boolean satisfiability: Theory and engineering. *Communications of the ACM*, 57(3):5, 2014.

[44] J. Williamson. Hadamard's determinant theorem and the sum of four squares. *Duke Mathematical Journal*, 11(1):65–81, 1944.

[45] E. Zulkoski, C. Bright, A. Heinle, I. Kotsireas, K. Czarnecki, and V. Ganesh. Combining SAT solvers with computer algebra systems to verify combinatorial conjectures. *Journal of Automated Reasoning*, 58(3):313–339, 2017.

[46] E. Zulkoski, V. Ganesh, and K. Czarnecki. MathCheck: A math assistant via a combination of computer algebra systems and SAT solvers. In A. P. Felty and A. Middeldorp, editors, *Proceedings of the 25th International Conference on Automated Deduction*, volume 9195 of *Lecture Notes in Computer Science*, pages 607–622. Springer, 2015.