

Myrvold's Results on Orthogonal Triples of 10×10 Latin Squares: A SAT Investigation

Curtis Bright^a Amadou Keita^b Brett Stevens^c

Submitted: Mar 13, 2025; Accepted: Jan 21, 2026; Published: Feb 13, 2026

© The authors. Released under the CC BY license (International 4.0).

Abstract

Ever since E. T. Parker constructed an orthogonal pair of 10×10 Latin squares in 1959, an orthogonal triple of 10×10 Latin squares has been one of the most sought-after combinatorial designs. Despite extensive work, the existence of such an orthogonal triple remains an open problem, though some negative results are known. In 1999, W. Myrvold derived some highly restrictive constraints in the special case in which one of the Latin squares in the triple contains a 4×4 Latin subsquare. In particular, Myrvold showed there were twenty-eight possible cases for an orthogonal pair in such a triple, twenty of which were removed from consideration. We implement a computational approach that quickly verifies all of Myrvold's nonexistence results and in the remaining eight cases finds explicit examples of orthogonal pairs—thus explaining for the first time why Myrvold's approach left eight cases unsolved. As a consequence, the eight remaining cases cannot be removed by a strategy of focusing on the existence of an orthogonal pair; the third square in the triple must necessarily be considered as well.

Our approach uses a Boolean satisfiability (SAT) solver to derive the nonexistence of twenty of the orthogonal pair types and find explicit examples of orthogonal pairs in the eight remaining cases. To reduce the existence problem into Boolean logic we use a duality between the concepts of *transversal representation* and *orthogonal pair* and we provide a formulation of this duality in terms of a composition operation on Latin squares. Using our SAT encoding, we find transversal representations (and equivalently orthogonal pairs) in the remaining eight cases in under two hours of computing on a large computing cluster.

Mathematics Subject Classifications: 05B15, 68R07

1 Introduction

A *Latin square* of order n is an $n \times n$ array filled with n distinct symbols, usually taken to be $\{0, 1, \dots, n - 1\}$, such that each symbol appears exactly once in each row and

^aSchool of Computer Science, University of Waterloo, Canada (cbright@uwaterloo.ca).

^bDepartment of Mathematics and Statistics, University of Windsor, Canada (keitaa@uwindsor.ca).

^cSchool of Mathematics and Statistics, Carleton University, Canada (brett@math.carleton.ca).

exactly once in each column. A *transversal* of a Latin square of order n consists of n cells of the square chosen so that there is exactly one cell from each row, exactly one cell from each column, and exactly n distinct symbols all together. There are many ways of representing a transversal, but we follow Myrvold [33] and represent a transversal by listing the symbols in the transversal in each column from left to right. For example, the highlighted transversal in $\begin{bmatrix} 0 & \textcolor{red}{1} & 2 \\ \textcolor{red}{2} & 0 & \textcolor{red}{1} \\ 1 & 2 & 0 \end{bmatrix}$ is represented by the row vector $[2, 1, 0]$. We call this row vector the transversal's *row representation*.

Two Latin squares A and B of order n are said to be *orthogonal* when all n^2 possible symbol pairs occur when the two squares are superimposed over each other. This happens exactly when the n cell positions of the same symbol in A form a transversal in B (regardless of the symbol chosen), thereby decomposing B into n non-overlapping transversals. A set of Latin squares that are pairwise orthogonal to each other are known as *mutually orthogonal Latin squares* (MOLS) and a set of k MOLS of order n are known as a k MOLS(n). For each order n , let $N(n)$ denote the largest value of k for which a k MOLS(n) exists. Determining values of $N(n)$ has a long history [1, Ch. III] and has been of intense interest to mathematicians ever since Euler conjectured in 1782 that $N(n) = 1$ for $n \equiv 2 \pmod{4}$. It is easily seen that $N(2) = 1$, and Tarry showed in 1900 that $N(6) = 1$ [39]. However, in 1959, Euler's conjecture was shown to be false by the discovery of a 2 MOLS(22) [6] and a 2 MOLS(10) [35]. In fact, in 1960 it was shown that $N(n) \geq 2$ for all $n > 6$ [7]. It is also known that $N(n) = n - 1$ if and only if a projective plane of order n exists. Projective planes exist for all prime powers, so the first order for which the value of $N(n)$ is uncertain is $n = 10$. It is unknown if $N(10) \geq 3$, and determining the value of $N(10)$ is one of the most prominent unsolved problems concerning MOLS. In particular, finding a 3 MOLS(10) or proving its nonexistence is a longstanding open problem in combinatorial design theory.

Although it is not known if a 3 MOLS(10) exists or not, there are several special results known about this case. Mann [28] proved that a 10×10 Latin square with a 5×5 Latin subsquare cannot belong to an orthogonal pair, let alone an orthogonal triple. Parker [36] proved that two orthogonal 10×10 Latin squares with orthogonal 3×3 Latin subsquares cannot be part of an orthogonal triple. Myrvold [33] considered a 10×10 Latin square L with a 4×4 Latin subsquare. She showed that it *is* possible for L to be part of an orthogonal pair, and further considered if L can be part of an orthogonal triple. Myrvold showed that orthogonal mates of L can be classified into seven possible mate pattern types. Furthermore, if L is in an orthogonal triple the other two squares in the triple can be classified into twenty-eight mate pattern type pairs. Myrvold ruled out the existence of twenty of the twenty-eight mate pattern type pairs, and this required only the consideration of constraints arising from two of the three putative squares. Her work left open the remaining eight cases:

The most obvious next step in extending the current work is to eliminate the remaining eight cases from consideration. [33]

We provide a reason why Myrvold's method was unable to rule out these eight cases, and show any argument ruling out these cases must necessarily be more involved—because orthogonal pairs in the remaining eight cases exist (though it is unclear if orthogonal

triples in the remaining eight cases exist). Thus, any argument ruling out the remaining eight cases must necessarily involve the triple as a whole, not only two of the three squares. We give more background on Latin squares and the formulation of Myrvold’s twenty-eight cases in Section 2.

Our approach uses a satisfiability (SAT) solver to explicitly construct a 2 MOLS(10) in each of the eight cases that Myrvold left open. Additionally, in under a second of compute time the SAT solver shows the nonexistence of a 2 MOLS(10) in the twenty cases solved by Myrvold. To use a SAT solver, it is necessary to reduce the problem of searching for the object in question to the problem of searching for a satisfying assignment to a formula in Boolean logic representing Myrvold’s framework and cases.

We reduce the problem of finding a 2 MOLS(10) in each of Myrvold’s twenty-eight cases to SAT—see Section 4 for a description of our encoding. We develop a SAT encoding of orthogonality that relies on an equivalence between the orthogonality of Latin squares and what Myrvold calls a “transversal representation” Latin square [33]. Myrvold uses this equivalence for “*designing computer programs for exploring squares and their mates*”. We provide a precise duality relating these two concepts via a composition operation on Latin squares and a generalization of Latin squares where only the columns (and not necessarily the rows) contain all n symbols (see Section 3). This transversal representation encoding allowed finding a 2 MOLS(10) for all of Myrvold’s previously unsolved cases in a reasonable amount of computation, even for a single desktop computer. By exploiting the parallelization ability of a large computing cluster, we were able to solve the hardest of the eight cases in less than two hours of real time—see Section 5 for more details.

2 Background

We define the notion of transversal representation and relate it to the orthogonality of Latin squares in Section 2.1. Next, we explain the transversal representation types classified by Myrvold [33] in Section 2.2, and give a brief description of satisfiability solving in Section 2.3. Lastly, we give a summary of related work in Section 2.4, with a focus on work applying automated reasoning tools to solve problems related to Latin squares.

2.1 Transversals and Orthogonality

It is well-known that a Latin square of order n has an orthogonal mate if and only if it can be decomposed into n disjoint transversals [41]. From the n disjoint transversals, a new Latin square can be formed by writing each transversal in its row representation and stacking the rows together. We call such a square a *transversal representation* of the original square. An example of a 4×4 Latin square D with four disjoint transversals and the associated transversal representation D' is provided in Figure 1. The pair (D, D') is known as a *transversal representation pair* or *TRP*.

Although we are primarily interested in Latin squares, in the course of our investigations, we found that it was helpful to consider the more general case of column-Latin squares. A *column-Latin* square of order n is an $n \times n$ array filled with n distinct symbols and in

$$D = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 0 & 3 \\ \hline 0 & 3 & 1 & 2 \\ \hline 2 & 1 & 3 & 0 \\ \hline 3 & 0 & 2 & 1 \\ \hline \end{array} \quad D' = \begin{array}{|c|c|c|c|} \hline 0 & 1 & 2 & 3 \\ \hline 1 & 0 & 3 & 2 \\ \hline 2 & 3 & 0 & 1 \\ \hline 3 & 2 & 1 & 0 \\ \hline \end{array}$$

Figure 1: A transversal representation pair of Latin squares of order four. Each transversal of D is highlighted in a different colour, and the row representations of the transversals are given in D' .

which each column contains distinct symbols (and is thus a permutation), but the rows are not required to contain distinct symbols. *Row-Latin* squares are defined similarly: the rows of the square must contain distinct entries, but the columns might not [24]. It follows immediately that an $n \times n$ array filled with n distinct symbols is a Latin square if and only if it is both row-Latin and column-Latin. For our purposes, the usefulness of column-Latin squares stems from the fact that two column-Latin squares can be composed in a sensible way to form a third column-Latin square which preserves structure related to orthogonality (see Section 3). Thus, we state most of our results in terms of column-Latin squares.

The concept of orthogonality of Latin squares translates directly to column-Latin squares. However, the concept of transversal needs some modification. A generalized transversal of a column-Latin square of order n must still be a selection of n entries from each row and column, but the entries may not all be distinct. Figure 2 shows an example of this generalization; note the generalized transversals highlighted in D_1 contain duplicate entries and therefore are not traditional transversals. However, the row representation construction can still be used to construct the column-Latin square D'_1 and we refer to the pair (D_1, D'_1) as a transversal representation pair of column-Latin squares.

$$D_1 = \begin{array}{|c|c|c|c|} \hline 0 & 1 & 3 & 2 \\ \hline 1 & 3 & 2 & 0 \\ \hline 3 & 2 & 1 & 1 \\ \hline 2 & 0 & 0 & 3 \\ \hline \end{array} \quad D'_1 = \begin{array}{|c|c|c|c|} \hline 0 & 0 & 2 & 1 \\ \hline 1 & 1 & 1 & 3 \\ \hline 2 & 2 & 3 & 0 \\ \hline 3 & 3 & 0 & 2 \\ \hline \end{array}$$

Figure 2: A transversal representation pair of 4×4 column-Latin squares. Note that the highlighted entries of D_1 are *not* transversals, but their row representations when placed in a 4×4 array do form a column-Latin square.

We now give purely logical definitions of orthogonal pair and transversal representation and state the definitions in a way that highlights the similarity between the concepts. Suppose $[a_0, \dots, a_{n-1}]$ is a row representing a generalized transversal of a column-Latin square B . This means if i is a row index, j and j' are two distinct column indices, and $B[i, j] = a_j$, then $B[i, j'] \neq a_{j'}$ (otherwise, both the j th and j' th entries of the generalized transversal are in row i , which is not allowed in any transversal, generalized or not).

Equivalently, if both $B[i, j] = a_j$ and $B[i, j'] = a_{j'}$, then the only possibility is that $j = j'$. This motivates the following definition.

Definition 1. Let A and B be order n column-Latin squares. Row i of A represents a transversal of B when $A[i, j] = B[i', j]$ and $A[i, j'] = B[i', j']$ imply $j = j'$. The square A is said to be a transversal representation of B when each row of A represents a transversal of B , i.e., for all $0 \leq i, i', j, j' < n$,

$$A[i, j] = B[i', j] \text{ and } A[i, j'] = B[i', j'] \text{ imply } j = j'.$$

Because Definition 1 is symmetric in A and B , A is a transversal representation of B if and only if B is a transversal representation of A . As before, we say (A, B) is a transversal representation pair or TRP.

On the other hand, if two column-Latin squares A and B are orthogonal this means that if (i, j) and (i', j') are two distinct (row, column) pairs then $(A[i, j], B[i, j]) \neq (A[i', j'], B[i', j'])$. Equivalently, it means that if both $A[i, j] = A[i', j']$ and $B[i, j] = B[i', j']$, the only possibility is that $(i, j) = (i', j')$. This motivates the following definition.

Definition 2. Let A and B be order n column-Latin squares. A is said to be orthogonal to B if for all $0 \leq i, i', j, j' < n$,

$$A[i, j] = A[i', j'] \text{ and } B[i, j] = B[i', j'] \text{ imply } j = j'.$$

Note that the equality of j and j' in Definition 2 also implies the equality of i and i' because A and B are column-Latin squares. The consequent in Definition 2 thus could equivalently have been written as the more typical $(i, j) = (i', j')$, but we use the simpler $j = j'$ in order to highlight the striking similarity between Definitions 1 and 2.

2.2 Transversal Representation Types

We now review Myrvold's results [33] on the possible transversal representation types of a 10×10 Latin square L containing a 4×4 Latin subsquare Ω . Without loss of generality, we assume the subsquare appears in the bottom-right of L , i.e., in the rows and columns labeled 6 to 9. We also assume L consists of the symbols from the set $\{0, 1, 2, \dots, 9\}$ and Ω consists of symbols from the set $\{0, 1, 2, 3\}$. We partition the other regions of L into Δ (lower-left), Γ (upper-right), and Σ (upper-left) as shown in Figure 3. Since the subsquare Ω is a Latin square containing symbols from the set $\{0, 1, 2, 3\}$, the rectangles Δ and Γ must take symbols only from the set $\{4, 5, 6, \dots, 9\}$ and each row and column of Σ must contain exactly $6 - 4 = 2$ symbols from the set $\{4, 5, 6, \dots, 9\}$.

Suppose the cells with symbols in $\{0, 1, 2, 3\}$ are coloured white. A transversal of L can be of five possible forms depending on how many white cells it takes from the Latin subsquare Ω . A transversal containing i white cells from Ω (i.e., in its last four columns) is said to be of form p_i (see Figure 3). Since any transversal will contain exactly four white cells in total, it must contain $4 - i$ white cells in its first six columns. Consider the entries of p_i that were chosen from the first six rows of L (i.e., Σ or Γ). We have $4 - i$

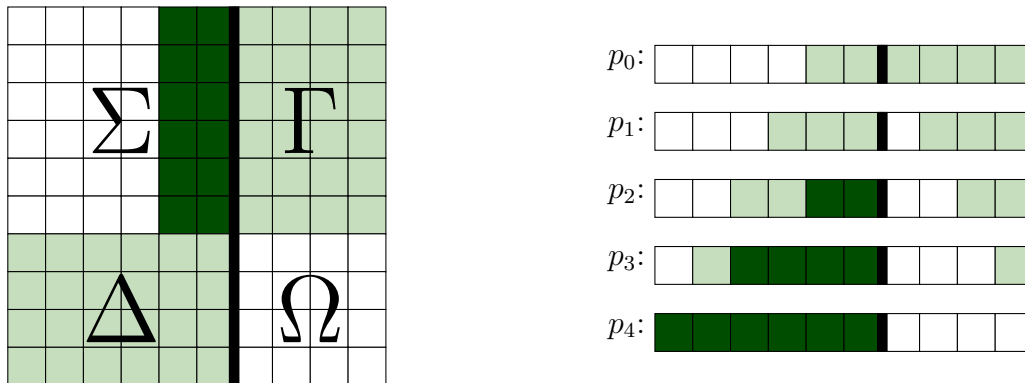


Figure 3: The Latin square L (left) and its possible transversal types (right). White cells represent symbols in $\{0, 1, 2, 3\}$, light cells represent symbols in the rectangles Δ and Γ , and dark cells represent the symbols $\{4, 5, \dots, 9\}$ in Σ . The cells of Σ are not shown in absolute positions; in actuality, each row and column of Σ has exactly two dark cells. Similarly, the transversal types are shown up to a permutation of the first six entries and the last four entries.

white entries (all from Σ) and $4 - i$ entries from the last four columns of L (i.e., from Γ), so there are $6 - 2(4 - i) = 2i - 2$ remaining entries. The only possibilities for these are the nonwhite entries of Σ , and we colour these entries dark. This results in the following lemma.

Lemma 3 ([33, Lemma 3.1]). *A transversal of type p_i contains exactly $2i - 2$ dark entries.*

A simple corollary of Lemma 3 is that p_0 is not a possible type, as it would have to contain -2 dark entries.

Let n_i be the number of transversals of type p_i in a transversal representation of L . Simple counting arguments give that the values $\{n_1, n_2, n_3, n_4\}$ satisfy the following Diophantine linear system.

$$\begin{array}{ll} n_i \geq 0 & \text{nonnegativity of the counts,} \\ n_1 + n_2 + n_3 + n_4 = 10 & \text{ten total transversals,} \\ n_1 + 2n_2 + 3n_3 + 4n_4 = 16 & \text{sixteen total symbols in } \Omega. \end{array}$$

There are seven possible solutions to this linear system and correspondingly seven transversal representation types of L . These types are denoted R, S, T, U, V, W, and X by Myrvold. Table 1 gives the transversal type counts of each case.

Up to ordering, there are $\binom{7}{2} = 21$ ways of choosing a pair with two different types, and 7 ways of choosing a pair with matching types, for a total of 28 possible transversal

Type	n_1	n_2	n_3	n_4
R	8	0	0	2
S	7	0	3	0
T	7	1	1	1
U	6	2	2	0
V	6	3	0	1
W	5	4	1	0
X	4	6	0	0

Table 1: A summary of Myrvold’s seven possible transversal types of L .

representation pair combinations. Under the assumption that L is part of an orthogonal triple, Myrvold [33, Thm 4.4] showed that the only possible pair types that could potentially be transversal representations of L simultaneously are (S, X), (U, U), (U, W), (U, X), (V, X), (W, W), (W, X), and (X, X).

2.3 Satisfiability Solving

In this section, we provide some basic preliminaries on Boolean logic and satisfiability (SAT) solving. A *SAT solver* is a program that can determine if a Boolean logic formula can be *satisfied*—that is, if there is a truth assignment under which the formula becomes true. In practice, the formulas provided to SAT solvers must be written in conjunctive normal form (CNF). Formulas in CNF only contain the Boolean connective operators \wedge (and), \vee (or), and \neg (not). These operators have meanings similar to those in everyday English: the formula $x \wedge y$ is true if and only if both x and y are true; the formula $x \vee y$ is true if and only if x or y (or both) are true; and the formula $\neg x$ is true if and only if x is false.

A *literal* is a Boolean variable or its negation, i.e., a formula of the form x or $\neg x$ where x is a Boolean variable. A *clause* is a disjunction of literals, i.e., a formula of the form $l_1 \vee \cdots \vee l_k$ where l_1, \dots, l_k are literals. Finally, a formula is in *conjunctive normal form* when it is a conjunction of clauses, i.e., a formula of the form $c_1 \wedge \cdots \wedge c_k$ where c_1, \dots, c_k are clauses.

When A is a conjunction of literals and B is a disjunction of literals, we use the notation $A \rightarrow B$ as shorthand for $\neg A \vee B$. By basic logic equivalences, the formula $(\neg \bigwedge_i a_i) \vee \bigvee_i b_i$ is equivalent to $\bigvee_i \neg a_i \vee \bigvee_i b_i$, which (after applying the simplification $\neg \neg x \equiv x$ to any doubly negated literal) is a clause. Thus, we consider the notation $A \rightarrow B$ to be shorthand for a clause when B is a clause and A is a conjunction of literals.

Although there is no guarantee that SAT solvers can solve the SAT problem in a feasible amount of time, modern SAT solvers are highly effective at solving many kinds of problems arising in practice [40], including mathematical problems such as the Boolean Pythagorean triples problem [18] and Lam’s problem of proving the nonexistence of a projective plane of order ten [10]. Although these problems at first seem unconnected to logic, they can be reduced to SAT due to the versatility of Boolean logic [11]. Another

advantage of using a SAT solver is that they offer a higher amount of confidence in a computational search. It is typically less error-prone to write a SAT encoding than it is to write optimized search code, and moreover, the SAT solver itself does not need to be trusted because it produces a proof certificate which can be later checked by simpler and independently-written software. This is particularly relevant when purporting to demonstrate the *nonexistence* of a mathematical object, such as in Lam’s problem of proving projective planes of order ten do not exist [22].

Lam’s problem was resolved in 1989 using a massive computer search by Lam, Thiel, and Swiercz [23]. In 2011, the search was independently performed by Roy [37]. Although these works are amazing achievements, they both crucially rely on highly optimized computer code that is essentially impossible to verify for correctness, and the programmers of the search code were upfront that the code may contain bugs. Indeed, discrepancies in the results of these searches were later found: a SAT-based search of Bright et al. [10] found inconsistencies in the intermediate counts provided by Lam et al., implying a small number of missing subcases in the proof. Also, the independent confirmation of Roy [37] was based in part on the nonexistence of a partial projective plane later determined to actually exist [9]. There is no formal proof that Bright et al.’s SAT-based resolution of Lam’s problem is without error—because the SAT encoding itself is unverified—but it does have the advantage that no *search code* has to be trusted.

2.4 Related Work

Extensive searches for a 3 MOLS(10) have been performed, and some important cases have been ruled out. For example, it is known that any such triple must only contain Latin squares with trivial symmetry groups [30]. Independent computer searches [10, 23, 37] have revealed that there is no projective plane of order ten, and because a projective plane of order n is equivalent to a $(n - 1)$ MOLS(n) [8, 32], these searches imply that no 9 MOLS(10)s exist or equivalently that $N(10) < 9$. Together with a result of Bruck [13], this implies that $N(10) \leq 6$ which is currently the best upper bound known on $N(10)$.

Egan and Wanless [16] enumerate MOLS of small orders, providing counts of orthogonal mates and classifications up to various equivalence notions for orders $n \leq 9$. They also present a set of three Latin squares L_1, L_2, L_3 of order 10 that is the closest known to forming a complete set of MOLS: L_1 is orthogonal to both L_2 and L_3 , and 91 out of the 100 symbol pairs are different when L_2 and L_3 are superimposed. They also showed that L_2 and L_3 have seven common disjoint transversals.

Numerous studies have leveraged SAT solving, integer programming, and constraint programming in order to search for Latin squares of various forms. Appa, Magos, and Mourtos [2, 3] integrated integer programming and constraint programming to tackle the problem of searching for mutually orthogonal Latin squares. Their comparative study against traditional constraint and integer programming algorithms revealed the effectiveness of combining integer and constraint programming in searching for 2 MOLS(n) for $n \leq 12$ and 3 MOLS(n) for $n \leq 9$. Rubin et al. [38] formulated a symmetry breaking method and also provided an alternative constraint programming encoding based on a theorem of Mann [27] which performed much better in their search for pairs of orthogonal

Latin squares. The SAT encoding that we use in our work can be viewed as a reformulation of their constraint programming encoding into Boolean satisfiability.

Ma and Zhang [26] use a general-purpose model searching program to find MOLS. They show a k MOLS(n) exists if and only if there exists a Latin square of order n which has $k - 1$ transversal matrices T_1, \dots, T_{k-1} with any two transversal matrices T_i and T_j ($i \neq j$) being transversal matrices of each other [26, Prop 1]. As a result, instead of searching for k MOLS(n), they searched for k Latin squares L, T_1, \dots, T_{k-1} that are mutual transversal matrices of each other. The initial Latin square L was defined as a function $f: \mathcal{R} \times \mathcal{C} \rightarrow \mathcal{D}$ on row indices \mathcal{R} , column indices \mathcal{C} , and symbol set \mathcal{D} . Similarly, the i th transversal matrix T_i ($1 \leq i \leq k - 1$) was defined as a function $f_i: \mathcal{D}_i \times \mathcal{C} \rightarrow \mathcal{R}$, where \mathcal{D}_i is the symbol set of L_i , the Latin square represented by the transversal matrix T_i . The formulae they used for encoding a k MOLS(n) then consist of three types:

1. Formulae to specify that f and f_i are Latin squares:

$$\begin{aligned} f(x_1, y) = f(x_2, y) &\rightarrow x_1 = x_2, & f(x, y_1) = f(x, y_2) &\rightarrow y_1 = y_2, \\ f_i(t_1, y) = f_i(t_2, y) &\rightarrow t_1 = t_2, & f_i(t, y_1) = f_i(t, y_2) &\rightarrow y_1 = y_2. \end{aligned}$$

2. Formulae to specify that f_i is a transversal matrix of f :

$$f(f_i(t, y_1), y_1) = f(f_i(t, y_2), y_2) \rightarrow y_1 = y_2.$$

3. Formulae to ensure that L_i and L_j are orthogonal by stating that T_i and T_j are a transversal representation pair:

$$(f_i(t_1, y_1) = f_j(t_2, y_1) \wedge f_i(t_1, y_2) = f_j(t_2, y_2)) \rightarrow y_1 = y_2.$$

Our encoding of a transversal representation pair uses formulae that are similar to their first two types, though our encoding is purely represented as a Boolean satisfiability problem which does not natively support expressions like $f(f_i(t, y_1), y_1)$. Constraints of type 3 could theoretically be replaced by constraints like those of type 2 (e.g., $f_i(f_j(t, y_1), y_1) = f_i(f_j(t, y_2), y_2) \rightarrow y_1 = y_2$), though it is unclear if this encoding variant was tried by Ma and Zhang. Our experience suggests that (at least for a SAT solver) it is preferable to encode a transversal representation pair using constraints of type 2 instead of constraints of type 3.

A Latin square that is orthogonal to its transpose is known as *self-orthogonal* and if it is additionally orthogonal to its anti-diagonal transpose it is known as *doubly self-orthogonal*. For orders $n \equiv 2 \pmod{4}$, the existence of doubly self-orthogonal Latin squares is unknown for $n > 10$. In 2011, Lu et al. [25] proved the nonexistence of a doubly self-orthogonal Latin square of order ten. They encoded the existence of a doubly self-orthogonal Latin square of order ten as a SAT problem and proved the nonexistence by showing the resulting SAT instance was unsatisfiable. To describe their encoding, let A be a self-orthogonal Latin square of order n , let A^T denote the transpose of A , and let A^* denote the transpose across the anti-diagonal of A , i.e., $A^T[x, y] = A[y, x]$ and $A^*[x, y] = A[n - 1 - y, n - 1 - x]$

where $0 \leq x, y < n$. In addition to the properties of a Latin square, they generated the constraints

$$\begin{aligned} & (A[x_1, y_1] = A[x_2, y_2] \wedge A[y_1, x_1] = A[y_2, x_2]) \\ & \rightarrow (x_1 = x_2 \wedge y_1 = y_2), \quad \text{i.e., orthogonality of } A \text{ and } A^T, \text{ and} \\ & (A[x_1, y_1] = A[x_2, y_2] \wedge A[n-1-y_1, n-1-x_1] = A[n-1-y_2, n-1-x_2]) \\ & \rightarrow (x_1 = x_2 \wedge y_1 = y_2), \quad \text{i.e., orthogonality of } A \text{ and } A^*. \end{aligned}$$

A *Costas array* of order n is an $n \times n$ grid with n dots and $n^2 - n$ empty cells, with one dot in every row and column, and with no two dots sharing the same relative horizontal, vertical, or diagonal displacement. A *Costas Latin square* is a Latin square in which the cells for each symbol form a Costas array; see Figure 4 for an example. Jin et al. [20]

0	1	2	3
1	0	3	2
3	2	1	0
2	3	0	1

Figure 4: An example 4×4 Costas Latin square.

used SAT solvers to search for Costas Latin squares. They established new existence and nonexistence results for various types of Costas Latin squares of even orders $n \leq 10$ including orthogonal pairs of Costas Latin squares. In their encoding, they define from the square A a new square TA by the rule $A[i, j] = k \rightarrow TA[k, j] = i$. This makes TA the $(3, 2, 1)$ -*parastrophe* of A (the Latin square obtained by swapping the meaning of rows and symbols), though they refer to TA as a transversal matrix. To encode orthogonality of (A, B) , they impose the constraints

$$x \neq y \rightarrow (TA[u, x] \neq TB[v, x] \vee TA[u, y] \neq TB[v, y]) \quad \text{for } 0 \leq x, y, u, v < n.$$

The $(3, 2, 1)$ -parastrophe is also called the *column inverse* since it can also be obtained by treating each column as a permutation of $[0, \dots, n-1]$ and replacing each column with its inverse [21]. In the rest of this paper, we will use the notation A^{-1} for the column inverse of A (see Section 3.1).

A Latin square of order n is *idempotent* when its diagonal consists of the entries $0, 1, \dots, n-1$ in order, and is *symmetric* when it is equal to its own transpose. A *golf design* of order n is a collection of $n-2$ idempotent symmetric Latin squares of order n that are mutually disjoint, meaning that any two Latin squares in the collection share no common symbols in any cell (except for the cells along their diagonals). Two golf designs are *orthogonal* if every Latin square in one design has an orthogonal mate in the other design.

Huang et al. [19] investigated the existence of orthogonal golf designs via constraint programming and satisfiability testing. They reformulated the orthogonal mate finding

problem as a transversal finding problem. They constructed the transversal matrix T of a Latin square L with the constraints

$$(y_1 = y_2 \vee L[T[x, y_1], y_1] \neq L[T[x, y_2], y_2]) \quad \text{for } 0 \leq x, y_1, y_2 < n,$$

and additionally used constraints specifying that T is a Latin square.

Latin squares are known as *diagonal* if they feature distinct symbols along both the main and back diagonals. Zaikin and Kochemazov [42] constructed SAT encodings to discover pairs of orthogonal diagonal Latin squares of order ten and pseudotriples of orthogonal diagonal Latin squares. A *pseudotriple* refers to a set of three Latin squares that nearly form an orthogonal triple, but the orthogonality condition is only required to hold on a subset of the cells of the Latin squares. They discovered a triple of diagonal Latin squares of order ten for which the orthogonality condition holds across 73 cells (the same 73 cells in each Latin square in the triple).

An *extended self-orthogonal diagonal Latin square* is a diagonal Latin square that is orthogonal to a diagonal Latin square in its main class—the *main class* of a Latin square being the set of Latin squares produced by application of row permutations, column permutations, symbol permutations, or interchanging the roles of rows, columns, and symbols. Extended self-orthogonal diagonal Latin squares generalize the notion of self-orthogonal diagonal Latin squares, since the transpose of a Latin square is always a member of its main class (obtained by interchanging the roles of rows and columns). Zaikin, Vatutin, and Bright [43] use a SAT solver to enumerate all extended self-orthogonal diagonal Latin squares up to order ten and show that in order ten no such squares are part of an orthogonal triple. Their SAT encoding for orthogonality is based off of the one we present in this paper relying on a consequence of Mann’s theorem described in Section 3.1.

In a separate recently published paper [12], we use a SAT encoding for orthogonality based on the one described in Section 4.2 in order to enumerate 2 MOLS(10) whose incidence matrices have at least two nontrivial linear dependencies. This enumeration had been previously completed using custom-written search code of Delisle [14] and was motivated by work of Dukes and Howard [15] which classified the kinds of linear dependencies that could occur in the incidence matrix of a hypothetical set of 4 MOLS(10). Dukes and Howard also showed that the incidence matrix of a 4 MOLS(10) must have at least two nontrivial linear dependencies. Based on a later computational search of Gill and Wanless [17], it is now known that the incidence matrix of any pair of squares in a 3 MOLS(10) must only have trivial linear dependencies. Consequently, the rank of the linear code generated by any pair of squares in a 3 MOLS(10) must be exactly 37.

3 Composition and Duality

In this section, we describe a duality between the concepts of orthogonality and transversal representation. First, in Section 3.1 we define a composition operation on column-Latin squares. Then in Section 3.2 we use the composition operation to concisely characterize the duality.

3.1 Composition of Column-Latin Squares

A column-Latin square of order n can be represented by $(c_0, c_1, \dots, c_{n-1})$ where c_j is the permutation of $[0, \dots, n-1]$ formed by the j th column. For any two permutations f and g on the same set, the *composition* fg is another permutation where $(fg)(i) = f(g(i))$, i.e., applying g then f . The composition of two column-Latin squares $F = (f_0, \dots, f_{n-1})$ and $G = (g_0, \dots, g_{n-1})$ is defined as

$$FG = (f_0g_0, \dots, f_{n-1}g_{n-1}).$$

The (i, j) th entry of FG is then $f_jg_j(i) = F[G[i, j], j]$. The *column inverse* of a column-Latin square F , denoted F^{-1} , is the column-Latin square in which each column is the inverse permutation of the corresponding column of F .

Let e denote the identity column permutation with $e(i) = i$ for $0 \leq i < n$ and $E = (e, \dots, e)$ the column-Latin square of order n formed by n copies of e . The following two lemmas appear in Laywine and Mullen [24, pp. 98–99], except stated in terms of row-Latin squares instead of column-Latin squares.

Lemma 4. *Let C be a column-Latin square. Then (C, E) is an orthogonal pair if and only if C is a Latin square.*

Lemma 5. *If $\{C_1, C_2, \dots, C_m\}$ is a set of mutually orthogonal column-Latin squares, then for any column-Latin square G , the set $\{C_1G, C_2G, \dots, C_mG\}$ comprises a set of mutually orthogonal column-Latin squares.*

The next proposition provides criteria establishing a necessary and sufficient condition for the orthogonality of two column-Latin squares. In particular, the existence of a Latin square of a certain form guarantees the orthogonality of the two column-Latin squares. The biconditional statement in the proposition was proven by Mann [27] and also appears in Norton [34, Thm. 2] and Laywine–Mullin [24, Thm. 6.6], though we strengthen the proposition by showing that when the squares are Latin (not just column-Latin) the square providing the guarantee of orthogonality arises as a transversal representation of one of the original two squares.

Proposition 6. *Let C and F be column-Latin squares. Then (C, F) is an orthogonal pair if and only if there is a Latin square Z such that $ZC = F$. Moreover, if in addition, C is a Latin square, then (Z, F) is a TRP.*

Proof. Suppose Z is a Latin square and $ZC = F$ for column-Latin squares C and F . By Lemma 4, (Z, E) is an orthogonal pair. By Lemma 5, (ZC, EC) is an orthogonal pair. Since $ZC = F$ and $EC = C$, it follows that (F, C) is an orthogonal pair.

Conversely, suppose (C, F) is an orthogonal pair. Let $Z = FC^{-1}$ (i.e., $ZC = F$). Since (C, F) is an orthogonal pair, by Lemma 5, (Z, E) is an orthogonal pair (since $FC^{-1} = Z$ and $CC^{-1} = E$). By Lemma 4, Z is a Latin square.

We now show that if C is a Latin square and F is a column-Latin square such that (C, F) is an orthogonal pair, then (Z, F) , which is equal to (Z, ZC) , is a TRP. Suppose

that (Z, F) is not a TRP. Then there exist $i, i', j, j' \in \{0, 1, 2, \dots, n-1\}$ where $j \neq j'$ with

$$\begin{aligned} Z[i, j] &= ZC[i', j] = Z[C[i', j], j], \text{ and} \\ Z[i, j'] &= ZC[i', j'] = Z[C[i', j'], j']. \end{aligned}$$

Since Z is a Latin square, the symbols in each of its columns are distinct. Thus, considering the entries of column j of Z , we must have $C[i', j] = i$ and $C[i', j'] = i$, but $C[i', j] = C[i', j']$ is a contradiction because the rows of C (in particular, row i') are permutations, implying $j = j'$. Thus (Z, F) is a TRP. \square

3.2 Orthogonal Pair / Transversal Representation Duality

We now state a duality between orthogonality and transversal representations. This duality was already used by Myrvold [33, Thm 1.1], but we show how the duality can be concisely formulated in terms of the composition operation on column-Latin squares—a convenient viewpoint that we were unable to find in the literature. Roughly speaking, Lemmas 7 and 8 are the analogue of Lemmas 4 and 5 with “orthogonal pair” replaced by “transversal representation pair”.

Lemma 7. *Let C be a column-Latin square. Then (C, E) is a TRP if and only if C is a Latin square.*

Proof. Let C be a column-Latin square and (C, E) be a TRP. It is enough to show that rows of C are each an n -permutation. Assume, for a contradiction, that this is not the case. Then for some $0 \leq i, j, j', k < n$ with $j \neq j'$, $C[i, j] = k = C[i, j']$. Since E is a transversal representation of C , row i of C has its t -th symbol from column t of E . Therefore, the symbol k is on two different rows of E , which contradicts the definition of E . Therefore, rows of C are each an n -permutation, and consequently, C is a Latin square.

Conversely, suppose C is a Latin square. Since all symbols are distinct on each row of C and the same on each row of E , then each row of C takes symbols from distinct rows and columns of E and the t -th symbol on each row is from column t of E . Thus E is a transversal representation of C . It follows that (C, E) is a TRP. \square

Lemma 8. *Let $\{C_1, C_2, \dots, C_m\}$ be a set of mutual TRPs of column-Latin squares, then for any column-Latin square G , the set $\{GC_1, GC_2, \dots, GC_m\}$ comprises mutual TRPs.*

Proof. It is enough to prove this statement for a set of two column-Latin squares. The columns of GC_1 and GC_2 are compositions of two permutations, therefore GC_1 and GC_2 are column-Latin squares. Assume, for a contradiction, that this is not the case. Suppose there exist $i, i', j, j' \in \{0, 1, 2, \dots, n-1\}$ where $j \neq j'$ with

$$GC_1[i, j] = GC_2[i', j] \text{ and } GC_1[i, j'] = GC_2[i', j'].$$

Thus by equality of the symbols

$$G[C_1[i, j], j] = G[C_2[i', j], j] \text{ and } G[C_1[i, j'], j'] = G[C_2[i', j'], j'].$$

Since G is a column-Latin square, the uniqueness of symbols in its columns provides that

$$C_1[i, j] = C_2[i', j] \text{ and } C_1[i, j'] = C_2[i', j'].$$

Since (C_1, C_2) is a TRP, we have $j = j'$. This contradicts our assumption. Thus (GC_1, GC_2) is a TRP. Therefore, the set consists of mutual TRPs. \square

Proposition 9. *Let C and F be column-Latin squares. Then (C, F) is a TRP if and only if there is a Latin square Z such that $CZ = F$. Moreover, if C is a Latin square, then Z is orthogonal to F .*

Proof. Assume there exists a Latin square Z such that $CZ = F$. By Lemma 7, (Z, E) is a TRP. By Lemma 8, (C, F) , which is equal to (CE, CZ) , is a TRP.

Conversely, assume (C, F) is a TRP. Let $Z = C^{-1}F$. Since (C, F) is a TRP and $(C^{-1}C, C^{-1}F) = (E, Z)$, by Lemma 8, (E, Z) is a TRP. Thus (E, Z) is a TRP. We have that Z is a Latin square by Lemma 7.

Now we prove that if C is a Latin square, Z and F are orthogonal. Assume, for a contradiction, that (Z, F) (where $F = CZ$) is not an orthogonal pair, i.e., there exist $i, i', j, j' \in \{0, 1, 2, \dots, n-1\}$ with $j \neq j'$ for which

$$Z[i, j] = Z[i', j'] \text{ and } F[i, j] = F[i', j'].$$

The second equation implies $C[Z[i, j], j] = C[Z[i', j'], j']$ an equality between two symbols in rows j and j' of C , which, after using the first equation, yields $C[Z[i, j], j] = C[Z[i, j], j']$. Since C is a Latin square, its rows are permutations, which implies $j = j'$ and contradicts the assumption that $j \neq j'$. Therefore, (Z, F) must be an orthogonal pair. \square

The following result describes the equivalence between a set of mutually orthogonal column-Latin squares and a set of mutually TRPs. The correctness of our SAT encoding relies on this equivalence.

Theorem 10 (cf. [33]). *Let \mathcal{C} denote a set $\{C_1, \dots, C_r\}$ of r column-Latin squares of order n .*

(a) *If \mathcal{C} contains mutually orthogonal squares, then the set*

$$\{Z_1, \dots, Z_r : Z_1 = C_1, Z_t = C_1 C_t^{-1} \text{ for } 2 \leq t \leq r\}$$

contains mutual TRPs.

(b) *If \mathcal{C} consists of mutual TRPs, then the set*

$$\{Y_1, \dots, Y_r : Y_1 = C_1, Y_t = C_t^{-1} C_1 \text{ for } 2 \leq t \leq r\}$$

contains mutually orthogonal pairs.

Proof. For (a), suppose the set $\{C_i : 1 \leq i \leq r\}$ consists of mutually orthogonal column-Latin squares of order n . Construct a set of r squares $\{Z_i : 1 \leq i \leq r\}$ by letting $Z_1 = C_1$ and $Z_t = C_1 C_t^{-1}$ for $2 \leq t \leq r$. Proposition 6 gives that each Z_t , $2 \leq t \leq r$ is a Latin square; further it ensures that (Z_1, Z_t) is a TRP. Observe that $Z_t C_t C_s^{-1} = Z_s$ for $2 \leq t, s \leq r$ where $t \neq s$. Since both C_t and C_s^{-1} are column-Latin squares, their composition is a column-Latin square. Thus (Z_t, Z_s) for $2 \leq t, s \leq r$ where $t \neq s$, being a TRP also follows from Proposition 6.

For (b), suppose the set $\{C_i : 1 \leq i \leq r\}$ consists of column-Latin squares of order n such that any two squares form a TRP. Construct a set of r squares $\{Y_i : 1 \leq i \leq r\}$ by letting $Y_1 = C_1$ and $Y_t = C_t^{-1} C_1$ for $2 \leq t \leq r$. Proposition 9 gives that each Y_t , $2 \leq t \leq r$ is a Latin square; and that Y_1 and Y_t are orthogonal. Observe that $C_s^{-1} C_t Y_t = Y_s$ for $2 \leq t, s \leq r$ where $t \neq s$. Since both C_s^{-1} and C_t are column-Latin squares, their composition is a column-Latin square. Therefore, Y_t being orthogonal to Y_s for $2 \leq t, s \leq r$ where $t \neq s$ also follows from Proposition 9. \square

4 Encoding and Implementation

In this section we describe our encoding of the problem of constructing transversal representation pairs (TRPs) into a Boolean satisfiability problem and how we use our encoding to search for TRPs for each of Myrvold's 28 possible types described in Section 2.2. Recall that Myrvold's 28 types describe TRPs (P, Q) for which P and Q are each transversal representations of a Latin square L of order $n = 10$ containing a 4×4 Latin subsquare.

To reduce the existence of the $n \times n$ square P into Boolean logic, we use n^3 Boolean variables $P_{i,j,k}$ (for $0 \leq i, j, k < n$) with $P_{i,j,k}$ denoting the fact that the (i, j) th entry of P is k . Similarly, another n^3 Boolean variables $Q_{i,j,k}$ for $0 \leq i, j, k < n$ represent the entries of the square Q .

Once these variables have been defined, we need to specify constraints that P and Q are Latin squares (see Section 4.1), are a transversal representation pair (see Section 4.2), and conform to one of Myrvold's 28 types (see Section 4.3). Additionally, we ensure that the white entries in the last four columns of P and Q appear in a way that is consistent with a 4×4 Latin subsquare Ω being in a square L having mutual transversal representations P and Q (see Section 4.4). We also describe a method of symmetry breaking which reduces the size of the search space by adding additional constraints which hold without loss of generality (see Section 4.5). Finally, once we have found a collection of TRPs, we run a postprocessing step on them, ensuring that the TRPs are pairwise inequivalent and that they cannot be extended to a set of three mutual TRPs (see Section 4.6). Our encoding scripts are written in Python and are freely available at doi.org/10.5281/zenodo.18130631.

4.1 Latin Square Constraints

First, we need to describe constraints on the variables $P_{i,j,k}$ (meaning that $P[i, j] = k$) asserting that P is a Latin square. Direct methods for doing this from the definition of a Latin square are well known and widely used; e.g., see (10.1)–(10.4) in Zhang's survey [44].

The direct method asserts that every cell of P contains *at least one* symbol and *at most one* symbol, i.e.,

$$\bigvee_{0 \leq i < n} P_{p,q,i} \quad \text{and} \quad \bigwedge_{0 \leq i < j < n} (\neg P_{p,q,i} \vee \neg P_{p,q,j}) \quad \text{for all } 0 \leq p, q < n.$$

Additionally, every column of P contains n distinct symbols,

$$\bigvee_{0 \leq i < n} P_{i,q,r} \quad \text{and} \quad \bigwedge_{0 \leq i < j < n} (\neg P_{i,q,r} \vee \neg P_{j,q,r}) \quad \text{for all } 0 \leq q, r < n,$$

and similarly every row of P contains n distinct symbols,

$$\bigvee_{0 \leq i < n} P_{p,i,r} \quad \text{and} \quad \bigwedge_{0 \leq i < j < n} (\neg P_{p,i,r} \vee \neg P_{p,j,r}) \quad \text{for all } 0 \leq p, r < n.$$

This encoding uses what is known as the binomial or pairwise encoding of the *exactly one* predicate [29] and uses $3n^2 \binom{n}{2} + 1$ clauses in total. While this encoding gave good performance, in our experiments we got slightly better performance with the cardinality constraint encoding of Bailleux and Boufkhad [4]. Their encoding reduces a constraint like $x_1 + \cdots + x_n = r$ (where r is a fixed integer between 0 and n and we think of the Boolean x_i s as $\{0, 1\}$ variables) into conjunctive normal form. Using this encoding we specify that P is a Latin square with the cardinality constraints

$$\sum_{0 \leq i < n} P_{p,q,i} = 1, \quad \sum_{0 \leq i < n} P_{i,p,q} = 1, \quad \sum_{0 \leq i < n} P_{p,i,q} = 1 \quad \text{for all } 0 \leq p, q < n,$$

and a similar encoding can be used to specify that Q is also a Latin square.

4.2 Transversal Representation Constraints

The direct encoding that (P, Q) is a TRP using the contrapositive of Definition 1 would be

$$(P_{i,j,k} \wedge P_{i,j',k'} \wedge Q_{i',j,k}) \rightarrow \neg Q_{i',j',k'} \quad \text{for all } 0 \leq i, i', j, j', k, k' < n \text{ with } j < j'.$$

This is because if row i of P has its j th entry as k and its (j') th entry as k' , then in whatever row of Q which has its j th entry as k (one such row must exist since Q is a Latin square) that row *cannot* have its (j') th entry as k' , or that row wouldn't represent a transversal. However, this encoding uses $n^4 \binom{n}{2} = \Theta(n^6)$ clauses of length 4 which is not ideal in practice. Instead, our encoding that (P, Q) is a TRP will assert the existence of the Latin square $Z = P^{-1}Q$ and by Proposition 9 this implies that P and Q are a transversal representation pair.

As before, the entries of the square Z are encoded via n^3 new variables $Z_{i,j,k}$ (with $0 \leq i, j, k < n$) and Z is enforced to be a Latin square using the same encoding described in Section 4.1. Now we need to enforce the relationship $Q = PZ$, which means that the

(i, j) th entry of Q is equal to the (i', j) th entry of P , where $i' = Z[i, j]$. Letting k represent the (i, j) th entry of Q , this gives the constraints

$$(Z_{i,j,i'} \wedge P_{i',j,k}) \rightarrow Q_{i,j,k} \quad \text{for all } 0 \leq i, i', j, k < n.$$

Moreover, because $P = QZ^{-1}$ and $Z = P^{-1}Q$, we similarly derive the constraints

$$\begin{aligned} (Z_{i,j,i'} \wedge Q_{i,j,k}) &\rightarrow P_{i',j,k} \quad \text{for all } 0 \leq i, i', j, k < n, \\ (P_{i',j,k} \wedge Q_{i,j,k}) &\rightarrow Z_{i,j,i'} \quad \text{for all } 0 \leq i, i', j, k < n. \end{aligned}$$

These last two kinds of constraints are technically redundant, but we found that they tended to improve the performance of the solving in practice.

Thus, our encoding that (P, Q) is a TRP uses $3n^4$ clauses and the $3n^2$ cardinality constraints $\sum_i Z_{i,j,k} = \sum_i Z_{j,k,i} = \sum_i Z_{j,i,k} = 1$ for all $0 \leq j, k < n$. Altogether, this TRP encoding uses $\Theta(n^4)$ clauses of length at most 3, and in practice this is preferable to the $\Theta(n^6)$ clauses of length 4 used by the direct encoding.

A similar $\Theta(n^4)$ clause encoding was previously derived by Zhang (see [44, Lemma 2]), for ensuring the orthogonality of a pair (A, B) of Latin squares of order n . Zhang's encoding for orthogonality uses a new predicate $\Phi(i, j, k)$ introduced via a clever trick and Zhang mentions that "*It is a challenge to develop a method which can automatically generate the predicates like Φ ...*" [45]. Zhang does not view Φ as a square, but viewing $\Phi(i, j, k)$ as asserting that $\Phi[i, j] = k$, Zhang uses constraints saying that Φ 's columns have distinct symbols and that the entries of A and B determine Φ 's entries. Following our notation, Zhang uses constraints of the form

$$(A_{i,j,k} \wedge B_{i,j,\ell}) \rightarrow \Phi(i, k, \ell), \quad \text{for all } 0 \leq i, j, k, \ell < n.$$

In light of the above and Proposition 6, this means that not only is Φ itself a Latin square, it can be naturally viewed as a transversal representation of one of the original Latin squares and conveniently expressed via a composition square.¹ Viewing Φ as a composition square, one can derive additional constraints on Φ using this extra structure (e.g., the entries of A and Φ determine the entries of B). As previously mentioned, such constraints are technically redundant, but tended to help the efficiency of the solver in our experiments.

4.3 Colour Constraints

We now describe how we encode that the square P is one of Myrvold's eight types described in Table 1; an identical encoding is used for Q . In order to do this, we need to be able to specify the *colour* of each cell in the square P to be either white, light, or dark. Let \mathbf{w} and \mathbf{d} represent fixed symbols that are not in our symbol set $\{0, \dots, n-1\}$.

¹The constraints used by Zhang causes the *columns* of Φ to represent transversals of B and for Φ to be the composition square BA^{-1} where the composition and inverse are defined *row-wise* instead of column-wise like in the rest of this paper.

We let the Boolean variable $P_{i,j,w}$ represent that the (i, j) th entry of P is white, and let the Boolean variable $P_{i,j,d}$ represent that the (i, j) th entry of P is dark. Otherwise, if both $P_{i,j,w}$ and $P_{i,j,d}$ are false, then the (i, j) th entry of P will be light. Note that dark variables are only necessary in the first six columns, since no dark entries appear in the last four columns (see Figure 3). Additionally, the position of the dark cells in the first six columns completely determines the position of the white cells in the first six columns—the whites containing the symbols $\{4, \dots, 9\}$ not darkly coloured—making the variables $P_{i,j,w}$ only necessary for $j \geq 6$. Altogether, we introduce n^2 new variables encoding the colours of P .

To ensure the symbols $\{0, \dots, 3\}$ are coloured white, we use the clauses

$$P_{i,j,r} \rightarrow P_{i,j,w} \quad \text{for all } 0 \leq i < n, 6 \leq j < n, \text{ and } 0 \leq r < 4,$$

and conversely to ensure that only symbols $\{0, \dots, 3\}$ are coloured white we use $P_{i,j,w} \rightarrow \bigvee_{0 \leq r < 4} P_{i,j,r}$ for all $0 \leq i < n$ and $6 \leq j < n$. Similarly, to ensure that only symbols $\{4, \dots, 9\}$ are coloured dark, we use the clauses

$$P_{i,j,d} \rightarrow \bigvee_{4 \leq r < n} P_{i,j,r} \quad \text{for all } 0 \leq i < n \text{ and } 0 \leq j < 6.$$

Recall that a transversal is said to be of type p_k when it has k whites in its last four entries. By Lemma 3, transversals of type p_k will also have $2k - 2$ dark entries in its first six entries. Thus, in order to specify that row i in P is of type p_k , we use the constraints

$$\sum_{0 \leq j < 6} P_{i,j,d} = 2k - 2 \quad \text{and} \quad \sum_{6 \leq j < n} P_{i,j,w} = k.$$

Here, like in Section 4.1, we think of Boolean variables as taking $\{0, 1\}$ values and encode the cardinality constraints with the encoding of Bailleux and Boufkhad [4]. We also know that each of the first six columns of P contain exactly two dark entries, so we use the cardinality constraints

$$\sum_{0 \leq i < n} P_{i,j,d} = 2 \quad \text{for all } 0 \leq j < 6.$$

Similarly, we also use n^2 Boolean variables $Q_{i,j,w}$ and $Q_{i,j,d}$ to represent the colours of the square Q and add similar constraints to those above (using the $Q_{i,j,w}$ and $Q_{i,j,d}$ variables in place of the $P_{i,j,w}$ and $P_{i,j,d}$ variables). We now have specified a coloured TRP (P, Q) with each of P and Q conforming to any of Myrvold's types R, S, \dots , X selected in advance. However, because P and Q are both transversal representations of the same coloured square L , it is important that their colours be *consistent* between themselves. In particular, the two entries coloured dark in each of the first six columns of P must match the two entries coloured dark in each of the first six columns of Q . (The white colours always match as they correspond exactly to the symbols $\{0, 1, 2, 3\}$, so if the dark colours match then so must the light colours.)

Suppose the (i, j) th entry of P has symbol k and is coloured dark. Then, in order for the colouring to be consistent, the entry of Q in the j th column having symbol k must also be coloured dark. The symbol k must exist in the j th column of Q because Q is a

Latin square, so say this happens in row i' . Then to express the consistency of the colours in P and Q we use the constraints

$$(P_{i,j,k} \wedge P_{i,j,d} \wedge Q_{i',j,k}) \rightarrow Q_{i',j,d} \quad \text{for all } 0 \leq i, i' < n, 0 \leq j < 6, \text{ and } 4 \leq k < n.$$

Although not strictly necessary, we also add constraints deriving the colour of cell (i, j) in P from the colour of cell (i', j) in Q , giving the constraints

$$(P_{i,j,k} \wedge Q_{i',j,d} \wedge Q_{i',j,k}) \rightarrow P_{i,j,d} \quad \text{for all } 0 \leq i, i' < n, 0 \leq j < 6, \text{ and } 4 \leq k < n.$$

4.4 Consistency with the 4×4 Subsquare Ω

Recall Myrvold's seven transversal representation types of a Latin square L are under the assumption that L has a 4×4 Latin subsquare Ω . As described in Section 2.2, we assume that the subsquare Ω contains the symbols $\{0, 1, 2, 3\}$ and appears in the lower-right of L . There are two possibilities for Ω up to isotopism, where two Latin squares are *isotopic* if one can be transformed into the other by row, column, or symbol permutations [30]. The two possibilities for Ω up to isotopism are the Cayley tables of \mathbb{Z}_4 and $\mathbb{Z}_2 \times \mathbb{Z}_2$, and we assume that Ω is either

$$\Omega_1 := \begin{array}{|c|c|c|c|} \hline 0 & 1 & 2 & 3 \\ \hline 1 & 2 & 3 & 0 \\ \hline 2 & 3 & 0 & 1 \\ \hline 3 & 0 & 1 & 2 \\ \hline \end{array}, \quad \text{or} \quad \Omega_2 := \begin{array}{|c|c|c|c|} \hline 0 & 1 & 2 & 3 \\ \hline 1 & 0 & 3 & 2 \\ \hline 2 & 3 & 0 & 1 \\ \hline 3 & 2 & 1 & 0 \\ \hline \end{array}.$$

Since we are searching for Latin squares P and Q that are both transversal representations of L , this restricts the possible locations for the white entries in the last four columns of P and Q . For example, if either Ω_1 or Ω_2 is the lower-right subsquare of L , then since P is a transversal representation of L , it cannot be the case that $P[i, 6] = 0$ and $P[i, 7] = 1$, regardless of the row i chosen. This is because the 0 in column 6 of L and the 1 in column 7 of L appear in the same row and therefore cannot appear in the same transversal.

Noting that the first row of Ω_1 and Ω_2 are both $[0, 1, 2, 3]$, we add the clauses

$$P_{i,j,j-6} \rightarrow \neg P_{i,j',j'-6} \quad \text{for all } 0 \leq i < n \text{ and } 6 \leq j < j' < n,$$

and use similar clauses for Q . Generalizing this, let ω_1 be a Boolean variable that is true when Ω_1 is to be used in L , and let ω_2 be a Boolean variable that is to be true when Ω_2 is to be used in L . We add the clauses

$$\begin{aligned} (\omega_1 \wedge P_{i,j,\Omega_1[i',j-6]}) &\rightarrow \neg P_{i,j',\Omega_1[i',j'-6]} \\ (\omega_2 \wedge P_{i,j,\Omega_2[i',j-6]}) &\rightarrow \neg P_{i,j',\Omega_2[i',j'-6]} \end{aligned}$$

for all $0 \leq i < n$, $i' \in \{1, 2, 3\}$, and $6 \leq j < j' < n$, and use similar clauses for Q . Specifying either Ω_1 or Ω_2 is to be used in L is done with the clause $\omega_1 \vee \omega_2$. If a particular subsquare Ω_1 or Ω_2 is desired, it can be enforced with either the unit clause ω_1 or the unit clause ω_2 .

4.5 Symmetry Breaking

The ordering of rows of a transversal representation square is arbitrary in the sense that if P is a transversal representation of Q , then the rows of P can be freely permuted while preserving the fact that it is a transversal representation of Q . Similarly, the rows of Q may also be permuted. Columns may not be permuted independently, but if (P, Q) is a TRP and the same permutation of columns is applied to both P and Q simultaneously, then the resulting new pair will also be a TRP. Similarly, the same permutation of symbols applied to both squares in a TRP maintains the property of the pair being a TRP. Since we have already supposed that the symbols in the lower-right 4×4 submatrix of L are in $\{0, 1, 2, 3\}$, in order to not disturb this structure all permutations on symbols will operate on $\{0, 1, 2, 3\}$ and $\{4, \dots, 9\}$ independently. Similarly, we only use permutations of the first six and last four columns when transforming a TRP into the normal form defined below.

By a *coloured* TRP we mean one whose cells have been assigned the colours {white, light, dark} corresponding to Myrvold's types from Section 2.2. If (P, L) is a coloured TRP where L has been coloured corresponding to Figure 3, then permutations of the rows of P will also permute the colour positions in P . Similarly, permutations of the columns of P and L simultaneously will permute the colour positions in (P, L) , whereas permuting the symbols $\{4, \dots, 9\}$ or $\{0, 1, 2, 3\}$ in (P, L) will not permute the colour positions in (P, L) .

Row permutations of P , row permutations of Q , column permutations of the first six or last four columns of (P, Q) , and symbol permutations of the first four or last six symbols of (P, Q) generate a group G of size $10!^2 \cdot 6!^2 \cdot 4!^2 \approx 4 \cdot 10^{21}$. We call two coloured TRPs *equivalent* if one can be transformed to the other using operations in G . The large size of G means that our search space contains a large number of TRPs that are equivalent. This artificially increases the size of the search space, and we would like to constrain the search space in order to limit the search to as few representatives from each equivalence class as possible—this is known as *symmetry breaking*. We are able to remove many representatives from the search by only searching for TRPs in the normal form defined below.

Definition 11. A coloured TRP (P, Q) is in *normal form* if the rows of each square are sorted by transversal type (i.e., if row i has type p_k and row $i' \geq i$ has type $p_{k'}$ then $k \leq k'$), all the rows of the same transversal type are sorted in increasing lexicographic order, and the first row of P is one of

$$\begin{array}{|c|c|c|c|c|c|c|c|c|c|} \hline 0 & 1 & 2 & 4 & 5 & 6 & 3 & 7 & 8 & 9 \\ \hline \end{array},$$

$$\begin{array}{|c|c|c|c|c|c|c|c|c|c|} \hline 0 & 1 & 3 & 4 & 5 & 6 & 2 & 7 & 8 & 9 \\ \hline \end{array}, \text{ or}$$

$$\begin{array}{|c|c|c|c|c|c|c|c|c|c|} \hline 0 & 2 & 3 & 4 & 5 & 6 & 1 & 7 & 8 & 9 \\ \hline \end{array}.$$

In Theorem 13, we demonstrate that every equivalence class of TRPs of the kind we are looking for contains at least one TRP in normal form. First, we prove a simple lemma used in the proof of Theorem 13.

Lemma 12. *Suppose Ω is a Latin square of order 4. Then Ω is isotopic to either Ω_1 or Ω_2 . In either case, Ω can be transformed into Ω_1 or Ω_2 without permuting column 0 or symbol 0.*

Proof. There are exactly two Latin squares of order 4 up to isotopy (Ω_1 and Ω_2) and a total of four *reduced* Latin squares of order 4 (i.e., with entries in the first row and column appearing in sorted order) [30]. The two additional reduced Latin squares of order four are both isotopic to Ω_1 and are given by

$$\Omega_3 := \begin{array}{|c|c|c|c|} \hline 0 & 1 & 2 & 3 \\ \hline 1 & 0 & 3 & 2 \\ \hline 2 & 3 & 1 & 0 \\ \hline 3 & 2 & 0 & 1 \\ \hline \end{array}, \quad \text{and} \quad \Omega_4 := \begin{array}{|c|c|c|c|} \hline 0 & 1 & 2 & 3 \\ \hline 1 & 3 & 0 & 2 \\ \hline 2 & 0 & 3 & 1 \\ \hline 3 & 2 & 1 & 0 \\ \hline \end{array}.$$

If Ω is isotopic to Ω_2 , it can be transformed into reduced form by using row permutations to put 0 in the upper-left corner, then symbol permutations of $\{1, 2, 3\}$ to make the first row $[0, 1, 2, 3]$, and then permuting the last three rows to transform the first column into $[0, 1, 2, 3]$. Since there is only one reduced Latin square of order 4 isotopic to Ω_2 , this must transform Ω to Ω_2 .

Otherwise, if Ω is isotopic to Ω_1 , use row and symbol permutations as above to transform it into reduced form, thereby transforming it into Ω_1 , Ω_3 , or Ω_4 . To transform Ω_3 into Ω_1 , swap columns 1 and 2, rows 1 and 2, and symbols 1 and 2. To transform Ω_4 into Ω_1 , swap columns 2 and 3, rows 2 and 3, and symbols 2 and 3. \square

Theorem 13. *Suppose (P, Q) , (P, L) , and (Q, L) are coloured TRPs where L contains a 4×4 Latin subsquare and is coloured according to Figure 3. Then (P, Q) is equivalent to a coloured TRP in normal form and the lower-right 4×4 Latin subsquare in L can be taken to be either Ω_1 or Ω_2 .*

Proof. Let (P, Q) be a coloured TRP satisfying the preconditions of the theorem that we want to transform to a pair in normal form. First, permute the rows of P to put together rows of the same transversal type p_i (for $i \in \{1, 2, 3, 4\}$) such that all rows of type p_k come before all rows of type $p_{k'}$ when $k < k'$. Next, permute the rows of Q in a similar fashion so the rows of Q are also sorted by transversal type.

Since all square types contain transversals of type p_1 , and none contain transversals of type p_0 , the above sorting process implies the first row of P is of type p_1 . Now use column permutations of the first six columns (and the last four columns) to position the colours of the first row of P in the following order: 3 white, 3 light, 1 white, 3 light. Following this, if the symbol of P in the upper-left corner is not symbol 0, use a symbol permutation to make it 0.

By Lemma 12, we can now use symbol permutations of $\{1, 2, 3\}$, simultaneous permutations of the last three columns of (P, Q, L) , and row permutations in L to ensure that the lower-right 4×4 subsquare of L is either Ω_1 or Ω_2 . Row permutations of L , symbol permutations of $\{1, 2, 3\}$, and column permutations of the last three columns will not disturb the colouring of the first row of P or the fact $P[0, 0] = 0$.

Afterward, apply permutations of the symbols $\{4, \dots, 9\}$ to P , Q , and L simultaneously to put the light entries of the first row of P into normal form. If $P[0, 1]$ and $P[0, 2]$ are not in ascending order, use a column permutation to sort them. As a result, the first three entries of the first row of P are now $[0, 1, 2]$, $[0, 1, 3]$, or $[0, 2, 3]$, so the first row of P is in one of the three cases given in Definition 11.

Finally, within each subset of rows of the same transversal type of P (and independently Q), permute the rows so they appear in increasing lexicographic order. The first row of P already begins with the symbol 0, so it will not be moved. \square

Thus, without loss of generality we can assume the TRP we are searching for is in normal form and so we add extra constraints into our encoding to enforce this. Fixing the lightly coloured entries in the first row of P and the $(0, 0)$ th symbol of P can be done by adding appropriate unit clauses (clauses of length 1), namely,

$$P_{0,0,0} \wedge \bigwedge_{3 \leq j \leq 5} P_{0,j,j+1} \wedge \bigwedge_{7 \leq j \leq 9} P_{0,j,j}.$$

The remaining entries in the first row of P are determined by the value of $P[0, 6]$, giving the constraints

$$P_{0,6,3} \rightarrow (P_{0,1,1} \wedge P_{0,2,2}), P_{0,6,2} \rightarrow (P_{0,1,1} \wedge P_{0,2,3}), \text{ and } P_{0,6,1} \rightarrow (P_{0,1,2} \wedge P_{0,2,3}).$$

Each constraint $x \rightarrow (y \wedge z)$ is broken into two clauses of length two ($x \rightarrow y$ and $x \rightarrow z$). Although not strictly necessary, clauses for other facts about the white entries in the first row of P , such as $P_{0,1,2} \rightarrow P_{0,2,3}$, are also included.

Enforcing the fact that rows are sorted by transversal type is done with the cardinality constraints discussed in Section 4.3, as these constraints allow us to fix which rows are of which types. For example, suppose that P is of type R, meaning that P consists of eight transversals of type p_1 and two transversals of type p_4 . Then we would enforce the first eight rows of P to be of type p_1 with $P_{i,6,w} + \dots + P_{i,9,w} = 1$ and $P_{i,0,d} + \dots + P_{i,5,d} = 0$ for $0 \leq i < 8$, and the last two rows of P to be of type p_4 with $P_{i,6,w} + \dots + P_{i,9,w} = 4$ and $P_{i,0,d} + \dots + P_{i,5,d} = 6$ for $i = 8$ and 9 .

Finally, we enforce that rows with the same transversal type in P are sorted in lexicographic order by ensuring their initial entries are increasing. For example, suppose rows i and $i + 1$ of P have the same transversal type. Then we add the constraint $P_{i,0,k} \rightarrow \neg P_{i+1,0,l}$ for all $0 \leq l < k < n$, which says that the initial entry of row $i + 1$ of P cannot be smaller than the initial entry of row i . We add the same constraints for Q as well.

4.6 Postprocessing

As we will describe in Section 5, the encoding presented thus far successfully found many TRPs (P, Q) corresponding to Myrvold's eight unsolved cases. We performed some postprocessing on these pairs to check if they were extendable to a triple of mutual transversal representations and also to check the pairs for equivalence.

First, we used a SAT solver to check all pairs (P, Q) for extendability to a triple. This was done by creating new SAT instances for each pair encoding both squares P and Q , along with a new Latin square L , and then asserting that (L, P) is a TRP and (L, Q) is a TRP by using the encoding described in Section 4.2 twice. The entries of P and Q were specified using unit clauses; i.e., if $P[i, j] = k$ then the clause $P_{i,j,k}$ was added to the SAT instance. Because of the presence of so many unit clauses these instances were highly constrained and in all cases were shown by the SAT solver to be unsatisfiable within 0.1 seconds. Thus, no pairs we found were extendable to a triple. However, this does not eliminate the possibility that there might exist a triple (P, Q, L) corresponding to some of Myrvold's cases, because we did not exhaustively enumerate all (P, Q) s for any of Myrvold's unsolved types.

Finally, we checked all the TRPs (P, Q) that we found to see if any were equivalent to each other. This was done by converting the TRP into its orthogonal pair representation $(P^{-1}Q, Q)$, reducing the orthogonal pair to a graph using the reduction given by Egan and Wanless [16], and finally checking the graphs for equivalence using the graph isomorphism tool NAUTY [31].

Precisely, the reduction from a $(k - 2)$ MOLS(n) to a graph is described using what is known as an orthogonal array. An orthogonal array for a $(k - 2)$ MOLS(n) is a matrix O of size $n^2 \times k$, with entries in $\{0, \dots, n - 1\}$, with every possible pair of symbols appearing exactly once in any two columns of O . Define an undirected graph G_O corresponding to O . The vertices of G_O are of three types:

- k type 1 vertices that correspond to the columns of O ,
- kn type 2 vertices that correspond to the symbols in each of the columns of O , and
- n^2 type 3 vertices that correspond to the rows of O .

Each type 1 vertex is joined to the n type 2 vertices that correspond to the symbols in its column. Each type 3 vertex is connected to the k type 2 vertices that correspond to the symbols in its row. Vertices are coloured according to their type so that isomorphisms are not allowed to change the type of a vertex.

After forming the graphs corresponding to all TRPs (P, Q) we found, NAUTY determined that no two graphs were isomorphic. Thus, we have confirmation that the SAT solver is indeed exploring different parts of the search space and that multiple inequivalent TRPs exist corresponding to Myrvold's unsolved cases. However, we did not attempt to perform an exhaustive search for TRPs in any of Myrvold's unsolved cases. Given the enormity of the search space, and the fact that no solutions were repeated even after several hundred solutions had already been found, we suspect that an exhaustive search would require a huge amount of additional computational resources or at least some more restrictive properties that could be applied to Myrvold's unsolved cases.

5 Results

We now discuss the results of our computational investigation into Myrvold's results. The computations were performed using the SAT solver Kissat 4.0.4 [5] run on AMD EPYC Zen 5 processors running at 2.7 GHz and equipped with 1 GiB of memory.

Recall Myrvold showed [33, Thm 4.4], if P and Q are both transversal representations of a Latin square of order ten containing a subsquare of order four, then up to ordering there are twenty-eight possible cases for P and Q and twenty of these cases can be ruled out. The eight possible cases Myrvold left remaining are (S, X), (U, U), (U, W), (U, X), (V, X), (W, W), (W, X), and (X, X).

We used our SAT encoding to generate twenty-eight SAT instances, one for each of Myrvold's cases. The twenty cases ruled out by Myrvold were each found to be unsatisfiable in under 0.2 seconds. The eight cases left open by Myrvold were all considerably harder to solve, but each was found to be satisfiable, explaining why Myrvold was unable to eliminate these eight cases from consideration. Kissat stops solving as soon as it finds a satisfying assignment of the provided instance, and we use the satisfying assignment reported by Kissat to form a coloured TRP in each of the eight cases (see the Appendix for explicit examples of TRPs in each case).

Because the satisfiable cases were significantly more difficult than the unsatisfiable cases, we found it useful to exploit parallelization when solving the satisfiable instances. We started 49 independent Kissat processes for each satisfiable case and each process was run on one processor core for up to one week. Each process was provided with a different random seed, so no two copies of Kissat would make the same choices during the solving process. Each process was terminated if Kissat did not find a solution within a week. Results from these searches are available in Table 2, and a scatterplot of the running times is given in Figure 5. There is a significant amount of variance in the running times, but in general the case (U, U) was the easiest to solve and the case (X, X) was the hardest to solve.

We summarize some statistical information about the TRPs we found in Table 3. In particular, for each pair type we provide the number of TRPs found that are compatible with the 4×4 subsquares Ω_1 and Ω_2 in L . In case (V, X), the solver was able to show there are no TRPs consistent with the choice Ω_1 in under 0.2 seconds. This can be explained by the fact that the square Ω_1 has no transversals—it follows that Ω_1 is inconsistent with square type V, because the white entries in a row of type p_4 must represent a transversal in Ω .

Usually the TRPs we found were consistent with only one of Ω_1 or Ω_2 , but two TRPs were consistent with both choices of Ω simultaneously. Both were of type (X, X) and one of these TRPs is provided as the example (X, X) pair in the appendix. Also listed in Table 3 are the minimum and maximum number of transversals and mates in each of the squares in the TRPs we found. It also reports on the number of *common* transversals in the TRPs (i.e., transversals of both squares in the TRP whose row representation is the same in both). Most TRPs had no common transversals, and none had more than two common transversals. This is an indication that the TRPs we found are not very close

to extending to a triple of mutual TRPs, since for (P, Q) to extend to a triple of mutual TRPs, P and Q must have at least n common transversals.

pair type	mean	median	min	max
(U, U)	31102.1	19619.4	748.6	98009.2
(S, X)	58780.5	38453.2	2282.4	175005.1
(U, W)	75043.9	56171.4	2659.8	399428.7
(W, W)	139198.5	97661.6	1662.1	timeout
(V, X)	147169.2	114191.0	2567.7	timeout
(U, X)	140560.4	117378.6	327.8	timeout
(W, X)	222515.7	176970.4	527.3	timeout
(X, X)	429809.6	580524.5	6747.1	timeout

Table 2: A summary of the running times (in seconds) of the instances for each of the eight pair types with solutions. Each pair type had 49 independently-solved SAT instances and were run with a one week timeout. The timeouts were included in the computation of each statistic and counted as running for a full week.

pair type	# solved	$\#\Omega_1$	$\#\Omega_2$	transversals	mates	common trans.
(U, U)	49	9	40	776–900	1–6	0–1
(S, X)	49	27	22	768–948	1–7	0–1
(U, W)	49	19	30	744–912	1–5	0–0
(W, W)	48	25	23	764–900	1–5	0–1
(V, X)	48	0	48	756–940	1–8	0–2
(U, X)	48	20	28	724–924	1–6	0–1
(W, X)	46	23	23	772–924	1–9	0–2
(X, X)	25	13	14	772–912	1–5	0–1

Table 3: A summary of the TRPs we found using 49 independently-solved SAT instances for each pair type. The table includes the number of solved SAT instances, the number of TRPs compatible with the 4×4 subsquares Ω_1 and Ω_2 , and the minimum and maximum number of transversals, mates, and common transversals appearing in the TRPs.

6 Conclusion

In this paper we use a satisfiability (SAT) solver to investigate Myrvold’s nonexistence results [33] on orthogonal triples of Latin squares of order ten. The SAT solver almost instantaneously rules out the cases that Myrvold ruled out, and more significantly, the SAT solver provides explicit examples of Latin square pairs in each of the cases that Myrvold was unable to rule out—providing an explanation for why Myrvold was unable to rule out

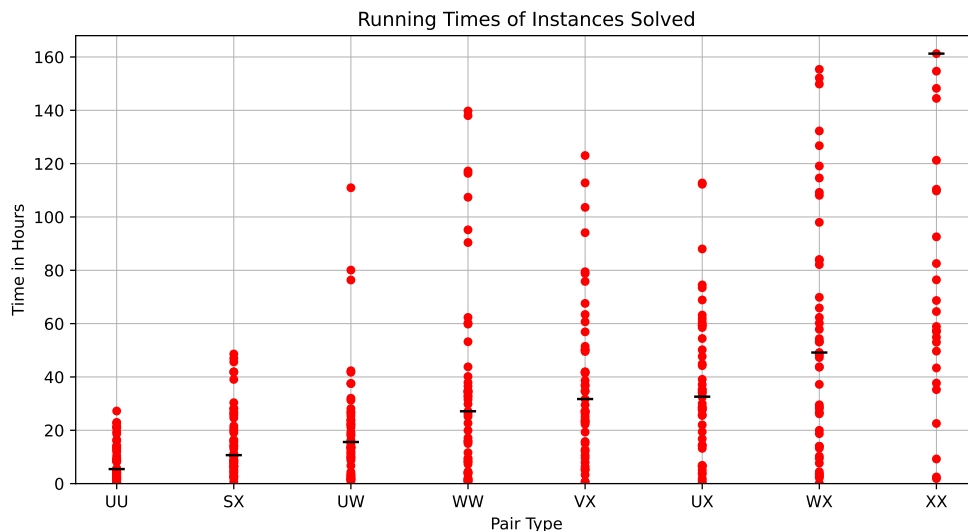


Figure 5: A scatterplot of the solver’s running time for each pair type. The median running time is shown as a solid black line. Timeouts are not plotted but are used in determining the median.

these cases and determining a negative resolution to the following question left open by Myrvold:

Possibly, with a bit more ingenuity, the remaining cases can be eliminated.

We show that pairs exist in the remaining cases, and so eliminating the remaining cases with “a bit more ingenuity” is probably not achievable—at the very least, any argument required to eliminate the remaining cases would need to be more sophisticated in having to rely on the existence of the third square, L . We were also able to show that requiring compatibility with the 4×4 Latin subsquare in L is not by itself sufficient to rule out any of the remaining cases. It would be interesting to know if some of the remaining cases could be ruled out by considering additional structure in L , but we leave this as future work.

In order to derive a concise and effective SAT encoding for our search we make use of a duality between orthogonal Latin squares and transversal representation pairs. Although such a duality has long been used in searches for Latin squares, we also give an explicit formulation of how this duality arises via a composition operation on Latin squares. We found this viewpoint useful when deriving our encoding and surprisingly we were not able to find it expressed in prior literature.

Acknowledgements

We thank the reviewers for their detailed feedback which improved the paper. In particular, a reviewer pointed out the possibility of adding constraints enforcing that the transversal representation pair is consistent with the 4×4 Latin subsquare in L . We also thank Tanbir Ahmed for his help during the editing process.

References

- [1] Abel, R.J.R., Colbourn, C.J., Dinitz, J.H.: Mutually orthogonal Latin squares (MOLS). In: *Handbook of Combinatorial Designs*, pp. 186–218. Chapman and Hall/CRC (2006). [doi:10.1201/9781420010541](https://doi.org/10.1201/9781420010541)
- [2] Appa, G., Magos, D., Mourtos, I.: Searching for mutually orthogonal Latin squares via integer and constraint programming. *European Journal of Operational Research* **173**(2), 519–530 (2006). [doi:10.1016/j.ejor.2005.01.048](https://doi.org/10.1016/j.ejor.2005.01.048)
- [3] Appa, G., Mourtos, I., Magos, D.: Integrating constraint and integer programming for the orthogonal Latin squares problem. In: Van Hentenryck, P. (ed.) *Principles and Practice of Constraint Programming - CP 2002*. Lecture Notes in Computer Science, vol. 2470, pp. 17–32. Springer Berlin Heidelberg (2002). [doi:10.1007/3-540-46135-3_2](https://doi.org/10.1007/3-540-46135-3_2)
- [4] Bailleux, O., Boufkhad, Y.: Efficient CNF encoding of Boolean cardinality constraints. In: Rossi, F. (ed.) *Principles and Practice of Constraint Programming – CP 2003*. Lecture Notes in Computer Science, vol. 2833, pp. 108–122. Springer Berlin Heidelberg (2003). [doi:10.1007/978-3-540-45193-8_8](https://doi.org/10.1007/978-3-540-45193-8_8)
- [5] Biere, A., Fleury, M.: Gimsatul, IsaSAT and Kissat entering the SAT competition 2022. *Proc. of SAT Competition: Solver and Benchmark Descriptions*, 2022 pp. 10–11 (2022), <https://helda.helsinki.fi/handle/10138/359079>
- [6] Bose, R.C., Shrikhande, S.S.: On the falsity of Euler’s conjecture about the non-existence of two orthogonal Latin squares of order $4t + 2$. *Proceedings of the National Academy of Sciences* **45**(5), 734–737 (1959). [doi:10.1073/pnas.45.5.734](https://doi.org/10.1073/pnas.45.5.734)
- [7] Bose, R.C., Shrikhande, S.S., Parker, E.T.: Further results on the construction of mutually orthogonal Latin squares and the falsity of Euler’s conjecture. *Canadian Journal of Mathematics* **12**, 189–203 (1960). [doi:10.4153/cjm-1960-016-5](https://doi.org/10.4153/cjm-1960-016-5)
- [8] Bose, R.C.: On the application of the properties of Galois fields to the problem of construction of Hyper-Græco-Latin squares. *Sankhyā: The Indian Journal of Statistics* **3**(4), 323–338 (1938), <http://www.jstor.org/stable/40383859>
- [9] Bright, C., Cheung, K., Stevens, B., Roy, D., Kotsireas, I., Ganesh, V.: A nonexistence certificate for projective planes of order ten with weight 15 codewords. *Applicable Algebra in Engineering, Communication and Computing* **31**(3–4), 195–213 (2020). [doi:10.1007/s00200-020-00426-y](https://doi.org/10.1007/s00200-020-00426-y)
- [10] Bright, C., Cheung, K.K.H., Stevens, B., Kotsireas, I., Ganesh, V.: A SAT-based resolution of Lam’s problem. *Proceedings of the AAAI Conference on Artificial Intelligence* **35**(5), 3669–3676 (2021). [doi:10.1609/aaai.v35i5.16483](https://doi.org/10.1609/aaai.v35i5.16483)
- [11] Bright, C., Gerhard, J., Kotsireas, I., Ganesh, V.: Effective problem solving using SAT solvers. In: Gerhard, J., Kotsireas, I. (eds.) *Maple in Mathematics Education and Research*. Communications in Computer and Information Science, vol. 1125, pp. 205–219. Springer International Publishing (2020). [doi:10.1007/978-3-030-41258-6_15](https://doi.org/10.1007/978-3-030-41258-6_15)

- [12] Bright, C., Keita, A., Stevens, B.: Orthogonal Latin squares of order 10 with two relations: A SAT investigation. *Discrete Mathematics, Algorithms and Applications* (2025). doi:[10.1142/s1793830925501563](https://doi.org/10.1142/s1793830925501563), to appear
- [13] Bruck, R.H.: Finite nets. II. Uniqueness and imbedding. *Pacific Journal of Mathematics* **13**(2), 421–457 (1963). doi:[10.2140/pjm.1963.13.421](https://doi.org/10.2140/pjm.1963.13.421)
- [14] Delisle, E.: The Search for a Triple of Mutually Orthogonal Latin Squares of Order Ten: Looking Through Pairs of Dimension Thirty-Five and Less. Master’s thesis, University of Victoria (2010), <http://hdl.handle.net/1828/2964>
- [15] Dukes, P., Howard, L.: Group divisible designs in MOLS of order ten. *Designs, Codes and Cryptography* **71**(2), 283–291 (2012). doi:[10.1007/s10623-012-9729-8](https://doi.org/10.1007/s10623-012-9729-8)
- [16] Egan, J., Wanless, I.M.: Enumeration of MOLS of small order. *Mathematics of Computation* **85**(298), 799–824 (2015). doi:[10.1090/mcom/3010](https://doi.org/10.1090/mcom/3010)
- [17] Gill, M.J., Wanless, I.M.: Pairs of MOLS of order ten satisfying non-trivial relations. *Designs, Codes and Cryptography* **91**(4), 1293–1313 (2023). doi:[10.1007/s10623-022-01149-6](https://doi.org/10.1007/s10623-022-01149-6)
- [18] Heule, M.J.H., Kullmann, O., Marek, V.W.: Solving and verifying the boolean Pythagorean triples problem via cube-and-conquer. In: Creignou, N., Le Berre, D. (eds.) *Theory and Applications of Satisfiability Testing – SAT 2016*. *Lecture Notes in Computer Science*, vol. 9710, pp. 228–245. Springer International Publishing (2016). doi:[10.1007/978-3-319-40970-2_15](https://doi.org/10.1007/978-3-319-40970-2_15)
- [19] Huang, P., Liu, M., Ge, C., Ma, F., Zhang, J.: Investigating the existence of orthogonal golf designs via satisfiability testing. In: *Proceedings of the 2019 International Symposium on Symbolic and Algebraic Computation*. pp. 203–210. ISSAC ’19, ACM (2019). doi:[10.1145/3326229.3326232](https://doi.org/10.1145/3326229.3326232)
- [20] Jin, J., Lv, Y., Ge, C., Ma, F., Zhang, J.: Investigating the existence of Costas Latin squares via satisfiability testing. In: Li, C.M., Manyà, F. (eds.) *Theory and Applications of Satisfiability Testing – SAT 2021*. *Lecture Notes in Computer Science*, vol. 12831, pp. 270–279. Springer International Publishing (2021). doi:[10.1007/978-3-030-80223-3_19](https://doi.org/10.1007/978-3-030-80223-3_19)
- [21] Keedwell, A.D., Dénes, J.: *Latin Squares and their Applications*, Second Edition. Elsevier (2015). doi:[10.1016/c2014-0-03412-0](https://doi.org/10.1016/c2014-0-03412-0)
- [22] Lam, C.W.H.: The search for a finite projective plane of order 10. *The American Mathematical Monthly* **98**(4), 305–318 (1991). doi:[10.1080/00029890.1991.12000759](https://doi.org/10.1080/00029890.1991.12000759)
- [23] Lam, C.W.H., Thiel, L., Swiercz, S.: The non-existence of finite projective planes of order 10. *Canadian Journal of Mathematics* **41**(6), 1117–1123 (1989). doi:[10.4153/cjm-1989-049-4](https://doi.org/10.4153/cjm-1989-049-4)
- [24] Laywine, C.F., Mullen, G.L.: *Discrete Mathematics Using Latin Squares*, vol. 49. John Wiley & Sons (1998)
- [25] Lu, R., Liu, S., Zhang, J.: Searching for doubly self-orthogonal Latin squares. In: Lee, J. (ed.) *Principles and Practice of Constraint Programming – CP 2011*. *Lecture*

- Notes in Computer Science, vol. 6876, pp. 538–545. Springer Berlin Heidelberg (2011). [doi:10.1007/978-3-642-23786-7_41](https://doi.org/10.1007/978-3-642-23786-7_41)
- [26] Ma, F., Zhang, J.: Finding orthogonal Latin squares using finite model searching tools. *Science China Information Sciences* **56**(3), 1–9 (2011). [doi:10.1007/s11432-011-4343-3](https://doi.org/10.1007/s11432-011-4343-3)
 - [27] Mann, H.B.: The construction of orthogonal Latin squares. *The Annals of Mathematical Statistics* **13**(4), 418–423 (1942). [doi:10.1214/aoms/1177731539](https://doi.org/10.1214/aoms/1177731539)
 - [28] Mann, H.B.: On orthogonal Latin squares. *Bulletin of the American Mathematical Society* **50**(4), 249–257 (1944). [doi:10.1090/s0002-9904-1944-08127-5](https://doi.org/10.1090/s0002-9904-1944-08127-5)
 - [29] Marques-Silva, J., Lynce, I.: Towards robust CNF encodings of cardinality constraints. In: Bessière, C. (ed.) *Principles and Practice of Constraint Programming – CP 2007*. Lecture Notes in Computer Science, vol. 4741, pp. 483–497. Springer Berlin Heidelberg (2007). [doi:10.1007/978-3-540-74970-7_35](https://doi.org/10.1007/978-3-540-74970-7_35)
 - [30] McKay, B.D., Meynert, A., Myrvold, W.: Small Latin squares, quasigroups, and loops. *Journal of Combinatorial Designs* **15**(2), 98–119 (2006). [doi:10.1002/jcd.20105](https://doi.org/10.1002/jcd.20105)
 - [31] McKay, B.D., Piperno, A.: Practical graph isomorphism, II. *Journal of Symbolic Computation* **60**, 94–112 (2014). [doi:10.1016/j.jsc.2013.09.003](https://doi.org/10.1016/j.jsc.2013.09.003)
 - [32] Moore, E.H.: Tactical memoranda I-III. *American Journal of Mathematics* **18**(3), 264 (1896). [doi:10.2307/2369797](https://doi.org/10.2307/2369797)
 - [33] Myrvold, W.: Negative results for orthogonal triples of Latin squares of order 10. *Journal of Combinatorial Mathematics and Combinatorial Computing* **29**, 95–106 (1999), <https://combinatorialpress.com/article/jcmcc/Volume%20029/vol-029-paper%207.pdf>
 - [34] Norton, D.: Groups of orthogonal row-latin squares. *Pacific Journal of Mathematics* **2**(3), 335–341 (1952). [doi:10.2140/pjm.1952.2.335](https://doi.org/10.2140/pjm.1952.2.335)
 - [35] Parker, E.T.: Orthogonal Latin squares. *Proceedings of the National Academy of Sciences* **45**(6), 859–862 (1959). [doi:10.1073/pnas.45.6.859](https://doi.org/10.1073/pnas.45.6.859)
 - [36] Parker, E.: On orthogonal Latin squares. 1960 *Institute on Finite Groups* **6**, 43–36 (1962). [doi:10.1090/pspum/006/0132704](https://doi.org/10.1090/pspum/006/0132704)
 - [37] Roy, D.J.: Confirmation of the Non-existence of a Projective Plane of Order 10. Master’s thesis, Carleton University (2011). [doi:10.22215/etd/2011-09202](https://doi.org/10.22215/etd/2011-09202)
 - [38] Rubin, N., Bright, C., Cheung, K., Stevens, B.: Improving integer and constraint programming for Graeco–Latin squares. In: 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI). IEEE (2021). [doi:10.1109/ictai52525.2021.00096](https://doi.org/10.1109/ictai52525.2021.00096)
 - [39] Tarry, G.: Le problème des 36 officiers. *Association Française pour l’Avancement des Sciences: Compte Rendu de la 29^{me} session en Paris 1900* **2**, 170–203 (1901)
 - [40] Vardi, M.Y.: Boolean satisfiability: theory and engineering. *Communications of the ACM* **57**(3), 5 (2014). [doi:10.1145/2578043](https://doi.org/10.1145/2578043)

- [41] Wanless, I.: Transversals in Latin squares: A survey. In: Chapman, R. (ed.) *Surveys in Combinatorics 2011*. pp. 403–437. Cambridge University Press (2011). [doi:10.1017/cbo9781139004114.010](https://doi.org/10.1017/cbo9781139004114.010)
- [42] Zaikin, O., Kochemazov, S.: The search for systems of diagonal Latin squares using the SAT@home project. *International Journal of Open Information Technologies* **3**(11), 4–9 (2015), <http://injoit.org/index.php/j1/article/view/239>
- [43] Zaikin, O., Vatutin, E., Bright, C.: Enumerating extended self-orthogonal diagonal Latin squares of order up to 10. *Journal of Integer Sequences* **28**(7) (2025), <https://cs.uwaterloo.ca/journals/JIS/VOL28/Zaikin/zaikin4.html>, article 25.7.4
- [44] Zhang, H.: Specifying Latin square problems in propositional logic. In: Veroff, R. (ed.) *Automated reasoning and Its Applications: Essays in Honor of Larry Wos*. pp. 115–146. MIT Press, Cambridge, Massachusetts (1997), <https://dl.acm.org/doi/10.5555/271101.271124>
- [45] Zhang, H.: Combinatorial designs by SAT solvers. In: *Handbook of Satisfiability*. pp. 819–858. IOS Press (2021). [doi:10.3233/faia201005](https://doi.org/10.3233/faia201005)

Appendix

In this appendix we provide eight explicit pairs we found which prove the existence of TRPs for Myrvold's eight unresolved cases [33].

type S

0	1	3	4	5	6	2	7	8	9
1	6	9	2	3	4	0	8	5	7
3	8	2	1	7	5	6	4	9	0
4	2	7	0	1	8	3	9	6	5
5	0	8	9	2	3	7	1	4	6
7	3	0	5	9	1	4	6	2	8
8	5	1	6	0	2	9	3	7	4
2	4	5	7	6	9	8	0	3	1
6	9	4	3	8	7	1	5	0	2
9	7	6	8	4	0	5	2	1	3

type X

2	6	1	4	7	3	5	9	0	8
4	9	3	5	2	0	6	8	7	1
7	1	2	9	0	4	8	5	6	3
8	0	5	3	1	6	4	2	9	7
0	5	7	8	3	9	1	4	2	6
1	4	6	0	9	5	7	3	8	2
3	7	9	6	8	1	2	0	4	5
5	3	4	2	6	8	9	7	1	0
6	8	0	7	4	2	3	1	5	9
9	2	8	1	5	7	0	6	3	4

type U

0	1	2	4	5	6	3	7	8	9
1	7	8	0	4	2	5	9	3	6
2	9	7	1	8	3	0	4	6	5
3	5	0	7	2	4	6	8	9	1
4	2	6	3	0	9	7	1	5	8
9	6	1	2	3	7	8	5	0	4
5	8	3	6	9	1	4	0	2	7
8	3	4	9	1	5	2	6	7	0
6	0	9	5	7	8	1	2	4	3
7	4	5	8	6	0	9	3	1	2

type U

0	3	6	1	9	7	5	8	4	2
2	5	1	8	0	6	4	9	7	3
3	0	4	6	8	2	7	5	1	9
6	9	2	0	1	4	8	3	5	7
8	7	0	3	5	1	9	2	6	4
9	2	3	7	4	0	1	6	8	5
1	6	5	4	7	3	2	0	9	8
7	1	8	5	3	9	6	4	2	0
4	8	9	2	6	5	0	7	3	1
5	4	7	9	2	8	3	1	0	6

type U

0	2	3	4	5	6	1	7	8	9
1	4	6	0	9	2	3	8	5	7
2	3	5	9	0	8	7	4	6	1
5	6	7	1	3	0	8	2	9	4
6	0	1	2	8	5	4	9	7	3
7	5	0	3	4	1	9	6	2	8
8	7	2	5	1	9	6	3	4	0
9	1	4	8	2	7	0	5	3	6
3	8	9	7	6	4	2	0	1	5
4	9	8	6	7	3	5	1	0	2

type W

0	3	7	2	4	9	5	8	1	6
1	5	3	7	2	8	6	9	0	4
3	7	1	6	5	2	0	4	9	8
5	0	4	9	1	3	2	6	8	7
9	2	6	3	8	0	7	1	4	5
2	4	8	1	6	5	9	7	3	0
6	1	5	0	7	4	8	3	2	9
7	9	2	8	3	6	4	0	5	1
8	6	9	4	0	1	3	5	7	2
4	8	0	5	9	7	1	2	6	3

type U

0	1	2	4	5	6	3	7	8	9
1	3	8	2	6	4	7	5	9	0
2	8	9	5	3	1	4	0	7	6
5	9	0	1	2	7	8	6	3	4
7	0	4	3	1	9	2	8	6	5
9	5	3	7	0	2	6	1	4	8
3	4	7	6	8	0	5	9	2	1
6	7	1	0	4	8	9	3	5	2
4	6	5	8	9	3	0	2	1	7
8	2	6	9	7	5	1	4	0	3

type X

0	5	9	1	8	3	7	4	6	2
3	2	1	5	6	7	0	8	4	9
5	3	6	4	1	0	9	2	7	8
6	4	0	8	3	2	1	7	9	5
1	9	3	6	4	5	2	0	8	7
2	7	8	3	9	6	5	1	0	4
4	8	2	0	7	9	6	5	3	1
7	1	5	9	0	4	8	3	2	6
8	6	4	7	2	1	3	9	5	0
9	0	7	2	5	8	4	6	1	3

type V

0	2	3	4	5	6	1	7	8	9
1	3	7	5	0	9	4	8	2	6
3	0	2	9	6	8	5	1	4	7
4	6	1	8	2	0	7	9	5	3
5	1	0	3	9	7	2	4	6	8
7	5	4	0	1	3	8	6	9	2
2	7	8	6	3	4	9	0	1	5
6	8	9	2	7	1	3	5	0	4
9	4	5	1	8	2	6	3	7	0
8	9	6	7	4	5	0	2	3	1

type X

0	3	4	9	8	1	7	2	6	5
1	5	0	8	6	2	9	7	3	4
2	6	3	0	9	8	4	5	7	1
4	1	5	2	3	9	0	6	8	7
3	8	7	6	2	5	1	4	9	0
5	9	2	4	7	0	8	3	1	6
6	4	8	7	0	3	2	1	5	9
7	0	6	1	5	4	3	9	2	8
8	2	9	5	1	7	6	0	4	3
9	7	1	3	4	6	5	8	0	2

type W

0	2	3	4	5	6	1	7	8	9
1	3	7	6	2	9	4	0	5	8
2	7	1	5	0	4	8	9	3	6
3	5	4	0	8	1	7	6	9	2
8	0	2	3	9	7	5	1	6	4
4	1	6	8	3	5	9	2	0	7
5	8	9	7	1	2	6	3	4	0
6	9	0	1	7	8	3	4	2	5
9	4	5	2	6	3	0	8	7	1
7	6	8	9	4	0	2	5	1	3

type W

0	3	5	8	1	7	2	4	9	6
2	0	6	1	5	9	7	8	4	3
5	9	3	2	4	1	8	0	6	7
6	1	2	5	8	0	4	3	7	9
9	7	0	4	3	2	5	6	1	8
1	5	8	3	6	4	9	7	2	0
3	8	1	9	7	6	0	2	5	4
4	2	7	0	9	8	6	5	3	1
8	6	4	7	2	3	1	9	0	5
7	4	9	6	0	5	3	1	8	2

type W

0	1	3	4	5	6	2	7	8	9
1	5	0	2	7	9	8	6	4	3
2	6	9	0	8	1	3	4	7	5
3	2	5	7	0	4	1	8	9	6
7	9	1	8	2	3	5	0	6	4
4	0	7	5	6	2	9	3	1	8
5	4	2	3	9	8	6	1	0	7
6	7	8	9	1	0	4	5	3	2
8	3	6	1	4	5	7	9	2	0
9	8	4	6	3	7	0	2	5	1

type X

6	1	7	2	0	8	3	9	5	4
7	3	0	5	9	1	4	2	8	6
8	7	2	0	3	4	5	6	1	9
9	0	5	1	8	3	6	7	4	2
0	2	4	8	6	9	7	1	3	5
1	6	8	4	2	5	0	3	9	7
2	8	1	7	4	6	9	5	0	3
3	4	6	9	5	2	8	0	7	1
4	5	9	3	1	7	2	8	6	0
5	9	3	6	7	0	1	4	2	8

type X

0	1	3	4	5	6	2	7	8	9
1	7	6	3	2	8	5	9	4	0
2	9	1	5	6	3	7	8	0	4
6	2	9	8	3	1	4	0	7	5
3	6	0	7	8	4	9	2	5	1
4	5	2	9	7	0	3	1	6	8
5	8	4	0	1	7	6	3	9	2
7	0	5	2	4	9	8	6	1	3
8	4	7	1	9	2	0	5	3	6
9	3	8	6	0	5	1	4	2	7

type X

0	7	2	8	1	5	9	6	3	4
3	2	5	4	0	8	7	1	9	6
6	8	0	2	7	3	1	5	4	9
8	9	3	0	4	1	5	2	6	7
1	0	4	7	9	6	3	8	2	5
2	6	9	3	5	7	0	4	1	8
4	3	7	5	2	9	6	0	8	1
5	4	1	6	8	0	2	9	7	3
7	1	8	9	6	2	4	3	5	0
9	5	6	1	3	4	8	7	0	2