Applicable Algebra in Engineering, Communication and Computing manuscript No.
(will be inserted by the editor)

A Nonexistence Certificate for Projective Planes of Order Ten with Weight 15 Codewords

Curtis Bright · **Kevin Cheung** · **Brett Stevens** · **Dominique Roy** · **Ilias Kotsireas** · **Vijay Ganesh**

Received: October 30, 2019 / Accepted: February 14, 2020

Abstract Using techniques from the fields of symbolic computation and satisfiability checking we verify one of the cases used in the landmark result that projective planes of order ten do not exist. In particular, we show that there exist no projective planes of order ten that generate codewords of weight fifteen, a result first shown in 1973 via an exhaustive computer search. We provide a simple satisfiability (SAT) instance and a certificate of unsatisfiability that can be used to automatically verify this result for the first time. All previous demonstrations of this result have relied on search programs that are difficult or impossible to verify—in fact, our search found partial projective planes that were missed by previous searches due to previously undiscovered bugs. Furthermore, we show how the performance of the SAT solver can be dramatically increased by employing functionality from a computer algebra system (CAS). Our SAT+CAS search runs significantly faster than all other published searches verifying this result.

Keywords Combinatorial search · Projective planes · Symbolic computation · Satisfiability checking · SAT+CAS

1 Introduction

A projective plane is a geometric structure where parallel lines do not exist. In other words, any two lines in a projective plane must meet at some point, a property that does not hold in the

C. Bright
University of Waterloo
E-mail: cbright@uwaterloo.ca
Webpage: `https://cs.uwaterloo.ca/~cbright/`

K. Cheung, B. Stevens
Carleton University

D. Roy
Canada Revenue Agency

I. Kotsireas
Wilfrid Laurier University

V. Ganesh
University of Waterloo

standard Euclidean plane. The existence of non-Euclidean planes is initially counterintuitive but they have been widely studied since the beginning of the 18th century. As a simple example of this phenomenon, consider the case of geometry on a sphere. In this case the lines on a sphere are the "great circles" of the sphere and any two distinct lines intersect in exactly two antipodal points.

A more exotic type of geometry known as *finite geometry* occurs when only a finite number of points exist. In this article we are concerned with finite projective geometry, i.e., geometry that include axioms that say that only a finite number of points exist and that parallel lines do not exist. A *finite projective plane* is a model of the finite projective geometry axioms (see Section 2 for the complete list). In particular, a finite projective plane is said to be of order n if there are $n+1$ points on every line.

An important open question in finite geometry concerns the orders n for which finite planes exist. Finite projective planes of order n can be constructed whenever n is a prime power but it is unknown if any exist when n is not a prime power. Despite a significant amount of effort no one has ever been able to construct a finite plane in an order n that is not a prime power and it has been widely conjectured that such a plane cannot exist [13].

A partial result was proven by Bruck and Ryser [11] who showed that n must be the sum of two integer squares if a finite plane of order n exists with n congruent to 1 or 2 $\pmod 4$. Bruck and Ryser's result implies that a projective plane of order six cannot exist. Every other $n < 10$ is a prime power and therefore a finite plane of order n does exist; the smallest order that is not a prime power and not covered by the Bruck–Ryser theorem is ten.

A first step towards solving the existence question in order ten was completed by MacWilliams, Sloane, and Thompson [42]. In their paper the error-correcting code generated by a hypothetical projective plane of order ten was studied. In particular, they showed that the search could be reduced to four cases that they called the weight 12, 15, 16, and 19 cases (see Section 2). Furthermore, they used a computer search to show that the weight 15 case did not lead to a projective plane of order ten.

In the 1980s a number of extensive computer searches were performed to settle the question of existence of a projective plane of order ten. In particular, the weight 12 case was solved by Lam, Thiel, Swiercz, and McKay [40], the weight 16 case was solved by Lam, Thiel, and Swiercz [38] (continuing work by Carter [14]) and the weight 19 case was solved by Lam, Thiel, and Swiercz [39], finally showing that a projective plane of order ten does not in fact exist.

Each of these cases required a significant amount of computational resources to solve, including about 2.7 months of computing time on a CRAY-1A supercomputer to solve the weight 19 case. More recently, Roy [48] performed a verification of the nonexistence of the projective plane of order ten using 3.2 months of computing time with 15 CPU cores running at 2.4 GHz. Several recent works [15,16,46] have also performed verifications of the weight 15 case using custom-written search code. These verifications used the programming language C or the programming languages of the computer algebra systems GAP and Mathematica. Additionally, Bruen and Fisher [12] showed that the weight 15 case can be solved using a result of Denniston [21] but this was also obtained via a computer search.

In this paper we perform a verification of the weight 15 case using the properties derived by MacWilliams, Sloane, and Thompson [42]. Our verification is unique in that we translate the properties of a projective plane into Boolean logic and then perform the search using a SAT solver. SAT solvers are known to be some of the best tools to perform combinatorial searches; for example, Heule, Kullmann, and Marek [29] state that today they are the "best solution" for most kinds of combinatorial searches. Even so, they mention that there are some

problems that SAT solvers have not yet been successfully applied to. In fact, they explicitly list the search for a projective plane of order ten as one of these problems:

> An example where only a solution by [special purpose solvers] is known is the determination that there is no projective plane of order 10 [...] To the best of our knowledge the effort has not been replicated, and there is definitely no formal proof.

The fact that we perform our search using a SAT solver means that we can produce a formally verifiable *certificate* that the weight 15 case does not lead to a solution. In contrast to all previous searches that have been completed, one can verify our results without needing to trust the particular choice of hardware, compiler, or search algorithms that we happened to use in our verification. Instead, one merely needs to trust our encoding of the problem into SAT (see Section 3) and our SAT instance generation script (see Section 4).

We do not claim our verification is a *formal proof* of nonexistence because it relies on mathematical results that (at least currently) have no machine-verifiable formal proof. However, compared to previous approaches our verification has the advantage that it is not necessary to trust code that implements a search algorithm. This is particularly important considering that efficient search algorithms often need to be written in a convoluted way to obtain optimum performance. In fact, while verifying that our SAT encoding was producing correct results we uncovered bugs in previous searches (see Section 4).

Our result is also a first step towards a formal proof. The SAT encoding is deliberately chosen to be as simple as possible so that (1) the possibility of an encoding bug is less likely, and (2) it will be as simple as possible to formally generate the SAT clauses directly from the axioms that define a projective plane of order ten. This approach of reducing a problem to SAT, solving the resulting SAT instance, formally verifying the nonexistence certificate, and then finally formally verifying the SAT encoding in a theorem prover has recently successfully formally verified the proofs of the Boolean Pythagorean triples conjecture [19,28] and the Erdős discrepancy conjecture for discrepancy up to three [33,34].

We also show how to use a computer algebra system (CAS) to greatly improve the efficiency of the SAT solver (see Section 3.2). This "SAT+CAS" approach of combining SAT solvers with computer algebra systems has recently been applied to a variety of problems including verifying the correctness of Boolean arithmetic circuits [32], finding new algorithms for 3×3 matrix multiplication [27], and finding or disproving the existence of certain kinds of combinatorial designs [7]. For more detailed surveys on the SAT+CAS paradigm and the kinds of problems that it has been applied to see [9,20]. Our SAT+CAS approach for solving the weight 15 case performs better than all other searches that have been previously published (see Section 4.3).

2 Projective plane preliminaries

A finite projective plane of order n consists of a set of lines and a set of points that satisfy the following axioms:

P1. There are $n+1$ points on every line and there are $n+1$ lines through every point.
P2. There is exactly one line between every two distinct points and every two distinct lines intersect in exactly one point.

A consequence of these axioms is that a finite projective plane of order n contains exactly n^2+n+1 points and exactly n^2+n+1 lines [31].

One convenient way of representing a finite projective plane is by an incidence matrix that encodes which points lie on which lines. This matrix has a 1 in the (i,j)th entry if point j is on line i and has a 0 in the (i,j)th entry otherwise. In this representation axiom P2 says that every pair of columns or pair of rows intersect exactly once (where two columns or rows *intersect* if they both contain a 1 in the same location). The number of times that two columns or rows intersect is given by the inner product (over the reals) of the two columns or rows, so axiom P2 says that the inner product product of any two columns or rows is exactly one.

It follows that a finite projective plane of order ten is equivalent to a $\{0,1\}$-matrix of size 111×111 such that:

P1. Every row and column contains exactly eleven 1s.
P2. The inner product of any two distinct rows or two distinct columns is exactly 1.

The ordering of the rows and columns of the incidence matrix of a projective plane is arbitrary and we say that two matrices are *equivalent* if one matrix can be transformed into the other by a reordering of the rows and columns.

Suppose P is the incidence matrix of a hypothetical projective plane of order ten. The elements of the row space of P (mod 2) are known as the *codewords* of P and the number of 1s in a codeword is known as the *weight* of the codeword. Let w_k denote the number of codewords of P of weight k. For example, $w_0 = 1$ because the zero vector is in the row space of P and no other codeword has a weight of zero. Also, it was shown by Assmus and Mattson [3] that the values of all w_k for $0 \leq k \leq 111$ can be determined from just the values of w_{12}, w_{15}, and w_{16}.

The relationships between the values of w_k are what ultimately lead to the contradiction that showed that a projective plane of order ten cannot exist. For example,

$$w_{12} = w_{15} = w_{16} = 0 \qquad \text{imply that} \qquad w_{19} = 24{,}675.$$

However, a number of exhaustive computer searches [14,38,39,40,42] found no codewords of weights 12, 15, 16, or 19—thus implying the hypothetical projective plane P cannot exist.

In fact, it is known that w_{15} and w_{19} cannot both be zero [14,24]. Furthermore, Carter [14] and Hall [24] show that the weight 19 codewords either arise from weight 12 codewords, weight 16 codewords, or are of a third kind that they call primitive. They show that if w_{15} and w_{16} are zero then the number of primitive weight 19 codewords must be positive. Thus, to show that a projective plane of order ten does not exist it suffices to show that there exists no codewords of P of weights 15, 16, or primitive weight 19 codewords. In the remainder of this paper we describe the construction of a SAT instance that necessarily has a solution if a codeword of P of weight 15 exists. We then provide a certificate that this SAT instance is unsatisfiable and therefore solve one of the three cases necessary to prove the nonexistence of a projective plane of order ten.

2.1 Incidence matrix structure

MacWilliams et al. [42] derive a number of properties that the structure of a projective plane of order ten with weight 15 codewords must satisfy. In particular, they show that up to equivalence the incidence matrix of such a projective plane can be partitioned into a 3×3 grid of submatrices as follows:

$$\begin{array}{rc} & \begin{array}{ccc} 15 & 60 & 36 \end{array} \\ \begin{array}{r} \text{heavy} \\ \text{medium} \\ \text{light} \end{array} & \begin{array}{c} 6 \\ 15 \\ 90 \end{array} \left(\begin{array}{ccc} 5 & 0 & 6 \\ 3 & 8 & 0 \\ 1 & 6 & 4 \end{array} \right) \end{array}$$

Here the numbers outside the matrix denote the number of rows or columns in that part of the matrix and the numbers inside the matrix are the number of 1s that appear in each row of the submatrix in that part of the matrix. MacWilliams et al. [42] call the first 15 columns the A points, the next 60 columns the C points, and the remaining columns the B points. Furthermore, Roy [48] calls the first 6 rows the *heavy* lines, the next 15 rows the *medium* lines, and the remaining rows the *light* lines.

MacWilliams et al. [42] also show that up to equivalence there is exactly one way of assigning the 1s in each submatrix except for the last two submatrices of the last row. Furthermore they provide an explicit representation of the unique assignment up to equivalence.

2.2 Initial entries

Up to equivalence, a number of entries of a projective plane of order ten containing a weight 15 codeword can be initialized in advance, including all entries in the first 21 rows and 15 columns [42]. In our search we focus on the first 75 columns and 51 rows of the projective plane. The entries of this submatrix that we fix in advance are shown in Figure 1.

The first 6 rows (the heavy lines) of the projective plane are taken to be identical with the representation of MacWilliams et al. [42]. The next 21 rows (the medium lines) are equivalent to MacWilliams' representation—we have only applied a column permutation to their representation to more clearly explain how we assigned the 1s in the later rows. In particular, the specific ordering of columns 16–75 that was used in Figure 1 was chosen in order to allow initializing the diagonal line of 1s that appears in rows 22–51 (see below).

The first 15 columns in Figure 1 are also specified to be identical to the representation given by MacWilliams et al. [42]. They choose to order the rows so that the light rows containing a 1 in the first column appear first, followed by the light lines containing a 1 in the tenth, fifteenth, eleventh, and fourteenth columns (in that order). This ordering was chosen in an attempt to maximize the number of overlapping columns with unassigned entries in the light rows. While searching for completions of the unassigned entries, conflicts between two light rows will occur in columns where both light rows contain unassigned entries—therefore maximizing the number of overlapping columns with unassigned entries tends to increase the number of conflicts and speed up the search.

This leaves the lower-right 30×60 submatrix of Figure 1. The zeros that appear in this submatrix are easily determined; if they were 1s then the column that they are on would intersect more than once with another column. For example, consider the 22nd entry of the 22nd line—if this entry was a 1 then the first and 22nd column would intersect twice, in contradiction to the matrix of Figure 1 being a partial projective plane.

Next we show that the diagonal line of 1s that appears on the left of the 30×60 submatrix can be assumed without loss of generality. For example, consider the light rows that contain a 1 in the first column (rows 22–27). By the projective plane axiom P2, these lines must share a point of intersection with the fourth medium line (row 10). By inspection, there are exactly six possible columns for this point of intersection (namely, columns 16–21). In other words, there must be a 1 in each row of the submatrix given by rows 22–27 and columns 16–21.

Since each of the rows 22–27 are already pairwise intersecting in the first column they must not be pairwise intersecting in the columns 16–21. Similarly, each of the columns 16–21 are already pairwise intersecting in the tenth row so they must not be pairwise intersecting in the rows 22–27. In other words, each row and column of the submatrix given by rows 22–27 and columns 16–21 contains at most a single 1. By reordering the rows 22–27 we can assume

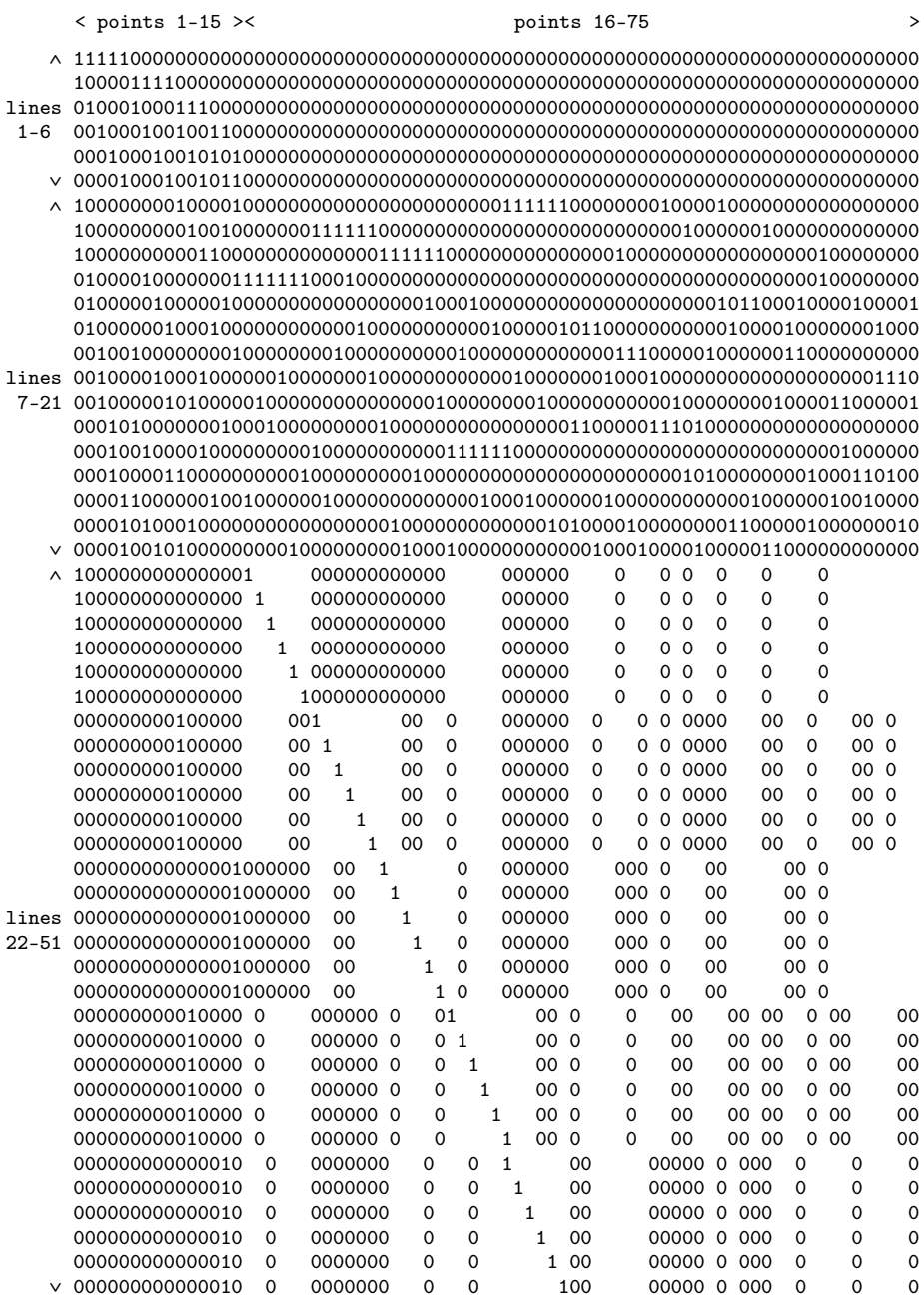

Fig. 1: Our initial instantiation of the first 75 columns and 51 rows of P where blank spaces represent unknown entries.

without loss of generality that the submatrix given by rows 22–27 and columns 16–21 is the identity matrix.

This explains why we may initialize the diagonal 1s that appear in the rows 22–27 of Figure 1. The same reasoning explains the initializations in the rows 28–33 (with columns 22–27), rows 34–39 (with columns 28–33), rows 40–45 (with columns 34–39), and rows 46–51 (with columns 39–44). Note that in the final case the set of columns that are used overlaps with columns that are used in the second last case, and this causes the diagonal to be offset in the last six rows.

Once the 1s in the lower-right submatrix of Figure 1 have been assigned some previously undetermined entries can be set to 0 but for simplicity we do not include these in Figure 1. In any case, it is mostly inconsequential if these entries are included in our initial instantiation or not. If they are not given the SAT solver will almost immediately discern these entries using Boolean constraint propagation on the clauses used in our projective plane encoding.

The entries of Figure 1 were all derived using mathematical arguments and not via a computer search. A SAT solver could also be used to derive some of these entries, but this would require a more complicated encoding (see Section 3). Thus, we prefer to take the entries of Figure 1 as given and fixed in advance.

3 SAT encoding

In this section we describe the encoding we used to show the nonexistence of codewords of weight 15 in a projective plane of order ten. Our encoding uses the Boolean variables $p_{i,j}$ where i and j are between 1 and 111. When $p_{i,j}$ is true it represents that the (i,j)th entry of P is 1 and when $p_{i,j}$ is false it represents that the (i,j)th entry of P is 0. Thus, when the (i,j)th entry in Figure 1 is a 1 we include the unit clause $p_{i,j}$ in our SAT instance and when the entry is a 0 we include the unit clause $\neg p_{i,j}$ in our SAT instance.

3.1 Incidence constraints

We now describe the constraints we used to specify that the incidence matrix defined by the Boolean variables $p_{i,j}$ forms a projective plane. In particular, axiom P2 from Section 2 says that all rows and columns intersect exactly once. We encode this axiom by splitting it up into the following two constraints:

1. The pairwise row and column inner products of P are *at most* one.
2. The pairwise row and column inner products of P are *at least* one.

Furthermore, in the second case, we found that it was only necessary to consider inner products between the medium rows and the light rows and the inner products between the first 15 columns and the later columns. We also only used the first 51 rows and 75 columns of P. Our searches found no satisfying assignments of even this strictly smaller set of constraints. The fact that these constraints are unsatisfiable therefore shows more than just the nonexistence of weight 15 codewords; we also show the nonexistence of partial projective planes that complete Figure 1.

We do not directly encode axiom P1 in our SAT instances. This axiom is present merely to exclude "degenerate" cases from being considered as projective planes (where the rows of degenerate projective planes of order ten have weights 1, 2, 110, or 111). However, an examination of even the first row of Figure 1 shows that degenerate cases are naturally

excluded by our encoding. Thus, the completions of P that satisfy axiom P2 naturally satisfy axiom P1. It is possible to encode axiom P1 in conjunctive normal form (for example, by using a sequential counter encoding [49]). However, this introduces new variables and in our experiments decreased the performance of the SAT solver.

3.1.1 Encoding that columns and rows intersect at most once

Consider rows i and j for arbitrary $1 \leq i, j \leq 111$ with $i \neq j$. To enforce that these rows intersect at most once we must enforce that there do not exist column indices k and l (where $1 \leq k, l \leq 111$ and $k \neq l$) such that $p_{i,k}$, $p_{i,l}$, $p_{j,k}$, and $p_{j,l}$ are all simultaneously true. In other words, for each pair of distinct indices (i, j) and (k, l) at least one variable $p_{i,k}$, $p_{i,l}$, $p_{j,k}$, or $p_{j,l}$ must be false; this also implies that pairwise all columns intersect at most once. As clauses in conjunctive normal form we encode this as

$$\bigwedge_{i<j} \bigwedge_{k<l} (\neg p_{i,k} \vee \neg p_{i,l} \vee \neg p_{j,k} \vee \neg p_{j,l}).$$

3.1.2 Encoding that columns and rows intersect at least once

Consider rows i and j for arbitrary $1 \leq i, j \leq 111$ with $i \neq j$. To enforce that these rows intersect at least once we must enforce that there exist a column index k with $1 \leq k \leq 111$ such that $p_{i,k}$ and $p_{j,k}$ are simultaneously true. This can be encoded as

$$\bigwedge_{i<j} \bigvee_{k} (p_{i,k} \wedge p_{j,k}),$$

however, this formula is not in conjunctive normal form (CNF) and therefore cannot be used directly with a standard SAT solver. However, in certain cases this formula easily simplifies into a formula in CNF.

In particular, consider the case when i is the index of a medium line and j is the index of a light line (i.e., $7 \leq i \leq 21$ and $22 \leq j \leq 111$). In this case, the truth values of the variables $p_{i,k}$ for all $1 \leq k \leq 111$ can be determined in advance. Their values are forced by the unit clauses encoding the entries displayed in Figure 1 and the fact that each row i is already known to contain eleven 1s (so $p_{i,k}$ must be false for $76 \leq k \leq 111$).

Let $S(i)$ denote the indices k such that $p_{i,k}$ is true. Then

$$\bigvee_{1 \leq k \leq 111} (p_{i,k} \wedge p_{j,k}) \quad \text{simplifies to} \quad \bigvee_{k \in S(i)} p_{j,k}.$$

Furthermore, the expression on the right can be determined in advance since the entries of row i of P are completely known.

We encode the fact that the columns k and l intersect at least once (where $1 \leq k \leq 15$ and $16 \leq l \leq 75$) in a similar way. Let $T(k)$ denote the set of row indices i such that $p_{i,k}$ is true; since the first 15 columns of P are known these sets can be determined in advance. Then we encode the fact that columns k and l intersect at least once by $\bigvee_{i \in T(k)} p_{i,l}$.

Altogether we encode these constraints in conjunctive normal form by

$$\bigwedge_{\substack{7 \leq i \leq 21 \\ 22 \leq j \leq 51}} \bigvee_{k \in S(i)} p_{j,k} \qquad \text{and} \qquad \bigwedge_{\substack{16 \leq l \leq 75 \\ k \in \{1,10,11,14,15\}}} \bigvee_{i \in T(k)} p_{i,l}.$$

Note that we have limited ourselves to only using variables from the first 51 rows and 75 columns in these clauses—in the right formula we only use the k for which the entries in $T(k)$ are from the set $\{22,\ldots,51\}$.

More than 51 rows and 75 columns can be used but these values were chosen in an attempt to minimize the number of variables necessary to show the impossibility completing the initial values of P into a complete projective plane. In particular, the entries of $S(i)$ are from the set $\{16,\ldots,75\}$ so it is necessary to use at least 75 columns. Using a smaller number of rows is possible—for example, only 45 rows could be used by taking $k \in \{1,10,11,15\}$ in the right formula. However, this SAT instance was experimentally found to be satisfiable (see Figure 3 in Section 4.1).

3.2 Symmetry breaking

When search spaces are highly symmetric, SAT solvers generally perform poorly because they typically have not been optimized to detect symmetries. Thus, in the presence of symmetry a SAT solver will tend to repeat the same search for every symmetry that exists. A common method of improving their performance is to add constraints that eliminate or "break" the symmetry [18]. Symmetry breaking is not essential to our method, but it does greatly increase its effectiveness. In Section 4 we provide timings for our method both with and without symmetry breaking.

The *symmetry group* of a given matrix is the set of row and column permutations that fix its entries. As an explicit example, consider the upper-left 6×15 submatrix of P:

```
111110000000000
100001111000000
010001000111000
001000100100110
000100010010101
000010001001011
```

The symmetry group of this matrix is isomorphic to S_6, the symmetric group of degree 6. It contains $6! = 720$ distinct permutations, including, for example, the permutation that swaps the first two rows and swaps column i with column $i+4$ for $2 \leq i \leq 5$. The symmetry group of the upper-left 21×75 submatrix of P is also isomorphic to S_6 and we call this symmetry group S.

We may apply the permutations from S to partial completions of the first 75 columns of P. Such an action necessarily produces another equivalent partial completion (up to a reordering of the light rows). In our search, we attempt to eliminate as many equivalent partial completions from the search space as possible, leaving only partial completions that are not equivalent to each other.

We now focus on the first six light rows (rows 22–27) and the completions of those rows. We use the permutations in S to transform completions of this submatrix into other equivalent completions of this submatrix. We only consider permutations of S that fix the first column of this submatrix, since it is not possible for any permutation that moves the first column to fix the upper-left 27×75 submatrix of Figure 1 (due to the 1s on the first column of rows 22–27). The subgroup of S fixing the first column of the upper-left 21×75 submatrix of P is isomorphic to $S_4 \times S_2$ and contains $4! \cdot 2 = 48$ permutations.

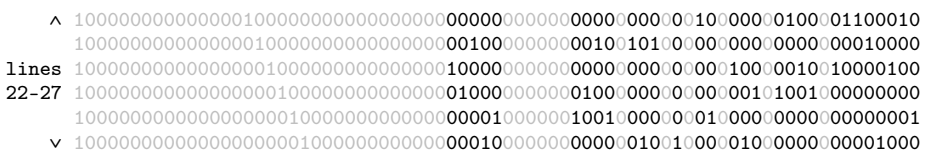

Fig. 2: A completion of rows 22–27 and the first 75 columns. Gray entries denote initially known entries and black entries denote the completions of the initially unassigned entries.

3.2.1 Programmatic SAT

We now describe our symmetry breaking method using the programmatic SAT paradigm [22]. A *programmatic SAT* solver can learn clauses in a programmatic fashion, for example through a piece of code that queries a computer algebra system. In our application, whenever a completion of the upper-left 27×75 submatrix of P is found we record it and programmatically learn clauses that block the completion as well as all completions that are equivalent to it. The search then continues until all inequivalent completions have been recorded.

See Figure 2 for an explicit example of such a completion. Say that C is the set of variables whose values were initially unknown but have been assigned to true in a completion. The set C must contain exactly 30 variables since there are 30 columns with unknown variables and each unassigned column must contain a single 1 in order to intersect with the first column exactly once.

Once the SAT solver finds a valid C we programmatically learn the clause $\bigvee_{p \in C} \neg p$ which says to block the completion specified by C (in order to perform an exhaustive search for all completions). Additionally, suppose that φ is a permutation of S that fixes the first column (as described above). We let $\varphi(C)$ denote the set of true variables in the completion generated by applying φ to the completion specified by C. Explicitly, $\varphi(C)$ is computed by applying the column permutations of φ to all the variables in C, followed by applying row permutations so that the 1s in columns 16–21 remain fixed (as they may have been disturbed by the column permutations).

For each completion C found by the SAT solver we record C and learn the clauses

$$\bigvee_{p \in \varphi(C)} \neg p \qquad \text{for all symmetries } \varphi \text{ fixing the first column.}$$

These clauses block all completions that are equivalent to C, leaving only the nonequivalent completions in the search space. We then continue the search in this manner until no completions remain. At the conclusion of the search we will have recorded a list of all nonequivalent completions of the upper-left 27×75 submatrix of P, i.e., a list of all the completions up to symmetry.

4 Results

In this section we discuss our implementation, timings, and compare our results with those of previous searches.

4.1 Implementation

A Python script of less than 200 lines was written to generate a SAT instance containing the clauses described in Section 3 (our source code is available at `uwaterloo.ca/mathcheck`). The instance contains $51 \cdot 71 = 3825$ distinct variables and 79,248 distinct clauses, including 3075 unit clauses and therefore 750 unknown variables. The symmetry breaking method described in Section 3.2 was implemented using the programmatic SAT solver MapleSAT [41] with the symmetry groups and row and column permutations computed by the computer algebra system Maple 2019 [5].

MapleSAT found 42,496 completions of the rows 22–27 of P, of which 1021 of these completions were inequivalent. This naturally splits the search space into 1021 distinct subspaces, one for each nonequivalent completion of the rows 22–27. We now discuss how these completions can be used to help the SAT solver solve the SAT instance containing the remaining rows 28–51.

The simplest option is simply to generate a distinct SAT instance for each distinct completion of the rows 22–27 and to include the true variables that appear in the completion as unit clauses. However, to avoid the overhead of calling a SAT solver 1021 times it is better to generate a single *incremental SAT* (see [45]) instance that contains 1021 sets of assumptions. A second option is to use a single SAT instance and add *blocking clauses* for each of the $42{,}496 - 1021 = 41{,}475$ completions that are equivalent to one of the remaining 1021 completions. For example, if C is the set of variables assigned to true in a completion then the completion can be blocked by adding the clause $\bigvee_{p \in C} \neg p$ into the instance.

Experimentally it was determined that it is possible to find completions of P using the first 45 rows and 75 columns. In fact, MapleSAT was able to find explicit completions of the upper-left 45×75 submatrix in about 2 seconds (see Figure 2). The completion shown in Figure 2 is special in that regardless which representative chosen for rows 22–27 the representative can always be extended to 45 rows. In each of the other 1020 cases there is at least one representative of the rows 22–27 for which it is impossible to extend that representative to 45 rows.

4.2 Timings

We used the SAT solver MapleSAT [41] running on an Intel i7 CPU at 2.7 GHz for all timings unless otherwise specified. The base SAT instance using 51 rows and 75 columns and no symmetry breaking was shown to be unsatisfiable in 6.3 minutes. A DRUP proof of size 1.6 GB was produced and was verified using the proof checker DRAT-trim [50]. Interestingly, the cube-and-conquer paradigm [29] is particularly effective on SAT instances of this form. The CnC solver of M.J.H. Heule [25] outperformed the non-programmatic version of MapleSAT by solving the SAT instance in 5.2 minutes (1.9 minutes for cubing using March_cu [30] and 3.3 minutes for conquering using Glucose [4]). The CnC solver generated a proof of size 1.2 GB (or 410 MB after trimmed and compressed in the binary DRAT format).

Using our programmatic encoding as described in Section 3.2, MapleSAT found all 42,496 completions (and 1021 nonequivalent completions) of the first 27 rows in 2.4 seconds. Afterwards, the incremental SAT instance showing that none of the 1021 nonequivalent completions of the first 27 rows can be extended to 51 rows was solved in 4.4 seconds. There does not seem to be a standard proof format for incremental SAT instances, so this run was not formally verified. However, we did formally verify that the SAT instance containing the

Fig. 3: An assignment to the first 75 columns and rows 22–45 of P that produces a 45×75 partial projective plane.

symmetry "blocking clauses" was unsatisfiable. This instance contained clauses blocking the 41,475 completions of the first 27 rows that are known to be equivalent to one of the remaining 1021 completions. MapleSAT solved this instance in 13 seconds and produced a DRUP proof of size 147 MB (or 35 MB after trimmed and compressed in the binary DRAT format, available at `uwaterloo.ca/mathcheck`).

We did not attempt to optimize the generation of the SAT instances because we wanted to make the generation scripts as straightforward as possible. In any case, the generation of the SAT instances required less than a second using our Python script.

4.3 Comparison with previous searches

There have been a number of previous searches verifying the nonexistence of projective planes of order ten containing weight 15 codewords [15,16,21,42,46,47,48]. To the best of our knowledge, all previous searches have relied on computers in an essential way—there is no known "purely theoretical" explanation for this result.

The first searches by Denniston [21] and MacWilliams, Sloane, and Thompson [42] used punch cards or paper tape and so we are not able to rerun their searches on modern hardware. Denniston used 90 minutes on an Elliott 4130 mainframe computer and MacWilliams et al. used 3 hours on a General Electric 635 mainframe computer. Their searches were similar though Denniston used the light rows that intersect the columns numbered 1, 10, 15, and 5, and some different symmetry removal argumentation.

The next search that we are aware of is by Roy [47] in 2005 and was completed using a special-purpose search program written in the programming language C, relying on the nauty symbolic computation library [44] for solving the graph isomorphism problem. We ran this program on the same hardware we used for our SAT searches and it completed in 78 minutes.

Authors	Year	Language	Time
Roy [47]	2005	C	78 minutes
Casiello, Indaco, and Nagy [15]	2010	GAP	3.3 minutes
Clarkson and Whitesides [16]	2014	C	27 seconds
Perrott [46]	2016	Mathematica	55 minutes
Bright et al. [this work]	2019	SAT	6.3 minutes
Bright et al. [this work]	2019	SAT+CAS	6.8 seconds

Table 1: A comparison of the searches for weight 15 codewords for which source code is available.

Casiello, Indaco, and Nagy [15] completed another search in 2010 using the computer algebra system GAP [23]. However, they claim to find no completions of the first 39 rows and 75 columns (using the same inital 21 rows and 15 columns from Figure 1). Closer inspection of their source code revealed that incorrect indices were used in one block compatibility check and this caused them to assert the nonexistence of a partial projective plane that actually exists (as demonstrated in Figure 3). After fixing the incorrect indices their program does appear to produce correct results and requires about 3.3 minutes (on the same hardware from above) to assert the nonexistence of completions of the first 51 rows and 75 columns.

Next, Clarkson and Whitesides [16] completed a search in 2014 using the light rows that intersect the columns numbered 1, 10, 15, 5, and 8. It was performed using a custom-written and highly optimized search program in the programming language C. We obtained the source code from the authors and it completed in 27 seconds on the same hardware from above.

In 2016, Perrott [46] completed a search using the computer algebra system Mathematica [51]. We were not able to run his code on the same hardware as above (due to the fact Mathematica is proprietary software). However, his search completed in 55 minutes running on an Intel Xeon X5675 at 3.07 GHz.

Roy [48] performed a search in 2011 using a similar approach as MacWilliams et al. [42] except attempting to complete the partial matrix column-by-column instead of row-by-row and using the columns that were labeled 76–111 in this paper. These 36 columns can be split into six *blocks* of six columns each, each block consisting of the columns that are incident to one of the first six rows of P. This search found no completions of the first four blocks in 2.3 minutes. We attempted to verify this result, but found completions of all six blocks (available at `uwaterloo.ca/mathcheck`). Thus, we conclude there was likely an undiscovered bug in the search program of [48] that caused the search to be incomplete. Unfortunately, this source code is no longer available so it is impossible to determine the source of the discrepancy for certain.

In comparison, we completed our search using MapleSAT (augmented with a programmatic learning method as described in Section 3.2) in 6.8 seconds. Of this time, 2.4 seconds was used to exhaustively find all nonequivalent completions of the first 27 rows and 4.4 seconds was used to show that all of these completions do not extend to 51 rows. Even counting the time it takes to generate the SAT instances and the overhead of calling MapleSAT, our search terminates in under 8 seconds. A comparison of the searches for which source code is available is shown in Table 1.

5 Related work

As recounted in the introduction, SAT solvers have been used to perform searches in many different combinatorial problems. Some of the first successes were computing van der Waerden numbers by Kouril and Paul [35] and Ahmed, Kullmann, and Snevily [2], computing Green–Tao numbers by Kullmann [36], as well as solving a special case of the Erdős discrepancy conjecture by Konev and Lisitsa [34]. Other more recent combinatorial applications include proving the Boolean Pythagorean triples conjecture [28] and a new case of the Ruskey–Savage conjecture [52], as well as computing Ramsey numbers [17], Williamson matrices [8], complex Golay sequences [10], and Schur numbers [26].

We are not aware of any work prior to ours that used SAT solvers to search for (or prove the nonexistence of) projective planes. However, there has been work formalizing the axioms of projective planes in the theorem prover Coq by Magaud, Narboux, and Schreck [43] and Braun, Magaud, and Schreck [6].

6 Conclusion

In this paper we have performed a verification of the first nonexistence result that was crucial in the renowned proof that a projective plane of order ten does not exist [37]. In particular, we verified that a projective plane of order ten does not generate codewords of weight fifteen. There have been a number of exhaustive searches for such a codeword but all previous searches are difficult to verify.

In particular, the searches [21,42,48] rely on code that can no longer be run. The paper [15] provides source code but as described in Section 4.3 their code has a bug that caused them to assert the nonexistence of a partial projective plane that we found actually exists. The paper [46] verifies the same result that we verified in this paper but does so using about ten pages of sophisticated Mathematica code, and similarly the works [16,47] rely on optimized C programs that are difficult to verify.

In contrast, we have given a simple translation of properties of a weight fifteen codeword into Boolean logic and have shown that these properties are sufficient to prove that such a codeword cannot exist. This was done by a simple Python script that generates a SAT instance that encodes the necessary properties in conjunctive normal form. Furthermore, we solved the resulting SAT instance and provide a 439 MB formally verifiable certificate that the SAT instance indeed has no solution (or a 37 MB certificate with symmetry breaking clauses included).

Additionally, in Section 3.2 we showed how to use a programmatic SAT solver coupled with a computer algebra system to perform symmetry breaking and greatly decrease the running time of the search. In particular, an off-the-shelf version of MapleSAT completed the search in 6.3 minutes, while a programmatic version of MapleSAT augmented with our CAS symmetry breaking method completed the search in 6.8 seconds. This demonstrates the utility of the SAT+CAS method and provides further evidence (as originally argued by [1] and independently by [53]) for the power of the method.

Our work shows for the first time that SAT solvers can be effectively used in the search for finite projective planes—our code is currently the fastest available that can verify the weight 15 nonexistence result (see Table 1). Although we do not provide a machine-checkable formal proof directly from the axioms of a projective plane we have performed the most rigorous verification of this nonexistence result to date. In particular, our code is much simpler than the code used in any previous approach to this problem. We were able to simplify the

code by relying on a SAT solver to do the hard exhaustive search work. As a bonus this also produces a nonexistence certificate that can be formally independently verified.

We expect the SAT+CAS paradigm to also be useful in proving the nonexistence of codewords of weight 16 and 19 as well—in fact, our system has already solved the former case. The basic SAT encoding is very similar to the encoding provided in this paper, though the symmetry breaking method used in this paper is specific to the weight 15 search. A number of structural differences between the cases cause the weight 16 and 19 searches to be more complicated—this will be the subject of future research.

References

1. Ábrahám, E.: Building bridges between symbolic computation and satisfiability checking. In: S. Linton (ed.) Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation, pp. 1–6. ACM (2015)
2. Ahmed, T., Kullmann, O., Snevily, H.: On the van der Waerden numbers $w(2;3,t)$. Discrete Applied Mathematics **174**, 27–51 (2014)
3. Assmus Jr., E.F., Mattson Jr., H.F.: On the possibility of a projective plane of order 10. Algebraic Theory of Codes II, Air Force Cambridge Research Laboratories Report AFCRL-71-0013, Sylvania Electronic Systems, Needham Heights, Mass (1970)
4. Audemard, G., Simon, L.: Predicting learnt clauses quality in modern SAT solvers. In: C. Boutilier (ed.) IJCAI-09: Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence, pp. 399–404 (2009)
5. Bernardin, L., Chin, P., DeMarco, P., Geddes, K.O., Hare, D.E.G., Heal, K.M., Labahn, G., May, J.P., McCarron, J., Monagan, M.B., Ohashi, D., Vorkoetter, S.M.: Maple programming guide (2019)
6. Braun, D., Magaud, N., Schreck, P.: Formalizing some "small" finite models of projective geometry in Coq. In: J. Fleuriot, D. Wang, J. Calmet (eds.) International Conference on Artificial Intelligence and Symbolic Computation, pp. 54–69. Springer (2018)
7. Bright, C., Đoković, D., Kotsireas, I., Ganesh, V.: A SAT+CAS approach to finding good matrices: New examples and counterexamples. In: P. Van Hentenryck, Z.H. Zhou (eds.) Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19), pp. 1435–1442. AAAI Press (2019)
8. Bright, C., Kotsireas, I., Ganesh, V.: A SAT+CAS method for enumerating Williamson matrices of even order. In: S.A. McIlraith, K.Q. Weinberger (eds.) Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), pp. 6573–6580. AAAI Press (2018)
9. Bright, C., Kotsireas, I., Ganesh, V.: SAT solvers and computer algebra systems: A powerful combination for mathematics. In: T. Pakfetrat, G. Jourdan, K. Kontogiannis, R. Enenkel (eds.) Proceedings of the 29th International Conference on Computer Science and Software Engineering, pp. 323–328 (2019)
10. Bright, C., Kotsireas, I., Heinle, A., Ganesh, V.: Enumeration of complex Golay pairs via programmatic SAT. In: C. Arreche (ed.) Proceedings of the 2018 ACM International Symposium on Symbolic and Algebraic Computation, ISSAC '18, pp. 111–118. New York, NY, USA (2018)
11. Bruck, R.H., Ryser, H.J.: The nonexistence of certain finite projective planes. Canad. J. Math **1**(191), 9 (1949)
12. Bruen, A., Fisher, J.C.: Blocking sets, k-arcs and nets of order ten. Advances in Mathematics **10**(2), 317–320 (1973)
13. Bush, K.A.: Unbalanced Hadamard matrices and finite projective planes of even order. Journal of Combinatorial Theory, Series A **11**(1), 38–44 (1971)
14. Carter, J.L.: On the existence of a projective plane of order ten. University of California, Berkeley (1974)
15. Casiello, D., Indaco, L., Nagy, G.P.: Sull'approccio computazionale al problema dell'esistenza di un piano proiettivo d'ordine 10. Atti del Seminario matematico e fisico dell'Università di Modena e Reggio Emilia **57**, 69–88 (2010)
16. Clarkson, K., Whitesides, S.: 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. Codish, M., Frank, M., Itzhakov, A., Miller, A.: Computing the Ramsey number $R(4,3,3)$ using abstraction and symmetry breaking. Constraints **21**(3), 375–393 (2016)
18. Crawford, J.M., Ginsberg, M.L., Luks, E.M., Roy, A.: Symmetry-breaking predicates for search problems. In: L.C. Aiello, J. Doyle, S.C. Shapiro (eds.) Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning, KR'96, pp. 148–159. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1996)

19. Cruz-Filipe, L., Marques-Silva, J., Schneider-Kamp, P.: Formally verifying the solution to the Boolean Pythagorean triples problem. Journal of Automated Reasoning (2018)
20. Davenport, J.H., England, M., Griggio, A., Sturm, T., Tinelli, C.: Symbolic computation and satisfiability checking. Journal of Symbolic Computation (in press)
21. Denniston, R.H.F.: Non-existence of a certain projective plane. Journal of the Australian Mathematical Society **10**(1-2), 214–218 (1969)
22. Ganesh, V., O'Donnell, C.W., Soos, M., Devadas, S., Rinard, M.C., Solar-Lezama, A.: Lynx: A programmatic SAT solver for the RNA-folding problem. In: A. Cimatti, R. Sebastiani (eds.) International Conference on Theory and Applications of Satisfiability Testing, pp. 143–156. Springer (2012)
23. The GAP Group: GAP – Groups, Algorithms, and Programming, Version 4.10.2 (2019). URL `https://www.gap-system.org`
24. Hall Jr., M.: Configurations in a plane of order ten. Annals of Discrete Mathematics **6**, 157–174 (1980)
25. Heule, M.J.H.: Cube-and-conquer tutorial (2018). URL `https://github.com/marijnheule/CnC`
26. Heule, M.J.H.: Schur number five. In: S.A. McIlraith, K.Q. Weinberger (eds.) Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), pp. 6598–6606. AAAI Press (2018)
27. Heule, M.J.H., Kauers, M., Seidl, M.: New ways to multiply 3×3-matrices. arXiv 1905.10192 (2019)
28. Heule, M.J.H., Kullmann, O., Marek, V.W.: Solving and verifying the Boolean Pythagorean triples problem via cube-and-conquer. In: N. Creignou, D. Le Berre (eds.) International Conference on Theory and Applications of Satisfiability Testing, pp. 228–245. Springer (2016)
29. Heule, M.J.H., Kullmann, O., Marek, V.W.: Solving very hard problems: Cube-and-conquer, a hybrid SAT solving method. In: C. Sierra (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pp. 4864–4868 (2017)
30. Heule, M.J.H., Kullmann, O., Wieringa, S., Biere, A.: Cube and conquer: Guiding CDCL SAT solvers by lookaheads. In: K. Eder, J. Lourenço, O. Shehory (eds.) Haifa Verification Conference, pp. 50–65. Springer (2011)
31. Kåhrström, J.: On projective planes. Technical report (2002)
32. Kaufmann, D., Biere, A., Kauers, M.: Verifying large multipliers by combining SAT and computer algebra. In: Proceedings of Formal Methods in Computer-Aided Design (to appear)
33. Keller, C.: SMTCoq: Mixing automatic and interactive proof technologies. In: G. Hanna, D.A. Reid, M. de Villiers (eds.) Proof Technology in Mathematics Research and Teaching, pp. 73–90. Springer International Publishing, Cham (2019)
34. Konev, B., Lisitsa, A.: Computer-aided proof of Erdős discrepancy properties. Artificial Intelligence **224**, 103–118 (2015)
35. Kouril, M., Paul, J.L.: The van der Waerden number $W(2,6)$ is 1132. Experimental Mathematics **17**(1), 53–61 (2008)
36. Kullmann, O.: Green-Tao numbers and SAT. In: O. Strichman, S. Szeider (eds.) International Conference on Theory and Applications of Satisfiability Testing, pp. 352–362. Springer (2010)
37. Lam, C.W.H.: The search for a finite projective plane of order 10. American Mathematical Monthly **98**(4), 305–318 (1991)
38. Lam, C.W.H., Thiel, L., Swiercz, S.: 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)
39. Lam, C.W.H., Thiel, L., Swiercz, S.: The non-existence of finite projective planes of order 10. Canad. J. Math **41**(6), 1117–1123 (1989)
40. Lam, C.W.H., Thiel, L., Swiercz, S., McKay, J.: The nonexistence of ovals in a projective plane of order 10. Discrete Mathematics **45**(2-3), 319–321 (1983)
41. Liang, J.H., Ganesh, V., Poupart, P., Czarnecki, K.: Learning rate based branching heuristic for SAT solvers. In: N. Creignou, D. Le Berre (eds.) Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5–8, 2016, Proceedings, pp. 123–140 (2016). URL `https://ece.uwaterloo.ca/maplesat/`
42. MacWilliams, F.J., Sloane, N.J.A., Thompson, J.G.: On the existence of a projective plane of order 10. Journal of Combinatorial Theory, Series A **14**(1), 66–78 (1973)
43. Magaud, N., Narboux, J., Schreck, P.: Formalizing projective plane geometry in Coq. In: T. Sturm, C. Zengler (eds.) International Workshop on Automated Deduction in Geometry, pp. 141–162. Springer (2008)
44. McKay, B.D., Piperno, A.: Practical graph isomorphism, II. Journal of Symbolic Computation **60**, 94–112 (2014)
45. Nadel, A., Ryvchin, V.: Efficient SAT solving under assumptions. In: A. Cimatti, R. Sebastiani (eds.) International Conference on Theory and Applications of Satisfiability Testing, pp. 242–255. Springer (2012)
46. Perrott, X.: Existence of projective planes. arXiv 1603.05333 (2016)

47. Roy, D.J.: Proving $w_{15} = 0$ in a hypothetical projective plane of order 10. Course Project for CSI 5165, University of Ottawa (2005)
48. Roy, D.J.: Confirmation of the non-existence of a projective plane of order 10. Master's thesis, Carleton University (2011)
49. Sinz, C.: Towards an optimal CNF encoding of boolean cardinality constraints. In: P. van Beek (ed.) International conference on principles and practice of constraint programming, pp. 827–831. Springer (2005)
50. Wetzler, N., Heule, M.J.H., Hunt, W.A.: DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In: C. Sinz, U. Egly (eds.) International Conference on Theory and Applications of Satisfiability Testing, pp. 422–429. Springer (2014)
51. Wolfram Research, Inc.: Mathematica, Version 12.0. Champaign, IL, 2019
52. Zulkoski, E., Bright, C., Heinle, A., Kotsireas, I., Czarnecki, K., Ganesh, V.: Combining SAT solvers with computer algebra systems to verify combinatorial conjectures. Journal of Automated Reasoning **58**(3), 313–339 (2017)
53. Zulkoski, E., Ganesh, V., Czarnecki, K.: MathCheck: A math assistant via a combination of computer algebra systems and SAT solvers. In: A.P. Felty, A. Middeldorp (eds.) International Conference on Automated Deduction, pp. 607–622. Springer, Cham (2015)